# Appendix

## A Derivation of the Volume Rendering Formula

This section provides the derivation of our 3D volume rendering formula Eq. 9 in the paper. Previous methods adopt a two-stage architecture to realize multi-view synthesis discretely, which is very time-consuming with redundant information. Hence we propose an orthogonal adaptive ray-sampling module to adaptively estimate the attributes required for image rendering of single-view face animation.

### A.1 Classical Discrete Volume Rendering Formula

Generally speaking, the volume rendering formula needs to be transformed from continuous integration to discrete summation, for which [14] designs a two-stage sampling method to realize the multi-view synthesis of a single object. Specifically, the classical volume rendering [11] calculates the pixel color along the camera ray as follows:

$$\begin{aligned} C &= \int_{t_s}^{t_e} \tau(t)\sigma(t)c(t)dt, \\ \tau(t) &= \exp(-\int_{t_s}^{t} \sigma(s)ds), \end{aligned} \tag{1}$$

where $\sigma$ represents the volume density, $c$ is the emitted color, $[t_s, t_e]$ is the integral interval, and $\tau$ is the transmittance. To make it implementable, the continuous integration is transformed to discrete summation with $D$ intervals. With some manipulations, [13] shows:

$$C = \sum_{i=1}^{D} c_i \tau_i (1 - \exp(-\sigma_i \delta_i)), \tag{2}$$

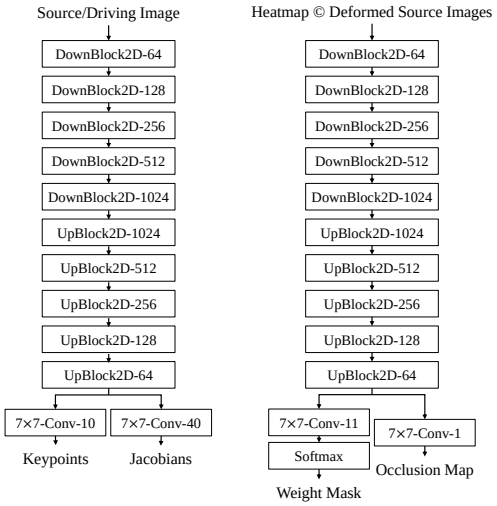
where $\delta_i = t_{i+1} - t_i$ is the $i$-th interval size, $\sigma_i$ and $c_i$ are constants to approximate the volume density and color values in $[t_i, t_{i+1}]$, respectively, and $\tau_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$.

### A.2 Volume Rendering of FNeVR

We introduce an orthogonal ray adaptive sampling module for single-view face animation. Following [14], we perform ray casting in the normalized device coordinate (NDC) space. For each image, we define $N = H \times W$ rays, where $W$ and $H$ are the width and height of the input image, respectively. Each ray is orthogonal to the near clipping plane, and the ray path from the pixel centre is calculated. According to Eq. 2 above, conventional neural rendering first uses a "coarse" network to roughly estimate the computational contribution of each uniform sampling location, and then further performs non-uniform discretization for a more informed sampling of points, which are inputted to another "fine" network for accurate estimation. Instead, our FVR only introduces one MLP network to directly estimate the voxel probability $p_\sigma$ of each voxel, which is the integral of the volume density $\sigma$ within a suitable interval size $\delta$. Therefore, for our model, the selection of the interval size $\delta$ is actually involved in the estimation of the voxel probability $p_\sigma$ and adaptively adjusted by the MLP. Finally, we rewrite Eq. 2 as the weighted sum of all sampled colors $p_{color}$ for each ray as:

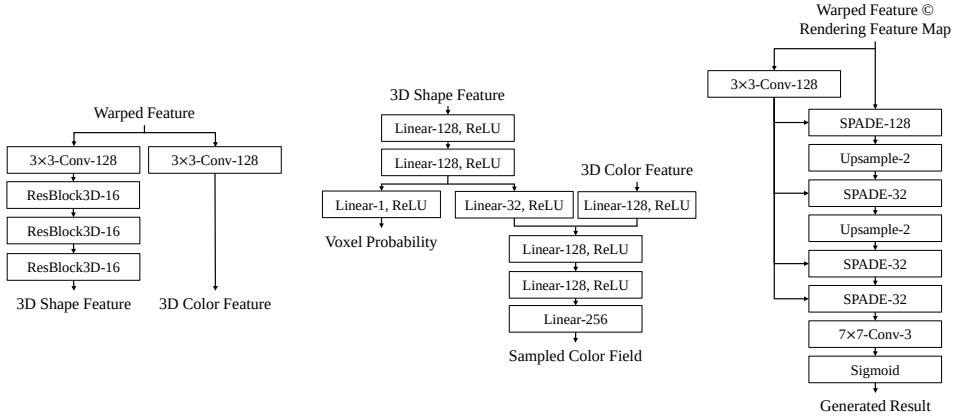
$$\begin{aligned} C &= \sum_{i=1}^{D} \tau_i (1 - \exp(-p_{\sigma,i})) p_{color,i} \\ \tau_i &= \exp(-\sum_{j=1}^{i-1} p_{\sigma,j}), \end{aligned} \tag{3}$$

which is Eq. 9 in the main paper. It is worth noting that our FVR can be applied to more generation tasks as long as appropriate 3D shape and color information is provided. For example, if we introduce reliable 3D body information to FVR, FVR can be applied to generate photo-realistic images of the body.




**(a) Keypoint Detector** **(b) Dense Motion Network**

Figure 1: Architecture of the motion estimation in our framework. ©: channel-wise concatenation. For the basic blocks UpBlock2D and DownBlock2D, please refer to Fig. 3(c) and Fig. 3(d), respectively.




**(a) 3D Feature Extractors** **(b) Orthogonal Adaptive Ray-Sampler** **(c) Rendering Decoder**

Figure 2: Architecture of the rendering component in our framework. ©: channel-wise concatenation. For the basic blocks ResBlock3D and SPADE, please refer to Fig. 3(e) and Fig. 3(b), respectively.

# B Implementation Details

## B.1 Architecture Details

**Keypoint Detector.** We employ the keypoint detector proposed by [17], which takes one image as input and adopts a UNet-based encoder-decoder architecture to extract the feature for keypoints prediction. Since we adopt a 2D warping strategy, only 2D CNN is required. Its detailed structure is shown in Fig. 1(a).

**Dense Motion Network.** We use the dense motion network proposed by [17] to predict an occlusion map indicating the regions to be inpainted and a group of weight masks related to the $K$ sparse motion fields, which are calculated by each pair of keypoints $\{p_{S,k}, p_{D,k}\}$ with their Jacobians. Then we can aggregate the sparse motions by a weighted sum to obtain the final dense motion field. The detailed structure of the dense motion network is shown in Fig. 1(b). In particular, the input to the network is the concatenation of $K$ heatmaps and $K+1$ deformed source images, where the heatmaps are calculated by the keypoints, and the deformed images are transformed by $K+1$ motion fields.

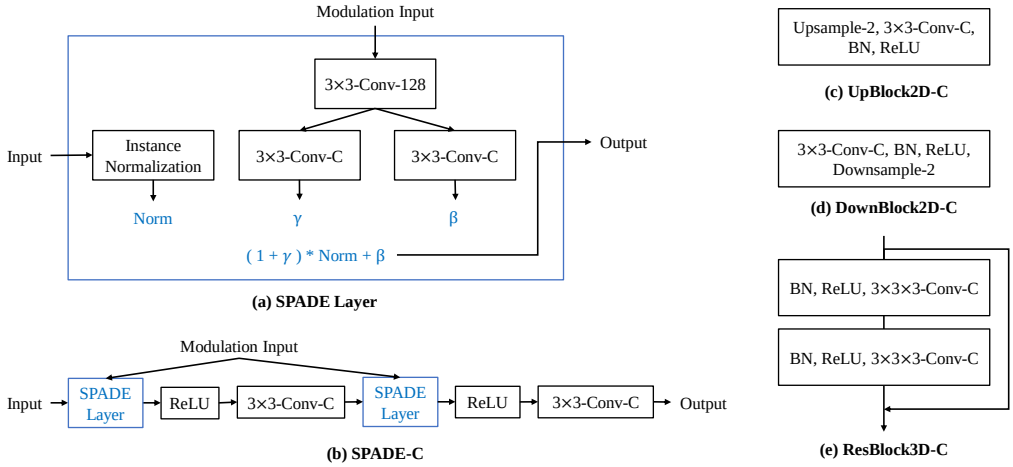



Figure 3: Architectures of the basic blocks.

**3D Feature Extractor.** Since we need to extract the 3D information of the warped feature for subsequent image rendering, we design a color extractor and a shape extractor, both of which first refine the 2D feature with the size of $256 \times 64 \times 64$ into 3D features with the size of $16 \times 8 \times 64 \times 64$. The detailed structure of the 3D feature extractors is shown in Fig. 2(a). $N_\sigma$ and $N_{color}$ are the final dimensions of the outputs of the color extractor and the shape extractor, respectively, and they are both set to 16.

**Orthogonal Adaptive Ray-Sampler.** Based on the extracted 3D features above, we can further calculate the sampled color field and the voxel probability, which are required for volume rendering. We adopt an MLP-based network to estimate the sampled color field with the size of $64 \times 64 \times 8 \times 256$ and the voxel probability with the size of $64 \times 64 \times 8 \times 1$. The detailed structure of the orthogonal adaptive ray-sampler is shown in Fig. 2(b).

**Rendering Decoder.** We design a rendering decoder based on the SPADE layer [16]. We adopt a series of the SPADE layers and upsampling layers to transform the warped feature and the volume rendering results into a photo-realistic image. The detailed structure of this decoder is shown in Fig. 2(c).

### B.2 Details of the Loss Functions

**Perceptual Loss.** During the training process, we use the VGG perceptual loss [10] to align the high-level semantic feature of the generated image with that of the ground truth. Specifically, we use the pre-trained VGG-19 [18] model $f_{VGG}$ to extract $L$ features of the generated image $x_{generated}$ and the driving frame $D_i$ and use the $L_1$ norm loss for similarity evaluation. Following [17], we first adopt the multi-scale pyramid by downsampling the generated image and ground truth to obtain four different resolution results, and then we aggregate the losses of the four resolutions to obtain the final result:

$$\mathcal{L}_P = \sum_{j=1}^{4} \sum_{l=1}^{L} L_1(f_{VGG,l}(D_{i,j}), f_{VGG,l}(x_{generated,j})). \tag{4}$$

**GAN Loss.** We use a discriminator $D_{GAN}$ with $N$ feature extractors to distinguish the generated results from the ground truth in $M$ scales. To be specific, we use the same generator-discriminator GAN loss as [17], including the least square loss [12] and the feature matching loss [20] to train an adversarial generative network:

$$\mathcal{L}_{G,generator,square} = \sum_{m=1}^{M} \mathbb{E}[(1 - D_{GAN,m}(x_{generated,m}))^2], \tag{5}$$

$$\mathcal{L}_{G,generator,matching} = \mathbb{E}\sum_{m=1}^{M}\sum_{n=1}^{N} L_1(D^{(n)}_{GAN,m}(D_{i,m}), D^{(n)}_{GAN,m}(x_{generated,m})), \tag{6}$$

$$\mathcal{L}_{G,discriminator} = \sum_{m=1}^{M} \mathbb{E}[(1 - D_{GAN,m}(D_{i,m}))^2] + \mathbb{E}[(D_{GAN,m}(x_{generated,m}))^2], \tag{7}$$

where we denote the $m$-th scale discriminator as $D_{GAN,m}$ and its $n$-th layer feature extractor as $D^{(n)}_{GAN,m}$. With the GAN loss, we further improve the authenticity of the generated images.

**Equivariance Loss.** To ensure the validity and consistency of the unsupervised keypoint detector, we use a known geometric transformation $\mathcal{T}$ to provide the equivariance constraints as stated in [17]. Concretely, for the keypoints $\{p_{\mathcal{T}(D),k}\}$, which are extracted from the transformed image $\mathcal{T}(D)$, and the keypoints $\{p_{D,k}\}$ of the original image $D$, we have:

$$\mathcal{L}_{E,keypoints} = \sum_{k=1}^{K} L_1(p_{D,k} - \mathcal{T}^{-1}(p_{\mathcal{T}(D),k})), \tag{8}$$

where $\mathcal{T}^{-1}$ is the inverse transformation of $\mathcal{T}$. Furthermore, we optimize the Jacobians $\{J_{D,k}\}$ by:

$$\mathcal{L}_{E,Jacobians} = \sum_{k=1}^{K} L_1(\mathbb{1} - J^{-1}_{D,k} \cdot J_{\mathcal{T}^{-1}} \cdot J_{\mathcal{T}(D),k}), \tag{9}$$

where $\mathbb{1}$ denotes the $2 \times 2$ identity matrix, $J_{\mathcal{T}^{-1}}$ is the Jacobian of the inverse geometric transformation $\mathcal{T}^{-1}$, and $\{J_{\mathcal{T}(D),k}\}$ are the Jacobians detected from the transformed image $\mathcal{T}(D)$.

# C Experiment Details

## C.1 Evaluation Metrics

We quantitatively compare our FNeRV with other state-of-the-art methods using the following metrics:

**$\mathcal{L}_1$ Distance.** We compute the averaged $\mathcal{L}_1$ distance between the generated and ground-truth images.

**Learned Perceptual Image Patch Similarity (LPIPS).** LPIPS [24] estimates the perceived distance of the generated image from the ground truth image by computing the cosine distances between the VGG network [18] features of the two images layer by layer and averaging them.

**Peak Signal-to-Noise Ratio (PSNR).** Using the ratio of the maximum possible power of the ground truth image to the mean square error between the reconstructed image and the ground truth to measure the image reconstruction quality.

**Structure Similarity Index Measure (SSIM).** From the perspective of image distortion modelling discussed in [22], the structural similarity of two images is measured by considering three different factors, brightness, contrast, and structure, where the mean is used as an estimate of brightness, the standard deviation as an estimate of contrast, and the covariance as a measure of structural similarity.

**Average Keypoint Distance (AKD).** We use a facial landmark detector [2] to detect facial landmarks from the reconstructed image and the ground-truth image and calculate the average distance between the two groups of landmarks. AKD can measure the accuracy of the facial posture of the generated frame and driving frame.

**Average Euclidean Distance (AED).** We use the face recognition network [1] to calculate the feature representations of the ground-truth image and the generated video frame and calculate their averaged Euclidean distance.

Source Sample 1 Sample 2 Sample 3 Sample 4 Sample 5

Figure 4: Illustration of same identity with different poses and expressions.

Table 1: Comparison of CSIM for the same identity with different poses and expressions.

| Sample | MoCo CSIM ↑ | Arc CSIM ↑ | Curricular CSIM ↑ |
|---|---|---|---|
| Sample 1 | 0.9853 | 0.8327 | 0.9131 |
| Sample 2 | 0.9962 | 0.9655 | 0.9663 |
| Sample 3 | 0.9203 | 0.7725 | 0.6524 |
| Sample 4 | 0.9034 | 0.6101 | 0.7626 |
| Sample 5 | 0.9327 | 0.7307 | 0.7074 |

**Frechet Inception Distance score (FID).** FID [7] is a metric to evaluate the image quality created by the generated model by imitating human perception of image similarity. It uses inception V3 [19] to compare the distribution of the generated image with the distribution of the real image used to train the generator.

**Cosine Similarity (CSIM).** The cosine similarity measures the difference between two individuals by using the cosine value of the angle between two vectors in a vector space. We use the pre-trained face recognition model [9] to generate embedded vectors to compute the cosine similarity to assess the quality of identity preservation.

In the same-identity case, we perform an experiment on all test videos from the Voxceleb [15] dataset. We treat the first frame as the source portrait and all the other frames as the driving video. Hence, we can get the synthesized frames with corresponding ground truth frames. We can directly calculate the evaluation metrics of $\mathcal{L}_1$, LPIPS, PSNR, SSIM, AKD, AED and FID between the real images and the synthetic images. For the cross-identity case, we randomly select 10 source images and 14 driving videos from each testing set of the Voxceleb and Voxceleb2 [4]. In this case, we do not have ground-truth images, so we can only calculate FID between the driving images and the synthetic images, and CSIM between the source images and the synthetic images.

### C.2 Reenactment Analysis

In the reenactment experiment, our FNeVR outperforms other SOTA methods on FID but has a certain gap with Face vid2vid [21] on CSIM. CSIM verifies the identity preservation according to the feature distance between the generated image and the source image, and is adopted in a number of studies [6; 21; 8]. According to our analysis, there are two reasons accounting for why CSIM of Face vid2vid is better than our FNeVR. On the one hand, 3D warping indeed provides a relatively stable face pose transformation with a high computational cost. On the other hand, we argue that it is not entirely reasonable to use CSIM to examine the preservation of face identity information, since 3D warping often leads to poor facial expression transformation, whereas it still produces good CSIM.

We think that expression and pose variation have a significant impact on CSIM, which can be proved by a simple experiment. Specially, we use three feature extraction networks (MoCo v2 [3], Arcface[5], and Curricularface [9]) to calculate the CSIM values between one source image and five sample images with the same person identity as the source image but different poses or expressions. As shown in Fig. 4, compared with the source image, Sample 1 and Sample 2 are slightly different in expressions, while Sample 3 has a great difference in pose. Meanwhile, Sample 4 and Sample 5 are disparate from the source image in both poses and expressions. The comparison of CSIM is shown in Table 1. Note that only Arc CSIM and Curricular CSIM focus on facial feature extraction. Obviously, despite the same identity, the variations in both poses and expressions have a great impact on CSIM, especially for detectors focusing on face features. Especially, there are pronounced decreases in

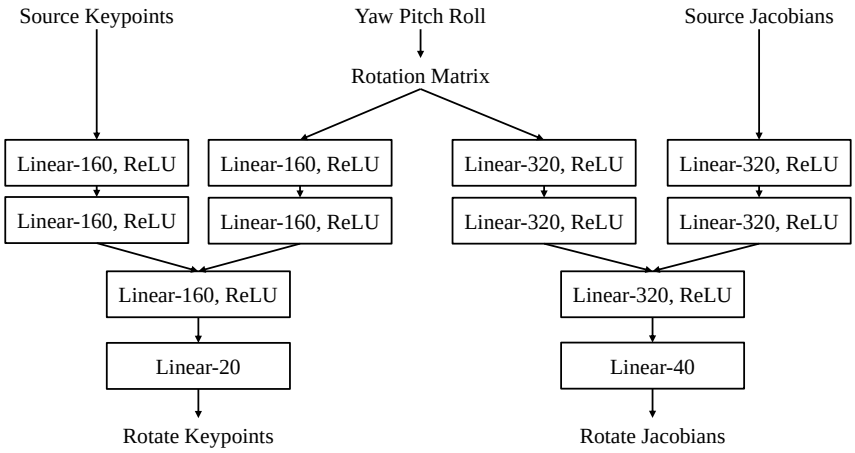



Figure 5: Architecture of the lightweight pose editor.


Figure 6: Additional experimental results of reconstruction.

CSIM for Sample 3, Sample 4 and Sample 5, indicating the considerable influence of facial poses. In other words, rich variations in facial poses and expressions make it difficult to maintain relatively high CSIM. Consequently, it is understandable that our model obtains lower CSIM than Face vid2vid. Finding more reasonable metrics to verify the preservation of identity information is part of our future research.

# D Additional Experimental Results

## D.1 Pose Editing

We design an MLP-based lightweight pose editor (LPE), and its detailed structure is shown in Fig. 5. In the training process, we input the deviations of the Euler angles $yaw$, $pitch$, and $roll$ between the source image and the driving frame provided by the 3D reconstruction network into LPE, together with the keypoint information of the source image. In the inference process, we can directly control the face pose with specific $yaw$, $pitch$, and $roll$ angles.

## D.2 Additional Comparisons with State-of-the-Art Methods

**Reconstruction.** In Fig. 6, we show three animation examples from the VoxCeleb [15] dataset, in which there is significant expression variation between each pair of two faces. From the generated results, we see that FOMM [17] can generate promising expression transformed images but with


Figure 7: Additional experimental results of reenactment without the relative motion transfer.

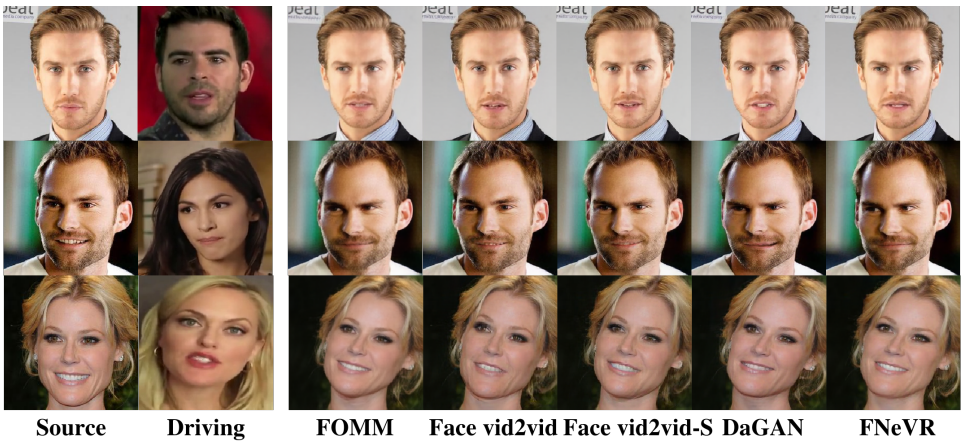



Figure 8: Additional experimental results of reenactment with the relative motion transfer.

blurred head profiles, while Face vid2vid [21] and DaGAN [8] generate the images with clear head profiles but poor expression transformation. For instance, Face vid2vid and DaGAN cannot transfer the visual directions or the corners of the mouth very well. Our FNeVR can combine their merits and generate promising expression transformed images with clear head profiles.

**Reenactment.** To enrich the reenactment experimental results shown in the main paper, we conduct additional experiments and show more samples in Fig. 7. Notably, in the cross-identity reenactment experiment of the main paper, all FOMM-based models [17; 21; 8] do not employ the relative motion transfer approach [17], in which the frame $D_{best}$ in the driving video whose pose is most similar to the source image $S$ is selected as the basis, and the motion fields of other driving images $D_i$ are calculated by the motion differences from $D_{best}$. Utilizing the relative motion transfer, we can maximize the preservation of identity information. Therefore, we conduct additional reenactment experiments based on this approach on VoxCeleb [15], and the results are shown in Fig. 8. It can be seen that Bilayer [23] has a greater advantage in maintaining identity information but cannot handle the image background. FOMM [17] has a desirable performance in expression transfer, but the generated results are distorted in some areas. Face vid2vid [21] works best on identity information due to the use of 3D representation. However, the facial details are not well captured, such as the inconsistency of the irises between the driving images and the source images. DaGAN [8] has a significant deficiency in maintaining identity information, and the contours of the faces in the generated images are completely deformed. Both Fig. 7 and Fig. 8 show that our FNeVR achieves the best results by maximizing the transfer of facial expressions without losing the facial features.

## E Social Impact

This work does not have a direct negative social impact. However, we should be aware of the power of realistic face animation and prevent it from being abused for malicious purposes.

## References


[1] Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. Openface: an open source facial behavior analysis toolkit. In *WACV*, 2016.

[2] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *ICCV*, 2017.

[3] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020.

[4] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *INTERSPEECH*, 2018.

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

[6] Michail Christos Doukas, Stefanos Zafeiriou, and Viktoriia Sharmanska. Headgan: One-shot neural head synthesis and editing. In *ICCV*, 2021.

[7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Asian Journal of Applied Science and Engineering*, 2017.

[8] Fa-Ting Hong, Longhao Zhang, Li Shen, and Dan Xu. Depth-aware generative adversarial network for talking head video generation. In *CVPR*, 2022.

[9] Yuge Huang, Yuhan Wang, Ying Tai, Xiaoming Liu, Pengcheng Shen, Shaoxin Li, Jilin Li, and Feiyue Huang. Curricularface: adaptive curriculum learning loss for deep face recognition. In *CVPR*, 2020.

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

[11] James T. Kajiya and Brian Von Herzen. Ray tracing volume densities. In *SIGGRAPH*, 1984.

[12] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017.

[13] Nelson Max. Optical models for direct volume rendering. *TVCG*, 1995.

[14] 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 *ECCV*, 2020.

[15] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: a large-scale speaker identification dataset. *INTERSPEECH*, 2017.

[16] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[17] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *NeurIPS*, 2019.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[19] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[20] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[21] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *CVPR*, 2021.

[22] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004.

[23] Egor Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and Victor Lempitsky. Fast bi-layer neural synthesis of one-shot realistic head avatars. In *ECCV*, 2020.

[24] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.