Lifting 2D StyleGAN for 3D-Aware Face Generation

Yichun Shi, Divyansh Aggarwal, Anil K. Jain
Michigan State University
shiyichu@msu.edu, aggarw49@msu.edu, jain@cse.msu.edu

(a) StyleGAN2 (b) Shape (c) Rotation (d) Relighting

Figure 1: Example results of LiftedGAN. Column (a) shows three random samples from the latent space of a pre-trained StyleGAN2 network. Columns (b)-(d) show the results of LiftedGAN. The proposed method lifts a pre-trained StyleGAN2 to a 3D generator by predicting additional depth information, which allows it to not only generate realistic face images, but also provides 3D control of the output, such as rotation and relighting. Our method does **NOT** need any annotation nor 3DMM model for training.

Abstract

We propose a framework, called LiftedGAN, that disentangles and lifts a pre-trained StyleGAN2 for 3D-aware face generation. Our model is "3D-aware" in the sense that it is able to (1) disentangle the latent space of StyleGAN2 into texture, shape, viewpoint, lighting and (2) generate 3D components for rendering synthetic images. Unlike most previous methods, our method is completely self-supervised, i.e. it neither requires any manual annotation nor 3DMM model for training. Instead, it learns to generate images as well as their 3D components by distilling the prior knowledge in StyleGAN2 with a differentiable renderer. The proposed model is able to output both the 3D shape and texture, allowing explicit pose and lighting control over generated images. Qualitative and quantitative results show the superiority of our approach over existing methods on 3D-controllable GANs in content controllability while generating realistic high quality images.

1. Introduction

Generative Adversarial Networks (GANs), such as StyleGAN [13, 14] have been demonstrated to generate high quality face images with a wide variety of styles. However, since these models are trained to generate random faces, they do not offer direct manipulation over the semantic attributes such as pose, expression etc. in the generated image. A number of studies have been devoted to achieving control over the generation process in order to be able to adjust pose and other semantic attributes in the generated face images. Among all these attributes, 3D information, such as pose, is the most desirable due to its applicability in face recognition [29] and face synthesis [34]. To achieve this, most existing approaches attempt to disentangle the latent feature space of GANs by leveraging external supervision on the semantic factors such as pose labels [29, 28], landmarks [9] or synthetic images [2, 35, 17], while some others [20] have explored an unsupervised approach for 3D controllability in the latent space. Although these feature manipulation based methods have shown ability to generate faces with high visual quality under assigned poses, it is unclear whether important content, such as identity, is indeed preserved when we change the pose parameters. Potential errors could arise from the generation process when the manipulated features are parsed by the network parameters (see Section 5.2).

In contrast to solutions that only output 2D images, building generative models with explicit 3D shapes could give

Study	Generative	Shape	Lighting	Supervision
Tulsiani *et al*. [30]		✓		Multi-view images
Kanazawa *et al*. [12]		✓		Keypoints, Silhouette
Gadelha *et al*. [4]	✓	✓		Silhouette
Henderson *et al*. [5, 6, 7]	✓	✓		Viewpoint, Silhouette
Lunz *et al*. [18]	✓	✓		None
Wu *et al*. [31]		✓	✓	None
Zhang *et al*. [33]		✓	✓	Viewpoint
Pan *et al*. [21]		✓	✓	None
Szabo *et al*. [26]	✓	✓		None
CONFIG [17]	✓		✓	Synthetic data
HoloGAN [20]	✓		✓	None
StyleRig [27]	✓		✓	3DMM
DiscoFaceGAN [2]	✓		✓	3DMM
Ours	✓	✓	✓	None

Table 1: Difference between our work and related work. The first half shows relevant work on unsupervised and weakly-supervised 3D reconstruction. The second half are relevant studies on 3D-controllable face generation.

a stricter control of the content. For general 3D objects, a thread of recent studies train 3D generative models from 2D images, but they mostly work on rendered images with coarse shapes [6, 7, 18], e.g. cars. For faces, which contain many fine-grained details, existing solutions for 3D image generation [23, 26] have suffered from collapsed results under large pose variations due to the difficulty of learning reasonable shapes.

In this paper, we propose a framework that shares the advantages of both 2D and 3D solutions. Given a pre-trained StyleGAN2, we distill it into a 3D-aware generator, which not only outputs the generated image, but its view points, light direction and 3D information, such as surface normal map. Compared with 3D generative models, our approach is able to output rendered images with higher quality, close to 2D generative models. Compared with 2D solutions based on feature manipulation, our model allows a stricter 3D control over the content by maneuvering the view point and shading of textured meshes, as in 3D generators. Qualitative and quantitative results show the superiority of our approach over existing methods in preserving identity as well as generating realistic high quality images.

The main contributions of the paper can be summarized as follows:

- A framework for 3D-aware face image generation, called LiftedGAN, which distills the knowledge from a pre-trained StyleGAN2.
- A self-supervised method for disentangling and distilling the 3D information in the latent space of StyleGAN2.
- A generator that outputs both high quality face images and their 3D information, allowing explicit control over pose and lighting.

2. Related Work

2.1. Pose-Disentangled 2D GANs

Recent progress in Generative Adversarial Networks has enabled generation of high-quality realistic images. This has resulted in a body of work to disentangle different factors of the generated images from GANs. These publications can be categorized into two types. The first type explicitly adds additional modules or loss functions during training to ensure the disentanglement of pose information. For example, Tran *et al*. [29] and Tian *et al*. [28] use pose labels while Hu *et al*. [9] and Zhao *et al*. [35] use landmarks and a 3DMM model, respectively to guide the training of an image-translation GAN for rotating input faces. For the generation task, Deng *et al*. [2] use a 3DMM model during GAN training to guide the learning of a disentangled pose factor. CONFIG [17] mixes real face images and synthesized images from a graphic pipeline with known parameters for training the GAN. HoloGAN [20], on the other hand, proposes to use a 3D feature projection module in the early stage of the generator to enable the rotation of the output face images. The second type, on the other hand, tries to manipulate a pre-trained GAN network to change the 3D information of the output. This is built upon the foundation that recent GANs [13, 14] provide a naturally disentangled latent space for generation. Similar to training-based methods, these studies use labels [24] or a 3DMM [27] to achieve the disentanglement of the pre-trained latent space. Shen *et al*. [25] have also proposed an unsupervised method to factorize the latent space of GANs. A clear drawback of all these methods is that they could not explicitly output the 3D shape of the object in the generated images, which is essential for strict 3D control over the content.

2.2. Unsupervised 3D Reconstruction and Generation from 2D Images

In contrast to the 2D-based solutions above, several studies have explored the possibility of reconstructing and generating 3D shapes from unlabeled 2D images. Here, "unlabeled" means that neither 3D shape nor view label is available during the training of the model. Such unsupervised reconstruction methods typically use a special cue to guide the learning of the 3D shape. For example, Tulsiani *et al*. [30] use the muiti-view consistency as the supervision; Kanazawa *et al*. [12] use the consistency between objects under the same category to learn the reconstruction model. Wu *et al*. [31] use the symmetry property of the objects to learn detailed shape and albedo from natural images. Handerson *et al*. [5, 6, 7] exploit the shading information from the synthesized images to reconstruct 3D meshes, which is further extended to a generative model by using a VAE structure. Most generative models, on the other hand, use a GAN-structure where adversarial loss provides the sig-

nal for 2D images rendered from the generated 3D shapes. Gadelha *et al*. [4] apply the discriminator to the silhouette of the generated voxels to train the generator. Lunz *et al*. [18] use a commercial renderer to guide a neural renderer to output images with shading for the discriminator. However, both methods utilize voxels, which cannot recover the fine-grained details nor the colors of the 3D surfaces. Recently, Szabo *et al*. [26] proposed to use vertex position maps as the 3D representation to directly train the GAN with textured mesh outputs. However, given the large degrees of freedom of such representation and the noisy signals from adversarial training, the output shapes of their work suffer from strong distortion. Concurrent with our work, Zhang *et al*. [33] and Pan *et al*. [21] have utilized StyleGAN to generate multi-view synthetic data for 3D reconstruction tasks. Zhang *et al*. [33] conduct manual annotation on offline-generated data while Pan *et al*. [21] propose to iteratively synthesize data and train the reconstruction network. Different from their work, our work builds a 3D generative model by simultaneously learning to manipulate StyleGAN2 generation and estimate 3D shapes.

3. Methodology

The main idea of our method is to train a 3D generative network by distilling the knowledge in StyleGAN2. The StyleGAN2 network is composed of two parts, a multi-layer perceptron (MLP) that maps a latent code $z \in \mathcal{Z}$ to a style code $w \in \mathcal{W}$ and a 2D generator G_{2D} that synthesizes a face image I from the style code w. Our goal is to build a 3D generator that disentangles the generation process of G_{2D} into different 3D modules, including albedo, shape, lighting and pose. These modules are then utilized to render a 2D face image.

3.1. 3D Generator

As shown in Figure 2, the 3D generator, denoted as G_{3D} is composed of five additional networks: D_V, D_L, D_S, D_T and M. Given a randomly sampled style code $\hat{w}$, the network D_V is a MLP that maps $\hat{w}$ to a 6-dimensional viewpoint representation V, including translation and rotation. D_L is another MLP that decodes $\hat{w}$ into a 4-dimensional output L: the x-y direction of the light, ambient light and diffuse light intensity. The style manipulation network M is a core module in our work. It serves to transfer a style code $\hat{w}$ to a new style code with specified light and view. In particular for 3D generator, it is used to create $w_0 = M(\hat{w}, L_0, V_0)$ where L_0 and V_0 are default parameters of neutralized lighting and viewpoint. Thus, StyleGAN2 $G_{2D}(w_0)$ outputs a neutralized face serving as texture map, as shown in Figure 2. This neutralized face is then de-lighted by L_0 under a Lambertain model to obtain the albedo map A. D_S and D_T are two deconvolution networks that map the disentangled style code w_0 to the shape representation S and a transformation map T, which are further explained in Section 3.1.1. Finally, a differentiable renderer R is used to output a rendered image $I_w = R(A, S, T, V, L)$.

3.1.1 Shape Representation

Prior work on 3D generation [26, 7] use 3D position maps to represent the mesh of the target object. The advantage of such an approach is that it could possibly disentangle the foreground and background. However, specifically for face, whose contour is often ambiguous given the irregular shape of hair, we found that forcing a separated foreground could easily lead to collapsed shapes on the boundary, which is also observed in [26]. Therefore instead, we use a depth map with a transformation map to represent the shape. The depth map is associated with the texture map to represent their positions, while the transformation map decides how much each pixel should be transformed when we rotate the face. Formally, during the face manipulation, for each pixel (i, j), whose original position and target position is given by $p_{i,j}^{(old)}$ and $p_{i,j}^{(tgt)}$, the new target position of the pixel using the transformation map would be:

$$p_{i,j}^{(new)} = (1 - T_{i,j})p_{i,j}^{(old)} + T_{i,j}p_{i,j}^{(tgt)}, \tag{1}$$

where $T_{i,j}$ is the corresponding value of pixel (i, j) on the transformation map T. All the foreground will be assigned with $T_{i,j} = 1$, while border pixels are forced to have 0 transformation freedom. The usage of transformation map allows us to dynamically transform the pixels to obtain a complete image without disentangling the background. See Figure 4 (d) for example effect of transformation map.

3.2. Loss Functions

Unlike other GAN-based methods [26], our method does not need any adversarial training. Since the relationship between the style space $\mathcal{W}$ and the image space are fixed by G_{2D}, we can directly use G_{2D} as a teacher model to supervise the G_{3D} for learning the image mapping. Then, we further use the symmetric constraint and multi-view data created by G_{2D} to enable the learning of 3D shape.

3.2.1 Reconstruction Loss

Let $\hat{w}$ be a randomly sampled code and $\hat{I}_w = G_{2D}(w)$ is a proxy image output by StyleGAN2. The rendered image is given by $I_w = R(A, S, T, V, L)$. The reconstruction loss for each sample is then defined as:

$$\mathcal{L}_{rec} = \left\| I_w - \hat{I}_w \right\|_1 + \lambda_{perc}\mathcal{L}_{perc}(I_w, \hat{I}_w), \tag{2}$$

where the second term refers to the perceptual loss [11] using a pre-trained VGG-16 network. Inspired by Wu et al. [31], we also add an additional reconstruction loss with

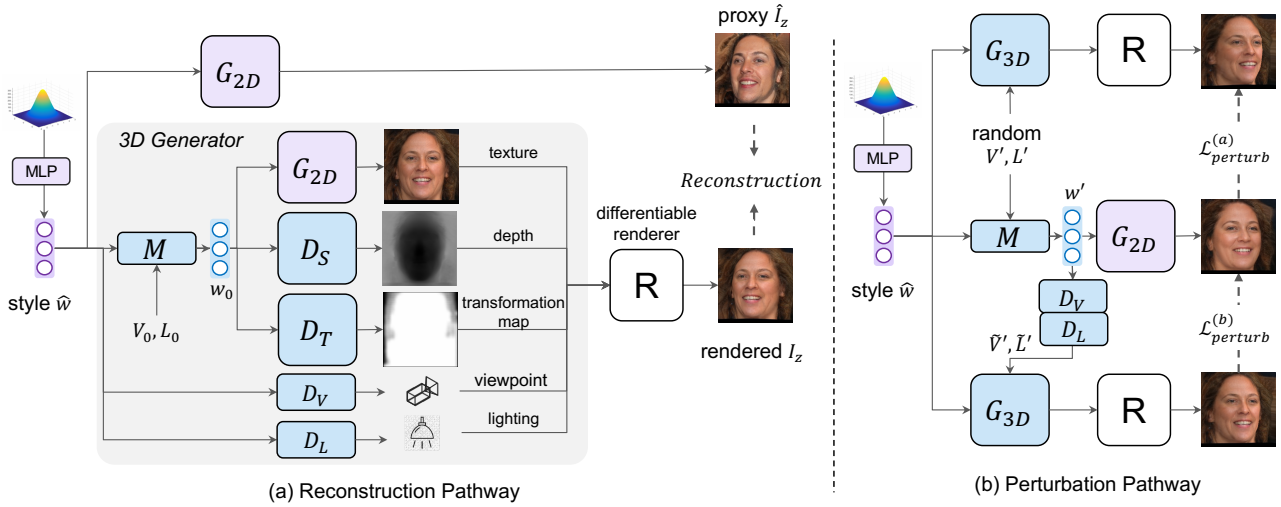


Figure 2: Overview of the proposed training framework for learning 3D generator. The framework mainly involves two pathways. First, for each randomly sampled image from StyleGAN2, we train the 3D generator to disentangle its latent code into 3D components and reconstruct the 2D image with symmetric constraint. Second, we randomly perturb the generated 3D face to obtain regularization from different views. This is achieved by simultaneously training the 3D components along with the style manipulation network, which creates pseudo ground-truth via the 2D generator to regularize the 3D face. The purple blocks in the figure indicate modules from the pre-trained StyleGAN2 and are not updated. The blue blocks are the modules to be trained. The texture images are relighted for rendering.

Figure 3: Example images for Section 3.2.2. The first row shows generated images of StyleGAN2 by manipulating the same style code with style manipulation network M. The second row shows rotated images with the differentiable render. The generated images of StyleGAN2 provides pseudo ground-truth for multi-view supervision. Note that during training, we only perturb once for each image.

flipped shape and albedo map to reconstruct the proxy image $\hat{I}_w$, which we denote as L_{flip}. We found such a symmetric regularization to be very helpful to construct a frontalized face as texture map.

3.2.2 Generator as Multi-view Supervision

The proxy images generated by G_{2D} provide supervision for learning the mapping from latent space to image space. However, learning 3D shapes remains an ill-posed problem. Although we employ the symmetric reconstruction loss [31], we found it insufficient to regularize the output shape, possibly due to the larger area of background and higher resolution of our training data. Here, since we already have a pre-trained generator, we utilize the G_{2D} as a multi-view data generator to provide supervision for rotated-views of the 3D face. This approach is inspired by recent findings that StyleGAN2 is able to generate different views of a target sample by changing its style code [24, 25, 33].

Formally, for each rendered sample $I_w = R(A, S, T, V, L)$, we randomly sample a different view point V' and lighting L' to render a rotated and relighted face image $I'_w = R(A, S, T, V', L')$. The objective here is to maximize the likelihood of this rotated face, $\log p(I'_w)$ for any random V' and light L' to make sure it look like a real face. Assuming the 2D generator is well trained to approximate the real data distribution, we could use the generator to estimate this likelihood. However, directly optimizing the likelihood is non-trivial, since it involves marginalizing over the latent space of G_{2D}. Here, given the style manipulation network M, for each perturbed image I'_w we have a corresponding $w' = M(\hat{w}, V', L')$, where we manipulate the original face in the latent space. If both the network M and the shape decoder D_S have well learned, I'_w and w' should match each other after perturbation. Thus, we optimize the joint probability $p(I'_w, w') = p_{G_{2D}}(I'_w|w')p(w')$, which is equivalent to minimizing the following loss function:

$$\begin{aligned} \mathcal{L}'_{perturb} == & -\log p_{G_{2D}}(I'_w|w') - \log p(w') \\ & = d(I'_w, G_{2D}(w')) + \beta \frac{\|w' - \mu_w\|^2}{2\sigma_w^2} \end{aligned} \tag{3}$$

Here, the prior $p(w')$ is approximated by a Gaussian distribution $\mathcal{N}(w; \mu_w, \sigma_w^2 \mathbf{I})$, where empirical mean μ_w and standard deviation σ_w of randomly generated style codes are used. Equation (3) can be understood as follows: for a randomly perturbed image $R(A, S, T, V', L')$, we use StyleGAN2 to synthesize proxy images $G_{2D}(w')$ to provide pseudo ground-truth for the target view and lighting while we are training the network M to learn geometric warping and relighting in the latent space. The second term regularizes the transferred w to be close to the prior distribution to ensure the quality of generated images. We note that $\log p(I'_w, w')$ can also be regarded as an approximation of the variational lower bound of $\log p(I'_w)$ (see supplementary). Thus, we are indirectly forcing each perturbed image to look realistic.

In practice, we found that directly optimizing Equation (3) would flatten the output shape. Given the outputs of G_{2D}, we observe that the problem is caused by (1) pose difference between the generated image $G_{2D}(w)$ and the target V', (2) the incapability of G_{2D} to synthesize images with diverse lighting. Thus, inspired by Pan *et al.* [21], we implement Equation (3) as two parts. The original I'_w is only used for optimizing the style manipulation network w' while A, S and T are optimized with re-estimated $\tilde{V}' = D_V(w')$ and $\tilde{L}' = D_L(w')$. The loss is given by:

$$\begin{aligned}
\mathcal{L}_{perturb} &= \mathcal{L}^{(a)}_{perturb} + \mathcal{L}^{(b)}_{perturb} \\
\mathcal{L}^{(a)}_{perturb} &= d(I'_w, G_{2D}(z_{perturb})) + \beta \frac{\|w' - \mu_w\|^2}{2\sigma_w^2} \\
\mathcal{L}^{(b)}_{perturb} &= d(R(A, S, T, \tilde{V}', \tilde{L}'), \hat{I}_{w'}) + \lambda_{LV_{cyc}} \mathcal{L}_{LV_{cyc}},
\end{aligned} \tag{4}$$

where

$$\mathcal{L}_{LV_{cyc}} = \left\| \tilde{V}' - V' \right\|^2 + \left\| \tilde{L}' - L' \right\|^2, \tag{5}$$

$I'_w = R(A, S, T, V', L')$ and $\hat{I}_{w'} = G_{2D}(w')$ are used as proxy images that are not updated. Our method shares the similar concept with Zhang *et al.* [33] and Pan *et al.* [21], which use StyleGAN2 to create synthetic training data. However, different from them, we do not need any manual annotation [33] nor iterative training [21]. An illustration of Equation 4 can be found in Figure 2. Figure 3 shows example proxy images generated by StyleGAN2 and rendered images with re-estimated parameters.

3.2.3 Regularization Losses

We also add a few regularization losses to constrain the output of our model. First, assuming that the identity shouldn't change after viewpoint perturbation, we regularize the identity variance loss:

$$\mathcal{L}_{idt} = \left\| f(I_{w_0}) - f(I'_w) \right\|^2, \tag{6}$$

where I_{w_0} is the texture map and f is a pre-trained face recognition network. Since the recognition network is not guaranteed to be pose invariant, we only apply this to images that are rotated within a certain range.

Second, in order to better utilize the shading information, we regularize the albedo maps, which is acquired by delighting the canonical face image:

$$\mathcal{L}_{reg_A} = \|K_A\|_*. \tag{7}$$

Here $K_A \in \mathbb{R}^{B \times HW}$ is the albedo matrix that is composed of filtered and vectorized albedo maps and $\|\cdot\|_*$ denotes the nuclear norm. Laplacian kernel is used for filtering the gray-scaled albedo maps to only keep the high-frequency information. The nuclear norm is a soft approximation of low-rank regularization, which enforces different albedo maps in a batch to have smaller laplacians while encouraging consistency across samples.

The overall loss function for training the 3D generator is:

$$\mathcal{L}_{G_{3D}} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{flip}\mathcal{L}_{flip} + \lambda_{perturb}\mathcal{L}_{perturb} \tag{8}$$

$$\lambda_{idt}\mathcal{L}_{idt} + \lambda_{reg_A}\mathcal{L}_{reg_A} \tag{9}$$

4. Implementation Details

We implement all the modules in this paper using Pytorch 1.5. The mesh rasterizer in Pytorch3D [22] is used for differentiable rendering. The StyleGAN2 [14] is trained on FFHQ dataset [13] with a Pytorch re-implentation [1]. The training images are cropped and resized to 256x256 with the MTCNN face detector [32]. The 2D StyleGAN2 is then used to train the 3D generator in the second stage. The hyper-parameters λ_{rec}, λ_{perc}, λ_{flip}, $\lambda_{perturb}$, β, $\lambda_{LV_{cyc}}$, λ_{idt} and λ_{reg_A} are set to 5.0, 1.0, 0.8, 2.0, 0.5, 2.0, 1.0 and 0.01, respectively. These hyper-parameters are chosen based on the qualitative results of generated samples during training. Due to the space limit, more implementation details, including the network architectures are provided in the *supplementary material*.

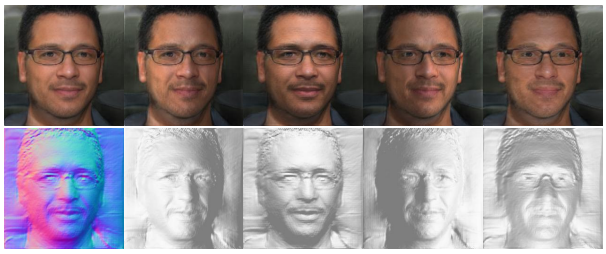
5. Experiments

5.1. Qualitative Results

Figure 4 shows a few examples of controlling the pose of the generated face image. The first row shows the results of manipulating the style code by network M, while the following rows show the results of our 3D generator. It could be seen that although latent manipulation is able to change the viewpoint of a specific face, it fails to generalize to larger poses, that are rarely seen in the training data. In contrast, the 3D generator, after distilling the 3D shape, is able to generalize to larger poses. More comparison between the style manipulation and 3D generator can be found in

[1] https://github.com/rosinality/stylegan2-pytorch

Figure 4: Rotation of generated faces. Row (a) shows the results of manipulating the style code with network M. Row (b) and row (c) show the manipulation results of the 3D generator. Row (d) shows the rotation process from a side view.



(a) Original (b) Left (c) Top (d) Right (e) Bottom

Figure 5: Example generated faces with different lighting. Column (a) shows a randomly sampled image and its normal map. Columns (b)-(d) show relighted images and their corresponding shading by changing the source direction of light.

the supplementary material. The last row shows the side view to see how the face is rotated. With the self-predicted transformation map, only the foreground is moving while the background remains relatively static.

Figure 5 shows an example generated image with different lighting conditions. By computing the normal map from the generated shape, we are able to generate face images with arbitrary lighting conditions, even those that were not seen in the original training dataset. In the second row of Figure 5, we show how the lighting from left, top, right and bottom creates the shading map, which results in the relighted images in the first row.

Figure 6 shows the shapes of a few virtual faces from different viewpoints. By using lighting and shading, our model is able to estimate not only the coarse structure of the face, but also fine-grained details such as hair and teeth.

A successful generative model should be able to provide

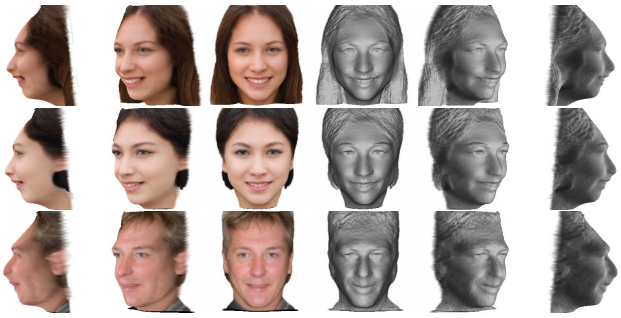


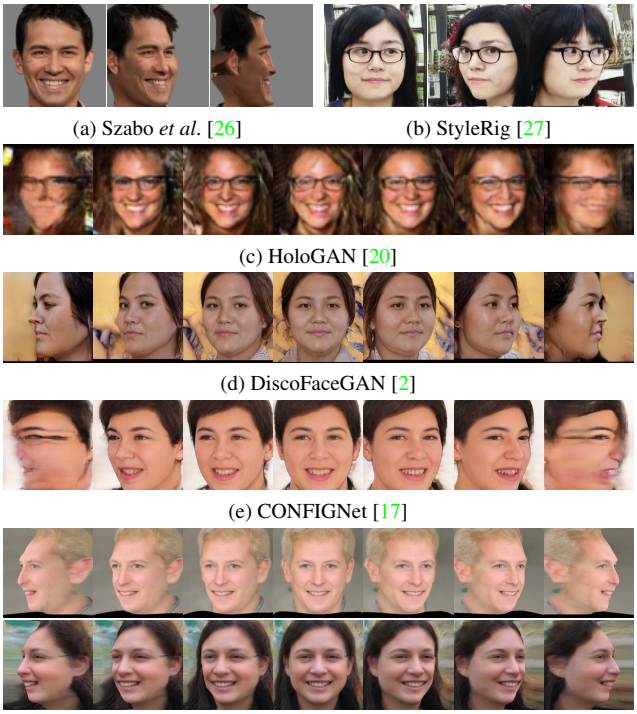
Figure 6: Visualization of 3D faces sampled from the latent space. In each row, we show the 3D shape with (left) and without (right) texture under different view angles. The background is clipped by depth for visualization purpose.

Figure 7: Example generation results between interpolated latent codes. Our model is able to achieve a smooth change between two disparate samples, indicating its potential to generate a diverse set of controllable face images.

smooth interpolated results between disparate samples to indicate that the model is not simply memorizing training samples [13]. In Figure 7, we show interpolated results between two faces. It could be seen that, by inheriting the style-image mapping from the 2D generator, our model is able to smoothly move from one face to another while generating realistic faces. This indicates that our model could potentially be used to generate a diverse set of faces with more effective samples compared to the original training set.

5.2. Comparison with Related Work

Figure 8 shows some generation results of state-of-the-art work on 3D-controllable GANs. Szabo *et al*. [26] used a purely GAN-based model, where the discriminator is the main source of supervision. Therefore, they are able to generate realistic rendered images. However, their 3D shapes suffer from strong distortion, potentially due to the unstable nature of adversarial loss. This could be clearly observed from the side view of their 3D shapes. In comparison, our 3D shapes are realistic even from side views. Similarly, Holo-



(a) Szabo *et al*. [26] (b) StyleRig [27]

(c) HoloGAN [20]

(d) DiscoFaceGAN [2]

(e) CONFIGNet [17]

(f) Ours

Figure 8: Qualitative comparison with state-of-the-art methods on 3D-controllable GANs. Note that all these faces are generated by randomly sampling from latent space, therefore we cannot compare the manipulation over the same face. The example faces in (c)-(f) are supposed to have a yaw degree of -60,-30,15,0,15,30,60.

GAN [26] is another unsupervised method based on adversarial loss. But since their 3D transformation is applied implicitly in the feature space, it is not able to provide explicit shapes of the generated images, and hence it fails to generate faces of poses that are not in the original training data. StyleRig [27], CONFIGNet [17] and DiscoFaceGAN [2] are three recent methods that attempt to disentangled 2D GAN with the supervision of 3DMM or other 3D engines. The difference is, StyleRig tries to disentangle a pretrained StyleGAN, as our method, while CONFIGNet and DiscoFaceGAN train a new generator from scratch with additional data generated by 3D models. Similar to HoloGAN, they only output the images but no other 3D components. From Figure 8, it can be seen that our model, though unsupervised, is able to generalize to larger yaw degrees than most of the baselines, which are rarely seen in the training data. More results can be found in the supplementary.

We further quantitatively compare our approach with three open-sourced methods: HoloGAN [20], DiscoFaceGAN [2] and CONFIGNet [17] in Figure 9. For each method, we randomly generate $1,000$ faces, each with multiple outputs under specified poses. Then a state-of-the-art face matcher, ArcFace [1] is used to compute the similarity between the generated frontal face and non-frontal faces. As can be seen in Figure 9, in spite of the good visual quality of all methods, the identity is hardly preserved after the face rotation except DiscoFaceGAN, which is trained with the supervision of a 3DMM. In comparison, our method is able to maintain the identity information better than most methods even without any supervision.

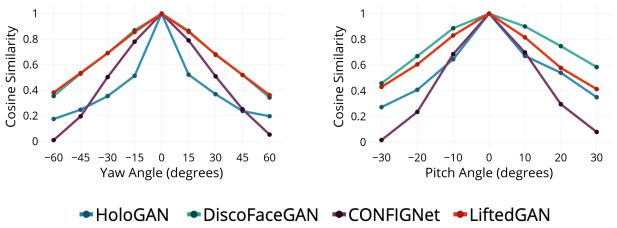


Figure 9: Quantitative comparison of the proposed LiftedGAN with state of the art works on 3D-controllable generative models. We compute the averaged identity similarity under different rotation angles using a face recognition method (ArcFace).

5.3. Ablation Study

In this section, we ablate over different loss functions to see their effect on the generated results. In particular, we remove symmetric reconstruction loss, perturbation loss, albedo consistency loss and identity regularization loss, respectively, to re-train a model for comparison. The evaluation is conducted from two perspectives. First, Fréchet inception distance (FID) [8] is used to evaluate the image quality, where we compare between the original 2D training data and randomly sampled images from our 3D Generator. Second, we also wish to evaluate how precise the estimated 3D shapes are. However, since our models are generative methods, for whose output we do not have ground-truth 3D annotations, it is hard to directly evaluate the 3D outputs. Therefore instead, we compute the estimated 3D shapes of $1,000$ generated images of StyleGAN2 with a state-of-the-art 3D reconstruction model [3] as pseudo ground-truth and compare them to the estimated shapes of our model. In detail, we evaluate both depth map error and angle prediction error. For depth map, since different camera parameters are used for our method and the estimated labels, we first normalize the depth map in terms of mean and standard deviation before computing their ℓ_2 distance on overlapped areas. The view angles are normalized by subtracting average angle. To provide a reference, we also train a state-of-the-art unsupervised face reconstruction model [31] on the StyleGAN2 generated faces and test it on the same set of images. The quantitative and qualitative results are shown in Table 2 and Figure 10, respectively. Without the symmetric reconstruction loss, we found that the model is not able to output a reasonable shape, even though it is still able to rotate the face to a certain degree. The perturbation loss is helpful for

Method	Image FID	3D Depth	3D Yaw	3D Pitch	3D Row
StyleGAN2 (G_{2D})	12.72	-	-	-	-
Wu *et al.* [31]	-	0.472	2.78	4.55	0.75
LiftedGAN w/o $\mathcal{L}_{flip}$	28.69	0.623	9.37	5.34	0.91
LiftedGAN w/o $\mathcal{L}_{perturb}$	21.30	0.548	2.68	5.03	1.02
LiftedGAN w/o $\mathcal{L}_{idt}$	30.63	0.498	1.82	4.66	1.15
LiftedGAN w/o $\mathcal{L}_{reg_A}$	27.34	0.467	1.89	4.81	1.06
LiftedGAN (proposed)	29.81	0.435	1.86	4.77	1.07

Table 2: Quantitative Evaluation of the ablation study. LiftedGAN refers to our method while StyleGAN2 refers to the output of base 2D GAN used for training. The 3D evluation is conducted on samples generated by StyleGAN2. The depth error is the ℓ_2 distance between normalized depth maps and the angle errors are reported in degrees. FID represents the Fréchet inception distance.

Figure 10: Qualitative results of ablation study.

learning detailed and realistic face shapes, especially from a side view. The albedo consistency loss and identity loss has a similar effect of regularizing the output shape, as can be seen in Table 2. We also notice that applying the perturbation loss cause a higher FID, we believe this is caused by the identity change in the style-manipulated faces created by StyleGAN2, whose supervision removes some details on the generated 3D faces. The problem could be potentially solved by adding additional regularization loss to the StyleGAN2 manipulation output, which we leave for future work.

5.4. Decoupling Structure and Style

The original StyleGAN2 is famous for being able to control and transfer the style of generated images [13, 14]. However, this notion of "style" is rather ambiguous: it could mean coarse structure (e.g. pose), fine-grained structure (e.g. expression), or color style. This is because the style control is learned in an unsupervised manner by AdaIN modules [10] at different layers. Since our work explicitly disentangles the 3D information, it implies that the 3D geometry and

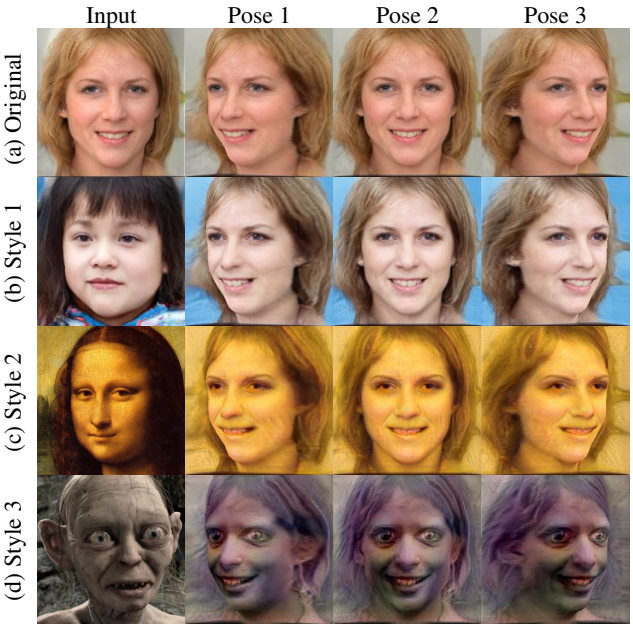


Figure 11: Example results of style transfer. Row (a) shows a randomly sampled image and its different views. Rows (b)-(d) show the results of transferring the style to images in row (a) while changing the pose of the generated face. The first column in rows (b)-(d) shows the input images used to extract the style.

other styles could be decoupled in our generator. Figure 11 shows a few examples of transferring different styles to a generated face while controlling the pose of that face. It could be seen that style transfer only affects the texture color and local structures, while the overall structure of the image is consistent across different poses, as controlled by the user.

6. Conclusions

We propose a method to distill a pre-trained StyleGAN2 into a 3D-aware generative model, called LiftedGAN. LiftedGAN disentangles the latent space of StyleGAN2 into pose, light, texture, a depth map and a transformation map from which face images can be generated with a differentiable renderer. During training, the 2D StyleGAN2 is used as a teacher network to generate proxy images to guide the learning of the 3D generator. The learning process is completely self-supervised, i.e. it neither requires 3D supervision nor 3D morphable model. During inference, the 3D generator is able to generate face images of selected pose angles and lighting conditions. Compared with 3D-controllable generative models that are based on feature manipulation, our model gives explicit shape information of generated faces and hence preserves the content better during face rotation. Compared with 3D generative models, our model provides more realistic face shapes by distilling the knowledge from StyleGAN2. Potential future includes: (1) combining our method with the training of StyleGAN2 to obtain a more

disentangled 2D generator and (2) extending the method to other object domains by incorporating limited labels.

References

[1] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019. 7

[2] Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3d imitative-contrastive learning. In *CVPR*, 2020. 1, 2, 7, 11

[3] 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 *CVPR Workshops*, 2019. 7

[4] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *International Conference on 3D Vision*, 2017. 2, 3

[5] Paul Henderson and Vittorio Ferrari. Learning to generate and reconstruct 3d meshes with only 2d supervision. In *BMVC*, 2018. 2

[6] Paul Henderson and Vittorio Ferrari. Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. *IJCV*, 2019. 2

[7] Paul Henderson, Vagia Tsiminaki, and Christoph H Lampert. Leveraging 2d data to learn textured 3d mesh generation. In *CVPR*, 2020. 2, 3

[8] 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. 7

[9] Yibo Hu, Xiang Wu, Bing Yu, Ran He, and Zhenan Sun. Pose-guided photorealistic face rotation. In *CVPR*, 2018. 1, 2

[10] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 8

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 3

[12] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 2

[13] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 1, 2, 5, 6, 8

[14] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 1, 2, 5, 8

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 10

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 10

[17] Marek Kowalski, Stephan J Garbin, Virginia Estellers, Tadas Baltrušaitis, Matthew Johnson, and Jamie Shotton. Config: Controllable neural face image generation. In *ECCV*, 2020. 1, 2, 7, 11

[18] Sebastian Lunz, Yingzhen Li, Andrew Fitzgibbon, and Nate Kushman. Inverse graphics gan: Learning to generate 3d shapes from unstructured 2d data. *arXiv preprint arXiv:2002.12674*, 2020. 2, 3

[19] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. 10

[20] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *CVPR*, 2019. 1, 2, 7, 11

[21] Xingang Pan, Bo Dai, Ziwei Liu, Chen Change Loy, and Ping Luo. Do 2d gans know 3d shape? unsupervised 3d shape reconstruction from 2d image gans. *arXiv:2011.00844*, 2020. 2, 3, 5

[22] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 5

[23] Mihir Sahasrabudhe, Zhixin Shu, Edward Bartrum, Riza Alp Guler, Dimitris Samaras, and Iasonas Kokkinos. Lifting autoencoders: Unsupervised learning of a fully-disentangled 3d morphable model using deep non-rigid structure from motion. In *ICCV Workshops*, 2019. 2

[24] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, 2020. 2, 4

[25] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. *arXiv:2007.06600*, 2020. 2, 4

[26] Attila Szabó, Givi Meishvili, and Paolo Favaro. Unsupervised generative 3d shape learning from natural images. *arXiv:1910.00287*, 2019. 2, 3, 6, 7

[27] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *CVPR*, 2020. 2, 7

[28] Yu Tian, Xi Peng, Long Zhao, Shaoting Zhang, and Dimitris N Metaxas. Cr-gan: learning complete representations for multi-view generation. In *IJCAI*, 2018. 1, 2

[29] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, 2017. 1, 2

[30] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *CVPR*, 2018. 2

[31] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *CVPR*, 2020. 2, 3, 4, 7, 8, 11

[32] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 2016. 5

[33] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. *arXiv:2010.09125*, 2020. 2, 3, 4, 5

[34] Jian Zhao, Lin Xiong, Panasonic Karlekar Jayashree, Jianshu Li, Fang Zhao, Zhecan Wang, Panasonic Sugiri Pranata, Panasonic Shengmei Shen, Shuicheng Yan, and Jiashi Feng. Dual-agent gans for photorealistic and identity preserving profile face synthesis. In *NeurIPS*, 2017. 1

[35] Jian Zhao, Lin Xiong, Jianshu Li, Junliang Xing, Shuicheng Yan, and Jiashi Feng. 3d-aided dual-agent gans for unconstrained face recognition. *IEEE Trans. on PAMI*, 2018. 1, 2

A. Joint Probability as Lower Bound

Here we show the relationship between $\log p(I'_w, w')$ and the likelihood $p(I'_w) = \int_w p_{G_{2D}}(x|w)p(w)dw$. In particular, by introducing an additional encoder $q(w)$, the variational lower bound [16] is given by:

$$\log p(I'_w) \geq \mathbb{E}_{w\sim q(w)}[\log \frac{p(I'_w, w)}{q(w)}]. \tag{10}$$

In the original Variational AutoEncoder [16], q should be a conditional distribution defined as an image encoder to approximate the posterior $p(w|I'_w)$. Here, we use the manipulation network to replace the image encoder. Formally, consider q to be a Gaussian distribution conditioned on the style code $\hat{w}$, perturbation parameters V' and L', i.e. $q(w|M(\hat{w}, V', L')) = q(w|w') = \mathcal{N}(w', \sigma^2_{w'}\mathbf{I})$, if we further consider q to be a deterministic approximation, i.e. $\sigma_{w'}$ is a fixed value and $\sigma_{w'} \to 0$, the lower bound in Equation 10 becomes $\log p(I'_w, w') + c$, where c is a constant and could be omitted.

B. Additional Implementation Details

We use an Adam optimizer [15] with a learning rate of 1e-4 to train the model. The 3D generator is trained for $30,000$ steps. For the perceptual loss, we average the ℓ_2 distance between the output of `relu2_2`, `relu3_3`, `relu4_3` to compute the loss. The feature maps are ℓ_2-normalized on each pixel. Further, the images are downsampled with a scale of $\times 1$, $\times 2$, $\times 4$ and the perceptual losses are averaged for different resolutions. For the 3D renderer, we assume a fov of 10. The output depth maps are normalized between $(0.9, 1.1)$. The output viewpoint rotation and translation are normalized between (-60°,60°) and (-0.1,0.1), respectively. The lighting coefficients are normalized to (0,1). For the perturbation, we empirically sample yaw angles from a uniform distribution over $[-45°,45°]$, and pitch angles from $[-10°,10°]$. The roll angle is fixed at 0. For the lighting, we found it a difficult task for StyleGAN2 to synthesize faces with manually chosen lighting parameters. Thus, we shuffle the lighting parameters across the batch. The identity regularization loss is only applied to faces that are perturbed with a yaw angle within $[-25°,25°]$.

B.1. Network Architectures

The viewpoint decoder D_V and light decoder D_L are both 4-layer MLPs with LeakyReLU [19] as activation function. The latent manipulation network is composed of two parts, the first part are composed of three 4-layer MLPs to encode latent code $\hat{w}$, viewpoint V_0 and light L_0, respectively, whose outputs are summed into a feature vector for the second part. The second part is another 4-layer MLP that outputs a new style code. The hidden size of all MLPs in our work is 512, same as the style code. For shape decoder

Shape Decoder	Output size
4-layer MLP	512
Deconv(512,512,4,1) + ReLU	512×4×4
Conv(512,512,3,1) + ReLU	512×4×4
Deconv(512,256,4,2) + GN(64) + ReLU	256×8×8
Conv(256,256,3,1) + GN(64) + ReLU	256×8×8
Deconv(256,128,4,2) + GN(32) + ReLU	128×16×16
Conv(128,128,3,1) + GN(32) + ReLU	128×16×16
Deconv(128,64,4,2) + GN(16) + ReLU	64×32×32
Conv(64,64,3,1) + GN(16) + ReLU	64×32×32
Deconv(64,32,4,2) + GN(8) + ReLU	32×64×64
Conv(32,32,3,1) + GN(8) + ReLU	32×64×64
Deconv(32,16,4,2) + GN(4) + ReLU	16×128×128
Conv(16,16,3,1) + GN(4) + ReLU	16×128×128
Upsample(2) + InjectConv(32,161,1)	16×128×128
Conv(16,16,3,1) + GN(4) + ReLU	16×256×256
Conv(16,16,5,1) + GN(4) + ReLU	16×256×256
Conv(16,1,5,1)	1×256×256

Table 3: The architecture of shape decoder.

Transformation Map Decoder	Output size
4-layer MLP	512
Deconv(512,512,4,1) + ReLU	512×4×4
Deconv(512,256,4,2) + GN(64) + ReLU	256×8×8
Deconv(256,128,4,2) + GN(32) + ReLU	128×16×16
Deconv(128,64,4,2) + GN(16) + ReLU	64×32×32
Deconv(64,32,4,2) + GN(8) + ReLU	32×64×64
Deconv(32,16,4,2) + GN(4) + ReLU	16×128×128
Upsample(2) + Conv(16,16,3,1) + GN(4) + ReLU	16×256×256
Conv(16,1,5,1)	1×256×256

Table 4: The architecture of transformation map decoder.

D_S and transformation decoder D_T, we mainly follow the decoder structure of Wu *et al*. [31], whose details are shown in Table 3 and Table 4, respectively. In detail, Conv(c_{in}, c_{out}, k, s) refers to a convolutional layer with c_{in} input channels, c_{out} output channels, a kernel size of k and a stride of s. Deconv is defined similarly. "GN" refers to the group normalization, where the argument is the number of groups. The "InjectConv" refers to a convolutional layer that takes the last feature map of 2D generator as input and injects its output to the decoder via summation. We found such a design helps to recover the facial details in the depth map.

B.2. Additional Results

In Figure 12, we show more results of our method and other baselines for rotating faces into different yaw angles. Figure 13 shows the results of rotating faces into different pitch angles. Note that for HoloGAN [20], we use the officially released code and train the model on our dataset. For CONFIGNet [17] and DiscoFaceGAN [2], since they involve additional synthesized data for training, we use their pre-trained models. Although most baselines are able to generate high-quality face images under near-frontal poses, clear content change could be observed in larger poses. In particular, HoloGAN and CONFIGNet generates blurred faces for larger poses, while the expression changes in the results of DiscoFaceGAN. In comparison, our method, though also suffer from quality degradation in larger poses, has a stricter control in terms of the content. We also provide the FID score of different methods in Table 5, where our method achieves second best image quality, even though our images are re-rendered from 2D generated images.

Method	FID↓
HoloGAN *et al*. [20]	100.28
DiscoFaceGAN *et al*. [2]	12.90
CONFIGNet *et al*. [17]	43.05
LiftedGAN (proposed)	29.81

Table 5: Quantitative Evaluation of the generated image quality between this work and baselines. The scores of "DiscoFaceGAN" and "CONFIGNet" are reported in their original papers. The scores of HoloGAN and our method are computed from sampled images of random poses.

In Figure 14 and Figure 15, we show additional examples of rotating generated faces with different yaw and pitch angles, respectively. In brief, we observe that the pre-trained StyleGAN2 (with our style manipulation network) is able to change poses to a certain degree, but fails to generate faces with larger pose angles that do not exist in the training data. Further, we see a similar trend of expression change when changing pitch angles as in DiscoFaceGAN, possibly due to the intrinsic bias in the FFHQ dataset. In contrast, the 3D generator distilled from StyleGAN2 extends this rotation capability to a larger degree with better content preservation.

Figure 12: Qualitative comparison with state-of-the-art methods on 3D-controllable GANs. Note that all these faces are generated by randomly sampling from latent space, therefore we cannot compare the manipulation over the same face. The example faces are supposed to have a yaw degree of -60,-45,-30,15,0,15,30,45,60.

Figure 13: Qualitative comparison with state-of-the-art methods on 3D-controllable GANs. Note that all these faces are generated by randomly sampling from latent space, therefore we cannot compare the manipulation over the same face. The example faces are supposed to have a pitch degree of -30,-20,-10,0,10,20,30.

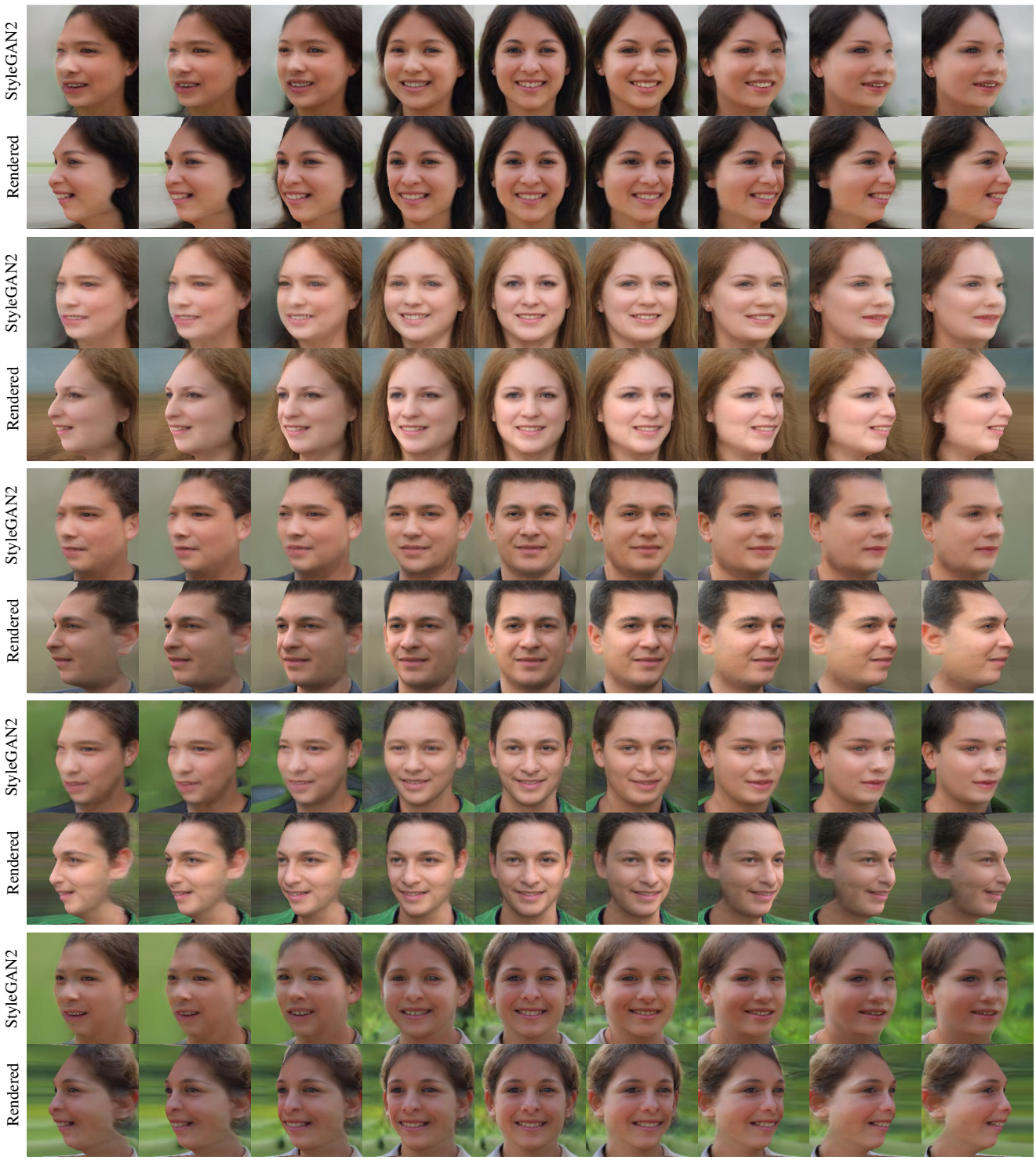


Figure 14: Results of rotating faces to yaw angles of -60,-45,-30,-15,0,15,30,45,60. For each two rows, the first row shows the results of generating by manipulating the StyleGAN2 latent style code, while the second row shows the results of 3D rendered faces.

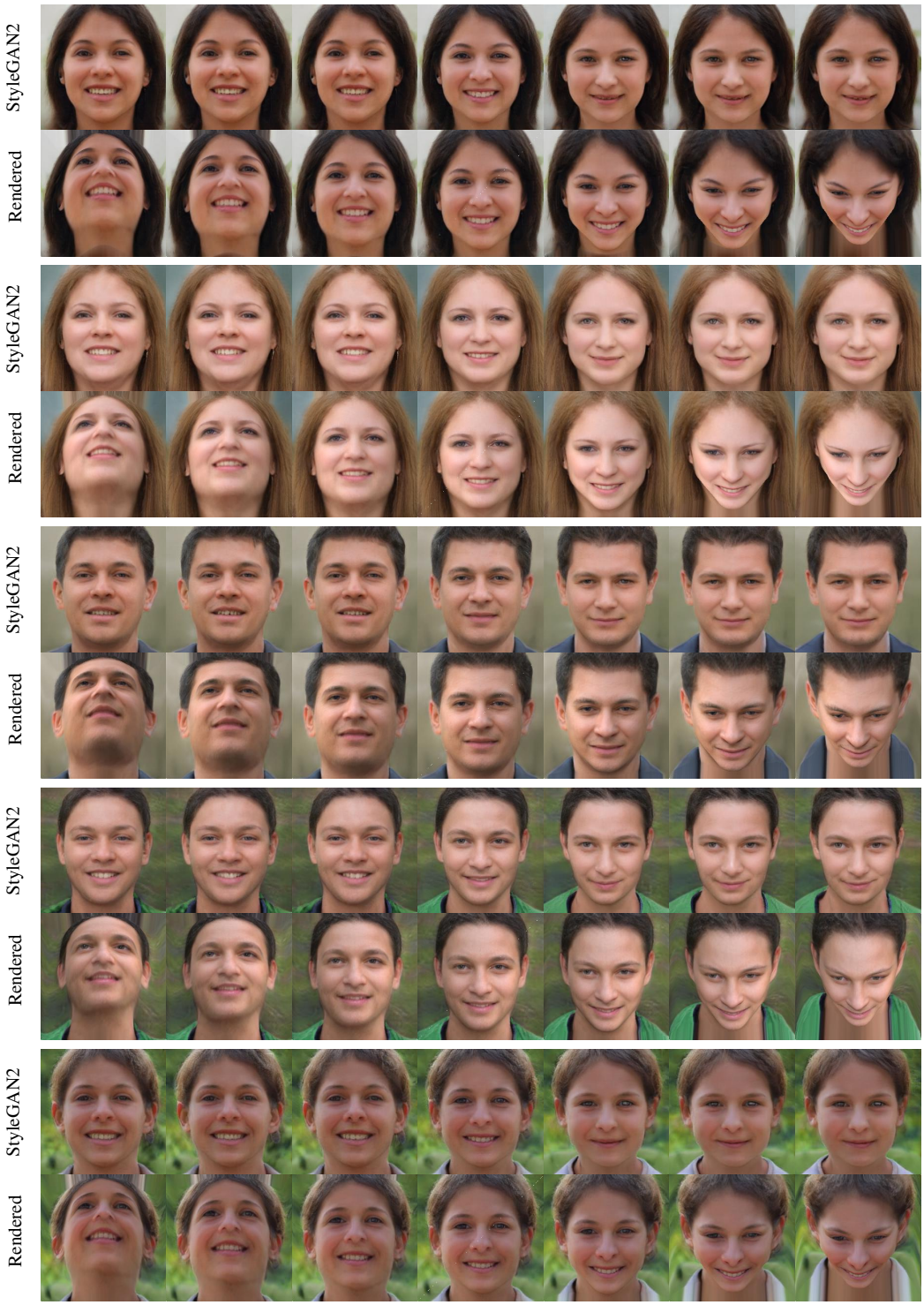


Figure 15: Results of rotating faces to pitch angles of -30,-20,-10,0,10,20,30. For each two rows, the first row shows the results of generating by manipulating the StyleGAN2 latent style code, while the second row shows the results of 3D rendered faces.