A Shading-Guided Generative Implicit Model for Shape-Accurate 3D-Aware Image Synthesis

Supplementary Material

Xingang Pan[1] **Xudong Xu**[2] **Chen Change Loy**[3] **Christian Theobalt**[1] **Bo Dai**[3]

[1]Max Planck Institute for Informatics [2]The Chinese University of Hong Kong
{xpan,theobalt}@mpi-inf.mpg.de xx018@ie.cuhk.edu.hk
[3]S-Lab, Nanyang Technological University
{ccloy, bo.dai}@ntu.edu.sg

In this supplementary material, we provide the implementation details, more qualitative results, and a discussion of broader impacts of our work.We also recommend readers to refer to the attached video demos for animated 3D-aware image synthesis.

1 Implementation Details

1.1 Model Architectures

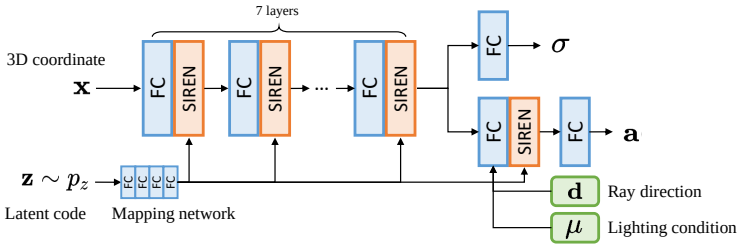


Figure 1: The generator architecture used in ShadeGAN.

Generator and discriminator. For the generator and discriminator architectures, we mainly follow the design of pi-GAN [1]. Specifically, the generator is an MLP formed by stacking fully-connected and FiLM-SIREN layers [2, 3] with 256 units, as shown in Fig. 1. The generator takes a 3D coordinate $\boldsymbol{x}$ and a 256d latent code $\boldsymbol{z}$ as inputs. It consists of two branches to predict the volume density σ and the albedo $\boldsymbol{a}$ respectively, where the albedo is also conditioned on the ray direction $\boldsymbol{d}$ and the lighting condition $\boldsymbol{\mu}$. Similar to StyleGAN [4], the latent code $\boldsymbol{z}$ is sent to a mapping network with four fully-connected and ReLU layers to produce the frequency and phase factors for the FiLM-SIREN layers of the generator. As for the discriminator, we adopt the same convolutional neural network (CNN) architecture as in [1], which uses CoordConv layers [5] and residual connections [6].

Surface tracking network. Our surface tracking network S is a decoder-style CNN as shown in Tab. 1 and Tab. 2. It takes the output of the mapping network and a camera pose $\boldsymbol{\xi}$ as inputs and outputs a 128×128 resolution depth map, where the camera pose $\boldsymbol{\xi}$ refers to pitch and yaw angles in this case. The abbreviations for the network layers are described below:

Conv(c_{in}, c_{out}, k, s, p): a convolution layer with c_{in} input channels, c_{out} output channels, kernel size k, stride s, and padding p.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Table 1: Architecture of the surface tracking network.

Layer	Output size
Linear(4608, 256) + concat($\boldsymbol{\xi} = (\text{pitch}, \text{yaw})$)	1
Linear(258, 256) + ReLU	1
DeConv(256, 256, 4, 1, 0) + ReLU	4
Conv(256, 256, 3, 1, 1) + ReLU	4
DeConv(256, 256, 4, 2, 1) + GN(32) + ReLU	8
Conv(256, 256, 3, 1, 1)	8
ResBlockUp(256, 128)	16
ResBlockUp(128, 64)	32
ResBlockUp(64, 32)	64
ResBlockUp(32, 16) + GN(4) + ReLU	128
Conv(16, 1, 5, 1, 2) + Sigmoid	128

Table 2: Network architecture for the ResBlockUp(c_{in}, c_{out}) in Tab.1. The output of Residual path and Identity path are added as the final output.

Residual path
GN($c_{in}/8$) + ReLU + Upsample(2)
Conv(c_{in}, c_{out}, 3, 1, 1) + GN($c_{out}/8$) + ReLU
Conv(c_{out}, c_{out}, 3, 1, 1)
Identity path
Upsample(2)
Conv(c_{in}, c_{out}, 1, 1, 0)

Deconv(c_{in}, c_{out}, k, s, p): a deconvolution layer with c_{in} input channels, c_{out} output channels, kernel size k, stride s, and padding p.

GN(n): a group normalization layer [7] with n groups.

BN and ReLU: the batch normalization layer [8] and the rectified linear unit.

Upsample(s): a nearest-neighbor upsampling layer with a scale of s.

ResBlockUp(c_{in}, c_{out}) and ResBlock($c_{in}, c_{out}, stride$): residual blocks as defined in Tab.2 and Tab.4.

Depth-predicting CNN. In our quantitative study on the BFM dataset, we train an additional CNN to predict the depth map of an input image, as described in Line 253-255 of the main paper. Here we provide the architecture of the depth-predicting CNN in Tab. 3, Tab. 2 and Tab. 4. It is trained using Adam [9] for 30 epochs with a learning rate of 1e-4.

Table 3: Architecture of the depth-predicting CNN. The input image is resized to 64×64 resolution. ResBlockUp is the same as Tab. 2 except that GNs are replaced by BNs.

Layer	Output size
Conv(3, 64, 4, 2, 1)	32
ResBlock(64, 128, 2)	16
ResBlock(128, 256, 2)	8
ResBlock(256, 512, 1)	8
ResBlock(512, 512, 1)	8
ResBlockUp(512, 256)	16
ResBlockUp(256, 128)	32
ResBlockUp(128, 64) + BN + ReLU	64
Conv(64, 1, 5, 1, 2) + Tanh	64

Table 4: Network architecture for the ResBlock(c_{in}, c_{out}, $stride$) in Tab.3. The output of Residual path and Identity path are added as the final output.

Residual path	
BN + ReLU + Conv(c_{in}, c_{out}, 3, $stride$, 1)	
BN + ReLU + Conv(c_{out}, c_{out}, 3, 1, 1)	
Identity path	
Identity	if $c_{in} = c_{out}$
Conv(c_{in}, c_{out}, 1, 1, 0)	if $c_{in} \neq c_{out}$

1.2 Training Details

We adopt a progressive growing training strategy as in [1]. During training, the image resolution grows from 32 to 128, while the batchsize and learning rate decrease as listed in Tab. 5. We use the Adam [9] optimizer with $\beta_1 = 0$, $\beta_2 = 0.9$. Our models are trained on 8 TITAN X Pascal GPUs. We use the same setting to train the pi-GAN [1] baseline for fair comparison.

For the camera pose $\boldsymbol{\xi}$, we follow pi-GAN [1] and constrain the camera position to the surface of a unit sphere and keep its direction to the origin. Thus, $\boldsymbol{\xi}$ could be determined by pitch and yaw angles, which are sampled from prior distributions. For BFM and CelebA, pitch $\sim \mathcal{N}(\pi/2, 0.155)$ and yaw $\sim \mathcal{N}(\pi/2, 0.3)$, while for Cats, pitch $\sim \mathcal{U}(\pi/2 - 0.5, \pi/2 + 0.5)$ and yaw $\sim \mathcal{U}(\pi/2 - 0.4, \pi/2 + 0.4)$. The lighting condition $\boldsymbol{\mu}$ consists of four factors (k_a, k_d, l_x, l_y), where the lighting direction could be determined by $\boldsymbol{l} = (l_x, l_y, 1)^T / \sqrt{l_x^2 + l_y^2 + 1}$. We use the multi-variate Gaussian distribution of $(2k_a - 1, 2k_d - 1, l_x, l_y)$ estimated using [10] as the prior distribution. The mean and covariance matrix for different datasets are provided in Tab. 6. For the results in Tab. 3 No.(5) of the main

Table 5: Training schedule. 'G_lr' and 'D_lr' represents the learning rate for generator and discriminator respectively.

Dataset	Iteration (k)	Batchsize	Image size	G_lr	D_lr
BFM	0-20	96	32	2e-5	2e-4
	20-40	48	64	2e-5	2e-4
	40-50	48	64	1e-5	1e-4
	50-80	16	128	2e-6	2e-5
CelebA	0-30	104	32	2e-5	2e-4
	30-70	64	64	2e-5	2e-4
	70-100	64	64	1e-5	1e-4
	100-150	16	128	2e-6	2e-5
Cats	0-20	128	64	6e-5	2e-4

paper, we use a simple manually tuned prior: $k_a \sim \mathcal{N}(0.6, 0.2)$, $k_d \sim \mathcal{N}(0.5, 0.2)$, $l_x \sim \mathcal{N}(0, 0.2)$, $l_y \sim \mathcal{N}(0.2, 0.05)$.

Table 6: Parameters of the lighting prior distributions.

Dataset	BFM				CelebA				Cats			
mean	0.058	0.141	-0.02	0.251	0.214	0.352	-0.006	0.389	-0.155	0.300	0.003	0.080
covariance	0.077	0.032	0	-0.002	0.081	0.004	0	-0.007	0.161	-0.008	0	0.006
	0.032	0.058	0	0.004	0.004	0.031	0	0.003	-0.008	0.043	0	0
	0	0	0.062	0	0	0	0.079	0	0	0	0.093	0
	-0.002	0.004	0	0.006	-0.007	0.003	0	0.006	0.006	0	0	0.009

In the volume rendering process, we sample m points within the near and far bounds t_n and t_f with a hierarchical sampling strategy as in NeRF [11]. For BFM and CelebA, $m = 24$, $t_n = 0.88$, and $t_f = 1.12$, while for Cats, $m = 40$, $t_n = 0.80$, and $t_f = 1.20$. In experiments with the surface tracking network, the ray sampling interval Δ_i and the number of sampling points m_i are gradually reduced as the training iteration i grows. We start to adopt this strategy after training for 5000 iterations. Specifically, we start with a large interval Δ_{max} and decrease to Δ_{min} with an exponential schedule as $\Delta_i = \Delta_{min} + \exp(-i\beta)(\Delta_{max} - \Delta_{min})$, where β controls the decay. m_i is defined similarly as $m_i = \lfloor m_{min} + \exp(-i\beta)(m_{max} - m_{min}) \rceil$. In our experiments, we set $\Delta_{max} = 0.24$, $\Delta_{min} = 0.06$, $m_{max} = m$, and $m_{min} = m/2$. β is set to 5.5e-5, 3e-5, and 1e-4 for BFM, CelebA, and Cats datasets respectively.

1.3 Datasets

Here we provide more information on the datasets used in our experiments.

BFM [12] is a synthetic human face dataset generated via the Basel face model. It consists of 160k images for training, 20k images for validation, and 20k images for testing. For each image, there is a corresponding ground-truth depth map. This dataset is released along with the code of Unsup3d [10, 13], which is protected under the MIT License.

CelebA [14] is a large-scale face attributes dataset with more than 200k images of human faces. According to the agreement on its official website [15], this dataset is available for non-commercial research purposes.

Cats dataset [16] contains 6444 images of cat heads. According to the official website [17], this dataset is available for research purposes.

All these datasets are obtained from their corresponding website mentioned above. These datasets do not contain personally identifiable information or offensive content, as they are ray images without identity labels.

1.4 GAN Inversion

With a pretrained ShadeGAN, we can perform GAN inversion for real image editing as shown in Fig. 9 of the main paper. Specifically, we revise the discriminator to be an encoder E by adding an additional fully-connected layer on top of the final feature. We randomly sample latent codes $\boldsymbol{z}$, camera poses $\boldsymbol{\xi}$, lighting conditions $\boldsymbol{\mu}$, and generate their corresponding images $\boldsymbol{I}_g$ via the generator. These data are used to train the encoder E to predict $\boldsymbol{\xi}$, $\boldsymbol{\mu}$, and the frequency and phase factors of the mapping network. The training of E takes 10k iterations with a learning rate of 2e-4. Given

a real target image, we first obtain the initial ξ, μ, frequency and phase factors via the trained encoder, and then finetune them by minimizing the image reconstruction error in the form of mean-squared-error.During finetuning, we jointly optimize ξ, μ, frequency and phase factors with the Adam optimizer for 400 iterations. The learning rate is initialized to 0.005, decaying by a half at 200, 300, and 350 iterations.

2 Qualitative Results

Figure 2: More qualitative examples of (a) pi-GAN [1] and (b) our ShadeGAN.

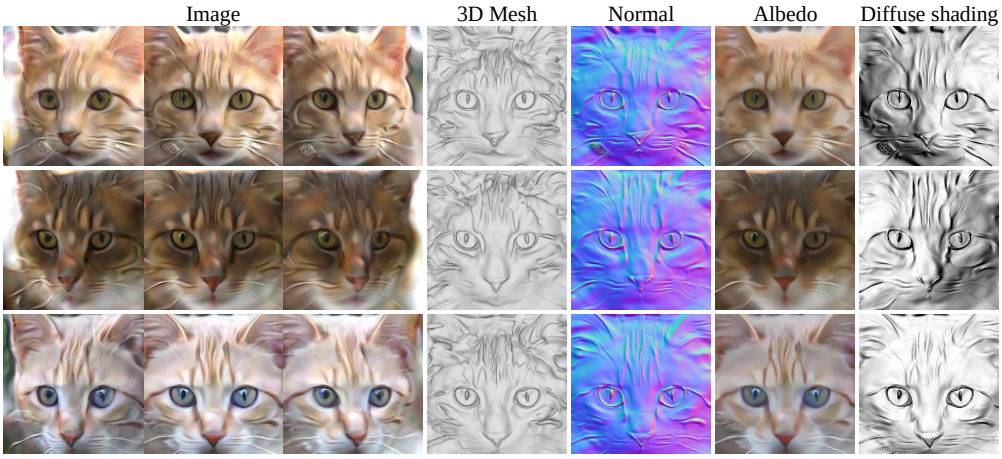
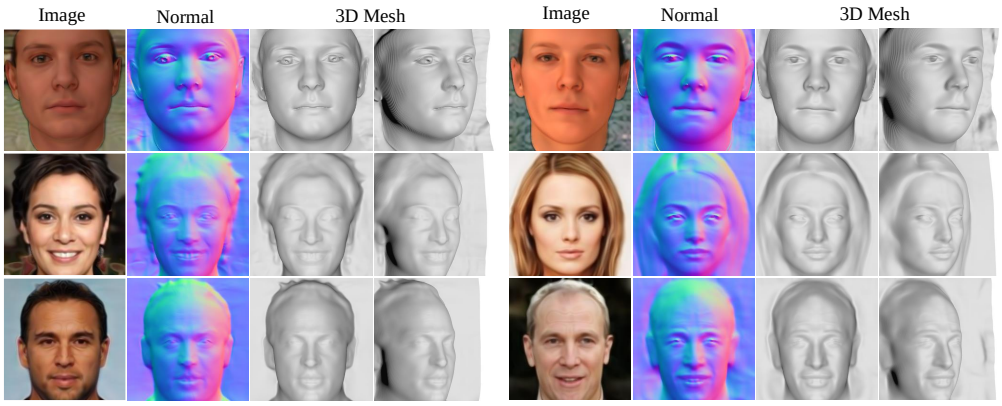


Figure 3: Qualitative results of ShadeGAN on the AFHQ dataset.



(a) with view conditioning (b) w/o view conditioning

Figure 4: Qualitative results of our method with and without view conditioning.

In this section, we provide more qualitative results of our method. In Fig. 2, we provide more qualitative comparisons between pi-GAN [1] and our ShadeGAN. We also show the results of our method on the AFHQ Cat dataset [18] in Fig. 3. Fig. 4 shows the effects of view conditioning in our model. The model without view conditioning looks slightly better in general, but would have slightly worse FID score as indicated in Tab. 3 of the main paper. Fig. 5 provides a qualitative comparison of the 3D shape reconstruction results corresponding to Tab. 1 of the main paper. Fig. 6 compares different ways of calculating normal directions, which shows that ours as described in Eq. 3 of the main paper is better. It is observed that our method predicts more accurate 3D shapes than GRAF [19] and pi-GAN [1]. Fig. 7 illustrates the effects of linearly interpolating the latent codes and lighting conditions, which creates a continuous image and shape morphing effect. We further show more examples on image relighting effects of ShadeGAN in Fig. 8.

3 Broader Impacts

This work provides a new approach for 3D-aware image synthesis. It could not only synthesize novel views of an instance in a 3D-consistent manner, but also generate the corresponding 3D shape that is more accurate than those produced by previous methods. The applications of our approach mainly lie in those related to 3D content creation, like augmented reality, virtual reality, and computer games. Besides, it could be used to edit 2D images, including those of human portraits. It could also possibly be extended for malicious use like deep fakes. Thus, any application or research that uses our approach has to strictly respect personality rights and privacy regulations.

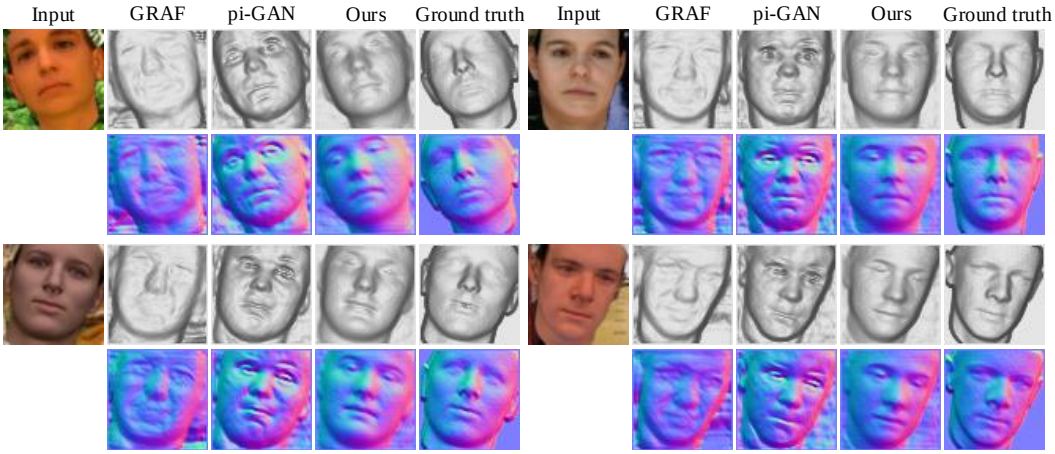


Figure 5: Qualitative results of 3D shape reconstruction corresponding to Tab. 1 of the main paper. We visualize the shapes and normals predicted by the depth-predicting CNNs. Our results are closer to the ground truth shapes than other baselines.

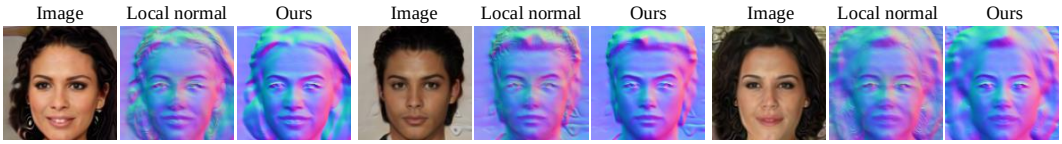


Figure 6: Comparisons of different normal direction calculation methods. 'Local normal' represents the alternative normal calculation method described in "Discussion" of Sec 3.2 in the main paper, which normalizes the normal direction at each local points on a ray. 'Ours' corresponds to Eq. 3 of the main paper. It can be observed that our results have less artifacts.

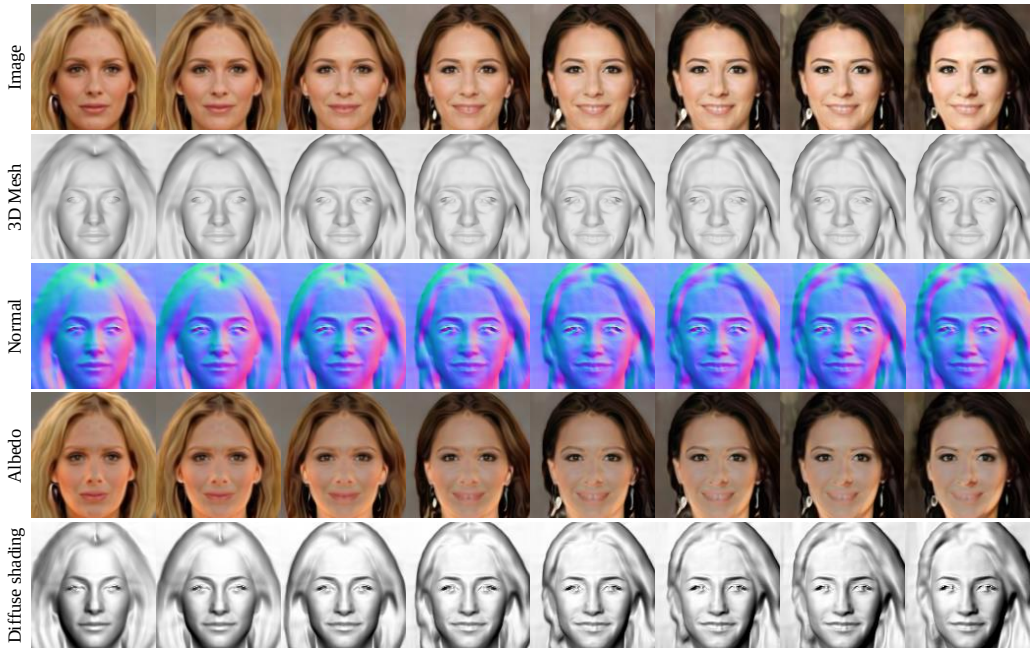


Figure 7: Linearly interpolating between two latent vectors and lighting conditions achieves continuous morphing on both images and shapes.

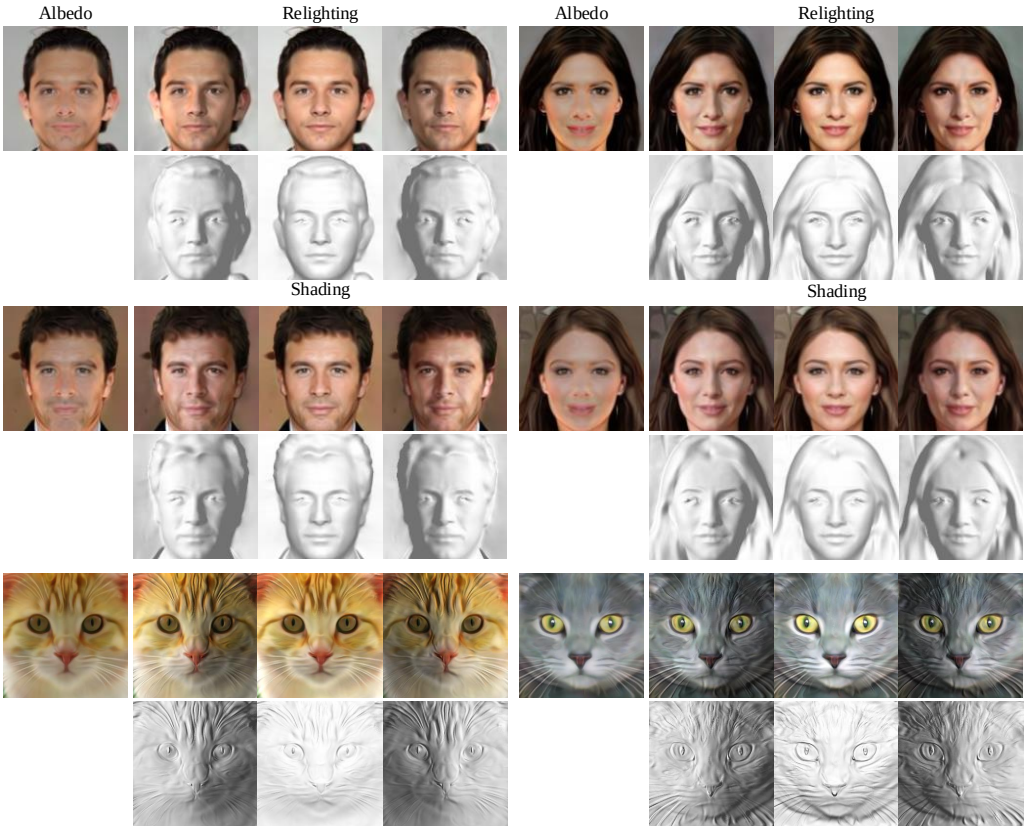


Figure 8: More image relighting examples produced by ShadeGAN.

References

[1] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis," in *CVPR*, 2021.

[2] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI*, 2018.

[3] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *NeurIPS*, 2020.

[4] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *CVPR*, 2019.

[5] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the coordconv solution," in *NeurIPS*, 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[7] Y. Wu and K. He, "Group normalization," in *ECCV*, 2018.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

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

[10] S. Wu, C. Rupprecht, and A. Vedaldi, "Unsupervised learning of probably symmetric deformable 3d objects from images in the wild," in *CVPR*, 2020.

[11] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[12] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3d face model for pose and illumination invariant face recognition," in *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, Ieee, 2009.

[13] "Unsup3d." `https://github.com/elliottwu/unsup3d`.

[14] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015.

[15] "Celeba." `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`.

[16] W. Zhang, J. Sun, and X. Tang, "Cat head detection-how to effectively exploit shape and texture features," in *ECCV*, 2008.

[17] "Cats." `https://web.archive.org/web/20150520175645/http://137.189.35.203/WebUI/CatDatabase/catData.html`.

[18] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "Stargan v2: Diverse image synthesis for multiple domains," in *CVPR*, 2020.

[19] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "Graf: Generative radiance fields for 3d-aware image synthesis," in *NeurIPS*, 2020.