NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review

Kyle (Yilin) Gao, *Graduate Student Member, IEEE,* Yina Gao, Hongjie He, Dening Lu, Linlin Xu, *Member, IEEE,* Jonathan Li, *Senior Member, IEEE*

Abstract—Neural Radiance Field (NeRF), a new novel view synthesis with implicit scene representation has taken the field of Computer Vision by storm. As a novel view synthesis and 3D reconstruction method, NeRF models find applications in robotics, urban mapping, autonomous navigation, virtual reality/augmented reality, and more. Since the original paper by Mildenhall et al., more than 250 preprints were published, with more than 100 eventually being accepted in tier one Computer Vision Conferences. Given NeRF popularity and the current interest in this research area, we believe it necessary to compile a comprehensive survey of NeRF papers from the past two years, which we organized into both architecture, and application based taxonomies. We also provide an introduction to the theory of NeRF based novel view synthesis, and a benchmark comparison of the performance and speed of key NeRF models. By creating this survey, we hope to introduce new researchers to NeRF, provide a helpful reference for influential works in this field, as well as motivate future research directions with our discussion section.

Index Terms—Neural Radiance Field, NeRF, Computer Vision Survey, Novel View Synthesis, Neural Rendering, 3D Reconstruction

1 Introduction

Neural Radiance Field (NeRF) models are novel view synthesis methods which use volume rendering with implicit neural scene representation via Multi Layer Perceptrons (MLPs). First introduced in ECCV 2020 by Mildenhall et al. [1], NeRF has achieved state of the art visual quality, produced impressive demonstrations, and inspired many subsequent works derived from this novel method. In the recent past (2022), NeRF models have found applications in photo-editing, 3D surface extraction, and large/city-scale 3D representation and view synthesis.

NeRF models have a few key advantages over other methods of novel view synthesis and scene representation.

- NeRF models are self supervised. They can be trained using only multi-view images of a scene. Unlike many other 3D neural representation or view synthesis methods, NeRF models require only images and poses to learn a scene, and do not require 3D/depth supervision. The poses can also be estimated using Structure from Motion (SfM) packages such as COLMAP [2], as was done in certain scenes in the original NeRF paper.
- NeRF models are photo-realistic. Compared to classical techniques such as [3] [4], as well as earlier novel view synthesis methods such as [5][6][7], neural 3D representation methods [8][9][10], the original NeRF model converged to better results in terms of visual quality, with more recent models performing even better.

NeRF models have attracted much attention in the computer vision community in the past two years, with more than 150 papers and preprints appearing on popular code aggregation website [1], and more than 200 preprints on arXiv. Many of the preprints were eventually published in top tier computer vision conferences such as CVPR, ICCV, and ECCV, with CVPR 2021 less than 10 NeRF papers, and CVPR 2022 publishing more than 50 papers on this topic. Similar trends can be seen in the other computer vision conferences as well. In 2022, the impact of NeRF is large and ever increasing, with the original NeRF paper by Mildenhall et al. receiving more than 1300 citations, and growing interest year-over-year. Given current interest and lack of existing comprehensive survey papers, we believe it necessary to organize a one such paper to help computer vision practitioners with this new topic.

The rest of this manuscript is organized as follows.

- Section 2 introduces existing NeRF surveys preprints (2.1), explains the theory behind NeRF volume rendering (2.2), introduces the commonly used datasets (2.3) and quality assessment metrics (2.4).
- Section 3 is the core of the paper, and introduces the influential NeRF publications, and contains the taxonomy we created to organize these works. Its subsections detail the different families of NeRF innovations proposed in the past two years, as well as recent applications of NeRF models to various computer vision tasks.
- Sections 4 and 5 discuss potential future research directions and applications, and summarize the survey.

1. https://paperswithcode.com/method/nerf

- *Corresponding authors: Jonathan Li, Linlin Xu.*
- *This research was partially funded by the Natural Science and Engineering Research Council of Canada under Grant RGPIN-2022-03741 and the Mitacs Accelerate Program under Project IT32340.*
- *Kyle Gao, Dening Lu, Linlin Xu, and Jonathan Li are with the Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontraio N2L 3G1, Canada (e-mail: y56gao, d62lu, l44xu, junli@uwaterloo.ca).*
- *Hongjie He and Jonathan Li are with the Department of Geography and Environmental Management, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada (e-mail: h69he@uwaterloo.ca , junli@@uwaterloo.ca).*
- *Yina Gao is with the Faculty of Engineering, University of Toronto (e-mail: Yina.gao@mail.utoronto.ca).*

2 BACKGROUND

2.1 Existing NeRF Surveys

In December 2020, Dellart and Yen-Chen published a fairly comprehensive preprint NeRF survey [11] including approximately 50 NeRF publications/preprints, many of which were eventually published in top tier computer vision conferences. We took inspiration from this preprint survey and used it as a starting point for our own survey. However, the work is only five pages, and does not include detailed descriptions. Moreover, the work only included NeRF papers from 2020 and early 2021 preprints, and was missing many later influential papers.

In December 2021, Zhang et al. [12] published a preprint survey for Multimodal image synthesis and editing in which they dedicated a paragraph for NeRF. They mostly focused on multi-modal NeRFs such as [13], [14], only citing these two and the original NeRF paper [1] in their survey, as well as four more papers [15] [16][17][18] in their supplementary materials.

In May 2022, Tewari et al. [19] published a state-of-the-art report on advances in Neural Rendering with a focus on NeRF models. It is to date the most comprehensive Neural Rendering survey style report, including many influential NeRF papers, as well as many other Neural Rendering papers. Our survey differs from this report in that our scope is completely focused on NeRF papers, giving detailed paper-by-paper summary of selected works. We also present a NeRF innovation technique taxonomy tree, and a NeRF application classification tree. We are able to include most of the 50+ NeRF based papers from CVPR 2022 (June 2022) in our survey.

2.2 Neural Radiance Field (NeRF) Theory

Neural Radiance Fields were first proposed by Mildenhall et al. [1] in 2020 for novel view synthesis. NeRFs achieved highly photo-realistic view synthesis of complex scenes and attracted much attention in the field. In its basic form, a NeRF model represents three-dimensional scenes as a radiance field approximated by a neural network. The radiance field describes color and volume density for every point and for every viewing direction in the scene. This is written as:

$$F(\mathbf{x}, \theta, \phi) \rightarrow (\mathbf{c}, \sigma), \tag{1}$$

where $\mathbf{x} = (x, y, z)$ is the in-scene coordinate, (θ, ϕ) represent the azimuthal and polar viewing angles, $\mathbf{c} = (r, g, b)$ represents color, and σ represents the volume density. This 5D function is approximated by one or more Multi-Layer Preceptron (MLP) sometimes denoted as F_Θ. The two viewing angles (θ, ϕ) are often represented by $\mathbf{d} = (d_x, d_y, d_z)$, a 3D Cartesian unit vector. This neural network representation is constrained to be multi-view consistent by restricting the prediction of σ, the volume density (i.e., the content of the scene) to be independent of viewing direction, whereas the color $\mathbf{c}$ is allowed to depend on both viewing direction and in-scene coordinate. In the baseline NeRF model, this is implemented by designing the MLP to be in two-stages. The first stage takes as input $\mathbf{x}$ and outputs σ and a high-dimensional feature vector (256 in the original paper). In the second stage, the feature vector is then concatenated with the viewing direction $\mathbf{d}$, and passed to an additional MLP, which outputs $\mathbf{c}$. We note that Mildenhall et al. [1] consider the σ MLP and the $\mathbf{c}$ MLP to be two branches of the same neural network, but many subsequent authors consider them to be two separate MLP networks, a convention which we follow from this point on.

Broadly speaking, novel view synthesis using a trained NeRF model is as follows.

- For each pixel in the image being synthesized, send camera rays through the scene and generate a set of sampling points (see (a) in Fig. 1).
- For each sampling point, use the viewing direction and sampling location to extract local color and density, as computed by NeRF MLP(s) (see (b) in Fig. 1).
- Use volume rendering to produce the image from these colors and densities (see (c) in Fig. 1).

In more detail, given volume density and color functions, volume rendering [20] is used to obtain the color $C(\mathbf{r})$ of any camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, with camera position $\mathbf{o}$ and viewing direction $\mathbf{d}$ using

$$C(\mathbf{r}) = \int_{t_1}^{t_2} T(t) \cdot \sigma(\mathbf{r}(t)) \cdot \mathbf{c}(\mathbf{r}(t), \mathbf{d}) \cdot dt, \tag{2}$$

where $T(t)$ is the accumulated transmittance, representing the probability that the ray travels from t_1 to t without being intercepted, given by

$$T(t) = \exp(-\int_{t_1}^{t} \sigma(\mathbf{r}(u)) \cdot du). \tag{3}$$

Novel views are rendered by tracing the camera rays $C(\mathbf{r})$ through each pixel of the to-be-synthesized image. This integral can be computed numerically. The original implementation [1] and most subsequent methods used a non-deterministic stratified sampling approach, where the ray was divided into N equally spaced bins, and a sample was uniformly drawn from each bin. Then, equation (2) can be approximated as

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i T_i \mathbf{c}_i, \text{ where } T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j). \tag{4}$$

δ_i is the distance from sample i to sample $i+1$. $(\sigma_i, \mathbf{c_i})$ are the density and color evaluated along the sample point i given the ray, as computed by the NeRF MLP(s). α_i the transparency/opacity from alpha compositing at sample point i, is given by

$$\alpha_i = 1 - \exp(\sigma_i \delta_i). \tag{5}$$

An expected depth can be calculated for the ray using the accumulated transmittance as

$$d(\mathbf{r}) = \int_{t_1}^{t_2} T(t) \cdot \sigma(\mathbf{r}(t)) \cdot t \cdot dt. \tag{6}$$

This can be approximated analogously to equation (4) approximating equation (2) and (3)

$$\hat{D}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i t_i T_i. \tag{7}$$

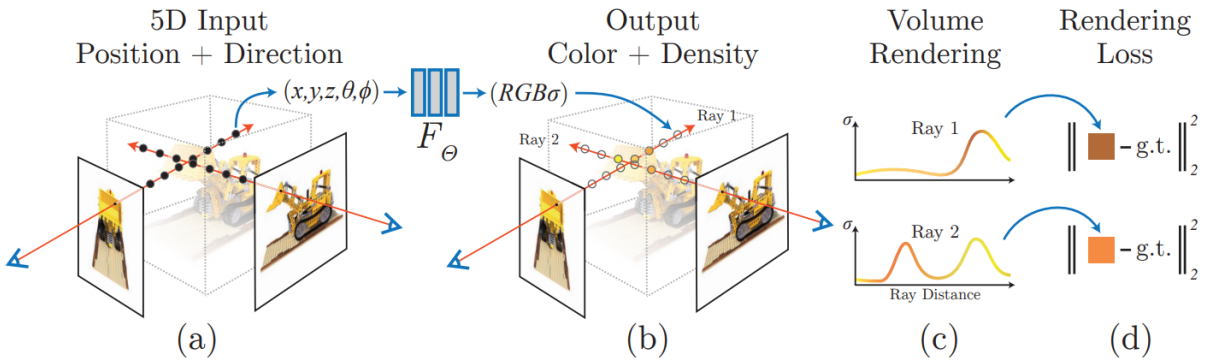

Fig. 1. The NeRF volume rendering and training process. Image sourced from [1]. (a) illustrates the selection of sampling points for individual pixels in a to-be-synthesized image. (b) illustrates the generation of densities and colors at the sampling points using NeRF MLP(s). (c) and (d) illustrate the generation of individual pixel color(s) using in-scene colors and densities along the associated camera ray(s) via volume rendering, and the comparison to ground truth pixel color(s), respectively.

Certain depth regularization [21] [22] [23] [24] methods use the expected depth to restrict densities to delta-like functions at scene surfaces, or to enforce depth smoothness.

For each pixel, a square error photometric loss is used to optimize the MLP parameters. Over the entire image, this is given by

$$L = \sum_{r \in R} ||\hat{C}(\mathbf{r}) - C_{gt}(\mathbf{r})||_2^2 \tag{8}$$

where $C_{gt}(\mathbf{r})$ is the ground truth color of the training image's pixel associated to $\mathbf{r}$, and R is the batch of rays associated to the to-be-synthesized image.

NeRF models often employ positional encoding, which was shown by Mildenhall et al. [1] to greatly improve fine detail reconstruction in the rendered views. This was also shown in more details, with corroborating theory using Neural Tangent Kernels in [25]. In the original implementation, the following positional encoding γ was applied to each component of the scene coordinate $\mathbf{x}$ (normalized to [-1,1]) and viewing direction unit vector $\mathbf{d}$

$$\gamma(v) = (sin(2^0\pi v), cos(2^0\pi v), sin(2^1\pi v), cos(2^1\pi v), \\ ..., sin(2^{N-1}\pi v), cos(2^{N-1}\pi v)), \tag{9}$$

where N is a user determined encoding dimensionality parameter, set to $N = 10$ for $\mathbf{x}$ and $N = 4$ for $\mathbf{d}$ in the original paper. However, modern researches have experimented and achieved great results with alternate forms of positional encoding including trainable parametric, integral, and hierarchical variants (see section 3).

2.3 Datasets

NeRF models are trained per-scene. Although there are some NeRF models designed to be trained from sparse input views or unposed images, typical NeRF models requires relatively dense images with relatively varied poses. The COLMAP [2] library is often used to extract camera poses prior to training when necessary.

The original NeRF paper [1] presented a synthetic dataset created from Blender (referred to as Realistic Synthetic 360 ° in [1]). The virtual cameras have the same focal length and are placed at the same distance from the object. The dataset is composed of eight scenes with eight different objects. For six of these, viewpoints are sampled from the upper hemisphere, for the two others, viewpoints are sampled from the entire sphere. These objects are "hotdog", "materials", "ficus", "lego", "mic", "drums", "chair", "ship". The images are rendered at 800×800 pixels, with 100 views for training and 200 views for testing. The "lego" scene was often used for visualization in subsequent NeRF papers.

The LLFF [5] consists of 24 real-life scenes captured from handheld cellphone cameras. The views are forward facing towards the central object. Each scene consists of 20-30 images. The COLMAP package was used to compute the poses of the images.

The DTU dataset [26] is a multi-view stereo dataset captured using a 6-axis industrial robot mounted with both a camera and a structured light scanner. The robot provided precise camera positioning. Both the camera intrinsics and poses are carefully calibrated using the MATLAB calibration toolbox [27]. The light scanner provides reference dense point clouds which provide reference 3D geometry. Nonetheless, due to self-occlusion, the scans of certain areas in certain scenes are not complete. The original paper's dataset consists of 80 scenes each containing 49 views sampled on a sphere of radius 50 cm around the central object. For 21 of these scenes, an additional 15 camera positions are sampled at a radius of 65 cm, for a total of 64 views. The entire dataset consists of 44 additional scenes that have been rotated and scanned four times at 90 degree interval. The illumination of scenes is varied using 16 LEDs, with seven different lighting conditions. The image resolution is 1600 × 1200.

The ScanNet dataset [28] is a large-scale real-life RGB-D multi-modal dataset containing more than 2.5 million views of indoor scenes, with annotated camera poses, reference 3D surfaces, semantic labels, and CAD models. The depth frames are captured at 640 × 480 pixels, and the RGB images are captured at 1296 × 968 pixels. The scans were performed using RGB-D sensors attached to handheld devices such as iPhone/iPad. The poses were estimated from BundleFusion [29] and geometric alignment of resulting mesh. The

Tanks and Temples dataset [30] is a 3D reconstruction from video dataset. It consists of 14 scenes, including individual objects such as "Tank" and "Train", and large scale indoor scenes such as "Auditorium" and "Museum". Ground truth 3D data was captured using high quality industrial laser scanner. The ground truth point cloud was used to estimate camera poses using least squares optimization of correspondence points.

The ShapeNet dataset [31] is a simplistic large scale synthetic 3D dataset, consisting of 3D CAD model classified into 3135 classes. The most used are the 12 common object categories subset. This dataset is sometimes used when object-based semantic labels are an important part of a particular NeRF model. From ShapeNet CAD models, software such as Blender are often used to render training views with known poses.

2.4 Quality Assessment Metrics

Novel view synthesis via NeRF in the standard setting use visual quality assessment metrics for benchmarks. These metrics attempt to assess the quality of individual images either with (full-reference) or without (no-reference) ground truth images. Peak Signal to Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [32], Learned Perceptual Image Patch Similarity (LPIPS) [33] are by far the most commonly used in NeRF literature. 1

PSNR is a no-reference quality assessment metric, given by

$$PSNR(I) = 10 \cdot \log_{10}(\frac{MAX(I)^2}{MSE(I)}) \tag{10}$$

where $MAX(I)$ is the maximum possible pixel value in the image (255 for 8bit integer), and $MSE(I)$ is the pixel-wise mean squared error calculated over all color channels. PNSR is also commonly used in other fields of signal processing and is well understood.

SSIM [32] is a full reference quality assessment metric. For a single patch, this is given by

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{11}$$

where $C_i = (K_i L)^2$, L is the dynamic range of the pixels (255 for 8bit integer), and $K_1 = 0.01, K_2 = 0.03$ are constants chosen by the original authors. We note that there is a more general form of SSIM given by (12) in the original paper. The local statistics $\mu's, \sigma's$ are calculated within a 11×11 circular symmetric Gaussian weighted window, with weights w_i having a standard deviation of 1.5 and normalized to 1. These are given by, without loss of generalization,

$$\mu_x = \sum_i w_i x_i \tag{12}$$

$$\sigma_x = (\sum_i w_i (x_i - \mu_x)^2)^{1/2} \tag{13}$$

$$\sigma_{xy} = \sum_i w_i (x_i - \mu_x)(y_i - \mu_y) \tag{14}$$

where x_i, y_i are pixels sampled from the reference and assessed images respectively. The patch-wise SSIM scores are averaged over the entire image in practice.

LPIPS [33] is a full reference quality assessment metric which uses learned convolutional features. The score is given by a weighted pixel-wise MSE of feature maps over multiple layers.

$$LPIPS(x, y) = \sum_l^L \frac{1}{H_l W_l} \sum_{h,w}^{H_l, W_l} ||w_l \odot (x_{hw}^l - y_{hw}^l)||_2^2 \tag{15}$$

where x_{hw}^l, y_{hw}^l are the reference and assessed images' feature at pixel width w, pixel height h, and layer l. H_l and W_l are the feature maps height and width at the corresponding layer. The original LPIPS paper used SqueezeNet [34], VGG [35] and AlexNet [36] as feature extraction backbone. Five layers were used in the original paper. The original authors offered fine-tuned and from-scratch configurations, but in practice, the pretrained networks are used as is.

3 Neural Radiance Field (NeRF)

3.1 Fundamentals

Mip-NeRF [37] (March 2021) used cone tracing instead of ray tracing of standard NeRF [1] (March 2020) volume rendering. They achieved this by introducing the Integrated Positional Encoding. To generate an individual pixel, a cone was cast from the camera's center along the viewing direction, through the pixel's center. This cone was approximated by a multivariate Gaussian, whose mean vector and variance matrix were derived (as functions of ray origin and direction, $(\mathbf{o}, \mathbf{d})$ which now defined the cone's axis, see Appendix A in [37]) to have the appropriate geometry, resulting in the Integrated Positional Encoding (IPE). This is given by

$$\begin{aligned} &\gamma(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathbb{E}_{\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})}[\gamma(\mathbf{x})] \\ &= \begin{bmatrix} \sin(\boldsymbol{\mu}_{\boldsymbol{\gamma}}) \odot \exp(-(1/2)\mathrm{diag}(\boldsymbol{\Sigma}_{\boldsymbol{\gamma}})) \\ \cos(\boldsymbol{\mu}_{\boldsymbol{\gamma}}) \odot \exp(-(1/2)\mathrm{diag}(\boldsymbol{\Sigma}_{\boldsymbol{\gamma}})) \end{bmatrix} \end{aligned} \tag{16}$$

where $\boldsymbol{\mu}_{\boldsymbol{\gamma}}, \boldsymbol{\Sigma}_{\boldsymbol{\gamma}}$ are the means and variances of the multivariate Gaussian lifted onto the positional encoding basis with N levels. This process is given by

$$\boldsymbol{\mu}_{\boldsymbol{\gamma}} = \mathbf{P}\boldsymbol{\mu}\ , \quad \boldsymbol{\Sigma}\boldsymbol{\gamma} = \mathbf{P}\boldsymbol{\Sigma}\mathbf{P}^T \tag{17}$$

where

$$\mathbf{P} = \begin{bmatrix} 1\,0\,0\,2\,0\,0\,...\,2^{N-1}\,0\,0 \\ 0\,1\,0\,0\,2\,0\,...\,0\,2^{N-1}\,0 \\ 0\,0\,1\,0\,0\,2\,...\,0\,0\,2^{N-1} \end{bmatrix}. \tag{18}$$

The diagonal entries of the variance matrix can be directly calculated in practice (see (16) in [37]). The resulting mip-NeRF model was multi-scale in nature, and the conic IPE performed anti-aliasing. The model outperformed the baseline NeRF [1], significantly so at lower resolutions.

Ref-NeRF [38] (December 2021) was built on mip-NeRF, and was designed to better model reflective surfaces. Ref-NeRF parameterized NeRF radiance based on the reflection of the viewing direction about the local normal vector. They modified the density MLP into a directionless MLP which not only outputs density and the input feature vector of the directional MLP, but also diffuse color, specular tint, roughness and surface normal. The diffuse color and specular tint were multiplied together, and added to the specular color (output of directional MLP) which gave the

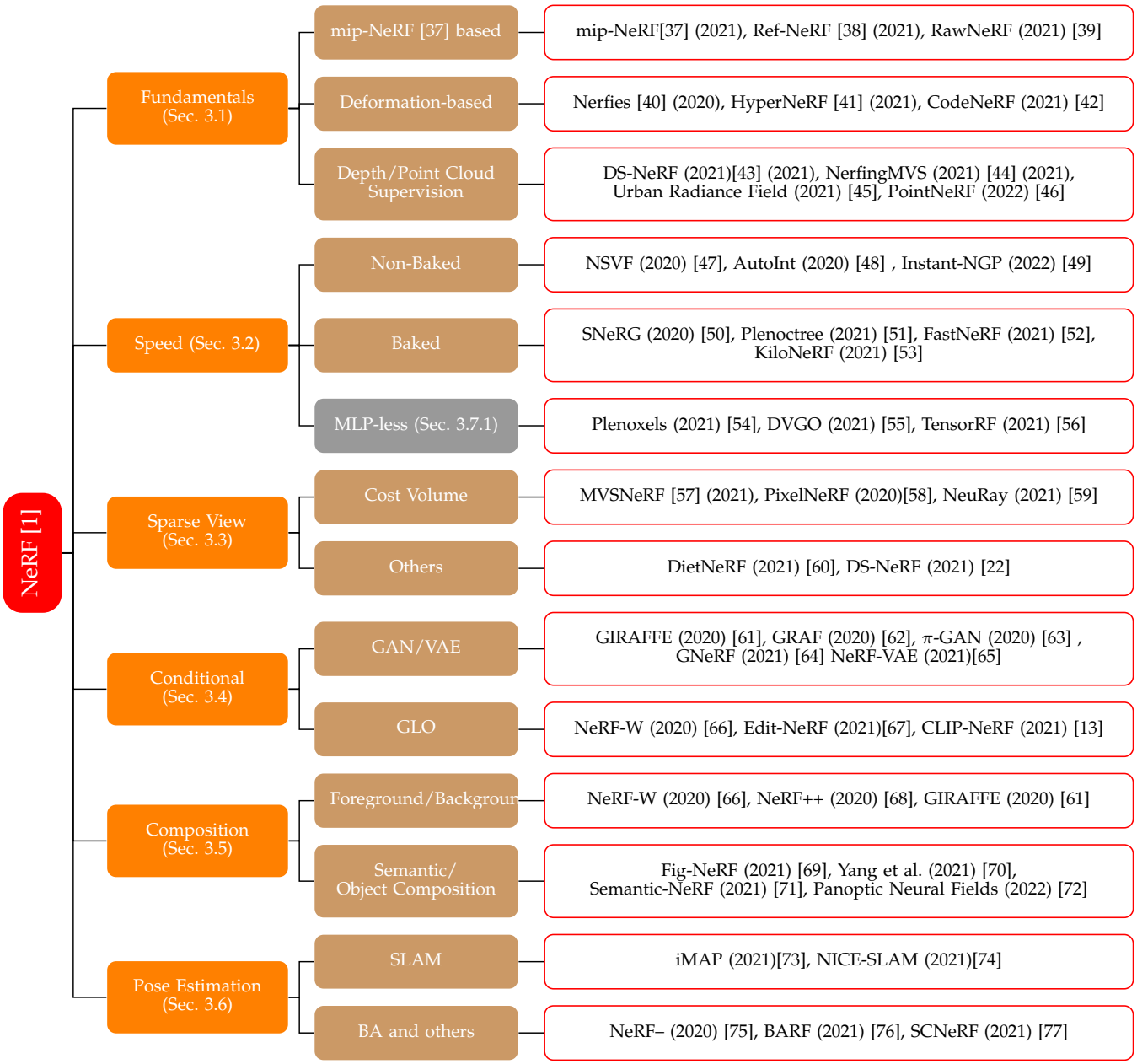

Fig. 2. Taxonomy of selected key NeRF innovation papers. The papers are selected using a combination of citations and GitHub star rating. We note that the MLP-less speed-based models are not strictly speaking NeRF models. Nonetheless, we decided to include them in this taxonomy tree due to their recent popularity and their similarity to speed based NeRF models.

final color. Additionally, they parameterized the directional vector using the spherical harmonics of vector sampled from a spherical Gaussian parameterized by the roughness. Ref-NeRF outperformed benchmarked methods, including mip-NeRF [37], NSVF [47], baseline NeRF [1], and non-NeRF models, on the Shiny Blender dataset (created by authors), the original NeRF dataset [1], and Real Captured Scenes from [79]. Ref-NeRF performed particularly well on reflective surfaces, and is able of accurately modelling specular reflections and highlights.

RegNeRF [21] (December 2021) aimed to solve the problem of NeRF training with sparse input view. Unlike most other methods which approached this task by using image features from pretrained networks as a prior conditioning for NeRF volume rendering, RegNeRF employed additional depth and color regularization. The model was tested on DTU [26] and LLFF [5] datasets, outperformed models such as PixelNeRF [58], SRF [80], MVSNeRF [57]. RegNeRF which did not require pretraining, achieved comparable performance to these models which were pre-trained on DTU and fine-tuned per scene. It outperformed Mip-NeRF and DietNeRF [60].

Ray Prior NeRF (RapNeRF) (May 2022) [81] explored a NeRF model better suited for view extrapolation, whereas standard NeRF models were better suited for interpolation. RapNeRF performed Random Ray Casting (RRC) whereby, given a training ray hitting a surface point $\mathbf{v} = \mathbf{o} + t_z\mathbf{d}$, a backward ray was cast starting from $\mathbf{v}$ towards a new origin $\mathbf{o}'$ using uniformly sampled perturbations in angles. RapNeRF also made use of a Ray Atlas (RA) by first extract-

TABLE 1
Comparison of select NeRF models on the synthetic NeRF dataset [1]

Method	Positional Encoding	Sampling Strategy	PSNR (dB)	SSIM	LIPIS	Training Iteration	Training Time	Inference Speed[1]
Baseline NeRF (2020)[1]	Fourier [1]	H [1][2]	31.01	0.947	0.081	100-300k	>12h	1
Speed Improvement								
JaxNeRF (2020)[78]	Fourier	H	31.65	0.952	0.051	250k	>12h	∼1.3
NSVF (2020) [47]	Fourier	Occupancy [47]	31.74	0.953	0.047	100-150k	-	∼10
SNeRG (2021) [50]	Fourier	Occupancy [50]	30.38	0.950	0.050	250k	>12h	∼9000
PlenOctree (2021) [51]	Fourier	H	31.71	0.958	0.053	2000k	>12h	∼3000
FastNeRF (2021) [52]	Fourier	H	29.97	0.941	0.053	300k	>12h	∼4000
KiloNeRF (2021) [53]	Fourier	Occupancy [53]	31.00	0.95	0.03	600k+150k+1000k	>12h	∼2000
Instant-NGP (2022) [49]	Hash [49]	Occupancy [49]	33.18	-	-	256k	∼5m	"orders of magnitude"
Quality Improvement								
mip-NeRF (2021)[37]	IPE [37]	H*(single MLP) [37]	33.09	0.961	0.043	1000k	∼3h	∼1
ref-NeRF (2021)[38]	IPE + IDE [38]	H*(single MLP)	35.96	0.967	0.058	250k	-	∼1
Sparse View/Few Shots								
MVSNeRF (2021)[57]	Fourier	Uniform	27.07	0.931	0.163	10k (*3 views)	∼15m*	∼1
DietNeRF (2021)[60]	Fourier	H	23.15	0.866	0.109	200k (*8 views)	-	∼1
DS-NeRF (2021)[22]	Fourier	H	24.9	0.72	0.34	150-200k (*10 views)	-	∼1

Speed-based and single-object based quality improvement models were selected for benchmark on the Synthetic NeRF dataset. For Positional Encoding and Sampling Strategy, unless indicated otherwise with a citation, identical entry denote that the particular strategy is from a previously indicated work.
[1]Inference speeds are given as speedup factor over the baseline NeRF.
[2]H and H* denote coarse to fine hierarchical sampling strategies.

ing a rough 3D mesh from a pretrained NeRF, and mapping training ray directions onto the 3D vertices. During training, a baseline NeRF was first trained to recover the rough 3D mesh, then RRC and RA are used to augment the training rays, with a predetermined probability. The authors evaluated their methods on the Synthetic NeRF dataset [1], and their own MobileObject dataset, showing that their RRC and RA augmentations can be adapted to other NeRF framework, and that it resulted in better view synthesis quality.

3.1.1 Deformation Fields

Park et al. introduced Nerfies [40] (November 2020), a NeRF model build using a deformation field which strongly improved the performance of their model in presence of non-rigid transformations in the scene (e.g., a dynamic scene). By introducing an additional MLP which mapped input observation frame coordinates to deformed canonical coordinates and by adding elastic regularization, background regularization, and coarse-to-fine deformation regularization by adaptive masking the positional encoding, they were able to accurately reconstruct certain non-static scenes which the baseline NeRF completely failed to do. An interesting application the authors found was the creation of multi-view "selfies" [2]. Concurrent to Nerfies was NerFace [82] (December 2020), which also used per-frame learned latent codes, and added facial expression as a 76-dimensional coefficient of a morphable model of constructed from Face2Face [83].

Park et al. introduced HyperNeRF [41] (June 2021), which built on Nerfies by extending the canonical space to a higher dimension, and adding an additional slicing MLP which describes how return to the 3D representation using ambient space coordinates. The canonical coordinate and ambient space coordinate were then used to condition the usual density and color MLPs of baseline NeRF models. HyperNeRF achieved great results in synthesizing views in scenes with topological changes with examples such as a human opening their mouth, or a banana being peeled.

CoNeRF [84] (December 2021) was built on HyperNeRF, but allowed for easily controllable photo editing via sliders, whose values are provided to a per-attribute Hypermap deformation field, parameterized by an MLP. This is done via sparse supervised annotation of slider values, and image patch masks, with a $L2$ loss term for slider attribute value, and a cross entropy loss for mask supervision. CodeNeRF achieved good results, using sliders to adjust facial expressions in their example dataset, which could have broad commercial applications for virtual human avatars.

3.1.2 Depth Supervision and Point Cloud Methods

By using supervising expected depth (6) with point clouds acquired from LiDAR or SfM, these models converge faster, converge to higher final quality, and require fewer training views than the baseline NeRF model. Many of these models were also built as few show/sparse view NeRF.

Deng et al. [22] (July 2021) used depth supervision from point clouds with a method named Depth-Supervised NeRF (DS-NeRF). In addition to color supervision via volume rendering and photometric loss, DS-NeRF also performs depth supervision using sparse point clouds extracted from the training images using COLMAP [2]. Depth is modelled as a normal distribution around the depth recorded by the sparse point cloud. A KL divergence term is added to minimize the divergence of the ray's distribution and this noisy depth distribution (See [22] for details). DS-NeRF was extensively tested on the DTU dataset [26], NeRF dataset [1], and the RedWood-3dscan dataset [85], outperforming benchmark methods such as baseline NeRF [1], pixelNerF [58] and MVSNeRF [57].

Concurrent to DS-NeRF is a work by Roessle et al. [43] (April 2021). In this work, the authors used COLMAP to extract a sparse point cloud, which was processed by a Depth Completion Network [86] to produce depth and uncertainty maps. In addition to the standard volumetric loss, the authors introduced a depth loss based on predicted

2. Popular self-portraits in social media

depth and uncertainty. The model was trained on RGB-D data from ScanNet [28] and Matterport3D [87] by introducing Gaussian noise to depth. The model outperformed DS-NeRF[22] marginally, and significantly outperformed baseline NeRF [1], and NerfingMVS [44].

NerfingMVS [44] (September 2021) used multi-view images in their NeRF model focused on depth reconstruction. In NerfingMVS, COLMAP was used to extract sparse depth priors in the form of a point cloud. This was then fed into a pretrained (fine-tuned on the scene) monocular depth network [88] to extract a depth map prior. This depth map prior was used to supervise volume sampling by only allowing sampling points at the appropriate depth. During volume rendering, the ray was divided into N equal bins, with the ray bounds clamped using the depth priors. The depth value D of a pixel was approximated by a modified version of (4). NerfingMVS outperformed previous methods on the ScanNet [28] dataset for depth estimation.

PointNeRF [46] (January 2022) used feature point clouds as an intermediate step to volume rendering. A Pretrained 3D CNN [89] was used to generate depth and surface probability γ from a cost volume created from training views, and produced a dense point cloud. A pretrained 2D CNN [35] was used to extract image features from training views. These were used to populate the point cloud features with image features, and probability γ_i of point p_i lying of a surface. Given the input position and view-direction, a PointNet[90]-like network was used to regress local density and color, which was then used for volume rendering. Using point cloud features also allowed the model to skip empty spaces, resulting in a speed-up of a factor of 3 over baseline NeRF. PointNeRF outperformed methods such as PixelNeRF [58], MVSNeRF [57], IBRNet [91] after per-scene optimization in the form of point growing and point pruning on the DTU dataset [26]. Point clouds acquired by other methods such as COLMAP can also be used in place of the 3D depth network-based point cloud, whereby per-scene optimization could be used to improve point cloud quality.

3.2 Improvements to Training and Inference Speed

In the original implementation by Mildenhall et al. [1], to improve computation efficiency, a hierarchical rendering was used. A naive rendering would require densely evaluating MLPs at all query points along each camera ray during the numerical integration (2). In their proposed method, they used two networks to represent the scene, one coarse and one fine. The output of the coarse network was used to pick sampling points for the fine network, which prevented dense sampling at a fine scale. In subsequent works, most attempts to improve NeRF training and inference speed can be broadly classified into the two following categories.

1) The first category trains, precomputes and stores NeRF MLP evaluation results into more easily accessible data structures. This only improves inference speed, albeit by a large factor. We refer to these models as baked models.
2) The second category are the non-baked models. These include multiple types of innovations. These models commonly (but not always) attempt to learn separate scene features from the learned MLPs' parameters, which in turn allows for smaller MLPs (e.g., learning and storing features in a voxel grid, which are then fed into MLPs which produce color and density) which can improve both training and inference speed at the cost of memory.

Other techniques such as ray termination (prevent further sampling points when accumulated transmittance approaches zero), empty space skipping, and/or hierarchical sampling (coarse+fine MLPs used in the original NeRF paper). These are also often used to further improve training and inference speed in conjunction.

A popular early re-implementation of the original NeRF in JAX [92], called JaxNeRF [78] (December 2020), was often used as benchmark comparison. This model was slightly faster and more suited for distributed computing than the original TensorFlow implementation.

In addition, a recent trend (CVPR 2022, 2022 preprints) introduced multiple NeRF adjacent methods which are based on category 2), using learned voxel/tree features. However, these methods skip over entirely the MLPs and performed volume rendering directly on the learned features. These are introduced in a later section (3.7.1) since they are, not strictly speaking, NeRF models.

3.2.1 Non-Baked

In Neural Sparse Voxel Fields (NSVF) (July 2020), Liu et al.[47] developed a voxel-based NERF model which models the scene as a set of radiance fields bounded by voxels. Feature representations were obtained by interpolating learnable features stored at voxel vertices, which were then processed by a shared MLP which computed σ and $\mathbf{c}$. NSVF used a sparse voxel intersection-based point sampling for rays, which was much more efficient than dense sampling, or the hierarchical two step approach of Mildenhall et al. [1]. However, this approach was more memory intensive due to storing feature vectors on a potentially dense voxel grid.

AutoInt (Dec 2020) [48] approximates the volume rendering step. By separating the discrete volume rendering equation 4 piecewise, then using their newly developed AutoInt, which trains the MLP Φ_θ by training its gradient (grad) networks Ψ^i_θ, which share internal parameters with, and are used to reassemble the integral network Φ_θ. This allowed for the rendering step to use much fewer samples, resulting in a ten times speed-up over baseline NeRF slight quality decrease.

Deterministic Integration for Volume Rendering (DIVeR) [93] (November 2021) took inspiration from NSVF [47], also jointly optimizing a feature voxel grid and a decoder MLP while performing sparsity regularization and voxel culling. However, they innovated on the volume rendering, using a technique unlike NeRF methods. DIVeR performed deterministic ray sampling on the voxel grid which produced an integrated feature for each ray interval (defined by the intersection of the ray with a particular voxel), which was decoded by an MLP to produce density and color of the ray interval. This essentially reversed the usual order between volume sampling and MLP evaluation. The method was evaluated on the NeRF Synthetic [1], BlendedMVS [94] and Tanks and Temple datasets [30], outperforming methods

such as PlenOctrees [51], FastNeRF [52] and KiloNeRF [53] in terms of quality, at comparable a rendering speed.

A recent innovation by Muller et al., dubbed Instant-Neural Graphics Primitives (Instant-NGP) [49] (January 2022) greatly improved NERF model training and inference speed. The authors proposed a learned parametric multi-resolution hash encoding that was trained simultaneously with the NERF model MLPs. They also employed advanced ray marching techniques including exponential stepping, empty space skipping, sample compaction. This new positional encoding and associated optimized implementation of MLP greatly improved training and inference speed, as well as scene reconstruction accuracy of the resulting NERF model. Within seconds of training, they achieved similar results to hours of training in previous NERF models.

3.2.2 Baked

A model by Hedman et al. [50] (July 2020) stored a precomputed NeRF on a sparse voxel. The method, called Sparse Neural Voxel Grid (SNeRG) stored precomputed diffused color, density, and feature vectors, which were stored on a sparse voxel grid in a process sometimes referred to as "Baking". During evaluation time, an MLP was used to produce specular color, which combined with the specular colors alpha composited along the ray, produced the final pixel color. The method was 3000 times faster than the original implementation, with speed comparable to PlenOctree.

The concurrent to SNeRG, PlenOctree [51] (March 2021) approach of Yu et al. achieved a inference time that was 3000 times faster than the original implementation. The authors trained a spherical harmonic NeRF (NeRF-SH), which instead of predicting the color function, predicted its spherical harmonic coefficients. The authors built an octree of precomputed spherical harmonic coefficients of the colors MLP. During the building of the octree, the scene was first voxelized, with low transmisivity voxels eliminated. This procedure could also be applied to standard NeRF (Non NeRF-SH models) by performing Monte Carlo estimations of the spherical harmonics components of the NeRF. PlenOctrees could be further optimized using the initial training images. This fine-tuning procedure was fast relative to the NeRF training.

In FastNeRF [52] (March 2021), Garbin et al. factorized color function $\mathbf{c}$ into the inner product of the output of the direction position dependent MLP (which also produces the density σ) and the output of a direction-dependent MLP. This allowed Fast-NeRF to easily cache color and density evaluation in a dense grid of the scene, which greatly improved inference time by a factor of 3000+. They also included hardware accelerated ray tracing [95] which skipped empty spaces, and stopped when the ray's transmittance was saturated.

Reiser et al. [53] (May 2021) improved on the baseline NeRF by introducing KiloNeRF, which separated the scene into thousands of cells, and trained independent MLPs for color and density predictions on each cell. These thousands of small MLPs were trained using knowledge distillation from a large pretrained teacher MLP, which we find closely related to "baking". They also employed early ray termination and empty space skipping. These two methods alone improved on the baseline NeRF's render time by a factor of 71. Separating the baseline NeRF's MLP into thousands of smaller MLP further improved render time by a factor of 36, resulting in a total of 2000-fold speed up in render time.

The Fourier Plenoctree [96] (February 2022) approach was proposed by Wang et al. in 2022. It was built for human silhouette rendering since it used the domain specific technique of Shape-From-Silhouette. The approach also takes inspiration from generalizable image conditioned NeRFs such as [57] and [58]. It first constructed a coarse visual hull using sparse views predicted from a generalization NeRF and Shape-From-Silhouette. Then colors and densities were densely sampled inside this hull and stored on a coarse Plenoctree. Dense views were sampled from the Plenoctree, with transmissivity thresholding used to eliminate most empty points. For the remaining points, new leaf densities and SH color coefficients were generated and the Plenoctree was updated. Then a Fourier Transform MLP was used to extract Fourier coefficients the density and SH color coefficients, which were fed into an inverse discrete Fourier transform to restore SH coefficients and density. According to the authors, using the frequency domain helped the model encode time-dependent information for dynamic scenes, such as the moving silhouettes modelled in the paper. The SH coefficients were then used to restore color. The Fourier Plenoctree can be fine-tuned on a per scene basis using the standard photometric loss (8).

A recent preprint (June 2022) created a proposed a lightweight method, MobileNeRF [97]. During training, MobileNeRF train a NeRF-like models based on a polygonal mesh with color, feature, and opacity MLPs attached to each mesh point. Alpha values were then discretized, and features were super-sampled for anti-aliasing. During rendering, the mesh with associated features and opacities are rasterized based on viewing position, and a small MLP is used to shade each pixel. The method was shown to be around 10 times faster than SNeRG [50].

EfficientNeRF [98] (July 2022) was based on PlenOctree [51], choosing to use spherical harmonics and to cache the trained scene in a tree. However, it made several improvements. Most importantly, EfficientNeRF improved the training speed by using momentum density voxel grid to store predicted density using exponential weighted average update. During the coarse sampling stage, the grid was used to discard sampling points with zero density. During the fine sampling stage, a pivot system was also used to speed up volume rendering. Pivot points were defined as points $\mathbf{x}_i$ for which $T_i\alpha_i > \epsilon$ where ϵ is a predefined threshold, and T_i and α_i are the transmittance and alpha values as defined in (4) and (5). During fine sample, only points near the pivot points are considered. These two improvements speed up the training time by a factor of 8 over the baseline NeRF [1]. The authors then cached the trained scene into a NeRF tree. This resulted in rendering speed comparable to FastNeRF [52], and exceeding that of baseline NeRF by thousands fold.

3.3 Few Shot/Sparse Training View NeRF

In pixelNeRF [58] (December 2020), Yu et al. used the pretrained layers of a Convolutional Neural Networks (and bilinear interpolation) to extract image features. Camera

rays used in NeRF were then projected onto the image plane and the image features were extracted for each query points. The features, view direction, and query points were then passed onto the NeRF network which produced density and color. General Radiance Field (GRF) [99] (Oct 2020) by Trevithick et al. took a similar approach, with the key difference being that GRF operated in canonical space as opposed to view-space for pixelNeRF.

MVSNeRF [57] (March 2021) used a slightly different approach. They also extracted 2D image features using a pretrained CNN. These 2D features were then mapped to a 3D voxelized cost volume using plane sweeping and a variance based cost. A pretrained 3D CNN was used to extract a 3D neural encoding volume which was used to generate per-point latent codes using interpolation. When performing point sampling for volume rendering, the NeRF MLP then generated point density and color using as input these latent features, point coordinate and viewing direction. The training procedure involves the joint optimization of the 3D feature volume and the NeRF MLP. When evaluating on the DTU dataset, within 15 minutes of training, MVSNeRF could achieve similar results to hours of baseline NeRF training.

DietNeRF [60] (June 2021) introduced the semantic consistency loss L_{sc} based on image features extracted from Clip-ViT [100], in addition to the standard photometric loss.

$$L_{sc} = \frac{\lambda}{2}||\phi(I) - \phi(\hat{I})||_2^2 \tag{19}$$

where ϕ performs the Clip-ViT feature extraction on training image I and rendered image $\hat{I}$. This reduced to a cosine similarity loss for normalized feature vectors (eq. 5 in [60]). DietNeRF was benchmarked on a subsampled NeRF synthetic dataset [1], and DTU dataset [26]. The best performing method for single-view novel synthesis was a pixelNeRF [58] model fine-tuned using the semantic consistency loss of DietNeRF.

The Neural Rays (NeuRay) approach, by Liu et al. [59] (July 2021) also used a cost volume approach. From all input views, the authors estimated cost volumes (or depth maps) using multi-view stereo algorithms. From these, a CNN is used to created feature maps G. During volume rendering, from these features, both visibility and local features are extracted and processed using MLPs to extract color and alpha. The visibility is computed as a cumulative density function written as a weighted sum sigmoid functions Φ

$$v(z) = 1 - t(z), \text{ where } t(z) = \sum_{i=1}^{N} w_i \Phi((z - \mu_i)/\sigma_i) \tag{20}$$

where w_i, μ_i, σ_i are decoded from G using an MLP. NeuRay also used an alpha based sampling strategy, by computing a hitting probability, and only sampling around points with a high hitting probability (see Sec. 3.6 in [59] for details). Like other NeRF models conditioned on extracted image features from pre-trained neural networks, NeuRay generalizes well to new scenes, and can be further fine-tuned to exceed the performance of the baseline NeRF model. NeuRay outperformed MVSNeRF on the NeRF synthetic dataset after fine-tuning both models for 30 minutes.

GeoNeRF[101] (November 2021) extracted 2D image features from every view using a pretrained Feature Pyramid Network. This method then constructed a cascading 3D cost-volume using plane sweeping. From these two feature representations, for each of the N query point along a ray, one view independent, and multiple view dependent feature tokens were extracted. These were refined using a Transformer [102]. Then, the N view-independent tokens are refined through an AutoEncoder, which returned the N densities along the ray. The N sets of view dependent tokens were each fed into an MLP which extracted color. These networks can all be pretrained and generalized well to new scenes as shown by the authors. Moreover, they can be fine tuned per-scene achieving great results on the DTU [26], NeRF synthetic [1], and LLF Forward Facing [5] datasets, outperforming methods such as pixelNeRF [58] and MVSNeRF [57].

Concurrent to GeoNeRF is LOLNeRF (November 2021) [103], which is capable of single-shot view synthesis of human faces, and is built similarly to π-GAN [63], but uses Generative Latent Optimization [104] instead of adversarial training [105].

NeRFusion (March 2022) also extracted a 3D cost volume from 2D image features extracted from CNN, which was then processed by a sparse 3D CNN into a local feature volume. However, this method performs this step for each frame, and then used a GRU [106] to fuse these local feature volumes into a global feature volume, which were used to condition density and color MLPs. NeRFusion outperformed IBRNet, baseline NeRF [1], NeRFingMVS[44], MVSNeRF [57] on ScanNet [28], DTU [26], and NeRF Synthetic [1] datasets.

AutoRF [107] (April 2022) focused on novel view synthesis of objects without background. Given 2D multi-view images, a 3D object detection algorithm was used with panoptic segmentation to extract 3D bounding boxes and object masks. The bounding boxes were used to define Normalized Object Coordinate Spaces, which were used for per-object volume rendering. An encoder CNN was used to extract appearance and shape codes which were used in the same way as in GRAF [62]. In addition to the standard photometric loss (8), an additional occupancy loss was defined as

$$L_{occ} = \frac{1}{|W_{occ}|} \sum_{u \in W_{occ}} \log(Y_u(1/2 - \alpha) + 1/2) \tag{21}$$

where Y is the object mask, and W_{occ} is either the set of foreground pixels, or background pixels. During test-time, the shape codes, appearance code, and bounding boxes were further refined using the same loss function. The method outperformed pixelNeRF [58] and CodeNeRF [42] for object-based novel view synthesis on the nuScene [108], SRN-Cars [6] and Kitti [109] datasets.

SinNeRF [24] attempted NeRF scene reconstruction from single images by integrating multiple techniques. They used image warping and known camera intrinsic and poses to create reference depth for depth-supervision of unseen views. They used adversarial training with a CNN discriminator to provide patch-wise texture supervision. They also use a pretrained ViT to extract global image features from reference patch and unseen patch, comparing them with an

L2 loss term a global structure prior. SinNeRF outperformed DS-NeRF [22], PixelNeRF [58], and DietNeRF [58] on the NeRF synthetic dataset [1], the DTU dataset[26] and the LLFF dataset [5].

Methods such as [22] and [43] from section 3.1.2 approach the sparse view problem by using point clouds for depth supervision.

3.4 (Latent) Conditional NeRF

Latent conditioning of NeRF models refers to using latent vector(s)/code(s) to control various aspects of NeRF view synthesis. These latent vectors can be input at various points along the pipeline to control scene composition, shape, and appearance. They allow for an addition set of parameters to control for aspects of the scene which changes image-to-image, while allowing for other parts to account for the permanent aspects of the scene, such as scene geometry. A fairly simple way to train image generators conditioned on latent code is to use Variational Auto-Encoder (VAE) [110] methods. These models use an Encoder, which turns images into latent codes following a particular probability distribution defined by the user, and a Decoder which turns sampled latent codes back into images. These methods are not as often used in NeRF models compared to the two following methods, as such, we do not introduce a separate subsection for VAE.

In NeRF-VAE [65] (January 2021), Kosiorek et al. proposed a generative NeRF model which generalized well to out-of-distribution scenes, and removed the need to train on each scene from scratch. The NeRF renderer in NeRF-VAE was conditioned on latent code which was trained using Iterative Amortized Inference [111][112] and a ResNet [113] based encoder. The authors also introduced an attention-based scene function (as opposed to the typical MLP). NeRF-VAE consistently outperformed the baseline NeRF with low number (5-20) of scene views, but due to lower scene expressivity, was outperformed by baseline NeRF when large number of views were available (100+).

Adversarial training is often used for generative and/or latent conditioned NeRF models. First developed in 2014, Generative Adversarial Networks (GANs) [105] are generative models which employ a generator G which synthesizes images from "latent code/noise", and a discriminator D which classifies images as "synthesized" or "real". The generator seeks to "trick" the discriminator, and make its images indistinguishable from "real" training images. The discriminator seeks to maximize its classification accuracy. These two networks are trained adversarially, which is the optimization of the following minimax loss/value function,

$$\min_G \max_D \mathbb{E}_{x\sim data}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z))] \tag{22}$$

where the generator generates images based on latent code z sampled from some distribution $p(z)$, which the discriminator compares to training image x. In GAN-based NeRF model, the generator G encompasses all novel-view synthesis steps, and is transitionally thought of as the NeRF model. The generator in this case also requires an input pose, in addition to a latent code. The discriminator D is usually an image classification CNN.

Another form of latent conditioning used by NeRF models is Generative Latent Optimization (GLO) [104]. In GLO, a set of randomly sampled latent codes $\{\mathbf{z_1}, ..., \mathbf{z_n}\}$, usually normally distributed, is paired to a set of images $\{I_1, ..., I_n\}$. These latent codes are input to a generator G whose parameters are jointly optimized with the latent code using some reconstruction loss L such as L_2. I.e., the optimization is formulated as

$$\min_{G,\mathbf{z_i},...,\mathbf{z_n}} \sum_{i=i}^{n} L(G(\mathbf{z_i}, \mathbf{u_i}), I_i) \tag{23}$$

where $\mathbf{u_i}$ represent the other inputs not optimized over (needed in NeRF but not necessarily for other models). According to the GLO authors, this method can be thought of as a Discriminator-less GAN.

3.4.1 Adversarially Trained Models

GRAF [62] (July 2020) was the first latent conditioned NeRF model trained adversarially. It paved way for many later works. A NeRF based generator was conditioned on latent appearance code $\mathbf{z_a}$ and shape code $\mathbf{z_s}$, and is given by

$$G(\gamma(\mathbf{x}), \gamma(\mathbf{d}), \mathbf{z_s}, \mathbf{z_a}) \rightarrow (\sigma, \mathbf{c}). \tag{24}$$

In practice, the shape code, conditioning scene density, was concatenated with the embedded position, which was input to the direction independent MLP. The appearance code, conditioning scene radiance, was concatenated with the embedded viewing direction, which was input to the direction dependent MLP. As per baseline NeRF, images were generated via volume sampling. These were then compared using a discriminator CNN for adversarial training.

Within three months of GRAF, Chan et al. developed π-GAN [63] (December 2020) which also used a GAN approach to train a conditional NeRF model. The generator was a SIREN-based [114] NeRF volumetric renderer, with sinusoidal activation replacing the standard ReLU activation in the density and color MLPs. π-GAN outperformed GRAF[62] and HOLOGAN on standard GAN datasets such as Celeb-A[115], CARLA [116] and Cats [117].

Pix2NeRF [118] (February 2022) was proposed as an adversarial trained NeRF model which could generate NeRF rendered images given randomly sampled latent codes and poses. Built from π-GAN [63], It is composed of a NeRF based generator $G : d, z \rightarrow I$, a CNN discriminator $D : I \rightarrow d, l$ and an Encoder $E : I \rightarrow d, z$. z is a latent code sampled from a multi-variate distribution, d is a pose, I is a generated image, and l is a binary label for real vs. synthesized image. In addition to the π-GAN loss, from which the adversarial architecture is based, the pix2NeRF loss function also include the following: 1) a reconstruction loss comparing $z_{predicted}$ and $z_{sampled}$ to ensure consistency of latent space, 2) a reconstruction loss ensuring image reconstruction quality, between I_{real} and $I_{reconstructed}$, where $I_{reconstructed}$ is created by the Generator from a z_{pred}, d_{pred} pair produced by the Encoder 3) a conditional adversarial objective which prevents mode collapse towards trivial poses (see original paper for the exact expressions). The model achieved good results on the CelebA dataset [115], the CARLA dataset [116], and the ShapeNet subclasses from SRN [119], but is outperformed by its backbone π-GAN for conditional image synthesis.

3.4.2 Jointly Optimized Latent Models

Edit-NeRF [67] (June 2021) allowed for scene editing using image conditioning from user input. Edit-NeRF's shape representation was composed of a category specific shared shape network F_{share}, and a instance specific shape network F_{inst}. F_{inst} was conditioned on $\mathbf{z_s}$ whereas F_{shared} was not. In theory, the F_{shared} behaved as a deformation field, not unlike [40]. The NeRF editing was formulated as a joint optimization problem of both the NeRF network parameters and the latent codes $\mathbf{z_s}, \mathbf{z_a}$, using GLO. They then conducted NeRF photometric loss optimization on latent codes, then on the MLP weights, and finally optimized both latent codes and weights jointly.

Innovating on Edit-NeRF, CLIP-NeRF's [13] (December 2021) neural radiance field was based on the standard latent conditioned NeRF, i.e., NeRF models conditioned on shape and appearance latent codes. However, by using Contrastive Language Image Pre-training (CLIP), CLIP-NeRF could extract from user input text or images, the induced latent space displacements by using shape and appearance mapper networks. These displacements could then be used to modify the scene's NeRF representation based on these input text or images. This step allowed for skipping the per-edit latent code optimization used in Edit-NeRF, resulting in a speed-up of a factor of $\sim$8-60, depending on the task. They also used a deformation network similar to deformable NeRFs [40] (called *instance-specific-network*) in Edit-NeRF [67] to help with modifying the scene based on latent space displacement. The model was trained in two stages. In the first, CLIP-NeRF's conditional latent NeRF models was trained adversarially. In the second, the CLIP shape and appearance mappers were trained using self-supervision. When applying CLIP-NeRF to images with unknown pose and latent codes, the authors used an Expectation-Maximization algorithm which then allowed them to use CLIP-NeRF latent code editing. CLIP-NeRF outperformed EDIT-NeRF in terms of inference speed, and post-edit metrics, especially for global edits, which was weakness of Edit-NeRF.

3.5 Unbound Scene and Scene Composition

In NeRF in the Wild (NeRF-W) [66] (August 2020), Martin-Brualla et al. addressed two key issues of baseline NeRF models. Real-life photographs of the same scene can contain per-image appearance variations due to lighting conditions, as well as transient objects which are different in each image. The density MLP was kept fixed for all images in a scene. However, NeRF-W conditioned their color MLP on a per-image appearance embedding. Moreover, another MLP conditioned on per-image transient embedding predicted the color and density functions of transient objects. These latent embeddings were constructed using Generative Latent Optimization. NeRF-W did not improve on NeRF in terms of rendering speed, but achieved much higher results in the crowded Phototourism dataset [120].

Zhang et al. developed the NeRF++ [68] (October 2020) model, which was adapted to generate novel views for unbound scenes, by separating the scene using a sphere. The inside of the sphere contained all foreground object and all fictitious cameras views, whereas the background was outside the sphere. The outside of the sphere was then reparametrized using as an inverted sphere space. Two separate NeRF models were trained, one for the inside the sphere and one for the outside. The camera ray integral was also evaluated in two parts. Using this approach, they outperformed the baseline NeRF on Tanks and Temples [30] scenes as well as scenes from Yucer et al. [121].

GIRAFFE [61] (November 2020) also was built with a similar approach to NeRF-W, using generative latent codes and separating background and foreground MLP for scene composition. GIRAFFE was based on GRAF. It assigned to each object in the scene an MLP, which produced a scalar density and a deep feature vector (replacing color). These MLPs (with shared architecture and weights) took as input shape and appearance latent vectors, as well as an input pose. The background was treated as all other objects, except with its own MLP and weights. The scene was then composed using a density weighted sum of features. A small 2D feature map was then created from this 3D volume feature field using volume rendering, which was fed into an upsampling CNN to produce an image. GIRAFFE performed adversarial training using this synthesized image, and a 2D CNN discriminator. The resulting model had a disentangled latent space, allowing for fine control over the scene generation.

Fig-NeRF [69] (April 2021) also took on scene composition, but focused on object interpolation and amodal segmentation. They used two separate NeRF models, one for the foreground, one for the background. Their foreground model was the deformable Nerfies model [40]. Their background model was an appearance latent code conditioned NeRF. They used two photometric losses, one for the foreground, one for the background. Fig-NeRF achieved good results for amodal segmentation and object interpolation, on datasets such as ShapeNet [31] Gelato [122], and Objectron [123].

Yang et al. [70] (September 2021) created composition model which can edit objects within the scene. They used a voxel-based approach [47], creating a voxel grid of features which is jointly optimized with MLP parameters. They used two different NeRFs, one for objects, one for the scene, both of which were conditioned on interpolated voxel features. The object NeRF was further conditioned on a set of object activation latent codes. Their method was trained and evaluated on ScanNet[28] as well as an inhouse ToyDesk dataset with instance segmentation labels. They incorporated segmentation labels with a mask loss term given by

$$w(\mathbf{r})_k ||\hat{O}(\mathbf{r})_k - M(\mathbf{r})||_2^2 \tag{25}$$

where $\hat{O}(\mathbf{r})_k = \sum_{i=i}^{N} T_i \alpha_i$ is the 2D object opacity, $M(\mathbf{r})$ is the kth object mask of 0s and 1s, and $w(\mathbf{r})_k$ is the balance weight between 0s and 1s in the mask. The authors edited objects within the scene by first obtaining the background colors and densities from the scene NeRF branch, pruning sample points at the object's location. Then the object's colors and densities are obtained from the object NeRF, and transformed according to user defined manipulations. Finally, all colors and densities are aggregated according to the discrete volume rendering equation (4). The authors's method outperformed baseline NeRF [1] as well as NSVF [47] on both their inhouse dataset and ScanNet.

NeRFReN [23] (November 2021) addressed the problem of reflective surfaces in NeRF view synthesis. The authors separated the radiance field into two components, transmitted $(\sigma^t, \mathbf{c}^t)$ and reflected $(\sigma^r, \mathbf{c}^r)$, with the final pixel value given by

$$I = I_t + \beta I_r \tag{26}$$

where β is the reflection fraction given by the geometry of the transmitted radiance field as

$$\beta = \sum_i T_{\sigma_i^t}(1 - \exp(-\sigma_i^t \delta_i))\alpha_i. \tag{27}$$

$T_{\sigma_i^t}$ is given by (3), and α_i by (5). In addition to the standard photometric loss, the authors used a depth smoothness L_d (eq. 8 in [23]) loss to encourage the transmitted radiance field to have the correct geometry, and likewise, a bidirectional depth consistency loss L_{bdc} (eq. 10 in [23]) for the reflected radiance field. NeRFReN was able to render reflective surface on the authors' RFFR dataset, outperforming benchmark methods such as baseline NeRF [1], and NerfingMVS [44], as well as ablation models. The method was shown to support scene editing via reflection removal and reflection substitution.

Panoptic Neural Field [72] separates the scene into foreground and objects. The background is represented by an MLP which outputs color, density, and semantic label. Each object's color and density are represented by their own MLP with a dynamic bounding box, foregoing the traditional approach of using a shared MLP with object specific latent codes. The method is capable of a wide variety of computer vision tasks such as 2D panoptic segmentation, 2D depth prediction, scene editing, as well as the standard view synthesis and 3D reconstruction.

3.6 Pose Estimation

iNeRF [124] (December 2020) formulated pose reconstruction as an inverse problem. Given a pre-trained NeRF, using the photo-metric loss 8, the Yen-Chen et al. optimized the pose instead of the network parameters. The authors used an interest-point detector, and performed interest region-based sampling. The authors also performed semi-supervision experiments, where they used iNeRF pose estimation on unposed training images to augment the NeRF training set, and further train the forward NeRF. This semi-supervision was shown by the author to reduce the requirement of posed photos from the forward NeRF by 25 %.

NeRF– [75] (February 2021) however jointly estimated NeRF model parameters and camera parameters. This allowed for the model to construct radiance fields and synthesize novel views only images, in an end-to-end manner. NeRF– overall achieved comparable results to using COLMAP with the 2020 NeRF model in terms of both view synthesis. However, due to limitations with pose initialization, NeRF– was most suited for front facing scenes, and struggled with rotational motion and object tracking movements.

Concurrent to NeRF– was the Bundle-Adjusted Neural Radiance Field (BARF) [76] (April 2021), which also jointly estimated poses alongside the training of the neural radiance field. BARF also used a coarse-to-fine registration by adaptively masking the positional encoding, similar to the technique used in Nerfies [40]. Overall, BARF results exceeded those of NeRF– on the LLFF forward facing scenes dataset with unknown camera poses by 1.49 PNSR averaged over the eight scenes, and outperformed COLMAP registered baseline NeRF by 0.45 PNSR. Both BARF and NeRF– used naive dense ray sampling for simplicity.

Jeong et al. introduced a self-calibrating joint optimization model for NeRF (SCNeRF) [77] (August 2021). Their camera calibration model can not only optimize unknown poses, but also camera intrinsic for non-linear camera models such as fish-eye lens models. By using curriculum learning, they gradually introduce the nonlinear camera/noise parameters to the joint optimization. This camera optimization model was also modular and could be easily used with different NeRF models. The method outperformed BARF [76] on LLFF scenes [5].

Sucar et al. introduced the first NeRF-based dense online SLAM model named iMAP [73] (March 2021). The model jointly optimizes camera pose and the implicit scene representation in the form of a NeRF model, making use of continual online learning. They used an iterative two-step approach of tracking (pose optimization with respect to NeRF) and mapping (bundle-adjustment joint optimization of pose and NeRF model parameters). iMAP achieved a pose tracking speed close to camera framerate by running the much faster tracking step in parallel to the mapping process. iMAP also used keyframe selection by optimizing the scene on a sparse and incrementally selected set of images.

GNeRF, a different type of approach by Meng et al [64] (March 2021), used pose as generative latent code. GNeRF first obtains coarse camera poses and radiance field with adversarial training. This is done by using a generator which takes a randomly sampled pose, and synthesized a view using NeRF-style rending. Then a discriminator compared the rendered view with the training image. An inversion network then took the generated image, and output a pose, which was compared to the sampled pose. This resulted in a coarse image-pose pairing. The images and poses were then jointly refined via a photometric loss in a hybrid optimization scheme. GNeRF was slightly outperformed by COLMAP based NeRF on the Synthetic-NeRF dataset, and outperformed COLMAP based NeRF on the DTU dataset.

Building on iMAP, NICE-SLAM [74] (December 2021) improved on various aspects such as keyframe selection and NeRF architecture. Specifically, they used a hierarchical grid based representation of the scene geometry, which was able to fill in gaps iMAP reconstruction in large scale unobserved scene features (walls, floors etc.) for certain scenes. NICE-SLAM achieved lower pose estimation errors and better scene reconstruction results than iMAP. The NICE-SLAM also only used $\sim 1/4$ of the FLOPs of iMAP, $\sim 1/3$ the tracking time and $\sim 1/2$ the mapping time.

3.7 Adjacent Methods

3.7.1 Fast MLP-less Volume Rendering

Plenoxel [54] (December 2021) followed in Plenoctree's footsteps in voxelizing the scene and storing a scalar for density and spherical harmonics coefficients direction dependent

color. However, surprisingly, Plenoxel skipped the MLP training entirely, and instead fit these features directly on the voxel grid. They also obtained comparable results to NeRF ++ and JaxNeRF, with faster training times by a factor of a few hundreds. These results showed that the primary contribution of NeRF models is the volumetric rendering of new view given densities and colors, and not the density and color MLPs themselves.

A concurrent paper by Sun et al. [55] (November 2021) also explored this topic. The authors also directly optimized a voxel grid of scalars for density. However, instead of using spherical harmonic coefficient, the authors use 12 and 24 dimensional features, and a small shallow decoding MLP. The authors used a sampling strategy analogous to the coarse-fine sampling of the original NeRF paper by training a coarse voxel grid first, and then a fine voxel grid based on the geometry of the coarse grid. The model was named Direct Voxel Grid Optimization (DVGO), which outperformed the baseline NeRF (1-2 days) of training with only 15 minutes of training on the Synthetic-NeRF dataset. The authors obtained a PSNR of 31.95 at voxel resolution 160^3 after 11 minutes of training, and a PSNR of 32.80 at voxel resolution of 256^3 after 22 minutes of trainign on a 2080Ti. They outperformed Plenoxel's 512^3 resolution model's 31.71 PSNR after 11 minutes of training on an RTX Titan.

Along the lines of MLP-less radiance field, TensorRF [56] (March 2022) also used neural-network-free volume rendering. TensorRF stored a scalar density and a vector color feature (SH harmonic coefficient, or color feature to be input into a small MLP) in a 3D voxel grid, which were then represented as a rank 3 tensor $T_\sigma \in R^{H\times W\times D}$ and a rank 4 tensor $T_c \in R^{H\times W\times D\times C}$, where H, W, D are the height, width and depth resolution of the voxel grid, and C is channel dimension. Then the authors used two factorization schemes: CANDECOMP-PARAFAC (CP) which factorized the tensors as pure vector outer products and Vector Matrix (VM) which factorized the tensors as vector/matrix outer products. These factorizations decreased the memory requirement from Plenoxels by a factor of 200 when using CP. Their VM factorization performed better in terms of visual quality, albeit at a memory tradeoff. The training speed was comparable to Pleoxels and much faster than the benchmarked NeRF models.

3.7.2 Others

IBRNet [91] (February 2021) was published in 2021 as a NeRF adjacent method for view synthesis that is widely used in benchmarks. For a target view, IBRNet selected N views from the training set who's viewing directions are most similar. A CNN was used to extract features from these images. For a single query point, for each of the i input view, the known camera matrix was used to project onto the corresponding image to extract color $\mathbf{c}_i$ and feature $\mathbf{f}_i$. An MLP was then used to refine these features $\mathbf{f}_i'$ to be multi-view aware and produce pooling weights w_i. For density prediction these features were summed using the weights. This is done for each query point, and the results (of all query points along the ray) were concatenated together and fed into a ray Transformer [102] which predicted the density. For color prediction, the $\mathbf{f}_i$'s and w_i's were fed into an MLP alongside relative viewing direction $\Delta\mathbf{d}_i$ to produce a color blending weight. The final color was simply the per-image color $\mathbf{c}_i$ summed using the blending weights. IBRNet in general outperformed baseline NeRF [1], and used $1/6$ the number of FLOPs in the experimental setup of the paper.

Compared to NeRF models, Scene Rendering Transformer (SRT) [125] (November 2021) took a different approach to volume rendering. They used a CNN to extract feature patches from scene images which were then fed into Encoder-Decoder Transformers [102], which along with camera ray and viewpoint coordinates $\{\mathbf{o}, \mathbf{d}\}$, which then produced the output color. The entire ray was queried at once, unlike with NeRF models. The SRT is geometry-free. and did not produce the scene's density function, nor relied on geometric inductive biases. The NeRFormer [126] (September 2021) is a comparable concurrent model which also uses Transformers as part of the volume rendering process. The paper also introduced the Common Objects in 3D dataset, which could gain popularity in the near future.

3.8 Applications

3.8.1 Urban

Urban Radiance Fields [45] (November 2021) aimed at applying NeRF based view synthesis and 3D reconstruction for urban environment using sparse multi-view images supplemented by LiDAR data. In addition to the standard photometric loss, they also use a LiDAR based depth loss L_{depth} and sight loss L_{sight}, as well as a skybox based segmentation loss L_{seg}. These are given by

$$L_{depth} = \mathbb{E}[(z - \hat{z}^2)], \tag{28}$$

$$L_{sight} = \mathbb{E}[\int_{t_1}^{t_2} (w(t) - \delta(z))^2 dt]. \tag{29}$$

$$L_{seg} = \mathbb{E}[S_i(\mathbf{r} \int_{t_1}^{t_2} (w(t) - \delta(z))^2 dt]. \tag{30}$$

$w(t)$ is defined as $T(t)\sigma(t)$ as defined in eq(3). z and $\hat{z}$ are the LiDAR measure depth and estimated depth (6), respectively. $\delta(z)$ is the Dirac delta function. $S_i(\mathbf{r}) = 1$ if the ray goes through a sky pixel in the ith image, where sky pixels are segmented through a pretrained model [145], 0 otherwise. The depth loss forces the estimated depth $\hat{z}$ to match the LiDAR acquired depth. The sight loss forces the radiance to be concentrated at the surface of the measured depth. The segmentation loss forces point samples along rays through to the sky pixels to have zero density. 3D reconstruction was performed by extracting point clouds form the NeRF model during volumetric rendering. A ray was cast for each pixel in the virtual camera. Then, the estimated depth was used to place the point cloud in the 3D scene. Poisson Surface Reconstruction was used to construct 3D mesh from this generated point cloud. The authors used Google Street View data on which the Urban Radiance Field outperformed NeRF [1], NeRF-W [66], mip-NeRF [37], and DS-NeRF[22] for both view synthesis and 3D reconstruction.

Mega-NeRF [128] (December 2021) performed large scale urban reconstruction from aerial drone images. Mega-NeRF used a NeRF++[68] inverse sphere parameterization to separate foreground from background. However, the authors

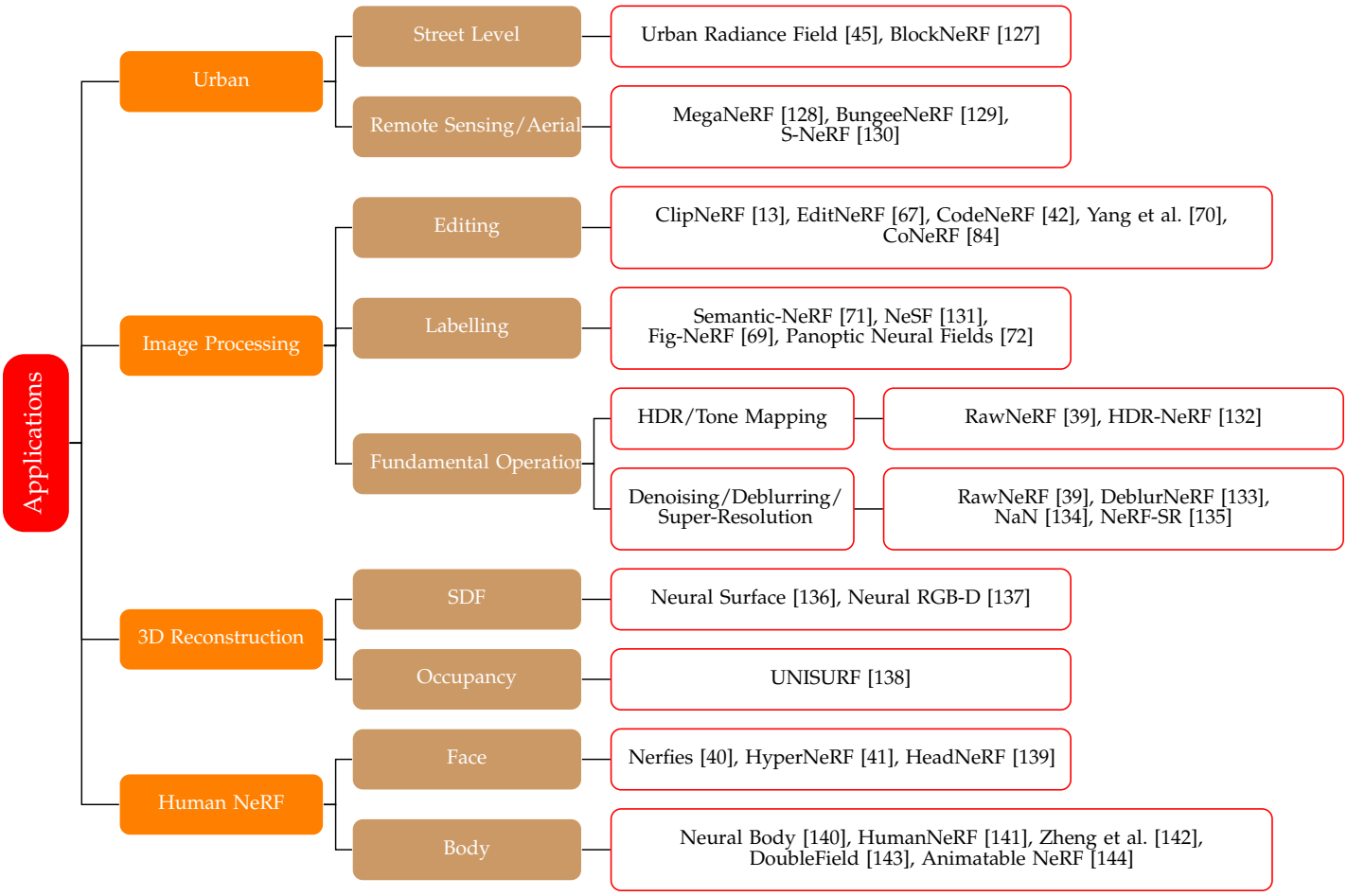

Fig. 3. Application of NeRF models. Papers are selected based on application as well as citation numbers and GitHub star rating.

extended the method by using an ellipsoid which better fit the aerial point of view. They incorporated the per-image appearance embedding code of NeRF-W [66] into their model as well. They partitioned the large urban scenes into cells each one represented by its own NeRF module, and train each module on only the images with potentially relevant pixels. For rendering, the method also cached a coarse rendering of densities and colors into an octree. During rendering for dynamic fly-through, a coarse initial view was quickly produced, and dynamically refined via additional rounds of model sampling. The model outperformed benchmarked baseline such as NeRF++[68], COLMAP based MVS reconstruction [2], and produced impressive fly-through videos.

Block-NeRFs [127] (February 2022) performed city-scale NeRF based reconstruction from 2.8 million street-level images. Such large scale outdoor dataset posed problems such as transient appearance and objects. Each Individual Block-NeRF was built on mip-NeRF [37] by using its IPE and NeRF-W[66] by using its appearance latent code optimization. Moreover, the authors used semantic segmentation to mask out transient objects such as pedestrians and cars during NeRF training. A visibility MLP was trained in parallel, supervised using the transitivity function (3) and the density value generated by the NeRF MLP. These were used to discard low visibility Block-NeRFs. Neighbourhoods were divided into blocks, on which a Block-NeRF was trained. These blocks were assigned with overlap, and images were sampled from overlapping Block-NeRFs and composited using inverse distance weighting after an appearance code matching optimization.

Other methods published in first class conferences such as S-NeRF [130] (April 2021), BungeeNeRF [129] (December 2021), also deal with urban 3D reconstruction and view synthesis, albeit from remote sensing images.

3.8.2 *Human Body*

Neural Body applied NeRF volume rendering to rendering humans with moving poses from videos. The authors first used the input video to anchor a vertex based deformable human body model (SMPL [146]). Onto each vertex, the authors attached a 16-dimensional latent code $\mathbf{Z}$. Human pose parameters $\mathbf{S}$ (initially estimated from video during training, can be input during inference) were then used to deform the human body model. This model was voxelized in a grid and then processed by a 3D CNN, which extracted 128-dimensional latent code (feature) at each occupied voxel. Any 3D point $\mathbf{x}$ was first transformed to SMPL coordinate system, then a 128-dimensional latent code/feature ψ was extract via interpolation. This was passed to the density MLP. The color MLP took in addition

the positional encoding of 3D coordinate $\gamma_x(\mathbf{x})$ and viewing direction $\gamma_d(\mathbf{d})$ and appearance latent code $\mathbf{l_t}$ (accounting for per frame difference in the video).

$$\sigma(\mathbf{x}) = M_\sigma(\psi(\mathbf{x}|\mathbf{Z}, \mathbf{S})). \tag{31}$$

$$c(\mathbf{x}) = M_c(\psi(\mathbf{x}|\mathbf{Z}, \mathbf{S}), \gamma_x(\mathbf{x}), \gamma_d(\mathbf{d}), \mathbf{l_t}). \tag{32}$$

Standard NeRF approaches struggled with the moving bodies, whereas the mesh deformation approach of Neural Body was able to interpolate between frames and between poses. State of the art models from top tier conferences in 2021/2022 such as Animatable NeRF [144] (May 2021), DoubleField [143] (Jun 2021), HumanNeRF [141] (Jan 2022), Zheng et al. [142] (Mar 2022) also innovated on this topic.

3.8.3 Image Processing

Mildenhall et al. created RawNeRF [39] (Nov 2021), adapting Mip-NeRF [37], to High Dynamic Range (HDR) image view synthesis and denoising. RawNeRF renders in a linear color space using raw linear images as training data. This allows for varying exposure and tone-mapping curves, essentially applying the post-processing after NeRF rendering instead of directly using post-processed images as training data. It is trained using a relative MSE loss for noisy HDR path tracing from Noise2Noise [147], given by

$$L = \sum_{i=1}^{N} \left(\frac{\hat{y_i} - y_i}{sg(\hat{y_i}) + \epsilon}\right)^2 \tag{33}$$

where $sg(.)$ indicates a gradient-stop (treating its argument as a constant with zero gradient). RawNeRF is supervised with variable exposure images, with the NeRF models' "exposure" scaled by the training image's shutter speed, as well as a per-channel learned scaling factor. It achieved impressive results in night-time and low light scene rendering and denoising. RawNeRF is particularly suited for scenes with low lighting. On the topic of NeRF based denoising, NaN [134] (April 10) also explored this emerging research area.

Concurrent to RawNeRF, HDR-NeRF [132] (Nov 2021) from Xin et al. also worked on HDR view synthesis. However, HDR-NeRF approached HDR view synthesis by using Low Dynamic Range training images with variable exposure time as opposed to the raw linear images in RawNeRF. RawNeRF modelled a HDR radiance $\mathbf{e}(\mathbf{r}) \in [0, \infty)$ which replaced the standard $\mathbf{c}(\mathbf{r})$ in (1). This radiance was mapped to a color $\mathbf{c}$ using three MLP camera response functions (one for each color channel) f. These represent the typical camera manufacturer dependent linear and non-linear post-processing. HDR-NeRF strongly outperformed baseline NeRF and NeRF-W [66] on Low Dynamic Range (LDR) reconstruction, and achieved high visual assessment scores on HDR reconstruction.

Other methods such as DeblurNeRF [133] (November 2021), NeRF-SR [135] (December 2021), NaN (April 2022) [134]. Focus on fundamental image processing tasks such as denoising, deblurring and super-resolution, allowing for high quality view synthesis from low quality training images.

Semantic-NeRF [71] (March 2021) was a NeRF model capable of synthesizing semantic labels for novel views. This was done using an additional direction independent MLP (branch) which took as input position and density MLP features, and produced point-wise semantic label $\mathbf{s}$. The semantic labels were also generated via volume rendering via

$$S(\mathbf{r}) = \sum_{i}^{N} T_i \alpha_i \mathbf{s}_i. \tag{34}$$

The semantic labels were supervised via a categorical cross entropy loss. The method was able to train with sparse semantic label (10% labelled) training data, as well as recover semantic label from pixel-wise noise and region/instance-wise noise. The method also achieved good label-super resolution results, and label propagation (from sparse point-wise annotation), and can be used for multi-view semantic fusion, outperforming non-deep learning methods. NeSF [131] (November 2021). The previously introduced Fig-NeRF [69] also approached this issue.

3.8.4 Surface Reconstruction

The scene geometry of NeRF model is implicit and hidden inside the neural networks. However, for certain applications, more explicit representations, such as 3D mesh are desired. For the baseline NeRF, it is possible to extract a rough geometry by evaluating and thresholding the density MLP. The methods introduced in this subsection used innovative scene representation strategies, changing the fundamental behaviour of the density MLP.

UNISERF [138] (April 2021) reconstructed scene surfaces by replacing the alpha value a_i at the i-th sample point used in the discretized volume rendering equation, given by (5), with a discrete occupancy function $o(\mathbf{x}) = 1$ in occupied space, and $o(\mathbf{x}) = 0$ in free space. This occupancy function was also computed by an MLP, and essentially replaced the volume density. Surfaces were then retrieved via root-finding along rays. UNISURF outperformed benchmark methods including using density threshold in baseline NeRF models, as well as IDR [148]. The occupancy MLP can be used to define an explicit surface geometry for the scene. A recent workshop by Tesla [149] showed that autonomous driving module's 3D understanding is driven by one such NeRF-like occupancy network.

Signed distance functions (SDF) give the signed distance of a 3D point to the surface(s) they define (i.e., negative distance if inside an object, positive distance if outside). They are often used in computer graphics to define surfaces which are the zero set of the function. SDF can be used for surface reconstruction via root-finding, and can be used to define entire scene geometries.

The Neural Surface (NeuS) [136] (June 2021) model performed volume rendering like the baseline NeRF model. However it used signed distance functions to define scene geometries. It replaces the density outputting portion of the MLP with an MLP which outputs the signed distance function value. The density $\rho(t)$ which replaced $\sigma(t)$ in the volume rendering equation (2) was then constructed as

$$\rho(t) = \max\left(\frac{-\frac{d\Phi}{dt}(f(\mathbf{r}(t)))}{\Phi(f(\mathbf{r}(t)))}, 0\right) \tag{35}$$

where $\Phi(\cdot)$ is the sigmoid function, and its derivative $\frac{d\Phi}{dt}$ is the logistic density distribution. The authors have

shown their model to outperform baseline NeRF, and have shown both theoretical and experimental justification for their method and their implementation of SDF based scene density. This method was concurrent to UNISERF and outperformed it on the DTU dataset [26]. Like with UNISURF, performing root finding on the SDF can be used to define an explicit surface geometry for the scene.

A concurrent work by Azinovic et al. [137] (April 2021) also replaced the density MLP with a truncated SDF MLP. They instead computed their pixel color as weighted sum of sampled colors

$$C(\mathbf{r}) = \frac{\sum_{i=1}^{N} w_i \mathbf{c}_i}{\sum_{i=1}^{N} w_i} \tag{36}$$

where w_i is given by a product of sigmoid function given by

$$w_i = \Phi(\frac{D_i}{tr}) \cdot \Phi(-\frac{D_i}{tr}) \tag{37}$$

where tr is the truncation distance, which cuts off any SDF value too far from individual surfaces. To account for possible multiple ray-surface intersection, subsequent truncation regions weighted to zero, and do not contribute to the pixel color. The authors also use a per-frame appearance latent code from NeRF-W [66] to account for to white-balance and exposure changes. The authors reconstructed the triangular mesh of the scene by using Marching Cubes [150] on their truncated SDF MLP, and achieved clean reconstruction results on ScanNet[28] and a private synthetic dataset (but is not directly comparable to UNISERF and NeuS since DTU results were not provided).

4 Discussion

4.1 Concerning Speed

The baseline NeRF models had both slow training and inference speed. Currently, speed based NeRF/NeRF-adjacent models use three main paradigms. They either 1) are baked (by evaluating an already trained NeRF model, and caching/baking its results), or 2) separate learned scene features from the color and density MLPs by using additional learned voxel/spatial-tree features, or 3) perform volume rendering directly on learned voxel features without use of MLPs. Additional speed up can be achieved by using more advanced methods during volume rendering such as empty space skipping and early ray termination. However, method 1) does not improve training time by its design. Methods 2) and 3) require additional memory since voxel/spatial-tree based scene features have larger memory footprint compared to small NeRF MLPs. Currently, Instant-NGP [49] shows the most promise by making use of a multi-resolution hashed positional encoding as additional learned features, the model could represent scenes accurately with tiny and efficient MLPs. This allowed extremely fast training and inference. The Instant-NGP model also finds applications in image compression and neural SDF scene representation. Alternatively, for method 3), factorized tensor [56] representation for the learned voxel grid reduced the memory requirement of the learned voxel features by two orders of magnitude and is also a viable research area. We expect future speed-based methods to follow methods 2) and 3) for which a highly impactful research direction would be to improve the data structure and design of these additional learned scene features.

4.2 Concerning Quality

For quality improvement, we found NeRF-W's [66] implementation of per-image transient latent code and appearance code to be influential. A similar idea was also found on the concurrent GRAF [62]. These latent codes allowed for the NeRF model to control per-image lighting/coloration change, as well as small changes in scene content. On NeRF fundamentals, we found mip-NeRF [37] to be most influential, as the cone tracing IPE was unlike any previous NeRF implementation. Ref-NeRF, built upon mip-NeRF, then further improved view-dependent appearance. Ref-NeRF is an excellent modern baseline for quality based NeRF research. Specific to image processing, innovations from RawNeRF [39] and DeblurNeRF [133] can be combined with Ref-NeRF as foundation to build extremely high quality denoising/deblurring NeRF models.

4.3 Concerning Pose Estimation and Sparse View

Given the current state of NeRF research, we believe non-SLAM pose estimation is a solved problem. The SfM using the COLMAP[2] package is used by most NeRF dataset to provide approximate poses, which is sufficient for most NeRF research. BA can also be used to jointly optimize NeRF models and poses during training. NeRF based SLAM is a relatively under-explored area of research. iMAP [73] and Nice-SLAM [74] offer excellent NeRF based SLAM frameworks which could integrate faster and better quality NeRF models.

Sparse View/few shot NeRF use 2D/3D feature extraction from multi-view images using pretrained CNN. Some also use point cloud from SfM for additional supervision. We believe many of these models already achieved the goal of few shot (2-10 views). Further small improvements can be achieved by using more advanced feature extraction backbones. We believe a key area of research would be combining sparse views methods and fast methods to achieve real-time NeRF models deployable to mobile devices.

4.4 Concerning Applications

We believe the immediate applications of NeRF are novel view synthesis and 3D reconstruction of Urban environment, and of human avatars. Further improvements can be made by facilitating the extraction of 3D mesh, point cloud or SDF from density MLPs and integrating faster NeRF models. Urban environment specifically require the division of the environment into separate small scenes, each to be represented by a small scene specific NeRF model. The baking or learning of separate scene features for speed based NeRF models for city scale models is an interesting research direction. For human avatars, the integration of a model which can separate view-specific lighting such as Ref-NeRF[38] would be highly beneficial to applications such as virtual reality and 3D graphics. NeRF is also finding applications in fundamental image processing tasks such as denoising, deblurring, upsampling, compression, and image

editing, and we expect more innovations in these areas in the near future as more computer vision practitioners adopt NeRF models.

5 Conclusion

Since the original paper by Mildenhall et al., NeRF models have made tremendous progress in speed, quality, and training view requirements, improving on all the weaknesses of the original model. NeRF models have found applications in urban mapping/modelling/photogrammetry, image editing/labelling, image processing, and 3D reconstruction and view synthesis of human avatars and urban environments. Both the technical improvements and the applications were discussed in detail in this survey, during the completion of which, we have noticed an ever-growing interest in NeRF models, and an ever-growing number of preprints and publications.

NeRF is an exciting new paradigm for novel view synthesis, 3D reconstruction, and neural rendering. By providing this survey, we hope to introduce more Computer Vision practitioners to this field, to provide a helpful reference of existing NeRF models, and to motivate future research with our discussions. We are excited to see future technical innovations and applications of Neural Radiance Fields.

References

[1] 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 *European conference on computer vision*. Springer, 2020, pp. 405–421.

[2] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[3] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 31–42.

[4] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 43–54.

[5] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–14, 2019.

[6] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[7] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: Learning dynamic renderable volumes from images," *arXiv preprint arXiv:1906.07751*, 2019.

[8] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3504–3515.

[9] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, "Local deep implicit functions for 3d shape," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4857–4866.

[10] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.

[11] F. Dellaert and L. Yen-Chen, "Neural volume rendering: Nerf and beyond," *arXiv preprint arXiv:2101.05204*, 2020.

[12] F. Zhan, Y. Yu, R. Wu, J. Zhang, and S. Lu, "Multimodal image synthesis and editing: A survey," *arXiv preprint arXiv:2112.13592*, 2021.

[13] C. Wang, M. Chai, M. He, D. Chen, and J. Liao, "Clip-nerf: Text-and-image driven manipulation of neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3835–3844.

[14] Y. Guo, K. Chen, S. Liang, Y.-J. Liu, H. Bao, and J. Zhang, "Ad-nerf: Audio driven neural radiance fields for talking head synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5784–5794.

[15] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole, "Zero-shot text-guided object generation with dream fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 867–876.

[16] K. Jo, G. Shim, S. Jung, S. Yang, and J. Choo, "Cg-nerf: Conditional generative neural radiance fields," *arXiv preprint arXiv:2112.03517*, 2021.

[17] J. Sun, X. Wang, Y. Shi, L. Wang, J. Wang, and Y. Liu, "Ide-3d: Interactive disentangled editing for high-resolution 3d-aware portrait synthesis," *arXiv preprint arXiv:2205.15517*, 2022.

[18] Y. Chen, Q. Wu, C. Zheng, T.-J. Cham, and J. Cai, "Sem2nerf: Converting single-view semantic masks to neural radiance fields," *arXiv preprint arXiv:2203.10821*, 2022.

[19] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi *et al.*, "Advances in neural rendering," in *Computer Graphics Forum*, vol. 41, no. 2. Wiley Online Library, 2022, pp. 703–735.

[20] J. T. Kajiya and B. P. Von Herzen, "Ray tracing volume densities," *ACM SIGGRAPH computer graphics*, vol. 18, no. 3, pp. 165–174, 1984.

[21] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. Sajjadi, A. Geiger, and N. Radwan, "Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5480–5490.

[22] 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*, 2022, pp. 12 882–12 891.

[23] Y.-C. Guo, D. Kang, L. Bao, Y. He, and S.-H. Zhang, "Nerfren: Neural radiance fields with reflections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 409–18 418.

[24] D. Xu, Y. Jiang, P. Wang, Z. Fan, H. Shi, and Z. Wang, "Sinnerf: Training neural radiance fields on complex scenes from a single image," 2022.

[25] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020.

[26] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, "Large scale multi-view stereopsis evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 406–413.

[27] J.-Y. Bouguet, *Camera Calibration Toolbox for Matlab*. CaltechDATA, May 2022. [Online]. Available: https://data.caltech.edu/records/20164

[28] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5828–5839.

[29] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration," *ACM Transactions on Graphics 2017 (TOG)*, 2017.

[30] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–13, 2017.

[31] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[32] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[33] 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*, 2018, pp. 586–595.

[34] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[37] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864.

[38] D. Verbin, P. Hedman, B. Mildenhall, T. Zickler, J. T. Barron, and P. P. Srinivasan, "Ref-nerf: Structured view-dependent appearance for neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5491–5500.

[39] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron, "Nerf in the dark: High dynamic range view synthesis from noisy raw images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 190–16 199.

[40] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Nerfies: Deformable neural radiance fields," *ICCV*, 2021.

[41] K. Park, U. Sinha, P. Hedman, J. T. Barron, S. Bouaziz, D. B. Goldman, R. Martin-Brualla, and S. M. Seitz, "Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields," *ACM Trans. Graph.*, vol. 40, no. 6, dec 2021.

[42] W. Jang and L. Agapito, "Codenerf: Disentangled neural radiance fields for object categories," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 949–12 958.

[43] 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. 12 892–12 901.

[44] Y. Wei, S. Liu, Y. Rao, W. Zhao, J. Lu, and J. Zhou, "Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5610–5619.

[45] K. Rematas, A. Liu, P. P. Srinivasan, J. T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari, "Urban radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 932–12 942.

[46] 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.

[47] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *NeurIPS*, 2020.

[48] D. B. Lindell, J. N. Martel, and G. Wetzstein, "Autoint: Automatic integration for fast neural volume rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 556–14 565.

[49] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127

[50] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking neural radiance fields for real-time view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5875–5884.

[51] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "PlenOctrees for real-time rendering of neural radiance fields," in *ICCV*, 2021.

[52] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin, "Fastnerf: High-fidelity neural rendering at 200fps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 346–14 355.

[53] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 335–14 345.

[54] A. Yu, S. Fridovich-Keil, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," *arXiv preprint arXiv:2112.05131*, 2021.

[55] C. Sun, M. Sun, and H.-T. Chen, "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5459–5469.

[56] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," *arXiv preprint arXiv:2203.09517*, 2022.

[57] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 124–14 133.

[58] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural radiance fields from one or few images," in *CVPR*, 2021.

[59] Y. Liu, S. Peng, L. Liu, Q. Wang, P. Wang, C. Theobalt, X. Zhou, and W. Wang, "Neural rays for occlusion-aware image-based rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7824–7833.

[60] A. Jain, M. Tancik, and P. Abbeel, "Putting nerf on a diet: Semantically consistent few-shot view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5885–5894.

[61] M. Niemeyer and A. Geiger, "Giraffe: Representing scenes as compositional generative neural feature fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 453–11 464.

[62] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "Graf: Generative radiance fields for 3d-aware image synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 154–20 166, 2020.

[63] 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.

[64] Q. Meng, A. Chen, H. Luo, M. Wu, H. Su, L. Xu, X. He, and J. Yu, "Gnerf: Gan-based neural radiance field without posed camera," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6351–6361.

[65] A. R. Kosiorek, H. Strathmann, D. Zoran, P. Moreno, R. Schneider, S. Mokrá, and D. J. Rezende, "Nerf-vae: A geometry aware 3d scene generative model," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5742–5752.

[66] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," in *CVPR*, 2021.

[67] S. Liu, X. Zhang, Z. Zhang, R. Zhang, J.-Y. Zhu, and B. Russell, "Editing conditional radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5773–5783.

[68] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "Nerf++: Analyzing and improving neural radiance fields," *arXiv:2010.07492*, 2020.

[69] C. Xie, K. Park, R. Martin-Brualla, and M. Brown, "Fig-nerf: Figure-ground neural radiance fields for 3d object category modelling," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 962–971.

[70] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui, "Learning object-compositional neural radiance field for editable scene rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 779–13 788.

[71] S. Zhi, T. Laidlow, S. Leutenegger, and A. J. Davison, "In-place scene labelling and understanding with implicit scene representation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 838–15 847.

[72] A. Kundu, K. Genova, X. Yin, A. Fathi, C. Pantofaru, L. J. Guibas, A. Tagliasacchi, F. Dellaert, and T. Funkhouser, "Panoptic neural fields: A semantic object-aware neural scene representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 871–12 881.

[73] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.

[74] 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*, 2022, pp. 12 786–12 796.

[75] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "NeRF−−: Neural radiance fields without known camera parameters," *arXiv preprint arXiv:2102.07064*, 2021.

[76] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "Barf: Bundle-adjusting neural radiance fields," in *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[77] Y. Jeong, S. Ahn, C. Choy, A. Anandkumar, M. Cho, and J. Park, "Self-calibrating neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5846–5854.

[78] B. Deng, J. T. Barron, and P. P. Srinivasan, "JaxNeRF: an efficient JAX implementation of NeRF," 2020. [Online]. Available: https://github.com/google-research/google-research/tree/master/jaxnerf

[79] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking neural radiance fields for real-time view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5875–5884.

[80] J. Chibane, A. Bansal, V. Lazova, and G. Pons-Moll, "Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7911–7920.

[81] J. Zhang, Y. Zhang, H. Fu, X. Zhou, B. Cai, J. Huang, R. Jia, B. Zhao, and X. Tang, "Ray priors through reprojection: Improving neural radiance fields for novel view extrapolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 376–18 386.

[82] G. Gafni, J. Thies, M. Zollhofer, and M. Nießner, "Dynamic neural radiance fields for monocular 4d facial avatar reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8649–8658.

[83] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2387–2395.

[84] K. Kania, K. M. Yi, M. Kowalski, T. Trzciński, and A. Tagliasacchi, "Conerf: Controllable neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 623–18 632.

[85] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun, "A large dataset of object scans," *arXiv preprint arXiv:1602.02481*, 2016.

[86] X. Cheng, P. Wang, and R. Yang, "Learning depth with convolutional spatial propagation network," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 10, pp. 2361–2379, 2019.

[87] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *arXiv preprint arXiv:1709.06158*, 2017.

[88] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman, "Learning the depths of moving people by watching frozen people," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4521–4530.

[89] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, "Mvsnet: Depth inference for unstructured multi-view stereo," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 767–783.

[90] 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.

[91] Q. Wang, Z. Wang, K. Genova, P. P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, "Ibrnet: Learning multi-view image-based rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4690–4699.

[92] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[93] L. Wu, J. Y. Lee, A. Bhattad, Y.-X. Wang, and D. Forsyth, "Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 200–16 209.

[94] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan, "Blendedmvs: A large-scale dataset for generalized multi-view stereo networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1790–1799.

[95] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison *et al.*, "Optix: a general purpose ray tracing engine," *Acm transactions on graphics (tog)*, vol. 29, no. 4, pp. 1–13, 2010.

[96] L. Wang, J. Zhang, X. Liu, F. Zhao, Y. Zhang, Y. Zhang, M. Wu, J. Yu, and L. Xu, "Fourier plenoctrees for dynamic radiance field rendering in real-time," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 524–13 534.

[97] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi, "Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures," *arXiv preprint arXiv:2208.00277*, 2022.

[98] T. Hu, S. Liu, Y. Chen, T. Shen, and J. Jia, "Efficientnerf efficient neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 902–12 911.

[99] A. Trevithick and B. Yang, "Grf: Learning a general radiance field for 3d representation and rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 182–15 192.

[100] 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.

[101] M. M. Johari, Y. Lepoittevin, and F. Fleuret, "Geonerf: Generalizing nerf with geometry priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 365–18 375.

[102] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[103] D. Rebain, M. Matthews, K. M. Yi, D. Lagun, and A. Tagliasacchi, "Lolnerf: Learn from one look," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1558–1567.

[104] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam, "Optimizing the latent space of generative networks," *arXiv preprint arXiv:1707.05776*, 2017.

[105] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[106] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[107] N. Müller, A. Simonelli, L. Porzi, S. R. Bulò, M. Nießner, and P. Kontschieder, "Autorf: Learning 3d object radiance fields from single view observations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3971–3980.

[108] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[109] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[110] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[111] Y. Kim, S. Wiseman, A. Miller, D. Sontag, and A. Rush, "Semi-amortized variational autoencoders," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2678–2687.

[112] J. Marino, Y. Yue, and S. Mandt, "Iterative amortized inference," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3403–3412.

[113] 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.

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

[115] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *European conference on computer vision*. Springer, 2016, pp. 87–102.

[116] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[117] W. Zhang, J. Sun, and X. Tang, "Cat head detection-how to effectively exploit shape and texture features," in *European conference on computer vision*. Springer, 2008, pp. 802–816.

[118] S. Cai, A. Obukhov, D. Dai, and L. Van Gool, "Pix2nerf: Unsupervised conditional p-gan for single image to neural radiance fields translation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3981–3990.

[119] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[120] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, and E. Trulls, "Image matching across wide baselines: From paper to practice," *International Journal of Computer Vision*, 2020.

[121] K. Yücer, A. Sorkine-Hornung, O. Wang, and O. Sorkine-Hornung, "Efficient 3d object segmentation from densely sampled light fields with applications to 3d reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 3, pp. 1–15, 2016.

[122] R. Martin-Brualla, R. Pandey, S. Bouaziz, M. Brown, and D. B. Goldman, "Gelato: Generative latent textured objects," in *European Conference on Computer Vision*. Springer, 2020, pp. 242–258.

[123] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. Grundmann, "Objectron: A large scale dataset of object-centric videos in the wild with pose annotations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7822–7831.

[124] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "inerf: Inverting neural radiance fields for pose estimation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1323–1330.

[125] M. S. Sajjadi, H. Meyer, E. Pot, U. Bergmann, K. Greff, N. Radwan, S. Vora, M. Lučić, D. Duckworth, A. Dosovitskiy *et al.*, "Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6229–6238.

[126] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 901–10 911.

[127] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-nerf: Scalable large scene neural view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8248–8258.

[128] H. Turki, D. Ramanan, and M. Satyanarayanan, "Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 922–12 931.

[129] Y. Xiangli, L. Xu, X. Pan, N. Zhao, A. Rao, C. Theobalt, B. Dai, and D. Lin, "Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering," in *The European Conference on Computer Vision (ECCV)*, 2022.

[130] D. Derksen and D. Izzo, "Shadow neural radiance fields for multi-view satellite photogrammetry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1152–1161.

[131] S. Vora*, N. Radwan*, K. Greff, H. Meyer, K. Genova, M. S. M. Sajjadi, E. Pot, A. Tagliasacchi, and D. Duckworth, "Neural semantic fields for generalizable semantic segmentation of 3d scenes," *Transactions on Machine Learning Research*, 2022, https://openreview.net/forum?id=ggPhsYCsm9.

[132] X. Huang, Q. Zhang, Y. Feng, H. Li, X. Wang, and Q. Wang, "Hdr-nerf: High dynamic range neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 398–18 408.

[133] L. Ma, X. Li, J. Liao, Q. Zhang, X. Wang, J. Wang, and P. V. Sander, "Deblur-nerf: Neural radiance fields from blurry images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 861–12 870.

[134] N. Pearl, T. Treibitz, and S. Korman, "Nan: Noise-aware nerfs for burst-denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 672–12 681.

[135] C. Wang, X. Wu, Y.-C. Guo, S.-H. Zhang, Y.-W. Tai, and S.-M. Hu, "Nerf-sr: High quality neural radiance fields using supersampling," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 6445–6454.

[136] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 171–27 183, 2021.

[137] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, "Neural rgb-d surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6290–6301.

[138] M. Oechsle, S. Peng, and A. Geiger, "Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5589–5599.

[139] Y. Hong, B. Peng, H. Xiao, L. Liu, and J. Zhang, "Headnerf: A real-time nerf-based parametric head model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 20 374–20 384.

[140] S. Peng, Y. Zhang, Y. Xu, Q. Wang, Q. Shuai, H. Bao, and X. Zhou, "Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9054–9063.

[141] F. Zhao, W. Yang, J. Zhang, P. Lin, Y. Zhang, J. Yu, and L. Xu, "Humannerf: Efficiently generated human radiance field from sparse inputs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7743–7753.

[142] Z. Zheng, H. Huang, T. Yu, H. Zhang, Y. Guo, and Y. Liu, "Structured local radiance fields for human avatar modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 893–15 903.

[143] R. Shao, H. Zhang, H. Zhang, M. Chen, Y.-P. Cao, T. Yu, and Y. Liu, "Doublefield: Bridging the neural surface and radiance fields for high-fidelity human reconstruction and rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 15 872–15 882.

[144] S. Peng, J. Dong, Q. Wang, S. Zhang, Q. Shuai, X. Zhou, and H. Bao, "Animatable neural radiance fields for modeling dynamic human bodies," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 14 314–14 323.

[145] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[146] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, pp. 248:1–248:16, Oct. 2015.

[147] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *arXiv preprint arXiv:1803.04189*, 2018.

[148] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman, "Multiview neural surface reconstruction by disentangling geometry and appearance," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2492–2502, 2020.

[149] A. Elluswamy. Tesla, workshop on autonomous driving. CVPR 2022. [Online]. Available: https://www.youtube.com/watch?v=jPCV4GKX9Dw

[150] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.