3D-Aware Generative Model for Improved Side-View Image Synthesis

– Supplemental Document –

Kyungmin Jo[1,*] Wonjoon Jin[2,*] Jaegul Choo[1] Hyunjoon Lee[3] Sunghyun Cho[2]

[1]KAIST
Daejeon, Korea
{bttkm, jchoo}@kaist.ac.kr

[2]POSTECH
Pohang, Gyeongbuk, Korea
{jinwj1996, s.cho}@postech.ac.kr

[3]Kakao Brain
Seongnam-si, Gyeonggi-do, Korea
malfo.lee@kakaobrain.com

A. Overview

In this supplemental document, we provide additional explanations of our methods (Sec. B), experiment details (Sec. C), additional results (Sec. D), and an additional discussion on the limitations (Sec. E).

B. Details on SideGAN

Our model is built on top of EG3D [2] with the additional dual-branched discriminator and background network. For the components borrowed from EG3D, we used the official PyTorch implementation of EG3D, which can be obtained from https://github.com/NVlabs/eg3d. In the following, we describe the implementation details of the dual-branched discriminator and background network.

B.1. Dual-Branched Discriminator

Our proposed dual-branched discriminator is composed of a shared block D^{s}_{ϕ}, an image branch D^{i}_{ϕ}, a pose branch D^{p}_{ϕ} and a pose encoder E_{ϕ} (Figure 3 in the main paper). A shared block takes an image as input and outputs a feature map with the spatial resolution of 8×8, which is fed to an image branch and a pose branch. A pose encoder has eight fully-connected layers. It takes a camera parameter $\boldsymbol{\xi}$ as input and modulates the output features of a pose branch. A camera parameter is a 25-dimensional vector, which is composed of the elements of intrinsic and extrinsic camera matrices.

*Both authors contributed equally to this research. Also, this work was done during an internship at Kakao Brain.

B.2. Background Network

We adopt the background network of EpiGRAF [11] to synthesize feature maps for the background region separately from the foreground region. The background synthesis in SideGAN is performed as follows. To synthesize a 2D feature map describing the background region, the background network utilizes the inverse sphere parameterization as done in EpiGRAF, which is stemmed from NeRF++ [13]. Specifically, the background network takes a 3D position $\mathbf{x}$ as input as well as an input latent vector $\mathbf{z}_{bg}$, and synthesizes a volume density and a feature vector encoding the color information at the input 3D position $\mathbf{x}$. Then, to obtain a 2D feature map for the entire background region for a certain camera pose, we first sample a set of camera rays, and sample 3D positions from the rays. Then, for each sampled 3D position, a density and a feature vector are synthesized using the background network. Finally, we obtain a 2D feature map by volume rendering using the synthesized densities and feature vectors.

We implemented the background network using the official implementation of EpiGRAF [11] with a slight modification. Here, we describe only the modifications that we made in our implementation. First, while the original background network of EpiGRAF produces a 3-dimensional color vector for an input 3D position, ours is modified to produce a 32-dimensional feature vector. This modification is made for smoothly welding the background network with our image generator, which produces a 32-dimensional feature map for the foreground region. Second, while EpiGRAF samples 16 3D positions for each camera ray, ours samples 12 3D

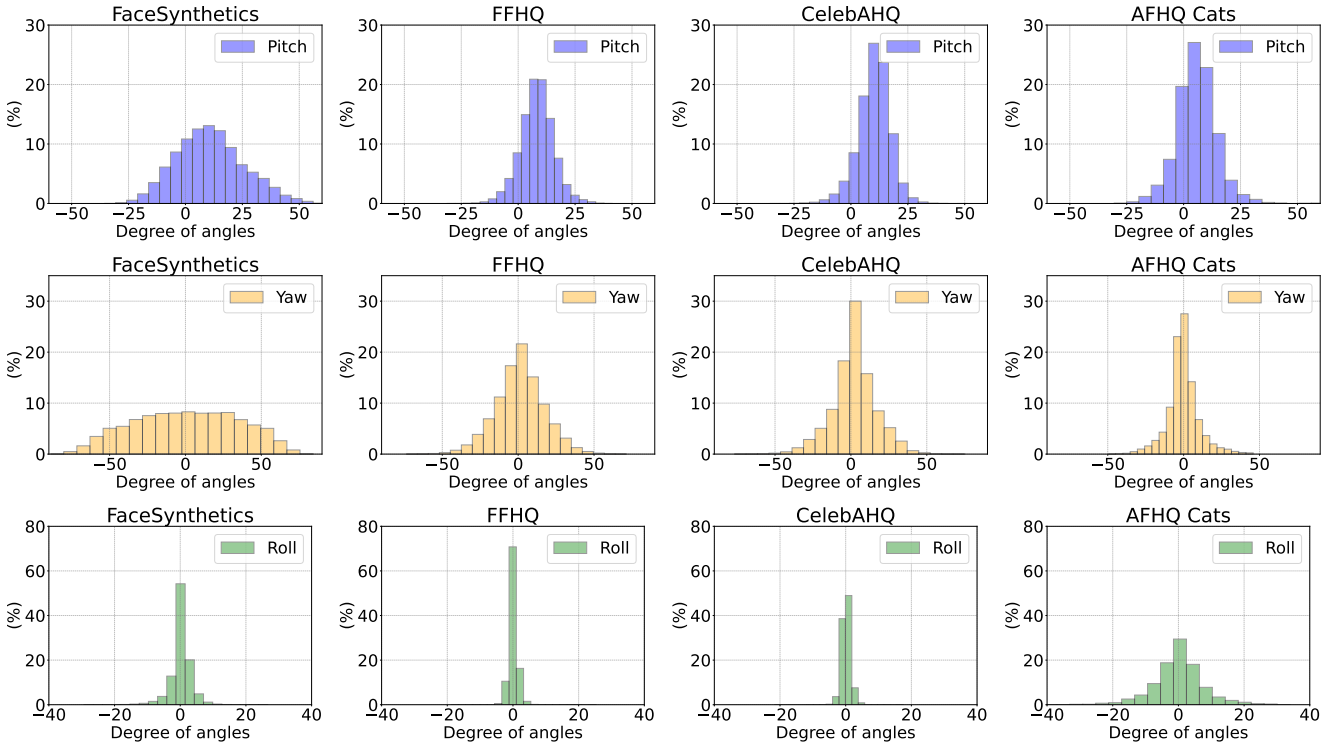


Figure 1. Histograms of camera poses of each dataset. From left to right, each column shows the distributions of each dataset: FaceSynthetics [12], FFHQ [9], CelebAHQ [8], and AFHQ Cats [4]. Also, each row indicates the distributions of pitch, yaw, and roll, sequentially. To be clear, 0° in each histogram corresponds to the frontal view.

positions due to the memory limit.

For the dataset where the background regions are already removed such as CelebAHQ [8], we turned off the background network in our experiments. Specifically, we construct our generator using only the image generator without the background network, and define the latent vector $\mathbf{z}$ as $\mathbf{z} = \mathbf{z}_{fg}$ instead of $\mathbf{z} = (\mathbf{z}_{fg}, \mathbf{z}_{bg})$.

B.3. Additional Uniform Pose Sampling

As mentioned in Section 4.3 in the main paper, we additionally sample camera parameters ξ from pre-defined distributions of three types of angles (pitch, roll, and yaw) for rendering fake images and learning the photo-realistic image synthesis for side-view images. We note that the additionally sampled camera poses are only used for the image branch D^{i}_{ϕ} of the discriminator during training. The pre-defined distributions are as follows. The values of the distribution are expressed in degrees and the zero value indicates the frontal view. Pitch and roll angles are sampled from $\mathcal{N}(0°, 14.90^2)$ and $\mathcal{N}(0°, 3.09^2)$, respectively, by referring to the camera distribution of FFHQ [8], a real human face dataset. On the other hand, yaw angles are sampled from the uniform distribution within $-120°$ to $120°$, which is wider than the ranges of the yaw angles of both real and synthetic face datasets as shown in Fig. 1. We empirically found that the sampling range of $[-120°, 120°]$ produces higher-quality results.

C. Details on the Experiments

C.1. Datasets

In this subsection, we describe the number of images, and the camera pose distribution of the datasets used for training. We use off-the-shelf pose estimation algorithms [6, 10] to obtain pseudo-ground-truth pose labels. We excluded images with incorrectly estimated poses from the training of our models. Subsequently, we constructed our training datasets with correctly estimated poses using about 69K, 29K, and 5K images from the FFHQ [9], CelebAHQ [8], and AFHQ Cats [4] datasets, respectively. Additionally, for the experiment with transfer learning as mentioned in Section 5 in the main paper, we also constructed the dataset using about 100K images from the FaceSynthetics [12] dataset.

Fig. 1 visualizes the distributions of the camera poses (yaw, roll, and pitch angles) in the real-world and

synthetic datasets (FFHQ [9], CelebAHQ [8], AFHQ Cats [4] and FaceSynthetics [12]). Among the three angles, the yaw angle corresponds to the horizontal rotation of a face, which SideGAN aims to address. As shown in the figure, all the real-world datasets have narrow distributions centered around 0° for all three angles, indicating that most of their images are frontal-view images. On the other hand, the FaceSynthetics dataset shows a wider distribution for the yaw angle ranging from -75° to 75°, indicating that it has a relatively large number of side-view images.

C.2. Transfer Learning

In this subsection, we describe details on the additional evaluation setting with transfer learning (Section 5 in the main paper). As mentioned, to evaluate the performance of SideGAN under the setting with sufficient knowledge for side-view images, we use the training strategy of EG3D [2] to pre-train all models on FaceSynthetics [12] and fine-tune the models on in-the-wild datasets.

In the fine-tuning, we adopt the freezeG scheme [1] to retain the knowledge learned from a synthetic dataset in the pre-training stage. Specifically, in the fine-tuning stage, we fix the weights of the mapping network and the first three convolutional blocks in the image generator while updating the weights of the other layers.

C.3. Training Iterations

We train our model on the CelebAHQ [8] and FFHQ [9] datasets using 25M images per GPU (3.906M iterations) from scratch. For the experiments with transfer learning, the number of images used in the pre-training stage is 25M per GPU (3.906M iterations) for FaceSynthetics [12] and the number of images used in the fine-tuning stage is 10M per GPU (1.563M iterations) for FFHQ, CelebAHQ, and AFHQ Cats [4]. To be clear, we use the same setting for training EG3D for a fair comparison.

All the experiments are performed utilizing four NVIDIA A100 GPUs. The training takes 4.5-5.5 days to learn from scratch and additional 2.5 days for transfer learning.

C.4. Comparison

In this subsection, we describe details on the quantitative comparison (Table 1 in the main paper) of SideGAN against π-GAN [3] and EG3D [2] such as the number of generated images and how we sampled the camera poses for rendering images for evaluation. In order to evaluate the image and shape qualities of 3D GANs for a wide range of viewing angles, we synthesize images with camera poses from the frontal to side viewpoints. The camera poses are sampled from the same distribution for all three algorithms for a fair comparison.

Image Quality. We randomly generate 50K images at randomly sampled camera poses to evaluate the image fidelity of 3D GANs based on the FID score [7]. The pitch and roll angles are sampled from Gaussian distributions $\mathcal{N}(0^\circ, 14.90^2)$ and $\mathcal{N}(0^\circ, 3.09^2)$, respectively. These distributions are obtained from the pose distributions of FFHQ [9], which has the largest number of images among the real face datasets. To evaluate the quality of images from the frontal to the side view, we randomly sample yaw angles from the uniform distribution $\mathcal{U}(-90^\circ, 90^\circ)$.

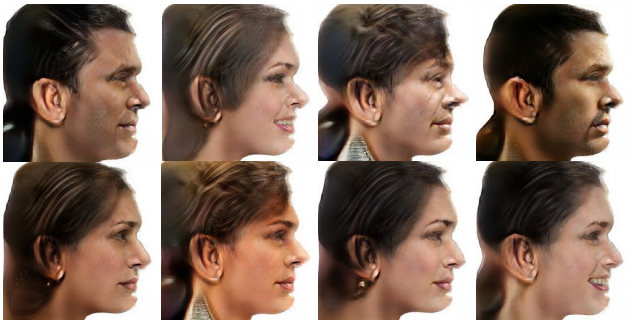
Shape Quality. We randomly generate 1024 images at randomly sampled camera poses to evaluate the shape quality of 3D GANs based on the depth error. We use the same yaw and pitch distribution used for the image fidelity evaluation. On the other hand, we set the roll angles to zero since the roll distribution obtained from the real face dataset has a standard deviation close to zero.

C.5. Analysis with respect to the Steep and Extrapolated Angles

Figure 7 in the main paper provides a comparison between SideGAN and EG3D [2] on the image quality with respect to camera poses. In the comparison, we compare the FID scores of our model and EG3D for three different cases of camera poses (near-frontal angles, steep angles, and extrapolated angles) using the FaceSynthetics dataset [12]. To this end, we generate three evaluation datasets to measure FID scores using images from the FaceSynthetics dataset. Specifically, for the case of 'near-frontal angles', we sampled images of yaw angles within $[-30^\circ, 30^\circ]$ from the FaceSynthetics dataset. For the case of 'steep angles', we sampled images of yaw angles within $[-50^\circ, -30^\circ]$ and $[30^\circ, 50^\circ]$, and for the case of 'extrapolated angles', we sampled images of yaw angles within $[-90^\circ, -50^\circ]$ and $[50^\circ, 90^\circ]$. In consequence, we obtained three evaluation datasets of 110,870, 56,578, and 32,552 images, respectively, and used them to measure the FID scores.

C.6. Training Baselines

We train the baseline models: π-GAN [3] on the CelebAHQ [8] and FFHQ [9] datasets, and EG3D [2] on the CelebAHQ [8], FFHQ [9] and AFHQ Cats [4] datasets. For each combination of the baseline models and datasets, we save checkpoints during training and



(a) CelebAHQ [8]

(b) FFHQ [9]

Figure 2. Additional synthesized images of our method at the side-view camera pose. In this figure, our models are trained without transfer learning.

use the best checkpoint with the lowest FID score [7] in our evaluations.

For training π-GAN, we use the experimental setups from the official implementation, which can be found from https://github.com/marcoamonteiro/pi-GAN. To be specific, we use the 'CelebA' setup provided in the official implementation of pi-GAN for training the model on the CelebAHQ and FFHQ datasets. For all experiments, the image resolution is set to 256×256 for a fair comparison with our model. Also, we utilize three stages of progressive learning with the following settings. The batch size is set to 18, 8, and 4 for each stage, respectively. Also, the image resolution is doubled at each step starting from 64×64. The number of points per each ray is fixed at 12 for all stages. The learning rate for the generator is set to 4×10^{-5}, 2×10^{-5}, and 1×10^{-5} for each stage, respectively. Also, the learning rate for the discriminator is set to 4×10^{-4}, 2×10^{-4}, and 1×10^{-4} for each stage, respectively. In addition, the number of images used for each stage is 10K, 55K, and 200K, respectively.

As mentioned in Section 5 in the main paper, we adopt the settings from EG3D to train our model except for a few things like image resolution, so we use the same settings to train EG3D for a fair comparison.

(a) CelebAHQ [8]

(b) FFHQ [9]

(c) AFHQ Cats [4]

Figure 3. Additional synthesized images of our method at the side-view camera pose. In this figure, our models are trained with transfer learning.

	Cosine similarity ↑		
	EG3D	SideGAN w/o $\mathcal{L}_{\text{id}}$	SideGAN
w/o transfer learning	0.664	0.726	0.836
w transfer learning	0.713	0.778	0.850

Table 1. Evaluation on multi-view consistency using ArcFace [5] cosine similarity. SideGAN shows better results of multi-view consistency than EG3D [2].

D. Additional Results

In this section, we present an evaluation on the multi-view consistency (Sec. D.1), additional side-view examples of our method (Sec. D.2), and qualitative comparisons of SideGAN with other baseline methods (Sec. D.3): π-GAN [3] and EG3D [2] on the CelebAHQ [8], FFHQ [9] and AFHQ Cats [4] datasets.

D.1. Evaluation on Multi-view Consistency

As done in EG3D, we measure ArcFace [5] cosine similarity of EG3D [2] and SideGAN trained on CelebAHQ [8] in order to assess the multi-view consistency of our method (Tab. 1). Note that we also evaluate our model without $\mathcal{L}_{\text{id}}$ for fair comparison since we used ArcFace for $\mathcal{L}_{\text{id}}$ in training (SideGAN w/o $\mathcal{L}_{\text{id}}$ in the table). As seen in the table, SideGAN outperforms the baseline regardless of the transfer learning.

D.2. Additional Side-view Examples

Fig. 2 and Fig. 3 show additional images synthesized by our method at the side viewpoint in each experimental setting with and without transfer learning, respectively, showing realistic synthesis results with diverse styles such as skin color and hairstyle.

D.3. Additional Qualitative Comparisons

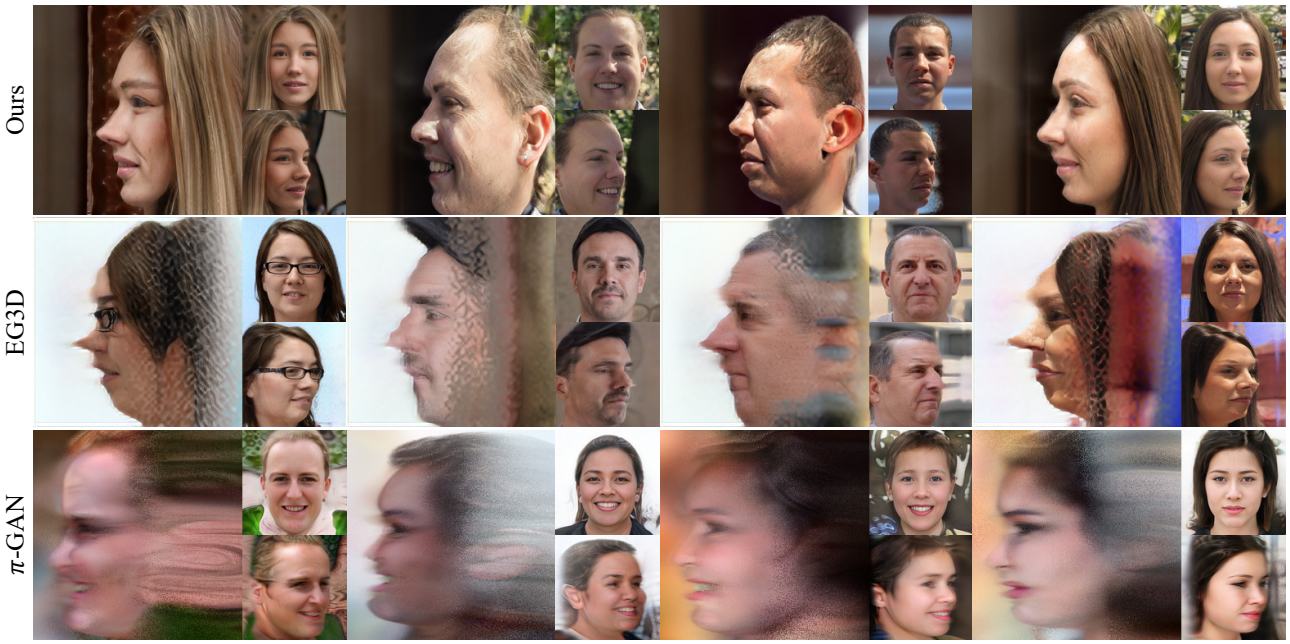
In order to demonstrate the superiority of our model in photo-realistic image synthesis covering a wide range of camera poses, we exhibit the generated images with variable setups of the camera poses. Fig. 4 presents synthesized multi-view images on the CelebAHQ and FFHQ datasets at the side, frontal and steep viewpoints, highlighting the clear images of our method. For all datasets, SideGAN shows more realistic images especially at the side viewpoint, compared to the other baselines that synthesize blurry images with noisy facial boundaries. Furthermore, SideGAN is effective in synthesizing realistic images at steep vertical viewing angles. Fig. 5 presents multi-view images at the viewpoints from the above, middle, and below, showing that SideGAN synthesizes clearer images than the other baselines thanks to our training methods.

Additionally, in order to demonstrate the superiority of our method under the setting with sufficient knowledge at side-viewing angles, we compare the generated images and 3D geometries between EG3D and SideGAN using the models trained with transfer learning. Fig. 6 presents multi-view images with wide yaw angles from the frontal to the steep viewpoints and underlying geometries of the side-view images, showing stable image quality from various viewing angles. For all datasets, SideGAN synthesizes photo-realistic images on a wide range of camera poses, due to the noisy-free underlying geometry, especially at the side viewpoint. On the contrary, the other baseline synthesizes blurry images and low-fidelity geometry at the side viewpoint.

E. Additional Discussion on Limitations

Our framework may sometimes generate repetitive patterns in the background region in synthesized images as shown in Fig. 3(c). This is possibly due to the small network size of the background network. In our work, we render the foreground and the background using the image generator and the background network, respectively. While this separation improves the quality of synthesized images at the side view as shown in Fig. 6, the background network burdens the memory to train the model, so we utilize a small network architecture for the background network, which may sometimes suffer from unnatural results. We expect that additional model capacity may lessen this phenomenon, which will be complemented in future work.

(a) CelebAHQ [8]



(b) FFHQ [9]

Figure 4. Multi-view consistent image synthesis at the diverse viewing angles. All models are trained without transfer learning. SideGAN outperforms the other baselines on CelebAHQ [8] and FFHQ [9], especially at the side viewpoint.

Figure 5. Multi-view consistent image synthesis at diverse pitch angles. All models are trained without transfer learning. SideGAN synthesizes higher-quality images with clear facial boundaries than the other baselines on CelebAHQ [8] and FFHQ [9] at the wide vertical viewing angles from the above to the below camera poses.

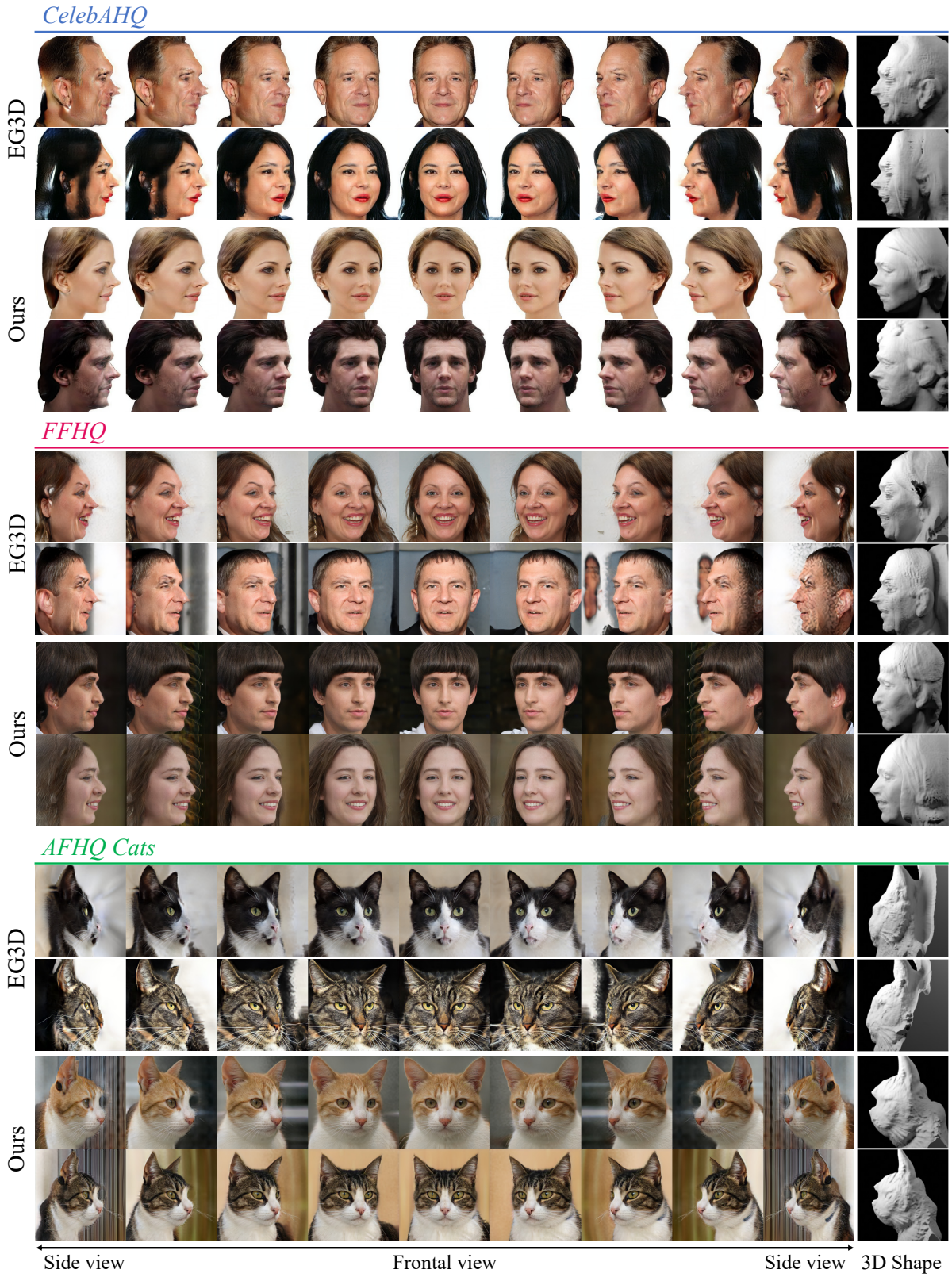


Figure 6. Multi-view consistent image synthesis at diverse yaw angles. All models are trained with transfer learning. SideGAN synthesizes higher-quality images with clear facial boundaries than EG3D [2] on CelebAHQ [8], FFHQ [9] and AFHQ Cats [4] at the wide horizontal viewpoints from the frontal to the side camera poses, especially at the side viewpoint.

References

[1] Freezeg. https://github.com/bryandlee/FreezeG. Accessed: 2022-11-08. 3

[2] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proc. of IEEE conference on computer vision and pattern recognition (CVPR)*, pages 16123–16133, 2022. 1, 3, 4, 5, 8

[3] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proc. of IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5799–5809, 2021. 3, 4

[4] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proc. of IEEE conference on computer vision and pattern recognition (CVPR)*, pages 8188–8197, 2020. 2, 3, 4, 8

[5] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proc. of IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4690–4699, 2019. 4, 5

[6] Yao Feng, Haiwen Feng, Michael J Black, and Timo Bolkart. Learning an animatable detailed 3d face model from in-the-wild images. *ACM Transactions on Graphics (ToG)*, 40(4):1–13, 2021. 2

[7] 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. *Proc. of the Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017. 3, 4

[8] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 2, 3, 4, 5, 6, 7, 8

[9] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. of IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4401–4410, 2019. 2, 3, 4, 6, 7, 8

[10] Taehee Brad Lee. Cat hipsterizer. https://github.com/kairess/cat_hipsterizer, 2018. Accessed: 2022-11-08. 2

[11] Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. Epigraf: Rethinking training of 3d gans. *arXiv preprint arXiv:2206.10535*, 2022. 1

[12] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Matthew Johnson, Virginia Estellers, Thomas J. Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone, 2021. 2, 3

[13] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 1