# Supplemental Material for NaviNeRF

## 1. Representing 3D Scenes with NeRF

As an implicit 3D reconstruction, Neural Radiance Fields (NeRF) maps scenes into a multi-layer perceptron (MLP) from limited views, which takes the position $\mathbf{x} \in \mathbb{R}$ and viewing direction $\mathbf{d} \in \mathbb{S}$ as input, for predicting emitted color $\mathbf{c}$ and volume density $\sigma$ of the targeted scene:

$$\mathbf{x} = (x, y, z),\ \mathbf{d} = (\theta, \phi),\ \mathbf{c} = (r, g, b)$$

where the $x, y, z$ represent the 3D location coordinate values and $\theta, \phi$ denote the 2D viewing direction, which ultimately implements a 5D vector-valued function. This continuous 5D scene representation is approximated by the MLP function $F_\Theta$.

$$F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$$

by optimizing the weights $\Theta$, the network map from 5D input to the corresponding volume density and directional emitted color. We also use the position encoding function in [4] to map each dimension of $\mathbf{x}$ with Fourier features:

$$\gamma(\mathbf{x}) = [sin(2^0\mathbf{x}), cos(2^0\mathbf{x}), ..., sin(2^{L-1}\mathbf{x}), cos(2^{L-1}\mathbf{x})]$$

Typically, $\gamma$ is a mapping from $\mathbb{R}$ into a higher dimensional space $\mathbb{R}^{2L}$. For the **inner refinement branch**, we employ the StyleNeRF baseline, to formalize NaviNeRF representations by conditioning NeRF with style vectors $w$ as follows:

$$\zeta_w^n(\mathbf{x}) = g_w^n \circ g_w^{n-1} \circ \ldots \circ g_w^1 \circ \gamma(\mathbf{x}), \text{ where } w = M(z), z \in \mathcal{Z}$$

where $M$ is an 8-layer mapping network that maps noise $z$ from the Gaussian space $\mathcal{Z}$ to the intermediate space $\mathcal{W}^+$. $g_w^i$ is the $i^{th}$ layer MLP whose weight matrix is modulated by the input style vector $w$. $\zeta_w^n$ is the $n^{th}$ layer feature of that point. Based on the conducted features, the model can then predict emitted color and volume density:

$$\mathbf{c}_w(\mathbf{x}, \mathbf{d}) = h_c \circ [\zeta_w^{n_c}(\mathbf{x}), \gamma(\mathbf{d})]$$
$$\sigma_w(\mathbf{x}) = h_\sigma \circ \zeta_w^{n_\sigma}(\mathbf{x})$$

where $h_c$ and $h_\sigma$ can be a linear projection or 2-layer MLPs. We take the default configuration of StyleNeRF that assume $n_c > n_\sigma$ as the visual appearance generally needs more capacity to model than the geometry. For the image $I \in \mathbb{R}$, the color of pixels is calculated by the volume rendering function which takes each camera ray $r$ for each pixel:

$$I_w(r) = \int_0^\infty p_w(t)\mathbf{c}_w(r(t), \mathbf{d})dt,$$

$$\text{where } p_w(t) = \exp\left(-\int_0^t \sigma_w(r(s))ds\right) \cdot \sigma_w(r(t))$$

where $p$ denotes a given camera pose and each ray $r(t) = o + t\mathbf{d}$ is calculated based on the original camera $o$. For the **outer navigation branch**, the image is generated in the same configuration but without the mapping network.

## 2. Datasets Details

**FFHQ** [22]: consists of 70k high-quality images, each with a resolution of 1024×1024. The dataset covers a wide range of global variation in terms of age, gender, and ethnicity, and also includes fine-grained representations such as accessories like eyeglasses, sunglasses, hats, and so on. To simplify the camera setup, we assume that the human face is captured at the origin, and the camera is located on the unit sphere, pointed towards the origin with a fixed field of view. The pitch and yaw of the camera are sampled from either a uniform or Gaussian distribution.

**AFHQ** [23]: comprise 15k high-quality images at a resolution of 512×512. The dataset includes three categories of cat, dog, and wildlife. The training images are merged directly, without the utilization of label information. Similarly to the FFHQ dataset, the pitch and yaw are sampled from a Gaussian distribution. Due to the inclusion of multiple domains and diverse images of various breeds (at least eight per domain), AFHQ presents a more challenging image-to-image translation problem. All images are aligned vertically and horizontally to center the eyes.

### 2.1. Training Protocol

The sampled latent code $z$ is from a Gaussian distribution of dimension 512. We append the shifts to $z$ through a learnable orthonormal matrix $S$ and an 8-layer fully-connected network $M$. The output matrix from $M$ is of size 18×512, where the shifted ($9^{\text{th}}$-$18^{\text{th}}$) style vectors are fed two-to-one into ($5^{\text{th}}$ - $9^{\text{th}}$) NeRF MLP layers. The decoder

is designed as a 4-layer MLP with Adam activation. We use batch sizes of 64 and 32 for the FFHQ and AFHQ datasets, respectively, taking into consideration their different scales. The default learning rate is 0.0005. The sampling dimension of $z$ is set to 512, where the $w$ vector is formed as 18×512. For pre-training, we follow the instructions outlined in StyleNeRF but resize all input data to 256×256. By default, we set batch size to 64 for every dataset, and the learning rate of generator and discriminator are set to 0.0025.

## 3. Competitive Models

We conduct the qualitative and quantitative comparison on NaviNeRF with three typical 3D-aware GANs (pi-GAN, GIRAFFE, StyleNeRF) and two editing-oriented models (FENeRF, CGOF++).

**pi-GAN** [24]: Since pi-GAN does not provide a pre-trained model for the FFHQ dataset, we train pi-GAN using the same configuration as the other 3D-aware GANs for comparison purposes.

**GIRAFFE** [25]: As GIRAFFE can be considered an improvement of GRAF, we only include GIRAFFE in our competitive experiments. We load the pre-trained checkpoints of GIRAFFE on the FFHQ dataset according to the official implementation.

**StyleNeRF** [26]: As our baseline model in the current stage, we pre-train a StyleNeRF generator on both the FFHQ and AFHQ datasets. In the competitive experiments, we load its pre-trained checkpoints on the FFHQ.

**FENeRF** [45]: We adopt the default implementation of FENeRF on the FFHQ to demonstrate the partial control capacity.

**CGOF++** [13]: We utilize the author's implementation of CGOF++ on the FFHQ to demonstrate the partial control capacity.

## 4. Additional Results

In Figure 10, we present additional results that demonstrate the capacity of NaviNeRF to identify and manipulate fine-grained attributes. To verify the ability of the disentanglement modules in identifying these attributes, we further removed the outer navigation branch and inner refinement procedure (that means we append inner shifts on every 18 dimension of $w$). As illustrated in Figure 9, without the disentanglement modules, the model can only manipulate global style and fails to discover underlying semantic directions. Moreover, in Figure 11 and 12, we showcase the pre-training results for the FFHQ and AFHQ datasets. However, even when the model is fully configured as StyleNeRF during pre-training, artifacts still appear in the generated images for the AFHQ dataset, as demonstrated in Figure 12. Despite the ambiguous and challenging data, the model still achieves fine-grained awareness, which validates the robustness of NaviNeRF. In future work, we plan to employ more 3D-aware models to improve the quality of generated scenes. Additionally, we will consider using GAN inversion technology to achieve real-time attributes manipulation by interactive controls from the user.

Figure 9. The generation results without outer navigation branch and inner refinement procedure. The model can only manipulate on global style but fails to discover underlying semantic directions.

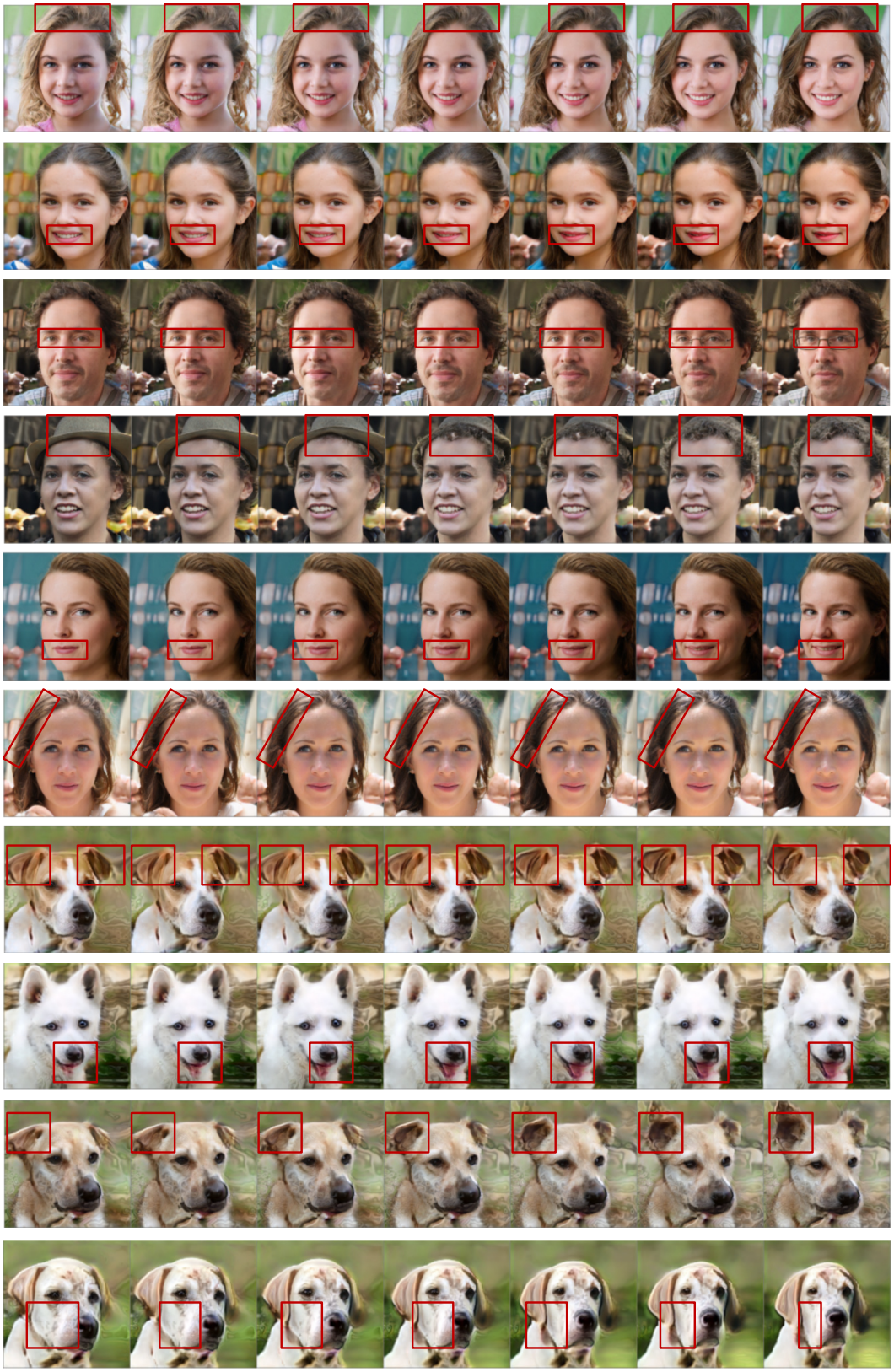

Figure 10. Additional results of fine-grained 3D disentanglement of NaviNeRF.

Figure 11. Pre-training results on the FFHQ $256^2$.

Figure 12. Pre-training results on the AFHQ $256^2$.