VariTex: Variational Neural Face Textures - Supplementary Document

Marcel C. Bühler[1] Abhimitra Meka[2] Gengyan Li[1,2] Thabo Beeler[2] Otmar Hilliges[1]
[1]ETH Zurich [2]Google
https://mcbuehler.github.io/VariTex

This is the supplementary document for *VariTex: Variational Neural Face Textures*. We list implementation details in Sec. 1, complement the experimental setup section from the main paper in Sec. 2, and discuss supplementary results and ablations in Sec. 3.

1. Implementation Details

1.1. Architectural Details

We implement VariTex in PyTorch Lightning [9] and use PyTorch3D [22]. Our pipeline contains four neural networks: An encoder that maps RGB images to a latent distribution; a face texture decoder that generates a neural face texture; an additive decoder that produces features for regions missing in the 3D face model; and a Feature2Image (rendering) network. Tables 4, 5, 6, and 7 list their architectural layers and the number of parameters in each layer. For brevity, we omit normalization layers, but include their parameter counts in the totals. All networks use ReLU activations and batch normalization [13].

1.2. Training Details

Optimization and Hyperparameters. We use Adam [16] to optimize the network parameters with an empirically determined learning rate of 0.001. We set the exponential decay rates for the first and second moments to 0.9 and 0.999. The training dataset is randomly divided into batches of size 7. The network trains for 44 epochs over the full training dataset, which takes approximately 96 hours on an NVIDIA Quadro RTX 6000/8000 GPU.

Objective Function. For the perceptual loss $\mathcal{L}_{VGG}$ (Eq. 2 in the main paper), the function $\phi_{VGG_j}(\cdot)$ extracts the j-th feature map from a VGG network [23]. The VGG network is pretrained on ImageNet [6], and the associated weights per feature map v_j are $\boldsymbol{v} = [\frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, 1]^\top$.

We use a two-scale patch discriminator D for photorealism [21]. The two scales are a) full scale and b) half the scale after average pooling across the spatial dimensions. Unlike the original architecture [21], we do not feed any segmentation masks and we use least squares as a loss function. The final loss is computed as the average across both discriminators.

Augmentation Parameters. We use the following affine transformations in our augmentation scheme (in Sec. 3.5 in the main paper): random in-plane rotation (up to $\pm 15°$), uniformly sampled translation (within 20% of the image height and width), random scaling between 100% and 120% of the image size, and random flips along the vertical axis ($p = 0.5$).

1.3. Texture Sampling

We sample the texture in the same way as previous works [24, 25], following the classical computer graphics pipeline. In particular, we employ the standard rasterization approach used by traditional renderers.

We project the per-vertex UV coordinates of a face model mesh [10] into the image space of the desired target camera. This produces a mapping from image pixels to UV coordinates $I_{x,y} \rightarrow UV_{u,v}$. We then bilinearly sample from the neural texture to the image space using this mapping. This yields a neural image representing the face: the face feature image.

2. Detailed Experimental Setup

In this section, we describe data preprocessing, provide an analysis of the head pose distribution in the training set, and give more details about the user study.

2.1. Preprocessing

Segmentation Network. We mask the images to the foreground. As there are no ground-truth foreground masks available for FFHQ [15], we predict them in a preprocessing step. We train a state-of-the-art semantic segmentation network [4, 5] on CelebAMask-HQ [19]. During training, we learn to predict all 19 semantic regions in CelebAMask-HQ, which we aggregate to a single foreground mask. As a backbone, we use DRN [27] and train it using a batch size of 12 and a cross-entropy loss for 150 epochs on images with resolution 512×512.

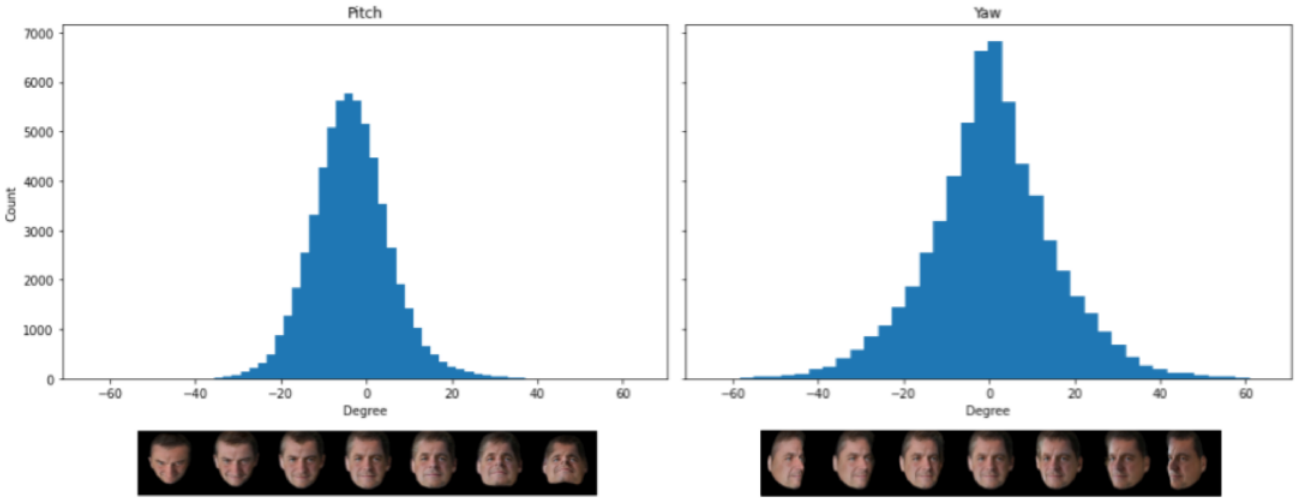

Figure 1. Estimated head pose distribution in the training set [14]. The histograms show the head pose distribution in the FFHQ training set [15], as estimated by MediaPipe [20]. Even though the dataset has a strong bias against extreme head poses, VariTex still generates consistent and realistic faces for up to 45° rotation from the frontal pose (Fig. 8 and 12).

	Ours vs. DFG [8]	**Ours** vs. ConfigNet [18]	**Ours** vs. GIF [11]
±15°	63.33 ± 3.81	62.07 ± 3.86	91.88 ± 2.14
±30°	77.62 ± 3.25	83.13 ± 3.05	91.76 ± 2.13
±45°	80.56 ± 1.12	95.61 ± 2.16	98.67 ± 0.63

Table 1. User study results (consistency task). The numbers (all in percent %) indicate how often participants preferred our method against related works (higher is better, equality at 50%), together with the corresponding 95% confidence intervals. Our method is clearly preferred in all settings. For extreme head poses ($\geq 30°$), our method outperforms related works by a margin of at least 27%. Please refer to Fig. 2 for a visual plot.

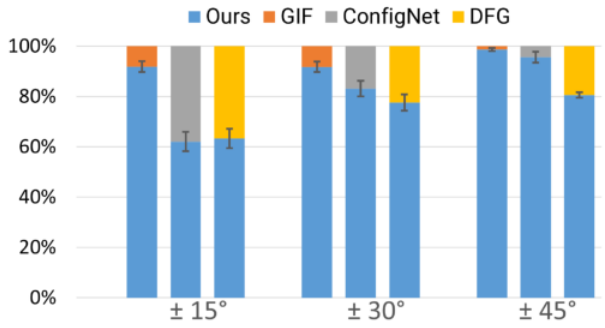

Figure 2. User study results (consistency task). We show how often participants preferred our images against related works (GIF [11], ConfigNet [18], DiscoFaceGAN [8]). Participants compared a set of 3 images of the same identity synthesized in different head poses (-15°, 0°, 15°), (-30°, 0°, 30°), and (-45°, 0°, 45°). The error bars give the 95% confidence intervals. Our method significantly outperforms the other state-of-the-art methods. Please refer to Tbl. 1 for the detailed numerical results and Fig. 19 for random examples used in our user study.

Face Model Fitting. We fit a 3D morphable face model [2, 10] to monocular RGB images in order to obtain shape and expression parameters and head pose. The fitting process consists of two stages. First, we estimate the head pose and a dense set of facial landmarks. Second, we solve an optimization problem to find the face model coefficients for shape and expression.

In the first step, we predict head pose and 468 3D facial landmarks with an existing pipeline, MediaPipe [20].

In the second step, we establish a set of 322 correspondences with the Basel Face Model 2017 [10]. With these correspondences, we solve an optimization problem to fit coefficients for shape $\boldsymbol{\alpha} \in \mathbb{R}^{199}$ and expression $\boldsymbol{\beta} \in \mathbb{R}^{100}$:

$$(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) = \underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{322} w_i (l_i - v_i(\boldsymbol{\alpha}, \boldsymbol{\beta}))^2, \qquad (1)$$

where v_i is a function that generates the MediaPipe-aligned vertices of the 3D face mesh corresponding to the i-th correspondence for shape and expression coefficients ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$); w_i are empirically chosen weights [28].

We filter out nine images where the MediaPipe face detection confidence is very low (which happens for low quality images and images with heavy occlusions, see Fig. 6).

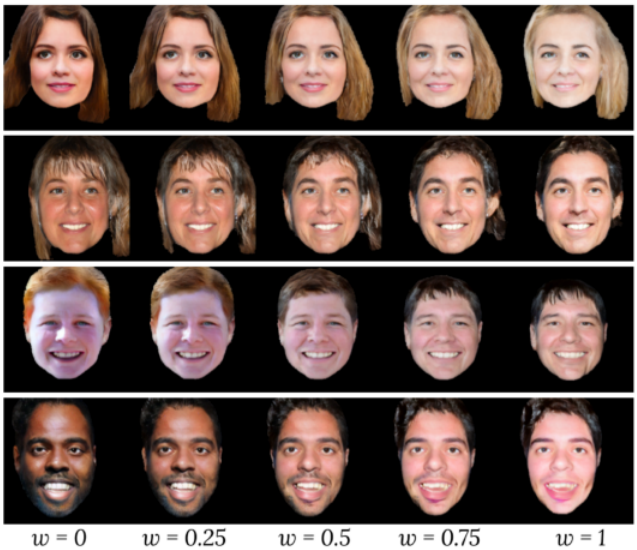

Figure 3. Interpolations. In each row, we linearly interpolate two identity codes $\boldsymbol{z}_A$ and $\boldsymbol{z}_B$. The images at the very left and right are generated using $\boldsymbol{z}_A$, and $\boldsymbol{z}_B$ respectively. The intermediate faces are generated by linearly interpolating the latent codes: $\text{linerp}(\boldsymbol{z}_A, \boldsymbol{z}_B, w) = w \cdot \boldsymbol{z}_A + (1 - w) \cdot \boldsymbol{z}_B$.

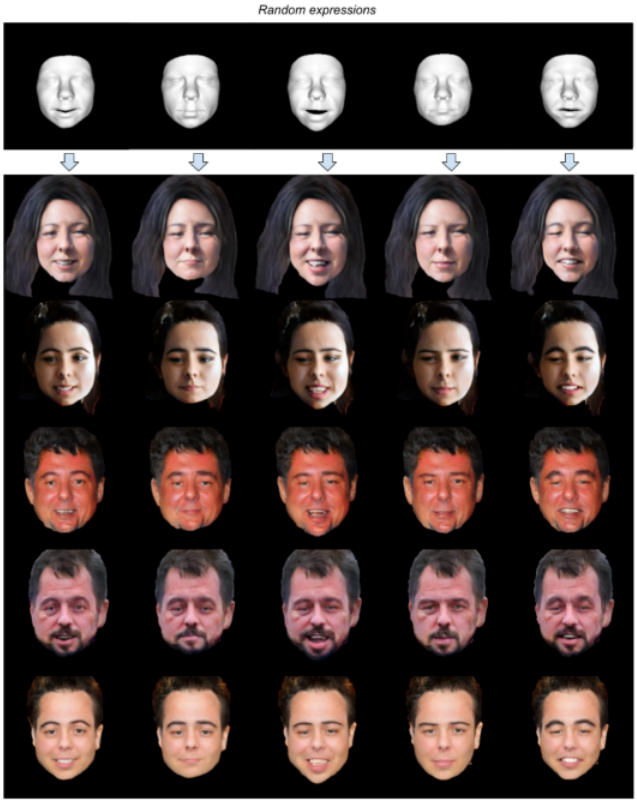

Figure 4. Identity consistency for random expressions. We sample random expressions and render them for multiple identities. The identity specific appearance of each person is preserved well.

Figure 5. Identity consistency for random shapes. In each column, we render the same random face shapes from the face model for different identities with a neutral expression.

Figure 6. All of the extreme outlier images that we removed from the FFHQ dataset [15].

For FFHQ [14], the fitting process results in 59,991 training and 10,000 test samples.

2.2. Head Pose Distribution in the Training Set

VariTex can generate consistent results for extreme target head poses. This is particularly remarkable because we train on a monocular dataset with a highly non-uniform distribution of head poses leading to a strong pose imbalance in our training data. Fig. 1 plots the distribution of pitch and yaw in the FFHQ [14] training set, as estimated by MediaPipe [20]. We do not use multi-view or multi-pose consistency losses to alleviate this problem, but rather rely on the natural geometry consistent design of our texture-space synthesis pipeline. Please see Sec. 3.4 for a more details.

2.3. User Study Setting

In order to evaluate the perceptual quality of our generated images, we conducted a user study in which we compared our results against three state-of-the-art methods [8, 11, 18]. We generated the results for these methods using the implementation and pre-trained models provided by them publicly on their project webpages. For a fair comparison, we removed the background for the methods where the face was not zoomed in [8, 11]. For the first question (quantifying photorealism), we showed randomly selected pairs for the same random head pose for both our method and one of theirs. We show such examples from our user study in Fig. 18. For the second question (identity consistency), we showed a set of 3 images for head poses in $\{\pm 45°, \pm 30°, \pm 15°\}$ for pitch, yaw, and pitch+yaw (each group also contains a frontal image with pose 0°). Fig. 19 shows such an example set of images for each head pose that was used in the user study. The order of the methods, as well as the order of the pairs, was randomized and not seen by the participants.

3. Supplementary Results

3.1. Comparison with Related Work

In the main paper, we qualitatively compare with related work in Fig. 4 by changing pitch and yaw simultaneously. We complement this comparison by reposing along each axis individually in Fig. 15 and 16. In addition, we complement the consistency scores in Tbl. 2 in the main paper with standard deviations in Tbl. 2.

3.2. User Study Results

The main paper reports user study results for identity consistency for poses $(-45°, 0°, 45°)$. In the following, we provide supplementary results for the same user study setting, but less extreme head poses: $(-30°, 0°, 30°)$ and $(-15°, 0°, 15°)$.

Fig. 2 illustrates how often participants preferred ours against the state-of-the-art methods. Our method outperforms the other methods with a margin of 30% for extreme head poses ($\pm 45°$). For less extreme poses, participants chose our method over the others in at least 61% of all examples.

We provide the numerical results and 95% confidence intervals in Tbl. 1. We assume a Gaussian distribution and compute the confidence intervals using a student's T distribution:

$$(\mu - t_{n-1} \cdot \frac{s}{\sqrt{n}}, \mu + t_{n-1} \cdot \frac{s}{\sqrt{n}}), \tag{2}$$

where n is the sample size, μ and s are the empirical mean and standard deviation, and t_{n-1} is the statistic for the 95% confidence interval of the two-tailed student's T distribution for $(n-1)$ degrees of freedom.

3.3. Supplementary Qualitative Results

We provide additional results for sampled expressions and shape in Figures 4 and 5. Fig. 12 shows another set of identities for different head poses. In Fig. 13, we reconstruct real images and re-pose them. The latent identity code used for the rendering is the distribution mean $\boldsymbol{\mu}_z$ predicted by the encoder: $\boldsymbol{z} = \boldsymbol{\mu}_z$. Note that unlike other works [1, 3], we do not find the latent code by optimization—it is directly predicted by the encoder. Not all input images are frontal, so the decoder network hallucinates the hidden regions. Fig. 8 shows renderings outside the main training distribution (plotted as histograms). For expression (top row), we vary the first expression coefficient, keeping all other coefficients zero. This yields cartoon-like geometries at the extremes, which are still rendered at good quality. For the pose, we show examples between -60°to 60°.

The bottom row of Fig. 7 shows renderings for the same latent code for the face texture $\boldsymbol{z}_{face}$ with sampled additive codes $\boldsymbol{z}_{additive}$. The top row visualizes the superimposition of the rendered images onto arbitrary backgrounds.

Figure 7. Top: The predicted masks allow alpha blending of backgrounds. Bottom: We combine the one latent code for the face with different additive codes.

Identity Mixing and Interpolations. VariTex can not only sample new identities from learned distributions but also walk the latent space. For example, it can gradually morph one face into another by linearly interpolating the latent identity code $\boldsymbol{z}$. We show some examples in Fig. 3.

3.4. Supplementary Ablations

In our main paper, we provide an ablation study for neural textures. In this supplementary, we complement our ablations by analyzing the effect of the proposed augmentation scheme (Sec. 3.5 in the main paper).

The augmentation scheme makes our model robust towards changes in head pose and translation of the output, which is required to render plausible exterior regions of the face for different head poses and positions inside the image. Without augmentation, the additive decoder tends to ignore the conditioning on the neural feature image and produces

	Similarity yaw ↔						
[18]	0.208 ± 0.086	0.509 ± 0.078	0.790 ± 0.045	-	0.795 ± 0.042	0.515 ± 0.075	0.257 ± 0.090
[11]	0.133 ± 0.094	0.264 ± 0.115	0.485 ± 0.114	-	0.487 ± 0.108	0.257 ± 0.108	0.117 ± 0.090
[8]	0.530 ± 0.074	0.691 ± 0.055	0.866 ± 0.031	-	0.863 ± 0.032	0.675 ± 0.058	0.521 ± 0.076
Ours	**0.568** ± 0.093	**0.729** ± 0.067	**0.874** ± 0.038	-	**0.873** ± 0.041	**0.732** ± 0.068	**0.585** ± 0.086
Ref [29]	0.855 ± 0.069	0.845 ± 0.067	0.726 ± 0.052	-	0.790 ± 0.048	0.773 ± 0.075	0.779 ± 0.051
	Similarity pitch ↕						
[18]	-0.008 ± 0.072	0.014 ± 0.078	0.459 ± 0.083	-	0.476 ± 0.087	0.095 ± 0.075	0.010 ± 0.067
[11]	0.039 ± 0.076	0.164 ± 0.102	0.400 ± 0.125	-	0.448 ± 0.111	0.191 ± 0.101	0.095 ± 0.087
[8]	0.270 ± 0.099	0.461 ± 0.097	0.781 ± 0.053	-	**0.826** ± 0.045	0.581 ± 0.077	0.388 ± 0.093
Ours	**0.416** ± 0.094	**0.611** ± 0.082	**0.821** ± 0.047	-	0.817± 0.048	**0.611**± 0.084	**0.420** ± 0.096
Ref [29]	0.719 ± 0.073	0.725 ± 0.077	0.753 ± 0.063	-	0.797 ± 0.055	0.805 ± 0.045	0.782 ± 0.040
	-45°	-30°	-15°	0°	15°	30°	45°

Table 2. Identity consistency for different head poses. We complement the cosine similarity scores from Tbl. 2 in the main paper with the standard deviations. Note that cosine similarity yields values in the range $[-1, 1]$.

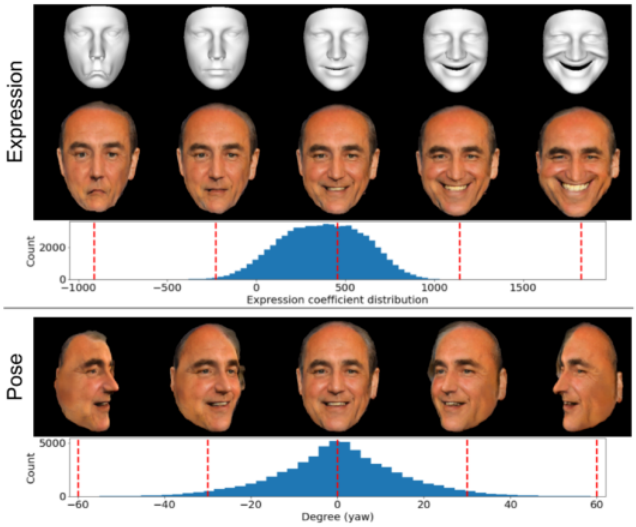

Figure 8. Samples from our model under extreme head pose and expression, extrapolating from the training data. Even for very unnatural facial geometry, the model outputs a reasonable rendering. The histograms plot the training distributions.

features in the wrong spatial location (e.g., for hair or eyes). Fig. 17 shows how the output image becomes distorted for a model trained without any augmentations and visualizes the effect of using lower-dimensional or RGB-only textures.

We provide FID [12] and consistency scores in Tbl. 3. Note that the consistency score is computed by ArcFace [7], which expects a cropped face. Therefore, even with misplaced exterior regions (as described above), a high consistency score can be attained, even though the generated image is of poor visual quality.

It might seem surprising that VariTex can consistently render such extreme head poses, despite being trained on mostly frontal faces (Fig. 1). A reasonable explanation is that the strict mapping from texture to image space assists the networks to learn extreme poses, even from very few examples. We conduct two experiments to support this explanation. Please recall that the neural texture projects to a pixel-aligned feature image, which is translated to RGB by the Feature2Image renderer (Fig. 3.1 in the main paper). First, we compare the facial appearance for different poses. In Fig. 9, we generate faces under different poses and warp the output images to a frontal pose. We observe that the neural renderer can adapt lighting (first row), but remains highly identity consistent.

We further investigate the behavior of the neural render by feeding manipulated and corrupted feature images. In Fig. 10, we split feature images in half and combine different identities. The rendered outputs are largely the same as a concatenation in the output space, which indicates that for the face regions defined by the texture, the renderer considers very small pixel neighborhoods. Still, the renderer adapts features when needed: For the facial contours and the concatenation line, the renderer changes the outputs, best visible for the hair in the example on the right. In Fig. 11, we randomly crop holes in the feature image. The rendered outputs are mostly locally affected around the cropped patch.

3.5. Limitations

We show in this paper that synthesizing novel identities in the texture space of a face model enables highly consistent reposing for the face region. However, the face model [10] covers only the frontal geometry of the face, leaving out regions such as the hair and mouth interior. This makes it very challenging to re-pose them. Our additive decoder partially mitigates this problem by producing plausible features for these regions on a per-frame basis. But

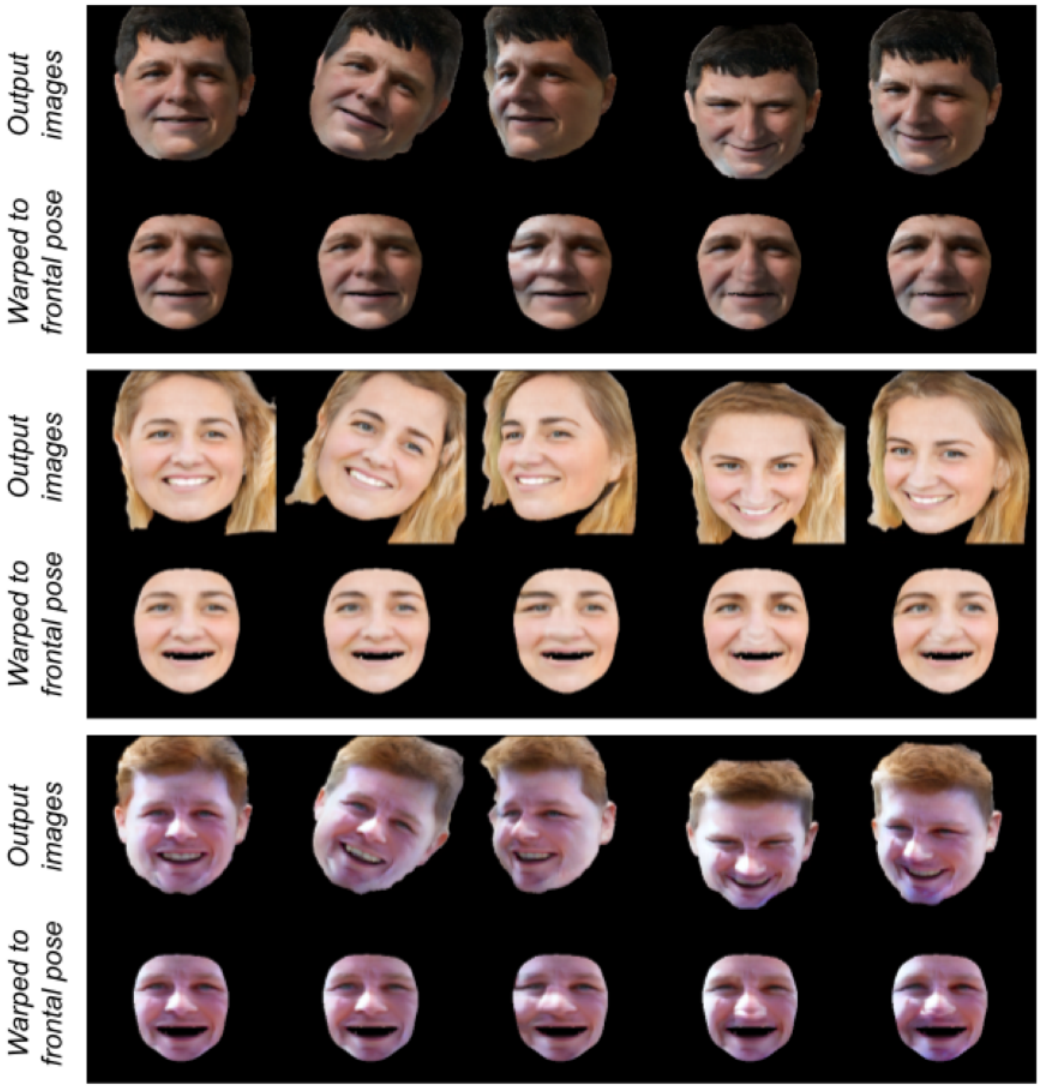

Figure 9. Investigating the behavior of the Feature2Image renderer for different poses. We generate faces under different poses and use the given 3D geometries to warp them to the frontal pose. The frontalized faces are highly identity consistent while the lighting might be changed (top and bottom).

the additive decoder has no notion of temporal consistency; the produced features yield variations over pose animations. This prevents the exterior regions from having the same quality of geometry consistency as the face in re-posed images. View consistent synthesis for such regions, particularly with unlabelled monocular training data, is a challenging and open research problem, which will be an interesting avenue for future work. A very recent work [26] shows a promising direction to solve this problem by proposing a full head model that includes hair.

A related problem is handling other objects usually associated with face images, such as glasses and hats. Since they are not part of the face model but are present in the training images, they are "texture-copied" into our synthesized images. But without their geometry, any reposing results in projectively distorted outputs. We show some examples in Fig. 14. Once again, a possible interesting solution would be to develop and use a graphics model that can provide the geometry for such objects.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4432–4441, 2019. 4

[2] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive tech-*

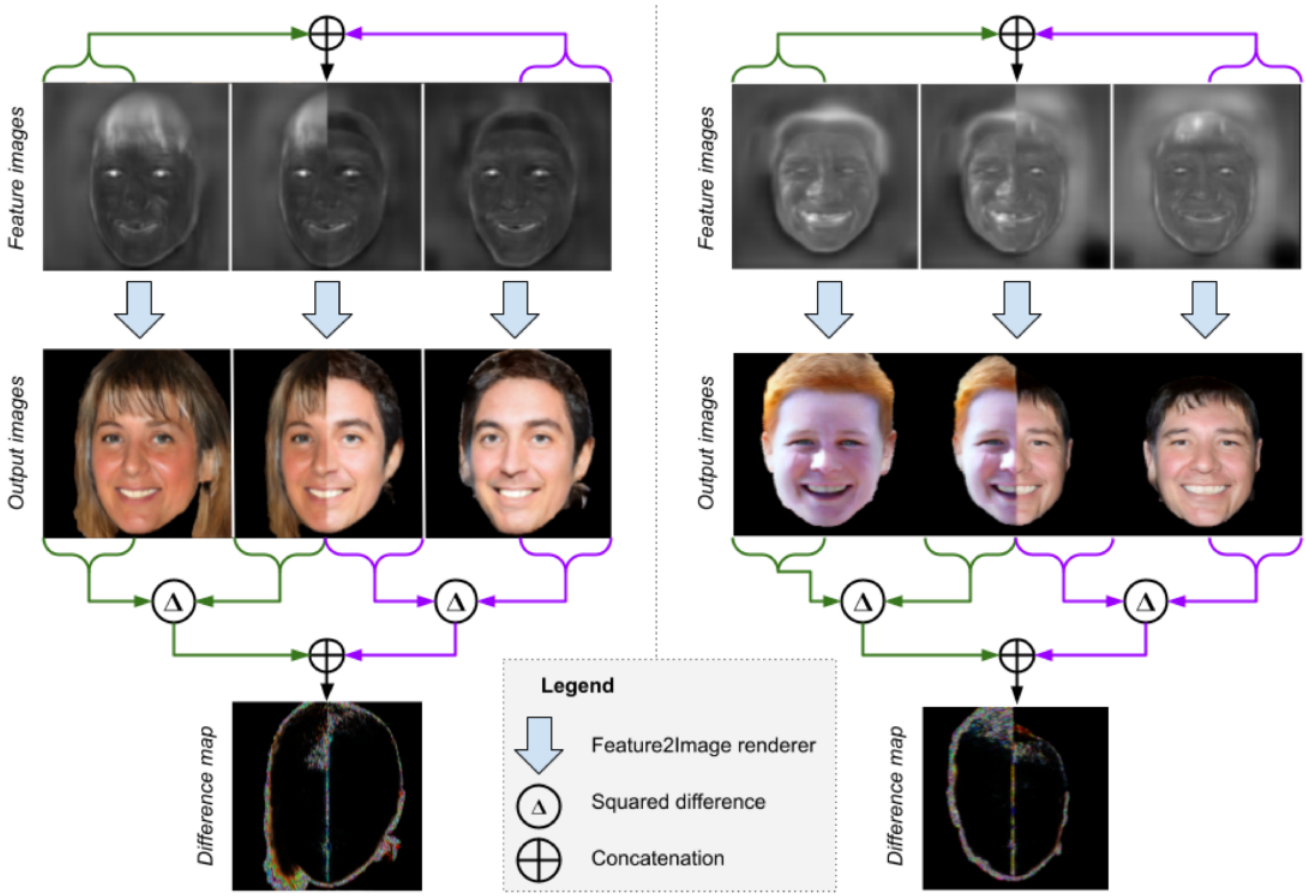

Figure 10. Investigating the behavior of the Feature2Image renderer on manipulated feature images. We concatenate partial feature images (top) for different identities under the same pose. We render them (middle) and compare the output images with the concatenation of the original renderings in a difference map (bottom). We observe that the Feature2Image network adapts the region around the concatenation line and the contours of the face. For most of the face, the renderer performs pixelwise feature-to-RGB translation, but it adapts the less well defined regions, where no 3D geometries are available.

	Flip	Rotation	Translation	Scaling	# Channels	$\mathcal{L}_{RGB}$	**FID** ↓	**Consistency (yaw)** ↑
3-dim + $\mathcal{L}_{RGB}$	✓	✓	✓	✓	3	✓	54.27	0.712 ± 0.123
3-dim w/o $\mathcal{L}_{RGB}$	✓	✓	✓	✓	3	×	47.87	0.684 ± 0.132
16-dim + $\mathcal{L}_{RGB}$	✓	✓	✓	✓	16	✓	37.96	0.724 ± 0.119
None	×	×	×	×	16	×	37.37	0.726 ± 0.114
f	✓	×	×	×	16	×	36.96	0.709 ± 0.120
f + r	✓	✓	×	×	16	×	45.03	0.673 ± 0.125
f + t	✓	×	✓	×	16	×	40.23	0.725 ± 0.118
f + t + s	✓	×	✓	✓	16	×	48.86	0.707 ± 0.119
f + r + s	✓	✓	×	✓	16	×	38.81	0.702 ± 0.124
Ours	✓	✓	✓	✓	16	×	**34.35**	$\mathbf{0.727} \pm 0.121$

Table 3. Extended version of our ablation study in the main paper (Sec. 5.4). We compare photorealism (FID [12]) and identity consistency (Sec. 4 in the main paper) for *neural* vs. *RGB* textures. A higher-dimensional neural texture can yield photorealistic outputs, while also maintaining high consistency. Fig. 17 shows visual examples. *# Channels* refers to the number of channels in the texture. The letters *f, r, t, s* stand for flipping, rotation, translation, and scaling, as described in Sec. 1.2.

niques, pages 187–194, 1999. 2

[3] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu,

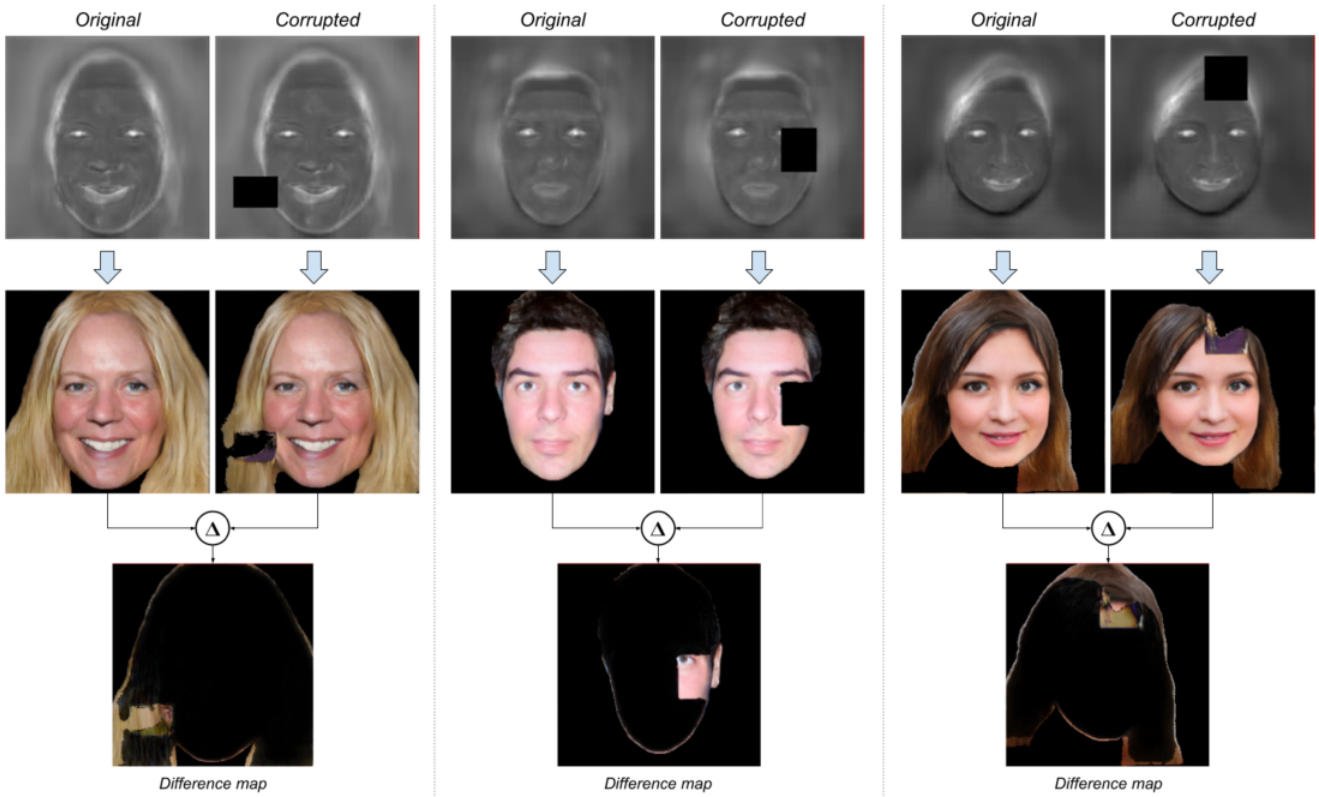

Figure 11. Rendering corrupted feature images. We randomly crop holes in the feature image (top), render the resulting corrupted versions (middle), and visualize their absolute difference (Δ, bottom). The Feature2Image renderer produces the same output for the unchanged features—the corruption mostly affects the local patch.

and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5799–5809, 2021. 4

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 1

[5] Liang Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Rethinking atrous convolution for semantic image segmentation liang-chieh. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. 1

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[7] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019. 5

[8] Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3d imitative-contrastive learning. In *IEEE Computer Vision and Pattern Recognition*, 2020. 2, 4, 5

[9] WA Falcon and .al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019. 1

[10] Thomas Gerig, Andreas Morel-Forster, Clemens Blumer, Bernhard Egger, Marcel Luthi, Sandro Schönborn, and Thomas Vetter. Morphable face models-an open framework. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 75–82. IEEE, 2018. 1, 2, 5

[11] Partha Ghosh, Pravir Singh Gupta, Roy Uziel, Anurag Ranjan, Michael J. Black, and Timo Bolkart. GIF: Generative interpretable faces. In *International Conference on 3D Vision (3DV)*, 2020. 2, 4, 5

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017. 5, 7

[13] Sergey Ioffe and Christian Szegedy. Batch normalization:

Figure 12. Additional results for different yaw angles. Our method preserves the identity very well even for very strong head rotations.

Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 1

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen.

Figure 13. Reposing real images. We encode unseen real images and rerender them under new head poses. The latent identity code $\boldsymbol{z}$ is directly predicted by the encoder without any further optimization. As the encoder was trained as a Variational Auto Encoder [17], the reconstructions do not perfectly match the original images but they look similar.

Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 2, 3

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 1, 2, 3

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[17] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 10

[18] Marek Kowalski, Stephan J. Garbin, Virginia Estellers, Tadas Baltrušaitis, Matthew Johnson, and Jamie Shotton. Config: Controllable neural face image generation. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 4, 5

[19] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5549–5558, 2020. 1

[20] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. 2, 3

[21] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–

Figure 14. Limitations. In the current implementation, objects in the face region such as glasses are synthesized as part of the face texture. When reposing such an image, the object gets distorted by perspective projection because it has no consistent geometry—it is simply projected on the surface of the face. This could potentially be mitigated by using explicit 3D geometry and a corresponding UV parameterization for such objects.

Layer	#Params	Output Shape
Conv2d	1.7 K	[64, 256, 256]
MaxPool2d	0	[64, 256, 256]
4× Conv2d	36.9 K	[64, 256, 256]
Conv2d	73.7 K	[128, 128, 128]
Conv2d	147 K	[128, 128, 128]
Conv2d	8.2 K	[128, 128, 128]
2× Conv2d	147 K	[128, 128, 128]
Conv2d	294 K	[256, 64, 64]
Conv2d	589 K	[256, 64, 64]
Conv2d	32.8 K	[256, 64, 64]
2× Conv2d	589 K	[256, 64, 64]
Conv2d	1.2 M	[512, 32, 32]
Conv2d	2.4 M	[512, 32, 32]
Conv2d	131 K	[512, 32, 32]
2× Conv2d	2.4 M	[512, 32, 32]
AdaptiveAvgPool2d	0	[512, 1, 1]
2× Linear	131K	[256]
Total	**11.2 M**	$\boldsymbol{\mu}_z$: [256]; $\boldsymbol{\sigma}_z^2$: [256]

Table 4. Encoder architecture. We report the parametric and pooling layers with their number of parameters and output shape (number of channels, height, width). The outputs of this network are the parameters of the latent distribution, consisting of a mean vector $\boldsymbol{\mu}_z \in \mathbb{R}^{256}$ and its diagonal covariance in the shape of a vector $\boldsymbol{\sigma}_z^2 \in \mathbb{R}^{256}$. For brevity, we omit activations (ReLU) and batch normalization. The output shapes do not contain a batch dimension.

Layer	#Params	Output Shape
Conv2d	2.4 M	[512, 4, 4]
Conv2d	1.2 M	[256, 8, 8]
Conv2d	131 K	[256, 8, 8]
4× Conv2d	589 K	[256, 8, 8]
Conv2d	589 K	[256, 16, 16]
Conv2d	65.5 K	[256, 16, 16]
3× Conv2d	589 K	[256, 16, 16]
Conv2d	589 K	[256, 32, 32]
Conv2d	65.5 K	[256, 32, 32]
3× Conv2d	589 K	[256, 32, 32]
Conv2d	294 K	[128, 64, 64]
Conv2d	32.8 K	[128, 64, 64]
3× Conv2d	147 K	[128, 64, 64]
Conv2d	73.7 K	[64, 128, 128]
Conv2d	8.2 K	[64, 128, 128]
3× Conv2d	36.9 K	[64, 128, 128]
Conv2d	36.9 K	[64, 256, 256]
Conv2d	4.1 K	[64, 256, 256]
2× Conv2d	36.9 K	[64, 256, 256]
Conv2d	9.2 K	[16, 256, 256]
Total	**12.4 M**	[16, 256, 256]

Table 5. Texture decoder architecture. We report the convolutional layers with their number of parameters and output shape (channels, height, width). For brevity, we omit activations (ReLU) and batch normalization. Please note that the output shapes do not contain a batch dimension.

2346, 2019. 1

[22] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 1

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 1

[24] Justus Thies, Mohamed Elgharib, Ayush Tewari, Christian Theobalt, and Matthias Nießner. Neural voice puppetry: Audio-driven facial reenactment. In *European Conference on Computer Vision*, pages 716–731. Springer, 2020. 1

[25] 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. 1

[26] Tarun Yenamandra, Ayush Tewari, Florian Bernard, Hans-Peter Seidel, Mohamed Elgharib, Daniel Cremers, and Christian Theobalt. i3dmm: Deep implicit 3d morphable model of human heads. *arXiv preprint arXiv:2011.14143*, 2020. 6

[27] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[28] Yu Yu, Kenneth Alberto Funes Mora, and Jean-Marc Odobez. Robust and accurate 3d head pose estimation through 3dmm and online head model reconstruction. In *2017 12th ieee international conference on automatic face & gesture recognition (fg 2017)*, pages 711–718. Ieee, 2017. 2

[29] Xucong Zhang, Seonwook Park, Thabo Beeler, Derek

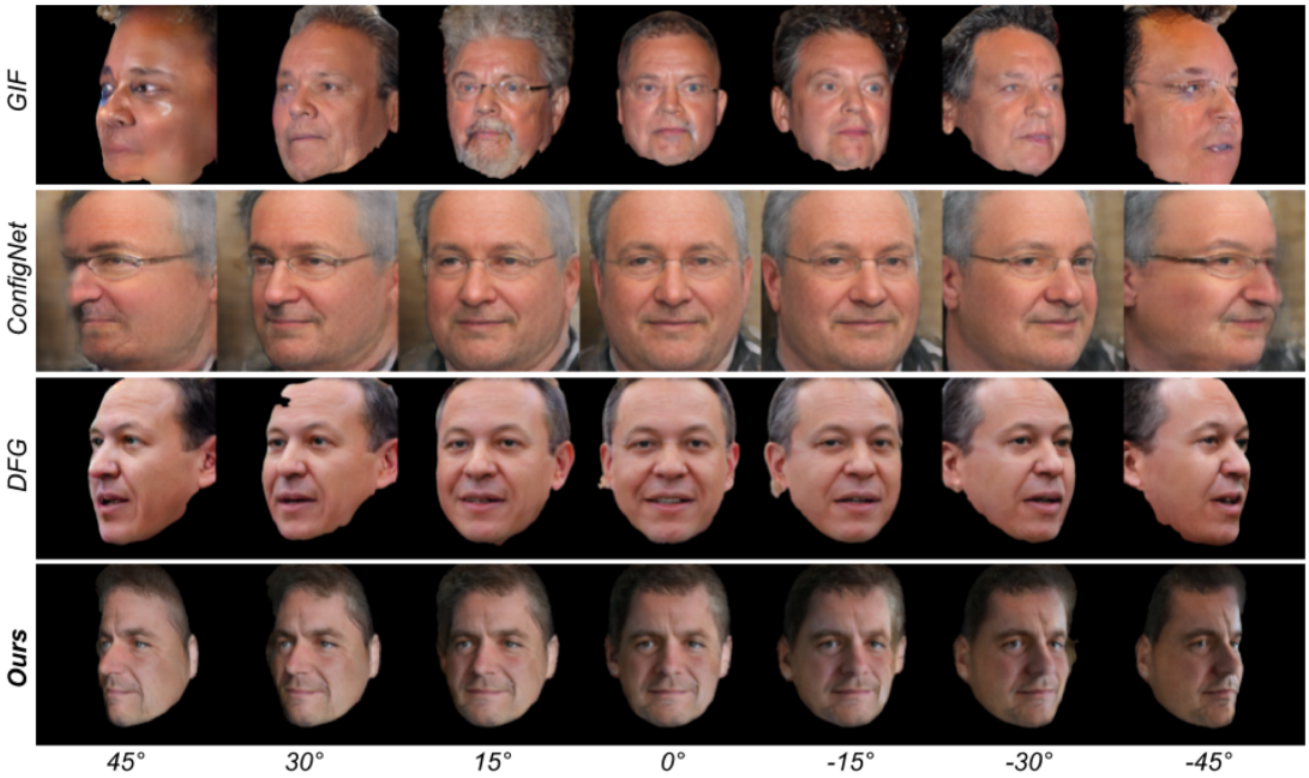

Figure 15. Comparison with state-of-the-art methods for head pose rotations around the vertical axis.

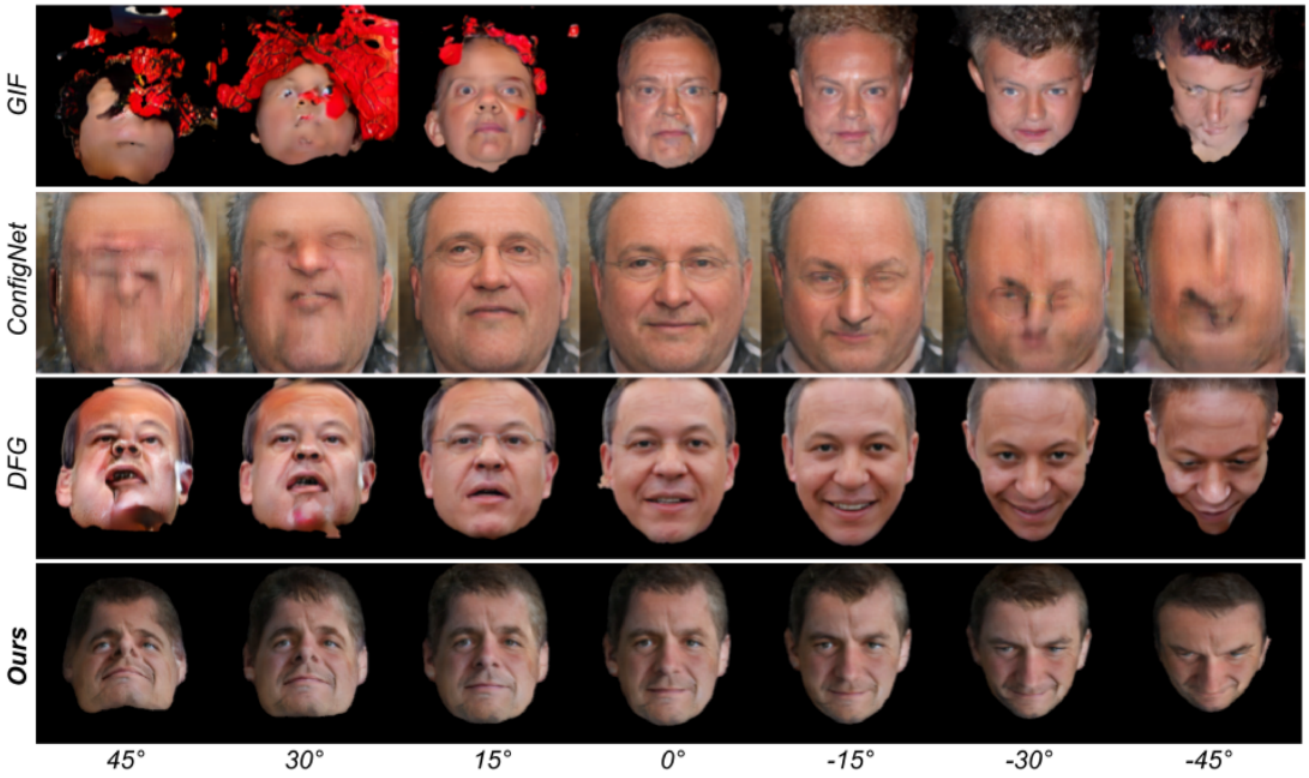

Figure 16. Comparison with state-of-the-art methods for head pose rotations around the horizontal axis.

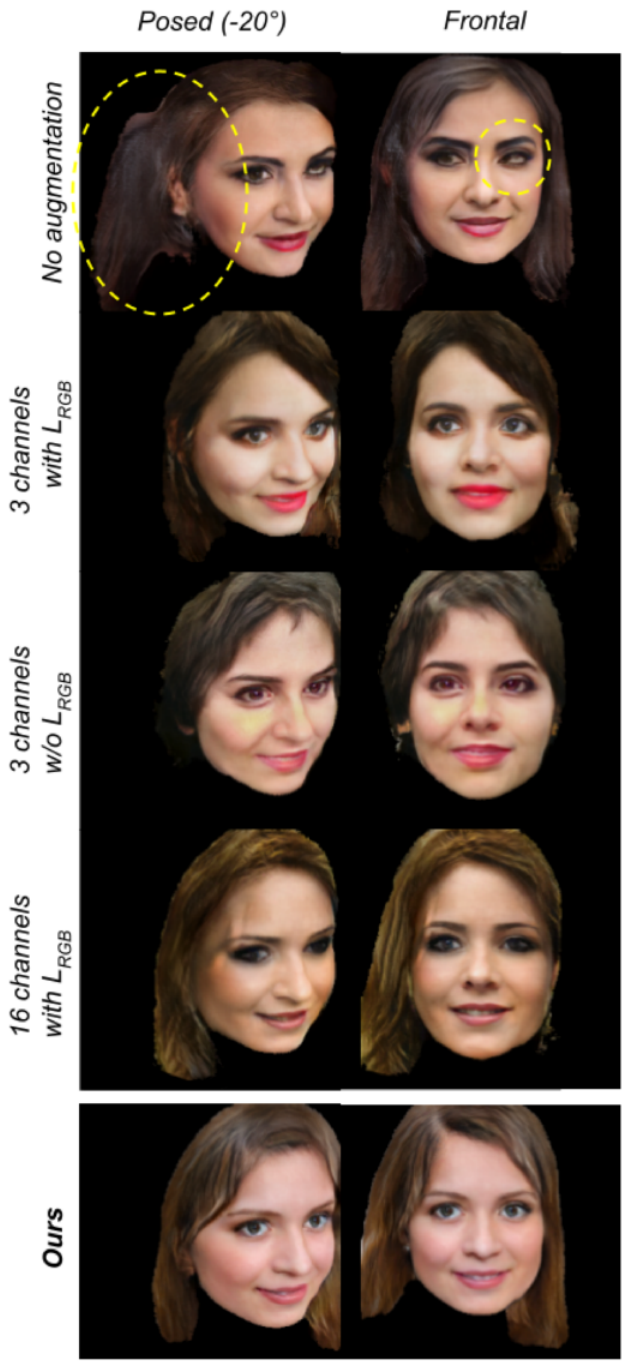

Figure 17. Ablation study visualizations. A model trained without any augmentations tends to misplace features where no 3D geometries are available. This can yield unrealistic shapes in the exterior region or artifacts (row 1). Low-dimensional textures produce blurred out images (row 2). Using a loss that enforces RGB in the first three texture channels yields unrealistic colors in the output (row 3 and 4).

Layer	#Params	Output Shape
Conv2d	2.4 M	[512, 4, 4]
Conv2d	2.4 M	[512, 8, 8]
Conv2d	262 K	[512, 8, 8]
3× Conv2d	2.4 M	[512, 8, 8]
Conv2d	1.2 M	[256, 16, 16]
Conv2d	131 K	[256, 16, 16]
Conv2d	626 K	[256, 16, 16]
Conv2d	589 K	[256, 16, 16]
Conv2d	626 K	[256, 16, 16]
Conv2d	589 K	[256, 32, 32]
Conv2d	65.5 K	[256, 32, 32]
Conv2d	626 K	[256, 32, 32]
Conv2d	589 K	[256, 32, 32]
Conv2d	626 K	[256, 32, 32]
Conv2d	294 K	[128, 64, 64]
Conv2d	32.8 K	[128, 64, 64]
Conv2d	165 K	[128, 64, 64]
Conv2d	147 K	[128, 64, 64]
Conv2d	165 K	[128, 64, 64]
Conv2d	73.7 K	[64, 128, 128]
Conv2d	8.2 K	[64, 128, 128]
Conv2d	46.1 K	[64, 128, 128]
Conv2d	36.9 K	[64, 128, 128]
Conv2d	46.1 K	[64, 128, 128]
Conv2d	36.9 K	[64, 256, 256]
Conv2d	4.1 K	[64, 256, 256]
Conv2d	46.1 K	[64, 256, 256]
Conv2d	36.9 K	[64, 256, 256]
Conv2d	9.2 K	[16, 256, 256]
Total	**19.2 M**	[16, 256, 256]

Table 6. Additive decoder architecture. We report the convolutional layers with their number of parameters and output shape (channels, height, width). For brevity, we omit activations (ReLU) and batch normalization. Please note that the output shapes do not contain a batch dimension.

Bradley, Siyu Tang, and Otmar Hilliges. Eth-xgaze: A large scale dataset for gaze estimation under extreme head pose and gaze variation. In *European Conference on Computer Vision*, pages 365–381. Springer, 2020. 5

Layer	#Params	Output Shape
Conv2d	18.5 K	[64, 256, 256]
Conv2d	36.9 K	[64, 256, 256]
MaxPool2d	0	[64, 128, 128]
Conv2d	73.9 K	[128, 128, 128]
Conv2d	147 K	[128, 128, 128]
MaxPool2d	0	[128, 64, 64]
Conv2d	295 K	[256, 64, 64]
Conv2d	590 K	[256, 64, 64]
MaxPool2d	0	[256, 32, 32]
Conv2d	1.2 M	[512, 32, 32]
Conv2d	2.4 M	[512, 32, 32]
MaxPool2d	0	[512, 16, 16]
Conv2d	4.7 M	[1024, 16, 16]
Conv2d	9.4 M	[1024, 16, 16]
ConvTranspose2d	2.1 M	[512, 32, 32]
Conv2d	4.7 M	[512, 32, 32]
Conv2d	2.4 M	[512, 32, 32]
ConvTranspose2d	524 K	[256, 64, 64]
Conv2d	1.2 M	[256, 64, 64]
Conv2d	590 K	[256, 64, 64]
ConvTranspose2d	131 K	[128, 128, 128]
Conv2d	295 K	[128, 128, 128]
Conv2d	147 K	[128, 128, 128]
ConvTranspose2d	32.8 K	[64, 256, 256]
Conv2d	73.8 K	[64, 256, 256]
Conv2d	36.9 K	[64, 256, 256]
Conv2d	260	[4, 256, 256]
Total	**31.1 M**	[4, 256, 256]

Table 7. Texture2Image Architecture. We report its layers with their number of parameters and output shape (channels, height, width). For brevity, we omit activations (ReLU) and batch normalization. Please note that the output shapes do not contain a batch dimension. The final output shape contains three channels for the RGB output image and one channel for the predicted foreground mask.

Figure 18. User study examples (photorealism task). Participants compare randomly selected pairs of images from each method. For a fair comparison, each such pair contains images in the same randomly-selected pose.

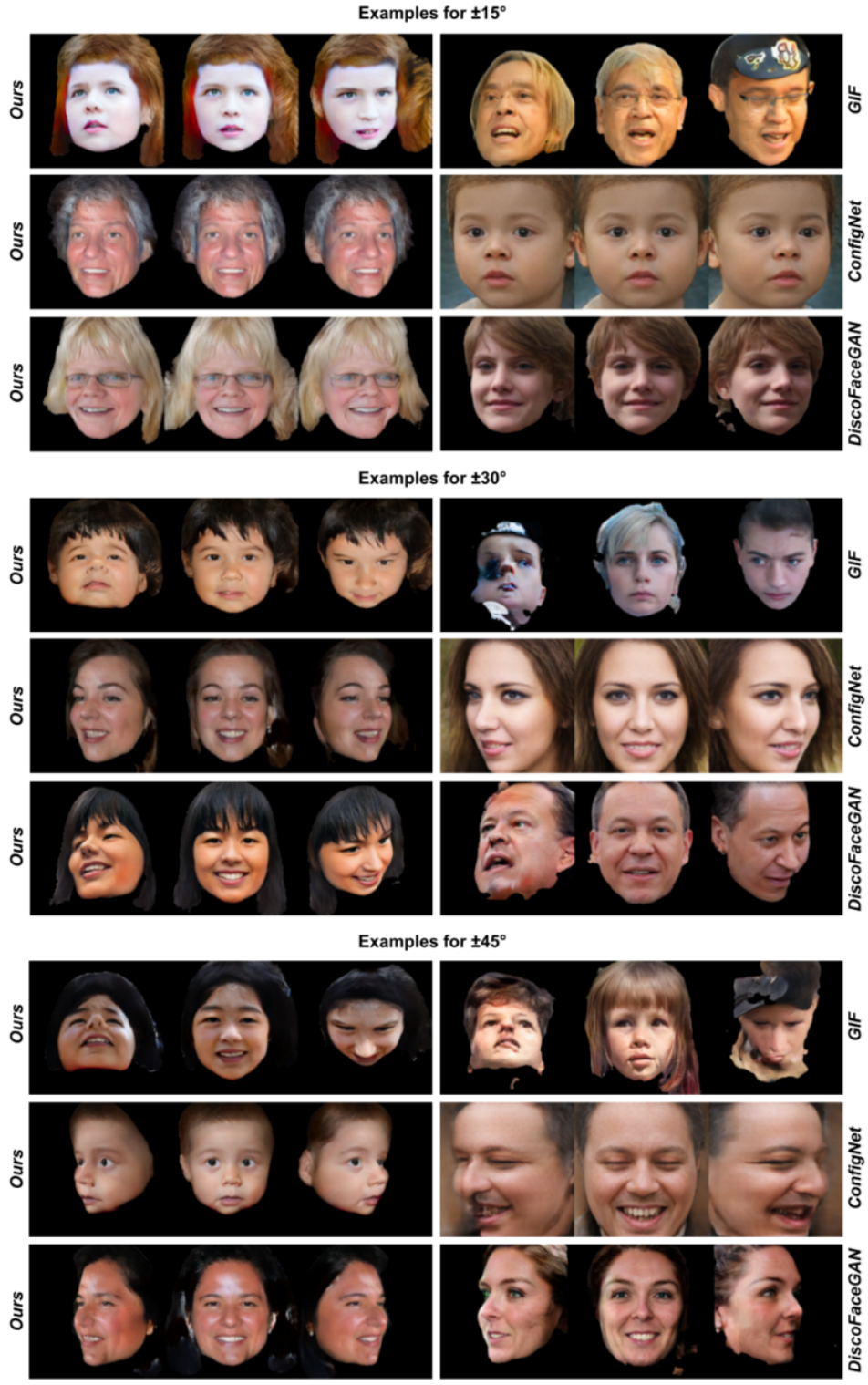

Figure 19. User study examples (consistency task). We showed a randomly selected set of 3 images for the evaluated head poses and asked the participants to choose the set where the person was represented more consistently. The sequence of questions and options were randomized and the labels were not shown to the user.