CIPS-3D: A 3D-Aware Generator of GANs
Based on Conditionally-Independent Pixel Synthesis

Peng Zhou\textsuperscript{1}, Lingxi Xie\textsuperscript{2}, Bingbing Ni\textsuperscript{1}, Qi Tian\textsuperscript{2}
\textsuperscript{1}Shanghai Jiao Tong University, \textsuperscript{2}Huawei Inc.
\{zhoupengcv, nibingbing\}@sjtu.edu.cn, 198808xc@gmail.com, tian.qi@huawei.com

\begin{itemize}
\item[(a)] GIRAFFE [42]
\item[(b)] pi-GAN [11]
\item[(c)] All these images are synthesized by CIPS-3D at the resolution of $512 \times 512$.
\end{itemize}

Figure 1. Three types of 3D-aware GANs. (a): There are apparent artifacts in the images generated by GIRAFFE [42]. (b): The images generated by pi-GAN [11] are blurred and lack details. (c): CIPS-3D can generate photo-realistic high-fidelity images. We fine-tune the base model trained on FFHQ so that the transferred model can generate other types of style images. Then we interpolate the base model and the transferred model to create a new model that can generate stylized images. CIPS-3D enables one to manipulate the pose of the stylized faces (the rightmost images) explicitly. For details, please refer to Secs. 4.4 and 4.5.

Abstract

The style-based GAN (StyleGAN) architecture achieved state-of-the-art results for generating high-quality images, but it lacks explicit and precise control over camera poses. The recently proposed NeRF-based GANs made great progress towards 3D-aware generators, but they are unable to generate high-quality images yet. This paper presents CIPS-3D, a style-based, 3D-aware generator that is composed of a shallow NeRF network and a deep implicit neural representation (INR) network. The generator synthesizes each pixel value independently without any spatial convolution or upsampling operation. In addition, we diagnose the problem of mirror symmetry that implies a sub-optimal solution and solve it by introducing an auxiliary discriminator. Trained on raw, single-view images, CIPS-3D sets new records for 3D-aware image synthesis with an impressive FID of 6.97 for images at the $256 \times 256$ resolution on FFHQ. We also demonstrate several interesting directions for CIPS-3D such as transfer learning and 3D-aware face stylization. The synthesis results are best viewed as videos, so we recommend the readers to check our github project at https://github.com/PeterouZh/CIPS-3D.

1. Introduction

Generative Adversarial Networks (GANs) [20] can synthesize high-fidelity images [10, 29–33] but lack an explicit control mechanism to adjust the viewpoint for the generated object. Previous methods alleviate this problem by finding the latent semantic vectors existing in pre-trained 2D GAN models [22, 26, 52, 53]. However, these methods can only roughly change the pose of the object implicitly and fail to render the object from arbitrary camera poses. Several 3D-aware methods have been proposed to enable explicit control over the camera pose [23, 40, 63]. However, these models are often limited to low-resolution images with disappointing artifacts.

Recently, there has been a growing interest in leveraging the neural radiance fields (NeRF) [39] to build 3D-aware GANs. To our knowledge, there exist two types of 3D-aware generators: (i) using a pure NeRF network as the generator [11, 51]; (ii) generating low-resolution feature maps with a NeRF network and then upsampling with a 2D CNN decoder [5, 42]. The former suffers from a low capacity of the generator because the NeRF network is memory-intensive, which limits the depth of the generator. Thus the synthesized images are blurred and lack sharp details.
This paper presents CIPS-3D, an approach to synthesize each pixel value independently, just as its 2D version did [4]. The generator consists of a shallow 3D NeRF network (to alleviate memory complexity) and a deep 2D implicit neural representation (INR) network (to enlarge the capacity of the generator) [14, 38, 45], without any spatial convolution or up-sampling operations. Interestingly, the design of our generator is consistent with the well-known semantic hierarchical principle of GANs [8, 61], where the early layers (i.e., the shallow NeRF network in our generator) determine the pose, and the middle and higher layers (i.e., the INR network in our generator) control semantic attributes and color scheme, respectively. The early NeRF network enables us to control the camera pose explicitly.

We found that CIPS-3D suffers from a mirror symmetry problem, which also exists in other 3D-aware GANs such as GIRAFFE [42] and StyleNeRF [5]. Rather than simply attributing this issue to the dataset bias, we explain why this problem exists. Going one step further, we propose to utilize an auxiliary discriminator to regularize the output of the NeRF network, thus successfully solving this problem (see Sec. 3.3). To train CIPS-3D at high resolution, we propose a training strategy named partial gradient backpropagation. Moreover, we provide a more efficient implementation for the Modulated Fully Connected layer (ModFC) [4, 33] to speed up the training (see Sec. 3.2).

We validate the advantages of our approach on high-resolution face datasets including FFHQ [32], MetaFaces [30], BitmojiFaces [1], CartoonFaces [2], and an animal dataset, AFHQ [15]. For 3D-aware image synthesis, CIPS-3D achieves state-of-the-art FID scores of 6.97 and 12.26 on FFHQ at 2562 and 10242 resolution, respectively, surpassing the StyleNeRF [5] proposed very recently. Moreover, we verify that CIPS-3D works pretty well in transfer learning settings and show its application for 3D-aware face stylization (see Secs. 4.4 and 4.5). We will release the code to the public. We hope that CIPS-3D will serve as a good base model for downstream tasks such as 3D-aware GAN inversion and 3D-aware image editing.

2. Related Work

Implicit neural representation is a powerful tool for representing scenes in a continuous and memory-cheap way compared to mesh/voxel-based ones. It is usually implemented by a multilayer perception (MLP). The implicit representation has been widely applied in 3D tasks [14, 18, 19, 35, 38, 45, 50] as well as some 2D tasks such as image generation [4, 56] and super-resolution [13, 59]. Equipped with volume rendering [28], NeRF-based methods [7, 12, 27, 36, 39, 57, 60, 62] enable novel view synthesis by learning an implicit function for a specific scene. Recently, there has been a trend combining NeRF with GANs [6, 20, 21, 49] to design 3D-aware generators [5, 11, 16, 41, 42, 51].

Like GIRAFFE [42] and StyleNeRF [5], CIPS-3D utilizes NeRF to render features instead of RGB colors. However, our method differs from GIRAFFE and StyleNeRF in several ways. Both GIRAFFE and StyleNeRF adopt a two-stage strategy, where they render low-resolution feature maps first and then upsample the feature maps using a CNN decoder. CIPS-3D synthesizes each pixel independently without any up-sampling and spatial convolution operations. Moreover, CIPS-3D represents 3D shape and appearance with NeRF and INR networks, respectively, which is convenient for transfer learning settings.

3. Method

3.1. NeRF Network for 3D Shape

**Shallow NeRF Network** A neural radiance field [39] is a continuous function $f$ whose inputs are 3D coordinates $x = (x, y, z)$ and viewing direction $d$, and whose outputs are emitted colors $c = (r, g, b)$ and volume density $\sigma$. A multi-layer perceptron (MLP) is usually used to parameterize the continuous function $f$:

$$f : \mathbb{R}^{\text{dim}(x)} \times \mathbb{R}^{\text{dim}(d)} \rightarrow \mathbb{R}^+ \times \mathbb{R}^3, \quad (x, d) \mapsto (\sigma, c). \quad (1)$$

The 3D coordinates are sampled along camera rays. Each ray corresponds to a pixel of the rendered image. Thus to render a high-resolution image of size $H \times W$, the number of rays is large (i.e., $H \times W$). Besides, to obtain accurate 3D shape, many points need to be sampled on each ray. As a result, the NeRF network is memory-intensive.

To alleviate memory complexity, we adopt a shallow NeRF network to represent 3D shape while assigning the task of synthesizing high-fidelity appearance to a deep 2D INR network. In particular, a shallow NeRF network (see Fig. 2b) containing only three SIREN blocks [54] is used as the initial layers of the GAN’s generator.

**Modulated SIREN Block** The vanilla NeRF is restricted to a specific scene with fixed geometry. Because the generated images are various for GANs, the shape of each image is different. To render different shapes with one NeRF, we condition the NeRF network on a noise vector $z_s$ so that different shapes can be obtained by sampling $z_s$. In particular, like StyleGAN [32], we use a mapping network $m_s : Z_s \rightarrow \mathcal{W}_s$ to map $z_s$ to $w_s$, and use $w_s$ to modulate the feature maps of the NeRF network (see Fig. 2a).

We adopt the strategy of pi-GAN [11]: modulating features with FiLM [17, 47], followed by a SIREN activation function [54]. The modulated SIREN block is given by

$$\phi (x) = \sin (\gamma \cdot (W x + b) + \beta), \quad (2)$$

where $W$ and $b$ are the weight matrix and bias vector of the SIREN block.

$$f : \mathbb{R}^{\text{dim}(x)} \times \mathbb{R}^{\text{dim}(d)} \rightarrow \mathbb{R}^+ \times \mathbb{R}^3, \quad (x, d) \mapsto (\sigma, c). \quad (1)$$

The 3D coordinates are sampled along camera rays. Each ray corresponds to a pixel of the rendered image. Thus to render a high-resolution image of size $H \times W$, the number of rays is large (i.e., $H \times W$). Besides, to obtain accurate 3D shape, many points need to be sampled on each ray. As a result, the NeRF network is memory-intensive.

To alleviate memory complexity, we adopt a shallow NeRF network to represent 3D shape while assigning the task of synthesizing high-fidelity appearance to a deep 2D INR network. In particular, a shallow NeRF network (see Fig. 2b) containing only three SIREN blocks [54] is used as the initial layers of the GAN’s generator.

**Modulated SIREN Block** The vanilla NeRF is restricted to a specific scene with fixed geometry. Because the generated images are various for GANs, the shape of each image is different. To render different shapes with one NeRF, we condition the NeRF network on a noise vector $z_s$ so that different shapes can be obtained by sampling $z_s$. In particular, like StyleGAN [32], we use a mapping network $m_s : Z_s \rightarrow \mathcal{W}_s$ to map $z_s$ to $w_s$, and use $w_s$ to modulate the feature maps of the NeRF network (see Fig. 2a).

We adopt the strategy of pi-GAN [11]: modulating features with FiLM [17, 47], followed by a SIREN activation function [54]. The modulated SIREN block is given by

$$\phi (x) = \sin (\gamma \cdot (W x + b) + \beta), \quad (2)$$

where $W$ and $b$ are the weight matrix and bias vector of the SIREN block.
where $\gamma = \text{Affine}(w_x)$ and $\beta = \text{Affine}(w_z)$ represent frequency and phase, respectively. The block contains a FC layer with $W$ and $b$ as the weight matrix and the bias. As shown in Fig. 2b, the NeRF network only contains three SIREN blocks to minimize runtime memory complexity. Our experiments show that the viewing direction $d$ will cause inconsistencies of face identities under multiple views (see Fig. 11a). Thus we do not take $d$ as input, which is different from the original NeRF [39]. Furthermore, instead of predicting the color $c$, we let the NeRF network predict a more general feature $v$ [42]. As a result, the proposed NeRF function is given by

$$g: \mathbb{R}^{\dim(w)} \times \mathbb{R}^{\dim(z)} \to \mathbb{R}^+ \times \mathbb{R}^{\dim(v)}, \ (x, z_s) \mapsto (\sigma, v), \quad (3)$$

where $v$ is a feature vector corresponding to point $x$. $z_s$ is a shape code that is shared by all pixels of a generated image.

**Volume Rendering** As described in Eq. (3), the neural radiance field represents a scene as the volume density $\sigma$ and feature vector $v$ at any point in space. Let $o$ be the camera origin. For each pixel, we cast a ray $r(t) = o + td$ from origin $o$ towards the pixel. We sample points along the camera ray $r(t)$ and transform the 3D coordinates of the points into volume densities and feature vectors using Eq. (3). Using the classical volume rendering [28], the overall feature vector $V_r$ corresponding to the ray $r(t)$ is given by

$$V_r = \int_{t_n}^{t_f} T(t)\sigma(r(t))v(r(t))dt, \quad T(t) = \exp \left(- \int_{t_n}^{t_f} \sigma(r(s))ds \right), \quad (4)$$

where $t_n$ and $t_f$ are near and far bounds, respectively.

### 3.2. INR Network for Appearance

**Deep INR Network** To generate images at $H \times W$ resolution, the NeRF network outputs feature maps of shape $\dim(V_r) \times H \times W$, where $V_r$ is the feature vector calculated by Eq. (4). Next, we need to convert these feature maps into the RGB space. We adopt an implicit neural representation (INR) network, where each pixel value is calculated independently [4] given the feature vector $V_r$. As shown in Fig. 2c, the INR network contains nine blocks, and each block contains two modulated fully connected layers (ModFC) [4, 33]. We add a tRGB (i.e., a fully connected layer) layer after each block to convert the intermediate feature maps to RGB values. The final RGB values are the summation of all intermediate RGB values.
Partial Gradient Backpropagation  Our generator network is a columnar structure without any up-sampling or down-sampling operations. Therefore, directly training it on high-resolution images is challenging due to limited GPU memory. Note that the generator synthesizes each pixel value independently. Leveraging this property, we propose a training strategy named partial gradient backpropagation for training on high-resolution images.

As shown in Fig. 3, to synthesize an image at $H \times W$ resolution, we randomly sample $n_r$ rays (green rays in Fig. 3), and then convert these rays into $n_r$ RGB values, with the gradient calculation enabled. On the other hand, the remaining $H \times W - n_r$ rays (grey rays in Fig. 3) are converted to RGB values, but the gradient calculation is disabled to save memory. Finally, all the RGB values are combined into a high-resolution image, which will be presented to the discriminator. This strategy allows us to train the generator on high-resolution images. Compared with the patch-based method [4, 51], partial gradient backpropagation ensures that the discriminator observes full natural images rather than image patches at low resolution.

Efficient Implementation for ModFC  Like the NeRF network, the INR network also adopts a style-based architecture. As shown in Fig. 2d, a mapping network $m_a : \mathcal{Z}_a \rightarrow \mathcal{W}_a$ turns $\mathcal{Z}_a$ into $\mathcal{W}_a$, where the stochasticity of appearance comes from the code $\mathcal{Z}_a$. Then, $\mathcal{W}_a$ is mapped to style vectors using affine layers (i.e., FC layer). The style vectors are injected into the INR network using Modulated Fully Connected (ModFC) layers.

CIPS [4] regards ModFC as a special case of $1 \times 1$ convolutional layer and implements ModFC with the off-the-shelf modulated convolutional layer [33]. The modulated convolution is implemented using grouped convolution, which is not efficient for ModFC. In fact, we can directly utilize the batch matrix multiplication (bmmn) to implement ModFC more efficiently. As shown in Fig. 4, ModFC consists of Mod, Demod, and a batch matrix multiplication operation. Mathematically, let $W \in \mathbb{R}^{d_{in} \times d_{out}}$ be the weights of a fully connected layer, $S \in \mathbb{R}^{b \times d_{in}}$ be a batch of style vectors, and $X \in \mathbb{R}^{b \times n \times d_{in}}$ be the input with $n$ being the length of the sequence. We first resize $W$ and $S$ to shapes of $1 \times d_{in} \times d_{out}$ and $b \times d_{in} \times 1$, respectively. The Mod operation is given by $W = W \otimes S$, where $\otimes$ stands for tensor-broadcasting multiplication and $W' \in \mathbb{R}^{b \times d_{in} \times d_{out}}$. The Demod operation is given by $W'' = W' \otimes \left( \sum_{d_{in}} (W'_{d_{in}})^2 + \epsilon \right)^{-\frac{1}{2}}$, where $\epsilon$ is a small constant and $W'' \in \mathbb{R}^{b \times d_{in} \times d_{out}}$. Finally, we use $W''$ to linearly map the input $X \in \mathbb{R}^{b \times n \times d_{in}}$ (i.e., $Y = X \times W''$, and $Y \in \mathbb{R}^{b \times n \times d_{out}}$), which is achieved through the batch matrix multiplication function.1 Experiments substantiate that this implementation is more efficient than the implementation using grouped convolution (see Fig. 9).

3.3. Overcoming Mirror Symmetry Issue with Auxiliary Discriminator

In practice, we found a mirror symmetry problem for the generator composed of the NeRF network and the INR network. As shown in Fig. 5c and Fig. 6, the direction of the bangs changes suddenly near the yaw angle of $\frac{\pi}{2}$. Interestingly, GIRAFFE [42], composed of a deeper NeRF network and a CNN decoder, also has this disturbing problem (see Fig. 5a). We identify two sources for the mirror symmetry: (i) the positional encoding function [39], and (ii) the mirror symmetry of the input coordinates of the NeRF network.

Definition 1. Let $(X, d_x)$ and $(Y, d_y)$ be two metric spaces. A mapping function $f : X \rightarrow Y$ is called distance preserving if for any $a, b \in X$, one has
d_y(f(a), f(b)) = d_x(a, b) . \hspace{1cm} (6)\n
Proposition 1. A positional encoding function $T : \mathbb{R}^3 \rightarrow \mathbb{R}^{3+6L}$ is given by
$$T(x, y, z; L) = (x, y, z, \gamma(x; L), \gamma(y; L), \gamma(z; L)), \hspace{1cm} (7)$$

1The function is bmmn(·) in PyTorch.
Eq. (9) shows that after positional encoding, the distance between angle $a$ and its symmetry point $c$ is less than the distance between $a$ and its neighbor $b$. This may cause the network to predict similar appearance for $a$ and $c$, causing mirror symmetry. Therefore, we discard the fixed positional encoding function and adopt a learnable strategy, i.e., the coordinates $(x, y, z)$ are mapped to a high-dimensional space by a fully connected layer followed by a sine activation [55]. However, mirror symmetry remains.

After closer inspection, we found that the essence of mirror symmetry lies in the symmetry of the coordinate system. As shown in Fig. 6, the coordinates of $a$ and $c$ are almost the same except that the $x$ coordinate differs by a minus sign. The network tends to leverage the symmetry of coordinates to learn a symmetrical appearance. Note that the symmetry is a double-edged sword, and it facilitates the fitting of symmetrical objects [44, 58] such as human faces, cat faces, cars, etc.

We notice that pi-GAN [11], whose generator is a pure NeRF network, does not suffer from mirror symmetry (see Fig. 5b). This substantiates that the discriminator can prevent the NeRF network from falling into the pitfall of mirror symmetry. However, both GIRAFFE's generator (NeRF + INR decoder) and our generator (NeRF + INR network) suffer from mirror symmetry. It turns out that the discriminator cannot effectively regularize the NeRF network when there is a 2D network between the 3D NeRF network and the discriminator. Therefore, we propose to utilize an auxiliary discriminator to supervise the output of the NeRF network directly. In particular, the output of the NeRF network is mapped to the RGB space by a fully connected layer (see Fig. 2b), and the RGB is presented to the auxiliary discriminator. As shown in Fig. 5d, the mirror symmetry disappeared after applying the auxiliary discriminator.

4. Experiments

4.1. Implementation details

During training, we let the virtual camera be located on the surface of a unit sphere and look at the origin. The pitch and yaw are randomly sampled from predefined distributions, respectively. Both the main discriminator and the auxiliary discriminator adopt the StyleGAN2 [33] discriminator architecture, but the auxiliary discriminator has fewer channels. The model is trained with a standard non-saturating logistic GAN loss with R1 penalty [37]. We adopt Adam [34] to train the networks with $\beta_0 = 0$ and $\beta_1 = 0.999$. Following pi-GAN [11], we adopt a progressive training strategy where we start training at $64^2$ resolution and progressively increase the resolution up to $512^2$. Note that the generator architecture remains fixed, but the resolution of the generator is increased by sampling more rays. We train the model with eight Tesla V100 GPUs.
Table 1. Comparison with SOTA on FFHQ. We computed FID and KID ($\times 10^3$) between 50k generated images and all training images using the torch-fidelity library [43]. † stands for quoting from the paper. CIPS-3D achieves the state-of-the-art for 3D-aware GANs, surpassing the very recent method StyleNeRF by clear margins.

| Methods          | 256 $\times$ 256 | 1024 $\times$ 1024 |
|------------------|------------------|-------------------|
|                  | FID↑ KID↓ FID↑ KID↓ |
| 2D               |                  |                   |
| StyleGAN2 [33]   | 4.30 1.07 2.86 0.53 |
| CIPS [4]         | 23.06 23.04 10.03 4.79 |
| 3D               |                  |                   |
| GIRAFFE [42]     | 63.33 59.94 - -  |
| pi-GAN [11]      | 34.56 26.58 35.97 28.09 |
| StyleNeRF [5]    | 8.00† 3.70† - -     |
| CIPS-3D (ours)   | 6.97 2.87 12.26 7.74  |

(a) Training at 64² resolution. (b) Training at 128² resolution.

Figure 8. Progressive training on FFHQ. Generators are initially trained at 64² resolution (a), then at 128² resolution (b). pi-GAN converges steadily at low resolution, but deteriorates at higher resolution. For efficiency of training, FID is calculated between 2048 generated images and 2048 real images.

Figure 9. (a) and (b): CIPS-3D has more generator parameters, but its training speed is higher than that of pi-GAN which adopts a pure NeRF generator. (c): Compared with group conv, bmm improves the speed of ModFC considerably. Please refer to Sec. 4.2 for details.

and batch size of 32. When the resolution of the generator reaches 512 $\times$ 512, the number of rays participating in the gradient calculation is set to 400² to save GPU VRAM.

4.2. Evaluation

Comparison with SOTA We evaluate image quality using Fréchet Inception Distance (FID) [24] and Kernel Inception Distance (KID) [9]. The baseline models contain the current start-of-the-art 3D-aware GANs: GIRAFFE [42], pi-GAN [11], and StyleNeRF [5]. We also present the results of 2D GANs, such as StyleGAN2 [33] and CIPS [4], for your reference. As shown in Tab. 1, our method sets new records for 3D-aware GANs on FFHQ [32] with impressive FID scores of 6.97 and 12.26 for images at 256² and 1024² resolution, respectively. Note that our method even outperforms the StyleNeRF [5] proposed very recently, in terms of FID (6.97 vs. 8.00) and KID (2.87 vs. 3.70) at 256² resolution. However, our method is still inferior to the 2D state of the art, StyleGAN2, leaving room for improvement in future work. We present some generated images in Fig. 1. Compared to other 3D-aware GANs, CIPS-3D generates images with sharper details.

Progressively Growing Training Fig. 8 shows the FID curves over the course of progressive training. CIPS-3D is clearly better than pi-GAN, especially at higher resolution. Note that it is not easy to train pi-GAN at high resolution. First of all, the generator of pi-GAN is a pure NeRF architecture being memory-intensive. Secondly, as shown in Fig. 8b, the convergence of pi-GAN has already deteriorated in the middle resolution (i.e., 128²). Additionally, we found that pi-GAN is sensitive to hyperparameters, and increasing the depth and capacity of the generator may cause pi-GAN to fail to converge. CIPS-3D circumvents these difficulties by designing a new generator architecture consisting of a shallow 3D NeRF network and a deep 2D INR network.

Efficiency Comparison As shown in Fig. 9a and 9b, CIPS-3D has more parameters but is more efficient than pi-GAN that adopts a pure NeRF generator. To increase the number of parameters, we increase the depth of the INR network (18 layers) and the dimension of the fully connected layers (512 dimensions). Although there are more layers, the training speed of CIPS-3D is still higher than that of pi-GAN because the 2D INR network is inherently higher than the 3D NeRF network in terms of training efficiency. Fig. 9c compares the speed of ModFC using different implementation methods. Compared with group conv, bmm improves the speed of ModFC considerably (i.e., from 15.18 batches/second to 27.78 batches/second, with batch size of 4096, input/output dimension of 512). We evaluate the speed 1000 times and report the average runtime.

4.3. Ablation Studies

We conduct ablation studies to help understand the individual components. As shown in the first two rows of Tab. 2, discarding the viewing direction $\mathbf{d}$ for the NeRF network improves FID moderately. Fig. 11 shows the images generated by these two comparison methods. We deliver two messages. First, incorporating the viewing direction $\mathbf{d}$ as input leads to inconsistencies in face identity (Fig. 11a). Second, the generator will suffer from the mirror symmetry
issue regardless of whether the viewing direction $d$ is used as input or not. We have explained the reason for the mirror symmetry in Sec. 3.3 and think it is due to the inherent symmetry of the coordinate system.

The second and third rows of Tab. 2 indicate that the learnable positional encoding function (FC + sine) deteriorates FID compared to the commonly used fixed positional encoding function. After adopting the auxiliary discriminator, the problem of mirror symmetry is solved, and the FID is improved as well (the third row vs. the fourth row in Tab. 2). Since the fixed positional encoding function (i.e., Eq. (7)) is not distance preserving, it theoretically may strengthen the problem of mirror symmetry, as analyzed in Sec. 3.3. Thus we decided to adopt the learnable positional encoding function. Our final method includes using only coordinates as input, a learnable positional encoding function, and an auxiliary discriminator.

Next, we study the effect of the number of pixels participating in calculating the gradient on performance. Fig. 12 plots the FID curves over the course of training. Ablations were conducted at 128$^2$ resolution on FFHQ. It is evident that too few pixels participating in gradient backpropagation during training will harm the performance. When the number of pixels for the gradient backpropagation reaches 96$^2$, the performance is comparable to that of using all pixels (i.e., 128$^2$). In addition, we experimentally found that using a small number of pixels (e.g., 48$^2$) can still achieve the performance of using all pixels, but at the cost of more training iterations.

### 4.4. Transfer Learning

In Fig. 10, we show the output images of the NeRF network and the corresponding output images of the INR network. The INR output images share the same pose as the NeRF images but own much richer textures. This indicates that the NeRF network is responsible for the posture and the INR network is responsible for the texture. Inspired by the FreezeG [3] which freezes the early layers of the generator, we freeze the NeRF network of our generator trained on FFHQ and only fine-tune the INR network for transfer learning settings. Freezing NeRF is critical for transfer learning because there are too few images in the target domain. It is almost impossible to learn 3D shapes from so few images. However, by reusing the weights of NeRF, we only need to update the 2D INR network to render textures of other domains.

We verified the efficacy of the method on four datasets, including MetFaces [30], BitmojiFaces [1], CartoonFaces [2], and even the animal dataset, AFHQ [15]. As shown in Fig. 13a, the transferred models generate images of different domains. Moreover, we can easily control the pose of the generated faces by explicitly manipulating the NeRF network.
4.5. 3D-aware Face Stylization

We consider interpolating the original base model trained on FFHQ and the transferred models (fine-tuned on target datasets) to create new models to generate images in novel domains. Note that the weights of the NeRF network of the base model and the transferred models are completely equal. We test two types of interpolating methods: (i) linearly interpolating all INR layers, and (ii) replacing the higher layers of the INR network of the base model with the corresponding layers of the transferred model [25, 48]. The former produces images between two domains, as shown in Fig. 13c. The images smoothly fade from one domain to another. The latter generates images that combine the structural characteristics of the content images and the style characteristics of the style images (see Fig. 13b). Besides, CIPS-3D allows one to manipulate the pose of all the stylized faces explicitly.

5. Limitation and Future Work

CIPS-3D works in a noise-to-image manner. Thus current stylization is limited to randomly generated images. To edit a real image, we need to project the image to the latent space of the generator. For this purpose, we need to study 3D-aware GAN inversion, and we leave this to future work.

6. Conclusion

This paper presents a style-based 3D-aware generator that synthesizes pixel values independently without any spatial convolution or up-sampling operation. We find that the symmetry of the input coordinates leads to the problem of mirror symmetry and propose to utilize an auxiliary discriminator to solve this problem. We look forward to applying the proposed generator to more interesting applications such as 3D-aware GAN inversion and image-to-image translation.
References

[1] BitmojiFaces. https://www.kaggle.com/mostafamozafari/bitmoji-faces. 2, 7
[2] CartoonFaces. https://www.kaggle.com/aranaud58/photo2cartoon/version/1?select=trainB. 2, 7
[3] FreezeG. https://github.com/bryandlee/FreezeG. 7
[4] Ivan Anokhin, Kirill Demochkin, Taras Khakhulin, Gleb Sterkin, Victor Lemptisky, and Denis Korzhenkov. Image Generators with Conditionally-Independent Pixel Synthesis. In CVPR. 2021. 2, 3, 4, 6
[5] Anonymous. StyleNeRF: A Style-based 3D Aware Generator for High-resolution Image Synthesis. In Submitted to The Tenth ICLR. 2021. 1, 2, 6, 7
[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In ICML, 2017. 2
[7] 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
[8] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. arXiv:1811.10597 [cs], 2018. 2
[9] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In ICLR. 2021. 6
[10] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. arXiv:1809.11096, 2018. 1
[11] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. ACM SIGGRAPH Computer Graphics, 18(3), 1984. 2, 3
[12] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local Deep Implicit Functions for 3D Shape. arXiv:1912.06126 [cs], 2020. 2
[13] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas Funkhouser. Learning Shape Templates with Structured Implicit Functions. arXiv:1904.06447 [cs], 2019. 2
[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In NeurIPS, 2014. 1, 2
[15] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In NeurIPS, 2017. 2
[16] Erika Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. GANSpace: Discovering Interpretable GAN Controls. In NeurIPS, 2020. 1
[17] Philipp Hennzerl, Niloy Mitra, and Tobias Ritschel. Escaping Plato’s Cave: 3D Shape From Adversarial Rendering. In ICCV, 2019. 1
[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In NeurIPS, 2017. 6
[19] Jialu Huang, Jing Liao, and Sam Kwong. Unsupervised Image-to-Image Translation via Pre-trained StyleGAN2 Network. arXiv:2010.05713 [cs], 2020. 8
[20] Ali Jahanian, Lucy Chai, and Phillip Isola. On the “steerability” of generative adversarial networks. arXiv:1907.07171 [cs], 2020. 1
[21] Wonbong Jang and Lourdes Agapito. CodeNeRF: Disentangled Neural Radiance Fields for Object Categories. In ICCV, 2021. 2
[22] Sylvain Paris. GANSpace: Discovering Interpretable GAN Controls. arXiv:2106.12423 [cs, stat], 2021. 2
[23] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. arXiv:1710.10196 [cs, stat], 2017. 1
[24] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training Generative Adversarial Networks with Limited Data. In NeurIPS, 2020. 1, 2, 7
[25] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-Free Generative Adversarial Networks. arXiv:2106.12423 [cs, stat], 2021. 1, 2
[26] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In CVPR, 2019. 1, 2, 6
[27] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN. arXiv:1912.04958 [cs, eess, stat], 2019. 1, 2, 3, 4, 5, 6
[28] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In ICLR, 2014. 5
