What You See is What You GAN: Rendering Every Pixel for High-Fidelity Geometry in 3D GANs

Supplementary Material

In this supplement, we first provide additional visual results (Sec. A1) and additional evaluations (Sec. A2). We follow with details of our implementation (Sec. A3) including the details of our adaptive sampling approach (Subsec. A3.4). We discuss experiment details (Sec. A4) such as the details of our Normal-FID evaluation metric and baselines. We finally provide discussion (Sec. A5) including limitations of our work that may be addressed in future work. Please refer to the accompanying video, which contains additional visual results and comparisons.

A1. Additional Qualitative Results

We first show both curated and uncurated results for both our FFHQ and AFHQ models. Curated FFHQ results can be seen in Fig. A1. Please note the highly-detailed and variable facial expressions along with well-defined 3D accessories like hats and glasses. Long hair is not pasted onto the foreground, but rather retains a 3D aspect. Curated AFHQ results can be seen in Fig. A2. Note the detailed textures of the cat geometry and the well-defined noses and ears. For unbiased presentation, we also show uncurated results (the first 8 seeds) for FFHQ (Fig. A3) and AFHQ (Fig. A4). All results shown are with Truncation $= 0.7$.

A2. Additional Evaluations

A2.1. Comparing Sampling Methods at Various Sampling Counts

In this section, we show the robustness of our proposed sampling strategy from the predicted $\hat{P}_{512}$ at very low sample counts, in comparison to unstratified and stratified sampling methods. In Fig. A5, we show our proposed *robust* sampling method in comparison to unstratified and stratified inverse transform sampling. At very low samples per pixel (spp), our method vastly out performs the standard sampling technique. Please see the insets where our method can handle depth discontinuities without jagged artifacts even at 8 samples per pixel.

We also render a pseudo ground truth image using 384 (192 coarse and 192 importance) samples and compare the PSNR of various sampling methods in Fig. A6. Most importantly for GAN training, our method's *worst-case* is significantly better than previous method's *worst-case*. This is integral to GAN training, where the discriminator will always focus on the easiest attribute to discriminate. As sampling artifacts cannot be amended by G, the worst-case results dictate how well the GAN converges.

Method	FID (20)↓	FID (50)↓	FID (96)↓
Mimic3D	53.57	13.31	5.37
EG3D	193.75	36.82	**4.70**
Ours	**5.28**	**4.97**	4.97

Table A1. FID comparison on FFHQ using various sample counts. The samples per pixel are given in the parentheses of the metric.

A2.2. Effectiveness of Adaptive Sampling

Fig. A7 demonstrates the effectiveness of our proposed adaptive sampling method compared to other baselines. By allocating a small portion of samples to uncertain regions (e.g., depth discontinuity; see the top right of Fig. A7), our method can generate an artifact-free result even at the depth count budget of 10 samples per pixel (10spp) compared to the same spp without adative sampling (top left), which has jaggy artifacts around the depth discontinuity. Without our sampler (bottom row of Fig. A7), the standard two-pass importance sampler [11] results in significant artifacts. Please see Subsec. A3.4 for the implementation details of our adaptive sampling.

A2.3. Single Image Reconstruction

We additionally showcase an application of our method for single-view 3D reconstruction in Fig. A8. The learned prior enables high quality reconstruction of images and 3D geometry, despite the under constrained nature of the problem. We incorporate Pivotal Tuning Inversion (PTI) [13], optimizing the latent code, camera, and noise buffers for 600 iterations, followed by optimization of the camera and generator weights for another 350 iterations with MSE and LPIPS [18] losses computed between the input view and rendering.

A2.4. Benchmarking against baselines at lower spp

We also compare other methods' ability to render with low sample count in Tab. A1. With just 20 samples, the rendering quality of our method drops by only .3 in FID, compared to the precipitous drop for other methods. This validates our strategy to jointly learn a sampler and ensure tight SDF surface for operation under a limited sample budget.

A3. Implementations Details

A3.1. Inference Details and Time

During inference, our method, by default, uses 17.6 depth samples (see Subsec. A3.4 for details) at high resolutions in addition to samples from the low-resolution probe, which is

equivalent to 12 samples at high resolutions; this results in 29.6 samples per ray at high resolutions. While our model learns view-dependent effects during training (see Eq. 5 in the main PDF), we use a constant frontal-viewing condition during inference for our qualitative results. Rendering from cached triplanes runs at 4.5 FPS using plain PyTorch scripts on a single A100 GPU and requires <15GB of VRAM.

A3.2. Training Details

In this section, we present the details of the training of our proposed model. For the schedule of the hyperparameters of the FFHQ model, please see Tab. A4. We train the FFHQ model for 28.2 million images with a batch size of 32 images on 16 80GB NVIDIA A100 GPUs, which takes about 11 days. The AFHQ model is finetuned from this model with adaptive discriminator augmentation [7] for 1.2 million images and R1 gamma value of 6, and all other hyperparameters the same as in the end of the FFHQ training.

A3.3. Network Details

The architecture of the generator for T' follows EG3D [2] exactly, except doubling its capacity (channel base from 32768 to 65536). As mentioned in the main paper, we add three extra Synthesis Blocks from StyleGAN2 [8] applied to the channels of T', in order to get the final triplane T– one for each of the orthogonal planes and each applied to one of the three slices of 32 channels of T'.

For the details of the architecture of the proposal network, please refer to Tab. A5. Slightly abusing notation, we have labelled the image of viewing directions corresponding to the target camera as ϕ_{128}, parameterized as normalized vectors per pixel. The other inputs, P_{128} (weights) and I_{128} (image), follow the same notation as the main paper.

For MLP_{SDF}, we embed the input 3D positions with the embedding from NeRF [11] and 6 frequencies (sampled in logspace). The architecture of this network is given in Table A2. The first two components of the 66 dimensional output correspond to the SDF value s and pre-activated β_{pre}. We map to the variance with the following equation: $\beta = 0.01 + \text{Tanh}(2 \cdot \beta_{\text{pre}})(0.01 - 0.0001)$. This activation ensures that the variances are approximately 0.01 at the beginning of training and prevents them from becoming negative.

We also show the details of MLP_c in Table A3. The positional encoding of the viewing direction is 2 frequencies sampled in logspace. We finally map the 3 output components c_{pre} to the RGB value as $\text{Sigmoid} \cdot (c_{\text{pre}})(1 + 0.002) - 0.001$.

A3.4. Details of Adaptive Sampling

As mentioned in the main paper, we use a proxy for the variance to adaptively allocate more samples to more difficult pixels. Specifically, considering the predicted high-resolution distributions $\hat{P}_{512}$, for each distribution, we compute a scalar value to dictate how many samples to allocate. We operate under the simplified assumption that we will allocate 16 samples to 90% of pixels, and 32 to the remaining 10%, resulting in total 17.6 samples per ray. To compute *which* pixels receive more samples, we compute a proxy for the variance.

To do so, for a given predicted distribution p, we compute the leftover probability mass after removing the largest 16 bins. Precisely, we consider the set of non-repeating non-negative integers less than 192,

$$S = \{(z_1, \ldots, z_{16}) \mid z_i \in \mathbb{Z}_{\geq 0}, z_i < 192, z_i \neq z_j \; \forall \, i \neq j\}.$$

We then find

$$S_{\text{max}} = \max_{(z_1,\ldots,z_{16}) \in S} \sum_{i=1}^{16} p_{z_i}.$$

The final scalar $1 - S_{\text{max}}$ is the leftover probability mass after removing the largest 16 bins. We choose the 10% of pixels from $\hat{P}_{512}$ which have maximized this quantity. A visualization of these pixels is given in Fig. A7. We can see that they are most concentrated on the depth discontinuities where the distributions may not be unimodal. Adaptively allocating samples in this manner allows us to accurately render the most challenging pixels without wasting too many samples on the "easier" distributions. As can be seen in Fig. A7, using the same total sample count, we can avoid jagged and inaccurate renders. For illustration purposes, we first show an example where all pixels are rendered with 10 samples (top-left of Fig. A7) and then with 9 samples for 90% of pixels, and 19 samples for the remaining 10%, resulting in an average sample count of 10 (top-middle of Fig. A7). The samples to which we allocate more samples are visualized (top-right of Fig. A7). We compare to varying sample counts with standard sampling in the bottom row of Fig. A7.

A3.5. Details of Stratified Sampling

As discussed in subsection 4.4, we compute the robust distribution q from the predicted distribution $\hat{p}$ from the proposal network. Let $I = \{i \in \mathbb{Z} : q_i > 0\}$ denote the set of non-zero bins each with equal probability (as in the main paper). For stratified sampling, we partition the unit interval into $c = |I|$ strata. We assume we are given a sample budget $s > 1$. We allocate $\lfloor \frac{s}{c} \rfloor$ samples to each of the c strata. We then allocate one extra sample to the $(s \bmod c)$ bins with maximal $\hat{p}_i$. Note that as we allocate more samples to a particular stratum, the distance δ_i between adjacent intrastratum samples shrinks, thus introducing no additional bias. In practice, we also clip the δ_i to the bin width to prevent outsized contributions from the endpoints of nonzero regions. For $s < c$, this is biased; however, in practice, due to our tightening regularization and adaptive allocation of samples, we almost always have $s \geq c$. Additionally, at extremely low spp, our method outperforms unbiased methods (see left column of Fig. A5).

Layer	Type	Input	Activation	Dimension
Input 0	Input	XYZ positions	-	3
Input 1	Input	Triplane features	-	32
1	PosEnc	Input 0	-	39
2	Concatenation	Input 1, Layer 1	-	71
3	Linear	Concatenation	Softplus	128
4	Linear	Layer 3	Softplus	128
5	Linear	Layer 4	-	66

Table A2. Architecture of the MLP_{SDF} network with embedding.

Layer	Type	Input	Activation	Dimension
Input 0	Input	Viewing Directions ϕ	-	3
Input 1	Input	f_{geo}	-	64
1	Embedding	Input 0	-	15
2	Concatenation	Input 1, Layer 1	-	79
3	Linear	Layer 2	Softplus	64
4	Linear	Layer 3	-	3

Table A3. Architecture of the MLP_c network with embedding for viewing direction.

A4. Experiment details

A4.1. Geometry Visualization

For geometry visualization, we extract iso-surface geometry using March Cubes [10]. We use the voxel resolution of 512^3 for comparisons and 1024^3 for our main results. For SDF-based methods (ours), geometry is extracted from an SDF field at the 0th level set. For NeRF-based methods (EG3D, Mimic3D, and Epigraf), the surface is extracted from the density field using the level set provided by the official script from the authors. We render these extracted models using Blender for visualization. To visualize normal maps, we derive the normal by taking the gradient of the SDF field for SDF-based methods and density field for NeRF-based methods with respect to positions.

A4.2. Normal-FID

As mentioned in the main text, we use normal maps extracted from the meshes of the NPHM [6] dataset. 255 subjects are scanned with highly variable expressions. We provide examples of these normal maps in Fig. A9. For Normal-FID computation, we ensure all coordinate conventions are consistent between baselines so that the color maps are likewise consistent. We sample all methods with truncation of 0.7 (cutoff $= 14$) due to the lack of diversity in the ground truth images (see Fig. A9). Using PyFacer [4], we also mask the background pixels to black. For EpiGRAF [14], we crop all sample images using HRN [9].

A4.3. Details of baseline methods

For all the baselines, we use publicly released pre-trained models if they are available; otherwise, we quote the FID numbers from previous work. For Mimic3D [3], Epigraf [14] and EG3D [2], we used the corresponding publicly released models from the original authors for N-FID and non-flatness score computation; unless explicitly specified otherwise, we also use the provided default evaluation options for all methods. For EG3D, Mimic3d, Epigraf this is 48 samples for a coarse pass and 48 samples for a fine pass for two-pass importance sampling. For StyleSDF this is 24 samples per ray.

For StyleSDF [12], we re-trained an FFHQ model at 512 resolutions and AFHQv2 cats-only model at 512 resolutions as they were not publicly available and used them for geometry evaluations. We train our StyleSDF geometry network on FFHQ using the publicly released code, for 200k iterations as recommended by the authors, on 8 A100 NVIDIA GPUs. For our StyleSDF geometry network on the AFHQv2 cats-only split, training with the provided AFHQ config from scratch was unstable and collapsed. Instead, we finetune the publicly released StyleSDF AFHQv2 all-animals geometry network on our cats-only split for 50k iterations and use that for evaluation.

A5. Discussion

A5.1. Limitation and Future Work

We showcase three failure cases of our method in Fig. A10. In the first row, we see that there are seams in the side of the face in both the geometry and rendering. We hypothe-

Hyperparameter	Number of Images (in millions)	Value
R1 Gamma	0-18	1
	18-25	4
	25 onwards	2
Neural Rendering Resolution	0-10	64
	10-18	128 (linearly increased over 1m images)
	18 onwards	512 (linearly increased over 0.2m images)
β_{target}	0-10	0.01
	10 onwards	0.001 (linearly decreased over 1m images)
Learning Rate Multiplier for MLP_c	0-25	2
	25 onwards	1
Render with Predicted Distributions	0-17	No
	17 onwards	Yes
Supervise Predicted Distributions	0-16	No
	16 onwards	Yes

Table A4. Schedule of hyperparameters given in millions of images the discriminator has seen during training. We train for 28.2m images total.

Layer	**Type**	**Activation**	**Upsample**	**Input Source(s)**	**Dimension**
Input 0	Input	-	-	P_{128}	191 x 128 x 128
Input 1	Input	-	-	I_{128}	3 x 128 x 128
Input 2	Input	-	-	ϕ_{128}	3 x 128 x 128
1	Concatenation	-	-	Inputs 0-2	197 x 128 x 128
2	Conv2D	ReLU	No	Layer 1	256 x 128 x 128
3	Conv2D	ReLU	No	Layer 2	256 x 128 x 128
4	Conv2D	ReLU	No	Layer 3	256 x 128 x 128
5	Conv2D	ReLU	No	Layer 4	256 x 128 x 128
6	Conv2D	ReLU	Yes	Layer 5	256 x 256 x 256
7	Conv2D	ReLU	No	Layer 6	256 x 256 x 256
8	Conv2D	ReLU	No	Layer 7	256 x 256 x 256
9	Conv2D	ReLU	Yes	Layer 8	256 x 512 x 512
10	Conv2D	None	No	Layer 9	191 x 512 x 512
11	BilinearUpsample	-	Yes	Input 0	191 x 512 x 512
12	Conv2D	ReLU	No	Layer 10 + Layer 11	256 x 512 x 512
13	Conv2D	ReLU	No	Layer 12	256 x 512 x 512
14	Conv2D	ReLU	No	Layer 13	256 x 512 x 512
15	Conv2D	Softmax	No	Layer 14	191 x 512 x 512

Table A5. Architecture of our proposal network.

size this issue may be related to the frontal camera bias in FFHQ and may be ameliorated by a more uniform sampling of cameras. In the second row, we see that in some samples with large amounts of specularity, the surface may become unnaturally rough, which may be remedied by additional regularization on the surface normal [17]. Finally, in the third row, we see a rare phenomena where density close to the camera occludes the subject.

Future works may utilize more balanced datasets with larger coverage around the entirety of the face [1]. Extending to the human body [5] or more general classes [15], is also extremely interesting. Combining our approach with a 3D lifting approach [16] using our method as 3D synthetic data, may allow high-fidelity geometry estimation from a single image.

References

[1] Sizhe An, Hongyi Xu, Yichun Shi, Guoxian Song, Umit Y. Ogras, and Linjie Luo. Panohead: Geometry-aware 3d full-head synthesis in 360deg. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4

[2] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3

[3] Xingyu Chen, Yu Deng, and Baoyuan Wang. Mimic3d: Thriving 3d-aware gans via 3d-to-2d imitation. *arXiv preprint arXiv:2303.09036*, 2023. 3

[4] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5203–5212, 2020. 3, 14

[5] Zijian Dong, Xu Chen, Jinlong Yang, Michael J Black, Otmar Hilliges, and Andreas Geiger. Ag3d: Learning to generate 3d avatars from 2d image collections. *arXiv preprint arXiv:2305.02312*, 2023. 4

[6] Simon Giebenhain, Tobias Kirschstein, Markos Georgopoulos, Martin Rünz, Lourdes Agapito, and Matthias Nießner. Learning neural parametric head models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21003–21012, 2023. 3, 14

[7] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[8] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[9] Biwen Lei, Jianqiang Ren, Mengyang Feng, Miaomiao Cui, and Xuansong Xie. A hierarchical representation network for accurate and detailed face reconstruction from in-the-wild images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 394–403, 2023. 3

[10] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Transactions on Graphics (ToG)*, 1987. 3

[11] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2

[12] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[13] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *arXiv preprint arXiv:2106.05744*, 2021. 1

[14] Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. Epigraf: Rethinking training of 3d gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[15] Ivan Skorokhodov, Aliaksandr Siarohin, Yinghao Xu, Jian Ren, Hsin-Ying Lee, Peter Wonka, and Sergey Tulyakov. 3d generation on imagenet. *arXiv preprint arXiv:2303.01416*, 2023. 4

[16] Alex Trevithick, Matthew Chan, Michael Stengel, Eric R. Chan, Chao Liu, Zhiding Yu, Sameh Khamis, Manmohan Chandraker, Ravi Ramamoorthi, and Koki Nagano. Real-time radiance fields for single-image portrait view synthesis. In *ACM Transactions on Graphics (SIGGRAPH)*, 2023. 4

[17] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 4

[18] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

Figure A1. Curated samples from our FFHQ model.

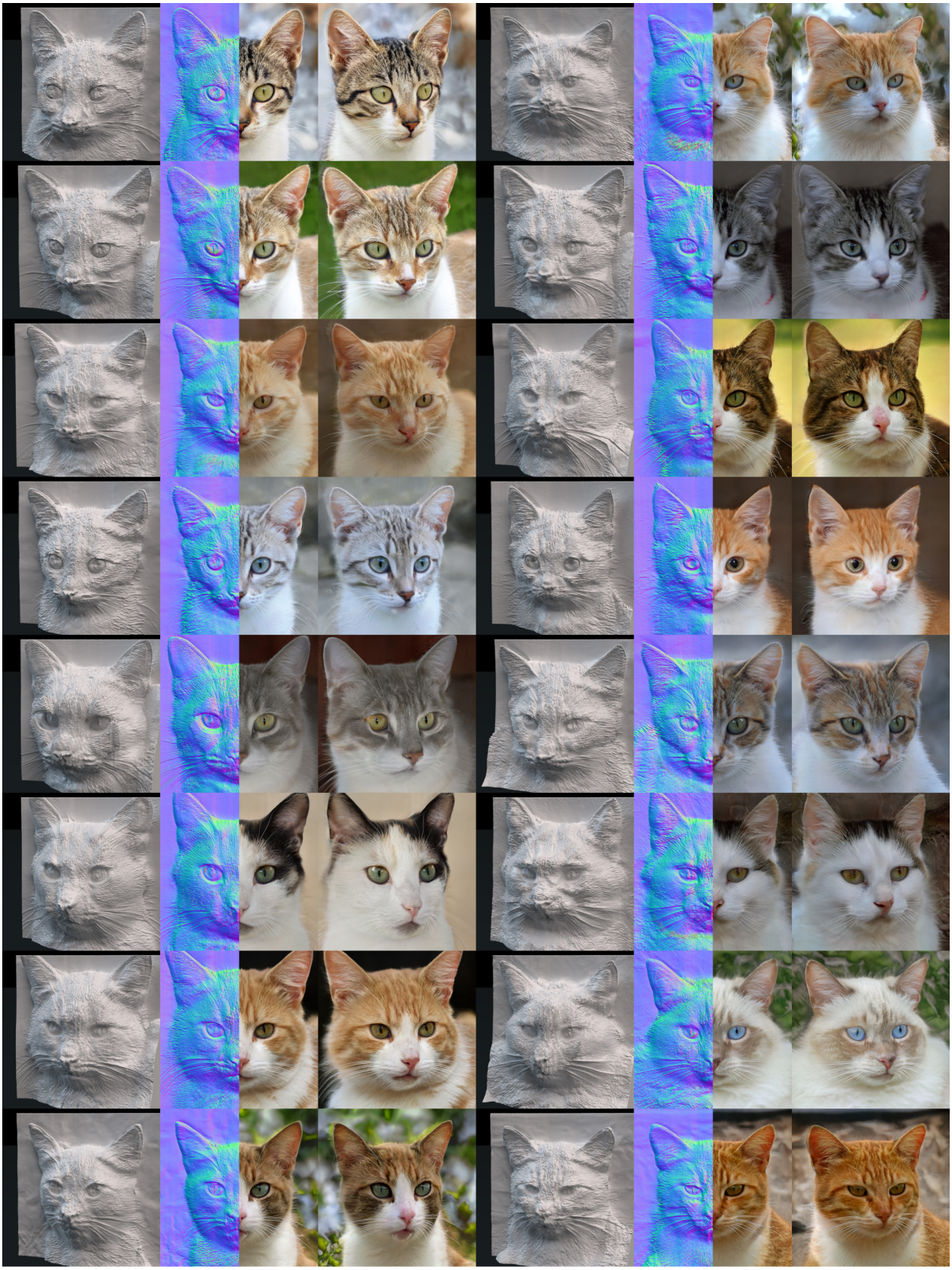

Figure A2. Curated samples from our AFHQ model.

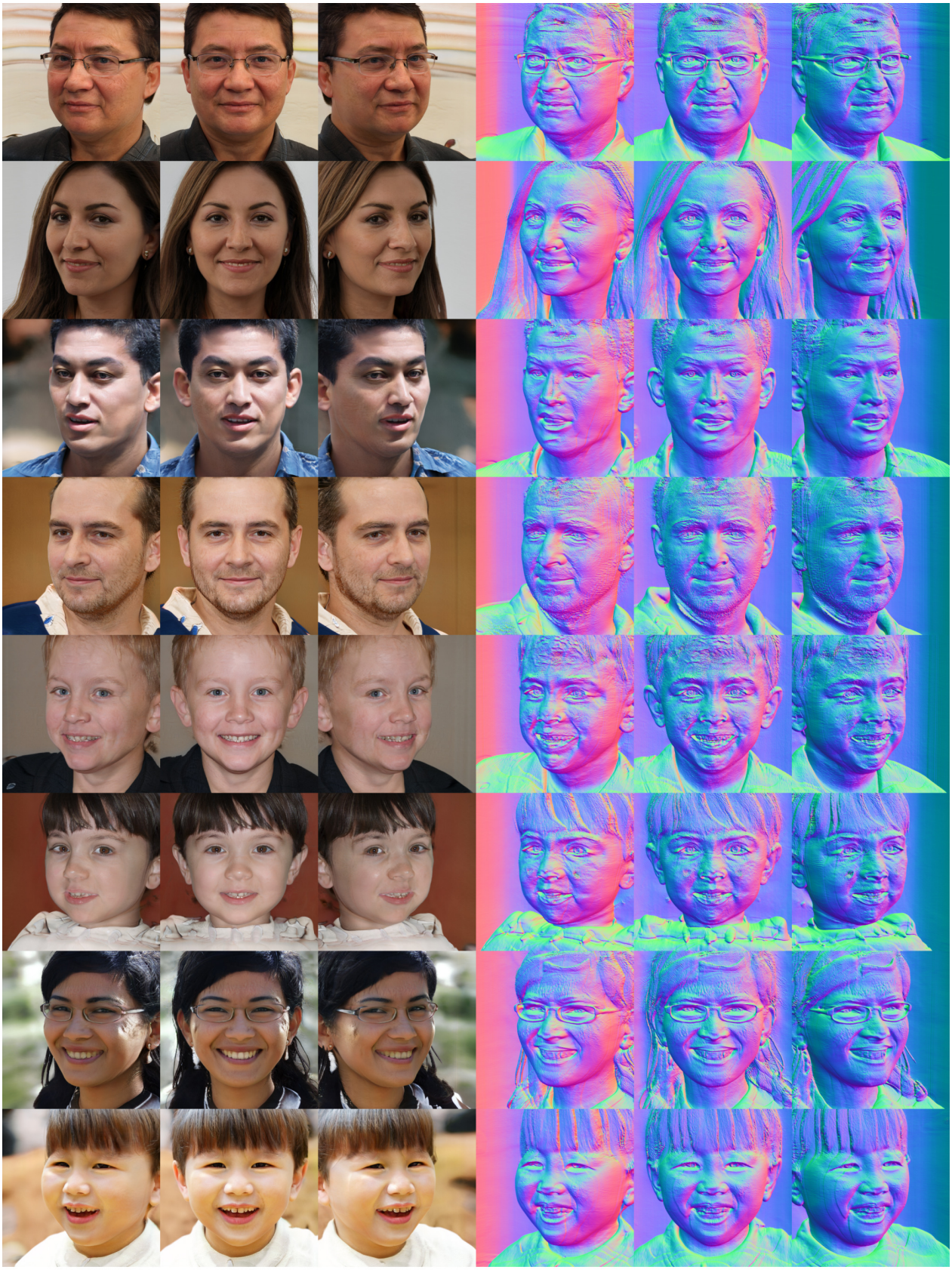

Figure A3. Uncurated (seeds 1-8) samples from our FFHQ model.

Figure A4. Uncurated (seeds 1-8) samples from our AFHQ model.

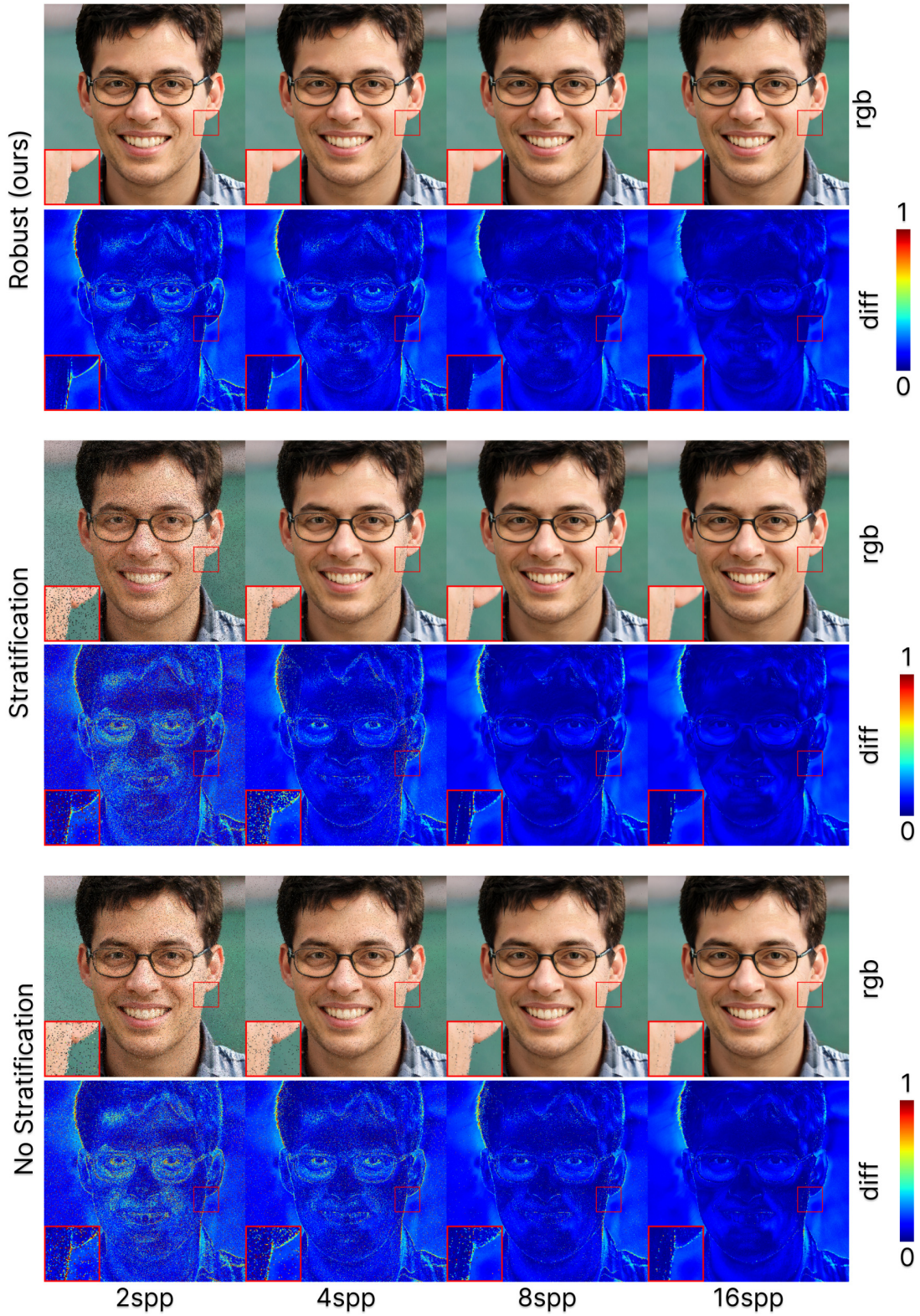

Figure A5. We show the rendering results of various sampling methods at different sample counts. We visualize both the rendering and a color map for the L2 error.

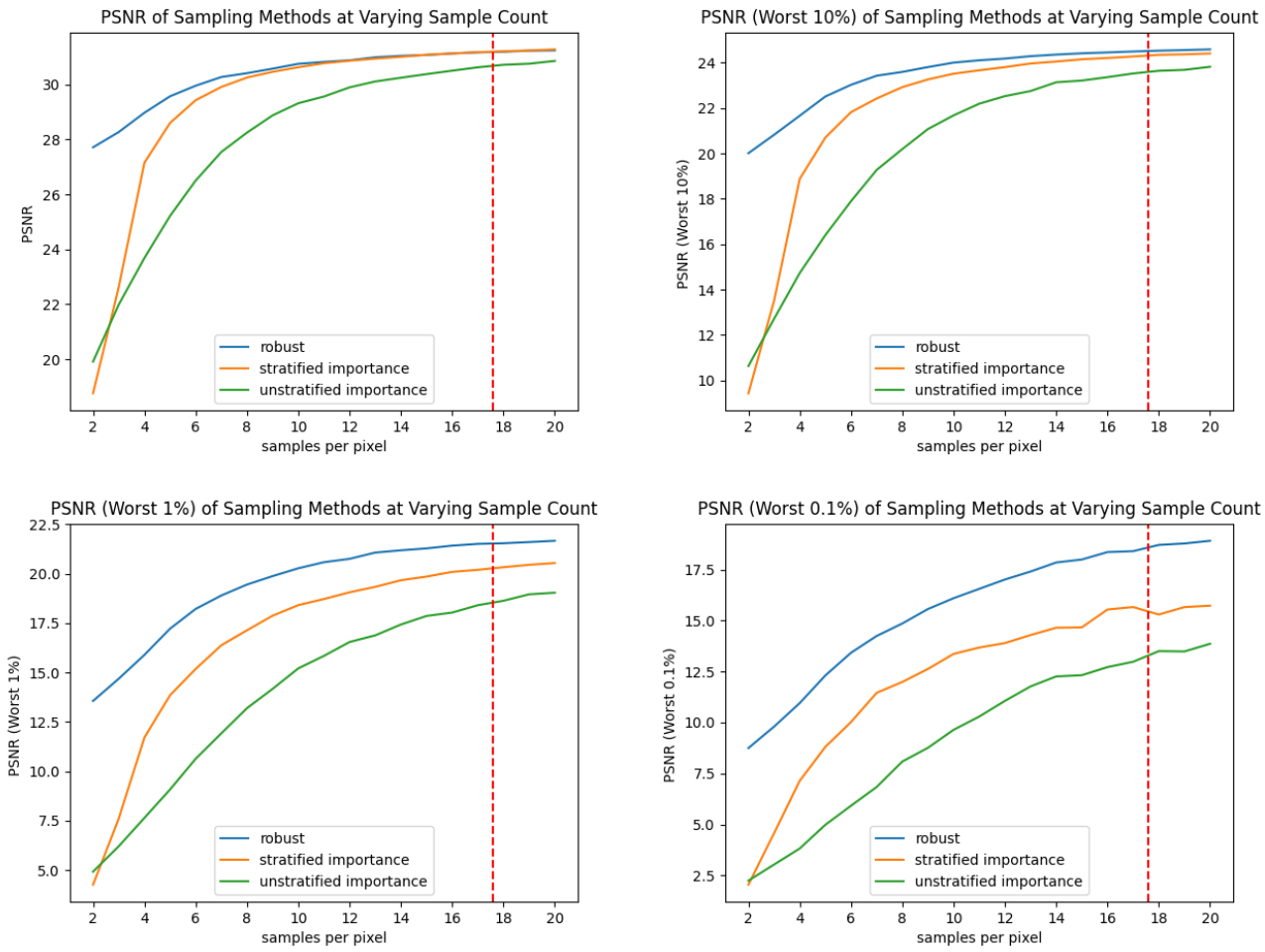

Figure A6. We show the PSNR for the worst percentage of pixels for all three sampling methods (subject is the same as Fig.A5). As seen in the charts, our method significantly outperforms the previous sampling methods at very low samples per pixel, e.g., 2 samples. At higher sample counts, our proposed sampling method has a significantly better *worst-case* result, e.g., for the worst 1% or 0.1% of pixels, as seen in the lower half. The importance of this is detailed in A2.1. The red dotted line indicates the maximal number of samples during training that fit on one 80gb A100 when rendering two images (per GPU).

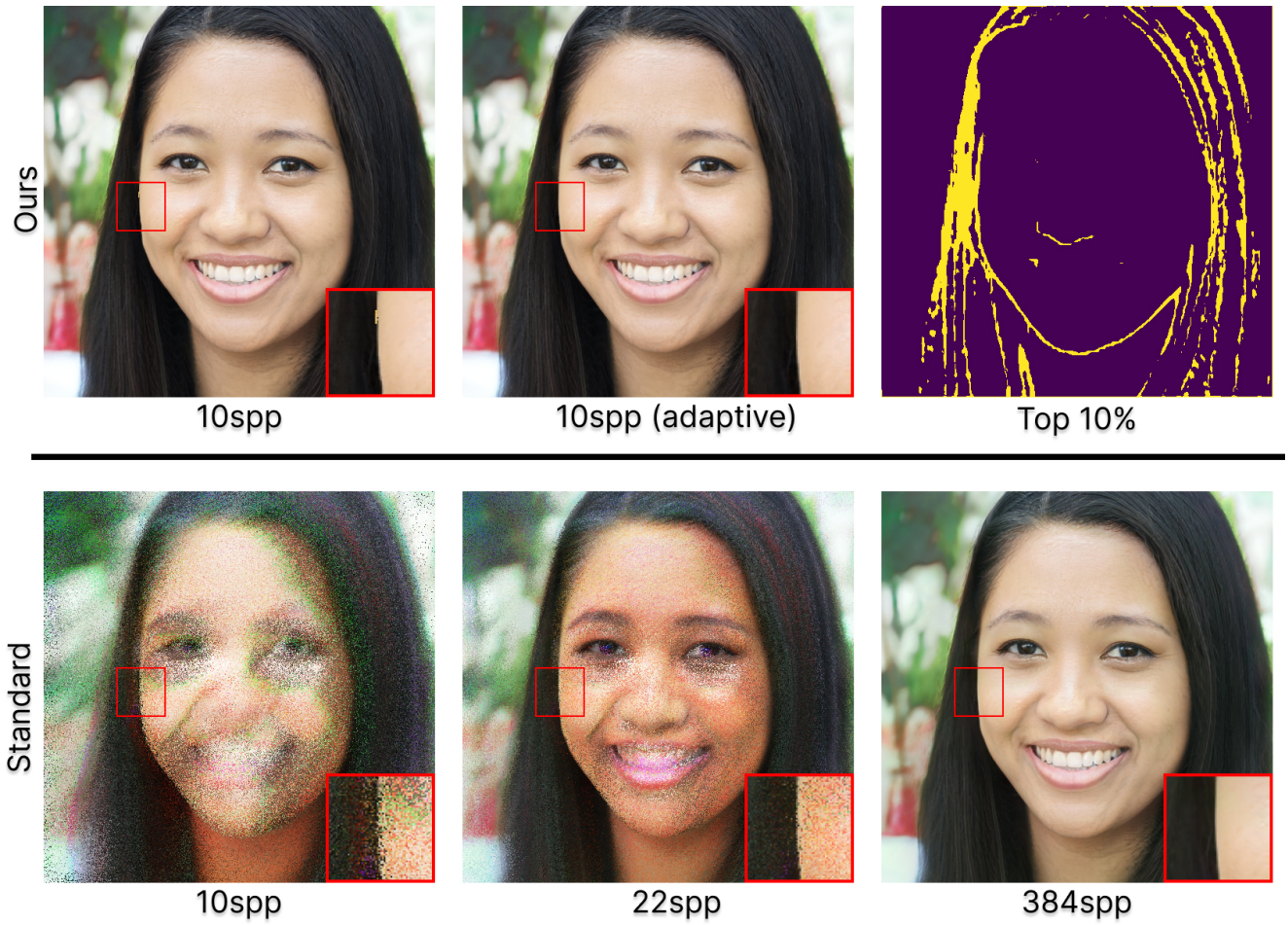

Figure A7. We visualize the effectiveness of our proposed adaptive sampling approach. In the top-left, we see that using 10 samples for all pixels results in jagged artifacts. Using the same number of samples, we allocate 9 samples to 90% of pixels and 19 to the remaining 10%, which prevents these jagged artifacts. The top 10% of pixels by the quantity computed in Subsec. A3.4 is visualized in the top right. In comparison, we also show the standard method without the learned sampler with 10 and 22 samples in the bottom-left and bottom-middle, respectively. 22 corresponds to 10 samples along with the 12 samples allocated for the initial probe P_{128}. Finally, we show the ground-truth rendering with 384 samples (192 coarse and 192 importance) in the bottom right.

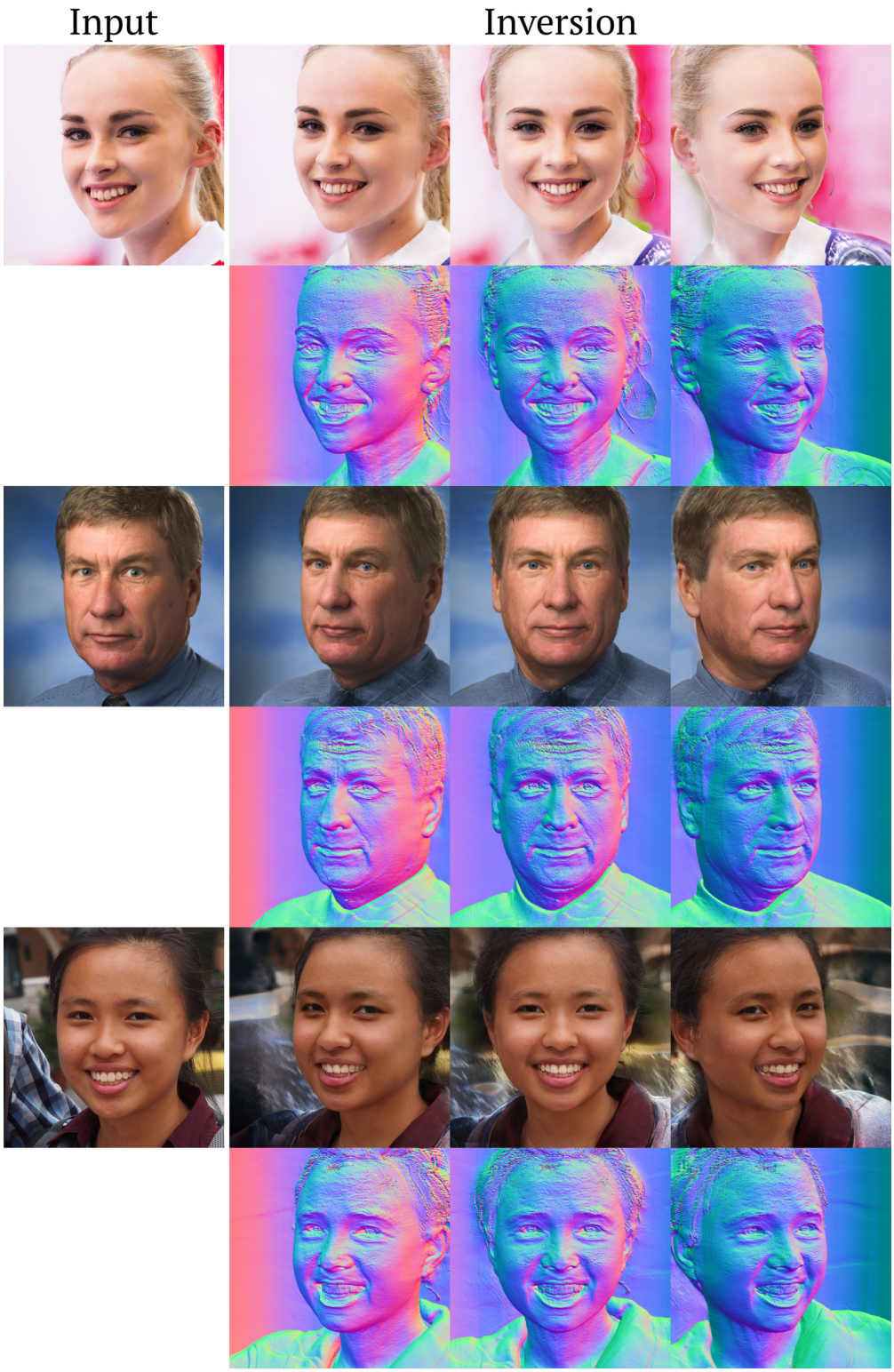

Figure A8. We showcase examples of single-view 3D reconstruction with our method. The left column shows the test image inputs, the right three columns show our inversion sample from three novel views.

Figure A9. 20 sample normal maps from [6] masked using Facer [4], from which we compute the N-FID score.

Figure A10. Three failure cases of our 3D generative model. First row shows seams on the side of the face; second row displays surface roughness to simulate specularity; and third row shows a rare phenomena where density appears close to the camera, occluding the subject.