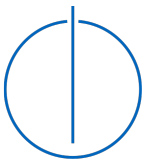
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Adversarial 3D Shape Reconstruction using Neural Fields

Zhuolun Zhou



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

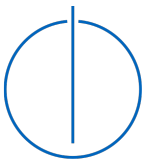
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Adversarial 3D Shape Reconstruction using Neural Fields

Rekonstruktion von 3D Formen mit Neuronalen Feldern und GANs

Author:	Zhuolun Zhou
Supervisor:	Daniel Cremers
Advisor:	Lukas Koestler, Tarun Yenamandra
Submission Date:	15.01.2023



I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.01.2023 Zhuolun Zhou

Abstract

3D reconstruction is a central topic in computer vision. Existing methods mostly focus on reconstruction accuracy and speed. However, many reconstruction results, though accurate, do not appear of high visual fidelity and realistic to human perception. This is because humans are not sensitive to geometric accuracy but can easily distinguish between a real scene and a reconstructed 3D scene. Meanwhile, most 3D generation methods can generate photo-realistic images but are not conditioned on existing objects. We aim to leverage 3D generation to make 3D reconstruction results appear more photo-realistic to human perception. Toward this goal, we propose a conditional neural radiance field with a 3D U-Net encoder and a progressive discriminator supervised by images with known poses. We condition the neural radiance field with a global feature vector and a feature volume that encodes local information. The neural field's input points are interpolated within the feature volume and are further transformed to color and density for volume rendering, producing images to be discriminated against real images. We evaluate the proposed method with perceptual similarity metrics such as FID, LPIPS, etc. Our method can refine the input geometry of unseen objects of a known category, thus showing its generalization capacity. The results validate our approach and show that our method can render images of higher visual fidelity than those directly rendered from the input geometry. Since we adopt a GAN framework, our method can also interpolate in the latent space.

Contents

1 Introduction

3D reconstruction[27, 18, 49, 53] and 3D generation[1, 2, 41, 32, 31, 47, 15, 43] have recently made tremendous progress. 3D reconstruction focuses on accurately recovering the geometry and/or appearance of real-world scenes, and 3D generation aims to synthesize high-fidelity images with control over underlying scene properties such as camera poses, geometry, appearance, etc.

However, for content creation, both 3D reconstruction and 3D generation methods have shortcomings. On the one hand, traditional 3D reconstruction methods emphasize reconstructed accuracy and can produce faithful scans[18], but they are often not photo-realistic. This is especially true if incomplete or noisy input data is given, which is often the case if an untrained user does the capture without professional equipment. On the other hand, current 3D generation methods [1, 2, 41, 32, 31] are mostly unconditional. They can synthesize scenes or objects with high visual fidelity, but these have no relation to the existing scenes or objects. [51, 20] represent 3D objects from a single or few images, but they do not provide geometry output. We consider the scenario where a user scans a scene or object for entertainment, e.g., to have a virtual 3D meeting within the scene or to place the object in a virtual reality game. The resulting 3D scan should faithfully represent the real scene. For example, if the room has a desk, a shelf, and a closet, the same should hold for the reconstructed scene. But the reconstructed geometry isn't required to match the real-world geometry exactly. Because humans are insensitive to geometric accuracy and would have no objections to imperceivable deviations such as changing the exact locations of the objects or minor changes in the object geometry.

To address this problem, we propose to study *Adversarial 3D Reconstruction* and leverage 3D generation to improve the visual fidelity of 3D reconstruction results. The main difference between classic 3D reconstruction and adversarial 3D reconstruction is the evaluation protocol. 3D reconstruction is always evaluated using objective, non-human metrics like the Chamfer distance, F-score, or the PSNR of rendered images. Indeed, changing from rigid geometry-based metrics to image-based novel-view metrics is one of the reasons why NeRF [27] achieved impressive results. If metrics are overly strict and measure quantities not of interest in the current setting, they greatly inhibit optimizing the actual quantity of interest. Therefore, the ultimate metric for user-centered reconstruction is the user's satisfaction. This implies that the gold standard for evaluation must be user studies. However, user studies are expensive and complicated to conduct, so

they are replaced in this work by qualitative results and perceptual metrics that correlate well with human perception.

We confine ourselves to the problem of reconstructing an object from images and imperfect geometry. The reconstruction is focused on visual fidelity rather than geometry accuracy. This means that the reconstructed geometry does not have to be perfectly accurate and could have slight deviations from the ground truth. But the rendered images or video from the object should be highly faithful to human perception. We use a 3D U-Net to encode the coarse input geometry. The output encodes global information by a latent feature vector and local information by a feature volume grid. The global feature vector FiLM [35] conditions the neural radiance field, while the feature volume conditions the neural field by allowing input 3D points to interpolate within the feature volume and aggregate features. The rendered images are further fed into a discriminator to encourage the photorealism of the generator.

In summary, the contributions of this work are:

- We propose a feature volume for local information encoding and a feature vector for global information of 3D objects to condition the neural radiance field.
- We introduce the adversarial loss in a GAN framework into 3D reconstruction.
- We implemented a conditioned neural radiance field for geometry refinement.

Background knowledge needed for understanding the following chapters is summarized in chapter 2. We give an overview of related work in chapter 3. Afterward, in chapter 4, we first give a brief overview and then in-detail description of our method. Experiments validating the proposed method are given in chapter 5. We summarize this work and discuss the potential future work in chapter 6.

2 Background

In this chapter, we recapitulate critical concepts used throughout the thesis. The original works are cited and should be considered for in-detail information.

2.1 Neural radiance field and volume rendering

Neural radiance field (NeRF) [27] belongs to the family of implicit models for representing 3D scenes or objects. Unlike explicit methods where geometry and appearance are stored directly, such as point cloud, voxel, mesh, etc., implicit models aim to represent the 3D world by an implicit function, e.g., a neural network. One example of implicit methods is Occupancy Networks[25], in which the network takes in 3D coordinates of a point and an observation and outputs the point's probability of occupancy. Similarly, NeRF learns a continuous, volumetric function that maps a 3D position to its color and density, which can subsequently be used to render an image from a specific view by volume rendering. Implicit methods allow free-view rendering and have expressive representation power and a low memory footprint.

NeRF is parameterized as a multi-layer perceptron (MLP) that takes as input a 3D coordinate $\boldsymbol{x} = (x, y, z)$ and the camera ray's viewing direction $\boldsymbol{d}$. The neural radiance field outputs both the spatially varying density $\sigma = \sigma(\boldsymbol{x}) : \mathbb{R}^3 \to \mathbb{R}$ and the view-dependent color $(r, g, b) = \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{d}) : \mathbb{R}^5 \to \mathbb{R}^3$. The MLP network $F_\Theta : (\boldsymbol{x}, \boldsymbol{d}) \to (\boldsymbol{c}, \sigma)$ approximates a continuous 5D scene representation. Its weight Θ is optimized by gradient descent to map from each 5D coordinate to its corresponding volume density and directional emitted color.

The view-dependency of RGB color $\boldsymbol{c}$ is used to enforce multiview consistency for non-Lambertian surfaces. In practice, the MLP F_Θ first processes the input 3D coordinate $\boldsymbol{x}$ with a few fully-connected layers and output σ and a feature vector. This feature vector is then combined with the viewing direction $\boldsymbol{d}$ and passed to some additional fully-connected layers that output the view-dependent RGB color $\boldsymbol{c}$.

Volume rendering is applied to the neural radiance field to render an RGB image from a specific camera pose. For each pixel on the RGB image, a ray is cast from the camera origin $\boldsymbol{o}$, and a number of points along this ray are sampled. NeRF then predicts the density σ and color $\boldsymbol{c}$ of these points based on the 3D position of the points and the ray direction $\boldsymbol{d}$. Volume rendering subsequently integrates the densities and colors of these

points along the ray and outputs the color for this pixel. The pixel color $\boldsymbol{C}$ for a camera ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$ with near and far bounds t_n and t_f is calculated using the volume rendering equation [23]:

$$\boldsymbol{C}(\boldsymbol{r}) = \int_{t_n}^{t_f} T(t)\sigma(\boldsymbol{r}(t))\boldsymbol{c}(\boldsymbol{r}(t), \boldsymbol{d})\, dt \tag{2.1}$$

2.2 Generative Adversarial Nets

Generative Adversarial Nets (GAN) [11] aims to solve the generative modeling problem: to learn the probability distribution of data and then generate new data samples from the same distribution. GAN addresses this problem by having two competing neural network modules, a generator G and a discriminator D, playing a minimax two-player game against each other. The generator aims to map a noise sample to a synthetic data sample as similar as possible to the real data to "fool" the discriminator, while the discriminator tries to distinguish the real data samples from synthesized samples. Below we give a formal description of the vanilla GAN.

The goal is to learn the data distribution $\boldsymbol{x} \sim p_{data}(\boldsymbol{x})$ and let the generator G generate samples as close to $p(\boldsymbol{x})$ as possible from input noise $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$. The discriminator $D(\boldsymbol{x})$ maps $\boldsymbol{x}$ to the probability that $\boldsymbol{x}$ comes from the $p_{data}(\boldsymbol{x})$ rather than from the generator. The generator $G(\boldsymbol{z}; \theta_g)$ and discriminator $D(\boldsymbol{x}; \theta_d)$ are parameterized as multi-payer perceptrons, where θ_g and θ_d are the respective network parameters. D is trained to maximize the probability of assigning 1 to real samples and 0 to samples generated from G. And D is trained to maximize $\log(D(G(\boldsymbol{z}))$, or minimize $\log(1 - D(G(\boldsymbol{z}))$. D and G play a two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.2}$$

3 Related work

This work is related to multiple fields. In this chapter, we group relevant papers into two domains: implicit scene representation and conditional NeRFs. Many works will not perfectly fit into our categorization or might fit into multiple domains, and we ask the reader to bear with us. Due to the vast amount of literature in this field, we cannot cover all aspects of these domains. We only list the most relevant works and works that have inspired us.

3.1 Implicit neural representation

Implicit neural representations (in some literature also referred to as coordinate-based representation or neural fields) are a novel way to parameterize fields by neural networks. By representing a 3D scene with a neural network parameterized continuous function such as a signed distance function or a radiance field, implicit neural representation has recently demonstrated its potential in 3D scene or shape representation. Many implicit methods outperform point-, voxel- and mesh-based traditional 3D representations in parameterizing geometry, and, in the meantime, they allow for learning priors over shapes.

3.1.1 Implicit neural representation of geometry

This section focuses on the implicit representation of 3D geometry. Below we list some milestones in this field.

DeepSDF[33] learns a continuous signed distance function (SDF) with an auto-decoder architecture conditioned on an optimized latent code to represent the surface of an entire class of 3D shapes. This representation implicitly encodes a shape's boundary as the zero-level set of the learned function. DeepSDF also proposed a novel encoder-less auto-decoder paradigm to condition the DeepSDF network on a latent vector. The latent vector is obtained by optimizing the loss of the network rather than as the output of an encoder network. Conditioning the scene representation network with a latent vector enables DeepSDF to represent multiple shapes and also interpolate between two shapes in the learned shape latent space.

OccupancyNet [25] and IM-Net [4] represent 3D geometry via an occupancy function. The object surface is defined by the decision boundaries of the neural network classifier,

which predicts the occupancy probability of a 3D coordinate in space. Both can generalize across ShapeNet by conditioning via concatenation. Depending on the application, the latent code is encoded by a CNN for an image, a PointNet [37] encoder for a point cloud, or a 3D convolutional neural network for voxel grids. ConvOccupancyNet [34] and IF-NET [5] further extended OccupancyNet [25] by using 3D CNNs or PointNet to process voxelized point clouds into embedding grids to locally parameterize an occupancy network, enabling detailed representation of more complicated scenes.

3.1.2 Neural volume rendering

In contrast to works in subsection 3.1.1, where geometry representation requires 3D supervision, neural volume rendering methods mostly require 2D appearance supervision.

Neural volume rendering (also known as differential rendering) refers to methods that generate images by casting a ray from the camera to the scene and taking an integral of quantities such as density and color over points sampled along the ray. Typically a neural network parameterizes the function which maps the 3D coordinates to density and color. Hence it also belongs to the coordinate-based representation and implicit representation methods. Neural volume rendering for novel synthesis is first introduced to the field by NeuralVolumes [22] and gained popularity and influence by NeRF [27] (see section 2.1). NeuralVolumes [22] leverages interpolated discrete voxel grids of color and density to represent object appearances. In contrast, NeRF learns a continuous neural field parameterized by a single ReLU MLP with positional encoding to achieve photo-realistic novel view synthesis.

The vanilla NeRF yields novel views of high visual quality but has many limitations. It only represents a single object and does not generalize to other objects or scenes. Also, vanilla NeRF is slow for training and rendering. It does not include geometry information and "bakes in" lighting. Many follow-up works tried to fill in these gaps. InstantNGP [30] significantly reduces the computation cost of NeRF by a multi-resolution hash encoding which allows the use of small neural networks without sacrificing quality. DS-NeRF [7] supervised NeRF with additional depth loss by a sparse set of points from Structure from Motion (SfM). [40] went one step forward from DS-NeRF by predicting and supervising dense depth maps and uncertainty based on the sparse points of SfM outputs.

3.2 Conditional NeRFs

The vanilla NeRF does not generalize to other objects, and conditional NeRFs address this issue by conditioning NeRF on an object-specific latent code. Object appearance and shape priors are incorporated by sharing MLP weights across object instances. The

conditional signal opens possibilities of 3D aware generative networks and conditional reconstruction from partial observations.

3.2.1 Generative NeRFs

2D image-based generative adversarial networks (GANs, see section 2.2) have successfully synthesized high-fidelity images. Still, they do not infer direct control over the underlying 3D scene parameters, such as camera and geometry. HoloGAN [31] attempts to model the 3D structure of the object in the synthesized image by incorporating a volumetric 3D representation. However, the use of explicit volume representation in HoloGAN [31] lacks the expressiveness for synthesizing high-fidelity images.

Generative NeRFs are 3D-aware GANs based on neural volume rendering that can generate 2D images with direct control over scene properties such as camera, appearance, and geometry. Below we list several relevant and important works in this field.

GRAF [41] conditions NeRF with separate shape and appearance latent codes and uses a patch-based discriminator that samples images at multiple scales to learn a generative model for high-resolution 3D-aware image synthesis. The shape codes and appearance codes are globally conditioned via concatenation. The generator predicts only image patches instead of an entire image due to the NeRF's high computation cost.

GIRAFFE[32] extended GRAF to generate scenes of multiple objects with a compositional NeRF for each object. The NeRF volumes of each object are translated, rotated, and scaled to combine into a single scene of multiple objects. Unlike NeRF and GRAF, which integrate over color and density output of MLP for volume rendering, GIRAFFE uses feature and density to render a low-resolution feature map to avoid the cost of generating high-resolution images. This feature map is upsampled and mapped to the final synthesized image by 2D CNNs with skip connections.

π-GAN conditions an implicit radiance field represented by a SIREN network [44] with an input noise vector. GRAF and GIRAFFE condition the NeRF by concatenating positional encoding with shape and appearance codes, while π-GAN conditions the SIREN by FiLM conditioning [35]. The sinusoidal SIREN activation also enables π-GAN to remove the positional encoding of NeRF, which is adopted by our method.

EG3D [2] leverages an expressive explicit-implicit network conditioned on a noise vector to synthesize high-resolution multi-view-consistent images and produce reliable 3D geometry. It proposed a tri-plane hybrid 3D representation in which a 3D point position is projected onto three feature planes. Point features are aggregated by summing the bilinearly interpolated feature vectors on the three feature planes. Compared with NeRF's implicit forward mapping of point coordinate to features via MLPs, and interpolating point coordinate inside a feature volume to get its feature vector, this tri-plane representation offers significant speed and memory benefits.

D3D [47] designs a 3D generative neural field aiming at disentangling geometry and appearance. π-GAN [1] and EG3D [2] have control over the camera parameters in the image synthesis process, but other scene properties, such as geometry and appearance, remain entangled and cannot be controlled independently. GRAF [41] and GIRAFEE [32] attempt to disentangle geometry from appearance, but their appearance information can leak through the geometry component. D3D uses a separate network for NeRF's color and density to achieve disentangled shape and appearance.

The methods above do not output textured meshes, and extracting textured surfaces from these methods is non-trivial; for example, the threshold of points' density to be determined as occupied has to be set manually and could be different for each object. GET3D [10] aims to mitigate this issue and synthesizes textured meshes by differentiable surface rendering rather than volume rendering. GET3D generates a 3D SDF defined on a deformable tetrahedral grid [9] and a texture field via two latent codes. The surface mesh is extracted from the 3D SDF by a differential marching tetrahedral algorithm. However, GET3D still relies on labeled 2D silhouettes as the discriminator input.

Texturify [43] aims to generate textures for untextured 3D shapes without shape-image correspondence or explicit 3D color supervision. The 3D GAN is conditioned on an input noise code and 3D shape. They encourage texture consistency across multiple views by a patch-consistency discriminator, which takes as input patches from multiple rendered views and real patches from a single image.

DreamFields [15] is a text-guided 3D GAN that can generate NeRFs corresponding to text prompts. NeRF is optimized by rendering multiple views so that they match a target caption according to the pre-trained CLIP [38] model.

3.2.2 NeRF-based 3D reconstruction

Methods discussed in subsection 3.2.1 are generative models and do not reconstruct the 3D representation from an existing object. This subsection discusses related works that condition NeRF on an existing object, such as an image or point cloud, to reconstruct the 3D representation of the object. This task is directly related to our work.

PlatonicGAN [13] can reconstruct a 3D volume from a single 2D input image. The 2D input image is encoded into a latent code which is fed to a generator to produce a 3D volume. The 3D volume is rendered into 2D images presented to the discriminator. It is worth noting that PlatonicGAN is not a NeRF-based method since its rendering is not neural volume rendering. A key idea of PlatonicGAN is to use GAN's adversarial supervision to improve 3D shapes. Adversarial supervision encourages rendered images from the 3D structure to be realistic. It allows deriving supervisory signals not just from renderings of predictions in the input view but also from novel views. This idea is also explored in HoloGAN [31], im2nerf [26] and ShSMesh [50]. Im2nerf aims to learn a neural

radiance field from a single input image in the wild. It leverages the adversarial loss on rendered images from novel to views to improve the multi-view consistency.

PixelNeRF[51] learns a continuous neural scene representation conditioned on one or few input images. A CNN processes images to extract features, and when querying a 3D point, it is projected on the image plane to aggregate the local point features. VisionNeRF [20] identifies PixelNeRF's shortcoming of rendering blurry predictions at viewpoints far from the source view and proposes an additional vision transformer [8] based global feature to learn long-range dependencies.

NICE-SLAM [53] learns a neural scene representation from an RGB-D image stream by tri-linearly interpolating points in a hierarchical feature grid. Our work also uses a similar single-level feature grid for point feature interpolation.

Points2nerf [54] generate NeRFs from 3D point clouds by a meta-learning hyper-network paradigm. The 3D point clouds are fed into an auto-encoder which generates weights of the NeRF network. This meta-learning paradigm is explored in detail in [46].

Point-NeRF [49] uses 3D neural points to model a neural radiance field. It first predicts depth for each image with a cost-volume-based 3D CNN from multi-view images and extracts 2D feature images from input images by a 2D CNN. The depth maps are unprojected to produce a point cloud, with each point having projected image features. Differentiable ray-marching then is only conducted nearby the neural point cloud. Each sampled shading point's feature is aggregated from its neural point neighbors and used for density and color computation. Compared with NICE-SLAM [53]'s feature grid encoding method, neural point cloud encodes the space more precisely, and only space around the underlying object is encoded. However, it depends on the multi-view stereo (MVS) algorithm, and its performance relies on the point cloud quality from MVS.

4 Methods

We introduce our model in three parts: the encoder, the decoder, and the discriminator. We start in section 4.1 with an overview of the whole model. Then, in section 4.2, we introduce our 3D U-Net [6] based volume encoder. Next, in section 4.3, we introduce our decoder network and neural renderer. Then, we describe the network architecture of the discriminator in section 4.4. Losses and evaluation metrics are discussed in section 4.5 and section 4.6. An alternative point cloud-based encoding approach is discussed in section 4.7.

4.1 System overview

Our ultimate goal is to improve and refine the input geometry, usually in a point cloud format, and render realistic images from it. We consider our input geometry as colored point clouds either from existing reconstruction methods or datasets with 3D objects such as ShapeNet and CO3D [39]. We also assume the RGB images and their corresponding camera poses are known, which are leveraged to supervise the neural rendering.

In order to condition NeRF on the colored point clouds, the natural idea is to encode the point clouds with points-based networks such as PointNet [37]. We discuss the PointNet encoder approach in section 4.6. However, this approach is proved in our experiments (see section 5.2) to be less effective in representing the 3D objects and rendering realistic images. We hypothesize that this is due to the enormous loss of geometry and appearance information when mapping the point clouds to a single latent vector.

Inspired by [34], we proposed a geometry encoding method based on 3D U-Net, which directly takes in colored voxel grids and outputs a volume of latent features for each voxel. We dubbed this volume of latent features "Feature volume". Point clouds are voxelized to create the colored voxel grids input for the 3D U-Net (see subsection 5.1.1 for details). Our intuition comes from NeRF's positional encoding and NeRF as a volume rendering method. Each sampled 3D point along the camera rays can be interpolated in the feature volume, and the features thus contain strong positional information. The decoder further maps the point's features to the four-dimensional density σ and RGB color. Similar to π-GAN [1], our decoder consists of a few FiLM-ed SIREN [35, 44] layers. Volume rendering takes the density and color of points along camera rays and

output a rendered image, which is further discriminated against real images to improve visual fidelity. An overview of our framework is shown in Figure 4.1.

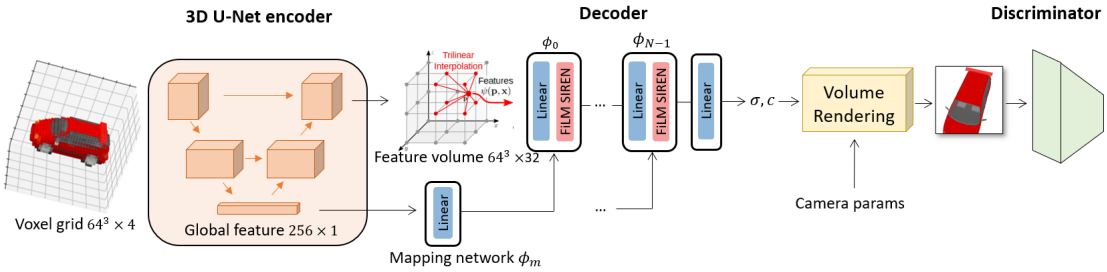


Figure 4.1: Our framework comprises several parts: a 3D U-Net encoder mapping the input voxel grid into the feature volume and a global feature, a decoder module of a few FiLM-ed SIREN layers [35, 44] which transform the interpolated feature volume and the global features to color and density, a volume rendering module which renders images given color and density field, and a progressive discriminator.

4.2 3D U-Net encoder

The encoder aims to embed both the global and local information of the 3D geometry input into latent space. 3D U-Net is an effective network to extract global and local information from volumetric data due to its multilevel contracting encoder part to analyze the whole volume and a successive expanding decoder part to combine fine-grained and coarse-grained features.

The original 3D U-Net [6] takes in a 3D volume and outputs a feature volume. We modify the 3D U-Net to additionally output a global feature from the coarsest level of the encoder path to encode global information of the input volume (see Figure 4.2). A single linear layer further maps this global feature to frequencies and phase shifts as a condition signal for the FiLM-ed SIREN layers in the decoder (see section 4.3).

Each voxel in our input volume consists of its occupancy and color and is thus four-dimensional. If the voxel is not occupied by the object, its occupancy and color are set to zero. We denote our input voxel grid as $\boldsymbol{V} \in \mathbb{R}^{r \times r \times r \times 4}$, where r is the voxel grid resolution. In our experiments, we use a voxel resolution of 64. We found in our experiments that 64-resolution outperform 32-resolution voxel grids (see ablation study in section 5.5), and using 128-resolution voxel grids leads to CUDA out-of-memory issues on our machine.

Our 3D U-Net encoder maps the input volume $\boldsymbol{V}$ to feature volume $\boldsymbol{F} \in \mathbb{R}^{r \times r \times r \times d}$ and a global feature $\boldsymbol{g} \in \mathbb{R}^{d_0}$:

$$(\boldsymbol{F}, \boldsymbol{g}) = U(\boldsymbol{V}) \tag{4.1}$$

where U denotes the 3D U-Net. In our experiments, we use $d = 32$ and $d_0 = 256$.

Inspired by [34], we choose the depth of the 3D U-Net such that the receptive field is equal to or larger than the size of the input volume. For the 3D U-Net with depth 3, the voxel at the deepest level (level 2) has a receptive field of 32^3 at the input level. For the 3D U-Net with depth 4, the voxel at the deepest level (level 3) has a receptive field of 68^3 at the input level. Since our input volume resolution is 64^3, we set our 3D U-Net depth to 4. We adapt the 3D U-Net implementation from [48]. The details of our 3D U-Net model architecture and output dimensions are shown in Figure 4.2.

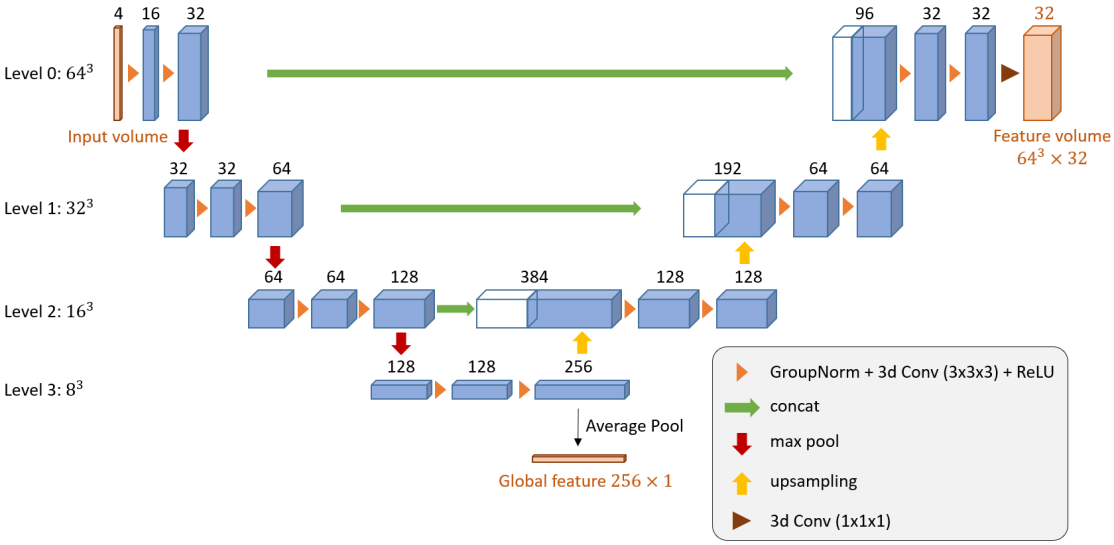


Figure 4.2: Our 3D U-Net [6] architecture. Blue boxes are intermediate feature volumes. Orange boxes are input volume and outputs. The spatial resolutions of intermediate feature volumes are on the figure's left. The number of channels is denoted above each feature volume. Different layers and operations are shown as symbols. The output feature volume is of $64^3 \times 32$ dimension, and the 256×1 dimensional global feature is average pooled from the coarsest level volume.

4.3 Decoder and neural rendering

We represent 3D objects implicitly with a neural radiance field parameterized by our decoder network. The decoder consists of a mapping network ϕ_m of a single linear layer, N FiLM-conditioned SIREN layers $\{\phi_i\}_N$, and an output linear layer ϕ_{out}. The decoder takes as input a 3D point in space $\boldsymbol{x} = (x, y, z)$ and outputs the spatially varying density $\sigma \in \mathbb{R}$ and color $\boldsymbol{c} = (r, g, b) \in \mathbb{R}^3$. Note that our color is independent of the viewing direction since we assume the object surfaces are Lambertian and the rendered images in our dataset are unshaded.

The mapping network maps the global feature $\boldsymbol{g}$ to frequencies $\{\boldsymbol{\gamma}_i\}_{i=0,\cdots,N-1}$ and phase-shifts $\{\boldsymbol{\beta}_i\}_{i=0,\cdots,N-1}$ that condition the FiLM-ed SIREN [35, 44] layers. Each 3D point $\boldsymbol{x} \in \mathbb{R}^3$ is bilinearly interpolated inside the feature volume $\boldsymbol{F} \in \mathbb{R}^{r\times r\times r\times d}$ into a point feature $\boldsymbol{f} \in \mathbb{R}^d$. This point feature is passed to the cascade of FiLM-ed SIREN layers and a final output layer ϕ_{out} to output density σ and color $\boldsymbol{c}$:

$$(\sigma, \boldsymbol{c}) = \phi_{out} \circ \phi_{N-1} \circ \phi_{N-2} \circ \cdots \circ \phi_0(\boldsymbol{f}) \tag{4.2}$$

where

$$\begin{aligned} \phi_{out}(\boldsymbol{x}) &= \boldsymbol{W}_{out}\boldsymbol{x} + \boldsymbol{b}_{out} \\ \phi_i(\boldsymbol{x}_i) &= \sin\left(\boldsymbol{\gamma}_i \cdot (\boldsymbol{W}_i\boldsymbol{x}_i + \boldsymbol{b}_i) + \boldsymbol{\beta}_i\right), \quad i = 0, \cdots, N-1 \\ \phi_{out}(\boldsymbol{x}) &= \boldsymbol{W}_{out}\boldsymbol{x} + \boldsymbol{b}_{out} \\ (\boldsymbol{\gamma}_{0\to N-1}, \boldsymbol{\beta}_{0\to N-1}) &= \phi_m(\boldsymbol{g}) = \boldsymbol{W}_m\boldsymbol{g} + \boldsymbol{b}_m \end{aligned} \tag{4.3}$$

We use volume rendering (see section 2.1) to render the neural radiance field from an arbitrary camera pose. To achieve this, We employ a pinhole camera model and compute the integrals along each ray as it passes through the volume by casting rays from the camera origin. At every sample point along the ray, our decoder predicts the volume density σ and color $\boldsymbol{c}$. The pixel color $\boldsymbol{C}(\boldsymbol{r})$ of camera ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$ with near and far bounds t_n and t_f is given by the volume rendering equation:

$$\boldsymbol{C}(\boldsymbol{r}) = \int_{t_n}^{t_f} T(t)\sigma(\boldsymbol{r}(t))\boldsymbol{c}(\boldsymbol{r}(t))\, dt, \quad \text{where } T(t) = \exp\left(-\int_{t_n}^{t_f} \sigma(\boldsymbol{r}(s))\, ds\right) \tag{4.4}$$

Similarly, the pixel depth d can be computed as:

$$d(\boldsymbol{r}) = \boldsymbol{d}_z^{cam} \int_{t_n}^{t_f} T(t)\sigma(\boldsymbol{r}(t))t\, dt \tag{4.5}$$

where $T(t)$ is given in Equation 4.4 and $\boldsymbol{d}_z^{cam}$ denotes the z component of the unit direction vector in the camera coordinate system. Note that without multiplying the

$\boldsymbol{d}_z^{cam}$ term, the result will be the distance of the pixel along the ray rather than the depth of the pixel.

In our experiments, we use four FiLM-ed SIREN layers since it achieves the best performance among two, four, and eight layers (see ablation study in section 5.5). Our decoder architecture is given in Table 4.1.

Layer name	Layer type	Number of neurons	Activation
Mapping network ϕ_m	Fully connected (FC)	2048 (input 256)	-
FiLM-ed SIREN ϕ_0	FC	256 (input 32)	Sin
FiLM-ed SIREN ϕ_1	FC	256	Sin
FiLM-ed SIREN ϕ_2	FC	256	Sin
FiLM-ed SIREN ϕ_3	FC	256	Sin
Output layer ϕ_{out}	FC	4	-

Table 4.1: The decoder architecture. We use $N = 4$ SIREN layers in experiments. The mapping network maps the 256×1 global feature to condition signals of FiLM-ed SIREN layers. FiLM-ed SIREN layers map the 32×1 interpolated point feature to density and RGB color. The number of neurons in the mapping network depends on the number of FiLM-SIREN layers since the mapping network outputs frequencies and phase shifts to condition the FiLM-SIREN layers. Each FiLM-SIREN layer requires frequency and phase-shift of same size as its number of neurons.

4.4 Discriminator

Following [1] and [16], we use a convolutional discriminator that grows progressively. π-GAN [1] shows that this progressive growing strategy helps stabilize and speed up training since it allows larger batch sizes at the beginning of training.

We start training with low resolutions and large batch sizes so that the generator can concentrate on generating rough geometry and appearances. To accommodate higher resolutions and discriminate fine details, we increase image resolution and add additional layers to the discriminator as training proceeds. Thanks to the implicit 3D representation of our decoder, rendered image resolution can be easily increased by sampling rays more densely at volume rendering. We start training at 32×32 and double the image resolution twice up to 128×128 during training.(see subsection 5.1.2 for details).

Layer	Layer parameters	Output shape
Input image		$3 \times 128 \times 128$
Conv-2d 1	$3 \times 64 \times 1$	$64 \times 128 \times 128$
LeakyReLU	0.2	$64 \times 128 \times 128$
ResidualCoordConvBlock 1	64×128	$128 \times 64 \times 64$
Input image		$3 \times 64 \times 64$
Conv-2d 2	$3 \times 128 \times 1$	$128 \times 64 \times 64$
LeakyReLU	0.2	$128 \times 64 \times 64$
ResidualCoordConvBlock 2	128×256	$256 \times 32 \times 32$
Input image		$3 \times 32 \times 32$
Conv-2d 3	$3 \times 256 \times 1$	$256 \times 32 \times 32$
LeakyReLU	0.2	$256 \times 32 \times 32$
ResidualCoordConvBlock 3	256×400	$400 \times 16 \times 16$
ResidualCoordConvBlock 4	400×400	$400 \times 8 \times 8$
ResidualCoordConvBlock 5	400×400	$400 \times 8 \times 8$
ResidualCoordConvBlock 6	400×400	$400 \times 2 \times 2$
Conv 2d	$400 \times 1 \times 2$	$1 \times 1 \times 1$

Table 4.2: The discriminator architecture, showing progressive growing stages. The indented Conv3d and LeakyReLU layers are only included in the network if it has corresponding input image. For example, if the input image size is 64×64, it will the passed to conv2d 2, LeakyReLU, ResidualCoordConvBlock 2 to 6, and the final conv2d. The layer parameters for conv 2d are in the format of input channel $\times$ output channels $\times$ kernel size. The layer parameter of LeakyReLU is the negative slope. And layer parameters for the ResidualCoordConvBlock are in the format of input channels $\times$ output channels

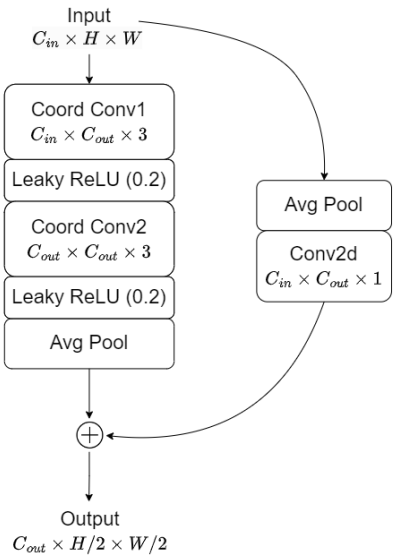


Figure 4.3: The ResidualCoordConvBlock architecture. The parameters of the Coord Conv [21] and the conv 2d are in the format: input channels × output channels × kernel size. The negative slope of LeakyReLU is set to 0.2.

4.5 Losses

We consider three losses: the GAN loss, the photometric loss, and the depth loss.

4.5.1 GAN loss

Our method can be viewed as a generative adversarial network with a conditioned generator. We train our method in a generative adversarial framework in which a generator (in our case, the 3D U-Net encoder U and the decoder Φ) and discriminator D compete in a zero-sum game. At training time, voxel grids $\boldsymbol{V}$ are sampled from the training set with distribution p_V. Real images I are sampled from the training set with distribution p_I. Similar to rendering images in data preparation, camera origins are randomly sampled from an upper-hemispherical volume, and cameras are directed at the origin of the world coordinate system (see subsection 5.1.1 for details). We denote the camera pose ξ distribution as p_ξ. Following [1], we use the non-saturating GAN loss with R1 regularization [24]:

$$\begin{aligned}\mathcal{L}(\theta_D, \theta_\Phi, \theta_U) = \mathbb{E}_{\boldsymbol{V}\sim p_V, \xi\sim p_\xi}\left[f(D(\Phi(U(\boldsymbol{V}), \xi)))\right] \\ + \mathbb{E}_{I\sim p_I}\left[f(-D(I)) + \lambda|\nabla D(I)|^2\right]\end{aligned} \tag{4.6}$$

where $f(u) = -\log(1 + \exp(-u))$, θ_D, θ_Φ and θ_U denotes the parameters of the discriminator, decoder and the 3D U-Net encoder.

In our experiments, we set the R1 regularization weight $\lambda = 10$, since we found smaller λ leads to diverged training.

4.5.2 Phtometric loss

An important difference between our method and GAN is that ours render images based on the input voxel grid $\boldsymbol{V}$ rather than generating a completely new image. Therefore, we can impose a photometric loss on the encoder and the decoder to enforce the rendered images from camera pose ξ to be as close as to the ground truth real images I_ξ of the same camera pose. We use a simple MSE loss as our photometric loss. The photometric loss for a single image is given by:

$$\mathcal{L}(\theta_\Phi, \theta_U; \boldsymbol{V}, \xi, I_\xi) = \frac{1}{H \times W \times 3} \|\Phi(U(\boldsymbol{V}), \xi) - I_\xi\|_F^2 \tag{4.7}$$

where H, W are the image resolution,and $\|\cdot\|_F$ denotes Frobenius norm.

4.5.3 Depth loss

Similar to the photometric loss, we can also compute the depth loss between the depth of the generated image from camera pose ξ and the ground truth depth map d_ξ from the same camera pose. The generated depth map is computed from Equation 4.5. The depth loss for a single image is given by:

$$\mathcal{L}(\theta_\Phi, \theta_U; \boldsymbol{V}, \xi, I_\xi) = \frac{1}{H \times W \times 3} \|\Phi_d(U(\boldsymbol{V}), \xi) - d_\xi\|_F^2 \tag{4.8}$$

where $\Phi_d(U(\boldsymbol{V}), \xi)$ is the generated depth map.

4.6 Metrics

We evaluate image quality using Fréchet Inception Distance (FID) [14], Learned Perceptual Image Patch Similarity (LPIPS) [52], Peak Signal-to-Noise Ratio (PSNR) and FID within each object, which we named object FID, or oFID.

FID and oFID measure the distance between two distributions (two datasets of images). LPIPS measures the perceptual similarity between two images. These three metrics are all based on neural network's hidden activations, and lower values indicate more similarity between the generated and real images. PSNR measures the per-pixel difference between two images, and a higher PSNR value means more similarity and hence better performance.

4.6.1 FID

FID measures the dissimilarity between two sets of images by computing the distance of the feature distributions, which are extracted from the sets of images, respectively. It has been demonstrated to correlate well with human perception of visual quality and is commonly used to evaluate the quality of GANs. FID is calculated by computing the Fréchet distance, also known as Wasserstein-2 distance, between two Gaussians fitted to feature representations of the Inception network [45]. We briefly introduce the computation of FID below.

Given two image datasets $S, S' \subset \Omega$, and a function $f : \Omega \to \mathbb{R}^n$. We compute $f(S), f(S') \subset \mathbb{R}^n$, and fit two gaussian distributions $\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu', \Sigma')$ respectively for $f(S)$ and $f(S')$. The mean μ, μ' are computed respectively as the mean of $f(S), f(S')$, and the covariance Σ, Σ' are computed respectively as the covariance of $f(S), f(S')$. The FID between the two image datasets is the Fréchet distance d_F between the two gaussian distributions:

$$\begin{aligned} \mathrm{FID}(S, S') &= d_F(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu', \Sigma')) \\ &= \|\mu - \mu'\|_2^2 + \mathrm{trace}\left(\Sigma + \Sigma' - 2(\Sigma\Sigma')^{\frac{1}{2}}\right) \end{aligned} \tag{4.9}$$

In practice, we use the `pool3` layer with 2048 dimensional features of the Inception v3 model trained on the ImageNet as the function f. We adopt the FID implementation from [42].

4.6.2 oFID

FID computes the distance between all images across the real and generated dataset. For instance, we compute FID between the generated images and real images of all car objects. However, it is also possible to compute a conditioned FID between real and generated images within each class (in our case, within each object), which can provide a more detailed perceptual similarity measurement for each 3D object. We noticed [29] also used this FID within each class, which they designated as intra FID. We name the FID within each 3D object as object FID (oFID) for clarity.

More precisely, the oFID of two image datasets $S = \bigcup_{y \in \mathcal{Y}} S_y$, $S' = \bigcup_{y \in \mathcal{Y}} S'_y$, where $\mathcal{Y}$ denotes the set of objects and S_y denotes the subset of S with only images from object y, is given as the average of FID of object:

$$\mathrm{oFID}(S, S') = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \mathrm{FID}(S_y, S'_y) \tag{4.10}$$

where the FID of two datasets is given in Equation 4.9.

It is worth mentioning that for oFID computation, we used the 768-dimensional `pre-aux` classifier features instead of `pool3` layer's 2048-dimensional features. Note that for FID, the number of samples to calculate the mean and covariance should be greater than the dimension of the coding layer, e.g. 2048 for the Inception `pool3` layer. Otherwise, the covariance is not full rank, resulting in complex numbers and NaNs by calculating the square root. If `pool3` layer features were used to compute FID for each subset S_y of an object, we would need to render more than 2048 images per object to compute the oFID just for evaluation, which is unnecessary. Instead, we choose the `pre-aux` classifier since it outputs 17×17 many 768-dimensional features per image, and thus at least three images per object are needed to stabilize the covariance matrix computation.

4.6.3 LPIPS

Similar to FID, LPIPS is also a perceptual distance measuring how similar two images are in a way that coincides the human judgment. It computes the similarity between the activations of two image patches for a pre-trained neural network. This measure has been shown to match human perception well. We briefly describe the way to compute LPIPS below. Interested readers are referred to [52] for details.

To compute a distance d_0 between two image patches x, x_0, we first compute deep embeddings, normalize the activations along the channel dimension, scale each channel by vector w, and take the ℓ_2 distance. d_0 is then averaged across spatial dimensions and across all layers. We use the default AlexNet [19] variant of LPIPS in the evaluation.

4.6.4 PSNR

PSNR measures the per-pixel difference between two images $I, I' \in \mathbb{R}^{H \times W \times 3}$:

$$\mathrm{PSNR}(I, I') = 10 \log_{10} \frac{\mathrm{MAX}}{\mathrm{MSE}(I, I')} = -10 \log_{10} \frac{\|I - I'\|_F^2}{H \times W \times 3} \tag{4.11}$$

where MSE denotes the mean square error, and MAX is the maximum possible value of the images, which we set to 1 since our images are in the range of $[0, 1]$.

We compute PNSR for each pair of the generated and real image from the same camera pose of the same object. The final PSNR value between the generated images and real images is averaged over all these pairs.

4.7 PointNet encoder approach

As an alternative to the 3D U-Net encoder, we propose the PointNet [37] encoder approach. In real-world datasets, point clouds are a more direct data format than the

voxel grids, which are the input of our 3D U-Net encoder. Voxelizing point clouds to relatively low-resolution voxel grids will inevitably lose high-frequency details. Directly encoding the point cloud thus has the prospect of learning a latent space that better represents the 3D objects.

PointNet uses max pooling, which is a symmetric function, to deal with unordered input points. Each input point is first fed into a few linear layers, shared by all the points, to high dimensional features. The features are then maxpooled to the global descriptor for the entire shape. We adapt the implementation of PointNet with Resnet blocks from [37]. The Resnet-PointNet architecture is given in Figure 4.5.

In the PointNet encoder approach, we replace the 3D U-Net encoder of the feature volume approach (see Figure 4.1) with the aforementioned Resnet-PointNet. The size of the mapping network ϕ_m is increased since we now rely solely on the global feature output to represent the latent space. The overview of this approach is given in Figure 4.4.

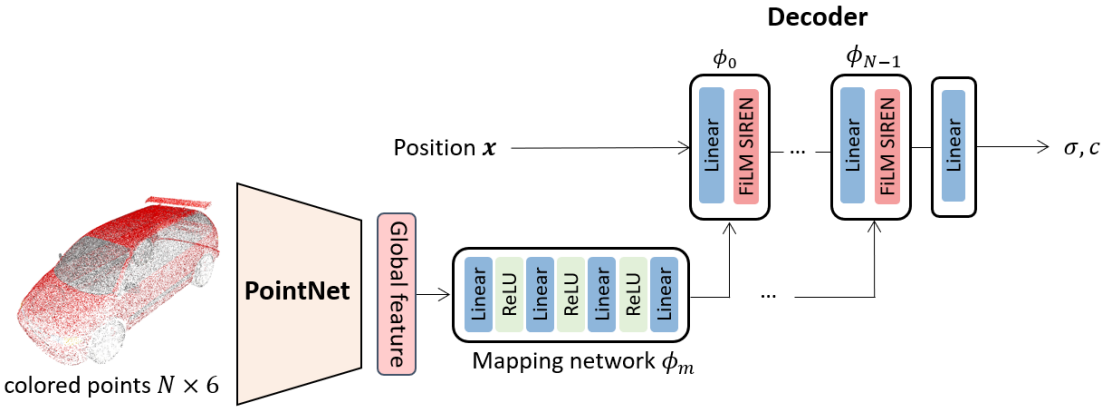
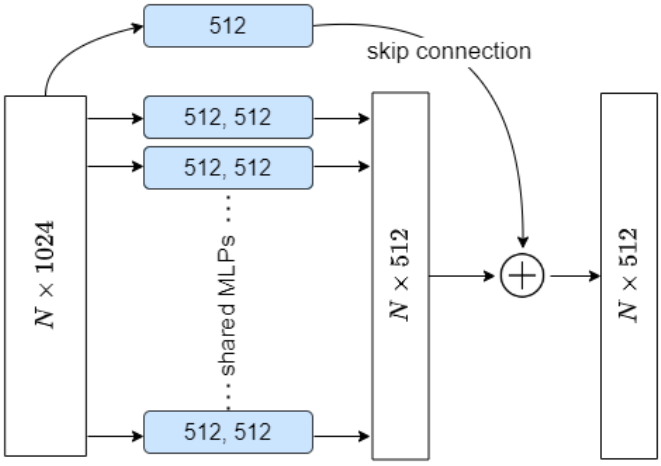
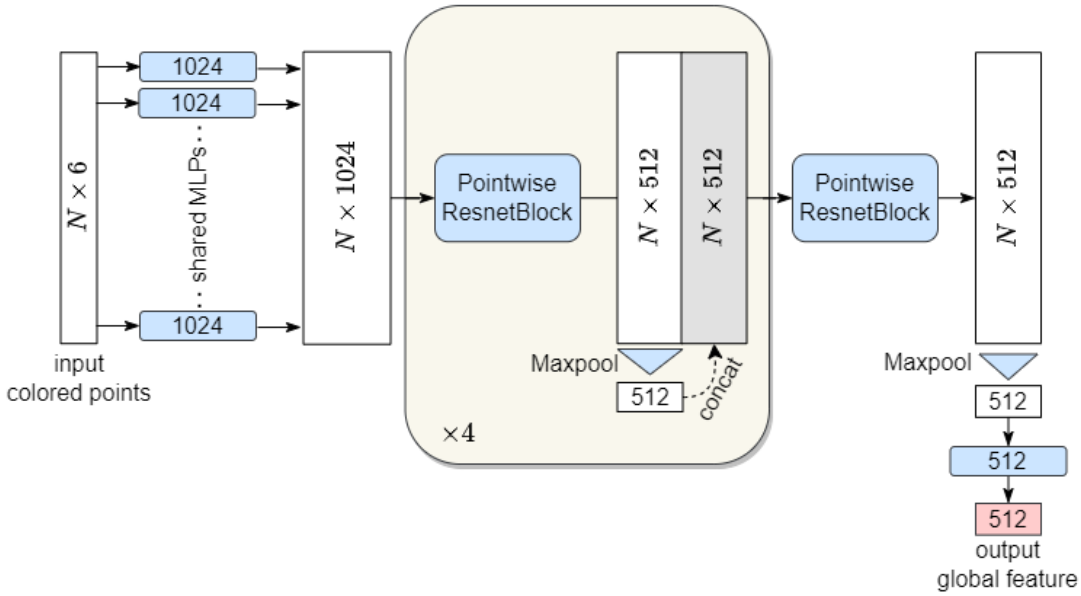


Figure 4.4: The overview of PointNet encoder approach. The rendering and discriminator part are the same as in Figure 4.1 and are thus not shown.



(a) Poinetwise Resnet Block of the ResnetPointNet. Blue boxes denote MLPs, and the numbers inside are the size of linear layers. The activation function for the MLPs is ReLU and is not shown in the diagram. White boxes are the points data.



(b) Our ResnetPointNet architecture. The model detail of Pointwise Resnet Block is given in (a). Note the ×4 in the middle block denotes that this block is repeated four times sequentially.

Figure 4.5: The ResnetPointNet model for the PointNet encoder approach. Shared MLPs for each point and max pooling guarantee the model is invariant against the reordering of input points.

5 Experiments

For the experiments, we focus on the rendering performance on the ShapeNet [3] synthetic dataset.

First, in section 5.1, we mention experimental details relevant to the following experiments. Then, in section 5.2, we evaluate the image quality of our method and compare ours with PointNet encoder approach and voxel surface rendering (introduced in section 5.2). In section 5.3, we evaluate the geometry and multiview consistency of our method's output and compare ours qualitatively with the voxel input geometry. Then, in section 5.4, we explore our method's ability to interpolate in the latent space. Afterward, in section 5.5, we perform ablation studies to investigate important design choices such as encoding methods, network size, loss terms, discriminator conditioning, etc.

5.1 Experimental setup

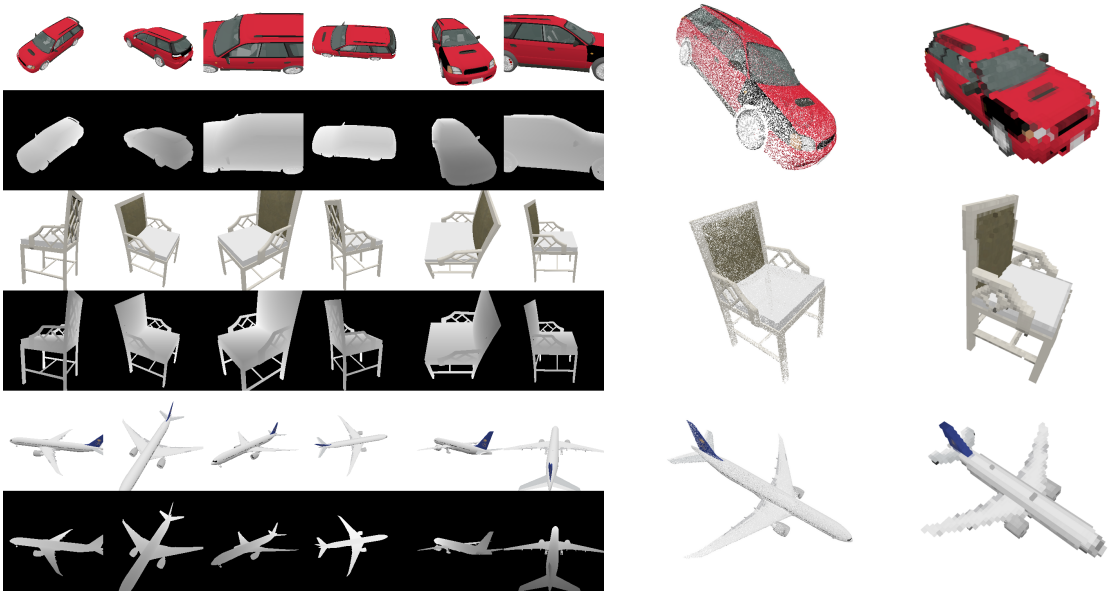
This section lists data preparation and implementation details of our method, either because they are relevant to reproduce our results, or because we consider them to be of interest to the reader.

5.1.1 Dataset and data preparation

We evaluate our method on the ShapeNet dataset.

ShapeNet contains clean 3D models of various categories. We experiment on three categories: car, chair, and plane. For each category, we use 1000 objects in the train set, 20 in the validation set, and 300 in the test set. We render 24 RGB images and depth maps per object and train on 23 images among them, leaving 1 image to validate the novel view synthesis. To reduce complexity, we render unshaded images, and therefore our NeRF model's RGB color output does not depend on the viewing direction. Each 3D object lies within a $[-0.55, 0.55]^3$ bounding volume. Cameras are directed to point toward the world coordinate system origin and are positioned uniformly in an upper hemisphere volume with a radius in the range of $[0.7, 1.5]$ to have zooming effects and to render the same object into different scales onto the image plane.

To obtain the colored voxel grid, we first back-project the rendered RGB images and depth maps to a colored point cloud with 100k points per object. Then, we create a



(a) images and depth maps (b) point clouds (c) voxels

Figure 5.1: ShapeNet dataset. (a) shows the rendered RGB images and their corresponding depth maps for a car, chair, and plane object. (b) shows the back-projected point clouds. (c) shows the visualization of the $64 \times 64 \times 64$ colored voxel grids.

$H \times W \times D \times 4$ voxel grid from the point cloud for each object, where H, W, and D are the voxel resolution, and each voxel has 1 occupancy bit and 3 RGB color values. We use voxel resolution of $64 \times 64 \times 64$, and the effect of voxel resolution is investigated in the ablation study. A voxel is occupied if any points fall into it, and the color value of a given voxel is the average color value of the points that fall into it. Since the point clouds are created by back projection, our colored point clouds and voxel grids only include the surface of an object and are hollow inside. We claim voxel grids with only the surface or shell of the object as occupied are common in real-world datasets since real-world scans only include surface geometry. As all the back-projected point clouds have similar scales and are within the $[-0.55, 0.55]^3$ bounding volume, we create our voxel grids with length 1.2 centered at the origin of the world coordinate system. See Figure 5.1 for dataset samples of RGB images, depth maps, back-projected point clouds, and the voxel grids of a car, chair, and plane object.

5.1.2 Implementation details

Hardware requirements and GPU memory saving techniques All experiments were implemented in `PyTorch`. We train our models across a single RTX 8000 GPU (48GB vRAM), a single A40 GPU (48GB vRAM), or two RTX 6000 GPUs (24GB vRAM). All experiments require less than 16 GB of RAM. Since the NeRF model and voxel grid have high memory costs, we use the following techniques to reduce memory consumption and/or speed up training.

- Distributed Data Parallel (DDP): We use `PyTorch`'s implementation of DDP to achieve data parallelism at the module level, which can run across multiple machines. With DDP, each data batch is split into sub-batches and fed into models on multiple GPUs. Gradients are synchronized; thus, each model on different GPUs is kept the same.
- Batch split: If a single batch is too large to load into the CUDA memory, it can be split into sub-batches and fed into the model every sub-batch. Losses are backpropagated at every sub-batch, and gradients are accumulated, while neural network parameters are updated after the entire batch has been processed.
- Automatic mixed precision training: Mixed precision training tries to match each operation to its appropriate datatype and allows some operations to use half-precision and others to use single-precision. We use `torch.cuda.amp` combined with gradient scaler to reduce runtime and memory usage.

Progressive upsampling As mentioned in section 4.4, we progressively grow the rendered image resolution and decrease correspondingly the batch size. We start training with 32×32 image and batch size 24. At step 5k, we upsample the image to 64×64 and decrease the batch size to 12 to keep the models and generated images in memory. At step 15k, we upsample the image further to 128×128 and keep the batch size as 12. The steps at which images are upsampled are determined such that each training image is input to the model the same amount of times at different resolutions. In our case, each training image of resolution 32×32 and 64×64 are fed into the model an average of 5.2 times.

The learning rates are also correspondingly progressively decayed. We use Adam optimizer [17] with $\beta_1 = 0$, $\beta_2 = 0.9$. The learning rates are initially set to 4×10^{-5} for the decoder, 8×10^{-5} for the 3D U-Net encoder, and 4×10^{-4} for the discriminator. The learning rates of the encoder and decoder are decayed by a factor of 0.5 at every upsampling step. The discriminator learning rate is decreased to 2×10^{-4} after the first

upsampling step and is then further decreased to 2×10^{-6} after the second upsampling step.

It is worth mentioning that we found decaying the discriminator learning rate also by a factor of 0.5 to 1×10^{-4} will diverge the training after the second upsampling step. By analyzing the GAN loss curve, we found after the second upsampling step, the discriminator is very confident that the generated images are fake. This indicates that the discriminator is too powerful for the generator (encoder and decoder), and training left the Nash equilibrium. Therefore, we significantly reduced the learning rate of the discriminator after the second upsampling step, and this issue was resolved.

NeRF parameters We sample 48 coarse points along each ray with near and far ends of $[0.2, 2.0]$. The hierarchical volume sampling further refines the coarsely sampled points to another 48 points along each ray by importance sampling. Our choice of near and far ends of rays guarantees that the entire object is within the range of the rays. The camera distribution is the same at rendering as in the data preparation.

Model details The 4-level 3D U-Net(see section 4.2) maps the $64 \times 64 \times 64 \times 4$ voxel grid to a $64 \times 64 \times 64 \times 32$ feature volume and a 256×1 latent vector, which we dubbed global feature. The global feature vector is subsequently mapped to FiLM-ed SIREN layers' frequencies and phase shifts input by a single linear layer. Unlike π-GAN's four linear layers mapping network, we choose a single linear layer as the mapping network because our global feature already encodes information of the latent space. The sampled ray points are bilinearly interpolated in the feature volume and fed into four FiLM-ed SIREN hidden layers of 256 units each. We found four SIREN layers to be the most suitable for performance and speed. We investigate the number of hidden layers in the ablation study. Same as in π-GAN [1], We use the progressive discriminator with CoordConv layers [21] and residual connections [12].

Metrics and losses We use the GAN loss plus photometric loss (see section 4.5) for most of our experiments. Depth loss is not included because we found it does not improve model performance (see ablation study in section 5.5).

FID, oFID, LPIPS, and PSNR (see section 4.6) are used to evaluate the methods. We rendered 24 real images and generated images from the same pose for all the 300 objects in the test set.

5.2 Evaluating image quality

In this section, we study the rendered image quality on our test set of ShapeNetCar, ShapeNetChair, and ShapeNetPlane. We compare our method with voxel surface rendering (see below) and point cloud encoding (see section 4.7).

5.2.1 Implementation and evaluation details

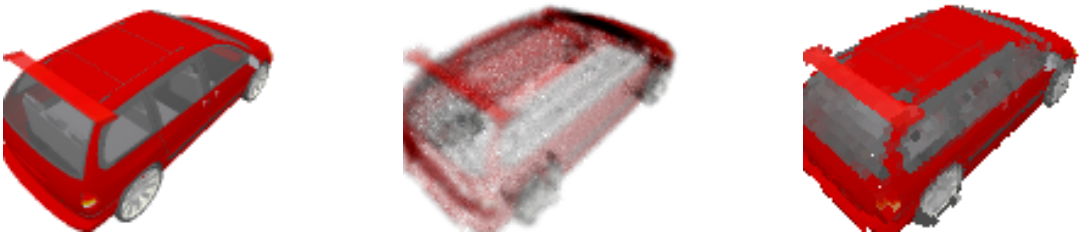
Feature volume method We train our feature volume method both with and without the discriminator. To smooth transitions between upsampling steps, we fade in [16] the contributions of new layers in the discriminator over 2000 iterations. During training, 48 points are sampled along the camera ray and refined by importance sampling. At inference, 96 points are sampled and further refined to produce finer rendering.

PointNet encoder approach The description of this approach is given in section 4.7. We found this approach significantly underperforms the feature volume approach without discriminator. Therefore, we only show the results of this method without the discriminator.

The 6D colored point clouds are encoded to a 512 dimension latent vector by a PointNet [37] based encoder network with ResNet blocks. This latent vector replaces the input Gaussian noise in π-GAN [1] and is mapped to frequencies and phase shifts as inputs of the FiLM-ed SIREN layers. Unlike the feature volume approach's single linear layer mapping network, the latent vector output of the PointNet encoder is passed through four linear layers with LeakyReLU activation. This mapping network setting is borrowed from π-GAN. We experimented on 256, 512, and 1024 dimensions of latent vector and found 512-dimensional latent code has marginal benefits over the other two settings.

To reduce run time and memory consumption, the dense 100k colored point clouds are randomly downsampled. Similar to progressively growing the image rendering resolution, we progressively increase the point cloud downsampling ratio. We use a downsampling ratio of 0.0625 for steps of 32×32 image resolution, 0.25 for steps 64×64 image resolution, and 1 (no downsampling) for steps of 128×128 image resolution.

Furthermore, we observed that the range of the latent vector's value grew increasingly large as training continued. With eight FiLM-ed SIREN layers, the latent vector increased in scale until the floating point overflowed. To prevent this problem and stabilize training, we penalize the l_2 norm of the latent vector by adding a regularizing term to the losses. Since the norm of the latent vector gradually stabilized during training, the weight of the regularizing term is decreased progressively from 0.01 for 32×32 image resolution to 0.001 for 64×64 image resolution and further to 0.0005 for 128×128 image resolution.



(a) Ground truth image (b) Volume rendering (c) Voxel surface rendering

Figure 5.2: Comparison of traditional volume rendering and our voxel surface rendering. (a) shows the ground truth image rendered from the object mesh. (b) shows the image rendered by volume rendering. (c) shows our voxel surface rendering result. Because the voxel grid in our dataset is only occupied at the object surface and is hollow inside, volume rendering will lead to transparent rendering images. Voxel surface rendering avoids this issue by retrieving the color of the first hit point along the ray.

In order to be comparable, we trained our feature volume method and the PointNet encoder approach with four FiLM-ed SIREN layers, and all the other hyperparameters are kept the same. Both approaches are trained to 35k steps on a single A40 or RTX 8000 GPU within 24 hours.

Voxel surface rendering Given our input colored voxel grid, it is possible to render images directly from a specific view. Since our voxel grid only stores the surface geometry and appearance of objects and is hollow inside, volume rendering by integrating the interpolated color and density along all the sampled points on the rays will render images with high transparency. To avoid this, we propose a voxel surface rendering method. Instead of integrating the color and density over all the points along the ray, the color of each ray is set to the color of the first point along this ray that hits the voxel surface. The color of each sampled point is acquired by nearest neighbor interpolation in the voxel grid. We sample 256 points along each ray, and the color of each ray is the color of the first voxel that the ray hits. Voxel surface rendering is a deterministic algorithm with input the same as our method, and hence can be a baseline method to compare our method. Figure 5.2 compares the volume rendering method and our voxel surface rendering.

5.2.2 Evaluation results

The quantitative evaluation results on cars, chairs, and planes of the ShapeNet test set considering the metrics (FID, oFID, LPIPS, PSNR) are given in Table 5.1. The corresponding qualitative results are shown in Figure 5.3. We also compare our method trained on 32^3 resolution voxel grids with voxel surface rendering on the same resolution voxel grids in Table 5.2 and Figure 5.4. The effects of adding discriminator are shown in Figure 5.5. We show more rendering results on the test set of cars, chairs, and planes in Figure 5.6.

Our feature volume method without discriminator on 64^3 dominates over other methods. Voxel surface rendering outputs accurate color and shapes, but the rendering images are zigzaggy at the edges. In comparison, our method outputs images with smooth outlines due to the continuous neural field representation. On 32^3 voxel grids, our method also gives much smoother rendering; however, we noticed ours might lose color information and render white or grey colors (see the last column and the fourth column to the right in Figure 5.4). We did not observe this loss of color in our method on 64^3 voxel grids.

PointNet encoder method is the worst-performing method. We assume this is because of the large information loss after mapping the $10^6 \times 3$ point clouds to a single 512×1 latent vector. This single latent vector lacks the representation power to reconstruct an entire neural field.

5.3 Evaluating geometry

In this section, we study the geometry quality of our feature volume method without discriminator on our test set of ShapeNet cars, chairs, and planes. We compare our method with the geometry of the input voxel grids.

5.3.1 Evaluation details

We keep the same setting as in section 5.2. For each category of cars, chairs, and planes, we train our method without discriminator for 35k steps and extract a density field with 512^3 spatial resolution for objects in the test set. The input voxel grids and the density field output are visualized in ChimeraX [36]. The density threshold has to be manually adjusted for visualization.

5.3.2 Evaluation results

A comparison of our methods trained on 64^3 voxel grids and the input 64^3 voxel grid for cars, chairs, and plans are given in Figure 5.7. We also show the comparison of ours trained on 32^3 voxel grids and the input 32^3 voxel grid for cars in Figure 5.8.

(a) cars

(b) chairs

(c) planes

Figure 5.3: Qualitative comparison on test set of ShapeNet cars, chairs, planes. In each subfigure, first row: ground truth images; second row: voxel surface rendered images; third row: PointNet encoder approach; fourth row: ours with discriminator; fifth row: ours without discrimination. PointNet encoder approach is clearly the worst performing method. Compared with voxel surface rendering, ours produce smoother outlines. Note that ours can recover fine geometric structures (see last column in (a), 3rd column in (b), 10th column in (c)), and even image patterns on the object (see 11th column in (b), and 5th column in (c)).

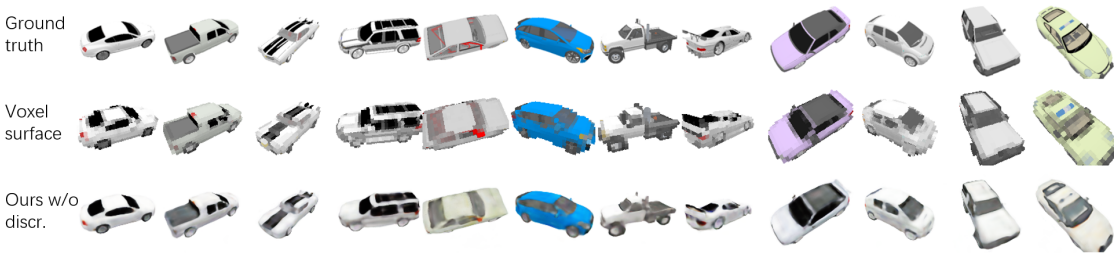


Figure 5.4: Qualitative comparison between voxel surface rendering and ours without discriminator on test set of ShapeNet cars with $32 \times 32 \times 32$ resolution voxel grids. First row: ground truth images; second row: voxel surface rendered images; third row: ours without discriminator. Ours render images with smoother outlines but lost color information on some cars (see first and fourth column to the right). We did not observe this loss of color information on ours trained with $64 \times 64 \times 64$ voxel grids.

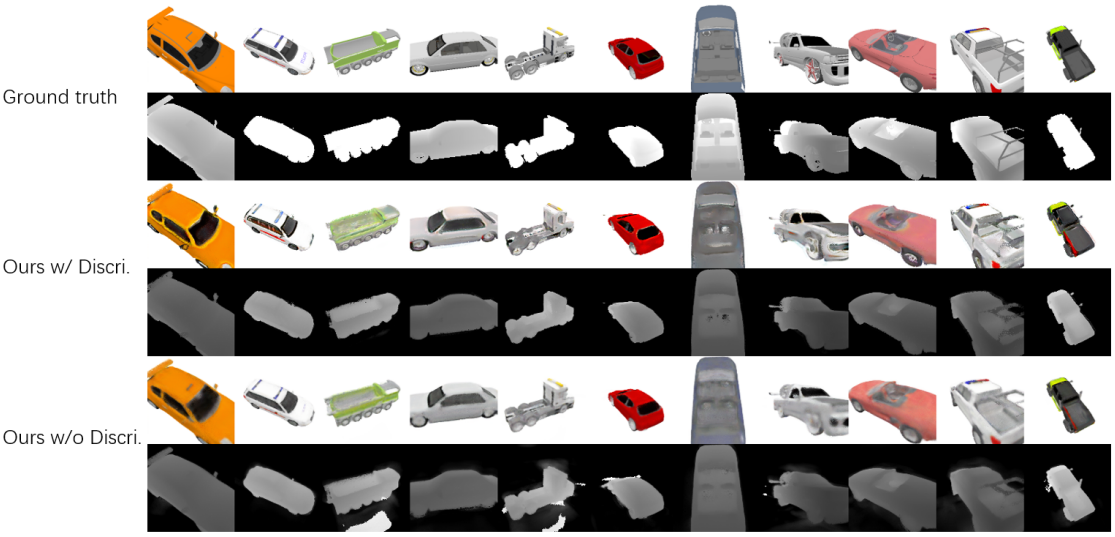


Figure 5.5: The comparison between ours with discriminator and ours without discriminator. We show both the image and its corresponding depth map. Adding discriminator can sharpen the image and produce more accurate and less noisy depth maps. Both models are trained for 35k steps.

Figure 5.6: More rendering results of ours without discriminator on test set of ShapeNet cars, chairs, and planes

	FID↓	oFID↓	LPIPS↓	PSNR↑
Voxel Surface Rendering	75.75	3.88	0.167	17.68
PointNet Encoder	181.95	6.24	0.357	17.14
Ours w/ discri.	**46.27**	**3.81**	0.138	20.26
Ours w/o discri.	56.11	3.84	**0.123**	**23.64**

(a) cars

	FID↓	oFID↓	LPIPS↓	PSNR↑
Voxel Surface Rendering	50.37	**4.22**	0.198	19.44
PointNet Encoder	191.96	7.05	0.437	19.97
ours w/ discri.	29.87	4.71	0.151	23.82
ours w/o discri.	**26.66**	**4.22**	**0.095**	**28.02**

(b) chairs

	FID↓	oFID↓	LPIPS↓	PSNR↑
Voxel Surface Rendering	45.04	**3.88**	0.166	20.54
PointNet Encoder	190.76	6.29	0.248	24.81
ours w/ discri.	44.14	4.32	0.128	25.60
ours w/o discri.	**31.24**	3.93	**0.078**	**29.90**

(c) planes

Table 5.1: Quantitative comparison on test set of ShapeNet cars, chairs, and planes. Ours outperforms other methods. We evaluate PointNet Encoder approach, ours with discriminator and ours without discriminator all at 35k steps.

Compared with the input voxel grids, our method outputs smoother geometry and can render realistic images. The smoothing effect of our method is more obvious in the 32^3 voxel resolution in Figure 5.8.

5.4 Feature interpolation

In this section, we interpolate the latent space of our method and show the rendered images of interpolated features.

(a) cars

(b) chairs

(c) planes

Figure 5.7: Comparison between input geometry and ours on ShapeNet test set. In each subfigure, first row: geometry of input 64^3 voxel grids; second row: density field output of ours without discriminator at 35k steps; third row: an RGB image rendering corresponding to the output geometry of the second row.

	FID↓	oFID↓	LPIPS↓	PSNR↑
Voxel Surface Rendering	126.65	**4.61**	0.246	14.80
ours w/o discri.	**112.90**	4.75	**0.197**	**20.72**

Table 5.2: On test set of ShapeNet cars with $32 \times 32 \times 32$ resolution voxel grids. We train ours with two FiLM-ed SIREN layers and evaluate at 60k steps. Ours outperforms voxel surface rendering on three metrics out of four.

5.4.1 Evaluation details

We evaluate our method without discriminator trained on 64^3 voxel grids for 35k steps. Linear interpolation is conducted between the latent features of two test set objects. Since our method encodes the voxel grids to a $64^3 \times 32$ feature volume and a 256×1 global feature, we experimented with three interpolation strategies to investigate the contribution of feature volume and global feature:

- Interpolate both the feature volume and the global feature;
- Interpolate only the global feature;
- Interpolate only the feature volume.

5.4.2 Evaluation results

The comparison of the three aforementioned interpolation strategies is shown in figure Figure 5.9. Interpolating both the feature volume and global feature produced a smooth transition between the two objects. Interpolating only the global feature failed to transform the source objects into the target object.

We surprisingly found that only interpolating the feature volume and keeping the global feature unchanged can also transform the source objects into the target objects. This indicates that the global feature might be redundant and our feature volume alone is sufficient to learn the latent space distribution. In light of this, we removed the global feature, and the mapping network, and changed the FiLM-ed SIREN layers to vanilla SIREN layers with skip connections. We trained this model with the same settings as before. Interpolating the feature volume of this model is also shown in Figure 5.9. However, removing the global feature leads to degrading performance, as we investigate in the ablation study (see Table 5.3). We show more feature interpolation results on cars, chairs, and planes in Figure 5.10. Note in Figure 5.10, both feature volume and global feature are interpolated.

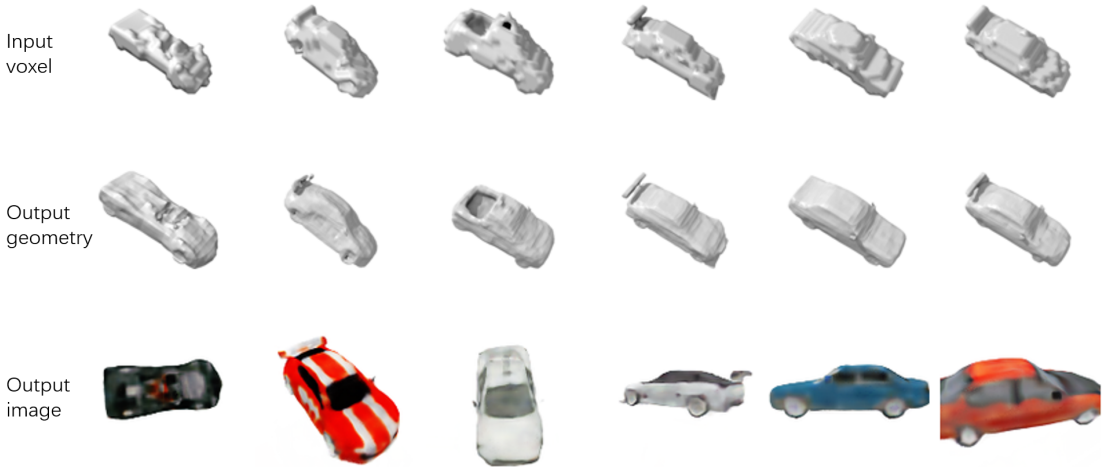


Figure 5.8: Comparison between input geometry and ours on the test set of ShapeNet cars with 32^3 resolution voxel grids. First row: geometry of input 32^3 voxel grids; second row: density field output of ours without discriminator trained on 32^3 voxel grids; third row: an RGB image rendering. Ours gives much smoother and more realistic geometry than input voxel grids.

5.5 Ablation study

This section considers the relevance of feature encoding approaches, depth loss term, the input voxel resolution, the 3D U-Net levels, the length of SIREN layers, and different discriminator conditioning strategies.

5.5.1 Evaluation details

We consider four feature encoding schemes:

- FV + g: Feature volume with global feature;
- FV: Feature volume without global feature, and SIREN layers do not have FiLM conditioning;
- FV w/ skip-layer: Feature volume without feature, SIREN layers do not have FiLM conditioning, and skip-connections are added to the SIREN layers;
- FP + g: Feature pyramid with the global feature. In the feature volume approach, 32-dimensional features of each sampled point along the camera ray are computed by interpolating the position of the points in the output $64^3 \times 32$ feature volume. In

the feature pyramid approach, we interpolate the position of points in each level's output of the 3D U-Net decoder (i.e. the $64^3 \times 32$, $32^3 \times 64$, and $16^3 \times 128$ feature volumes), hence the name feature pyramid.

We also experimented with adding depth loss to the photometric loss, training with 32^3 resolution voxel grids, progressively growing the voxel resolution from 32 to 64 to 128 at every upsampling step, different length of SIREN layers, and a deeper 5-level 3D U-Net. We tried training on 128^3 resolution voxel grids from the start. However, this setting easily leads to CUDA out-of-memory issues on a 48GB A40 or RTX 8000 machine. This could be solved by a more efficient storage method of the sparse voxel data, and we leave it to future work.

During the ablation study, we only change one hyperparameter at a time, and all the other settings are kept the same. Models are trained on ShapeNet cars for 25k steps and are evaluated on the test set with 48 sampling steps.

For the ablation of discriminator conditioning, we compared the input concat [28] and the projection conditioning [29] with no conditioning. The condition signal is a randomly rendered image from the same object. We also experimented with choosing the conditional image as the nearest or the farthest from the input view, and we found choosing the random image has a marginal benefit over the other two. All models in this ablation are trained for 15k steps.

5.5.2 Evaluation results

The quantitative evaluation of different settings is given in Table 5.3. We adopted the setting of the bottom row across all the previous experiments, and this setting outperforms the other settings in two metrics out of four. Feature volume with global features of eight SIREN layers (2nd row to the bottom) has comparable performance. However, this setting is much slower than our adopted setting due to a larger network size.

Results in section 5.4 have shown that interpolating only the feature volume shares similar results as interpolating both the feature volume and global feature, which implies that the global feature might be redundant. However, our ablation shows that removing the global feature will lead to degraded performance (compare 2nd, 3rd, and the bottom row in Table 5.3).

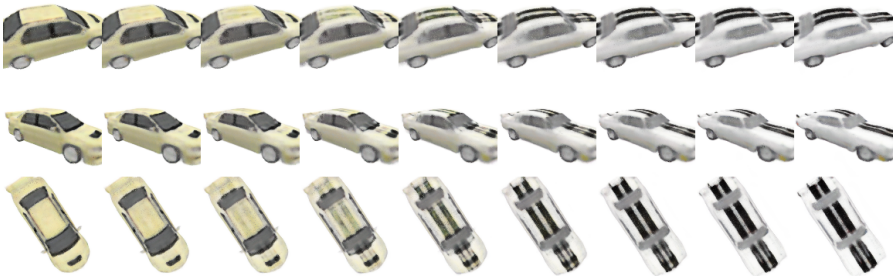
The evaluation of discriminator conditioning is given in Table 5.4. As expected, adding discriminator will increase the perception similarity (FID, oFID and LPIPS) but decrease the pixel-to-pixel image difference (PSNR). Projection conditioning outperforms input concat conditioning but only has a marginal improvement over discriminator without conditioning.

Feature encoding	3D U-Net level	Depth loss	Voxel Res.	# SIREN	FID↓	oFID↓	LPIPS↓	PSNR↑
FP + g	4	✗	64	4	72.56	4.18	0.130	**23.40**
FV	4	✗	64	4	122.06	5.26	0.194	21.71
FV w/ skip-layer	4	✗	64	4	92.71	4.50	0.153	22.70
FV + g	5	✗	64	4	74.30	4.20	0.130	23.24
FV + g	4	✓	64	4	75.48	4.22	0.131	23.26
FV + g	4	✗	32	4	132.06	5.07	0.201	20.96
FV + g	4	✗	32 → 128	4	190.02	7.32	0.481	15.35
FV + g	4	✗	64	2	75.19	4.22	0.131	23.30
FV + g	4	✗	64	8	**69.00**	4.17	**0.128**	23.35
FV + g	4	✗	64	4	71.30	**4.09**	**0.128**	23.36

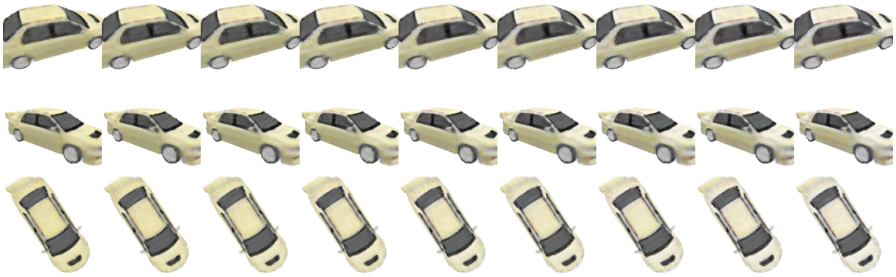
Table 5.3: Ablation study. Each row represents one experiment setting and only differs at one setting from the bottom row (indicated by blue), which is our adopted setting across all the previous experiments. Considering both speed and performance, our adopted setting is the best among the others. Progressively increasing the input voxel resolution unexpectedly diverged after upsampling the voxel grid; this indicates that our 3D U-Net does not scale to different input voxel resolutions. Adopting a progressively growing architecture of 3D U-Net as in [16] might solve this problem. We leave this to future work.

discriminator style	FID↓	oFID↓	LPIPS↓	PSNR↑
no conditioning	**53.06**	4.10	0.144	21.42
Input concat	60.60	4.18	0.140	21.30
Projection	53.86	**4.09**	**0.137**	21.45
✗	72.70	4.40	0.157	**22.95**

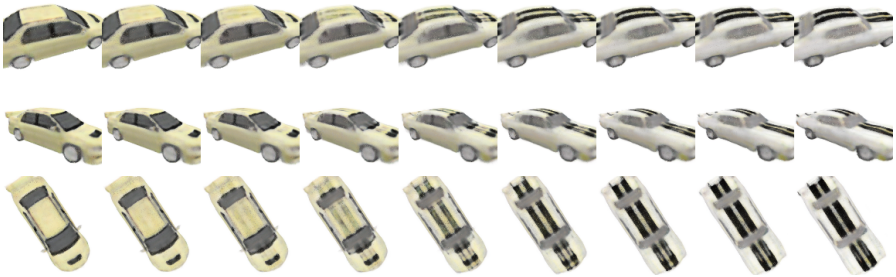
Table 5.4: Ablation of discriminator conditioning. First row: unconditioned discriminator; second row: discriminator with input concat conditioning; third row: discriminator with projection conditioning; fourth row: without a discriminator.



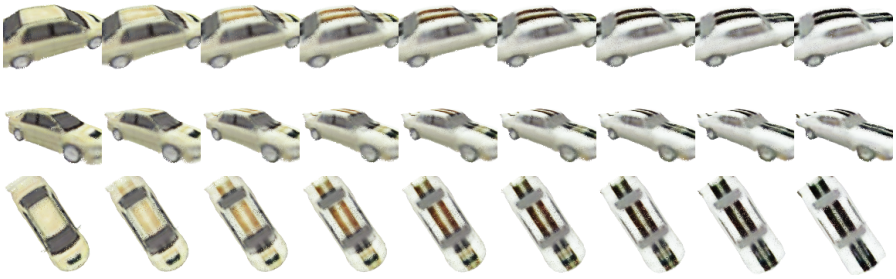
(a) Interpolate both the feature volume and global feature



(b) Only interpolate the global feature



(c) Only interpolate the feature volume



(d) Interpolate feature volume on the model with global feature and mapping network removed.

Figure 5.9: Feature interpolation. Interpolating the feature volume alone can transform the yellow car into a white car with stripes.

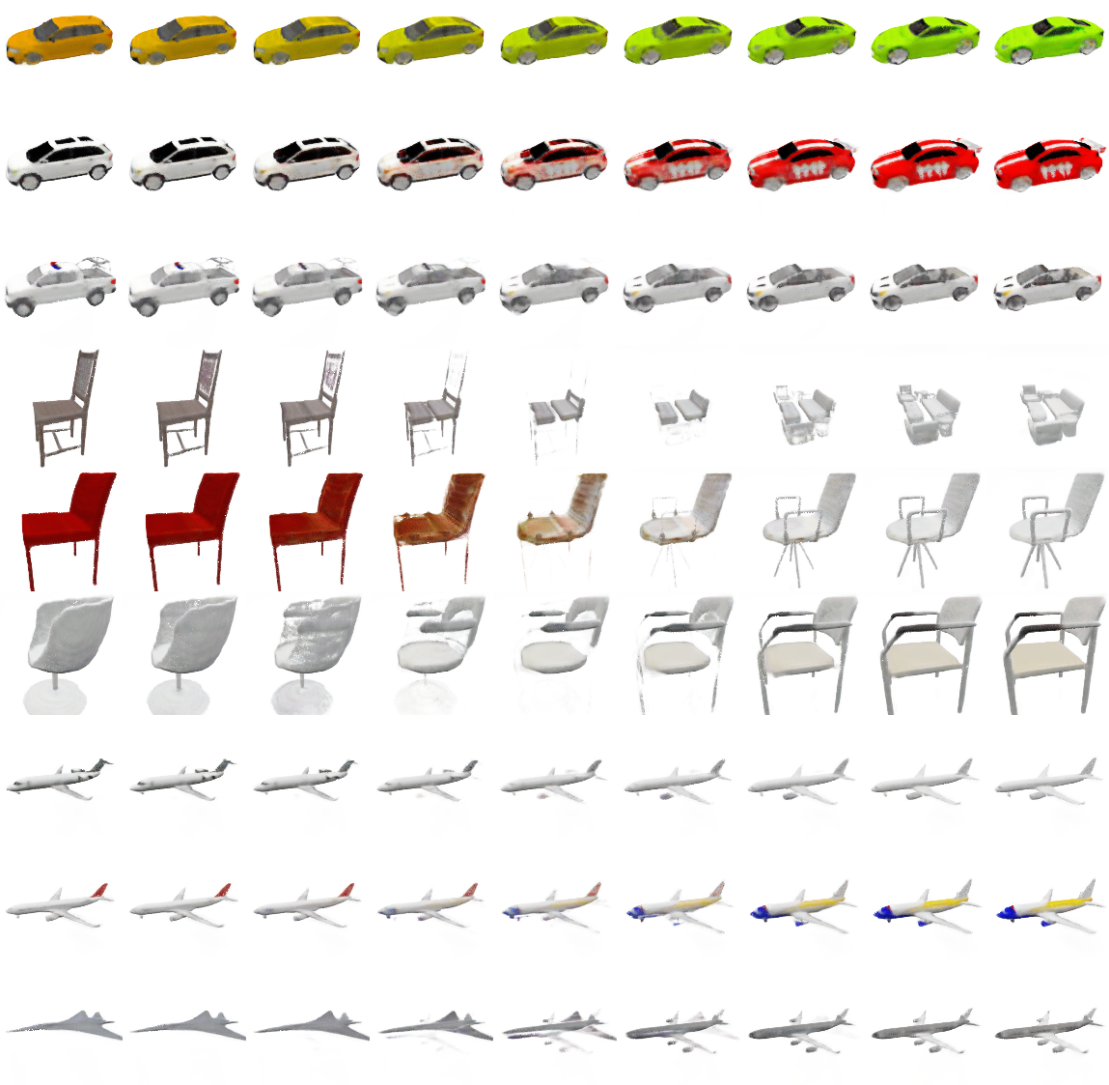


Figure 5.10: More feature interpolation results. The smooth transition of color and geometry from one object to another shows our method efficiently learned the latent space distribution. Since the feature volume includes strong spatial information, row 4, 6, and 9 show some artifacts (e.g. parts of the source and target object coexist) in the middle columns.

6 Conclusion

We briefly discuss limitations and future work and then summarize this work.

6.1 Limitations and future work

One limitation of the proposed method is the lack of evaluation on low-quality input. The input voxel grids to our method from the ShapeNet synthetic dataset are already of relatively good quality. The geometry and color in input voxel grids are accurate and have no noise. Also, note that our method works in the canonical space, where all objects are aligned to a consistent orientation. This makes the learning of 3D representation much easier. For example, if all the cars are aligned and heading in the same direction, NeRF can easily learn to represent an average car or a rough shape of cars and only needs to modify the average car for a new car's input. However, in real-world datasets, neither high-quality geometry nor aligned objects are accessible. The geometry of real-world data set, such as CO3D [39], are point clouds acquired from the back-projected depth and RGB images and are often noisy due to inaccurate pose and depth predictions. And the point clouds are not aligned to a canonical pose, which NeRF will struggle on. Training our method on real-world datasets will improve its robustness against noisy inputs. Our discriminator can, in theory, deal with noisy inputs since it forces the generator to render realistic images even if the input geometry is inaccurate. Noise in the input geometry will have a minor influence on the neural radiance field representation if our discriminator is well-trained. For the unaligned object problem on real-world datasets, future work could include a transformation prediction module, such as the T-Net module in PointNet [37], to align the geometry before passing it to the encoder.

Another limitation of our method is that we did not fully exploit the discriminator to improve the quality of the neural radiance field. We expected a significant performance improvement with the introduction of the discriminator and the adversarial loss. In our experiments, including the discriminator improved sharpness and depth predictions but did not boost the visual quality significantly. Also, our experiments show that training with discriminators is prone to diverged training. We resolved this issue partly by decreasing the discriminator learning rate. We believe fine-tuning the discriminator hyperparameters and a better way to condition and train the discriminator will lead to more performance improvement.

During training, we only generate one image for one object at each step. In future work, generating multiple images of one object at each step and feeding those synthesized images together with multiple real images from a single object into the discriminator (similar in [43] could help increase multi-view consistency and faster NeRF's convergence.

One potential improvement is the encoder module. Experiments showed that encoding the voxel grids by 3D U-Net and interpolating in the feature volume is far more effective than encoding the point clouds by PointNet (see section 5.2). However, we assume this is due to the feature interpolation rather than the encoder. In our feature volume approach, point position input to the neural radiance field is replaced by interpolated features in the feature volume. We didn't interpolate the point position with the point features in the PointNet encoder approach since direct interpolation is only feasible within a volume. As a result, the PointNet encoder approach did not condition the neural field with rich local information. Future work could explore interpolating the point position with encoded point features, as in Point-NeRF [37]. Encoding local information of 3D objects in a volume is restricted by the voxel grid resolution, which limits the performance of our method. Also, voxelized grids contain fewer high-frequency geometry details due to the inevitable information loss after voxelization. We believe adopting a point cloud encoding method with a similar global and local features encoding scheme can greatly improve the performance of our method.

Another potential future research direction is the volume discriminator. Current 3D GANs still employ a 2D discriminator based on images. We think it is possible to design a 3D discriminator which takes in 3D input, such as voxel grids, point clouds, or meshes, and discriminates the generated 3D geometry and the real geometry.

6.2 Conclusion

This work's main contribution is a geometry-conditioned GAN framework with a conditioned neural radiance field that can render realistic images with high visual quality and details from any view, which is enabled by our local (feature volume) and global (latent global feature) encoding scheme.

We show in experiments (see section 5.2) that our method can generate images that are more realistic than directly rendered from the input geometry. This is primarily due to the effective feature volume encoding method that can extract deep expressive features from the geometry and the learned continuous neural radiance field that can reconstruct and refine the geometry based on the latent features. We validate our design decisions through an extensive ablation study (see section 5.5).

Our experiments (see section 5.3) show that the reconstructed geometry represented by the neural radiance field is of high quality and has smoother outlines than the input

voxel grids. Moreover, with the help of the conditioned GAN framework, our method can interpolate in the latent space and generate variants of the input geometry (see section 5.4).

Bibliography

[1] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 5799–5809.

[2] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. "Efficient Geometry-aware 3D Generative Adversarial Networks." In: *arXiv*. 2021.

[3] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[4] Z. Chen and H. Zhang. "Learning implicit fields for generative shape modeling." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5939–5948.

[5] J. Chibane, T. Alldieck, and G. Pons-Moll. "Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. June 2020.

[6] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. "3D U-Net: learning dense volumetric segmentation from sparse annotation." In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

[7] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan. "Depth-supervised NeRF: Fewer Views and Faster Training for Free." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: *International Conference on Learning Representations*. 2021.

[9] J. Gao, W. Chen, T. Xiang, A. Jacobson, M. McGuire, and S. Fidler. "Learning deformable tetrahedral meshes for 3d reconstruction." In: *Advances In Neural Information Processing Systems* 33 (2020), pp. 9936–9947.

[10] J. Gao, T. Shen, Z. Wang, W. Chen, K. Yin, D. Li, O. Litany, Z. Gojcic, and S. Fidler. "GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images." In: *Advances In Neural Information Processing Systems*. 2022.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger. Vol. 27. Curran Associates, Inc., 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[13] P. Henzler, N. J. Mitra, and T. Ritschel. "Escaping plato's cave: 3d shape from adversarial rendering." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9984–9993.

[14] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. "GANs trained by a two time-scale update rule converge to a local nash equilibrium." In: *Advances in neural information processing systems* 30 (2017).

[15] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole. "Zero-Shot Text-Guided Object Generation with Dream Fields." In: (2022).

[16] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation." In: *International Conference on Learning Representations*. 2018.

[17] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[18] L. Koestler, N. Yang, N. Zeller, and D. Cremers. "TANDEM: Tracking and Dense Mapping in Real-time using Deep Multi-view Stereo." In: *Conference on Robot Learning (CoRL)*. 2021. arXiv: 2111.07418.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Vol. 25. Curran Associates, Inc., 2012.

[20] K.-E. Lin, L. Yen-Chen, W.-S. Lai, T.-Y. Lin, Y.-C. Shih, and R. Ramamoorthi. "Vision Transformer for NeRF-Based View Synthesis from a Single Input Image." In: *WACV*. 2023.

[21] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski. "An intriguing failing of convolutional neural networks and the coordconv solution." In: *Advances in neural information processing systems* 31 (2018).

[22] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. "Neural Volumes: Learning Dynamic Renderable Volumes from Images." In: *ACM Trans. Graph.* 38.4 (July 2019), 65:1–65:14.

[23] N. Max. "Optical models for direct volume rendering." In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pp. 99–108.

[24] L. Mescheder, A. Geiger, and S. Nowozin. "Which training methods for GANs do actually converge?" In: *International conference on machine learning*. PMLR. 2018, pp. 3481–3490.

[25] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. "Occupancy networks: Learning 3d reconstruction in function space." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4460–4470.

[26] L. Mi, A. Kundu, D. Ross, F. Dellaert, N. Snavely, and A. Fathi. "im2nerf: Image to neural radiance field in the wild." In: *arXiv preprint arXiv:2209.04061* (2022).

[27] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." In: *ECCV*. 2020.

[28] M. Mirza and S. Osindero. "Conditional generative adversarial nets." In: *arXiv preprint arXiv:1411.1784* (2014).

[29] T. Miyato and M. Koyama. "cGANs with Projection Discriminator." In: *International Conference on Learning Representations*. 2018.

[30] T. Müller, A. Evans, C. Schied, and A. Keller. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding." In: *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127.

[31] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang. "Hologan: Unsupervised learning of 3d representations from natural images." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7588–7597.

[32] M. Niemeyer and A. Geiger. "GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields." In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[33] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[34] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. "Convolutional occupancy networks." In: *European Conference on Computer Vision*. Springer. 2020, pp. 523–540.

[35] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. "Film: Visual reasoning with a general conditioning layer." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[36] E. F. Pettersen, T. D. Goddard, C. C. Huang, E. C. Meng, G. S. Couch, T. I. Croll, J. H. Morris, and T. E. Ferrin. "UCSF ChimeraX: Structure visualization for researchers, educators, and developers." In: *Protein Science* 30.1 (2021), pp. 70–82.

[37] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[38] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. "Learning transferable visual models from natural language supervision." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[39] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny. "Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction." In: *International Conference on Computer Vision*. 2021.

[40] B. Roessle, J. T. Barron, B. Mildenhall, P. P. Srinivasan, and M. Nießner. "Dense depth priors for neural radiance fields from sparse input views." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12892–12901.

[41] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. "GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis." In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[42] M. Seitzer. *pytorch-fid: FID Score for PyTorch*. `https://github.com/mseitzer/pytorch-fid`. Version 0.3.0. Aug. 2020.

[43] Y. Siddiqui, J. Thies, F. Ma, Q. Shan, M. Nießner, and A. Dai. "Texturify: Generating Textures on 3D Shape Surfaces." In: *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part III*. Vol. 13663. Springer, 2022, pp. 72–88.

[44] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. "Implicit neural representations with periodic activation functions." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7462–7473.

[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[46] M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. T. Barron, and R. Ng. "Learned initializations for optimizing coordinate-based neural representations." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2846–2855.

[47] A. Tewari, X. Pan, O. Fried, M. Agrawala, C. Theobalt, et al. "Disentangled3D: Learning a 3D Generative Model with Disentangled Geometry and Appearance from Monocular Images." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1516–1525.

[48] A. Wolny. *3d u-net model for volumetric semantic segmentation written in pytorch.* URL: `https://github.com/wolny/pytorch-3dunet` (visited on 01/10/2023).

[49] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann. "Point-nerf: Point-based neural radiance fields." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5438–5448.

[50] Y. Ye, S. Tulsiani, and A. Gupta. "Shelf-supervised mesh prediction in the wild." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8843–8852.

[51] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. "pixelNeRF: Neural Radiance Fields from One or Few Images." In: *CVPR*. 2021.

[52] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[53] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022.

[54] D. Zimny, T. Trzciński, and P. Spurek. "Points2NeRF: Generating Neural Radiance Fields from 3D point cloud." In: *arXiv preprint arXiv:2206.01290* (2022).