Supplemental Material
Efficient Geometry-aware 3D Generative Adversarial Networks

Eric R. Chan *†[1,2], Connor Z. Lin*[1], Matthew A. Chan*[1], Koki Nagano*[2], Boxiao Pan[1], Shalini De Mello[2], Orazio Gallo[2], Leonidas Guibas[1], Jonathan Tremblay[2], Sameh Khamis[2], Tero Karras[2], and Gordon Wetzstein[1]

[1]Stanford University [2]NVIDIA

In this supplement, we first provide additional experiments (Sec. 1) and visual results (Sec. 2). We follow with details of our implementation (Sec. 3), including further descriptions of model architecture and training process, as well as hyperparameters. We discuss experiment details (Sec. 4), such as datasets and baselines, and further explanations for experiments such as inversion. Lastly, we consider artifacts (Sec. 5) that may be targets of future work. We encourage readers to view the accompanying supplemental video, which contains additional visual results, including a live demonstration of real-time synthesis.

1. Additional experiments

1.1. Analyzing pose/facial expression correlation in FFHQ

Fig. 1 plots the likelihood a subject from FFHQ [17] is smiling (measured by [38]), against head yaw (computed by [9]). The plot indicates that individuals facing towards the camera are more likely to be smiling than are individuals who are facing away from the camera. An intuitive explanation for this phenomenon is that people who are knowingly being photographed, as in portrait images, are more likely to be smiling than people who are photographed candidly.

Left uncompensated for, this correlation between pose and facial expressions incentivizes "expression warping", where the expressions of synthesized faces shift as we move the camera. We propose dual discrimination (Section 4.3 of the main paper) and generator pose conditioning (Section 4.4 of the main paper) to reduce such expression warping.

1.2. COLMAP reconstruction

To further validate the multi-view consistency of our method, we employ COLMAP [32, 33] to reconstruct a point-cloud of a synthesized video sequence (Fig. 2). We reconstruct a video sequence of 128 frames, taken from an oval trajectory similar to the camera paths shown in the supplemental video. We use COLMAP's "automatic" reconstruction, without specifying camera parameters. The re-

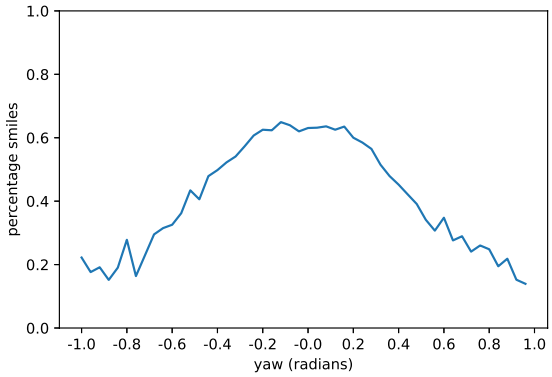


Figure 1. We plot the probability of smiling against head yaw angle, as measured by [38]. People looking at the camera are more likely to be smiling than people angled away, indicating a correlation between scene appearance and camera pose.

Figure 2. COLMAP [32, 33] reconstruction of 128 frames of synthesized video (top) which followed an oval trajectory. The resulting dense, well-defined point cloud (bottom) is indicative of highly multi-view-consistent rendering.

*Equal contribution.
†Part of the work was done during an internship at NVIDIA.

sulting point cloud is dense and well-defined, indicating that our 3D GAN produces highly multi-view-consistent renderings.

1.3. Regularizing generator pose conditioning

Figure 3. Naively applying generator pose conditioning results in a degenerate solution because the generator is always aware of the location of the rendering camera. Such an approach produces reasonable renderings when taken from the "intended" viewing angle, (i.e. the camera pose the generator was conditioned on). However, if we freeze the conditioning information and move the camera at inference, it is clear that the model has learned to produce "billboards" angled towards the known location of the camera.

As described in Section 4.4 of the main paper, we regularize generator pose conditioning by randomly swapping the conditioning pose of the generator with another random pose with 50% probability. Fig. 3 shows the result of training a model with generator pose conditioning but without any swapping regularization—the generator always receives, as a conditioning input, the true pose of the rendering camera. The model learns a degenerate solution in which it creates a "billboard" angled towards the rendering camera. We prevent this degenerate solution by randomly swapping the conditioning camera pose with an alternative pose sampled from the dataset pose distribution. For models shown, we swap the conditioning vector with 100% probability at the start of training; the swapping probability is linearly decayed to 50% over the first 1M images. For the remainder of training, we maintain 50% swapping probability.

1.4. Robustness to imprecise camera poses

Our method expects a dataset in which each image is labeled with an approximate camera pose, in order to enable sampling camera poses from the dataset distribution and discriminator pose conditioning. While such labelling can be easily performed with pre-trained pose extractors on humans [9] and cats [20], extracting accurate poses may be difficult for some datasets. This section evaluates reliance on discriminator pose conditioning and on accurate camera poses. We train five additional models on FFHQ 256^2: a "baseline" configuration without discriminator pose conditioning, and four discriminator-pose-conditioned models where camera poses are corrupted with increasing levels of

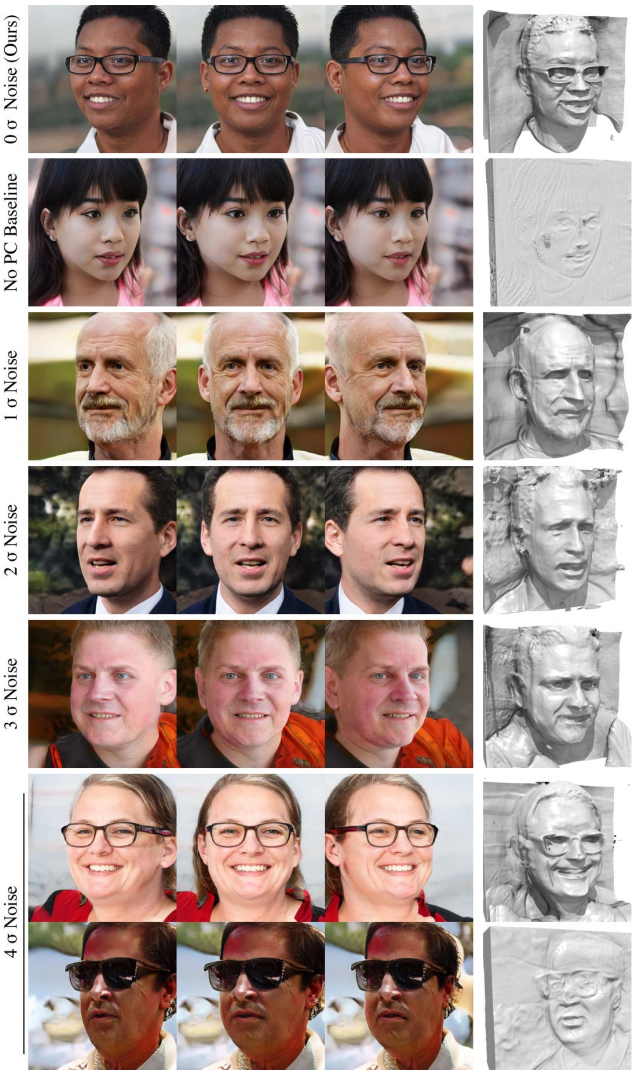


Figure 4. In order to gauge robustness to the accuracy of the supplied camera poses, we compare a baseline without discriminator pose conditioning against discriminator-pose-conditioned models where camera extrinsics are corrupted by noise. Without discriminator pose conditioning, the model learns a degenerate solution in which heads are drawn as a texture flattened onto a plane. Even highly imprecise extrinsics (e.g. camera poses corrupted by three standard deviations of noise) are capable of resolving this degenerate solution and allow recovery of accurate 3D shapes.

random noise. We calculate the 4×4 standard deviation matrix, σ, by taking the standard deviation across the dataset of ground-truth 4×4 camera pose matrices. We train four models with "imprecise" camera poses: ($1\ \sigma$, $2\ \sigma$, $3\ \sigma$, $4\ \sigma$) where the input camera poses matrices are corrupted with 1, 2, 3, and 4 standard deviations of Gaussian noise, respectively. We train these five ablations on FFHQ 256^2 with a shortened training curriculum of 4M images, in order to save computational resources.

Fig. 4 shows the results of this experiment. Without discriminator pose conditioning, the model falls into a degen-

	FFHQ					Cats		Cars	
	FID↓	KID↓	ID↑	Depth↓	Pose↓	FID↓	KID↓	FID↓	KID↓
GIRAFFE 128^2	—	—	—	—	—	—	—	27.3	1.703
GIRAFFE 256^2	31.5	1.992	0.64	0.94	.089	16.1	2.723	—	—
π-GAN 128^2	29.9	3.573	0.67	0.44	.021	16.0	1.492	17.3	0.932
Lift. SG 256^2	29.8	—	0.58	0.40	.023	—	—	—	—
Ours 128^2	—	—	—	—	—	—	—	**2.75**	**0.097**
Ours 256^2	**4.8**	**0.149**	0.76	**0.31**	**.005**	**3.88**	**0.091**	—	—
Ours 512^2	**4.7**	**0.132**	**0.77**	0.39	**.005**	**2.77**†	**0.041**†	—	—

Table 1. Quantitative evaluation using FID, KID×100, identity consistency (ID), depth accuracy, and pose accuracy for FFHQ [17] and FID, KID×100 for AFHQv2 Cats [7,16] and ShapeNet Cars [6,35]. Labeled is the image resolution of training and evaluation. † Trained with adaptive discriminator augmentation [15].

erate solution in which it renders textures on a flat plane, without properly capturing the 3D shape of scenes. Providing even very imprecise camera poses is enough to break this tendency; conditioning the discriminator on camera poses distorted by three standard deviations of Gaussian noise still produces accurate 3D shapes. With extreme noise (e.g. four standard deviations), some scenes maintain the correct 3D structure while others are flattened onto the plane. Our results indicate that while our method requires additional information to prevent collapse, only very weak supervision is necessary. Future work may examine this tendency further and discover ways to prevent this undesirable behavior without requiring images to be labelled with poses.

1.5. Extrapolation to steep camera angles

Fig. 5 provides a visual comparison of our method against baselines for generating views from steep camera poses. We note that the FFHQ [17] dataset is primarily composed of front-facing images—few images depict faces from extreme yaw angles, and even fewer images depict faces from extreme pitch angles. Nevertheless, reasonable extrapolation to the edges of the pose distribution is a desirable quality and indicates reliance on a robust 3D representation.

Lifting StyleGAN [34], which represents scenes as a textured mesh, demonstrates consistent rendering quality. However the steep camera angles reveal inaccurate 3D geometry (e.g. foreshortened faces) learned by the method. π-GAN [5], reasonably extrapolates to steep angles but exhibits visible quality degradation at the edges of the pose distribution. GIRAFFE [29], being highly reliant on view-inconsistent convolutions, has difficulty reproducing angles that are rarely seen in the dataset. If we force GIRAFFE to extrapolate beyond the camera poses sampled at training (e.g. the leftmost and rightmost images of Fig. 5b), we receive degraded, view-inconsistent images rather than renderings from steeper angles. The problem is amplified for pitch (Fig. 5a) because the dataset's pitch range is even narrower.

Our method, despite also using 2D convolutions, is less reliant on view-inconsistent convolutions for considering the placement of features in the final image. By utilizing an expressive 3D representation as a "scaffold", our method provides more reasonable extrapolation to rare views in both pitch and yaw than methods that more strongly depend on image-space convolutions for image synthesis, such as GIRAFFE [29].

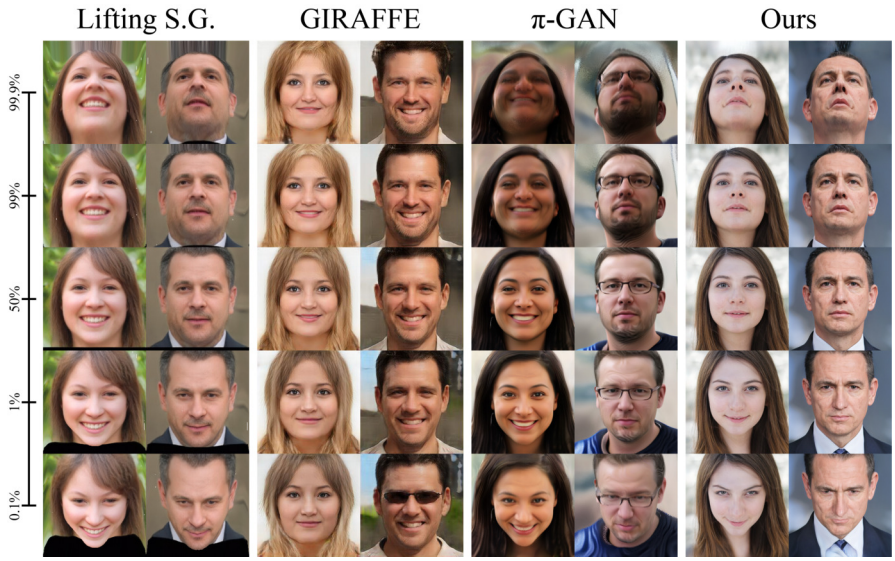
1.6. Additional quantitative results

Table 1 is an expanded version of Table 2 of the main manuscript that provides additional quantitative metrics, including Kernel Inception Distance [2] for all datasets and image quality evaluations for ShapeNet Cars. Strong relative performance on Cars, a dataset in which camera poses are distributed uniformly about the sphere, is evidence that our method is not restricted to face-forward datasets like FFHQ [17] and AFHQv2 [7, 16].

2. Additional visual results

Style mixing, in shapes. Fig. 6 shows the underlying shapes of the style mixing [17] examples in Fig. 8 of the main manuscript. While mixed examples inherit most of their shape structure from the modulations of the backbone's low-resolution layers, the modulations of the high-resolution layers can influence fine details in the shape, such as eye regions and hair patterns. The results were obtained from a model trained without style-mixing regularization.

Additional single image 3D reconstructions. Fig. 7 provides additional 3D reconstructions of single test images through Pivotal Tuning Inversion (PTI) [31] of a model trained on FFHQ 512^2. A pipeline for high-fidelity, single-image reconstruction of faces that does not require explicit 3D ground-truth training data opens the door for many



(a) Extrapolation to steep pitch angles.

(b) Extrapolation to steep yaw angles.

Figure 5. We compare methods in their extrapolation to steep camera viewing angles. Labelled is the percentile for camera pitch or yaw. A yaw angle in the 96th percentile means 96% of training poses are less steep, i.e. 4% of training poses are beyond the given pose.

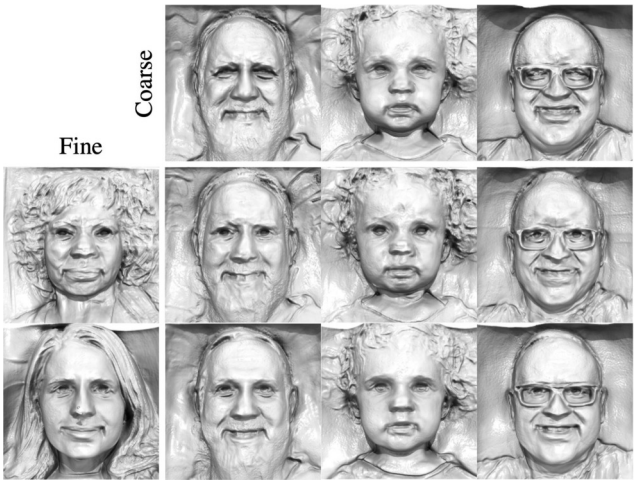


Figure 6. Style-mixing [16–18] shapes from a model trained on FFHQ 512^2, without truncation. Aligns with Fig. 8 of the main manuscript, which shows color renderings of the same seeds. The result illustrates that while a mixed example inherits the majority of its structure from its "coarse" input (i.e. modulations of layers 0-6), the "fine" input (i.e. modulations of layers 7-13) can influence the more delicate details of the shape (e.g. eye regions, hair patterns), in addition to having much control over the overall colors in rendered images.

Figure 7. Additional single-view 3D reconstructions of test images demonstrate a use for our generator's learned prior over facial features.

promising applications, such as photo-to-avatar creation.

Shapenet Cars. Fig. 8 contains uncurated renderings from random camera poses for models trained with ShapeNet Cars [6, 35]. This experiment serves as a demonstration that our method is capable of operating successfully on datasets that include camera poses that span the entire 360° camera azimuth and 180° camera elevation distributions, unlike 2.5D GANs [34], which are intended for face-forward datasets.

Additional selected examples synthesized with AFHQv2 Cats. Fig. 9 shows renderings and shapes for selected examples, synthesized by our method trained on AFHQv2 Cats [7, 16] 512^2.

Uncurated examples synthesized with AFHQv2 Cats. Fig. 10 provides uncurated examples of cats produced by GIRAFFE [29], π-GAN [5], and our method, trained at image rsolutions of 256^2, 128^2, and 512^2, respectively.

Uncurated examples synthesized with FFHQ. Fig. 11 provides uncurated examples of faces produced by our method, trained with FFHQ [17] 512^2. We apply truncation [4, 17, 25], with $\psi = 0.5$.

Latent code interpolation. Fig. 12 provides linear interpolations between latent codes for selected examples produced by our method trained on FFHQ 512^2. Our result illustrates that our 3D GAN inherits the well-behaved latent space of the StyleGAN2 [18] backbone, which enables smooth interpolations in both color renderings and underlying shapes.

Additional selected examples synthesized with FFHQ Fig. 13 depicts renderings and shapes for selected examples, synthesized by our method trained on FFHQ 512^2.

3. Implementation details

We implemented our 3D GAN framework on top of the official PyTorch implementation of StyleGAN2, an updated version of which is available at `https://github.com/NVlabs/stylegan3`. Most of our training parameters are identical to those of StyleGAN2 [18], including the use of equalized learning rates for the trainable parameters [14], a minibatch standard deviation layer at the end of the discriminator [14], exponential moving average of the generator weights, and a non-saturating logistic loss [12] with R1 regularization [26].

Two-stage training. In order to save computational resources, we perform the majority of the training at a neural rendering resolution of 64^2, before gradually stepping the resolution up to 128^2. Note that the final image resolution remains fixed throughout training (e.g. 256^2 or 512^2). We implement this simply by bilinearly resizing the raw neural rendering I_{RGB} to 128^2 before it is operated on by the super-resolution module. Thus, the super-resolution module always receives a 128^2-sized feature map as an input, regardless of the actual neural rendering resolution. In contrast to previous progressive growing strategies [5, 14] that double the resolution in a single step, we gradually increase

Figure 8. Qualitative comparison of uncurated examples of cars. All methods are sampled with truncation [4, 17, 25], using $\psi = 0.7$.

the neural rendering resolution, pixel-by-pixel, over 1 million images, i.e., (64^2, 65^2, 66^2, ..., 126^2, 127^2, 128^2). We continue training with the resolution fixed at 128^2 for an additional 1.5 million images, for a total of 2.5M iterations of fine-tuning. This two-stage training procedure provides a roughly $2\times$ speed-up versus training from scratch at full resolution and produces similar results to training at full neural rendering resolution from scratch.

Backbone. Our backbone (i.e., StyleGAN2 generator) follows the implementation of [18], with a mapping network of 8 hidden layers. For all of our experiments (regardless of final image resolution), the backbone operates at a resolution of 256^2. We modify the output convolutions such that they produce a 96-channel output feature image, which we reshape into three planes, each of shape $256 \times 256 \times 32$. Unlike approaches that require pre-trained 2D image GANs [34], we do not utilize pre-trained StyleGAN2 checkpoints for the backbone; the entire pipeline is trained end-to-end.

Figure 9. Curated examples from a model trained on AFHQv2 [7, 16] 512^2.

For large datasets, such as FFHQ [17] and ShapeNet Cars [6, 35], we train from scratch with random initialization; for small datasets, such as AFHQv2 [7, 16], we follow prevailing methodology [15] by fine-tuning from a checkpoint trained on a larger dataset.

Decoder and volume rendering. Our decoder is implemented as an MLP with a single hidden layer of 64 hidden units and uses the softplus activation function. The decoder takes as input a 32-channel aggregated feature vector; it produces a 33-channel vector that we split into a scalar density prediction and a 32-channel feature. We use neural volume rendering [27] of features [29], with two-pass importance sampling. For FFHQ [17] and AFHQv2 [7, 16], we use 48 uniformly-spaced and 48 importance samples per ray; for ShapeNet Cars, we use 64 uniformly-spaced and 64 importance samples per ray. When rendering videos that feature thin surfaces, we found it beneficial to increase the samples per ray during inference to reduce flicker.

Super-resolution. We implement our super-resolution model with two 'blocks' of StyleGAN2's modulated convolutions [18], with noise inputs disabled. The blocks contain convolutions of channel-depth 128 and 64, respectively.

Figure 10. Uncurated examples of cats, for GIRAFFE [29] 256^2, π-GAN 128^2, and our method 512^2. All methods are sampled with truncation [4, 17, 25], using $\psi = 0.7$.

Discriminator. Our discriminator is a StyleGAN2 [18] with two modifications. First, to enable dual discrimination, we adjust the input layer to accept six-channel input images, rather than 3-channel input images. Fig. 14 provides a diagram that illustrates the creation of these six-channel inputs, for both real and generated images. Second, we condition the discriminator on the camera parameters of the incoming image to help prevent degenerate shape solutions; we follow the class-conditional discriminator modifications of [15] to inject this information.

Mixed Precision. To speed up training, we use a similar mixed-precision methodology as [15]. We use FP16 in the four highest resolution blocks of the discriminator and in both blocks of our super-resolution module. We do not use FP16 in our generator backbone.

R1 Regularization. We use R1 regularization [26] with $\gamma = 1$ for all datasets and resolutions, except for ShapeNet Cars, where we use $\gamma = 0.1$. Regularization strengths were informally chosen based on values that have shown success with previous methods [15, 18].

Density Regularization. Further experiments, conducted after our initial submission, suggested that additional regularization over the estimated density field reduced the prevalence of undesirable seams and other shape artifacts. Similar to the total variation regularization used in previous work [23], our density regularization encourages smoothness of the density field. For each generated scene in the batch, we randomly sample points $\mathbf{x}$ in the volume and also sample additional 'perturbed' points that are offset with a small amount Gaussian noise $\delta\mathbf{x}$. Our density regularization loss is an L2 loss that minimizes the difference between the estimated densities $\sigma(\mathbf{x})$ and $\sigma(\mathbf{x} + \delta\mathbf{x})$. We apply our

Figure 11. Images and geometry for seeds 0-31, synthesized using a model trained on FFHQ [17] 512^2. Sampled with truncation [17], using $\psi = 0.5$.

density regularization over 1000 pairs of randomly sampled points every four training iterations.

Training. We train all models with a batch size of 32. We use a discriminator learning rate of 0.002 and a generator learning rate of 0.0025. Following [16], we blur images as they enter the discriminator, gradually reducing the blur amount over the first 200K images. Unlike [18], we train without style-mixing regularization.

Using the two-stage training discussed previously, we train at a resolution of 64^2 for 25M images and at 128^2 for an additional 2.5M images. Using a neural rendering resolution of 64^2, our 3D GAN framework takes $\sim$24 seconds to train on 1000 images (24 s/kimg) on 8 Tesla V100 GPUs; this increases to 46 s/kimg at a neural rendering resolution of 128^2. For reference, StyleGAN3-R [16] achieves training rates of 20 s/kimg on similar hardware.

Our total training time on 8 Tesla V100 GPUs is on the order of 8.5 days (7 days of 64^2 training, plus 1.5 days of 128^2 fine-tuning), compared to 6 days on similar hardware for StyleGAN3-R.

Inference-time depth samples. We use neural volume rendering [27] with two-pass importance sampling to render feature images from our tri-plane representation. We found that increasing the number of samples per ray at inference time can reduce unwanted flickering when rendering videos that feature thin objects such as eye glasses. For clips shown in the supplemental video, we double both the number of coarse samples (from 48 to 96) and the number of fine samples (from 48 to 96), bringing the total number of depth samples per ray to 192. Increasing the number of samples per ray incurs a penalty to the rendering speed. When using 96 total depth samples per ray, frame rates are reduced to approximately 24 frames per second with tri-plane caching – down from 36 frames per second when using the default 48 samples. Images shown in the main manuscript were synthesized without increasing the number of depth samples along each ray.

AFHQv2. Following [15], we fine-tune from FFHQ-trained models to achieve optimum performance on Cats. Beginning from a checkpoint trained on FFHQ, we train for 6.2M images at a neural rendering resolution of 64^2; and for an additional 2.6M images, while fine-tuning the neural rendering resolution to 128^2. Because π-GAN and GIRAFFE were not designed with the benefits of adaptive discriminator augmentation (ADA) [15], we also do not use ADA for our method at 256^2, in an effort to keep comparisons across methods fair. We use adaptive discriminator augmentation

Figure 12. Linear interpolations between latent codes, showing renderings and shapes.

with its default settings, for our method only at 512^2 .

4. Experiment details

4.1. Baselines

π-GAN [5] is a 3D-aware GAN that relies upon a FiLM-conditioned MLP with periodic activation functions for camera-controllable synthesis. We utilized the official code (https://github.com/marcoamonteiro/pi-GAN) and trained until convergence with the parameters recommended for analogous datasets.

GIRAFFE [29] is a 3D-aware GAN that incorporates a compositional 3D scene representation to enable controllable synthesis. We utilized the official code (https://github.com/autonomousvision/giraffe) and trained until convergence with the parameters recommended for analogous datasets.

Lifting StyleGAN [34] is a method for disentangling and lifting a pre-trained StyleGAN2 image generator to 3D-aware face generation. The original Lifting StyleGAN manuscript reports results on a slightly tighter crop of FFHQ than we used. Because we had difficulty matching the quality of Lifting StyleGAN's pre-trained model when we trained it from scratch on our less-cropped dataset, we instead used their official pre-trained model for their tighter crops and the FID score reported in their manuscript. We utilized the offical code, found here: (https://github.com/seasonSH/LiftedGAN).

StyleGAN2 is a style-based GAN that achieves state-

Figure 13. Additional selected examples, from a model trained on FFHQ [17] 512^2.

of-the-art image quality for 2D image synthesis and features a well-behaved latent space that enables image manipulation. We obtained a pre-trained checkpoint for StyleGAN2 on FFHQ 512^2 from the collection of official models (https://catalog.ngc.nvidia.com/orgs/nvidia/teams/research/models/stylegan2). Following the recommended tuning of [15], we trained both StyleGAN2 config F and the 512×512 config from [15], sweeping R1 [26] regularization strength, $\gamma = \{0.2, 0.5, 1, 2, 5, 10, 20\}$. The best result for AFHQv2 was obtained with StyleGAN2 config F, after training for 10M images at $\gamma = 1$.

4.2. Dataset Details

FFHQ We prepare our dataset by starting with the "in-the-wild" version of the FFHQ dataset [17], which is composed of uncropped, original PNG images of people sourced from Flickr. We use an off-the-shelf face detection and pose-extraction pipeline [9] to both identify the face region and label the image with a pose. We crop the images to roughly the same size as the original FFHQ dataset.

We assume fixed camera intrinsics across the entire dataset, with a focal length of $4.26 \times image\_width$, equivalent to a standard portrait lens. We prune a small number of images that resisted face detection; our final dataset contains 69957 images. We augment the dataset with horizontal

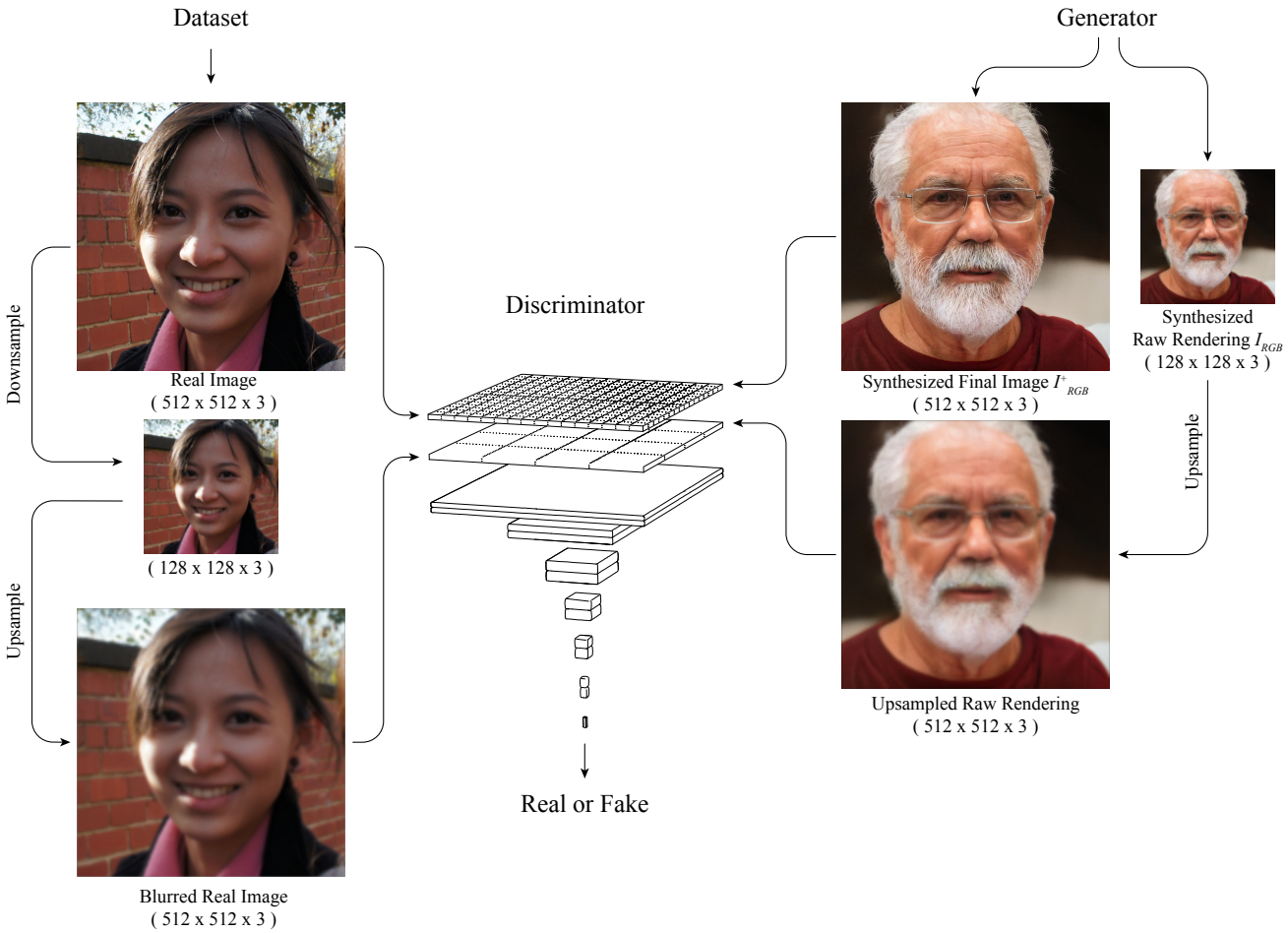


Figure 14. In dual-discrimination, we discriminate on a six-channel concatenation of the final image and the raw neural rendering, in order to maintain consistency between high-resolution final images and view-consistent (but low resolution) neural renderings. This diagram illustrates how we obtain a six-channel discriminator input tensor for both real and fake images. Our generator produces both a 512^2 final rendering (I^{+}_{RGB}) as well as the (128^2) raw neural rendering (I_{RGB}). The raw rendering, I_{RGB} is the first three channels of the 32-channel rendered features, I_F. We create a six-channel discriminator input by upsampling the raw image to 512^2 and concatenating it with the final image to form a ($512 \times 512 \times 6$) discriminator input tensor. For real images, we extract a 512^2 real image from the dataset and downsample it to the same size as I_{RGB} to obtain an analogue for I_{RGB}. We then upsample this image back to 512^2 and concatenate it with the original image to form a ($512 \times 512 \times 6$) discriminator input tensor. The downsample-then-upsample operation has the effect of blurring the original image.

flips.

AFHQv2 We used the AFHQv2 dataset [16], which is a higher-quality version of the original AFHQ dataset [7]. AFHQv2 provides closeups for animal faces including cats, dogs, and wildlife. We use the 'cats' split, which contains approximately 5000 images, for our experiments. As with FFHQ, we assume fixed camera intrinsics across the dataset; for simplicity, we use identical intrinsics to FFHQ. Camera poses were extracted via landmark detection [20] and an open-source Perspective-n-Point algorithm [3]. We augment the dataset with horizontal flips.

ShapeNet Cars For additional validation, we compare methods on ShapeNet Cars [6, 35] to evaluate performance on a dataset that contains views from all angles. We adopted the dataset and setup from [35], which is composed of 128^2 resolution renderings of synthetic cars, each labelled with camera parameters. The dataset contains 2457 unique cars; each car is rendered from 50 views randomly sampled from the entire sphere. We use the known camera parameters for each image and do not augment the dataset with image space augmentations.

4.3. Single scene overfitting.

To illustrate the effectiveness of our architecture, we evaluate the relative performance of the tri-plane 3D representation against a comparable voxel-based hybrid representation and Mip-NeRF [1] on the *Family* scene of Tanks & Temples [19] dataset as desribed in Section 3 of the main manuscript. We use the pre-processed images, as well as the training/test split, of [22]. We use 512 uniformly-spaced depth samples and 256 importance samples per ray and a ray batch size of 6400. The tri-planes are treated as learnable parameters of shape $3\times48\times512\times512$. The dense voxel parameters were chosen to optimize quality for comparable parameter count as the tri-planes; the voxel features are of shape $18\times128\times128\times128$. Both voxel and tri-plane hybrid representations are coupled with two-layer, 128 hidden unit decoders with Fourier feature embeddings [36]. We train voxel and cube representations for 200K iterations; we train Mip-NeRF for the recommended 1M iterations.

4.4. Pivotal tuning inversion.

We use off-the-shelf face detection [9] to extract appropriately-sized crops and camera extrinsics from test images and we resize each cropped image to 512^2. We follow Pivotal Tuning Inversion (PTI) [31], optimizing the latent code for 500 iterations, followed by fine-tuning the generator weights for an additional 500 iterations.

For inversion of grayscale images, we convert the generator's 3-channel, RGB renderings to perceived luminance, Y, before computing image distance loss during optimization. This allows the generator's prior to colorize the renderings. To compute single-channel luminance from 3-channel RGB images, we use $Y = 0.299R + 0.587G + 0.114B$. For grayscale optimization, we use 400 latent code inversion steps and 250 generator fine-tuning steps.

4.5. Evaluation Metrics

FID and KID. We compute Fréchet Inception Distance (FID) [13] and Kernel Inception Distance (KID) [2] image quality metrics between 50k generated images and all training images using the implementation provided in the StyleGAN3 [16] codebase.

Geometry. We follow a similar procedure to [34] in the evaluation of geometry. We generate 1024 images and depth maps from random poses that match the dataset pose distribution. With the application of a pre-trained 3D face reconstruction model [9], we generate a "pseudo" ground-truth depth map for each generated image. Next we limit both the generated depth maps and "pseudo" ground-truth depth maps to the facial regions as defined by the reconstruction model. Finally, we normalize all depth maps to zero mean, unit variance and calculate the L2 distance between them.

Multi-view consistency. We evaluate multi-view consistency and face identity preservation for models trained on FFHQ [17] by measuring ArcFace [8] cosine similarity. For each method, we generate 1024 random faces and render two views of each face from poses randomly selected from the training dataset pose distribution. For each image pair, we measure facial identity similarity [8] and compute the mean score.

Pose accuracy. We evaluate pose accuracy with the help of a pre-trained face reconstruction model [9]. With [9], we detect pitch, yaw, and roll from 1024 generated images then compute L2 loss against the ground truth poses to determine each model's pose drift.

Runtime. We evaluate runtime for each model by calculating the average framerate over a 400 frame sequence. We process frames consecutively, i.e., with batch size 1. In order to give each method a best-case-scenario, we ignore operations such as copying rendered frames from GPU to CPU and saving files to disk.

FACS estimation In Section 5.2 of the main paper, we quantitatively measure the effect of dual discrimination and generator pose conditioning at preserving facial expressions across multi-view face videos. To evaluate facial expressions, we employ a proprietary facial tracker that measures detailed movement of sub-regions of the face in terms of Facial Action Coding System (FACS) [10] coefficients. Specifically, our facial tracker measures all 53 FACS blendshape coefficients defined in Li et al. [21] and we compared the variability in the 'mouthSmile_L' and 'mouthSmile_R' blendshape coefficients across the different videos.

4.6. Visualization of Geometry

To visualize shapes, we sample the volume to obtain a 512^3 cube of density values and extract the surface of the scene as a mesh using Marching Cubes [24]. We found that a levelset between 0 and 10 generally yielded visually appealing results. Renderings of shapes shown in this manuscript were generated using ChimeraX [11].

5. Discussion

5.1. Shape artifacts

Despite significant improvements in the quality of the 3D geometry compared to previous methods, our synthesized shapes are not free from artifacts, which are visible in geometry renderings throughout the main paper and supplement

(e.g. Fig. 11, Fig. 13). Sunken eye sockets allow the illusion of eyes that follow the viewing camera, even when the geometry and neural renderings are view-consistent; such "hollow face illusions" have demonstrated similar effects in the physical world. Similarly, deep creases near the corners of mouths enable the creation of "view-inconsistent" effects that in fact are faithful to the underlying shapes. Future work that incorporates stronger dataset priors, e.g. that eyeballs are convex, may help resolve these artifacts.

While our method produces more-detailed eyeglasses than previous methods, it tends to produce "goggles"—the sides of the eyeglasses are opaque where there should be empty space. Future neural rendering methods that can accurately model lens refraction may enable more faithful reconstruction of eyeglasses and other objects that contain transparent elements.

In some shapes and renderings generated by our method, a seam is visible between the face and the rest of the head. While we find the optional density regularization in Sec. 3 helps reduce such artifacts, we hypothesize that recent hybrid-SDF rendering solutions [30, 37, 39], which have shown promising results in robust geometry recovery from images, may yield improved shapes with fewer artifacts.

In the interests of simplicity, we model the scene with a single 3D representation, without any explicit background handling. Consequently, the generator learns to represent backgrounds of images with textured surfaces fused to foreground objects. Future work that models backgrounds with a separate 3D representation [28, 29, 40] may enable isolation of foreground objects.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In ICCV, 2021.

[2] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In International Conference on Learning Representations (ICLR), 2018.

[3] G. Bradski. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In International Conference on Learning Representations (ICLR), 2019.

[5] 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 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[6] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012, 2015.

[7] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020.

[8] Jiankang Deng, Jia Guo, Xue Niannan, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In CVPR, 2019.

[9] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In IEEE Computer Vision and Pattern Recognition Workshops, 2019.

[10] Paul Ekman and Wallace V. Friesen. Facial Action Coding System: A Technique for the Measurement of Facial Movement. Consulting Psychologists Press, 1978.

[11] Thomas D Goddard, Conrad C Huang, Elaine C Meng, Eric F Pettersen, Gregory S Couch, John H Morris, and Thomas E Ferrin. Ucsf chimerax: Meeting modern challenges in visualization and analysis. Protein Science, 27(1):14–25, 2018.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems (NeurIPS), 2014.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a nash equilibrium. In Advances in Neural Information Processing Systems (NeurIPS), 2017.

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In International Conference on Learning Representations (ICLR), 2018.

[15] 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.

[16] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In Advances in Neural Information Processing Systems (NeurIPS), 2021.

[17] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[18] 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.

[19] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics, 36(4), 2017.

[20] Taehee Brad Lee. Cat hipsterizer, 2018. https://github.com/kairess/cat_hipsterizer.

[21] Ruilong Li, Karl Bladin, Yajie Zhao, Chinmay Chinara, Owen Ingraham, Pengda Xiang, Xinglei Ren, Pratusha Prasad, Bipin Kishore, Jun Xing, et al. Learning formation of physically-based face attributes. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[22] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In Advances in Neural Information Processing Systems (NeurIPS), 2020.

[23] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. ACM Transactions on Graphics (SIGGRAPH), 2019.

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

[25] Marco Marchesi. Megapixel size image creation using generative adversarial networks, 2017.

[26] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In International Conference on Machine Learning (ICML), 2018.

[27] 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.

[28] Michael Niemeyer and Andreas Geiger. CAMPARI: Camera-aware decomposed generative neural radiance fields. arXiv preprint arXiv:2103.17269, 2021.

[29] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[30] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In IEEE International Conference on Computer Vision (ICCV), 2021.

[31] 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.

[32] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[33] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In European Conference on Computer Vision (ECCV), 2016.

[34] Yichun Shi, Divyansh Aggarwal, and Anil K Jain. Lifting 2D stylegan for 3D-aware face generation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[35] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In Advances in Neural Information Processing Systems (NeurIPS), 2019.

[36] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In Advances in Neural Information Processing Systems (NeurIPS), 2020.

[37] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. Advances in Neural Information Processing Systems (NeurIPS), 2021.

[38] Jie Wu. Facial expression recognition pytorch, 2018. https://github.com/WuJie1010/Facial-Expression-Recognition.Pytorch.

[39] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. arXiv preprint arXiv:2106.12052, 2021.

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