

Neural Radiance Fields: Past, Present, and Future

Ansh Mittal¹[0000-0001-6648-6064]

¹ University of Southern California, Los Angeles CA 90007, USA

<https://www.usc.edu/>

² {anshm}@usc.edu

Abstract. The various aspects like modeling and interpreting 3D environments and surroundings have enticed humans to progress their research in 3D Computer Vision, Computer Graphics, and Machine Learning. An attempt made by Mildenhall et al in their paper about NeRFs (Neural Radiance Fields) led to a boom in Computer Graphics, Robotics, Computer Vision, and the possible scope of High-Resolution Low-Storage Augmented Reality and Virtual Reality-based 3D models have gained traction from res with more than 500 preprints related to NeRFs published. This paper serves as a bridge for people starting to study these fields by building on the basics of Mathematics, Geometry, Computer Vision, and Computer Graphics to the difficulties encountered in Implicit Representations at the intersection of all these disciplines. This survey provides the history of rendering, Implicit Learning, and NeRFs, the progression of research on NeRFs, and the potential applications and implications of NeRFs in today’s world. In doing so, this survey categorizes all the NeRF-related research in terms of the datasets used, objective functions, applications solved, and evaluation criteria for these applications.

Keywords: NeRFs (Neural Radiance Fields) · Rendering · Computer Vision · Deep Learning · Computer Graphics · Robotics · Augmented Reality · Virtual Reality · Implicit Machine Learning

1 Introduction

Recently, implicit representation functions and machine learning have pervaded through different applications and problem statements unsolvable in traditional machine learning. And applications related to View Synthesis, 3D Rendering, and Modeling have also started to leverage Radiance Fields (also known as NeRFs). The preprints papers in this domain have been rising exponentially with applications in different disciplines like 3D Computer Vision, Astronomy, Climate change, and many other areas of study that require images/3D geometry of locations. Although the work in Radiance Fields started as a means to offer better View Synthesis, after the development in the literature and GPU technology, it has permeated other fields such as Image Reconstruction, Super

Resolution, Pose Estimation, 3D Aware Image Synthesis, Depth Estimation, Image Generation, 3D Reconstruction, Neural Rendering, and other computer vision related problems. So, this survey aims to comprehensively categorize the models, extensions, and applications of NeRFs in terms of datasets, objective functions used, and problem statements tackled, evaluation measures adopted while discussing the techniques and key concepts associated with the field of NeRFs (Neural Radiance Fields) in the research literature. Despite the comprehensiveness of surveys related to Differentiable Rendering, Neural Radiance Fields, and Differential Geometry, they fail to cater to someone starting research in this field. So, this survey is able to bridge that gap while also discussing the history of how NeRF came to be.

This research paper make the following important contributions to the already existing research literature (such as Gao et al [63]).

1. This paper can be used by a layman to understand basic concepts of mathematics, computer graphics, computer vision, geometry (differential geometry), etc.
2. It makes a distinct classification based on the Loss Functions, Evaluation metrics, Applications, and by the year it became online (on arxiv or other sources on the Internet)
3. It comprehensively tries to discuss the development of Neural Radiance Fields and how Fourier Features can be used to model high-frequency function for rendering [209,147,194]
4. Further, based on the past and the papers published in 2020–2023, this paper tries to trace out a path of future development of Neural Radiance Fields and Neural Rendering

The §1 presents a brief introduction that this paper aims to resolve i.e. to delineate a classification for NeRF models and discuss about its history and future. The remainder of this paper has been organized as follow. §2 discusses the fundamental and basic concepts and principles of interest in relation to the research related to Neural Rendering and Novel View Synthesis. §3 discusses the various neural architectures before the conceptions of NeRFs. It further discusses various surveys that have been put forward in this domain and the need for this survey. §4 discusses the chronological description of the models, extensions, and improvements over Neural Radiance Fields. §5 briefly discusses the datasets that have been leveraged till now in the field of NeRFs. Further, these are also then taxonomized according to different objectives (discussed in §6), different evaluation metrics discussed in §7), and different applications (discussed in §8). All sections from §5–8 discuss the research in the past couple of years in the field of NeRFs (limited till December 2022). Finally, the future prospects of this family of models is explored in §9. §10 concludes the paper and discusses some future prospects for this research.

2 Basic Principles and Concepts

This section defines the concepts that should be helpful to the reader when considering the possible objective functions and applications that Radiance Fields tries to solve. This is by no means a comprehensive list of concepts and the reader is advised to search on his own. For instance, Voxel Grids, Octree, Triangular meshes, and Radiance Fields (from 2020 onwards) are some of the ways that are used to store and represent 3D geometries.

These concepts can be divided into the basics of Computer Graphics and Computer Vision. Since, NeRFs combine the fields of Computer Graphics, Computer Vision, and Signal Processing, it is important to understand all these concepts for better understanding the rest of the literature.

2.1 Computer Graphics

This section briefly describes some terms in Computer Graphics and that are going to be helpful in reading the remainder of this article.

- **Pixel:** In computer graphics, a pixel (short for "picture element") is the smallest unit of a digital image. It represents a single point in a 2D grid of square or rectangular cells called a raster image. Each pixel contains information about its color and brightness, typically represented using binary values. The combination of millions of pixels arranged in a specific pattern creates a digital image that can be displayed on a screen or printed on paper. In computer vision, a pixel is also the smallest unit of a digital image, but its meaning is slightly different. In this context, a pixel is a mathematical representation of an image that enables algorithms to analyze and process visual information. Each pixel is assigned a numerical value based on its intensity, which can range from 0 (black) to 255 (white) in grayscale images. In color images, each pixel is represented by three or four values that define its red, green, and blue color components, and optionally its alpha channel for transparency.

In both computer graphics and computer vision, pixels are fundamental to the creation and processing of digital images.

- **Resel:** In computer graphics research, a resel (short for "resolution element") is a unit of measurement used to quantify the resolution of an image. It is a mathematical construct that represents the smallest distinguishable element in an image, based on the size of the image's point spread function (PSF). The PSF describes how a point of light in the scene is blurred by the imaging system before being recorded by a camera or other sensor. The resel is defined as the area of the PSF that contains 50% of the total energy of the point of light. In other words, it represents the smallest area in the image that can be resolved as a distinct feature.

The resel is a useful metric in computer graphics research because it can be used to compare the resolution of different imaging systems, or to evaluate the effectiveness of image processing algorithms in enhancing or preserving

image detail. It is also used in fields such as microscopy and astronomy to quantify the resolution of optical systems.

- **Voxel** is a commonplace term used in 3D Computer Graphics and 3D Computer Vision. It represents a discrete value on a grid in 3D space. It is analogous to Pixel (in 2D bitmap space). Voxels don't have their position information encoded but a voxel is rendered based on relative position to other voxels. A voxel has heavy use-case in GIS systems [34], Healthcare systems based on Computer Vision [65,60,199,40,149], and Autonomous Vehicles [125,41,39,30]. These can be used to form another data structure which is discussed further.
- **Tixel** or a tactile pixel is smallest measuring/transmitting element of a tactile element of a tactile matrix. This is one of the fundamental units for Haptic Technologies [104]. Here, the tactile matrix is a machine-readable system, which transmits information from surface to the receiver that is the exoskeleton interface for tactile capturing.
- **Texel** or Texture element or Texture equivalent of a Pixel, is the most fundamental measuring unit of Texture. Textures have an array of texels (much like our computer images which have arrays of pixels). They are image regions obtained through simple procedures like Thresholding and the midpoint of centroids of two surrounding texels can be used to define the spatial relationship between two texels in a texture.
- **Image Sharpening** refers to techniques that highlight the edges and finer details in an image. It is widely used as a preprocessing step for object classification. This is done to increase local contrast and sharpening image. In essence, it works by adding a high-pass filtered version of original image to the image again. It is one of the most commonly used process for sharpening one-dimensional signals and is called unsharp masking. Since, an image can be considered a 2D-signal, hence, this operation can also be used for images. This operation can also be represented by the eq 1

$$S_{i,j} = \hat{x}_{i,j} + \lambda \cdot F(\hat{x}_{i,j}) \quad (1)$$

where $\hat{x}_{i,j}$ is the original pixel values for the image at the coordinates (i, j) , $F(\cdot)$ is the high-pass filter for the image, λ is the tuning parameter greater than or equal to zero, whereas the RHS represents the sharpened pixel for the image at the coordinates (i, j) . λ decides the desirable sharpness of the image. In case of RGB images, $x_{i,j}$, λ , and $S_{i,j}$ become vectors with 3D.

- **Data Storage techniques** These are some of the data storage techniques in 3D Computer Vision and Computer Graphics. This list might not be an exhaustive list but includes most of the formats referred during the future discussions in this paper.
 - **Voxel Grids** represent the grid created by voxels, which are the extensions of “Pixels” into the third dimension. Each voxel represents a small cubic volume in space and can store various properties such as occupancy, color, or texture. Voxel grids are commonly used for tasks such as 3D reconstruction, scene understanding, and object recognition. They offer

several advantages over other data structures, such as point clouds or meshes, including Regularity, Completeness, and Efficiency. Voxel grids can be constructed from various sources of 3D data, such as point clouds, depth maps, or mesh models. The size and resolution of the voxel grid can be adjusted to balance the trade-off between accuracy and computational efficiency. Once constructed, voxel grids can be used for a variety of tasks such as object segmentation, feature extraction, and shape analysis.

- **Polygon (Triangular) Meshes** is a collection of faces, vertices, and edges leveraged to define the shape of the polyhedra object under consideration in 3D Computer Vision, Computer Graphics. These meshes might contain only triangular (Triangular Meshes), Quadrilaterals (Quads), simple Polygons (N-gons) as this can help in simplifying rendering. Occasionally, they might be composed of Polygon with Holes or Concave Polygons.
- **QuadTree** is a tree data structure to partition a two-dimensional space by recursively dividing it into four quadrants or regions. Data at leaf node varies but the leaf generally represent a “unit of interesting spatial information”. The subdivided regions may be square or rectangular, or can even have other arbitrary shapes. It was first coined in 1974 by Finkel et al [58]. Quadtrees and its variants generally tend to decompose the space into adaptable cells. Each of these cells have a maximum capacity and the cells can split when the maximum capacity is reached. A Tree-Pyramid is the spatial analog of the complete trees. Hence, each of its nodes has four child nodes except the leaf nodes. Further, all leave are on the same level. The data in the Tree-Pyramid can be stored compactly in a Binary heap [201].
- **Octree** is the 3D equivalent of a Quadtree where each node has exactly eight children nodes. This data structure can be used to partition a 3D space by dividing it into eight octants and is often used in 3D graphics and game engines such as Unity and Unreal Engine. The term was pioneered by Donald Meagher at Rensselaer Polytechnic Institute [143]. Octree, much like quadtrees may be used as Point-Region Octree (PR-Octree; node stores an explicit 3D point, the center of subdivision of that node; point defines one corner for all eight children), Matrix-based Octree (MX-Octree;)
- **Point Cloud** is a discrete representation of points in space. These points may represent a 3D object or a shape. 3D scanners or photogrammetry softwares are used for generating a Point Cloud. These types of representations are used for several purposes such as 3D computer-aided design (CAD) models for manufacturing parts, for metrology, quality inspections, animations, rendering, etc.
- **Surface Normals** describe the orientation of a surface at a particular point. A surface normal is a vector that is perpendicular to the surface at the given point. Surface normals are often used to determine the shading or lighting of a 3D object, and are a key component in many rendering algorithms. The

surface normal vector can be computed using the gradient of the surface at the given point. Eqn 2 defines the gradient for a smooth, continuous surface.

$$\Delta f(x, y, z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right) \quad (2)$$

where $f(x, y, z)$ is a scalar function that describes the surface. The surface normal vector can then be computed by normalizing the gradient vector as given by the eqn 3.

$$\mathcal{N} = \frac{\Delta f(x, y, z)}{\|\Delta f(x, y, z)\|} \quad (3)$$

where $\|v\|$ denotes the magnitude of vector v . The resulting \mathcal{N} vector is a unit vector that points in the direction of the surface normal.

Surface normals can be computed for various types of 3D data, such as point clouds, meshes, or voxel grids. In practice, surface normals are often computed using a local neighborhood of points or vertices, such as by averaging the normals of adjacent triangles in a mesh or by fitting a plane to a set of nearby points in a point cloud.

- **Transmittance** refers to the fraction of light that passes through a medium as it travels from one point to another. It is often used in computer graphics to model the effects of atmospheric scattering on the appearance of objects in a scene. Transmittance is also used in machine learning to describe the ability of a material or substance to transmit light through it, which is important in fields such as optics and photonics.
- **Reflectance** is the fraction of incident light that is reflected by a surface. It is often used in computer graphics and machine learning to model the appearance of materials and to infer properties of objects from images. Reflectance can be affected by factors such as surface roughness, surface orientation, and the wavelength of the incident light.
- **Radiance** refers to the amount of light that is emitted, reflected, or transmitted by a surface in a given direction. It is typically measured in units of power per unit area per unit solid angle (e.g., watts per steradian per square meter). Radiance is important in computer graphics because it determines the color and brightness of objects in a scene as they are rendered from different viewpoints and under different lighting conditions. In machine learning, radiance is also used to describe the behavior of electromagnetic radiation in the context of remote sensing and other applications.
- **Lighting**
 - **Specular Lighting** is a type of lighting effect in computer graphics that simulates the way light reflects from the shiny surface (such as glass or metal). Its primarily used for creating highlights and glints on such surfaces that helps to make them appear more realistic. It is based on the Phong reflection model (discussed later when discussing Bidirectional Reflectance Distribution Function) that describes how light interacts with surface in terms of three components: Ambient, Specular, Diffuse. Specular component is responsible for simulating the light off shiny surfaces.

The amount of specular lighting at a given point on a surface is determined by the angle between the incoming light ray and the normal vector of the surface at that point, as well as the angle between the outgoing reflection ray and the viewing direction. These angles are defined by the incident vector \mathcal{L} , the surface normal or the normal \mathcal{N} , and the viewing direction vector \mathcal{V} . Hence, the angle between incident light ray and surface normal is given by their dot product as depicted in the eqn 4

$$\cos \theta = \mathcal{L} \cdot \mathcal{N} \quad (4)$$

Reflection of the light ray from the surface is given by vector \mathcal{R} that is the result of incident ray reflecting about the surface normal. This has been given in the eqn. 5.

$$\mathcal{R} = 2(\mathcal{L} \cdot \mathcal{N})\mathcal{N} - \mathcal{L} \quad (5)$$

and the angle between the reflected ray \mathcal{R} and the viewing direction \mathcal{V} is given by the eqn. 6.

$$\cos \alpha = \mathcal{R} \cdot \mathcal{V} \quad (6)$$

The amount of specular lighting at a given point on the surface is then given by the eqn. 7

$$\mathcal{I}_{spec} = K_s \times \mathcal{I}_{light} \times \cos^n \alpha \quad (7)$$

where \mathcal{I}_{light} is the intensity of the incoming light, K_s is a material property that controls the amount of specular reflection, and n is the shininess or roughness of the surface. The shinier the surface, the higher the value of n and the more concentrated the specular reflection will be. In summary, specular lighting is a key component of the Phong reflection model in computer graphics, and is used to simulate the reflection of light off of shiny, reflective surfaces. The amount of specular lighting is determined by the angle between the incoming light ray and the surface normal, as well as the angle between the reflection ray and the viewing direction, and is controlled by material properties such as K_s and the shininess of the surface.

- **Diffuse Lighting** simulates the way light scatters off of matte, non-reflective surfaces, such as wood or paper. It is used to create subtle shading and shadows on such surfaces, which can help to make them appear more realistic.

The amount of diffuse lighting, much like Specular Lighting, at a given point on a surface is determined by the angle between the incoming light ray and the normal vector of the surface at that point. This angle is defined by the incident vector \mathcal{L} and the normal vector \mathcal{N} . The angle between the incident light ray and the surface normal is given by the dot product of the above mentioned values as given in the eqn. 8:

$$\cos \theta = \mathcal{L} \cdot \mathcal{N} \quad (8)$$

The amount of diffuse lighting at a given point on the surface is then given by the eqn 9 given below.

$$\mathcal{I}_{diff} = K_d \times \mathcal{I}_{light} \times \cos \theta \quad (9)$$

where \mathcal{I}_{light} is the intensity of the incoming light, K_d is a material property that controls the amount of diffuse reflection, and $\cos \theta$ is the cosine of the angle between the incident light ray and the surface normal. The higher the value of $\cos \theta$, the more parallel the incoming light is to the surface, and the brighter the resulting diffuse reflection will be. So, diffuse lighting is used to simulate the scattering of light off of non-reflective surfaces. The amount of diffuse lighting is determined by the angle between the incoming light ray and the surface normal, and is controlled by material properties such as K_d .

- **Ambient Lighting** is a type of lighting effect in computer graphics that simulates the way that light is scattered and reflected by the objects in the environment. This is used to simulate the subtle, overall illumination of a scene and can help to create a more realistic and natural-looking image.

In the Phong reflection model, ambient lighting is the simplest component and represents the light that is scattered uniformly in all directions. It is not affected by the direction of the light source or the surface normal, but is instead a constant value that is added to the overall color of the object. The amount of ambient lighting is determined by a material property known as the ambient reflection coefficient, denoted as K_a . The amount of ambient lighting at a given point on a surface is given by the eqn. 10:

$$\mathcal{I}_{amb} = K_a \mathcal{I}_{light} \quad (10)$$

where \mathcal{I}_{light} is the intensity of the ambient light in the scene. The value of K_a determines how much of this ambient light is reflected by the surface. Hence, ambient lighting is a simple and uniform type of lighting effect that simulates the overall illumination of a scene. It is not affected by the direction of the light source or the surface normal, but is instead a constant value that is added to the overall color of the object. The amount of ambient lighting is controlled by a material property known as the ambient reflection coefficient.

- **Albedo** is the property of a surface to be able to reflect sunlight from the sun (/heat from the sun). Hence, light colored surface have a high albedo as they reflect quite a lot of sun rays whereas dark surfaces absorb the rays from the sun. In more technical terms, it can be stated as the ratio of a measure of Diffuse Reflection of Solar radiation to the total amount of Solar radiation.

In computer graphics research, albedo refers to the proportion of incoming light that is reflected by a surface. It is an important parameter used in material models to determine the color and reflectance properties of objects in a scene. The albedo of a surface is typically represented as a value between 0 and 1, where a value of 0 indicates that the surface absorbs all of the incident light, and a value of 1 indicates that the surface reflects all of the incident light.

The albedo of a surface can be calculated using the eqn 11 where \mathcal{A} is albedo, \mathcal{L} is the total amount of light that falls on the surface (incident light), and reflected light \mathcal{R} is amount of light that is reflected by the surface.

$$\mathcal{A} = \mathcal{R}/\mathcal{L} \quad (11)$$

Albedo is often used along with other material properties (like specular reflectivity, roughness) in Computer Graphics for modeling appearance of surfaces in a scene. This dependence of albedo can be mathematically represented as given in the eqn 12.

$$\rho(\theta) = \frac{\rho_0 + \rho_1 \cos \theta}{1 + \rho_0 \cos \theta} \quad (12)$$

where $\rho(\theta)$ represents directional hemisphere reflectance factor representing ratio of reflected radiation to incident radiation at θ incidence angle, ρ_0, ρ_1 are specular and diffuse reflectance coefficient for ratios of reflected to incidence rays for perfectly smooth (reflective) surface and perfectly rough (diffuse) surface. This means the surface albedo depends on both specular and diffuse reflectance factors in turn depending on the roughness, texture, reflectivity. As the angle of incidence θ increases, the specular reflectance factor ρ_0 becomes more important, and the albedo of the surface increases. However, if the surface is surrounded by objects that absorb light, the amount of reflected light will be reduced, which will decrease the albedo of the surface. In addition to the surrounding environment, the albedo of a surface can also be affected by the presence of shadows or other sources of occlusion. If a surface is partially shaded, the amount of incident light will be reduced, which will decrease the albedo of the surface. Overall, the albedo of a surface is a complex phenomenon that depends on several factors. Albedo is an important concept in computer graphics research because it plays a crucial role in determining the overall appearance of objects in a scene, and it can be used to create realistic lighting and shading effects.

- **Lambertian** of a surface is the property that defines an ideal “matte” or diffusely reflecting surface. In computer graphics, it is a type of surface that scatters incident light equally in all directions. It is also known as a diffuse surface, and it is an idealized model of many real-world surfaces such as paper, cloth, and matte painted surfaces.

The Lambertian model assumes that the amount of light scattered by a surface is proportional to the cosine of the angle between the incoming light direction and the surface normal. This can be expressed using an eqn. 13

where ρ is reflectance factor constant value between 0 and 1.

$$\mathcal{LR} = \frac{\rho}{\pi} \quad (13)$$

The Lambertian model is characterized by a constant reflectance factor, which is independent of the angle of incidence of the light. This means that a Lambertian surface appears equally bright from all directions, and it does not produce any specular highlights or reflections.

In computer graphics, the Lambertian model is often used as a simplifying assumption for simulating the lighting of diffuse surfaces, as it is a simple and efficient model that can produce realistic results in many cases. However, it is important to note that not all surfaces behave like Lambertian surfaces, and more complex models may be necessary for accurately simulating the lighting of some surfaces, such as metals or plastics.

- **BRDF (Bidirectional Reflectance Distribution Function)** is a mathematical function to describe how light is reflected by surface in different directions. It is a fundamental concept in Computer Graphics and Computer Vision. In Virtual Environment, it is used to model the reflection of light from surfaces. It is represented by 4 variables which are incoming light direction (ω_i), outgoing light direction (ω_o), surface normal (\mathcal{N}), and light wavelength (λ). It has been represented by the eqn. 14 where $dL(\omega_o)$, $dE(\omega_i)$ are the differential amount of light reflected in the direction of ω_o and differential amount of incident energy in direction of ω_i .

$$f(\omega_i, \omega_o, \mathcal{N}, \lambda) = \frac{dL(\omega_o)}{dE(\omega_i)}, \quad (14)$$

This function describes the amount of light reflected from surface in particular direction for certain incident light direction and surface normal. Many models have been proposed for the BRDF with different parameters and assumptions. One such common model is the Lambertian BRDF which integrates the eqns. 13–14 to get a model that assumes that the surface is perfectly diffuse and reflects light equally in all directions and given by eqn. 15

$$f(\omega_i, \omega_o, \mathcal{N}, \lambda) = \frac{\rho}{\pi} \quad (15)$$

Another model is called Phong BRDF which models specular reflections from shiny surfaces and can be given by the eqn. 16

$$f(\omega_i, \omega_o, \mathcal{N}, \lambda) = \frac{K_d}{\pi} + K_s \frac{n+2}{2\pi} \cos^n \theta, \quad (16)$$

where n is the specular exponent, higher values for which result in sharper specular component, θ is the angle between the perfect specular reflective direction and outgoing direction with angles clamped to $\pi/2$ to avoid negative value, K_d, K_s are the diffuse reflectivity (fraction of the incoming energy that is reflected diffusely) and specular reflectivity (fraction of the incoming

energy that is reflected specularly). This equation can be written in the form of eqn. 17

$$f(\omega_i, \omega_o, \mathcal{N}, \lambda) = \frac{K_d}{\pi} + \frac{K_s}{\pi} (\mathcal{N} \cdot h)^\alpha, \quad (17)$$

where h is halfway vector between ω_o and ω_i and α is the specular component to control the specular highlight. Overall, the BRDF is an important concept in computer graphics and computer vision, as it allows us to model how light interacts with surfaces in virtual environments. By understanding the properties of different BRDF models, we can create more realistic and compelling virtual environments, and develop better algorithms for computer vision applications such as object recognition and tracking.

- **Color Spaces** are mathematical models that define a range of possible colors and provide a mechanism for mapping those colors to numerical values. In computer vision and computer graphics, color spaces are used to represent and process colors in digital images and video. Here are some of the most commonly used color spaces:
 - **RGB (Red-Green-Blue) color space** represents colors as combinations of red, green, and blue values. It is an additive color model, meaning that colors are created by adding together different amounts of red, green, and blue light. RGB color space is widely used in digital displays and image capture devices.
 - **CMYK (Cyan-Magenta-Yellow-Black) color space** is used primarily in printing. It represents colors as combinations of cyan, magenta, yellow, and black values. CMYK is a subtractive color model, meaning that colors are created by subtracting different amounts of ink from a white substrate.
 - **HSV (Hue-Saturation-Value) color space** represents colors using three parameters. Hue represents the color itself, saturation represents the intensity of the color, and value represents the brightness of the color. HSV color space is often used in image processing applications. This can be represented by the following eqns 18–23.

$$R' = R/255; G' = G/255; B' = B/255; \quad (18)$$

$$C_{max} = \max(R', G', B'), \quad C_{min} = \min(R', G', B') \quad (19)$$

$$\Delta = C_{max} - C_{min} \quad (20)$$

$$H = \begin{cases} 0^\circ, \Delta = 0, \\ 60^\circ \times \left(\frac{G' - B'}{\Delta} \text{mod} 6 \right), C_{max} = R' \\ 60^\circ \times \left(\frac{B' - R'}{\Delta} + 2 \right), C_{max} = G' \\ 60^\circ \times \left(\frac{R' - G'}{\Delta} + 4 \right), C_{max} = B' \end{cases} \quad (21)$$

$$S = \begin{cases} 0^\circ, & C_{max} = 0, \\ \frac{\Delta}{C_{max}}, & C_{max} \neq 0 \end{cases} \quad (22)$$

$$V = C_{max} \quad (23)$$

- **LAB color space** is a color space designed to separate color information from brightness. It consists of three components: L, which represents brightness, and a and b, which represent the color in the green-red and blue-yellow axes. LAB color space is commonly used in image processing applications such as color correction and color-based segmentation.
- **YUV color space** separates color information from brightness, similar to LAB. The Y component represents brightness, while the U and V components represent the color in the blue-yellow and green-red axes. YUV color space is commonly used in video processing and compression applications.
- **XYZ color space** is a standard color space used by the International Commission on Illumination (CIE). It is based on the human visual system and represents all colors visible to the human eye. XYZ color space is often used as an intermediate color space in color transformations.

– **Artifacts** In computer graphics, artifacts refer to any unwanted or unintended visual distortions or anomalies that occur in an image or animation. Artifacts can be caused by various factors, such as limitations of hardware or software, errors in algorithms or data processing, or limitations in the human visual system.

There are many different types of artifacts that can occur in computer graphics, including:

1. **Aliasing:** This occurs when an image or object appears jagged or pixelated, especially along edges or curves. Aliasing is caused by undersampling, where the resolution of the image or object is not high enough to accurately represent the details. In general, it refers to the unwanted effects that arise when high-frequency signals appear in computer graphics. An instance of this can be when downscaling any image from its original size so its quite common in NeRFs as they work with a downsampled image dataset.

Aliasing is a common problem in computer graphics that occurs when the resolution of an image is not high enough to accurately represent a high-frequency signal, such as a diagonal line or a curve. This results in the appearance of jagged edges or stair-step patterns, which can make the image look pixelated or low-quality.

Aliasing is caused by the sampling rate, or the frequency at which the image is sampled. The Nyquist-Shannon sampling theorem states that the sampling rate must be at least twice the highest frequency component of the signal in order to accurately represent it. If the sampling rate is too low, high-frequency components of the signal will be lost or undersampled, leading to aliasing.

In computer graphics, aliasing can occur in several contexts. One common example is when rendering lines or curves using rasterization techniques. In this case, the line or curve is represented as a series of pixels, and if the sampling rate is too low, the line or curve will appear jagged or stair-stepped.

To reduce aliasing in computer graphics, a variety of techniques have been developed, including anti-aliasing. Anti-aliasing works by oversampling the image, and then using filtering techniques to smooth out the jagged edges or stair-step patterns. This can be done by averaging the colors of neighboring pixels, or by using more advanced filtering techniques such as Gaussian filtering.

The amount of aliasing in an image can be quantified using a measure known as the Nyquist frequency, which is defined as half the sampling rate. The Nyquist frequency represents the highest frequency component that can be accurately represented in the image. If a signal has a frequency component that is higher than the Nyquist frequency, it will be undersampled and will result in aliasing.

In summary, aliasing is a common problem in computer graphics that occurs when the resolution of an image is not high enough to accurately represent a high-frequency signal. This results in jagged edges or stair-step patterns, which can make the image look pixelated or low-quality. The Nyquist-Shannon sampling theorem states that the sampling rate must be at least twice the highest frequency component of the signal in order to accurately represent it. Techniques such as anti-aliasing can be used to reduce aliasing in computer graphics.

2. **Moiré patterns:** These are interference patterns that occur when two or more repetitive patterns overlap, causing a new pattern to emerge. Moiré patterns can be seen in printed images or screens, and they can also occur in computer graphics when rendering textures or patterns.
3. **Shadow acne:** This occurs when shadows appear jagged or pixelated, especially on curved surfaces. Shadow acne is caused by numerical errors in the algorithms used to calculate shadows.
4. **Banding:** This occurs when gradients appear as distinct bands of color, rather than a smooth transition. Banding can be caused by limitations in color depth or by compression algorithms that reduce the number of colors used in an image.
5. **Blooming:** This occurs when bright areas of an image bleed into darker areas, causing a halo effect. Blooming is caused by limitations in the dynamic range of the display or by overexposure in the original image.
6. **Ghosting:** This occurs when fast-moving objects or camera movements leave a trail or afterimage. Ghosting is caused by limitations in the refresh rate of the display or by motion blur.
7. **Artifacts in 3D rendering:** These can include glitches or errors in geometry or texture mapping, flickering or popping of objects, or distortions caused by limitations in the rendering engine.

To avoid or minimize artifacts in computer graphics, various techniques and technologies are used, such as anti-aliasing, high dynamic range rendering, advanced shadowing algorithms, and color correction. Additionally, careful calibration and testing can help identify and address potential sources of artifacts.

So it can be said that artifacts in computer graphics are any unwanted or unintended visual distortions or anomalies that occur in an image or animation. Artifacts can be caused by various factors, such as limitations of hardware or software, errors in algorithms or data processing, or limitations in the human visual system. There are many different types of artifacts, and various techniques and technologies are used to avoid or minimize them.

- **Alpha Compositing** is a technique used in computer graphics to combine two or more images together based on their transparency values. This technique is commonly used in compositing and image editing applications.

In alpha compositing, each pixel of an image is represented by four values: $RGB\alpha$. The α value represents the transparency of the pixel, with a value of 0 indicating complete transparency and a value of 1 indicating complete opacity. When two images are combined using alpha compositing, the alpha values of each pixel are used to determine how much of each image should be visible in the final result. It is described by the following eqn 24 given below.

$$\mathcal{C} = ((1 - \alpha)\mathcal{C}_1) + (\alpha\mathcal{C}_2) \quad (24)$$

where $\mathcal{C}, \mathcal{C}_1, \mathcal{C}_2$ represent the color of final pixel, pixel in the first image, and pixel in the second image. This equation essentially blends the colors of the two images together based on their alpha values, with the second image contributing more to the final result if its alpha value is higher.

One common use of alpha compositing is to add an image with a transparent background to another image. For example, if an image of a person with a transparent background is overlaid onto a background image using alpha compositing, the person will appear to be seamlessly integrated into the background.

Another use of alpha compositing is to create effects such as fades, glows, and shadows. By manipulating the alpha values of an image, it is possible to create the illusion of transparency and depth, which can be used to add visual interest and realism to a scene.

- **Baking** is the process of performing offline calculations extensively and caching results in a Vertex Attributes/Texture Maps. When coming across this term in future, it might be used in context of generating low-level models, lightmaps, or normal maps.

Baking in computer graphics refers to the process of precomputing and storing lighting and other surface properties of 3D models or scenes into textures or other forms of data that can be used to speed up rendering and improve performance. This process is commonly used in real-time applications such as video games, where real-time rendering performance is crucial.

Baking is done by simulating and rendering the lighting and other surface properties of a 3D model or scene at a high level of detail, and then storing the results in a texture or other form of data. This precomputed data can then be used during real-time rendering to quickly and efficiently simulate the lighting and surface properties of the 3D model or scene.

There are several types of data that can be baked, including lighting, shadows, ambient occlusion, and reflection maps. Each of these types of data can be computed and stored separately, and then combined during real-time rendering to create a more realistic and detailed image.

The process of baking can be broken down into several steps. First, the 3D model or scene is prepared for baking by optimizing the geometry, setting up the lighting and material properties, and defining the areas that need to be baked. Next, the baking process itself is performed, which involves rendering the scene from multiple angles and at different levels of detail to capture the necessary data. Finally, the baked data is stored in a texture or other form of data that can be used during real-time rendering.

The amount of data that needs to be stored for baking can be quite large, especially for complex scenes or models. To optimize performance, various compression and optimization techniques can be used to reduce the size of the data without compromising quality.

In summary, baking in computer graphics is the process of precomputing and storing lighting and other surface properties of 3D models or scenes into textures or other forms of data that can be used to speed up rendering and improve performance. This process involves simulating and rendering the lighting and other surface properties of a 3D model or scene at a high level of detail, and then storing the results in a texture or other form of data. Baking can be used to compute and store data for lighting, shadows, ambient occlusion, and reflection maps, among other things.

- **Dynamic Range** refers to the range of brightness or luminance values that can be represented in an image or scene. It is a measure of the difference between the brightest and darkest parts of the image, and is typically expressed in terms of the ratio between the maximum and minimum luminance values. It can be categorized as follows.

- **HDR (High-Dynamic Range)** refers to the ability to capture, store, and display a wider range of brightness and color values than traditional displays. HDR technology is becoming increasingly popular in computer graphics and is commonly used in applications such as video games, virtual reality, and video production.

Traditional displays typically have a limited range of brightness and color values that they can display, typically around 8 bits per channel (or 24 bits total for RGB color), resulting in a maximum of 256 levels of brightness and color. In contrast, HDR displays can support a much wider range of brightness and color values, with some displays supporting up to 10 or 12 bits per channel, resulting in up to thousands of levels of brightness and color.

HDR is achieved through a combination of hardware and software tech-

nologies. In order to capture and store HDR images, cameras and sensors with higher dynamic range capabilities are used, along with file formats such as EXR or HDR that can store a wider range of brightness and color values. HDR images can be processed and manipulated using specialized software that can take advantage of the increased dynamic range, allowing for more realistic and visually stunning results.

In addition to capturing and storing HDR images, HDR displays are also required to properly display HDR content. This requires the use of specialized hardware and software that can handle the increased dynamic range and color gamut, and can map the wider range of brightness and color values to the limited range of values that can be displayed on the screen.

One common technique used in HDR is tone mapping, which is the process of mapping the wide range of brightness and color values in an HDR image to the limited range of values that can be displayed on a traditional display. This can be done using a variety of techniques, including logarithmic or linear mapping, or through more advanced algorithms such as tone mapping operators.

In summary, High Dynamic Range (HDR) in computer graphics refers to the ability to capture, store, and display a wider range of brightness and color values than traditional displays. HDR is achieved through the use of specialized hardware and software that can handle the increased dynamic range and color gamut, and can map the wider range of brightness and color values to the limited range of values that can be displayed on the screen. HDR technology is becoming increasingly popular in computer graphics and is used in applications such as video games, virtual reality, and video production.

- **LDR (Low-Dynamic Range)** refers to images or video that have a limited range of brightness and color values. LDR images are typically created by capturing or rendering images with a fixed range of brightness and color values, and displaying them on traditional displays with a limited dynamic range.

Traditional displays typically have a limited range of brightness and color values that they can display, typically around 8 bits per channel (or 24 bits total for RGB color), resulting in a maximum of 256 levels of brightness and color. LDR images are captured, processed, and displayed within this limited range, resulting in images that may appear less vibrant or less realistic than HDR images.

In LDR images, the range of brightness and color values is limited by the range of the display device. For example, if an image is captured with a range of brightness values that exceeds the range of the display device, the brightness values will be compressed or clipped to fit within the display range. This can result in loss of detail in the highlights or shadows of the image.

LDR images are still commonly used in many applications, such as traditional video production and broadcast television, where the limited

dynamic range of displays is sufficient for the intended use. LDR images can also be created from HDR images through a process called tone mapping, which maps the wider range of brightness and color values in an HDR image to the limited range of values that can be displayed on a traditional display.

In summary, Low Dynamic Range (LDR) in computer graphics refers to images or video that have a limited range of brightness and color values. LDR images are typically created by capturing or rendering images with a fixed range of brightness and color values, and displaying them on traditional displays with a limited dynamic range. LDR images are still commonly used in many applications, such as traditional video production and broadcast television. LDR images can also be created from HDR images through a process called tone mapping.

- **Yaw, Pitch, Roll** refer to the three degrees of freedom that are associated with the movement of objects on their axes. There are six degrees of freedom for a rigid body. An object can move up-down, left-right, forward-backward, pitch, yaw, roll. This is better depicted by the fig. 1 given below. It would become clear by looking at the following figure.

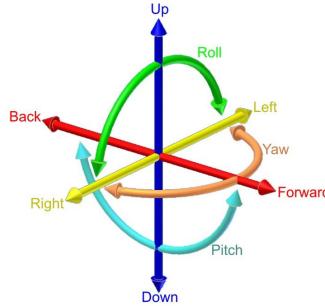


Fig. 1: A schematic representation of six degrees of freedom

- **Gimbal Lock** is a phenomenon in 3D computer graphics and animation that makes the rotations of an object limited or ambiguous, that causes unexpected or incorrect behavior. It occurs when using Euler angles to represent the 3D space orientation of an object.

One of the common measures to represent 3D space orientation of an object is the usage of Euler angles. And these Euler angles are used to define three rotations around all three different axes. However, when the object is rotated to certain positions, the rotations can become aligned or parallel, resulting in a loss of a degree of freedom in the rotation. This leads to the condition called Gimbal Lock.

When visualizing Gimbal Lock, assume a set of three nested rings, each representing a rotation around a different axis (e.g., x , y , and z). When the rings are aligned, the rotations around the y and z axes become the same,

resulting in a loss of 2-degrees of freedom in the rotation.

Assume having an object with three Euler angles representing its orientation: yaw/heading (rotation around the y-axis), pitch (rotation around the x-axis), and roll (rotation around the z-axis). Then this object orientation can be represented as a rotation matrix \mathbf{R} , which can be calculated as given in the eqn. 25 below.

$$\mathbf{R} = \mathbf{R}_x \times \mathbf{R}_y \times \mathbf{R}_z, \quad (25)$$

where \mathbf{R}_x , \mathbf{R}_y , and \mathbf{R}_z are given by the following eqns. 26.

$$\mathbf{R}_z = \begin{pmatrix} \cos \theta_r & -\sin \theta_r & 0 \\ \sin \theta_r & \cos \theta_r & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_p & -\sin \theta_p \\ 0 & \sin \theta_p & \cos \theta_p \end{pmatrix}, \mathbf{R}_y = \begin{pmatrix} \cos \theta_y & 0 & \sin \theta_y \\ 0 & 1 & 0 \\ -\sin \theta_y & 0 & \cos \theta_y \end{pmatrix}, \quad (26)$$

However, when the pitch angle approaches $\pm 90^\circ$, the rotation around the y-axis (yaw) and the rotation around the z-axis (roll) become aligned, resulting in a loss of a degree of freedom. This is gimbal lock, and it can cause unexpected or incorrect behavior in animations and simulations.

One solution to gimbal lock is to use quaternion rotations instead of Euler angles, as quaternions do not suffer from gimbal lock. Alternatively, other rotation representations, such as axis-angle or matrix representations, can also be used to avoid gimbal lock.

2.2 Mathematics and Geometry

This section deals with the basic terminology in the discipline of Mathematics and Geometry which is useful for the rest of the survey.

- **Hadamard Product** is an operation that takes two matrices of the same size and multiplies their corresponding entries together to produce a new matrix of the same size. The Hadamard product is also known as the element-wise product, point-wise product, or Schur product. The Hadamard product of two matrices \mathcal{A} and \mathcal{B} of the same size is denoted by the symbol \odot in the eqn 27 given below.

$$(\mathcal{A} \odot \mathcal{B})_{ij} = \mathcal{A}_{ij} \odot \mathcal{B}_{ij} \quad (27)$$

where i and j represent the row and column indices of the matrices. Hence, the Hadamard product produces a new matrix where each element is the product of the corresponding elements in the original matrices. For instance, the Hadamard product of the following eqn 28 given below.

$$\mathcal{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \text{and } \mathcal{B} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}, \text{then } \mathcal{A} \odot \mathcal{B} = \begin{pmatrix} 5 & 12 \\ 21 & 32 \end{pmatrix} \quad (28)$$

This is a useful operation in various applications of linear algebra, such as image processing, computer vision, and machine learning. For example, in image processing, the Hadamard product can be used to perform point-wise

operations on images, such as contrast enhancement or color manipulation. In machine learning, the Hadamard product can be used to perform element-wise multiplication of feature vectors, which can be useful for regularization or feature selection.

- **Kronecker Product** is a mathematical operation that takes two matrices and produces a new matrix whose entries are all possible products of the entries of the original matrices. It is denoted by the symbol \otimes . Now, when given 2 matrices $\mathcal{A}_{(m \times n)}$ and $\mathcal{B}_{(p \times q)}$, the Kronecker product $\mathcal{A} \otimes \mathcal{B}$ is given by the eqn 29

$$(\mathcal{A} \otimes \mathcal{B})_{ij} = \mathcal{A}_{ij} \times \mathcal{B}_{kl} \quad (29)$$

where i, j, k, l represent row, column indices of the matrices. So, Kronecker product produces a new matrix where each entry is product of the corresponding entries in original matrices, arranged in a matrix block structure. For instance, the Hadamard product of the following eqn 30 given below.

$$\begin{aligned} \mathcal{A} &= \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \text{ and } \mathcal{B} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}, \\ \text{then } \mathcal{A} \otimes \mathcal{B} &= \begin{pmatrix} 1 \times \mathcal{B} & 2 \times \mathcal{B} \\ 3 \times \mathcal{B} & 4 \times \mathcal{B} \end{pmatrix} \Rightarrow \begin{pmatrix} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \end{pmatrix} \end{aligned} \quad (30)$$

The Kronecker product is a useful operation in various applications of linear algebra, such as signal processing, computer vision, and quantum mechanics. For example, in signal processing, the Kronecker product can be used to model the behavior of complex systems by combining simpler models. In computer vision, the Kronecker product can be used to construct feature descriptors for image analysis. In quantum mechanics, the Kronecker product is used to describe the composite state of multiple quantum systems.

- **Signed Distance Function** describes the distance between a point in space and a geometrical object. It is often used in computer graphics, particularly in the field of 3D rendering, to create realistic images of 3D objects. This function returns a signed value, which indicates whether the point is inside or outside the object. If the point is inside the object, the value returned by the SDF is negative; if the point is outside the object, the value is positive. The general equation for an SDF function is given in eqn. 31

$$f(p) = d(p, O) \quad (31)$$

where p is point in space, O is a geometrical object, and d represents the euclidean distance between p and O . The different types of SDFs are given below.

- **Sphere Signed Distance Function** is used to describe a sphere and its equation is as given by eqn 32

$$f(p) = \text{length}(p - o_c) - r \quad (32)$$

where p is point in space, o_c is sphere center, r is sphere radius, **length** represents the Euclidean distance.

- **Box Signed Distance Function** is used to describe a box and its equation is as given by eqn 33

$$\begin{aligned} f(p) = & \max(|(p_x - center_x)| - size_x, 0.0) \\ & + \max(|(p_y - center_y)| - size_y, 0.0) \\ & + \max(|(p_z - center_z)| - size_z, 0.0) \end{aligned} \quad (33)$$

where p is point in space, o_c is box center, **size** is box size.

- **Cylinder Signed Distance Function** is used to describe a cylinder and its equation is as given by eqn 34

$$f(p) = \text{length}(p.xz - o_c.xz) - r \quad (34)$$

where p is point in space, o_c is cylinder center, r is cylinder radius, **length** represents the Euclidean distance.

- **Truncated Signed Distance Function** is a mathematical function that is used to represent a 3D shape in a volumetric way. It is commonly used in 3D computer vision and robotics applications, such as 3D reconstruction, SLAM, and object recognition. The TSDF function is an extension of the Signed Distance Function (SDF) that allows the representation of both the exterior and interior of an object. It does this by truncating the distance function at a certain threshold value, which creates a smooth transition between the interior and exterior regions of the object. TSDF can be represented by the eqn 35

$$\mathcal{TSDF}(p) = \min \left(\max \left(\frac{f(p)}{\text{dis}_{trunc}}, -1 \right), 1 \right) \quad (35)$$

where p is 3D point in space, $f(p)$ is SDF at p and **dis_{trunc}** is truncated threshold to determine distance beyond where function is truncated. Output of TSDF is scalar that indicates distance between p and nearest surface of the object. The output sign indicates whether the point is inside or outside of object. There are different types of TSDF used in Computer Vision as given below.

- * **Volumetric TSDF** is used to represent a 3D object as a voxel grid. The distance values are stored in the voxels of the grid, and the grid is updated as new data is acquired.
- * **Point-based TSDF** is used to represent a 3D object as a set of 3D points. The distance values are computed for each point, and the points are updated as new data is acquired.
- * **Mesh-based TSDF** is used to represent a 3D object as a mesh. The distance values are computed for each vertex of the mesh, and the mesh is updated as new data is acquired.
- **NSDF (Neural Signed Distance Function)** is a type of neural network architecture that is used to learn a continuous representation of

a 3D object from a set of 3D points. It is commonly used in 3D shape reconstruction, object recognition, and robotics applications. The NSDF function takes as input a 3D point and outputs a signed distance value, which indicates the distance between the point and the nearest surface of the object. The sign of the distance value indicates whether the point is inside or outside the object. NSDF can be represented by eqn 36–37 given below.

$$f(p) = \text{MLP}(p), \text{ so} \quad (36)$$

$$\mathcal{NSDF}(p) = \frac{f(p)}{\|f(p)\|_2} \quad (37)$$

where p is 3D point, $\text{MLP}(p)$ is MLP output applied to p , and $\|f(p)\|_2$ is the L2 norm of the MLP output. The NSDF output is a unit vector that points towards the nearest surface of the object. There are different types of NSDF used in computer vision and robotics applications. Some of them are:

- * **PointNet++-based NSDF** is based on the PointNet++ architecture, which is a neural network architecture that is designed to work directly on point clouds. The NSDF is learned by training the PointNet++ network to predict the signed distance value for each point in the input point cloud.
- * **Implicit Function-based NSDF** is based on an implicit function that defines the 3D object surface as the zero level set of the function. The NSDF is learned by training a neural network to approximate the implicit function using a set of 3D points.
- * **Atlas-based NSDF** is based on an atlas of 3D shapes, where each shape is represented as an NSDF. The NSDF is learned by training a neural network to predict the correct NSDF for a given shape in the atlas.
- * **Occupancy-based NSDF** is based on an occupancy function that defines the occupancy of a 3D space. The NSDF is learned by training a neural network to predict the occupancy value for each point in the input point cloud.
- **Dual Contouring SDF** is a method for generating a 3D mesh from the SDF representation of an object. It uses a technique called dual contouring to extract the surface mesh from the SDF.
- **Distance Field SDF** is a method for generating a voxel grid that represents the SDF of an object. It is useful for collision detection and path planning in robotics applications.
- **Occupancy SDF** is similar to the Distance Field SDF, but instead of representing the SDF as a voxel grid, it represents it as an occupancy grid, where each grid cell is either occupied or unoccupied. It is useful for object recognition and scene understanding in computer vision applications.

- **Fast Marching SDF** is a method for computing the SDF of an object using the Fast Marching Method, which is an algorithm for solving partial differential equations. It is useful for real-time applications that require fast computation of the SDF.
- **Level Set SDF** is a method for representing the SDF as the zero level set of a higher-dimensional function. It is useful for modeling the motion of deformable objects and for tracking the evolution of interfaces in fluids and other physical systems.
- **Plenoptic Function** is a mathematical representation of complete information of light rays travelling in the 3D space. It describes how light is emitted from the scene, travels through space, interacts with objects, and is finally captured by a sensor or observer. The plenoptic function is a powerful tool for modeling and simulating various optical effects, such as depth of field, motion blur, and light field imaging. It is a seven-dimensional function that takes into account the position of the light source, its wavelength, the direction of the light rays, and time. This function can be depicted by the eqn 38 given below.

$$\mathcal{P}\mathcal{F}(x, y, z, \theta, \phi, t, \lambda) = \mathcal{I} \quad (38)$$

where x, y, z is 3D space position of the point, θ, ϕ represent the direction of light rays (or sometimes the viewing direction, since light direction is dependent on the viewer direction), t represents time, and \mathcal{I} is the radiance of light intensity at the queried position, directions, and times. In computer graphics, the plenoptic function is often used to simulate the behavior of light in virtual scenes. One extension of the plenoptic function is the bidirectional reflectance distribution function (BRDF) (discussed). There are several extensions of the Plenoptic Function (not related to Computer Graphic), which include:

- **Plenoptic Sampling Function** represents the plenoptic function in a discrete form, by sampling the function at discrete positions and directions. The Plenoptic Sampling Function is used in the design of plenoptic cameras.
- **Compressed Plenoptic Function** compresses the Plenoptic Function by exploiting the sparsity of the function. It is useful for efficient storage and transmission of plenoptic data.
- **Multi-View Plenoptic Function** represents the plenoptic function from multiple viewpoints. It takes into account the position and orientation of multiple cameras or sensors and combines the plenoptic data from each viewpoint to create a single multi-view plenoptic function.
- **Time-Varying Plenoptic Function** represent the plenoptic function over time. It takes into account the time-varying nature of light and captures the changes in the plenoptic function over time.
- **Polarization Plenoptic Function** represents the polarization state of light in addition to the position, direction, and time. It captures the polarization properties of light, which are important for a range of applications including remote sensing and biomedical imaging.

- **SPF (Surface Plenoptic Function)** is the 5-D functional representation of the intensity of light reflected from the surface at every point on the surface, in all directions, and at all times. It is a generalization of the Plenoptic Function that takes into account the surface geometry and the orientation of the surface normal. It is often depicted by the eqn 39 given below.

$$\mathcal{SPF}(x, y, z, \theta, \phi) = \mathcal{L} \quad (39)$$

The SPF is useful for a range of applications in computer vision, graphics, and robotics, including surface reconstruction, object recognition, and image-based rendering. One of the main advantages of the SPF is that it provides a compact representation of the surface appearance that can be used to synthesize novel views of the surface from any viewpoint and under any lighting conditions. To estimate the SPF from images, several methods have been proposed in the past, including Shape-from-Polarization, Shape-from-Shading, and Shape-from-Gradient. These methods use different cues to estimate the surface geometry and the surface normal, which are then used to estimate the SPF.

The SPF is a powerful representation of surface appearance that can capture complex lighting effects and surface properties such as reflectance and transparency. However, it requires accurate estimation of the surface geometry and orientation, which can be challenging in some cases, particularly for complex or textured surfaces.

- **BTF (Bidirectional Texture Function)** describes the variation of a surface's appearance with respect to viewing and illumination angles. BTF is a mathematical model that describes the appearance of a surface as it changes with respect to different viewing and illumination directions. It is a higher-dimensional extension of the BRDF, which describes the reflection properties of a surface. The BTF is typically defined by the eqn. 40

$$\mathcal{BTF}(x, y, \omega_i, \omega_o, \omega_v) \quad (40)$$

where x and y are the spatial coordinates of the surface, ω_i and ω_o are the incoming and outgoing light directions, and ω_v is the viewing direction. The BTF describes how the surface appearance changes as the illumination and viewing directions change. The BTF can be measured using specialized imaging techniques that capture the surface appearance under a variety of viewing and illumination conditions. These measurements can be used to construct a BTF model that can be used for rendering and simulation of the surface appearance under arbitrary lighting and viewing conditions.

The BTF has many applications in computer graphics, including virtual reality, video games, and visual effects. It allows for realistic rendering of surface appearance under complex lighting conditions, such as indirect illumination and multiple light sources. The BTF is often used in conjunction with other reflectance models, such as the BRDF and the BSDF

(bidirectional scattering distribution function), to create more accurate and realistic renderings of surfaces.

These extensions of the Plenoptic Function are important for a range of applications in computer vision, robotics, remote sensing, and other fields that involve capturing, analyzing, and synthesizing light in the 3D space.

- **Spherical Gaussians** are a type of RBF (Radial Basis Function) that are commonly used in computer graphics and computer vision applications to model and represent 3D data. They are particularly useful for tasks such as surface reconstruction, shape analysis, and point cloud processing. The basic form of the spherical Gaussian function is given by the eqn 41.

$$\mathcal{G}(x) = e^{-\frac{\|x\|^2}{\sigma^2}} \quad (41)$$

where x is 3D vector for point in space, $\|x\|$ is its Euclidean distance, and σ scales and controls the size of Gaussian function.

Spherical Gaussians are often combined to give a weighted sum of multiple Gaussians, resulting in a more expressive and flexible function as described in eqn 42

$$f(x) = \sum_{i=1}^N w_i \times \mathcal{G}(x - c_i) \quad (42)$$

where w_i, c_i are the weight factor and center position, and it's taken over a total of N centers.

The parameters of the spherical Gaussians, including the center positions, weight factors, and scaling factor, can be learned from data using various techniques such as least squares or maximum likelihood estimation. The resulting function can then be used to interpolate or extrapolate 3D data, such as point clouds or surface meshes.

Spherical Gaussians have several desirable properties, such as smoothness, isotropy, and rotational symmetry. These properties make them well-suited for modeling and analyzing 3D data, particularly when the data is noisy or incomplete. Spherical Gaussians are also efficient to compute, making them practical for real-time applications.

- **Eikonal Equation** is a partial differential equation that describes the propagation of waves. It is used in fields such as computer vision and graphics, to model various phenomena, including surface deformation and motion of fluids. It can have several forms but one of the general form of the eikonal equation is given by the eqn. 43

$$|\nabla u(x)| = f(x) \quad (43)$$

In computer graphics and computer vision, the Eikonal equation is used to compute the distance function to a surface or object. Given an input point cloud or surface, the Eikonal equation is solved to compute the distance function to the surface at every point in space. This distance function can be used to create level sets, which are surfaces that represent a fixed distance away

from the input surface. Applications of the Eikonal Equation in computer graphics and computer vision include Ray Tracing, Shape from shading, Image segmentation, Medical imaging, level set computation, distance map computation, texture synthesis, path planning, collision avoidance.

- **Fourier Series** is a periodic representation of a function $f(x)$. It is defined as a sum of sine and cosine functions of different frequencies and is mathematically given by the following eqn 44 with a period of 2π , and a_0, a_n, b_n being coefficients depending on function $f(x)$.

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)] \quad (44)$$

Now, these coefficients can be given based on the following formulae given in eqn 45

$$\begin{aligned} a_0 &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) dx \\ a_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx) dx \\ b_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx) dx \end{aligned} \quad (45)$$

- **Fourier Transform** helps to transform function in the time domain to function in the frequency domain. It and its inverse are given by the eqn 45 where k is frequency variable, i is the imaginary unit, $F(x)$ is the Fourier transform, and $f(x)$ is the inverse fourier transform.

$$\begin{aligned} F(x) &= \int_{-\pi}^{\pi} f(x) \exp^{-2\pi i k x} dx \\ f(x) &= \int_{-\pi}^{\pi} F(k) \exp^{2\pi i k x} dk \end{aligned} \quad (46)$$

The Fourier transform allows us to analyze the frequency content of a function and is widely used in signal processing, image processing, and many other fields of engineering and science.

- **Discrete Fourier Transform** is a mathematical technique that is a discrete variant of the Fourier Transform discussed above. It transforms a discrete-time sequence of N complex numbers $x[n]$ into a sequence of N complex numbers $X[k]$ representing the frequency content of the signal. It is defined by the eqn 47 where k is frequency index ranging from 0 to $N - 1$.

$$X[k] = \sum_{n=0}^{N-1} \left(x[n] \exp^{-\frac{2\pi i k n}{N}} \right) \quad (47)$$

It can be calculated using matrix multiplication of the $N \times N$ DFT matrix F and the $N - 1$ column vectors.

- **FFT (Fast-Fourier Transform)** is used for computing the DFT of a sequence of N complex numbers in $O(n \log n)$ rather than $O(n^2)$ required by the naive DFT. The FFT algorithm exploits the symmetry and periodicity properties of the DFT to compute the transform more efficiently.

The FFT algorithm divides the sequence $x[n]$ into smaller sub-sequences, computes the DFT of each sub-sequence recursively, and combines the results to obtain the final DFT. The algorithm can be implemented using either

a radix-2 or a radix-4 butterfly structure, which exploits the symmetry and periodicity properties of the DFT to reduce the number of multiplications and additions required. In computer vision, the FFT is widely used in image processing for operations such as image filtering, convolution, and correlation. These operations involve the computation of the Fourier transform of the image and the filter kernel, multiplication in the frequency domain, and the inverse Fourier transform to obtain the filtered image. The FFT algorithm allows these operations to be performed efficiently on large images in real-time applications such as video processing and computer vision systems.

- **Positional Encodings** is used in Natural Language Processing and sequence modeling to inject the position information of each token in a sequence into the input representation. The idea is to provide the model with information about the position of each token, which is crucial for tasks such as language understanding and translation.

In Transformer-based models, the positional encoding is added to the input embeddings before being fed to the model. The positional encoding is calculated based on the position of each token in the sequence and a set of learned parameters, which encode the position information into a continuous vector space.

The equation for positional encoding is given in eqn 48 where $\text{PE}(pos, 2i)$ and $\text{PE}(pos, 2i + 1)$ are the $2i$ -th and $(2i + 1)$ -th elements of the positional encoding vector for the pos -th position in the sequence, d_{model} is the dimensionality of the model, and i is the index of the embedding dimension.

$$\begin{aligned} \text{PE}_{(pos, 2i)} &= \sin \left(\frac{pos}{10000^{2i/d_{\text{model}}}} \right) \\ \text{PE}_{(pos, 2i+1)} &= \cos \left(\frac{pos}{10000^{2i/d_{\text{model}}}} \right) \end{aligned} \quad (48)$$

The application of positional encoding is not limited to natural language processing. It can also be used in computer vision and other domains where sequential information is important. For example, in image processing, positional encoding can be used to encode the position information of each pixel in an image, which can be useful for tasks such as object detection, segmentation, and classification. In machine learning, positional encoding can be used to improve the performance of models that work with sequences, such as time series forecasting and speech recognition.

In NeRF, the input to the neural network is a sequence of 3D points along each camera ray, which are used to sample the radiance and volume density at each point. To encode the positional information of each point, NeRF uses a positional encoding scheme similar to the one used in Transformer-based models.

The positional encoding in NeRF is calculated by the eqn. 49 given below where PE_i is the i -th element of the positional encoding vector, d is the dimensionality of the positional encoding, and x , y , and z are the 3D coordinates of the point. The sin and cos terms allow the encoding to capture both periodic and non-periodic features in the spatial structure of the scene.

$$\begin{aligned} \text{PE}_{\text{ray}}(i, 2k) &= \sin\left(\frac{i}{10000^{2k/D}}\right) \\ \text{PE}_{\text{ray}}(i, 2k + 1) &= \cos\left(\frac{i}{10000^{2k/D}}\right) \end{aligned} \quad (49)$$

The use of positional encoding in NeRF allows the model to learn a continuous function that can map any point in the 3D space to its corresponding color and opacity. This enables the generation of photo-realistic 3D scenes from a set of 2D images, which can have various applications in fields such as computer graphics, virtual reality, and augmented reality.

- **MHRE (Multi-Resolution Hash Encoding)** is used in Computer Science to represent images using a compact binary code. It is based on the idea of using multiple image resolutions to get local and global frequency information. Here, the image is divided into multiple sub-regions and computing a hash code for each sub-region at different resolutions. Hash code for each subregion is obtained by comparing the pixel values in the sub-region to a predefined threshold. If a pixel value is greater than the threshold, the corresponding bit in the hash code is set to 1, otherwise it is set to 0. The hash codes for all sub-regions at each resolution level are concatenated to form the final binary code for the image.

The idea behind MRHE encoding of an image I , i^{th} sub-region hash code h_i , resolution level r_i (with total m resolutions) is given by the eqn. 50

$$\mathcal{M}\mathcal{H}\mathcal{R}\mathcal{E}(I) = [h_1^{r_1}, h_2^{r_1}, \dots, h_n^{r_1}, h_1^{r_2}, h_2^{r_2}, \dots, h_n^{r_2}, \dots, h_n^{r_m}] \quad (50)$$

The MRHE technique is particularly useful in applications such as image retrieval, object recognition, and image matching, where large-scale image databases need to be searched efficiently. The compact binary code generated by MRHE can be used to quickly compare images and retrieve similar images from a database. Additionally, the multiple resolutions used in MRHE capture both local and global information, making it robust to variations in lighting, texture, and scale.

- **Spectral Bias (/F-principle)** is a phenomenon that can occur in computer vision tasks when the data used to train a model has a specific spectral structure that the model can exploit, leading to better performance than expected based on its ability to capture the underlying statistical structure of the data.

In computer vision, the Fourier Transform is commonly used to analyze the frequency domain of an image. The F-Principle states that the performance of a machine learning model can be improved if the Fourier Transform of the training data has a specific spectral structure, such as a concentrated or sparse distribution of frequencies. This spectral structure can be exploited by the model to learn more efficiently or accurately than if the distribution of frequencies were more uniformly distributed.

For example, consider a dataset of images of faces. If the images in the dataset have a similar lighting condition, such as all being taken in bright,

natural light, then the spectral structure of the dataset will be biased towards certain frequencies related to that lighting condition. If a model is trained on this dataset, it may learn to exploit this spectral bias and perform well on images with similar lighting conditions, but may perform poorly on images with different lighting conditions.

To mitigate the effects of spectral bias, it is important to use diverse and representative datasets during training, and to evaluate the performance of the model on out-of-distribution data to ensure it is generalizing well to new situations. Additionally, techniques such as data augmentation, which introduces variations in the training data, can help to reduce the impact of spectral bias and improve the generalization performance of the model.

- **NTKs (Neural Tangent Kernels)** are a mathematical tool used to analyze the training dynamics of neural networks. It allows us to understand how the weights of a neural network change during training and how this affects the network's ability to generalize to new data.

In the context of computer vision, NTK has been used to analyze the performance of neural networks on image classification tasks. Additionally, NTK has been applied to novel applications such as Neural Radiance Fields (NeRF) which is a method for synthesizing photo-realistic 3D scenes from 2D images. The eqn 51 represents an NTK where $K_{NTK}(x, x')$ is the NTK between input vectors x and x' , $f(x)$ is the output of the neural network for input x , and w_i represents the weights of the neural network.

$$K_{NTK}(x, x') = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \frac{\partial f(x)}{\partial w_i} \frac{\partial f(x')}{\partial w_i} \quad (51)$$

The NTK is computed as the limit of the average product of the partial derivatives of the network output with respect to the weights over an infinite number of weights.

In the context of NeRF, NTK is used to analyze the performance of a neural network that maps 2D images to 3D scenes. By analyzing the NTK, researchers can gain insight into the relationship between the 2D images and the resulting 3D scene and design more effective neural network architectures. Specifically, NTK allows for the efficient computation of the Jacobian matrix, which is used to estimate the gradient of the loss function during training.

- **Tracing and Casting** are two of the important techniques used in Computer Vision and Computer Graphics. In Vision, tracing refers to the process of tracking the movement of objects/points in sequence of images/frames. This can be done by using various algorithms and techniques, such as optical flow, feature tracking, and point matching. The goal of tracing in computer vision is to understand the motion and dynamics of objects in a scene and extract useful information, such as the velocity, acceleration, and trajectory of the objects. Whereas in Graphics, tracing refers to the process of generating photorealistic images by tracing the paths of light rays as they interact with virtual objects in a scene. This is typically done using ray tracing algorithms, which simulate the behavior of light as it interacts with objects in

the scene. The goal of tracing in computer graphics is to generate realistic images that accurately simulate the lighting and materials of the virtual objects in the scene.

Casting, on the other hand, refers to the process of projecting an object or point onto a surface or plane in both computer vision and computer graphics. In computer vision, casting is used to estimate the 3D position of an object or point in the world from a 2D image or frame. This is typically done using methods such as perspective projection or homography, which involve modeling the relationship between the 3D world and the 2D image. The goal of casting in computer vision is to estimate the position and orientation of objects in the world from a 2D image or frame.

In computer graphics, casting is used to simulate the behavior of light as it interacts with objects in a scene. This is typically done using methods such as shadow casting and light mapping, which involve modeling the way that light interacts with objects in the scene to create realistic shadows and lighting effects. The goal of casting in computer graphics is to generate realistic images that accurately simulate the behavior of light in the scene.

- **Path Tracing** is a widely used algorithm in computer graphics for rendering realistic images of 3D scenes. It is a Monte Carlo method that simulates the behavior of light as it interacts with objects in the scene. The algorithm works by tracing a path of light rays from the camera through the scene, randomly sampling the possible paths of light rays as they reflect, refract, and scatter off of objects in the scene. Path tracing has been used in a wide range of applications, from film and animation to video game development. One of the seminal papers on path tracing is work done by Kajiya et al [103], which introduced the concept of a physically-based model for rendering 3D scenes. Another influential work by Matt Pharr and Greg Humphreys [171] provides a comprehensive introduction to the theory and practice of physically-based rendering, including path tracing.
- **Ray Marching** is used to sample multiple points from the 3D space rather than just one point where the ray intersects with the 3D surface. This method is used when analytical methods fail to provide reliable results. Its used in computer graphics to render 3D scenes using volumetric data by tracing a ray through a volume and sampling the density and color of the volume along the ray's path. Ray marching is commonly used to generate realistic images of clouds, smoke, and other volumetric phenomena. Seminal works on ray marching [121,53] introduced the concept of volume rendering and the ray marching algorithm. Another work by William E. Lorensen and Harvey E. Cline [168] proposed a variant of ray marching for generating isosurfaces of volumetric data.
- **Ray Tracing** is a technique used for 3D offline rendering. It is achieved by recursively tracing the path of rays through a scene. It tracing is a technique used in graphics and vision to generate realistic images by tracing the path of light as it interacts with objects in a scene. It works by tracing rays of light from the camera through each pixel and calculating

how the light interacts with objects in the scene, including reflection, refraction, and shadows. The first work to introduce ray tracing concept was by Turner Whitted [218] as it presented a recursive algorithm for computing reflections and refractions.

- **Distributed Ray Tracing** is used in computer graphics and computer vision to generate realistic images by simulating the propagation of light in a scene using a distributed network of processors. It involves dividing the rendering process into smaller tasks, distributing them across multiple processors, and then combining the results to produce a final image. It is achieved by some modifications to the process of Ray Tracing where multiple rays are traced through each pixel to model soft phenomenon such as soft shadows, depth of fields. One of the first works to introduce this concept was presented by Cook et al [38] and it initiated the concept of parallelizing the ray tracing algorithm across multiple processors.
- **Beam Tracing** uses pyramid-shaped beams to address some of the shortcomings of the ray-tracing. These shortcomings include Aliasing effects and the creation of artifacts while rendering.
- **Ray Casting** is generally a 2.5 D rendering method and is achieved by casting non-recursive rays from the camera into the scene.
- **Sphere Tracing**, also known as raymarching or distance field raymarching, is an algorithm for rendering 3D objects using signed distance functions (SDFs). An SDF is a function that, for any point in space, returns the shortest distance to the surface of the object.
- **Marching Cube** is a method for implicit surface triangulation. So, it is an algorithm for generating polygonal meshes from 3D scalar fields, such as voxel data or isosurfaces. The algorithm works by dividing the volume into small cubes and determining the surface intersection points within each cube.

The following steps constitute the Marching Cube algorithm.

1. Divide the volume into small cubes.
2. For each cube, evaluate the scalar field at its 8 vertices.
3. Determine the surface intersections by comparing the scalar values at the vertices to a predefined isovalue.
4. Create triangles to connect the intersection points within each cube.
5. Combine the triangles to form the final polygonal mesh.

The mathematical formulation for each step is as follows.

1. Let $V(x, y, z)$ be a scalar field representing the 3D volume.
2. For each cube vertex, evaluate the scalar field: $v = V(x, y, z)$
3. Create a bitmask by comparing v to the isovalue.
4. Use the bitmask to look up the corresponding edge intersections.
5. Calculate the intersection points: $I = (p1 + p2) / 2$, where $p1$ and $p2$ are the endpoints of the intersecting edge.
6. Generate triangles connecting the intersection points within the cube.

Applications for Marching Cube include

- * Medical imaging and visualization (CT, MRI, etc.)
- * Geological data visualization

- * Fluid dynamics simulations
- * Terrain generation in video games and virtual environments
- **SLAM** It is a technique often used at the intersection of Robotics and Computer Vision. It further helps with several simple or complex Robotic operations. It generally requires the leverage of C++ to work with robots for manipulation, localization, path finding, etc.
- **RGB-D Registration** is the use of Depth along with RGB features to register for different application such as Nuclear Magnetic Resonance Imaging (NMRI), CT scans, Geospatial imaging (when non-azimuthal orbit), etc.
- **DeepV2D (Deep Video to Depth)** is a deep learning-based method for estimating depth maps from a video sequence, leveraging differentiable Structure-from-Motion (SfM) to improve the overall performance. The key idea of DeepV2D is to integrate the geometric principles of SfM with a deep learning framework to jointly optimize for depth, camera motion, and 3D structure. This approach allows the model to learn depth estimation while enforcing geometric consistency across multiple frames in a video sequence. The main components of the DeepV2D framework are:
 1. CNN-based Depth Estimation: A convolutional neural network (CNN) is used to estimate the initial depth maps for each frame in the video sequence. This network is trained to predict depth maps from single images, but its predictions may not be geometrically consistent across frames.
 2. Differentiable SfM: The differentiable SfM module enforces geometric consistency across frames by optimizing the camera motion (poses) and refining the depth estimates. The module consists of differentiable components for camera pose estimation, warping, and a photometric loss function. The differentiability of the SfM module allows the gradients to flow back through the entire pipeline during training, improving both depth estimation and camera motion estimation.
 3. Joint Optimization: DeepV2D jointly optimizes depth maps, camera poses, and 3D structure by minimizing a combination of photometric and geometric loss functions. The photometric loss measures the difference between the input images and the images reconstructed by warping neighboring frames using the estimated depth and camera poses. The geometric loss enforces smoothness in the depth maps and penalizes large deviations from the initial CNN-based depth predictions.
- DeepV2D is trained in an end-to-end manner using backpropagation, allowing the model to learn depth estimation and camera motion jointly. The integration of differentiable SfM in the deep learning framework helps the model to enforce geometric consistency across video frames and improve the overall performance of depth estimation.
- **COLMAP** is a library written in C++ for two of the vital operations in Computer Vision and Computer Graphics namely, MVS and SfM which have been discussed below.
 - **MVS (Multi-View Stereo)** is a computer vision technique used to reconstruct 3D geometry from multiple 2D images taken from different

viewpoints. It builds on the principles of stereo vision, which involves estimating the depth of a scene by comparing the slight differences in appearance between two or more images, known as parallax.

The main steps involved in Multi-View Stereo are:

1. **Image Acquisition:** Capture a set of images of a scene or object from multiple viewpoints, ensuring sufficient overlap and coverage. Ideally, the camera's intrinsic (focal length, principal point, etc.) and extrinsic (position and orientation) parameters should be known or estimated.
2. **Feature Extraction and Matching:** For each image, identify distinctive points or features and match them across the different images. This can be achieved using feature detectors and descriptors, such as SIFT, SURF, ORB, etc.
3. **Pairwise Stereo Matching:** Compute the depth or disparity maps for each pair of images, based on the feature matches. This can be done using block-matching, dynamic programming, or graph-cut algorithms, for example.
4. **3D Reconstruction:** Merge the depth or disparity maps from all image pairs into a single, consistent 3D model. This can involve techniques like depth map fusion, volumetric reconstruction, or point cloud merging.
5. **Post-processing (optional):** Refine the 3D model by applying methods like meshing, texturing, and hole-filling to improve the visual quality and completeness of the result.

Multi-View Stereo has various applications, including:

- * 3D scanning and modeling of objects and environments
- * Photogrammetry
- * Augmented and virtual reality
- * SLAM (Robotics and autonomous navigation)
- * Cultural heritage preservation

In summary, Multi-View Stereo is a technique used to reconstruct 3D geometry from multiple 2D images taken from different viewpoints. It involves feature extraction and matching, pairwise stereo matching, and 3D reconstruction, and can be applied in diverse fields such as photogrammetry, augmented reality, and robotics.

- **SfM (Structure from Motion)** is a computer vision technique used to estimate the 3D structure of a scene and the motion (camera positions and orientations) from a set of 2D images taken from different viewpoints. It combines elements of both feature-based image matching and 3D reconstruction to create a sparse 3D point cloud or a more detailed 3D model.

The main steps involved in Structure-from-Motion are:

- * **Image Acquisition:** Capture a set of images of a scene or object from multiple viewpoints, ensuring sufficient overlap and coverage.
- * **Feature Extraction and Matching:** For each image, identify distinctive points or features and match them across the different im-

ages. This can be achieved using feature detectors and descriptors, such as SIFT, SURF, ORB, etc.

- * **Estimating Camera Poses and 3D Structure:** Use the feature matches to simultaneously estimate the camera's extrinsic parameters (position and orientation) and the 3D coordinates of the features in the scene. This can be done using algorithms like bundle adjustment, which minimizes the reprojection error between the observed 2D feature positions and the reprojected 3D points.
- * **Incremental Reconstruction:** Add images one at a time, updating the camera poses and 3D structure iteratively. This process is known as incremental SfM and helps maintain accuracy and consistency in the reconstruction.
- * **Dense Reconstruction (optional):** Perform a dense Multi-View Stereo (MVS) on the sparse 3D point cloud obtained from SfM to generate a more detailed 3D model.
- * **Post-processing (optional):** Refine the 3D model by applying methods like meshing, texturing, and hole-filling to improve the visual quality and completeness of the result.

In summary, Structure-from-Motion is a technique used to estimate the 3D structure of a scene and the motion (camera positions and orientations) from a set of 2D images taken from different viewpoints. It involves feature extraction and matching, camera pose and 3D structure estimation, and optional dense reconstruction and post-processing, and can be applied in diverse fields such as photogrammetry, augmented reality, and robotics.

- **Structured Light Scanning and Rendering** is a 3D operation for getting RGB-D data using LASERs to scan the surface of the object which needs to be sampled for the data. The LiDAR technology works on these principles. LiDAR is often adjusted on cars for autonomous driving capabilities.
- **Time-of-Flight Scanning** refers to the scanning carried out using planes or quadcopters for the purpose of understanding the terrain under the position where the plane/drone is flying.
- **Jitter** refers to the noise while streaming a sequence of images using different medium. This can restrain the company from getting optimal sampled input.
- **Image and Facial Features**
 - **SIFT (Scale-Invariant Feature Transform)** [137] is a feature extraction method proposed by David Lowe in 1999. It is designed to be invariant to scale, rotation, and illumination changes. The main steps of the SIFT algorithm are:
 1. Scale-space extrema detection: Identify potential interest points by searching for local maxima and minima across multiple scales.
 2. Keypoint localization: Refine the candidate keypoints by eliminating low-contrast and poorly localized points.
 3. Orientation assignment: Assign a dominant orientation to each keypoint based on local gradient histograms.

4. Keypoint descriptor computation: Create a feature descriptor for each keypoint using histograms of gradient orientations in the keypoint's local neighborhood.
- **SURF (Speeded-Up Robust Features)**, proposed by Herbert Bay et al. in 2006, is an image feature extraction method inspired by SIFT but designed to be faster and more efficient. The main steps of the SURF algorithm are:
 1. Interest point detection: Use the Hessian matrix to detect blob-like structures in the image at multiple scales.
 2. Keypoint localization: Identify the local maxima and minima of the determinant of the Hessian matrix as potential keypoints.
 3. Orientation assignment: Assign a dominant orientation to each keypoint based on the Haar wavelet responses in its local neighborhood.
 4. Keypoint descriptor computation: Create a feature descriptor for each keypoint using Haar wavelet responses in the keypoint's local neighborhood.
- **ORB (Oriented FAST and Rotated BRIEF)** proposed by Ethan Rublee et al. in 2011, is a fast binary descriptor that combines the strengths of the FAST keypoint detector and the BRIEF descriptor. The main steps of the ORB algorithm are:
 1. Interest point detection: Use the Hessian matrix to detect blob-like structures in the image at multiple scales.
 2. Keypoint detection: Use the FAST (Features from Accelerated Segment Test) corner detector to identify interest points in the image.
 3. Orientation assignment: Compute an orientation for each keypoint based on the intensity centroid of its local neighborhood.
 4. Keypoint descriptor computation: Create a feature descriptor for each keypoint using the BRIEF (Binary Robust Independent Elementary Features) descriptor, which is modified to be rotation-invariant.
- **3DMM feature** refer to 62D features for the face alignment (yaw, pitch, roll) that helps to understand the face of the person being considered for the sampled input. 3DMM, introduced by Volker Blanz and Thomas Vetter in 1999, is a statistical model that represents the 3D shape and texture of human faces. The 3DMM is generated by analyzing a dataset of 3D face scans, capturing the variations in shape and texture across individuals. The main components of a 3DMM are:
 1. Shape model: A principal component analysis (PCA) is applied to the 3D facial geometry data to create a linear shape model, which can represent different face shapes as a combination of the mean face shape and a set of principal components.
 2. Texture model: Similarly, PCA is applied to the texture data (e.g., color or reflectance) of the 3D face scans to create a linear texture model.
 3. Model fitting: Given a 2D image or a set of images, the 3DMM can be fit to the input data by estimating the shape and texture coefficients,

as well as the camera parameters and illumination conditions, that minimize the discrepancy between the input

- **RANSAC (RANdom SAmple Consensus)** is an iterative algorithm used in the field of computer vision and other areas for robustly estimating model parameters from a dataset that contains outliers. The main idea behind RANSAC is to repeatedly select a random subset of the data points, fit the model to this subset, and evaluate the quality of the fit. The model with the highest consensus from the data points is chosen as the final result. The RANSAC algorithm consists of the following steps:

- Randomly select a minimal subset of data points: Choose the minimum number of data points required to fit the model (e.g., 2 points for a line, 3 points for a plane, etc.).
- Fit the model: Estimate the model parameters using the selected data points.
- Evaluate the model: Calculate the error between the model’s predictions and the remaining data points. If the error is below a predefined threshold, consider the data point as an inlier, otherwise, as an outlier.
- Count the inliers: Determine the number of inliers that support the fitted model.
- Iterate: Repeat steps 1-4 a predefined number of times or until a stopping criterion is met (e.g., the proportion of inliers exceeds a threshold or the maximum number of iterations is reached).
- Refine the model: Fit the model again using all inliers from the best iteration to obtain the final model parameters.

RANSAC is particularly useful when dealing with data that contains a significant number of outliers, as it is resistant to their influence. The algorithm is widely used in computer vision tasks, such as feature matching, homography estimation, fundamental matrix estimation, and camera pose estimation.

Hence, RANSAC is an iterative algorithm for robustly estimating model parameters from a dataset with outliers. It works by repeatedly selecting a random subset of data points, fitting the model, and evaluating the consensus of the data points. The model with the highest consensus is chosen as the final result. RANSAC is widely used in computer vision and other fields where robust parameter estimation is required.

- **Selective Search** is a region proposal algorithm used in the context of computer vision, specifically for object detection tasks. The main idea behind Selective Search is to generate potential bounding boxes or regions of interest that are likely to contain objects in an image. These regions are then used as input for a classifier or a more complex detection pipeline to identify and classify objects.

Selective Search combines the strengths of both exhaustive search and segmentation to create a diverse set of region proposals. The algorithm is based on the assumption that objects can be identified at different scales and are often characterized by distinct colors, textures, or intensity patterns.

The main steps of Selective Search are:

1. Generate initial regions: Perform an initial over-segmentation of the image using a simple method like the Felzenszwalb and Huttenlocher's graph-based segmentation algorithm. This step creates many small, highly detailed segments.
2. Extract features: For each segment, compute features like color, texture, and shape.
3. Hierarchical grouping: Recursively merge the most similar segments based on a similarity measure computed from the extracted features. This step creates a hierarchical structure of regions, where each level represents a different granularity or scale.
4. Generate region proposals: Collect regions from different levels of the hierarchy and use non-maximum suppression to remove redundant or overlapping proposals. This step produces a diverse set of potential object locations.

Selective Search is particularly useful as a preprocessing step for object detection algorithms, as it reduces the number of candidate regions that need to be processed by the actual object detector. This can significantly speed up the detection process and improve the overall performance. Selective Search was widely used in combination with traditional object detectors, like those based on the Histogram of Oriented Gradients (HOG) features, and early deep learning-based detectors, like R-CNN.

- **Bundle-Adjustment** is an optimization technique used in computer vision and photogrammetry for refining estimates of the 3D structure of a scene and the camera parameters (poses and intrinsics) from a set of overlapping images. It is often employed as a final step in Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipelines to improve the accuracy and consistency of the 3D reconstruction.

The main idea behind Bundle Adjustment is to minimize the reprojection error, which is the difference between the observed 2D positions of features in the images and the reprojected 3D points onto the images using the estimated camera parameters and 3D structure. Bundle Adjustment is a non-linear least squares optimization problem, as the relationship between the camera parameters, 3D structure, and image measurements is non-linear.

The main components of Bundle Adjustment are:

- **Camera parameters:** The intrinsic (e.g., focal length, principal point, lens distortion) and extrinsic (e.g., rotation and translation) parameters of each camera in the scene.
- **3D structure:** The 3D coordinates of the scene points that are visible in the images.
- **Image measurements:** The 2D coordinates of the features in each image, usually obtained through feature matching.
- **Objective function:** The sum of the squared reprojection errors between the observed image measurements and the reprojected 3D points, given the current estimates of the camera parameters and 3D structure.
- **Optimization algorithm:** A non-linear least squares solver, such as the Levenberg-Marquardt algorithm, is used to iteratively minimize the objective function.

So, Bundle Adjustment is an optimization technique used to refine the estimates of the 3D structure of a scene and the camera parameters by minimizing the reprojection error. It is widely employed in computer vision and photogrammetry pipelines, such as Structure-from-Motion and Multi-View Stereo, to improve the accuracy and consistency of the 3D reconstruction.

- **Shepard Metzler objects** are complex 3D shapes, originally introduced by psychologists Roger Shepard and Jacqueline Metzler in a 1971 study on mental rotation. In the context of computer vision, these objects have been adopted as a tool to assess and evaluate the performance of models in tasks related to 3D understanding, such as 3D object recognition, pose estimation, and unsupervised feature learning.

Shepard-Metzler objects are typically constructed by combining several simple 3D shapes, such as cubes or cylinders, into a single complex structure. The resulting objects exhibit a high degree of variability in terms of their geometry, making them well-suited for testing the ability of computer vision algorithms to recognize, analyze, and understand 3D structures.

Some common use cases of Shepard-Metzler objects in computer vision research include:

- Benchmarking: Researchers use Shepard-Metzler objects as a benchmark dataset to evaluate the performance of their algorithms in tasks like 3D object recognition and pose estimation. By using these complex objects, they can test the robustness and generalization capabilities of their models.
- Unsupervised feature learning: Shepard-Metzler objects are used to study unsupervised feature learning in the context of 3D understanding. Researchers may train models to learn meaningful representations of these objects, which can then be used for downstream tasks such as classification, segmentation, or reconstruction.
- Mental rotation tasks: Inspired by the original psychological study, computer vision researchers can use Shepard-Metzler objects to explore how models perform mental rotation tasks, which involve determining whether two 3D objects are the same when they are presented at different orientations.
- **Super-Resolution Tasks** aim to enhance the quality of an image or recover missing or degraded information. While super-resolution often refers to increasing the spatial resolution of an image, it also encompasses other image enhancement tasks like denoising, dehazing, deblurring, and inpainting. These tasks address different types of image degradation and improve the overall quality and visual appearance of an image.
 - **Image Denoising** is the process of removing noise from an image while preserving its original details. Noise can be introduced during image acquisition (e.g., due to sensor limitations) or transmission (e.g., compression artifacts). Denoising techniques range from traditional filtering methods, such as Gaussian filters or non-local means, to more advanced deep learning-based approaches, like denoising autoencoders and convolutional neural networks.

- **Image Dehazing** refers to the removal of haze or fog from an image to improve its visibility and contrast. Haze is caused by the scattering of light due to the presence of atmospheric particles, which can result in low contrast and color shift. Dehazing methods usually estimate the transmission map and atmospheric light to recover the haze-free image. Classical methods include Dark Channel Prior and Guided Filtering, while deep learning-based methods leverage convolutional neural networks for end-to-end dehazing.
- **Image Deblurring** aims to recover a sharp image from a blurred one, typically caused by camera shake, object motion, or out-of-focus effects. Deblurring techniques estimate the blur kernel (also known as point spread function) and use it to restore the sharp image. Traditional methods include blind deconvolution and Wiener filtering, while deep learning-based methods utilize convolutional neural networks to either directly estimate the sharp image or the blur kernel.
- **Image Inpainting** is the process of filling in missing or corrupted parts of an image in a visually plausible manner. This task requires not only filling in the missing regions but also preserving the structure and texture of the surrounding areas. Classical inpainting methods include diffusion-based techniques and patch-based approaches like exemplar-based inpainting. More recent deep learning-based methods leverage generative models, such as autoencoders and generative adversarial networks (GANs), to learn the underlying structure and texture patterns and generate plausible content for the missing regions.
- **SISR (Single Image Super-Resolution)** aims to recover a high-resolution (HR) image from a single low-resolution (LR) input image. Traditional methods include interpolation-based techniques (e.g., bicubic interpolation, Lanczos resampling) and learning-based approaches (e.g., sparse coding). More recently, deep learning-based methods, such as super-resolution convolutional neural networks (SRCNN) and generative adversarial networks (GANs), have demonstrated significant improvements in reconstructing HR images with more accurate textures and details.
- **Video super-resolution** aims to improve the spatial resolution of video sequences while maintaining temporal consistency. This task presents additional challenges due to the need to consider motion between frames and avoid flickering artifacts. Techniques for video super-resolution include multi-frame methods, which exploit the temporal information present in multiple frames, and deep learning-based methods, which use convolutional neural networks (CNNs) or recurrent neural networks (RNNs) to model the spatial and temporal relationships between frames.
- **Image colorization** is the process of adding color to grayscale images while preserving their original structure and texture. Traditional methods often require user input in the form of color scribbles or reference images to guide the colorization process. Deep learning-based approaches, such as convolutional neural networks (CNNs) and generative adversar-

ial networks (GANs), have shown the ability to automatically learn color patterns and generate plausible colorizations without user intervention.

- **Image-to-Image translation** is a more general task that involves converting an input image from one domain or representation to another, such as converting a sketch to a photorealistic image or transforming a day-time scene to a night-time scene. Deep learning-based methods, such as conditional GANs (e.g., Pix2Pix) and cycle-consistent GANs (e.g., CycleGAN), have been successful in learning the mapping between different image domains without requiring paired training data.
- **Depth estimation** is the task of recovering depth information from 2D images, which can be used for various applications like 3D reconstruction, autonomous navigation, or augmented reality. Traditional methods typically rely on stereo vision or structured light, while more recent deep learning-based approaches use convolutional neural networks (CNNs) to predict depth maps directly from single images or monocular video sequences.
- **Image Warping**, also known as image transformation or image registration, is a process used in computer vision and graphics to manipulate an image's shape, size, and orientation by applying a transformation to its pixel coordinates. The goal of image warping is to change the appearance of an image while preserving its structure and content. It is commonly used in tasks such as image stitching, rectification, stabilization, and view synthesis. Image warping can be performed using different types of transformations, including:
 1. Rigid Transformations: These transformations preserve the distances between points and include rotation, translation, and their combination. Rigid transformations are often used in image registration to align images taken from different viewpoints or at different times.
 2. Affine Transformations: Affine transformations extend rigid transformations by adding scaling and shearing effects. They preserve parallelism but not necessarily angles or distances. Affine transformations are commonly used in tasks like image rectification, where the goal is to remove perspective distortion and align parallel lines in the image.
 3. Homographies: Homographies are projective transformations that map points in one image plane to another while preserving straight lines. They can represent a wide range of transformations, including rotation, translation, scaling, shearing, and perspective distortion. Homographies are frequently used in tasks like image stitching, where multiple images of a scene are combined into a single panoramic image, and image rectification.
 4. Non-linear or Deformable Transformations: These transformations can model complex, non-linear changes in the image geometry, such as bending, stretching, or twisting. Non-linear transformations are often employed in tasks like image morphing, where one image is smoothly transformed into another, or in deformable registration, where images with non-rigid structures need to be aligned.

To apply a transformation to an image, a mapping function is used to determine the correspondence between the source image's pixel coordinates and the destination image's pixel coordinates. Then, a resampling or interpolation method, such as nearest-neighbor, bilinear, or bicubic interpolation, is applied to estimate the pixel values in the destination image.

- **Decal sheet or Texture Atlas** is a “sticker” picture that is applied onto a 3D mesh to represent a texture.
- **UV-mapping** is the process of 3D surface projection to a 2D image for texture mapping. Here, U and V represents the axes of 2D texture because in Computer Graphics, X, Y, Z are used to denote axes of 3D surfaces, points, objects and W is used to denote calculation of Quaternions rotations. In **UV texture map** [151], an ordinary image can be used to paint a 3D surface with color (much like what is done in Unity). The UV mapping technique involves “programmatically” copying a triangle-shaped section of the image map and pasting it onto a triangle on the object to assign pixels in the picture to surface mappings on the polygon [152]. UV texturing is an alternative to projection mapping; it simply maps into a texture space rather than the physical space of the object (e.g., utilizing any pair of the model’s X, Y , and Z coordinates or any modification of the position). The UV texture coordinates are used in the rendering computation to specify how to paint the three-dimensional surface.

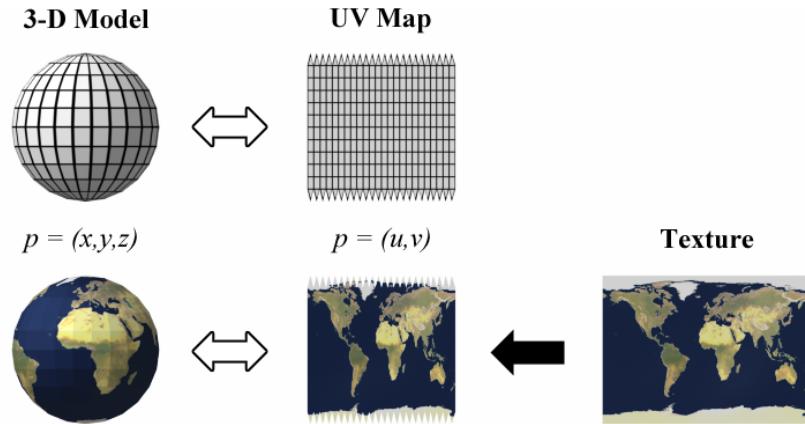


Fig. 2: A schematic representation of an application of a texture in the UV space related to effect in 3D space

UV coordinates can be generated for each vertex of the mesh for a model created as a polygon mesh. A way to achieve this is by unfolding the triangular

meshes at the seams, hence, laying out triangles on flat surface automatically. A UV sphere might be transformed into an equirectangular projection. The textures can be painted on the triangle individually after the model is unwrapped and when rendering the 3D object, it can take up the appropriate texture from the texture atlas.

UV coordinates are applied to each face optionally [152] so a shared spatial vertex position can have different UV coordinates for each of its triangles so adjacent triangles may be cut apart and positioned on texture maps' different areas. For finding the UV coordinates of a point \hat{A} on a sphere with \hat{d} (distance of the point from the sphere's origin) when sphere's poles are aligned with the Y , then the eq. 52

$$u = 0.5 + \frac{\arctan2(d_z, d_x)}{2\pi}, v = 0.5 + \frac{\arcsin(d_y)}{\pi} \quad (52)$$

Hence, the UV-mapping process can be described in simple three stage approach where the mesh is first unwrapped, a texture is created and applied to the respective face of the polyon.

- **Snell's Law**, also known as the **Law of Refraction**, describes the relationship between the angles of incidence and refraction for a light wave passing through the interface between two media with different refractive indices. It is named after Dutch mathematician Willebrord Snellius, who formulated the law in 1621.

Snell's Law states that the ratio of the sine of the angle of incidence (θ_1) to the sine of the angle of refraction (θ_2) is equal to the ratio of the refractive indices of the two media:

$$\eta_1 * \sin \theta_1 = \eta_2 * \sin \theta_2 \quad (53)$$

where η_1, η_2 are the refractive indices of the first and second media, respectively. The refractive index of a medium is a measure of how much the speed of light is reduced inside the medium compared to its speed in a vacuum. θ_1 is the angle of incidence, which is the angle between the incident light ray and the normal (a line perpendicular to the interface) at the point where the light ray enters the second medium. θ_2 is the angle of refraction, which is the angle between the refracted light ray and the normal at the point where the light ray enters the second medium. Snell's Law is a fundamental principle in optics and is used to explain various phenomena such as refraction, total internal reflection, and the formation of images by lenses. It is also employed in the design and analysis of optical systems, including cameras, microscopes, and fiber optics.

- **Fresnel's Law**, also known as Fresnel's Equations, describes the behavior of light when it encounters the interface between two different media, such as air and glass. It is named after French physicist Augustin-Jean Fresnel, who developed the equations in the early 19th century. Fresnel's Law helps explain how light is reflected and transmitted (refracted) at the interface between two media with different refractive indices. These equations are

fundamental to understanding various optical phenomena and are used in the design and analysis of optical systems.

Fresnel's Law consists of two equations that describe the reflection (eqn 54) and transmission (eqn 55) coefficients for the parallel (p) and perpendicular (s) components of the electric field of an electromagnetic wave as described by the eqn 56.

- **Reflection coefficients:**

$$\begin{aligned} R_s &= \left(\frac{\eta_1 * \cos \theta_1 - \eta_2 * \cos \theta_2}{\eta_1 * \cos \theta_1 + \eta_2 * \cos \theta_2} \right)^2 \\ R_p &= \left(\frac{\eta_1 * \cos \theta_2 - \eta_2 * \cos \theta_1}{\eta_1 * \cos \theta_2 + \eta_2 * \cos \theta_1} \right)^2 \end{aligned} \quad (54)$$

- **Transmission coefficients:**

$$\begin{aligned} T_s &= 1 - R_s \\ T_p &= 1 - R_p \end{aligned} \quad (55)$$

The total reflection and transmission coefficients can be computed as the average of the parallel (p) and perpendicular (s) components:

$$\begin{aligned} R &= (R_s + R_p)/2 \\ T &= (T_s + T_p)/2 \end{aligned} \quad (56)$$

These coefficients represent the fraction of the incident light that is reflected and transmitted at the interface between the two media.

Fresnel's Law is essential for understanding various optical phenomena such as reflection, refraction, total internal reflection, and the behavior of polarized light. It is also used in the design and analysis of optical systems, including lenses, mirrors, coatings, and filters.

(Work in Progress...)

3 History of Neural Rendering and NeRFs

The inception of works in Novel View Synthesis can be traced back to early research in computer graphics and computer vision, where researchers started to explore the use of 3D models and rendering which were later adapted for synthesizing novel views of a scene. One such work for rendering “Hierarchical geometric models for visible surface algorithms” [36] allowed for more efficient visibility determination and rendering of complex scenes. Although this paper focused on rendering and not on novel view synthesis, the concepts and techniques presented later had a significant impact on the development of model-based novel view synthesis. A work [66] that was influence by this early work introduced an IBR (image-based rendering) technique that used a hierarchical data structure to represent the scene radiance as a function of position and direction efficiently.

It leveraged the idea of hierarchical geometric models to render efficiently and determine visibility. It allowed for the synthesis of novel views from a set of input images without requiring explicit 3D geometry, making it an early powerful method for view synthesis and image-based rendering.

Some attempts of Model-free Novel View Synthesis from Images Rendering date back to 1990s during the time when the foundational book “Digital Image Warping” [231] came out, providing a comprehensive overview of image warping and morphing techniques. This book presented various image transformation techniques, including geometric transformations, spatial transformations, and image resampling. It covered the mathematical foundations and algorithms behind all these techniques, such as coordinate transformations, interpolation methods, and sampling theory. The concepts presented in this book have laid the groundwork for various novel view synthesis methods, as they require image warping to generate new views from input images. By offering both theoretical background and practical examples, “Digital Image Warping” has served as a valuable resource for researchers and practitioners in computer graphics, image processing, and computer vision, influencing the development of numerous techniques and algorithms.

The book “Three-dimensional computer vision: a geometric viewpoint” [57] introduced the geometric aspects of 3D computer vision comprehensively by building on mathematical foundations of projective geometry, camera models, and estimation techniques essential for many computer vision tasks. It further presented various methods for recovering the 3D structure of a scene from images, such as stereo vision, SfM, and SfS (shape from shading). The geometric concepts and techniques it covered became the basis of many view synthesis algorithms. By providing a thorough understanding of the geometric principles underlying 3D computer vision, it enabled researchers to develop more advanced and accurate methods for tasks such as novel view synthesis, scene reconstruction, and object recognition. Both these books contributed vitally to the field of computer vision and novel view synthesis by providing foundational knowledge on image warping and 3D geometry. They influenced and progressed the development of numerous techniques and algorithms in computer vision, laying the groundwork for researchers to explore more advanced and sophisticated methods for novel view synthesis and related tasks.

The “Digital Image Warping” [231] delved into practical aspects and applications of image warping, such as image registration, image mosaicking, and image morphing. Later works like Feature-based Image Metamorphosis [12], View Interpolation for Image Synthesis [29], Image metamorphosis using snakes and free-form deformations [120], Representation of scenes from collections of images [116], and View Morphing [190], were heavily inspired by this work. At that time, Feature-based Image Metamorphosis had been used to transform one image into another by smoothly interpolating the features of the source and target images. The metamorphosis process involves establishing correspondences between key features in the source and target images, interpolating the geometric and color information, and blending the two images over time to create a

smooth transition. Although it wasn't a direct technique for novel view synthesis, it was applied to this task with minor modifications such as capturing multiple images of the scene, estimation of camera parameters (SfM or Bundle Adjustment can be used), Identifying feature correspondences (SIFT, SURF, or ORB), estimating Novel views (interpolating the camera parameters of the input images), warping of input images (warping process leverage techniques like piecewise affine warping or thin-plate splines), and finally blending warped images (weighted blending).

An image-based rendering technique called view interpolation was introduced in “View Interpolation for Image Synthesis” [29] which allowed for the generation of novel views of a scene from a sparse set of input images. The method worked by estimating the depth information between the input images, followed by warping the images to the desired viewpoint, and finally combining the warped images using a blending technique. This approach reduced the need for explicit 3D models and enables the synthesis of novel views with relatively simple computations. Later, “Representation of scenes from collections of images” [116] proposed a technique to construct a 3D representation of a scene from a collection of images taken from different viewpoints. It involved estimating the geometry of the scene using stereo vision techniques, constructing a 3D mesh based on the depth information, and using image warping to project the textures onto the mesh. The scene representation would then be used for various applications, such as view synthesis or object recognition. “Image metamorphosis using snakes and free-form deformations” [120] shared some similarities with “Digital Image Warping” and presented an alternative method for image morphing to combines snakes (active contours) and free-form deformations to produce smooth and visually appealing transitions between images. It involved using snakes (instead of features) to establish correspondences between the source and target images and using free-form deformations to interpolate the features and warp the images. It addressed some of the limitations of traditional feature-based morphing techniques and improves the quality of the morphing results. “View Morphing” [190] is an image-based rendering technique that generates novel views of a scene by interpolating between two or more input images. The approach involves estimating the scene geometry, warping the input images to align them with the desired viewpoint, and blending the images to create a smooth transition between the viewpoints. The technique is able to handle occlusions and disocclusions in the scene and can be used for various applications, such as view synthesis, image mosaicking, and video summarization.

“3-D scene representation as a collection of images” proposed to use a collection of images taken from different viewpoints, along with their camera parameters, to represent a 3D scene. They develop a methodology for registering these images in a common coordinate system and use this registration to synthesize novel views of the scene by interpolating between the input images. The approach is image-based, as it does not rely on explicit 3D geometry but rather uses the input images and their associated camera parameters to synthesize new views of the scene. By focusing on the input images and their registration, this

method demonstrates the potential of image-based representations for 3D scene understanding and view synthesis tasks without the need for constructing detailed 3D models. The “Light Field Rendering” [122] introduced the concept of light field rendering, a technique for synthesizing novel views of a scene using a dense set of input images. These light fields represent the radiance of a scene as a function of position and direction and can be thought of as a 4D function. The paper described how to efficiently sample, reconstruct, and render the light field using a set of 2D images, allowing for the synthesis of new views without the need for explicit 3D geometry. Further, research such as “Physically-Valid View Synthesis by Image Interpolation” [189] presented a view synthesis technique that uses image interpolation to create novel views of a scene. It was based on the idea of “view morphing,” which combined two or more input images to synthesize a new view using epipolar constraints. The paper discussed a method that took into account the geometric relationships between the input images, the camera positions, and the scene structure, ensuring that the synthesized views are physically valid. This research on Morphing has later also been used in different research constituting different surfaces and objects [68].

Proceedings such as “Image-based Modeling and Rendering” [45] have hence been influential contributors for the shift of focus from model-free novel view synthesis approaches to model-based approaches. The talk surveyed different image-based rendering techniques, which included both model-based and image-based methods for synthesizing novel views of a scene. The authors discussed the advantages and limitations of both approaches, highlighting the need for more accurate and efficient algorithms for handling 3D data in novel view synthesis. The traction gained from such keynote talks gained traction and led to the development in the field of model-based novel view synthesis. Further, it was also driven by the following advantages which accumulated over time.

1. **Availability of 3D data:** With advances in 3D scanning technologies and the increasing availability of 3D models, researchers have been able to leverage this data to create more accurate and realistic novel view synthesis algorithms.
2. **Advances in computer graphics and computational power:** The development of sophisticated rendering techniques and the increase in computational power have made it possible to work with complex 3D models and generate realistic novel views in real-time.
3. **Improved algorithms for 3D reconstruction:** Researchers have developed more accurate and efficient algorithms for recovering the 3D structure of a scene from images, making it possible to use model-based approaches for novel view synthesis.
4. **Handling occlusions:** Model-based approaches can better handle occlusions and disocclusions in the scene, as they have explicit 3D information about the scene geometry.

Later attempts in Neural Rendering usually involved lengthy pipeline operations for creating virtual models of the real scenes. These operations start with a scanning process (using camera images) to capture the photometric properties,

and raw scene geometry is captured using depth scanners or dense stereo matching. The Raw Scene Geometry capture is often noisy and provides incomplete point cloud data. This incomplete data can further be processed by applying surface reconstruction and meshing approaches. Particularly, with the context of 3D reconstruction with commodity RGB-D sensors, researchers have made significant progress enabling robust tracking [93,156,35,228,43], and large scale 3D scene capture [27,161,253]. Then, the material and texture estimation operations are leveraged to determine the photometric properties from the obtained mesh of surface fragments and store them in different data structures like texture maps [16], bump maps [15], view-dependent textures [46], and surface light fields [232]. Basically, textures are alternative that mitigate the missing spatial resolution problem where finding the consistent UV-mapping becomes vital as geometry and color are decoupled. To compensate for wrong geometry and camera drift, some approaches are based on finding non-rigid warps [268,87].

Lastly, the computationally-heavy rendering operations (ray-tracing/radiance transfer estimation) generate photo-realistic views of the modeled scene. An advantage of non-rigid warps discussed earlier is that 3D reconstructed content can be visualized by standard rendering techniques and approach directly generalizes to 4D capture as long as it can be reliably tracked [155,52,90]. The reconstructions discussed earlier can be used to synthesize images from novel viewpoints. This pipeline of operations has been constructed and polished by research in computer graphics society for several years. Despite that this pipeline generates highly realistic results under controlled settings, several operational stages of this pipeline and consequently the entire pipeline remains brittle until the addition of the domain of Machine Learning. This augmentation has hence encouraged people from Machine Learning, Computer Vision, and Computer Graphic society to come together and make this pipeline more robust. But, the imperfections in reconstruction directly appear when translating the embeddings to visual artifacts in renderings, that became the major hurdle in 2020s to make content creation from real-world accessible.

With the growth of Neural Networks and the inception of CNN-based architectures like AlexNet [115], VGG-Net [193], and ResNets [79], the focus shifted for Machine Learning techniques to permeate the domain of Computer Vision and Computer Graphics. This is evident by the following research literature which leverages CNNs exhaustively. “View Synthesis by Appearance Flow” [270] introduced an end-to-end trainable CNN model that learned to synthesize target views by predicting appearance flows, which were then used to warp input images, allowing for view synthesis without explicitly estimating 3D geometry. “Deep View Morphing” [98] proposed a deep learning-based approach to view morphing, leveraging CNNs to learn and predict dense correspondence maps between source and target images, enabling the generation of high-quality novel views. Habtegebrial et al., in “Fast View Synthesis with Deep Stereo Vision” [76], developed a deep learning-based method for view synthesis that combined stereo vision techniques with a CNN, aiming to produce accurate novel views with reduced computational complexity compared to other learning-based approaches.

Lastly, “InLoc: Indoor Visual Localization with Dense Matching and View Synthesis” [208] by Taira et al. presented a large-scale indoor localization system that matched dense local features between a query image and a database of 3D point clouds [229], utilizing view synthesis to improve the matching performance and achieve accurate and robust localization in challenging indoor environments. This was an application of stereo vision models in indoor 3D scene reconstruction.

An alternative direction is to fill in the missing content by leveraging coarse geometry proxy based on high-resolution 2D textures [87]. **PBG (Point-based Graphics)** approaches for rendering and re-rendering pipelines use a collection of points or unconnected disks (surfels) for geometry modeling instead of Surface Mesh Estimation [123,70,69,111]. Whereas, the IBR (Image-Based Rendering) techniques [142,190,66,122] warp the Camera Coordinate System using coarse approximations (low-frequency embeddings in case of functional representation) of scene geometry to obtain the photorealistic views, where the 3D geometry proxy is only used to select suitable views for cross-projection and view-dependant blending [83,19,21,264,81]. Finally, the deep neural rendering methods [92,153,24,20,80] replace the physics-based rendering with generative Neural Network to rectify some of the mistakes of rendering network. One such method that has gained traction from Computer Vision society is what is the primary focus of this survey—NeRF.

LLFF (Local Light Field Fusion) (example depicted in fig. 4a) is one prior work done by Mildenhall et al [147], some of whose authors were also contributors for the work under focus in this survey. LLFFs leverage MPI (Multiplane Image) scene representations [269] for view synthesis from an irregular grid of sampled views for expanding each sampled view into a local light fields [122] and then renders novel views by fusing these fields. LLFFs achieve perceptual quality similar to Nyquist Rate Sampling (with $4000 \times$ less views) by creating a bound for real world scenes creation/rendering which specifies the density of sample views for a given scene by extending the traditional plenoptic sampling [22] theory. Here, the LLFFs result in a quadratic decrease in the number of input views based on the number of predicted planes for each layered scene representation and still maintain the Nyquist level performance by specializing to the subset of natural scenes. Further, this work was corroborated with a smartphone app and web-cum-desktop application for capturing and viewing respectively.

This architecture is based on the idea that prescriptive sampling is vital for useful IBR-algorithms. Hence, it extends the prior plenoptic sampling theory to decrease the dense sampling requirements for traditional light field rendering by decomposing a scene into D depth ranges and individual-range-based light field samples. This allows the camera sampling interval to be increased by a factor of D and the light field spectrum emitted by a scene within each depth range to lie within a densely packed shape that is D times tighter than the full scene’s double-wedge spectrum as depicted in fig. 3. This synthesis pipeline also leaves a possibility for it to be used in the future light field hardwares to reduce required cameras. As discussed by Chai et al [22], the Light Field’s Fourier support

(without any occlusions and Lambertian effects [254]) lies inside double-wedge shape within scene depths limit $[z_{min}, z_{max}]$ as given by the figure 3, whereas occlusion expands the limits of this shape in the light field as the occluder acts as convolver due to the kernel (lying on the line corresponding to the occluder's depths) with farther scene content. Further, Fourier support of the light field gets limited due to the closest occluder which convolves the furthest scene content's line resulting in a parallelogram-like-shape as depicted in fig 3. This can only be half packed when compared to double-wedge shape and hence required maximum camera sampling interval Δ_u for an occluded light field and is given by eqn. 57

$$\Delta_u \leq \frac{1}{2K_x f \left(\frac{1}{z_{min}} - \frac{1}{z_{max}} \right)}, \quad (57)$$

In the continuous light field B_x and camera spatial resolution Δ_x , K_x is maximum spatial frequency in sampled light field. It is given by the eqn. 58

$$K_x = \min \left(B_x, \frac{1}{2\Delta_x} \right) \quad (58)$$

LLFFs render novel views from an MPI scene representation (that consist of a set of evenly disparity sampled fronto-parallel RGB α planes from view frustum of reference camera) from local-neighborhood's continuous valued camera poses by alpha compositing the color rays into the novel views with the “over” operation [172]. This is similar to the early work by Lacroute et al [118] and can be considered an encoding of a local light field much like light field displays [226,227]. Further, it alpha composite the depth range fields or shield fields [119] from back to front for computing the full scene light field to decrease the Sampling Rate for occlusions. These fields plays a vital role in handling occlusions as the Fourier Spectra Union have less degree of confidence when compared to the original occluded light fields as depicted in fig 3. Each layer-by-layer alpha compositing step increases the Fourier support by convolving previously accumulated light-field's spectrum with occluded depth layer spectrum. The width of Fourier Spectrum Support Parallelogram for the reconstructed light field for each occluded depth range light field will enjoy full Fourier Support width and is given and illustrate by eqn. 59 and fig 3, respectively.

$$\frac{2K_x f \left(\frac{1}{z_{min}} - \frac{1}{z_{max}} \right)}{D}, \quad (59)$$

Hence, LLFFs are able to extend the layered plenoptic sampling framework and handle occlusions using depth maps and increasing the required sampling interval by a factor of D and is given by the eqn. 60.

$$\Delta_u \leq \frac{D}{2K_x f \left(\frac{1}{z_{min}} - \frac{1}{z_{max}} \right)}, \quad (60)$$

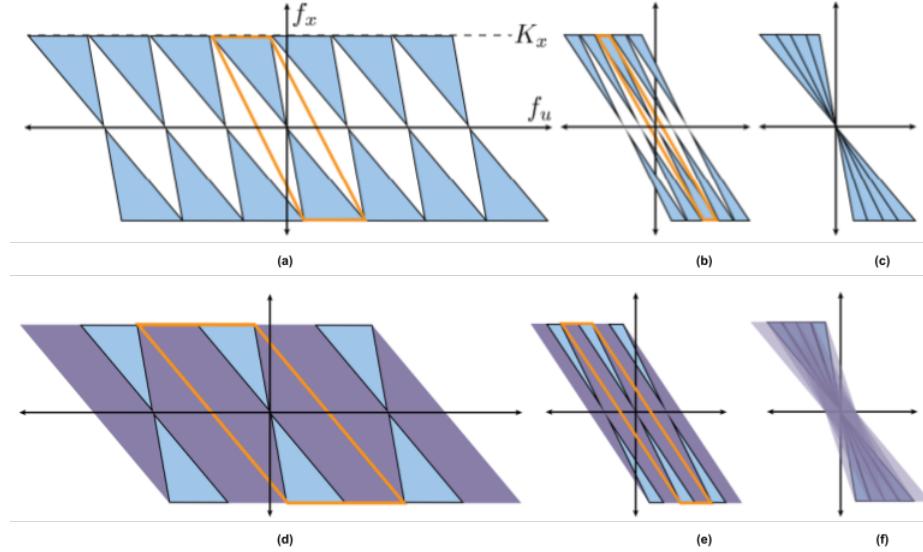


Fig. 3: Traditional Plenoptic Sampling without occlusions [22] in (a) where the double-wedge (in blue) encloses the Fourier support of a light field without occlusions. Here, the double-wedge width is used to calculate Nyquist Rate Sampling, that determines by minimum and maximum scene depths $[z_{min}, z_{max}]$ and K_x (max spatial frequency). In (b), the light field is split into D non-overlapping layers and Nyquist Sampling Rate decreases by a factor of D and (c) represents that without occlusions, the full light field spectrum is sum of the spectra from each layer. In LLFF [147] extends this to mitigate occlusions as continuous light fields are reconstructed from MPIs. In (d), occlusions expand the Fourier supports to — gm (where blue represents support without occlusions and purple regions depict the occlusion expanding the Fourier support) doubling the Nyquist Sampling Rate. Further, (e) depicts that separate reconstruction of the light field for D layers decrease the Nyquist Sampling Rate by a factor of D whereas (f) depicts the case that full light spectrum cannot be reconstructed by summation over individual layer spectra because their support union is smaller than support union in (a). Hence, the need for alpha compositing individual light layers from back to front in the primal domain to compute the full light field. Figures used from LLFF original paper [147]

In addition to this, LLFFs also introduced an additional caveat for the scene's bounding volume to be within a frustums of atleast two neighboring sampled views by introducing the concept of finite field of view instead of infinite field of view [22,254]. This bound is given by the camera sampling interval 61 and the constrain given in the eqn 62.

$$\Delta_u \leq \frac{W\Delta_x z_{min}}{2f}, \quad (61)$$

$$\Delta_u \leq \min \left(\frac{D}{2K_x f \left(\frac{1}{z_{min}} - \frac{1}{z_{max}} \right)}, \frac{W\Delta_x z_{min}}{2f} \right) \quad (62)$$

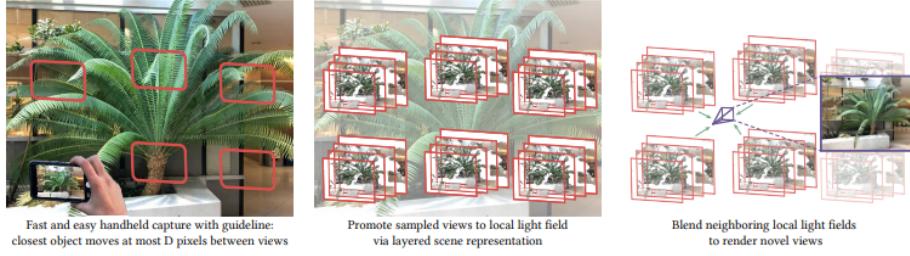
Now, if we set $z_{max} = \infty$ for the traditional assumption and set $K_x = \frac{1}{2\Delta_x}$ for allowing maximum representable frequency, then we have eqn 63.

$$\frac{\Delta_u}{\Delta_x z_{min}} = d_{max} \leq \min \left(D, \frac{W}{2} \right) \quad (63)$$

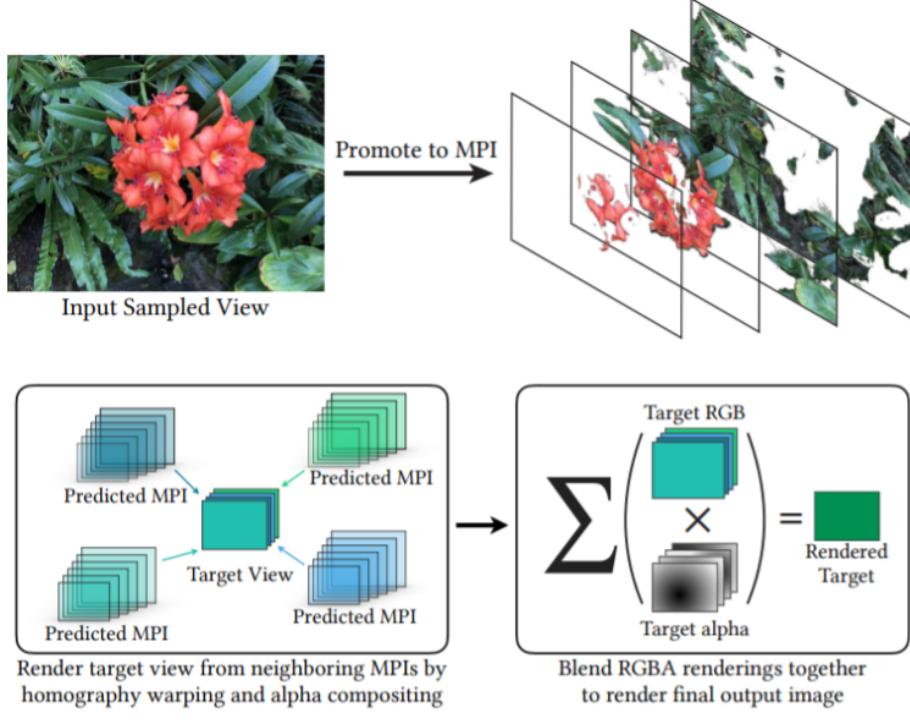
where d_{max} is the maximum pixel density of any scene between adjacent scenes. Hence, the maximum density i.e. d_{max} must be less than $\min(D, W/2)$ and it reduces to the Nyquist bound when $D = 1$. Hence, this promotion to D depth layers decreases the required sampling rate by a factor of D until the required field of view overlaps for estimation of stereo geometry. Now, further as we know that any scene can be a function of two viewing directions hence this sampling rate compounds to D^2 sampling reduction.

LLFFs practically synthesize new views from set of input images and camera poses by first using a CNN to promote each captured input image to an MPI and then reconstructing novel views by blending the rendered light field from the nearby MPIS as depicted in fig 4a and fig 4b. Firstly, 5 images of a scene are taken (1 reference image and 4 nearest neighbors in 3D space) and reprojected to D depth planes and sampled linearly within reference view frustum to get $5 H \times W \times 3$ PSVs (plane sweep volumes). LLFFs employ a 3D CNN for taking 5 PSVs as inputs to concatenate them along channel dimension which results in opacity α for each MPI pixel (x, y, d) and 5 color selection weights sum to unity at each MPI. This parametrization is leveraged instead of fg-bg parameterization [269] as it prevents an MPI from incorporating content occluded from the reference view but visible in other input views. 3D convolutions enables the architecture to predict MPIS with D number of variable planes by involving convolutions across the height, width, and depth instead of fully 2D convolutions. This enables to jointly choose disparity and view sampling densities to satisfy the eqn. 63

LLFFs reconstruct interpolated views as a weighted combination of renderings from multiple nearby MPIS. This process combines local light field approximations to a near-plane spanning light field and a far-planed determines by the input views' field-of-view. This allows to render new biew path without constrains for 3D translation and rotation as in standard light field rendering. LLFFs consider the accumulated alpha values as vital for each MPI rendering



(a) A step-by-step representation of Multi-Plane Image Compositing in Local Light Field Fusion by Mildenhall et al [147]



(b) LLFF (Local Light Field Fusion) [147]: Promotion to MPI Scene Representation for each input image with D RGB α planes at regularly sampled depths in the input view's camera frustum. Here, each MPI can be used to render continuously-valued novel views using alpha compositing of color value along rays within a local neighborhood within a novel view's (imaginary) camera.

when blending to allow MPI rendering to “fill in” content occluded from camera views. LLFFs prediction network leverage RGB image sets C_k with camera poses p_k and produce MPI sets M_k for each image. Each RGB α MPI plane was homographically warped into target pose p_t 's reference frame using a predicted MPI M_k and then alpha composited together from back to front. All these steps lead

to an RGB image ($C_{t,k}$) and an alpha image ($\alpha_{t,k}$) where the output is rendered at pose p_t using MPI pose p_k . Further, the final RGB output C_t is obtained by blending rendered images $C_{t,k}$ from multiple MPIS as one MPI does not contain all content visible from the new camera pose as illustrated in fig. 4b and given by eqn 64.

$$C_t = \frac{\sum_k w_{t,k} \alpha_{t,k} C_{t,k}}{\sum_k w_{t,k} \alpha_{t,k}}, \quad (64)$$

where the method leverages scalar blending weights $w_{t,k}$ (can be smooth filter) that are modulated by the corresponding accumulated alpha images ($\alpha_{t,k}$) and normalized so that $\alpha = 1$. For instance, modulating the blending weights by referring to eqn. 64 accumulates alpha values prevents artifacts in C_t . Bilinear Interpolation from the 4 nearest MPIS is leveraged for regular grid of data rather than the ideal sinc function interpolation for extremely limited number of sampled views and efficiency. All five MPIS are used for irregularly sampled data and $w_{t,k} \propto \exp(-\gamma \ell(p_t, p_k))$. Here, ℓ is L^2 distance between translation vectors of poses p_t and p_k , and γ is given by eqn 65.

$$\gamma = \frac{f}{Dz_{min}}, \quad (65)$$

where f is the focal length and z_{min} is the minimum distance to the scene. This strategy for MPI neighborhood blending is particularly efficient for non-Lambertian renderings. Virtual apparent depth of a specularity changes with viewpoint [207] for curved surfaces and hence, specularities can appear as curves in light fields' epipolar slices and lines can appear for diffuse points. Here, each of the MPI represents a specularity for a local range of views by placing the specularity at a single virtual depth. This approach leverages real world and synthetic dataset of natural scenes [200,175,186,187]. These results that quantitatively validate and demonstrate that LLFFs match the perceptual quality of Nyquist rate Sampling while leveraging $\approx 4000\times$ fewer input images has been demonstrated for over 60 diverse real world scenes and this work paved way for future IBR algorithms.

Further, there have been many countless works from 2015–2020 on representing scenes as unstructured or weakly structured feature representations [210,233,55] but the first work to implicitly learn the function for scene as continuous representation was presented in **SRNs (Scene Representation Networks)**.

SRNs (proposed by Sitzmann et al; depicted in fig 5) encapsulated both the geometry and the appearance of a scene by leveraging camera poses [197]. This training was performed end-to-end without explicit supervision in 3D geometry, purely from a set of posed 2D images. These networks played an important role in constructing the basis for NeRFs and gaining better evaluations for scene reconstructions. All this was done using Differential Ray Marching using LSTM (Long-Short Term Memory networks). Here, the objective was to combine multiple different disciplines such as Geometric Deep Learning, Neural Scene Representation, and Neural Image Synthesis, and Differentiable Ray Marching to

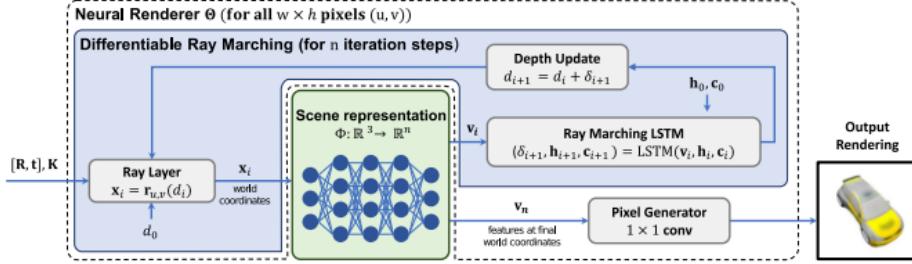


Fig. 5: A model schematic representation of SRNs (Scene Representation Networks) by Sitzmann et al [197]

represent the scene as a continuous function ϕ responsible for matching a feature representation \mathbf{v} of learned scene properties for a particular spatial location:

$$\phi = \mathbb{R}^3 \rightarrow \mathbb{R}^n, \mathbf{x} \mapsto \phi(\mathbf{x}) = \mathbf{v} \quad (66)$$

where \mathbf{v} may be a feature vector that encodes visual information related to the scene or higher order information such as signed distance of \mathbf{x} to closest surface. Hence, this work became one of the first works to formulate Scenes as continuous formulation. Earlier works used to store information in a discretized manner such as Voxel Grids' discretization in \mathbb{R}^3 [141,105,157,271,215,196] and Point Clouds [174,91,128,145] may have sparsely discretized points anywhere in the space \mathbb{R}^3 . Here, the concepts of SRNs was the first one to encode all the information of a scene in a function which was represented by an MLP (Multi-layer Perceptron) which limited the scene information to the capacity of the MLP. Further, SRNs perform Neural Rendering θ by mapping scene representation ϕ , intrinsic \mathbf{K} , and extrinsic \mathbf{E} to an image I as given by the following equation.

$$\theta : \chi \times \mathbb{R}^{3 \times 4} \times \mathbb{R}^{3 \times 3} \rightarrow \mathbb{R}^{H \times W \times 3}, (\phi, \mathbf{E}, \mathbf{K}) \mapsto \theta(\phi, \mathbf{E}, \mathbf{K}) = I \quad (67)$$

where χ represents universal space for functions. Here, these networks faced a disadvantage at implicitly storing the geometry in the MLPs. Hence, a learned neural ray marching algorithms was used in this approach and a pixel generator network was proposed to map geometry/appearance features to colors. The differentiable Ray Marching Algorithm takes inspiration from Classic Sphere Tracing algorithm [77] where it solves the eqn 68 by starting at a distance close to the camera and learning the step distance and stepping such that ray intersects the geometry (where each step has signed distance length to the closest surface point of the scene) [164].

$$\begin{aligned} & \arg \min d \\ & \text{s.t. } \mathbf{r}_{u,v}(d) \in \Omega, d > 0 \end{aligned} \quad (68)$$

where Ω refers to all the points on the surface of the scene. The step length is learned using a RM-LSTM (Ray Marching Long-Short-Term Memory) [85].

Further, the authors behind SRN also leverage a pixel generator architecture which is based on 1×1 convolutions instead of 2D convolutions. This is because the 2D convolutions may hinder multi-view consistency in case if they are capturing the context of local area which is disrupted when bringing the camera closer to another adjacent local area of a scene but it also discounts the ability of these networks to work in a smaller memory budget. SRNs further generalize across different scenes by jointly optimizing the following objectives by leveraging stochastic gradient descent.

$$\begin{aligned} \arg \min_{\{\theta, \Psi, \{\mathbf{z}_j\}_{j=1}^m\}} & \sum_{j=1}^M \sum_{i=1}^N \underbrace{\|\Theta_\theta(\Phi_{\psi(\mathbf{z}_j)}, \mathbf{E}_i^j, \mathbf{K}_i^j)\|_2^2}_{\mathcal{L}_{\text{img}}} \\ & + \underbrace{\lambda_{\text{depth}} \|\min(\mathbf{d}_{i, \text{final}}^j, 0)\|_2^2}_{\mathcal{L}_{\text{depth}}} \\ & + \underbrace{\lambda_{\text{latent}} \|\mathbf{z}_j\|_2^2}_{\mathcal{L}_{\text{latent}}} \end{aligned} \quad (69)$$

Here, the first loss was an l_2 loss which listed the similarity of the constructed or rendered image based on ground-truth of that particular path of area in a scene, whereas $\mathcal{L}_{\text{depth}}$ is the constraint in eq. 68, and finally \mathbf{z}_j on Gaussian Distribution priors. This equation can also further be modified for the purpose of Few-Shot Reconstruction. The scope of experimentation is beyond the scope of this paper, so, the readers are further advised to study the paper in detail to get a better idea of the implementation details.

Lombardi et al [135] presented **NV (Neural Volumes)** as a way to learn both the geometry and appearance variations simultaneously that led to better generalization across viewpoints. Their work has been depicted in figure 6. To address the issue of terminating in poor local minima when optimizing over a mesh-based representation using gradient-based optimization, the authors used a volumetric representation as a function of color and opacity at each position in 3D space. Hence, helping in mitigating problems such as accountability for reflectance variability or tracking topological evolution in dynamic media in motion capture systems [78,184,4,241]. Further, during the optimization of this encoder-decoder model, semi-transparent representation of geometry disperses gradient information along the ray of integration, effectively widening the basin of convergence, enabling the discovery of good solutions. To overcome the limitation of using intensive memory for the sparse voxel grids, the authors worked with a warping technique that indirectly escapes the restrictions imposed by a regular voxel grid structure, allowing the learning algorithm to make the best use of available memory. Further, since the voxel grid structure isn't used directly there were less grid-based artifacts in this approach.

Much like SRNs, the NV method is based on two main parts of the architecture which consist of an encoder-decoder network that converges after mapping image to a 3D volume $\mathbf{V}(\mathbf{x})$, and differentiable ray marching for rendering image

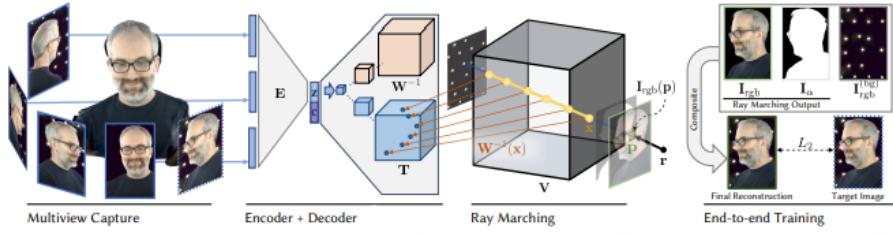


Fig. 6: A block representation of NVs (Neural Volumes) by Lombardi et al [135]

from \mathbf{V} with a given camera parameters. This is analogous to an Auto-Encoder with fixed function volume rendering operation for the final layer. Here, the model maps from 3D location space to the 3D positions, $\mathbf{x} \in \mathbb{R}$ as stated in eqn. 70 given below.

$$\mathbf{V} : \mathbb{R}^3 \rightarrow \mathbb{R}^4, \mathbf{V}(\mathbf{x}) = (\mathbf{V}_{\text{rgb}}(\mathbf{x}), \mathbf{V}_\alpha(\mathbf{x})), \quad (70)$$

where $\mathbf{V}_{\text{rgb}}(\mathbf{x}) \in \mathbb{R}^3$ and $\mathbf{V}_\alpha(\mathbf{x})$ are the color and opacity (range $[0, \infty]$) at the 3D coordinate \mathbf{x} . This enables the model to learn soft discrete volume representation that allowed gradients to flow backward easily and model fine and transparent structures (like hair and smoke).

NV also discusses the potential of leveraging only 3 images from 3 orthogonally placed cameras to achieve maximize convergence while downsampling an image by a factor of 8. Here, the Encoder is a Convolutional Variational encoder whereas the authors leverage an MLP for Decoder (which demands prohibitive memory requirement to produce reconstructions with higher quality). Hence, Lombardi et al [135] model the volume function as a discrete 3D grid of voxels hence, gaining the output tensor in the form of discretized voxels. The authors here optimized the decoder as a function $S(\mathbf{x}; \mathbf{Y}) : \mathbb{R}^3 \rightarrow \mathbb{R}^C$ to samples from the grid \mathbf{Y} by scaling continuous values in the range $[-1, 1]$ to $[1, D]$ in each dimension followed by trilinear interpolation. Hence, for general case of 3D cube with center at \mathbf{x}_0 with W sides is given by the eqn. 71.

$$\mathbf{V}(\mathbf{x}; \mathbf{z}) = S \cdot \left(\frac{\mathbf{x} - \mathbf{x}_0}{W/2}; \mathbf{g}(\mathbf{z}) \right), \quad (71)$$

Note that here the $\mathbf{g}(\mathbf{z})$ represents the MLP used for decoding of the Voxel Grid. The authors also discussed how they can translate a similar architecture with $1 \times 1 \times 1$ cube with 1024 channel for duplicating the function generated by the MLP. Later, a softplus activation to restrain the voxels to positive values rather than negative values that might occur after decoing. Further, a warping function is applied to the voxel decoder to mitigate the wasteful behavior of the Voxel Grids (also solved by Octree-based approaches [181] but they are complex and require object distribution to be known *a priori*). Hence, the authors used an inverse warping formulation as given in the equation 72 below.

$$\mathbf{W}^{-1}(\mathbf{x}) \rightarrow \mathbf{y}, \text{ where } \mathbf{x}, \mathbf{y} \in \mathbb{R}^3, \quad (72)$$

This helps to map points from a warp volume $\mathbf{W}^{-1}(\mathbf{x})$ to a template volume $\mathbf{T}(\mathbf{x})$ where both volumes are decoded from the dynamic latent code \mathbf{z} . Hence, in the eqn. 72, \mathbf{x} is a 3D coordinate in the output space and \mathbf{y} is the corresponding 3D coordinate in RGB α template volume. Hence, the mapping can be depicted by using the eqn. 73.

$$\mathbf{V}_{\text{RGB}\alpha}(\mathbf{x}) = \mathbf{T}_{\text{RGB}\alpha}(\mathbf{W}^{-1}(\mathbf{x})). \quad (73)$$

These warp fields were developed as a mixture of Affine Warps to produce an affine warp field without encountering any overfitting issues or any non-linear bending. This formulation has been depicted by the eqn. 74–75 given below.

$$\mathbf{W}^{-1}(\mathbf{x}) = \sum_i \mathbf{A}_i(\mathbf{x}) a_i(\mathbf{x}), \quad (74)$$

$$\text{with } \mathbf{A}_i(\mathbf{x}) = \mathbf{R}_i(\mathbf{s}_i \circ (\mathbf{x} - \mathbf{t}_i)), \quad a_i(\mathbf{x}) = \frac{w_i(\mathbf{A}_i(\mathbf{x}))}{\sum_j w_i(\mathbf{A}_i(\mathbf{x}))}, \quad (75)$$

where $\mathbf{A}_i(\mathbf{x})$ is the i^{th} affine transformation, w_i is the weight of each warp, and \circ is the element-wise multiplication. Further, the authors also made the RGB decoder constrained by viewpoint conditions by given the normalized direction of the camera to the decoder along the encodings as an input. Further, the convolutions for the RGB and α component were disentangled as RGB values were conditioned on the viewpoint. Further, for the ray-tracing and End-to-End training this approach leverages Semi-Transparent Volume Rendering and a combination of Reconstruction Priors and Hyperparameter Tuning.

Here, the ray marching accumulates not only color but also opacity to model the occlusions and when opacity hits the value of 1, then no color is accumulated on the ray. Hence, the rendering process for this approach is as given by the eqn. 76–77.

$$\mathbf{I}_{\text{rgb}}(\mathbf{p}) = \int_{t_{min}}^{t_{max}} \mathbf{V}_{\text{rgb}}(\mathbf{r}_o + t\mathbf{r}_d) \frac{d\alpha(t)}{dt} dt, \quad (76)$$

$$\text{where } \alpha(t) = \min \left(\int_{t_{min}}^t \mathbf{V}_\alpha(\mathbf{r}_o + s\mathbf{r}_d) ds, 1 \right), \quad (77)$$

$$\text{and } \mathbf{r}_d = \frac{\mathbf{P}^{-1} \mathbf{p} - \mathbf{r}_o}{\|\mathbf{P}^{-1} \mathbf{p} - \mathbf{r}_o\|} \in \mathbb{R}^3, \quad (78)$$

where $\mathbf{I}_{\text{rgb}}(\mathbf{p})$ is the color at a pixel location in the camera coordinate system with center $\mathbf{r}_o \in \mathbb{R}^3$, ray direction is given by the eqn. 78 for the Image Coordination System-to-World Coordinate System.

Color calibration based on per-camera and per-channel bias b and gain g ensures that the reconstructed images is accounted for the slight differences in

overall intensity in the image. Further, a background estimation based on the eqn. 79

$$\widehat{\mathbf{I}}_{\text{rgb}}(\mathbf{p}) = (1 - \mathbf{I}_\alpha(\mathbf{p})) \mathbf{I}_{\text{rgb}}^{(\text{bg})}(\mathbf{p}) + (g\mathbf{I}_{\text{rgb}}(\mathbf{p}) + b), \quad (79)$$

where $\mathbf{I}_{\text{rgb}}^{(\text{bg})}$ and $\widehat{\mathbf{I}}_{\text{rgb}}$ are the background image extracted based on static or non-changing background, and final image, respectively. Further, NV introduces two priors in its architecture—total variation of the log voxel opacities, and beta distribution ($\text{Beta}(0.5, 0.5)$) on the final image opacity. These priors are demonstrated by the eqn. 80–81.

$$\mathcal{P}_{\text{var_lvoxop}}(\mathbf{V}_\alpha) = \frac{1}{N} \sum_x \lambda_{\text{var_lvoxop}} \left\| \frac{\partial}{\partial \mathbf{x}} \log \mathbf{V}_\alpha(\mathbf{x}) \right\| \quad (80)$$

$$\text{and } \mathcal{P}_\beta(\mathbf{I}_\alpha) = \frac{1}{P} \sum_{\mathbf{p}} \lambda_\beta [\log(\mathbf{I}_\alpha(\mathbf{p})) + \log(1 - \mathbf{I}_\alpha(\mathbf{p}))] \quad (81)$$

where \mathbf{p} , P , and \mathcal{P} represent a pixel in image, number of pixels, and the priors used. Further, the training objective of NV is based on the equation 82 where $\mathbf{I}_{\text{rgb}}^*$ is the ground truth image and $D_{KL}(\mathbf{z} || N(0, 1))$ is the KL divergence between the latent encoding \mathbf{z} and Bell Distribution used in a VAE (Variational Autoencoder).

$$\begin{aligned} l(\theta) = & \frac{1}{P} \sum_{\mathbf{p}} \left\| \widehat{\mathbf{I}}_{\text{rgb}}(\mathbf{p}) - \mathbf{I}_{\text{rgb}}^*(\mathbf{p}) \right\|^2 \\ & + \lambda_{\text{KL}} D_{\text{KL}}(\mathbf{z} || N(0, 1)) \\ & + \mathcal{P}_{\text{var_lvoxop}}(\mathbf{V}_\alpha) \\ & + \mathcal{P}_\beta(\mathbf{I}_\alpha). \end{aligned} \quad (82)$$

Various usecases and results for the same have been discussed in the paper. Further, we refrain from discussing the implementation, training, and hyperparameter details for this network architecture hence the reader is advised to read the paper for more comprehensive intuition about the author's approach.

Thies et al proposed **DNR (Deferred Neural Rendering)** [212] to leverage traditional graphics components and make those learnable using Neural Networks. The main idea behind this work is to use imperfect 3D content from photometric reconstructions with incomplete or otherwise noisy surface geometry to produce photo-realistic (re-renderings). The contribution here was the introduction of *Neural Textures* that were learned as feature maps from the scene capture process. These are stored similar to texture maps on top of 3D mesh proxies. High Dimensional Feature Maps contain more information (when compared to traditional texture maps) that were encoded in high dimensional feature maps and used deferred neural pipeline for interpretation. In this approach, both the Deferred Neural Renderer and Neural Textures were trained end-to-end which enabled synthesizing photo-realistic images with imperfections in the original 3D

content. This approach gave significant explicit control over generated output such as synthesizing temporally-consistent video re-renders of recorded 3D scenes in 3D space. This allowed Neural Textures to coherently re-render and manipulate existing video content in real-time static and dynamic environments. Further, this approach has been leveraged for several applications such as Novel View Synthesis, Scene Editing, Facial Reenactments. Moreover, this rendering that relied on geometry proxy (needs to be reconstructed in pre-processing step) was much faster than traditional reenactments for high-resolution outputs. A drawback of this approach was the retraining of neural textures for every object fed to the DNR. This pipeline allowed generalization to happen on 10 objects used, which suggested applications in the domain of transfer learning, segmentation, scene illumination, surface reflectance, novel shading, and lighting capabilities with usage of this pipeline. Further, this work had the capability to be extended beyond the traditional 2D to volumetric grids. The Average Precision for the Ground Truth proposal has been given in the eqn 83.

$$AP@n = \frac{1}{GTP} \sum_k^n P@k \times rel@k \quad (83)$$

Another Neural Rendering approach called **NPBG (Neural Point-Based Graphics)** (proposed by Aliev et al [2]) explores the idea of augmenting each point with a learnable neural descriptor that encodes local geometry and appearance. Authors claim that this approach results in compelling results for scenes scanned with standard RGB cameras and hand-held commodity RGB-D sensors even with the objects that are challenging for standard mesh-based rendering. The idea of NPBG combines the ideas behind IBR, NPG, and Neural Rendering discussed briefly earlier. It uses a Deep CNN (Convolutional Neural Network) to generate photorealistic scenes from novel viewpoints by leveraging Scene Geometry Raw Point-Clouds Representation. Latent Vector (here, Neural Descriptors) learning/estimation facilitates realistic rendering by describing both geometric and photometric properties of scenes in conjunction with learning of the rendering network. The fig. 7 gives an intuition into how NPBG works and how the rendering algorithm is jointly optimized while learning the Neural Descriptors. This approach formulates the problem of Neural Rendering with the point clouds ($P = \{p_1, p_2, \dots, p_N\}$) and M-dimensional Neural Descriptors ($\mathbb{D} = d_1, d_2, \dots, d_N$) given and the requirement for the renderer to obtain a new view with a camera \mathcal{C} (with extrinsic parameter and intrinsic parameters).

The authors suggest to outline the rendering process such that descriptors are used as pseudo-colors to project the point into the target view. Further, the rendering network is used to map the pseudo-colors to photorealistic RGB space. Formally, NPBG creates an M -channel raw-image of size $W \times H$, and for each point p_i projecting to (x, y) given by the eqn. 84

$$S(\mathbf{P}, \mathbf{D}, C)[[x], [y]] = d_i, \quad (84)$$

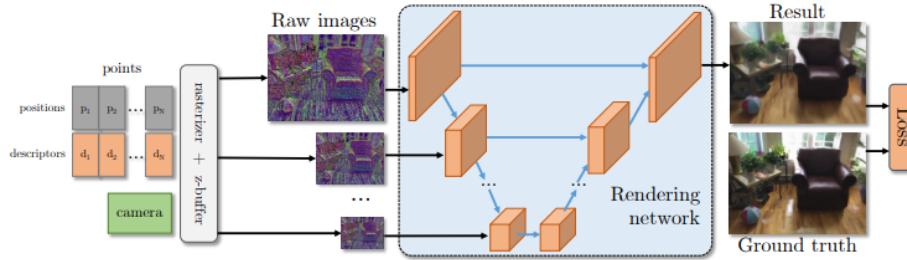


Fig. 7: A block diagram representation of NPBGs (Neural Point Based Graphics) by Aliev et al [2]

where $[a]$ is nearest integer for $a \in \mathbb{R}$. Further, this approach proposes to leverage a z-buffer to remove occluded points as several points can project onto the same pixel but the lack of topological information in point clouds can result in the *bleeding* problem (where occluded surfaces and background points could be seen through from front surface). This approach proposes to leverage multi-scale progressive rendering to avoid relying on choice of disk radius (important for *splatting* which uses a 3D elliptic disk with an estimated radius to replace each point). Here, a sequence of images $S[1], S[2], \dots, S[T]$, where the i-th image has the size of $\frac{W}{2^t} \times \frac{H}{2^t}$ is obtained by rendering a point cloud on a canvas pyramid of different resolution T times by performing simple point-cloud projection. Hence, $S[1]$ has the maximum details and is prone to the most surface bleeding whereas $S[T]$ suffers from least surface bleeding but has the minimum geometric details. Every other image in sequence ($S[2], S[3], \dots, S[T - 1]$) has different detail-bleeding tradeoffs. Now, all the raw sequence of images are mapped to three channel RGB image I using a U-Net [183] (with Gated Convolutions [252] to handle sparse inputs) based rendering network (\mathcal{R}_θ as given in the eqn, 85 below).

$$I(\mathbf{P}, \mathbf{D}, C, \theta) = \mathcal{R}_\theta(S[1](\mathbf{P}, \mathbf{D}, C), \dots, S[T](\mathbf{P}, \mathbf{D}, C)) \quad (85)$$

The, the raw images $S[i]$ are concatenated to the U-Net first block with corresponding resolutions (this coarse-to-fine mechanism and mipmapping [] are similar). Hence, rendering network provides the mechanism for implicit detail selection in this case. Further, OpenGL is used for rasterization of raw image sequences $S[1], S[2], \dots, S[T]$. The dimensionality of the descriptors is set to eight ($M = 8$).

The fitting process in this system assumes that during the fitting operation, there are K different scenes are available. Given point cloud \mathbf{P}^k and training ground truth set with L_k RGB images $\mathbf{I}^k - \{I^{k,1}, I^{k,2}, \dots, I^{k,L_k}\}$ and known camera parameters $\{C^{k,1}, C^{k,2}, \dots, C^{k,L_k}\}$ for k -th scene, helps formulate objective function \mathcal{L} (mismatch between the rendered and ground truth RGB image) as

given in the eqn. 86.

$$\mathcal{L}(\theta, \mathbf{D}^1, \mathbf{D}^2, \dots, \mathbf{D}^K) = \sum_{k=1}^K \sum_{l=1}^{L_k} \left(\Delta \left(\mathcal{R}_\theta(\{S[i](\mathbf{P}^k, \mathbf{D}^k, C^{k,l})\}_{i=1}^T) \right), I^{k,l} \right). \quad (86)$$

Here, Δ represents the mismatch between ground truth and rendered images, \mathbf{D}^k are neural descriptors for point clouds of the k -th scene and a perceptual loss [51,102] helps compute the mismatch for activations of a pretrained VGG16 network [193]. Both the neural descriptors and Rendering network parameters in the training set of the scenes are used for the optimizing the loss in eqn. 86 by the ADAM optimizer [109]. This system can be used for transfer/incremental learning which suggests that the results for new viewpoints tend to be better when rendering network is fitted to multiple scenes of a similar kind when compared to a single scene. Hence, this process is two-stage training process where the network is first pretrain for the rendering network on a scene family, followed by fine-tuning the rendering network to a new scene. In the later stage, the learning process 86 starts with no neural scenes descriptors and pretrained weights of the rendering network.

The inspiration of NeRF architecture can be said to be derived from Mildenhall and Tancik's own work [147,209], Sitzmann et al's **SIREN (Implicit Neural Representations with Periodic Activation Functions)** [194] networks introduced in the same year. SIRENs [194] introduced a novel approach for constructing implicit representations using periodic activation functions. This technique has shown to provide a more efficient and stable way to represent complex functions with deep neural networks. Whereas, “Fourier features let networks learn high frequency functions in low dimensional domains” [209] proposed a way to augment neural networks with Fourier features to allow them to learn high-frequency functions (real-life image representations) in low-dimensions. This approach enables neural networks to represent complex functions with fewer parameters and to learn more efficiently. Finally, the NeRFs build on the previous ideas to propose a method to represent 3D scenes using a continuous function for the plenoptic function that can be implicitly stored as Neural Representation. This approach allowed for high-quality view synthesis by rendering images from any viewpoint in a scene using much less storage when compared to polygonal meshes or pointclouds. (Work in Progress for Surveys related to NeRFs...)

4 Important Architectures, Models, and Extensions

This section briefly discusses the Radiance Fields architecture and discusses the improvements, extensions, and models for different applications. These research works have been briefly described according to the year they were published on Arxiv. The author wants to keep the source of the research related to the Radiance Field Literature so as to stay consistent in the source of information

- **Research in 2020:** This subsection briefly discusses the research articles in 2020. NeRFs were introduced in the March of this year as a preprint rolled out by Mildenhall et al [148].

- **Original NeRF** [148] is the model that uses fully connected deep neural network to represent a continuous 3D scene function, which takes a 3D point and a viewing direction as inputs and outputs a color and density function values.
- **Fourier Features Let Neural Networks Learn High Frequency Functions in Low Dimensional Domains** [209] introduces Fourier features, which improve the ability of neural networks to learn high-frequency functions in low-dimensional domains. The authors propose mapping input coordinates to a higher-dimensional space using a random Fourier feature mapping, which significantly improves the network's ability to model high-frequency functions. This technique proves to be particularly useful for tasks such as geometric modeling and texture synthesis.
- **GRAFs (Generative Radiance Fields)** [188] extend NeRFs to the domain of generative modelling. While NeRF focuses on learning a continuous 3D representation of a scene from a set of 2D images with known camera parameters, GRAFs aim to generate novel 3D scenes and synthesize views of those scenes without relying on explicit 3D geometry. GRAFs combine the NeRFs representation with a generative model like VAE (Variational Auto-Encoder) or a GAN (Generative Adversarial Networks). The generative model learns a latent space that encodes the underlying structure of the 3D scene with the conditional NeRF mapping the points in this latent space to 3D radiance fields. The training objective typically involves a combination of reconstruction and regularization loss. After training, GRAFs can sample latent vectors from the learned latent spaces and synthesize views of the scenes using conditional NeRF model. This makes GRAFs particularly useful for tasks such as view synthesis, 3D scene interpolation, and data augmentation, as they can generate diverse and realistic 3D scenes without relying on explicit 3D geometry.
- **NSVF (Neural Sparse Voxel Fields)** [131] extends NeRF by introducing a sparse voxel field representation, which significantly reduces memory requirements and accelerates rendering. This method combines a sparse voxel octree with a neural network to model scene appearance, enabling efficient reconstruction and rendering of large scale scenes.
- **NeRF-W (NeRF in the Wild)** [140] is an extension of the original NeRF model to handle more complex, real-world scenes with uncontrolled illumination. NeRF-W introduces additional input features to represent the spatially-varying lighting conditions, improving the model's ability to generalize across diverse scenes.
- **Neural Reflectance Fields** [14] presented a method to capture spatially-varying appearance and fine-geometric details of real-world objects using NeRF. It introduces a multi-scale reflectance field representation and a multi-view photometric stereo method to estimate per-point reflectance and normals, enabling high-fidelity reconstruction of complex objects.

- **Neural Face Reflectance Fields** [211] proposed a method to reconstruct neural face reflectance fields from a single monocular input image. This method estimates per-pixel surface normals, albedo, lighting parameters and combines them with NeRF-based representation to synthesize high-quality novel views of the face.
- **GRF (Learning a General Radiance Fields for 3D Representation and Rendering)** [214] presents a method for learning a 3D representation of a scene and rendering it using neural radiance fields. The authors introduce a multi-scale representation that allows the network to model both global structures and fine details of a scene. The resulting model can be used for various tasks, such as novel view synthesis, relighting, and geometric reconstruction.
- **NeRF++** [257] model builds upon the original NeRF by incorporating additional features such as spatially-varying reflectance and incorporating auxiliary tasks to improve training. The main goal of NeRF++ is to enhance the quality of the generated images while maintaining robustness and efficiency.
- **Neural Scene Graphs for Dynamic Scenes** [165] focuses on representing and rendering dynamic 3D scenes using neural scene graphs. The authors propose a method that combines scene graph representations with neural radiance fields to enable efficient rendering of complex and dynamic scenes. The method can generate high-quality images of dynamic scenes with realistic lighting and reflections, as well as handle occlusions and disocclusions.
- **GIRAFFE (Compositional Generative Neural Feature Fields)** [160] presents a compositional generative model that represents scenes as neural feature fields. The method combines a scene graph representation with a conditional NeRF model to synthesize novel views of complex scenes with varying objects, poses, and appearances, enabling high-quality image synthesis and scene editing.
- **DeRF (Decomposed Radiance Fields)** [177] introduced a method to separate the global illumination, local shading, and surface color components of a scene, enabling efficient view synthesis and relighting. The method decomposes a neural radiance field into these components and learns a hierarchical representation that captures the spatial and directional dependencies in the scene.
- **NerFies (Deformable Neural Radiance Fields)** [166] extended NeRF to represent non-rigidly deforming objects, such as human faces and bodies. The method combines a neural radiance field with a parametric deformation model, enabling reconstruction and view synthesis of dynamic scenes from monocular video.
- **Space-Time Neural Irradiance Fields** [235] introduced a space-time representation that models the temporal dynamics of free-viewpoint video. The method extends NeRF to represent time-varying appearance and geometry, enabling high-quality view synthesis and interpolation of dynamic scenes.

- **Neural Scene Flow Fields** [127] presented a method to represent the scene flow of dynamic objects using NeRF. The method learns a neural scene flow field that models the temporal evolution of object appearance and geometry. By encoding both spatial and temporal information, it enables high-quality view synthesis of dynamic scenes from a limited number of input views.
- **D-NeRF (Neural Radiance Fields for Dynamic Scenes)** [173] extended NeRF to handle dynamic scenes by learning a temporally coherent neural radiance field from a set of multi-view images with known camera parameters. D-NeRF models the time-varying appearance and geometry by conditioning the neural radiance field on a latent code that varies with time. The method introduces an additional loss term to enforce temporal smoothness in the learned radiance field, allowing for high-quality view synthesis of dynamic scenes with temporal consistency. This approach enables the reconstruction of dynamic scenes from multi-view videos and synthesizes novel views while maintaining the appearance and motion continuity.
- **pixelNeRF** [250] extends NeRF to generate high-quality novel views from one or few input images, instead of requiring multiple images with known camera parameters. It leverages 2D convolutions to efficiently aggregate information from input images and incorporates this information into the neural radiance field, allowing for improved view synthesis from limited input data.
- **Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction** [59] proposed a method to reconstruct dynamic neural radiance fields for human faces from a single monocular video. The method captures the 4D facial geometry and appearance, allowing for high-quality facial avatar reconstruction and novel view synthesis with temporal consistency.
- **NeRD (Neural Reflectance Decomposition)** [17] presented a method for learning scene-specific neural reflectance models from image collections. The method involves decomposing input images into albedo, normal, and lighting components, allowing for more accurate and flexible representations of complex scenes. This approach can be used to synthesize novel images, re-light scenes, and perform geometry-aware image editing.
- **NeRV (Neural Reflectance and Visibility Fields)** [202] introduced a method to represent both reflectance and visibility fields for relighting and view synthesis tasks. The method leverages a novel hierarchical representation that enables efficient and high-quality rendering of complex scenes under novel lighting conditions and viewpoints.
- **iNeRF (Inverting Neural Radiance Fields)** [248] presented a method to invert neural radiance fields to estimate the pose of an object or scene given a single image. By learning a mapping from 2D images to the latent space of the radiance field, iNeRF enables accurate pose estimation without explicit 3D geometry.

- **Portrait Neural Radiance Fields** [62] focused on generating portrait neural radiance fields from a single input image. The method leverages a combination of geometric priors, facial landmarks (such as 3DMM features), and a neural radiance field representation to reconstruct 3D facial geometry and appearance, enabling high-quality novel view synthesis. The priors are incorporated from a dataset of 3D facial scans. The resulting method can generate novel views and relight the face, even with limited input data.
- **Object-Centric Neural Scene Rendering** [73] worked on an object-centric representation for neural scene rendering. The method represents scenes as a collection of object-centric neural radiance fields and learns to compose them into a global scene representation, enabling high-quality view synthesis and intuitive scene editing.
- **NeRFFlow (Neural Radiance Flow)** [54] extends NeRF to model the temporal dynamics of video sequences. The method learns a continuous 4D radiance field that represents both the spatial and temporal information in the video, enabling high-quality view synthesis and video processing tasks, such as interpolation and extrapolation.
- **Compositional Radiance Fields** [224] presents a method to learn compositional radiance fields for dynamic human heads. The method combines a parametric human head model with a neural radiance field representation, allowing for high-quality view synthesis and editing of dynamic human head models.
- **Non-Rigid Neural Radiance Fields** [213] proposes a method to reconstruct and synthesize novel views of non-rigidly deforming scenes from monocular video. The method combines a neural radiance field with a non-rigid deformation model, enabling high-quality reconstruction and view synthesis of dynamic scenes.
- **Neural Body** [170] paper proposes a method for representing and synthesizing novel views of dynamic humans using structured latent codes and implicit neural representations. The authors introduce a method that decomposes the input data into body shape, pose, and appearance components, enabling the network to learn a more accurate and flexible representation of the human body. The method can generate realistic and temporally coherent novel views of dynamic humans, even with sparse input data. Hence, the authors build upon the NeRF framework by introducing structured latent codes to represent dynamic humans. Specifically, they decompose the input data into body shape, pose, and appearance components. This decomposition allows the model to capture the complex and dynamic nature of human bodies more effectively than standard NeRF. (Work in Progress...)
- **Research in 2021:** This subsection briefly discusses the research articles in the following year 2021. There were 110 research papers related to the field of Neural Radiance Fields (not including survey papers) in 2021.
 - **Non-line-of-sight Imaging via Neural Transient Fields** [192] presented a method for non-line-of-sight (NLOS) imaging using neural tran-

sient fields. The authors proposed a learning-based framework that leverages transient light measurements to reconstruct hidden objects. The key idea was to represent the volumetric distribution of light in a scene using a neural network, which learns the mapping from transient measurements to the 3D geometry and reflectance of the hidden objects. While not a direct extension of NeRF, this work shares similarities with NeRF in representing volumetric information using neural networks.

- **Pixel-Aligned Volumetric Avatars** [176] introduced a method for creating pixel-aligned volumetric avatars from multi-view video data. It proposed a learning-based approach that combines the benefits of traditional multi-view stereo and neural rendering methods. By using a fully convolutional neural network to predict a volumetric representation, the method generates high-quality avatars with consistent appearance across multiple views. Although this research does not directly extend NeRF, it shares the goal of creating high-quality 3D representations that can be efficiently rendered into 2D images.
- **A-NeRF (Articulated Neural Radiance Fields)** [204] was an extension of NeRF that focused on learning human shape, appearance, and pose. It proposed a method that represents humans as an articulated structure, where each body part has its own local neural radiance field. This decomposition allows A-NeRF to capture the complex geometry and appearance of humans more effectively than the original NeRF. The method can reconstruct detailed 3D models of humans and render novel views with realistic appearance and accurate poses.
- **NeRF– (Neural Radiance Fields Without Known Camera Parameters)** [223] was an extension of NeRF that relaxes the requirement of having known camera parameters. The authors propose a method that jointly learns the neural radiance field and camera parameters from a collection of uncalibrated input images. By incorporating differentiable camera optimization, the method can estimate both the 3D scene representation and the camera parameters simultaneously. NeRF– enables the use of NeRF for scenarios where camera parameters are not known or only partially available.
- **ShaRF (Shape-Conditioned Radiance Fields)** [180] is an extension of NeRF that focuses on learning 3D scene representations from a single input view. It combines a shape-conditioned radiance field with a shape prior learned from a large dataset of 3D models. The method leverages this shape prior to guide the learning of the radiance field, which enables accurate 3D reconstructions and novel view synthesis even with limited input data. ShaRF demonstrates the potential of using NeRF in single-view scenarios by incorporating additional prior knowledge about the underlying 3D structure.
- **NeuTex** [236] is an extension of NeRF that focuses on improving the efficiency of volumetric neural rendering by introducing neural texture mapping. The authors propose a two-stage approach that decouples geometry and appearance: first, a coarse geometry is reconstructed using

a neural radiance field, and then, a neural texture map is learned to refine the appearance. The training paradigm involves a combination of a differentiable ray-marching algorithm and texture loss. The method allows for faster rendering and lower memory requirements compared to the original NeRF, while maintaining high-quality results for tasks like novel view synthesis and relighting.

- **Mixture of Volumetric Primitives for Efficient Neural Rendering** [136] introduces an improvement over NeRF by representing scenes as a mixture of volumetric primitives, such as spheres and cylinders, which reduces the complexity of the scene representation. The training paradigm involves optimizing the parameters of these primitives, their positions, and a neural radiance field for each primitive. The loss function is a combination of the rendering loss and a regularization term. The method enables more efficient rendering and storage while maintaining the quality of the generated images, making it suitable for real-time applications and rendering large scenes.
- **DONeRF (Depth Oracle Networks-based Compact Radiance Fields)** [154] is an extension of NeRF that aims to achieve real-time rendering of compact radiance fields. The authors propose using a depth oracle network to predict the depth of the scene, which reduces the number of samples required for rendering. The training paradigm involves a two-stage approach: first, the depth oracle network is trained using supervised depth data, and then, the neural radiance field is trained using the predicted depth. The loss function combines a rendering loss and a depth loss. DONeRF significantly accelerates the rendering process while maintaining high-quality results, making it suitable for real-time applications.
- **NeX** [230] is an improvement over NeRF that focuses on real-time view synthesis by using a neural basis expansion approach. The authors propose representing the scene as a linear combination of basis functions, which are learned using a neural network. The training paradigm involves optimizing the neural network to predict the basis coefficients and the radiance field. The loss function is a combination of the rendering loss and a regularization term. NeX enables real-time view synthesis with high-quality results and lower memory requirements compared to NeRF, making it suitable for real-time applications and virtual reality.
- **FastNeRF** [64] is an improvement over NeRF that aims to achieve high-fidelity neural rendering at high frame rates. The authors propose a multi-scale hierarchical approach that accelerates rendering by leveraging a coarse-to-fine strategy. The training paradigm involves training multiple neural radiance fields at different levels of detail, and the loss function combines the rendering loss and a multi-scale consistency loss. FastNeRF significantly accelerates the rendering process while maintaining high-quality results, making it suitable for applications that require high frame rates, such as video games and virtual reality.

- **Neural Lumigraph Rendering** [107] is an extension of NeRF that combines the benefits of traditional Lumigraph rendering and neural radiance fields. The authors propose a method that leverages a sparse set of input views to learn a multi-plane representation of the scene. This multi-plane representation is then used as input to a neural rendering network, which generates high-quality novel views. The training paradigm involves optimizing the neural rendering network using a rendering loss, which measures the difference between the ground truth and the predicted images. This method offers a more efficient and flexible rendering solution than NeRF and can be used for applications like view synthesis and relighting.
- **iMAP (Implicit Mapping and Positioning)** [205] is an application of NeRF that focuses on real-time simultaneous mapping and positioning in 3D environments. The authors propose a method that learns an implicit 3D map of the environment using a neural radiance field and estimates the camera pose simultaneously. The training paradigm involves optimizing the neural network to minimize the re-projection error and a depth consistency loss. iMAP demonstrates the potential of using NeRF for real-time 3D mapping and positioning tasks, such as robot navigation and augmented reality.
- **Mip-NeRF** [10] is an improvement over NeRF that introduces a multi-scale representation to reduce aliasing artifacts in the rendered images. The authors propose a mipmapping approach that learns a hierarchy of neural radiance fields at different levels of detail. The training paradigm involves optimizing the neural networks at each level of detail using a rendering loss and a consistency loss between adjacent levels. Mip-NeRF achieves higher-quality rendering with fewer aliasing artifacts, making it suitable for applications that require high-quality image synthesis, such as virtual reality and film production.
- **Kilo-NeRF** [178] is an improvement over NeRF that aims to speed up rendering by using thousands of tiny multilayer perceptrons (MLPs) instead of a single large MLP. The authors propose dividing the scene into small regions and training a separate MLP for each region, which allows for more efficient rendering and parallelization. The training paradigm involves optimizing the MLPs using a rendering loss that measures the difference between the ground truth and the predicted images. Kilo-NeRF significantly accelerates the rendering process while maintaining high-quality results, making it suitable for applications that require real-time rendering, such as video games and virtual reality.
- **PlenOctrees** [249] is an improvement over NeRF that focuses on real-time rendering by introducing a hierarchical data structure called PlenOctrees. The authors propose representing the scene using an octree, where each node stores a neural radiance field that models the local geometry and appearance. The training paradigm involves optimizing the neural radiance fields using a rendering loss and a multi-scale consistency loss. PlenOctrees enables real-time rendering of neural radiance fields with

high-quality results and lower memory requirements compared to NeRF, making it suitable for real-time applications and virtual reality.

- **Baking Neural Radiance Fields** [82] is an improvement over NeRF that focuses on real-time view synthesis by converting learned neural radiance fields into a more efficient representation for rendering. The authors propose a method that bakes the neural radiance field into a set of layered depth images (LDIs) and then uses these LDIs for real-time rendering. The training paradigm involves optimizing the neural radiance field using a rendering loss that measures the difference between the ground truth and the predicted images. This method significantly accelerates the rendering process while maintaining high-quality results, making it suitable for real-time applications such as video games and virtual reality.
- **MINE (Multiview Image-based Neural Extensions)** [124] is an extension of NeRF that combines the benefits of multiplane images (MPIs) and NeRF for novel view synthesis. The authors propose a method that learns a continuous MPI representation using a neural network, which can be used for efficient rendering of novel views. The training paradigm involves optimizing the neural network using a rendering loss and a depth consistency loss. MINE demonstrates the potential of combining the strengths of MPIs and NeRF to achieve high-quality view synthesis with lower computational requirements, making it suitable for real-time applications and virtual reality.
- **MVSNeRF (Multi-view Stereo Neural Radiance Fields)** [25] is designed to leverage the strengths of both multi-view stereo (MVS) and NeRF techniques. MVSNeRF combines a coarse MVS reconstruction with the NeRF framework to generate high-quality novel views. This hybrid approach takes advantage of the geometric information provided by MVS to improve the efficiency and quality.
- **GNeRF (Generative Adversarial Network-based Neural Radiance Field)** [144] is an extension of NeRF that introduces a GAN-based framework for learning neural radiance fields without known camera poses. The authors propose a method that jointly learns the neural radiance field and camera poses using a combination of a rendering loss, an adversarial loss, and a cycle consistency loss. The training paradigm involves optimizing the neural radiance field, camera poses, and the discriminator network. GNeRF demonstrates the potential of using GANs to learn high-quality neural radiance fields in scenarios where camera poses are unknown, making it suitable for applications like novel view synthesis and 3D reconstruction.
- **In-Place Scene Labelling and Understanding with Implicit Scene Representation** [265] presents an application of NeRF that focuses on in-place scene labeling and understanding using implicit scene representations. The authors propose a method that learns a joint representation of geometry, appearance, and semantic information using a neural radiance field. The training paradigm involves optimizing the neural network

using a combination of rendering loss, depth consistency loss, and a semantic loss. This method demonstrates the potential of using NeRF for scene understanding tasks, such as semantic segmentation and object recognition, in addition to view synthesis.

- **CAMPARI (Camera-Aware Multiplane Radiance Fields)** [159] is an improvement over NeRF that introduces a camera-aware decomposed generative framework for learning neural radiance fields. The authors propose a method that decomposes the input images into depth layers and learns a separate neural radiance field for each layer. The training paradigm involves optimizing the neural networks using a combination of rendering loss, depth consistency loss, and a cycle consistency loss. CAMPARI enables more efficient rendering and better handling of occlusions, making it suitable for applications like novel view synthesis and relighting.
- **NeRF-VAE (Neural Radiance Fields - Variational Autoencoder)** [113] is an extension of NeRF that combines the geometry-aware representation of NeRF with the generative capabilities of VAEs. The authors propose a method that learns a latent space for 3D scenes, which can be used for generating novel scenes and interpolating between existing ones. The training paradigm involves optimizing the neural network using a combination of rendering loss, KL-divergence loss, and a reconstruction loss. NeRF-VAE demonstrates the potential of using NeRF for generative modeling tasks, such as 3D scene generation and interpolation.
- **Unconstrained Scene Generation with Locally Conditioned Radiance Fields** [50] presents an extension of NeRF that focuses on unconstrained scene generation using locally conditioned radiance fields. The authors propose a method that conditions the neural radiance field on local spatial information, allowing it to generate complex and diverse scenes. The training paradigm involves optimizing the neural network using a combination of rendering loss and a perceptual loss. This method demonstrates the potential of using NeRF for generative modeling tasks, such as scene synthesis and object generation, in an unconstrained setting.
- **Putting NeRF on a Diet** [95] presents an improvement over NeRF that focuses on semantically consistent few-shot view synthesis. The authors propose a method that leverages semantic information to guide the neural radiance field learning process, enabling it to synthesize novel views with as few as one input image. The training paradigm involves optimizing the neural network using a combination of rendering loss, semantic consistency loss, and a cycle consistency loss. This method demonstrates the potential of using NeRF for few-shot view synthesis tasks while maintaining semantic consistency, making it suitable for applications like virtual reality and film production.
- **Decomposing 3D Scenes into Objects via Unsupervised Volume Segmentation** [203] presents an application of NeRF that focuses on decomposing 3D scenes into objects using unsupervised volume segmen-

tation. The authors propose a method that learns to segment the scene into different objects by leveraging the geometric information provided by the neural radiance field. The training paradigm involves optimizing the neural network using a combination of rendering loss and a segmentation loss. This method demonstrates the potential of using NeRF for unsupervised object segmentation tasks, which can be useful for applications like 3D object recognition and scene understanding.

- **Convolutional Neural Opacity Radiance Fields** [138] is an improvement over NeRF that introduces a convolutional architecture for learning neural radiance fields. The authors propose a method that replaces the fully connected layers in NeRF with convolutional layers, allowing the model to leverage local spatial information and achieve better performance. The training paradigm involves optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images. This method demonstrates the potential of using convolutional architectures in NeRF, leading to more efficient and accurate rendering, making it suitable for applications like novel view synthesis and relighting.
- **NARF (Neural Articulated Radiance Fields)** [162] is an extension of NeRF that focuses on modeling articulated objects, such as humans or animals, by leveraging the articulated structure of the object. The authors propose a method that combines NeRF with a kinematic model to represent the geometry and appearance of articulated objects in a coherent and continuous manner. The training paradigm involves optimizing the neural network using a combination of rendering loss and a rigidity loss that encourages a consistent geometry across poses. NARF demonstrates the potential of using NeRF for modeling articulated objects, making it suitable for applications like character animation and motion capture.
- **Neural RGB-D Scene Reconstruction** [8] is an application of NeRF that focuses on 3D scene reconstruction from RGB-D (color and depth) images. The authors propose a method that leverages the depth information available in RGB-D images to learn a more accurate and efficient neural radiance field. The training paradigm involves optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images. This method demonstrates the potential of using NeRF for 3D scene reconstruction tasks with RGB-D images, making it suitable for applications like robotics and augmented reality.
- **BARF (Bundle-Adjusting Neural Radiance Fields)** [129] is an improvement over NeRF that introduces bundle adjustment, a technique commonly used in structure-from-motion, for refining camera poses and optimizing the neural radiance field simultaneously. The authors propose a method that jointly optimizes the camera poses and the neural radiance field, leading to more accurate and consistent scene reconstruction. The training paradigm involves optimizing the neural network and camera

poses using a combination of rendering loss and a bundle adjustment loss. BARF demonstrates the potential of incorporating bundle adjustment in NeRF, making it suitable for applications like 3D reconstruction and novel view synthesis.

- **SRF (Stereo Radiance Fields)** [33] is an extension of NeRF that focuses on learning view synthesis for sparse views of novel scenes. The authors propose a method that leverages stereo pairs of input images to learn a more accurate and efficient neural radiance field. The training paradigm involves optimizing the neural network using a combination of rendering loss, depth consistency loss, and a stereo consistency loss that encourages consistency between the stereo pairs. SRF demonstrates the potential of using NeRF for view synthesis tasks with sparse input views, making it suitable for applications like virtual reality and film production.
- **FiG-NeRF (Figure-Grounded Neural Radiance Fields)** [238] is an extension of NeRF that focuses on 3D object category modeling by learning a figure-ground decomposition of the scene. The authors propose a method that learns separate neural radiance fields for the object and the background, enabling the model to focus on the object's geometry and appearance. The training paradigm involves optimizing the neural network using a combination of rendering loss and a figure-ground consistency loss that encourages a consistent decomposition of the scene. FiG-NeRF demonstrates the potential of using NeRF for 3D object category modeling tasks, making it suitable for applications like object recognition and scene understanding.
- **Shadow Neural Radiance Fields** [49] is an application of NeRF that focuses on multi-view satellite photogrammetry, specifically on reconstructing 3D geometry and appearance of Earth's surface from satellite images. The authors propose a method that extends NeRF to account for the complex illumination effects caused by shadows and atmospheric scattering in satellite images. The training paradigm involves optimizing the neural network using a combination of rendering loss, depth consistency loss, and a shadow consistency loss. This method demonstrates the potential of using NeRF for satellite photogrammetry tasks, making it suitable for applications like Earth observation and remote sensing.
- **UNISURF (Unifying Neural Implicit Surfaces and Radiance Fields)** [163] is an improvement over NeRF that aims to unify neural implicit surface representations and neural radiance fields for multi-view reconstruction. The authors propose a method that combines the advantages of both representations, enabling the model to learn geometry and appearance more accurately and efficiently. The training paradigm involves optimizing the neural network using a combination of rendering loss, depth consistency loss, and a surface consistency loss. UNISURF demonstrates the potential of combining neural implicit surfaces and radiance fields for multi-view reconstruction tasks, making it suitable for applications like 3D reconstruction and novel view synthesis.

- **Editable Free-Viewpoint Video using a Layered Neural Representation** [256] presents an application of NeRF that focuses on editable free-viewpoint video using a layered neural representation. The authors propose a method that decomposes the scene into a set of layered neural radiance fields, enabling users to edit the content of the video by modifying individual layers. The training paradigm involves optimizing the neural network using a combination of rendering loss, depth consistency loss, and a layer consistency loss. This method demonstrates the potential of using NeRF for editable free-viewpoint video tasks, making it suitable for applications like video editing and content creation.
- **Animatable Neural Radiance Fields** [169] is an extension of NeRF that focuses on modeling dynamic human bodies by incorporating a parametric human body model into the neural radiance field framework. The authors propose a method that learns a neural radiance field conditioned on the pose and shape parameters of the human body model, enabling the generation of realistic animations. The training paradigm involves optimizing the neural network using a combination of rendering loss, pose consistency loss, and a shape consistency loss. This method demonstrates the potential of using NeRF for animating human bodies, making it suitable for applications like character animation and motion capture.
- **Editing Conditional Radiance Fields** [133] presents an extension of NeRF that focuses on editing conditional radiance fields, allowing users to edit the appearance and geometry of a scene. The authors propose a method that conditions the neural radiance field on an editing vector, enabling the model to generate edited versions of the scene by modifying the vector. The training paradigm involves optimizing the neural network using a combination of rendering loss, depth consistency loss, and an editing consistency loss. This method demonstrates the potential of using NeRF for editing tasks, making it suitable for applications like content creation, virtual reality, and film production.
- **Dynamic View Synthesis from Dynamic Monocular Videos** [61] is an application of NeRF that focuses on dynamic view synthesis from dynamic monocular videos. The authors propose a method that extends NeRF to handle dynamic scenes by conditioning the neural radiance field on time, allowing the model to learn a time-varying geometry and appearance. The training paradigm involves optimizing the neural network using a combination of rendering loss, temporal consistency loss, and a geometric consistency loss. This method demonstrates the potential of using NeRF for dynamic view synthesis tasks, making it suitable for applications like video editing, virtual reality, and film production.
- **Recursive-NeRF** [244] is an improvement over NeRF that aims to make the model more efficient by using a dynamically growing neural radiance field. The authors propose a method that recursively refines the radiance field by training smaller MLPs (Multilayer Perceptrons) on top of the existing ones, allowing the model to adapt to new views with minimal additional computation. The training paradigm involves

optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images. Recursive-NeRF demonstrates the potential of using a dynamically growing NeRF for efficient view synthesis, making it suitable for real-time applications.

- **Stylizing 3D Scene via Implicit Representation and HyperNetwork** [32] presents an extension of NeRF that focuses on stylizing 3D scenes using implicit representation and hypernetworks. The authors propose a method that combines NeRF with a hypernetwork, enabling the model to generate stylized versions of the scene by learning a style-specific radiance field. The training paradigm involves optimizing the neural network using a combination of rendering loss, style consistency loss, and a content consistency loss. This method demonstrates the potential of using NeRF for stylizing 3D scenes, making it suitable for applications like content creation, virtual reality, and film production.
- **NeRFactor** [259] is an improvement over NeRF that focuses on factorizing shape and reflectance under an unknown illumination. The authors propose a method that extends NeRF to learn a separate neural representation for geometry, reflectance, and illumination, allowing the model to disentangle these factors and reconstruct the scene more accurately. The training paradigm involves optimizing the neural network using a combination of rendering loss, reflectance consistency loss, and a geometry consistency loss. NeRFactor demonstrates the potential of using NeRF for disentangling shape, reflectance, and illumination, making it suitable for applications like 3D reconstruction and relighting.
- **Neural Actor** [132] is an extension of NeRF that focuses on neural free-view synthesis of human actors with pose control. The authors propose a method that conditions the neural radiance field on the pose of the actor, allowing the model to generate novel views of the actor in different poses. The training paradigm involves optimizing the neural network using a combination of rendering loss, pose consistency loss, and a geometric consistency loss. Neural Actor demonstrates the potential of using NeRF for synthesizing human actors with pose control, making it suitable for applications like character animation, virtual reality, and film production.
- **Light Field Networks** [195] is an improvement over NeRF that aims to speed up rendering by using a single network evaluation. The authors propose a method that learns a global light field representation, which can be sampled to generate novel views without querying the network for each ray. The training paradigm involves optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images. Light Field Networks demonstrates the potential of using a global light field representation for efficient rendering, making it suitable for real-time applications.
- **NeRF in detail (Learning to Sample for View Synthesis)** [3] is an improvement over NeRF that focuses on learning to sample rays more effectively for view synthesis. The authors propose a method that com-

bines NeRF with a sampling network, which learns to predict importance sampling weights along each ray to improve the efficiency of rendering. The training paradigm involves optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images. NeRF in detail demonstrates the potential of using importance sampling to enhance the efficiency of NeRF, making it suitable for real-time view synthesis tasks.

- **NeuS (Neural Surfaces)** [222] is an improvement over NeRF that aims to learn neural implicit surfaces by volume rendering for multi-view reconstruction. The authors propose a method that combines volume rendering with implicit surface representations, enabling the model to learn geometry and appearance more accurately and efficiently. The training paradigm involves optimizing the neural network using a combination of rendering loss, depth consistency loss, and surface consistency loss. NeuS demonstrates the potential of using volume rendering with implicit surface representations for multi-view reconstruction tasks, making it suitable for applications like 3D reconstruction and novel view synthesis.
- **Volume Rendering of Neural Implicit Surfaces** [246] presents an extension of NeRF that focuses on volume rendering of neural implicit surfaces. The authors propose a method that combines NeRF with a voxel-based representation, enabling the model to learn a more efficient and accurate representation of the scene. The training paradigm involves optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images. Volume Rendering of Neural Implicit Surfaces demonstrates the potential of using voxel-based representations with NeRF, making it suitable for applications like 3D reconstruction and novel view synthesis.
- **HyperNeRF** [167] is an improvement over NeRF that focuses on learning higher-dimensional representations for topologically varying neural radiance fields. The authors propose a method that extends NeRF to handle higher-dimensional input spaces, allowing the model to represent more complex and topologically varying scenes. The training paradigm involves optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images. HyperNeRF demonstrates the potential of using higher-dimensional representations to enhance the expressiveness of NeRF, making it suitable for modeling complex scenes and objects.
- **Animatable NeRFs from Monocular RGB Videos** [26] presents an application of NeRF that focuses on animatable neural radiance fields from monocular RGB videos. The authors propose a method that extends NeRF to learn a time-varying radiance field from monocular RGB videos, allowing the model to generate novel views of dynamic scenes. The training paradigm involves optimizing the neural network using a combination of rendering loss, temporal consistency loss, and a geometric consistency loss. This method demonstrates the potential of using

NeRF for animating dynamic scenes, making it suitable for applications like video editing, virtual reality, and film production.

- **Fast Training of Neural Lumigraph Representation using Meta Learning** [13] presents an improvement over NeRF that focuses on fast training of neural lumigraph representation using meta-learning. The authors propose a method that combines NeRF with meta-learning techniques, enabling the model to learn a neural lumigraph representation more quickly. The training paradigm involves optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images, along with a meta-learning loss that measures the generalization performance of the model. This method demonstrates the potential of using meta-learning to speed up the training of NeRF, making it suitable for real-time applications.
- **Depth-Supervised NeRF** [47] is an improvement over NeRF that focuses on using fewer views and faster training without sacrificing quality. The authors propose a method that incorporates depth supervision into the NeRF training process, allowing the model to learn more accurate and efficient representations with fewer input views. The training paradigm involves optimizing the neural network using a combination of rendering loss and depth consistency loss. Depth-Supervised NeRF demonstrates the potential of using depth supervision to enhance the efficiency of NeRF, making it suitable for applications with limited input views.
- **3D Neural Scene Representation for VisuoMotor Control** [126] presents an application of NeRF that focuses on 3D neural scene representation for visuomotor control. The authors propose a method that extends NeRF to learn a compact and efficient 3D scene representation, which can be used for visuomotor control tasks like robotic manipulation and navigation. The training paradigm involves optimizing the neural network using a combination of rendering loss and a task-specific loss that measures the performance of the visuomotor control policy. This method demonstrates the potential of using NeRF for visuomotor control applications, making it suitable for robotics and autonomous systems.
- **Unsupervised Discovery of Object Radiance Fields** [251] presents an extension of NeRF that focuses on unsupervised discovery of object radiance fields. The authors propose a method that learns object-centric radiance fields without any supervision, allowing the model to discover and represent objects in a scene in an unsupervised manner. The training paradigm involves optimizing the neural network using a combination of rendering loss, object consistency loss, and a scene consistency loss. Unsupervised Discovery of Object Radiance Fields demonstrates the potential of using NeRF for unsupervised object discovery and representation, making it suitable for applications like 3D reconstruction and scene understanding.

- **Neural Rays for Occlusion-aware Image-based Rendering** [134] is an improvement over NeRF that focuses on occlusion-aware image-based rendering. The authors propose a method called Neural Rays, which extends NeRF by explicitly modeling occlusion in the radiance field. This enables the model to generate more accurate and realistic novel views, especially in scenes with complex occlusions. The training paradigm involves optimizing the neural network using a combination of rendering loss and an occlusion-aware loss. Neural Rays demonstrate the potential of incorporating occlusion information into NeRF, making it suitable for applications like image-based rendering and 3D reconstruction.
- **Differentiable Surface Rendering via Non-Differentiable Sampling** [37] presents an improvement over NeRF that focuses on differentiable surface rendering using non-differentiable sampling. The authors propose a method that combines NeRF with a differentiable rendering framework that can handle non-differentiable sampling operations, enabling the model to learn more accurate and efficient representations of surfaces. The training paradigm involves optimizing the neural network using a combination of rendering loss and a surface consistency loss. Differentiable Surface Rendering via Non-Differentiable Sampling demonstrates the potential of using differentiable rendering with NeRF, making it suitable for applications like 3D reconstruction and novel view synthesis.
- **FLAME-in-NeRF** [5] is an application of NeRF that focuses on free-view face animation. The authors propose a method that combines NeRF with the FLAME model, a parametric face model that encodes facial shape, expression, and pose information. This enables the generation of novel views of animated faces with high-quality and realistic appearance. The training paradigm involves optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images. FLAME-in-NeRF demonstrates the potential of using NeRF for face animation, making it suitable for applications like virtual reality, video games, and film production.
- **iButter** [221] is an application of NeRF that focuses on generating interactive bullet time effects for human free-viewpoint rendering. The authors propose a method that extends NeRF to handle dynamic human scenes, enabling the model to generate novel views of humans in motion with a bullet time effect. The training paradigm involves optimizing the neural network using a combination of rendering loss and a temporal consistency loss. iButter demonstrates the potential of using NeRF for interactive bullet time generation, making it suitable for applications like video editing, virtual reality, and film production.
- **Self-Calibrating Neural Radiance Fields** [97] presents an improvement over NeRF that focuses on self-calibrating neural radiance fields. The authors propose a method that extends NeRF by learning to self-calibrate camera parameters, enabling the model to handle input data with noisy or incomplete camera parameters. The training paradigm in-

volves optimizing the neural network using a combination of rendering loss and a calibration consistency loss. Self-Calibrating Neural Radiance Fields demonstrate the potential of incorporating self-calibration into NeRF, making it suitable for applications like 3D reconstruction and novel view synthesis, especially when working with imperfect input data.

- **NeRFingMVS** [225] presents an improvement over NeRF that focuses on indoor multi-view stereo. The authors propose a method called NeRFing MVS, which extends NeRF by incorporating guidance from a multi-view stereo reconstruction to optimize the neural radiance fields for indoor scenes. The training paradigm involves optimizing the neural network using a combination of rendering loss and a guidance loss from the multi-view stereo reconstruction. NeRFing MVS demonstrates the potential of incorporating multi-view stereo guidance into NeRF, making it suitable for applications like indoor scene reconstruction and novel view synthesis.
- **CodeNeRF** [96] is an extension of NeRF that focuses on disentangling neural radiance fields for object categories. The authors propose a method that learns disentangled representations of shape and appearance in neural radiance fields, enabling the model to generate novel views of objects from different categories with consistent appearance properties. The training paradigm involves optimizing the neural network using a rendering loss that measures the difference between the ground truth and the predicted images, along with a disentanglement loss that encourages separation of shape and appearance features. CodeNeRF demonstrates the potential of using disentangled representations in NeRF, making it suitable for applications like object synthesis and 3D modeling.
- **Learning Object-Compositional Neural Radiance Fields for Editable Scene Rendering** [242] presents an extension of NeRF that focuses on learning object-compositional neural radiance fields for editable scene rendering. The authors propose a method that learns neural radiance fields for individual objects and composes them into a scene, enabling the model to generate novel views of editable scenes. The training paradigm involves optimizing the neural network using a combination of rendering loss, object consistency loss, and a scene consistency loss. This method demonstrates the potential of using object-compositional radiance fields in NeRF, making it suitable for applications like scene editing and virtual reality.
- **Stochastic Neural Radiance Fields** [191] is an extension of NeRF that focuses on quantifying uncertainty in implicit 3D representations. The authors propose a method that incorporates stochasticity into the neural radiance field, enabling the model to represent and quantify uncertainty in its predictions. The training paradigm involves optimizing the neural network using a combination of rendering loss and a stochastic consistency loss that encourages the model to capture uncertainty. Stochastic Neural Radiance Fields demonstrate the potential of incor-

porating uncertainty quantification into NeRF, making it suitable for applications like robust 3D reconstruction and probabilistic rendering.

- **Neural Human Performer** [117] is an application of NeRF that focuses on learning generalizable radiance fields for human performance rendering. The authors propose a method that extends NeRF to handle dynamic human performances, enabling the model to generate novel views of humans in motion with high-quality and realistic appearance. The training paradigm involves optimizing the neural network using a combination of rendering loss and a temporal consistency loss. Neural Human Performer demonstrates the potential of using NeRF for human performance rendering, making it suitable for applications like virtual reality, video games, and film production.
- **TöRF (Time-of-Flight Radiance Fields)** [7] is an extension of NeRF that incorporates Time-of-Flight (ToF) information to enable view synthesis for dynamic scenes. The authors propose a method that combines neural radiance fields with ToF depth measurements, making it possible to generate novel views of dynamic scenes with moving objects. The training paradigm involves optimizing the neural network using a combination of rendering loss and a ToF depth consistency loss. TöRF demonstrates the potential of using ToF information in NeRF, making it suitable for applications like robotics, autonomous vehicles, and augmented reality.
- **Vision-Only Robot Navigation in a Neural Radiance World** [1] presents an application of NeRF for vision-only robot navigation. The authors propose a method that leverages neural radiance fields to build a 3D representation of the environment, allowing a robot to navigate using only visual information. The training paradigm involves optimizing the neural network using a combination of rendering loss and a navigation loss that encourages the model to learn a useful representation for navigation. This approach demonstrates the potential of using NeRF for vision-only robot navigation, making it suitable for applications like autonomous robots and drone navigation.
- **Neural Radiance Fields Approach to Deep Multi-View Photometric Stereo** [106] presents an extension of NeRF that focuses on deep multi-view photometric stereo. The authors propose a method that combines neural radiance fields with multi-view photometric stereo to reconstruct the shape and appearance of objects in a scene. The training paradigm involves optimizing the neural network using a combination of rendering loss and a photometric consistency loss. This method demonstrates the potential of using NeRF for photometric stereo, making it suitable for applications like 3D reconstruction and virtual reality.
- **LENS (Localization Enhanced by NeRF Synthesis)** [150] is an application of NeRF that focuses on improving localization tasks using synthesized views. The authors propose a method that leverages NeRF to generate synthetic views that help in refining the localization estimates. The training paradigm involves optimizing the neural network using a

combination of rendering loss and a localization loss that encourages the model to generate useful views for localization. LENS demonstrates the potential of using NeRF for enhancing localization tasks, making it suitable for applications like robotics, autonomous vehicles, and augmented reality.

- **NeRS (Neural Reflectance Surfaces)** [255] is an extension of NeRF that focuses on sparse-view 3D reconstruction in the wild. The authors propose a method that combines neural radiance fields with an additional reflectance component, enabling the model to handle challenging real-world scenes with sparse views. The training paradigm involves optimizing the neural network using a combination of rendering loss and a reflectance consistency loss. NeRS demonstrates the potential of using NeRF for sparse-view 3D reconstruction, making it suitable for applications like 3D modeling and virtual reality.
- **StyleNeRF** [71] is an extension of NeRF that incorporates style-based generators for high-resolution image synthesis. The authors propose a method that combines neural radiance fields with a style-based generator, enabling high-quality 3D-aware image synthesis. The training paradigm involves optimizing the neural network using a combination of rendering loss and a style consistency loss. StyleNeRF demonstrates the potential of using NeRF for high-resolution image synthesis, making it suitable for applications like virtual reality, 3D modeling, and artistic image generation.
- **CIPS-3D** [267] is an extension of NeRF that focuses on generating images using conditionally-independent pixel synthesis. The authors propose a method that combines neural radiance fields with a GAN-based generator, enabling 3D-aware image synthesis. The training paradigm involves optimizing the neural network using a combination of rendering loss and a GAN loss. CIPS-3D demonstrates the potential of using NeRF for 3D-aware image generation, making it suitable for applications like 3D modeling, virtual reality, and artistic image generation.
- **H-NeRF** [239] is an extension of NeRF that focuses on rendering and temporal reconstruction of humans in motion. The authors propose a method that combines neural radiance fields with temporal information, enabling the model to reconstruct and render dynamic human motions. The training paradigm involves optimizing the neural network using a combination of rendering loss and a temporal consistency loss. H-NeRF demonstrates the potential of using NeRF for human motion reconstruction and rendering, making it suitable for applications like animation, virtual reality, and video game development.
- **Dex-NeRF** [89] is an application of NeRF that focuses on grasping transparent objects. The authors propose a method that leverages NeRF to generate a 3D representation of transparent objects, allowing a robot to grasp them using visual information. The training paradigm involves optimizing the neural network using a combination of rendering loss and a grasping loss that encourages the model to learn a useful representa-

tion for grasping. Dex-NeRF demonstrates the potential of using NeRF for grasping transparent objects, making it suitable for applications like robotics, automation, and manufacturing.

- **Neural-PIL** [18] is an extension of NeRF that focuses on reflectance decomposition using pre-integrated lighting. The authors propose a method that combines neural radiance fields with a pre-integrated lighting model, enabling the decomposition of reflectance properties of the scene. The training paradigm involves optimizing the neural network using a combination of rendering loss and a reflectance decomposition loss. Neural-PIL demonstrates the potential of using NeRF for reflectance decomposition, making it suitable for applications like 3D modeling, virtual reality, and computer graphics.
- **Template NeRF** [72] is an extension of NeRF that models dense shape correspondences from category-specific object images. The authors propose a method that leverages a template neural radiance field to establish dense shape correspondences across different instances of an object category. The training paradigm involves optimizing the neural network using a combination of rendering loss and a correspondence loss. Template NeRF is suitable for applications like 3D modeling, computer graphics, and object recognition.
- **DIVeR (Deterministic Integration for Volumetric Rendering)** [234] is an improvement over NeRF, offering real-time and accurate neural radiance fields with deterministic integration for volume rendering. The authors propose a deterministic integration method that accelerates the rendering process while maintaining accuracy. The training paradigm involves optimizing the neural network using a rendering loss. DIVeR demonstrates the potential for real-time volume rendering applications, such as virtual reality and 3D modeling.
- **DVGO (Direct Voxel Grid Optimization)** [206] is an improvement over NeRF, focusing on super-fast convergence for radiance field reconstruction. The authors propose a direct voxel grid optimization method that accelerates the convergence of the NeRF training process. The training paradigm involves optimizing the neural network using a rendering loss. DVGO is suitable for applications like 3D modeling, computer graphics, and real-time rendering.
- **Mip-NeRF 360** [11] is an extension of NeRF, focusing on unbounded anti-aliased neural radiance fields. The authors propose a method that leverages mipmapping to reduce aliasing artifacts in NeRF-based renderings. The training paradigm involves optimizing the neural network using a rendering loss. Mip-NeRF 360 is suitable for applications like virtual reality, computer graphics, and 3D modeling, where high-quality renderings are essential.
- **VaxNeRF** [112] is an improvement over NeRF that introduces voxel acceleration for faster rendering. The authors propose a method that combines voxel-based acceleration with neural radiance fields, resulting in faster rendering times without sacrificing quality. The training

paradigm involves optimizing the neural network using a rendering loss. VaxNeRF is suitable for real-time rendering applications, such as virtual reality and 3D modeling.

- **GeoNeRF** [101] is an extension of NeRF that incorporates geometric priors for better scene representation. The authors propose a method that integrates geometric information into the NeRF framework, improving the quality and accuracy of the rendered images. The training paradigm involves optimizing the neural network using a combination of rendering loss and a geometric prior loss. GeoNeRF is suitable for applications like 3D modeling, computer graphics, and virtual reality.
- **NeRF in the Dark** [146] is an extension of NeRF that focuses on high dynamic range (HDR) view synthesis from noisy raw images. The authors propose a method that leverages NeRF for rendering HDR images from raw, noisy input data. The training paradigm involves optimizing the neural network using a combination of rendering loss and a denoising loss. NeRF in the Dark is suitable for applications like photography, computer graphics, and virtual reality, where HDR rendering from noisy raw images is essential.
- **Deblur-NeRF** [139] is an extension of NeRF that focuses on reconstructing neural radiance fields from blurry images. The authors propose a method that leverages NeRF to render sharp images from input data that is blurred due to camera motion or other factors. The training paradigm involves optimizing the neural network using a combination of rendering loss and a deblurring loss. Deblur-NeRF is suitable for applications like photography, computer graphics, and virtual reality, where rendering sharp images from blurry inputs is important.
- **HDR-NeRF (High Dynamic Range Neural Radiance Fields)** [88] is an extension of NeRF, focusing on high dynamic range (HDR) neural radiance fields. The authors propose a method that enables NeRF to handle HDR images by modeling both radiance and color. The training paradigm involves optimizing the neural network using a rendering loss. HDR-NeRF is suitable for applications like photography, computer graphics, and virtual reality, where HDR rendering is essential.
- **iLabel (Interactive Neural Scene Labelling)** [266] is an application of NeRF that focuses on interactive neural scene labeling. The authors propose an interactive framework for labeling scenes with semantic information using a neural radiance field. The training paradigm involves optimizing the neural network using a combination of rendering loss and a semantic loss. iLabel is suitable for applications like 3D scene understanding, computer graphics, and virtual reality.
- **Urban Radiance Fields** [179] is an application of NeRF that focuses on modeling and rendering complex urban environments. The authors propose a method that leverages NeRF to capture and render detailed urban scenes. The training paradigm involves optimizing the neural network using a rendering loss. Urban Radiance Fields is suitable for applications like urban planning, virtual reality, and computer graphics.

- **NeRFReN (Neural Radiance Fields with ReflectioNs)** [74] is an extension of NeRF that incorporates reflections into the rendering process. The authors propose a method that models reflections in neural radiance fields by incorporating a reflection term in the NeRF formulation. The training paradigm involves optimizing the neural network using a rendering loss. NeRFReN is suitable for applications like computer graphics, virtual reality, and 3D modeling, where realistic reflections are important.
- **Hallucinated Neural Radiance Fields in the Wild** [31] is an improvement over NeRF, focusing on rendering hallucinated neural radiance fields in the wild. The authors propose a method that leverages NeRF to generate plausible renderings of scenes with limited or no input data. The training paradigm involves optimizing the neural network using a rendering loss. Hallucinated-NeRF is suitable for applications like computer graphics, virtual reality, and 3D modeling, where input data may be scarce or unavailable.
- **NeuSample (Neural Sample Field for Efficient View Synthesis)** [56] is an improvement over NeRF that focuses on efficient view synthesis. The authors propose a method that learns a neural sample field to guide the sampling process, reducing the number of required samples for high-quality renderings. The training paradigm involves optimizing the neural network using a rendering loss. NeuSample is suitable for applications like virtual reality, computer graphics, and real-time rendering.
- **RegNeRF (Regularized Neural Radiance Fields)** [158] is an improvement over NeRF that focuses on view synthesis from sparse inputs. The authors propose a method that introduces regularization techniques to the NeRF framework to improve its performance when synthesizing views from a limited number of input images. The training paradigm involves optimizing the neural network using a combination of rendering loss and regularization losses (e.g., smoothness and sparsity losses). RegNeRF is suitable for applications like 3D modeling, computer graphics, and virtual reality, where view synthesis from sparse input data is a common challenge.
- **Zero-Shot Text-Guided Object Generation with Dream Fields (also known as Dream Fields)** [94] is an application of NeRF for text-guided object generation. The authors propose a method that generates 3D objects conditioned on a given textual description. The training paradigm involves optimizing a neural network with a combination of rendering loss and textual conditioning loss. Applications include 3D object generation, virtual world creation, and content generation based on textual input. It extends NeRF by integrating textual guidance into the framework.
- **Efficient NeRF with learned depth-guided sampling** [130] is an improvement over NeRF that focuses on speeding up the rendering process by incorporating learned depth-guided sampling. The training paradigm

involves optimizing the neural network using a combination of rendering loss and a depth-guided sampling loss. Applications include real-time rendering, computer graphics, and virtual reality. This method improves NeRF’s efficiency by learning depth-guided sampling strategies.

- **Learning Neural Light Fields with Ray-Space Embedding Networks** [6] is an extension of NeRF for learning neural light fields. The authors propose a method that learns a compact representation of light fields using a ray-space embedding network. The training paradigm involves optimizing the neural network using a rendering loss. Applications include light field rendering, computer graphics, and virtual reality. It extends NeRF by using a compact representation for light fields.
- **Neural Head Avatars** [67] is an application of NeRF that focuses on generating 3D head avatars from monocular RGB videos. The authors propose a method that reconstructs neural radiance fields for human heads from single-view RGB video input. The training paradigm involves optimizing the neural network using a combination of rendering loss and additional losses for facial details. Applications include virtual avatars, video conferencing, and virtual reality. It extends NeRF by applying it to human head reconstruction from single-view video input.
- **NeRF-SR** [220] is an improvement over NeRF that focuses on generating high-quality neural radiance fields using super-sampling. The authors propose a method that employs super-sampling during the training and rendering processes to improve the quality of the generated views. The training paradigm involves optimizing the neural network using a rendering loss. Applications include high-quality rendering, computer graphics, and virtual reality. It improves NeRF’s rendering quality by incorporating super-sampling.
- **MoFaNeRF (Morphable Neural Radiance Fields)** [272] is an extension of NeRF that focuses on creating morphable neural radiance fields. The authors propose a method that learns a single, unified neural radiance field capable of representing multiple object instances within a category. The training paradigm involves optimizing the neural network using a combination of rendering loss and morphing loss. Applications include 3D object generation, animation, and computer graphics. It extends NeRF by learning a single representation capable of representing multiple object instances.
- **HumanNeRF (Generalizable Neural Human Radiance Field from Sparse inputs)** [263] is an extension of NeRF that focuses on reconstructing human shapes and appearance from sparse input views. The authors propose a method that leverages skeletal and semantic constraints to improve generalization and efficiency. The training paradigm involves optimizing the neural network using a combination of rendering loss and additional losses for skeletal and semantic constraints. Applications include human shape reconstruction, animation, and computer graphics. It extends NeRF by incorporating human-specific constraints.

- **Dense Depth Priors for Neural Radiance Fields from Sparse Input Views** [182] is an improvement over NeRF that incorporates dense depth priors to handle sparse input views. The authors propose a method that leverages dense depth information as a prior to improve the quality and stability of the reconstructed scene. The training paradigm involves optimizing the neural network using a combination of rendering loss and depth prior loss. Applications include scene reconstruction from sparse views and computer graphics. It improves NeRF by incorporating dense depth priors.
- **CG-NeRF (Conditional Generative Neural Radiance Fields)** [100] is an extension of NeRF that introduces conditional generative neural radiance fields. The authors propose a method that conditions the neural radiance field on external information, such as object attributes or scene context. The training paradigm involves optimizing the neural network using a combination of rendering loss and conditioning loss. Applications include conditional scene generation and computer graphics. It extends NeRF by introducing conditional generation capabilities.
- **Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields** [217] is an extension of NeRF that focuses on modeling structured view-dependent appearance. The authors propose a method that explicitly models view-dependent reflections and refractions in the scene. The training paradigm involves optimizing the neural network using a rendering loss and additional losses for view-dependent appearance. Applications include rendering scenes with complex reflections and refractions, and computer graphics. It extends NeRF by modeling structured view-dependent appearance.
- **Geometry-Guided Progressive NeRF** [28] is an improvement over NeRF that focuses on efficient neural human rendering. The authors propose a method that leverages geometric priors and a progressive training strategy to improve rendering efficiency and generalization. The training paradigm involves optimizing the neural network using a rendering loss and additional losses for geometric constraints. Applications include human rendering, animation, and computer graphics. It improves NeRF by incorporating geometric priors and a progressive training strategy.
- **Deep Visual Constraints (Neural Implicit Models for Manipulation Planning from Visual Input)** [75] is an application of NeRF that focuses on manipulation planning using visual input. The authors propose a method that learns an implicit model of the scene and integrates it into a planning framework. The training paradigm involves optimizing the neural network using a combination of rendering loss and planning loss. Applications include robotic manipulation planning, computer vision, and robotics. It extends NeRF by applying it to manipulation planning from visual input.
- **CLIP-NeRF** [219] is an extension of NeRF that combines textual and visual information for scene manipulation. It leverages the CLIP (Contrastive Language-Image Pretraining) model to condition the neural ra-

diance fields on both text and image inputs. The training paradigm involves optimizing the neural network using a combination of rendering loss and CLIP loss. Applications include text-and-image-based scene editing and computer graphics. It extends NeRF by introducing text-and-image-driven manipulation capabilities.

- **Neural Radiance Fields for Outdoor Scene Relighting** [185] is an application of NeRF that focuses on relighting outdoor scenes. The authors propose a method that models outdoor lighting conditions and captures view-dependent appearance, allowing for relighting under various conditions. The training paradigm involves optimizing the neural network using a rendering loss and additional losses for lighting and appearance. Applications include outdoor scene rendering, virtual reality, and computer graphics. It applies NeRF to outdoor scene relighting.
- **CityNeRF** [237] is an extension of NeRF that scales the representation to large-scale city scenes. The authors propose a method that leverages a hierarchical structure and efficient rendering techniques to handle city-scale data. The training paradigm involves optimizing the neural network using a rendering loss and additional losses for spatial coherence and efficiency. Applications include city-scale scene rendering, mapping, and urban planning. It extends NeRF by scaling the representation to city-scale scenes.
- **HeadNeRF (A Real-Time NeRF-based Parametric Head Model)** [86] is an application of NeRF that focuses on real-time head modeling. The authors propose a method that combines a parametric head model with a neural radiance field, enabling real-time rendering and animation of head models. The training paradigm involves optimizing the neural network using a rendering loss and additional losses for parametric constraints. Applications include real-time head rendering, animation, and computer graphics. It applies NeRF to real-time head modeling.
- **GRAM (Generative Radiance Manifolds)** [48] is an extension of NeRF that introduces generative radiance manifolds for 3D-aware image generation. The authors propose a method that leverages a manifold-based representation to model scene geometry and appearance, enabling high-quality image synthesis. The training paradigm involves optimizing the neural network using a combination of rendering loss and manifold loss. Applications include 3D-aware image generation and computer graphics. It extends NeRF by introducing generative radiance manifolds.
- **Solving Inverse Problems with NeRFGANs** [44] is an extension of NeRF that combines the power of NeRF with GANs to solve inverse problems. The authors propose a method that uses NeRF as a generator and a discriminator to optimize the solution for various inverse problems. The training paradigm involves optimizing the neural network using a combination of rendering loss and adversarial loss. Applications include solving inverse problems in computer graphics and computer vision. It extends NeRF by integrating it with GANs to solve inverse problems.

- **HVTR (Hybrid Volumetric-Textural Rendering)** is an extension of NeRF that combines volumetric and textural rendering for human avatars. The authors propose a method that jointly learns geometric, appearance, and textural information, enabling more realistic rendering of human avatars. The training paradigm involves optimizing the neural network using a combination of rendering loss, texture loss, and geometric loss. Applications include human avatar rendering, animation, and virtual reality. It extends NeRF by introducing hybrid volumetric-textural rendering.
- **Mega-NeRF** [216] is an extension of NeRF that scales the representation to large-scale scenes, suitable for virtual fly-throughs. The authors propose a method that leverages a hierarchical structure and efficient rendering techniques to handle large-scale data. The training paradigm involves optimizing the neural network using a rendering loss and additional losses for spatial coherence and efficiency. Applications include large-scale scene rendering, virtual reality, and computer graphics. It extends NeRF by scaling the representation to large-scale scenes.
- **3D-Aware Image Synthesis via Learning Structural and Textural Representations** [240] is an application of NeRF that focuses on 3D-aware image synthesis. The authors propose a method that learns structural and textural representations to synthesize high-quality images while considering 3D geometry. The training paradigm involves optimizing the neural network using a combination of rendering loss, structure loss, and texture loss. Applications include 3D-aware image synthesis, computer graphics, and computer vision. It applies NeRF to 3D-aware image synthesis.
- **Learning Implicit Body Representations from Double Diffusion Based Neural Radiance Fields** [245] is an extension of NeRF that focuses on learning implicit body representations from double diffusion-based neural radiance fields. The authors propose a method that leverages a double diffusion process to capture fine-scale details and appearance information. The training paradigm involves optimizing the neural network using a combination of rendering loss, diffusion loss, and additional losses for geometry and appearance. Applications include human body modeling, animation, and virtual reality. It extends NeRF by introducing double diffusion-based neural radiance fields.
- **BANMo (Building Animatable 3D Neural Models)** [243] is an application of NeRF that focuses on building animatable 3D neural models from casual videos. The authors propose a method that leverages spatiotemporal information to learn animatable neural models that can be applied to a variety of video inputs. The training paradigm involves optimizing the neural network using a combination of rendering loss, temporal coherence loss, and additional losses for animation and appearance. Applications include video-based animation, computer graphics, and virtual reality. It applies NeRF to animatable 3D neural models.

- **InfoNeRF (Ray Entropy Minimization for Few-Shot Neural Volume Rendering)** [108] is an improvement over NeRF that focuses on few-shot neural volume rendering. The authors propose a method that minimizes ray entropy to improve the quality of rendered images when trained with limited input views. The training paradigm involves optimizing the neural network using a combination of rendering loss and ray entropy loss. Applications include few-shot neural volume rendering, computer graphics, and computer vision. It improves NeRF by introducing ray entropy minimization for few-shot rendering. (Work in Progress...)
 - **Research in 2022:** This subsection briefly discusses the research articles in the last year. (Work in Progress...)
 - **Research in 2023:** This subsection briefly discusses the research articles in this year till the month of April. (Work in Progress...)
- (Work in Progress...)

5 Supporting Datasets

This section discusses the datasets that have been generated or directly used for different types of tasks and evaluation purposes for a comprehensive review of Radiance Fields and similar research ideas that are discussed later in this survey. These comprise of the following.

- **LLFF (Light Field for the Language of Light and Form) dataset** provides the real-world scenes at 4K ultra-high resolution of training views [147]. This dataset comprises of 8 forward-facing scenes with each scene having views between 20 views and 60 views. These scenes were (4032×3024) while NeRF-based methods use scenes with resolution (1008×756) for training the MLPs and rendering (inference) from them. Further, NeRF-based methods also use COLMAP [186,187] for estimating the corresponding camera poses. This dataset has been used for the following research articles:
 - Original NeRF (Neural Radiance Fields) [148]
 - NeRF-W (Neural Radiance Fields in the Wild) [140]
- **Tanks & Temples dataset** was introduced by Knapitsch et al [110] in 2017. It is widely used benchmark for evaluating 3D reconstruction algorithms. The dataset consists of a diverse set of High-Resolution RGB-D (color and depth) scans of light-scale indoor and outdoor scenes, captured using a high-quality structured light scanner. The scenes in the dataset are categorized based on “Tanks” (for indoor scenes), and “Temples” (for outdoor scenes). The dataset provides ground truth 3D models of the scenes, which are reconstructed from the RGB-D scans using a state-of-the-art 3D reconstruction pipeline. These ground truth models can be used to quantitatively evaluate the performance of 3D reconstruction algorithms by comparing their output against the reference models. It also comprises of evaluation metrics, including “mean per-point distance” between the reconstructed and

reference models, and “completeness”, which measures the percentage of the reference model that is covered by the reconstructed model. This dataset is used in research to evaluate and compare performance of various 3D reconstruction algorithms, including MVS, Volumetric Fusion, and Learning-based approaches such as NeRF and its variants. By providing a challenging and diverse set of scenes with ground truth data, the Tanks and Temples dataset enables the development and assessment of new 3D reconstruction methods in a standardized and reproducible manner.

This dataset has been used for the following research articles:

- IDR (Implicit Differentiable Renderer) [247]
- MVSNeRF (Multi-View Stereo Neural Radiance Fields) [25]

– **ScanNet or ‘Studio’ dataset** [42] is a large-scale dataset for 2.5D and 3D understanding tasks, introduced in 2017. The dataset contains RGB-D scans of more than 1500 indoor scenes, captured using a consumer-grade depth sensor (Microsoft Kinect). The scenes include various residential and office environments, covering a wide range of object categories and room layouts. It provides rich annotations for each scan, including instance-level semantic segmentation, object bounding boxes, and object-level 3D CAD model alignments. Additionally, camera poses and camera calibration parameters are provided, enabling the reconstruction of the 3D scene structure from the RGB-D scans. It has been widely used for various 3D understanding tasks, such as semantic segmentation, 3D object detection, and scene classification. It has also been used to train and evaluate 3D reconstruction algorithms, including learning-based methods like Neural Radiance Fields (NeRF) and its variants. For neural rendering-based applications, ScanNet serves as an important source of data to train models that can understand and reconstruct indoor scenes. By leveraging the rich annotations provided by ScanNet, researchers can develop and evaluate methods that perform tasks such as view synthesis, relighting, and object insertion or removal. The availability of camera poses and calibration parameters allows neural rendering models to incorporate geometric constraints and multi-view consistency during training, leading to improved performance and generalization.

For neural rendering-based applications, ScanNet serves as an important source of data to train models that can understand and reconstruct indoor scenes. By leveraging the rich annotations provided by ScanNet, researchers can develop and evaluate methods that perform tasks such as view synthesis, relighting, and object insertion or removal. The availability of camera poses and calibration parameters allows neural rendering models to incorporate geometric constraints and multi-view consistency during training, leading to improved performance and generalization.

This dataset has been used for the following research articles:

- GIRAFFE (Compositional Generative Radiance Fields) [160]
 - DeRF (Decomposed Radiance Fields) [177]
- **Synthetic-NeRF dataset** [140], introduced by Martin-Brualla et al. in the paper ”NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections,” is a collection of synthetic scenes created to train and evaluate

Neural Radiance Field (NeRF) models and their variants. The dataset consists of several synthetic scenes, each containing a 3D object rendered with different camera viewpoints, lighting conditions, and backgrounds. By using synthetic data, the dataset provides precise ground truth geometry, camera poses, and lighting parameters, which are difficult to obtain for real-world scenes. It is valuable for training and evaluating NeRF models, as it allows researchers to assess the performance of their methods under controlled conditions with known ground truth. This helps in understanding the limitations and strengths of the models, as well as identifying areas for improvement. The dataset is also useful for studying the impact of various factors, such as lighting, camera viewpoint, and object complexity, on the performance of NeRF-based methods.

For 3D reconstruction and neural rendering applications, Synthetic NeRF dataset serves as a benchmark to evaluate the quality of the generated reconstructions and renderings. The ground truth geometry and camera parameters enable quantitative evaluation of the reconstructed 3D models and synthesized views, allowing for a fair comparison between different methods. This helps in advancing the state-of-the-art in 3D reconstruction and neural rendering by identifying the best-performing models and techniques.

This dataset has been used for the following research articles:

- GIRAFFE (Compositional Generative Radiance Fields) [160]
- PixelNeRF [250]

– **Blender Dataset** is a collection of synthetic scenes created using the open-source 3D software Blender. The dataset is commonly used to train and evaluate neural rendering models, including NeRF and its variants, for tasks such as 3D reconstruction and view synthesis. It typically consists of a variety of scenes, each containing 3D objects rendered with different camera viewpoints, lighting conditions, and backgrounds. By using synthetic data, the dataset provides precise ground truth geometry, camera poses, and lighting parameters that are difficult to obtain for real-world scenes.

The Blender dataset is valuable for training and evaluating neural rendering models, as it allows researchers to assess the performance of their methods under controlled conditions with known ground truth. This helps in understanding the limitations and strengths of the models, as well as identifying areas for improvement. The dataset is also useful for studying the impact of various factors, such as lighting, camera viewpoint, and object complexity, on the performance of neural rendering-based methods.

For 3D reconstruction and neural rendering applications, the Blender dataset serves as a benchmark to evaluate the quality of the generated reconstructions and renderings. The ground truth geometry and camera parameters enable quantitative evaluation of the reconstructed 3D models and synthesized views, allowing for a fair comparison between different methods. This helps in advancing the state-of-the-art in 3D reconstruction and neural rendering by identifying the best-performing models and techniques.

– **ShapeNet dataset** [23] is a large-scale collection of 3D object models, introduced in 2015. The dataset contains over 3 million 3D models across 55

categories, spanning a wide range of object types and complexity levels. The models in ShapeNet are represented as watertight 3D meshes, and many of them are annotated with additional information, such as semantic labels, part hierarchies, and alignment with 2D images. It is widely used for various 3D understanding tasks, including 3D shape classification, part segmentation, and 3D object detection. It has also been utilized for 3D reconstruction tasks, where researchers train models to predict the 3D shape of objects from input data such as 2D images, depth maps, or point clouds.

In the context of neural rendering applications such as Neural Radiance Fields (NeRF) and its variants, ShapeNet can be used to create synthetic datasets for training and evaluation. By rendering the 3D models in ShapeNet with different camera viewpoints, lighting conditions, and backgrounds, researchers can generate ground truth data for tasks like view synthesis and 3D reconstruction. This allows for the development and evaluation of NeRF-based methods under controlled conditions with known ground truth geometry and camera poses. This dataset enables the study of various factors that affect the performance of NeRF-based methods, such as object complexity, lighting, and camera viewpoint. Additionally, the dataset allows for the evaluation of NeRF models' ability to generalize across different object categories and shape variations.

- **Kubric Synthetic Dataset:** The authors for the FORGE (Few-view Object Reconstruction that GEneralizes) [99] constructed this dataset for computer vision community. However, Kubric is an open-source software package that enables the creation of custom synthetic datasets by rendering 3D scenes. By leveraging Kubric, researchers and practitioners can generate synthetic datasets tailored to their specific needs, including datasets for 3D reconstruction and neural rendering applications such as Neural Radiance Fields (NeRF) and its variants. This dataset allows users to define 3D scenes with various objects, materials, lighting conditions, and camera poses. The software package is built on top of the Blender rendering engine, which supports both real-time and offline, high-quality rendering. Kubric provides a Python API to programmatically create and manipulate 3D scenes, making it easy to generate large amounts of data with diverse variations.

For 3D reconstruction and neural rendering applications, custom synthetic datasets created using Kubric can be used for training and evaluation purposes. Researchers can render the defined 3D scenes with different camera viewpoints, lighting conditions, and backgrounds to generate ground truth data for tasks like view synthesis and 3D reconstruction. This allows for the development and evaluation of NeRF-based methods under controlled conditions with known ground truth geometry and camera poses. Using Kubric-generated synthetic datasets enables the study of various factors that affect the performance of NeRF-based methods, such as object complexity, lighting, and camera viewpoint. Additionally, the custom datasets allow for the evaluation of NeRF models' ability to generalize across different object categories and shape variations.

This dataset has been used for the following research articles:

- **GIRAFFE** (Compositional Generative Neural Feature Fields) [160]
- **ETH-XGaze**, introduced by [260], consists of high-resolution images with various head poses and gaze directions, and has been acquired by leveraging a multi-view camera system under different lighting conditions. It contains more than 1 million annotated images of 80 subjects, captured using a custom multi-camera rig in a variety of head poses and gaze directions. The training set consists of 80 subjects with a total of 756K frames. Each of these frames is composed of 18 different view camera view images. The test-set (also person-specific) contains 15 subjects with total of two hundred images for each subject with ground truth gaze labels. The dataset provides detailed annotations, including 3D gaze vectors, 3D head poses, 2D eye landmarks, and facial landmark locations.
 While the primary purpose of the ETH-XGAZE dataset is for gaze estimation, it can also be used for related computer vision tasks, such as head pose estimation and facial landmark detection. However, the dataset is not specifically designed for 3D reconstruction or neural rendering applications like Neural Radiance Fields (NeRF) and its variants.
 Nevertheless, ETH-XGAZE could potentially be used as a source of data for training and evaluating NeRF-based models on tasks involving facial geometry and appearance. The multi-view nature of the dataset and the availability of head pose information may allow researchers to develop models that can reconstruct 3D facial geometry or perform view synthesis of faces under various head poses and gaze directions. Additionally, the dataset could be used to study the effects of different head poses and gaze directions on the performance of NeRF-based methods.
- **MPII Gaze dataset** [262] is a dataset primarily designed for gaze estimation. It contains more than 213,000 images of 337 subjects, captured in natural, everyday settings using a variety of devices such as webcams, laptops, and mobile phones. The dataset provides annotations for 3D gaze vectors and head poses, as well as 2D eye landmarks and facial landmark locations. While the primary purpose of the MPII Gaze dataset is for gaze estimation, it can also be used for related computer vision tasks, such as head pose estimation and facial landmark detection. However, the dataset is not specifically designed for 3D reconstruction or neural rendering applications like Neural Radiance Fields (NeRF) and its variants.
 Nevertheless, MPII Gaze could potentially be used as a source of data for training and evaluating NeRF-based models on tasks involving facial geometry and appearance. The availability of head pose information and the diverse set of images captured in natural settings may allow researchers to develop models that can reconstruct 3D facial geometry or perform view synthesis of faces under various head poses and gaze directions. Additionally, the dataset could be used to study the effects of different head poses and gaze directions on the performance of NeRF-based methods.
- **MPIIFaceGaze**, curated by Zhang et al [262] is based on gaze estimation dataset from MPIIGaze [261]. MPIIFaceGaze contains 3000 face images with 2-D gaze labels for every 15 subjects. While the primary purpose of

the MPIIFaceGaze dataset is for gaze estimation, it can also be used for related computer vision tasks, such as head pose estimation and facial landmark detection. However, the dataset is not specifically designed for 3D reconstruction or neural rendering applications like Neural Radiance Fields (NeRF) and its variants.

Nevertheless, MPIIFaceGaze could potentially be used as a source of data for training and evaluating NeRF-based models on tasks involving facial geometry and appearance. The availability of head pose information and the diverse set of images captured in natural settings may allow researchers to develop models that can reconstruct 3D facial geometry or perform view synthesis of faces under various head poses and gaze directions. Additionally, the dataset could be used to study the effects of different head poses and gaze directions on the performance of NeRF-based methods.

- **ColumbiaGaze** (curated by Smith et al [198]) is a dataset composed of 5,880 high-resolution images from 56 people, captured in controlled laboratory conditions using a single high-resolution camera. The dataset provides annotations for head pose and gaze direction, as well as eye and facial landmark locations. The images were acquired with 5-fixed head poses and 21 fixed gaze directions per pose. While the primary purpose of the Columbia Gaze dataset is for gaze estimation, it can also be used for related computer vision tasks, such as head pose estimation and facial landmark detection. However, the dataset is not specifically designed for 3D reconstruction or neural rendering applications like Neural Radiance Fields (NeRF) and its variants.
Nevertheless, Columbia Gaze could potentially be used as a source of data for training and evaluating NeRF-based models on tasks involving facial geometry and appearance. The availability of head pose information and the high-resolution images captured under controlled conditions may allow researchers to develop models that can reconstruct 3D facial geometry or perform view synthesis of faces under various head poses and gaze directions. Additionally, the dataset could be used to study the effects of different head poses and gaze directions on the performance of NeRF-based methods.
- **GazeCapture dataset** is a dataset with different poses for different amounts of illumination conditions and obtained through a crowd-sourcing endeavor [114]. Its text set only contains 150 people. The dataset contains more than 2.4 million images of over 1,500 subjects, captured using the front-facing cameras of various iOS devices in natural, everyday settings. The dataset provides annotations for 2D gaze coordinates on the device screen, along with facial landmark locations and head pose information. While the primary purpose of the GazeCapture dataset is for gaze estimation, it can also be used for related computer vision tasks, such as head pose estimation and facial landmark detection. However, the dataset is not specifically designed for 3D reconstruction or neural rendering applications like Neural Radiance Fields (NeRF) and its variants.
Nevertheless, GazeCapture could potentially be used as a source of data for training and evaluating NeRF-based models on tasks involving facial

geometry and appearance. The availability of head pose information and the diverse set of images captured in natural settings may allow researchers to develop models that can reconstruct 3D facial geometry or perform view synthesis of faces under various head poses and gaze directions. Additionally, the dataset could be used to study the effects of different head poses and gaze directions on the performance of NeRF-based methods.

- **DTU Robot Image Dataset**, created by the Technical University of Denmark (DTU), is a dataset designed for multi-view stereo (MVS) and 3D reconstruction tasks. It consists of 60 scenes captured using a robotic arm and a high-resolution camera. The dataset provides images, camera parameters (intrinsic and extrinsic), and dense ground-truth point clouds for each scene, making it an excellent resource for evaluating 3D reconstruction and neural rendering methods.

The usage of the DTU Robot Image Dataset for 3D reconstruction and different neural rendering applications such as Neural Radiance Fields (NeRF) includes:

1. **Training:** Researchers can use the DTU dataset to train NeRF-based models on a diverse set of scenes with varying complexities and geometries. The high-resolution images and accurate camera parameters facilitate learning and generalization of the models across different scenes.
2. **Evaluation:** The DTU dataset can be used to evaluate the performance of NeRF-based models on 3D reconstruction and view synthesis tasks. With dense ground-truth point clouds, researchers can quantitatively measure the accuracy of the reconstructed geometry and compare it with other methods.
3. **Benchmarking:** The DTU Robot Image Dataset serves as a valuable benchmark for comparing different 3D reconstruction and neural rendering methods. It has been widely used in the research community, allowing for fair comparisons between different methods in terms of reconstruction quality, rendering speed, and other performance metrics.

In summary, the DTU Robot Image Dataset is a useful resource for training, evaluating, and benchmarking 3D reconstruction and neural rendering methods such as NeRF. Its diverse set of scenes, high-resolution images, and accurate camera parameters provide a challenging testbed for developing and assessing the performance of various methods in reconstructing complex real-world scenes.

This dataset has been used for the following research articles:

- Original NeRF (Neural Radiance Fields) [148]
- MVSNeRF (Multi-View Stereo Neural Radiance Fields) [25]

- **CLEVR (Compositional Language and Elementary Visual Reasoning) dataset**, introduced by Johnson et al. in their paper "CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning," is designed for visual question answering (VQA) and scene understanding tasks. It contains images of 3D-rendered objects with varying shapes, colors, materials, and sizes, along with their corresponding scene graphs and semantic information.

Though CLEVR is not explicitly designed for 3D reconstruction or neural rendering, it can still be used for training and evaluating NeRF-based models and related applications. The potential usage of CLEVR for 3D reconstruction and different neural rendering applications such as Neural Radiance Fields includes:

- **Training:** Researchers can use the CLEVR dataset to train NeRF-based models on synthetic scenes with known ground truth geometry and appearance. The dataset provides diverse and complex object configurations, which can help models learn to represent and render various 3D structures.
- **Evaluation:** The CLEVR dataset can be used to evaluate the performance of NeRF-based models on tasks such as view synthesis, 3D reconstruction, and object-centric rendering. Given the ground truth scene graphs and semantic information, quantitative evaluation of the model performance is possible.
- **Object-Centric Neural Rendering:** Since the CLEVR dataset focuses on object-centric scenes, it can be particularly useful for NeRF-based methods that aim to learn object-centric radiance fields. Researchers can leverage the dataset's scene graphs and semantic information to develop models that can synthesize novel views of individual objects or object compositions.
- **Scene Understanding:** The CLEVR dataset can also be used to study the role of scene understanding in neural rendering applications. Researchers can explore how incorporating semantic information and scene graphs into NeRF-based models may improve their performance in rendering and view synthesis tasks.

While the CLEVR dataset is primarily designed for visual question answering and scene understanding tasks, it can be repurposed for 3D reconstruction and neural rendering applications involving NeRF. Its diverse and complex object configurations, along with the availability of ground truth geometry and semantic information, make it a valuable resource for training and evaluating NeRF-based models in object-centric rendering and scene understanding scenarios.

- **Matterport3D dataset**, introduced by Chang et al. in their paper "Matterport3D: Learning from RGB-D Data in Indoor Environments," is a large-scale dataset designed for various computer vision tasks, including 3D reconstruction, semantic segmentation, and scene understanding. The dataset contains 10,800 panoramic RGB-D images captured from 90 different indoor scenes, such as homes, offices, and public spaces. It provides high-resolution images, depth data, camera poses, and a 3D mesh reconstruction for each scene, making it a valuable resource for 3D reconstruction and neural rendering applications.

The usage of the Matterport3D dataset for 3D reconstruction and different neural rendering applications such as Neural Radiance Fields (NeRF) includes:

1. Training: Researchers can use the Matterport3D dataset to train NeRF-based models on a diverse set of indoor scenes with complex geometries and lighting conditions. The high-resolution images, depth data, and accurate camera poses facilitate learning and generalization across various scenes and environments.
2. Evaluation: The Matterport3D dataset can be used to evaluate the performance of NeRF-based models on tasks such as view synthesis, 3D reconstruction, and handling occlusions in indoor environments. The dataset's 3D mesh reconstructions enable quantitative evaluation of the model's performance in reconstructing complex scenes.
3. Benchmarking: The Matterport3D dataset serves as a valuable benchmark for comparing different 3D reconstruction and neural rendering methods in indoor environments. By testing on the same dataset, researchers can compare their methods against others in terms of reconstruction quality, rendering speed, and other performance metrics.
4. Scene Understanding: The Matterport3D dataset can also be used to study the role of scene understanding in neural rendering applications. Researchers can explore how incorporating semantic information from the dataset's semantic segmentation labels into NeRF-based models may improve their performance in rendering and view synthesis tasks.

The Matterport3D dataset is a useful resource for training, evaluating, and benchmarking 3D reconstruction and neural rendering methods such as NeRF, particularly in indoor environments. Its diverse set of scenes, high-resolution images, depth data, and camera poses provide a challenging testbed for developing and assessing the performance of various methods in reconstructing complex real-world indoor scenes.

This dataset has been used for the following research articles:

- GIRAFFE (Compositional Generative Neural Feature Fields) [160]
- DeRF (Decomposed Radiance Fields) [177]

(Work in Progress...)

6 Objective Function

Since objective functions (or mostly known as Cost or Loss function) in Machine learning play an integral part for training the model for different applications. Here, we discuss the

- **Functional Loss** is a generic term in the context of neural rendering and NeRF models. However, it can refer to any loss function that captures the difference between the target functionality or behavior of the generated data and the ground truth data. The equation for a functional loss would depend on the specific application and the desired behavior or property that needs to be preserved. Functional loss could potentially be used to ensure that the generated images or 3D representations have certain desired properties or behavior, such as smoothness, continuity, or consistency across different

views. It has different applications in the domain of Computer Vision and Neural Rendering such as the following.

- **Neural Rendering:** Functional loss could be used in NeRF models to enforce specific properties, such as temporal consistency in dynamic scenes or smoothness in the generated images.
- **Image-to-Image Translation:** In tasks like style transfer, functional loss could be used to ensure that the generated images maintain certain properties, such as preserving the structure of the input image or generating images with the desired style or texture.
- **3D Reconstruction:** Functional loss could be used in 3D reconstruction tasks to enforce geometric or topological properties, such as smoothness, compactness, or the preservation of sharp features.

As functional loss is a broad term and its definition varies depending on the application, it's difficult to pinpoint specific NeRF models, architectures, or extensions that use this loss function. However, it's worth noting that many NeRF-based models incorporate additional loss functions alongside reconstruction loss to enforce specific properties or behavior. For example, NeRF-W [140] uses perceptual loss in addition to reconstruction loss to improve the quality of the generated images. Other NeRF models may incorporate similar loss functions to enforce the desired properties or behavior.

- **Reconstruction Loss (or Photometric Loss),** is a measure of the discrepancy between the predicted image and the ground truth image. It is commonly used in image synthesis, neural rendering, and other computer vision tasks. In the context of NeRF, reconstruction loss is used to penalize the difference between the predicted colors and the ground truth colors for a given set of rays. The reconstruction loss is usually calculated as the MSE between the predicted and ground truth colors, averaged over all pixels in the image:

$$L_{recons} = \frac{1}{N} \sum ||C_{pred} - C_{gt}||^2 \quad (87)$$

where N is the number of pixels, C_{pred} is the predicted color, and C_{gt} is the ground truth color.

Applications of this function in the Neural Rendering and other tasks include:

- **Neural Rendering:** Reconstruction loss is widely used to train NeRF models and their variants. It ensures that the predicted colors align well with the ground truth colors, leading to accurate and realistic view synthesis.
- **Image-to-Image Translation:** In tasks such as style transfer, reconstruction loss is used to preserve the content of the input image while applying the desired style.
- **Depth Estimation:** Reconstruction loss can be used in combination with depth consistency loss to ensure the predicted depth maps align with the ground truth depth maps.

Almost all NeRF models, architectures, and extensions use reconstruction loss as their primary loss function, including but not limited to:

- The original NeRF (Neural Radiance Fields) [148]
- NeRF-W (NeRF in the Wild) [140]
- MVSNeRF (Multi-View Stereo Neural Radiance Fields) [25]
- GIRAFFE (Compositional Generative Neural Feature Fields) [160]
- DeRF (Decomposed Radiance Fields) [177]
- NeRFies [166]
- PixelNeRF [250]

Reconstruction loss is a fundamental component of the training process for NeRF-based models, as it helps the models learn to generate accurate and realistic images that match the ground truth data.

- **Perceptual Loss**, also known as feature loss or content loss, is a measure of the discrepancy between the high-level features of the predicted image and the ground truth image. It is commonly used in image synthesis, neural rendering, and other computer vision tasks where preserving high-level features is important. Perceptual loss is often computed using pre-trained deep neural networks, such as VGG or ResNet, as feature extractors.
The perceptual loss is usually calculated as the MSE between the high-level features of the predicted and ground truth images, extracted from the pre-trained network as shown in the eqn 88

$$L_{percp} = \frac{1}{N} \sum ||F(C_{pred}) - F(C_{gt})||^2 \quad (88)$$

where N is the number of pixels, $F(\cdot)$ is the feature extraction function, C_{pred} is the predicted color, and C_{gt} is the ground truth color.

Applications of this function in the Neural Rendering and other tasks include:

- **Neural Rendering:** Perceptual loss can be used in NeRF models and their variants to ensure that the predicted images preserve high-level features and structures, leading to more perceptually pleasing view synthesis.
- **Image-to-Image Translation:** In tasks such as style transfer or image super-resolution, perceptual loss is used to ensure that the translated images maintain the content and structure of the input image while applying the desired style or enhancing the resolution.
- **GANs:** Perceptual loss is often used in GANs to help generate images that are more visually pleasing and have more realistic high-level features.

While perceptual loss is not as commonly used in NeRF-based models as reconstruction loss, there are some NeRF models and extensions that leverage perceptual loss to improve the quality of the generated images:

- NeRF-W (NeRF in the Wild) [140] uses a combination of reconstruction loss and perceptual loss to improve the quality of the generated images in unconstrained settings.
- Non Rigid NeRF [213] uses perceptual loss along with reconstruction loss to ensure that the generated images maintain the high-level structures and appearance of the ground truth data.

These are a few instances of NeRF models and extensions that use perceptual loss. The use of perceptual loss can help improve the visual quality of the generated images by focusing on high-level features, which are often more important for human perception.

- **Disentanglement Loss** aims to encourage a learned representation to capture distinct factors of variation in the data, such as geometry, appearance, illumination, or motion. This loss function is often used in unsupervised and self-supervised learning tasks, where ground truth labels for the factors of variation are not available.

The equation for disentanglement loss depends on the specific implementation and the factors of variation that need to be disentangled. One common approach for disentanglement loss is the β -VAE (Variational Autoencoder) framework, which modifies the standard VAE objective with an additional weighting factor β as shown in the eqn 89

$$L_{disent} = L_{reconst} + \beta * L_{KL} \quad (89)$$

Here, $L_{reconst}$ is the reconstruction loss, L_{KL} is the Kullback-Leibler divergence term that enforces the latent distribution to be close to a prior distribution (e.g., a standard Gaussian), and β is a hyperparameter that controls the trade-off between reconstruction and disentanglement.

Usage for Neural Rendering and other tasks are as follows.

- **Neural Rendering:** Disentanglement loss can be used in NeRF models and their variants to separate distinct factors of variation, such as geometry and appearance, enabling more controllable editing and manipulation of the generated scenes.
- **Unsupervised Representation Learning:** Disentanglement loss is often used in unsupervised learning tasks, such as autoencoders and VAEs, to learn meaningful and interpretable representations of the data.
- **Image-to-Image Translation:** Disentanglement loss can be used in style transfer tasks to separate content and style factors, enabling the transfer of one factor while preserving the other.

In the context of NeRF models, architectures, and extensions that use disentanglement loss or a similar concept:

- DeRF [177] explicitly disentangles the geometry and appearance factors in the learned radiance field representation, enabling more controllable editing and manipulation of the generated scenes.
- GIRAFFE [160] uses a compositional approach to separate object geometry and appearance, which allows for more flexible scene manipulation and editing.
- Object-Centric Neural Scene Rendering [73]: This work disentangles object geometry and appearance by learning separate latent spaces for each factor, enabling more controllable and intuitive scene editing.

These are a few examples of NeRF models and extensions that use disentanglement loss or a similar concept to separate distinct factors of variation in

the learned representations. Disentangling factors of variation can be beneficial for a variety of tasks, as it leads to more interpretable, controllable, and flexible representations.

- **Color Loss**, also known as color consistency loss, measures the difference between the predicted colors and the ground truth colors in a rendered image. In the context of Neural Radiance Fields (NeRF), color loss is essentially a part of the reconstruction loss, which measures the difference between the rendered radiance and the observed radiance from the input images. The color loss can be formulated using various metrics, with Mean Squared Error (MSE) being the most common.
For a given input ray r and a set of input images, NeRF models predict the radiance for each point along the ray. The color loss can be calculated as the mean squared error between the predicted color C_{pred} and the ground truth color C_{gt} as shown in the eqn 90

$$L_{color} = \|C_{pred} - C_{gt}\|^2 \quad (90)$$

Applications of this function in the Neural Rendering and other tasks include:

- **Neural Rendering:** In NeRF models and other neural rendering approaches, color loss is used to ensure that the generated images or 3D representations have accurate and consistent colors compared to the input images.
- **Image-to-Image Translation:** Color loss can be used in tasks like style transfer or image synthesis to ensure that the generated images maintain certain color properties from the input image or the target style.
- **3D Reconstruction:** Color loss can be used in 3D reconstruction tasks to measure the difference between the colors of the reconstructed 3D model and the ground truth model, encouraging accurate and detailed 3D reconstructions.

As mentioned earlier, color loss is essentially a part of the reconstruction loss in the context of NeRF models. Almost all NeRF models, including the original NeRF [148], NeRF-W [140], and many other NeRF extensions, employ some form of color loss as a part of the reconstruction loss to ensure that the generated images or 3D representations have accurate and consistent colors compared to the input images.

(Work in Progress...)

7 Evaluation Metrics

Since, we are discussing a comprehensive survey of paper, it is vital for readers to understand what are the metrics used to evaluate Radiance Fields. The main focus of this discussion is about PSNR (Peak Signal to Noise Ratio), SSIM (structural similarity index measure), LPIPS (Learned Perceptual Image Patch Similarity) [258], Chamfer distance, etc.

- **PSNR (Peak-Signal-to-Noise Ratio)** is a widely-used metric to evaluate image quality, particularly for image compression and restoration tasks. It measures the ratio between the maximum possible signal power (i.e., the maximum value of pixel intensity) and the power of the corrupting noise (i.e., the mean squared error between the original and reconstructed images). Higher PSNR values indicate better image quality. However, PSNR can be a poor indicator of perceptual similarity, as it does not capture human perception of image quality accurately.
- **SSIM (Structural Similarity Index)** is a more perceptually relevant metric that compares two images based on their luminance, contrast, and structure. It takes into account changes in pixel intensities, spatial dependencies, and contrasts in texture. SSIM ranges from -1 to 1, where 1 indicates a perfect match between the original and reconstructed images. SSIM provides a better assessment of image quality concerning human perception compared to PSNR.
- **LPIPS (Learned Perceptual Image Patch Similarity)** [258] is a metric to evaluate the perceptual similarity of the rendered views/poses when compared to the ground truth for that particular image from that particular viewing direction. It is a perceptual metric that uses deep learning to measure the similarity between two images. It is based on the features extracted from a pre-trained convolutional neural network (CNN) and is designed to align better with human judgments of image similarity. Lower LPIPS values indicate higher perceptual similarity between the images being compared. LPIPS is more robust to small geometric and textural differences and is better suited for evaluating generative models and image synthesis tasks.
- **NIQE (Natural Image Quality Evaluator)** is a metric for evaluating the quality of natural images. It measures the degree of naturalness of an image by comparing its statistics to those of a large database of natural images. The formula for NIQE is not publicly available, but it involves computing various image statistics such as the mean, standard deviation, and higher-order moments.
- **Chamfer Distance** is a geometric metric used to evaluate the similarity between two point sets, typically employed in 3D reconstruction tasks. It measures the average distance between each point in one set to its nearest neighbor in the other set. Lower Chamfer Distance values indicate better geometric alignment between the point sets. While it is not directly used to evaluate image quality in Radiance Fields, Chamfer Distance can be used to assess the quality of the underlying 3D geometry reconstructed by the model.
- **KID (Kernel Inception Distance)** is a metric used to evaluate the quality and diversity of generated images in generative models, such as GANs and VAEs. KID compares the feature distributions of real images and generated images by measuring the distance between the embeddings of these images in a feature space, typically obtained from an Inception network (a deep convolutional neural network pretrained on a large-scale image classification task, such as ImageNet).

The main idea behind KID is to measure the similarity between the distributions of real and generated images by comparing the statistics of their feature embeddings. The distance between these distributions is computed using the MMD (Maximum Mean Discrepancy) metric with a RBF (radial basis function) kernel.

To compute the KID, the following steps are typically taken:

1. **Feature Extraction:** Extract feature embeddings for both the real images and generated images using an Inception network. This involves passing the images through the Inception network and extracting the activations from one of the intermediate layers, which serve as the feature embeddings.
2. **Compute MMD:** Calculate the Maximum Mean Discrepancy (MMD) between the feature embeddings of the real and generated images. MMD is a metric that measures the distance between two probability distributions by comparing the means of samples drawn from these distributions. The MMD can be computed as shown in the eqn. 91.

$$MMD^2 = \|\mu_p - \mu_q\|^2 + 2 * \sum (k(p_i, q_j) - k(p_i, p_j) - k(q_i, q_j)) \quad (91)$$

where μ_p and μ_q are the means of the feature embeddings for the real images (p) and generated images (q), respectively, and $k(.,.)$ is the RBF kernel function applied to pairs of feature embeddings.

3. **Compute KID:** The KID is simply the MMD value computed in step 2.

KID has several advantages over other metrics for evaluating the quality and diversity of generated images, such as the Inception Score (IS) and Frechet Inception Distance (FID). For example, KID is unbiased, which means it provides a more reliable estimate of the true distance between the distributions of real and generated images, and it is less sensitive to the choice of the number of samples used for the computation. Additionally, KID is computationally efficient and can be calculated using fewer samples than other metrics, making it a popular choice for evaluating generative models in various research studies and applications.

- **FID (Frechet Inception Distance)** [84] is another metric used to assess generative models, comparing the feature distributions of real and generated images using the Inception-v3 network. FID captures both the quality and diversity of generated images and is considered a more reliable metric compared to Inception Score. Like Inception Score, FID can be applied to 2D projections of Radiance Fields-generated scenes to assess their visual quality and diversity.
- **ATE (Absolute Trajectory Error)** is a metric used to evaluate the accuracy of an estimated trajectory in comparison to a ground truth trajectory. ATE is commonly used in the evaluation of SLAM (Simultaneous Localization and Mapping) algorithms, visual odometry, and other robotics or computer vision tasks where a system's position and orientation are estimated over time.

ATE is computed by comparing the estimated trajectory and the ground truth trajectory, taking into account both the position and the orientation of the system at each time step. To compute the ATE, the following steps are typically taken:

1. Alignment: Since the estimated trajectory and the ground truth trajectory might have different scales, rotations, or translations, they need to be aligned before computing the error. This alignment is usually achieved by finding a transformation matrix (a combination of rotation, translation, and scale) that minimizes the difference between the two trajectories.
2. Error Calculation: After the alignment, the ATE is calculated as the RMSE (root mean squared error) between the aligned estimated trajectory and the ground truth trajectory. This involves computing the Euclidean distance between the corresponding positions in the two trajectories and averaging the squared distances over all time steps. Then, the square root of the average is taken to obtain the final ATE value.

Mathematically, given an estimated trajectory $P = p_1, p_2, \dots, p_n$ and a ground truth trajectory $Q = q_1, q_2, \dots, q_n$ (after alignment), the ATE can be calculated as shown in the eqn. 92.

$$ATE = \sqrt{\left(\frac{1}{n} * \sum ||p_i - q_i||^2 \right)} \quad (92)$$

where n is the number of time steps, p_i and q_i are the positions at time step i in the estimated and ground truth trajectories, respectively, and $||p_i - q_i||$ is the Euclidean distance between p_i and q_i .

ATE is a useful metric for evaluating the performance of various algorithms that estimate a system's trajectory over time, as it provides an overall measure of the accuracy and consistency of the estimated positions and orientations compared to the ground truth.

- **RPE (Relative Pose Error)** (RPE_r (Relative Rotation Error), RPE_t (Relative Translation Error)) is a metric used to evaluate the accuracy of an estimated trajectory in comparison to a ground truth trajectory, taking into account both the position and orientation of a system over a specific time interval. RPE is commonly used in the evaluation of SLAM (Simultaneous Localization and Mapping) algorithms, visual odometry, and other robotics or computer vision tasks where a system's position and orientation are estimated over time.

Unlike Absolute Trajectory Error (ATE), which compares the estimated and ground truth trajectories directly, RPE focuses on the relative changes in pose between pairs of time steps. This makes it a more suitable metric for evaluating systems where the global consistency of the trajectory is less important than the local consistency.

To compute the RPE, the following steps are typically taken:

1. Select Time Intervals: Choose a specific time interval (Δ) over which the relative pose changes will be evaluated. This can be a fixed interval or a variable one, depending on the application.

2. Compute Relative Poses: For both the estimated trajectory and the ground truth trajectory, compute the relative pose changes between pairs of poses separated by the chosen time interval.
3. Error Calculation: After computing the relative poses, calculate the error between the estimated relative poses and the ground truth relative poses. This can be done using various metrics, such as the Euclidean distance for position errors, and the angular distance for orientation errors. The overall RPE can then be calculated as the RMSE between the corresponding relative pose errors.

Mathematically, given an estimated trajectory $P = p_1, p_2, \dots, p_n$ and a ground truth trajectory $Q = q_1, q_2, \dots, q_n$, and a time interval delta, the RPE can be calculated as shown in the eqn 93

$$RPE = \text{sqrt} \left(\frac{1}{n - \Delta} * \sum (\|\Delta p_i - \Delta q_i\|^2) \right) \quad (93)$$

where n is the number of time steps, Δp_i and Δq_i are the relative pose changes between time steps i and $i + \Delta$ in the estimated and ground truth trajectories, respectively, and $\|\Delta p_i - \Delta q_i\|$ is the error metric used to measure the difference between the relative poses (e.g., Euclidean distance for position errors, angular distance for orientation errors).

RPE is a useful metric for evaluating the performance of various algorithms that estimate a system's trajectory over time, as it provides a measure of the local consistency of the estimated poses compared to the ground truth, which can be more informative than global consistency measures like ATE in certain applications.

- **AVI (Across View-direction Inconsistency)** was introduced in Bahat et al [9] to demonstrate the advantages of performing operations in the volumetric domain to produce renderings that are geometrically consistent for different viewing directions. AVI quantifies the pairwise-geometrical inconsistency between test frames adjacent to each other (in terms of viewing direction). This is done by leveraging Ground Truth Optical Flow extracted from every adjacent frame pairs in the synthetically rendered scene. The authors warp the first frame onto the second and calculate the induced per-pixel RMS difference across different channels to get per-pixel error map. Finally, the Optical Flow is used to mask-out map regions corresponding to the objects waning and waxing between two frames to compute an error map which can then be averaged to calculate AVI. The AVI can be denoted using the eq. 94 given below.

$$\text{AVI} = \frac{1}{n} \sum_{r \in \hat{i}_{sk}^j(r)} \text{Mask}_{j,j+1} \cdot \|\text{Warp}(\hat{i}_{sk}^j)(r) - \hat{i}_{sk}^{j+1}(r)\|_2 \quad (94)$$

where $\text{Warp}(\cdot)$ represents the warping operation in accordance with the Optical Flow, $\text{Mask}_{j,j+1}$ is binary mask to filter out inconsistent image regions, n is the number of pixels, and $\hat{i}_{sk}^j(r)$ and $\hat{i}_{sk}^{j+1}(r)$ depict the two adjacent

frames during Super Resolution. This metric was used to determine the consistency for the approach by Bahat et al [9] discussed later in this survey.

Both the optical flow for forward flow $f_{j \rightarrow j+1}$ and backward flow $f_{j+1 \rightarrow j}$ amongst all adjacent view-direction pair $j, j+1$. The warping function uses the backward optical flow $f_{j+1 \rightarrow j}$ between adjacent viewing directions by the eq. 95.

$$\text{Warp}(\hat{i}_{sk}^j)(r) = \hat{i}_{sk}^j(r + f_{j+1 \rightarrow j}(r)) \quad (95)$$

where r denotes the 2D pixel location.

The warping can produce ghosting artifacts due to occlusions (pixels in frame j with no correspondence in $j+1$). Masking prevents ghosting artifacts from dominating the AVI metric and hence an error mask was introduced. This is done by first counting the number of pixels matched using forward flow for the same pixel r as given by the eq. 96:

$$\text{Count}_{j,j+1}(r) = \#\{q | \lfloor q + f_{j \rightarrow j+1}(q) \rfloor = r\} \quad (96)$$

The pixels without any correspondence from viewing direction $j+1$ to viewing direction j is defined according to mask equation in eq. 97.

$$\tilde{\text{Mask}}_{j,j+1} = \begin{cases} 0, & \text{if } \text{Count}_{j,j+1}(r) = 0 \\ 1, & \text{otherwise} \end{cases} \quad (97)$$

The floor operation while computing Count can lead to ghosting operations due to sub-pixel occlusions, the authors apply morphological closing followed by erosion to the mask by leveraging cross structure element of unit radius. Hence, we get the final error mask as given in the eq. 98.

$$\text{Mask}_{j,j+1} = \text{erosion}(\text{closing}(\tilde{\text{Mask}}_{j,j+1})) \quad (98)$$

Hence, during the computation of AVI score, only unmasked elements are considered during computing mean. So, only the values for n as depicted in eq. 99 are considered in eq. 94

$$n = \#\{r | \text{Mask}_{j,j+1}(r) = 1\} \quad (99)$$

Figure 8 displays the effects of the raw masks and the different morphological and it can be seen that the same motion vectors and hence, same masks were used for computing AVI score for each method in the paper.

(Work in Progress...)

8 Different Explored Applications

This literature tries to list the different applications for NeRFs. Further, a rough statistics for the applications of Neural Radiance Fields have been shown in the fig 9. Further, we discuss these applications briefly. This can help build the basis for how the future of NeRFs and Neural Rendering.



Fig. 8: **Effects of Mask ($\tilde{M}_{j,j+1}$ or $\tilde{M}_{j,j+1}^{\text{diln}}$)**. To remove ghosting artifacts from the ship mast’s (warped image), masking was employed before the error calculation. (b-d) depicts the masking effect with/without processing and morphological image operators; (d) depicts the fully processed mask that yields significantly reduced ghosting artifacts

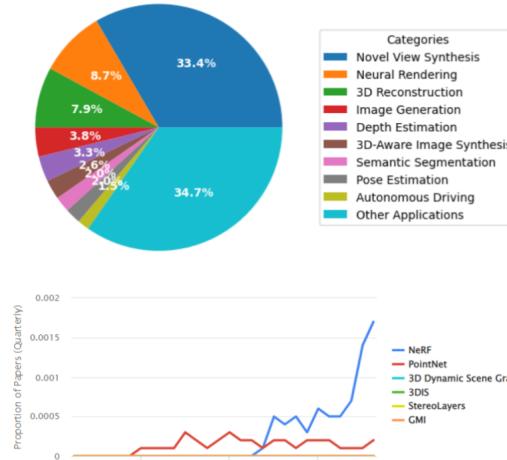


Fig. 9: The statistics related to NeRF-based models, extensions and improvements based on applications³

- **Shape Estimation** involves recovering the 3D structure of an object or scene from 2D images or point clouds. Applications include 3D modeling, reverse engineering, medical imaging, and robotics. Techniques such as stereo vision, photometric stereo, and depth sensing are used to estimate the shape of objects.
- **Pose Reconstruction** aims to recover the 3D position and orientation of an object or a camera in a coordinate system. This task is crucial for applications like robotics, autonomous navigation, augmented reality, and motion capture. Methods such as bundle adjustment, visual odometry, and SLAM can be used for pose reconstruction.

³ These statistics and graphs have been taken from <https://paperswithcode.com/method/nerf>

- **Pose Estimation** deals with estimating the position and orientation of a specific object or human body parts in 2D or 3D space. Applications include human-computer interaction, animation, sports analysis, and surveillance. Techniques used for pose estimation include keypoint detection, part-based models, and deep learning-based approaches.
- **View Reconstruction** aims to generate novel views of a scene or object from a set of input images. This technique is used in virtual reality, image-based rendering, and video synthesis. Methods such as light field rendering, image-based rendering, and neural rendering (e.g., NeRF) can be employed for view reconstruction.
- **Multi-View Reconstruction** is a 3D reconstruction technique that uses multiple images of a scene or object captured from different viewpoints. Applications include 3D modeling, computer-aided design, and cultural heritage preservation. Techniques such as SfM, MVS, and volumetric fusion are used for multi-view reconstruction.
- **Super-Resolution** is the process of enhancing the resolution and quality of an image by combining information from multiple low-resolution images or by learning a mapping from low-resolution to high-resolution images. Applications include satellite imaging, medical imaging, and video upscaling. Techniques such as sparse representation, deep learning-based methods (e.g., SRGAN), and multi-frame super-resolution can be used for this task.
- **Gaze Redirection** refers to the process of manipulating the appearance of a person’s eyes in an image or video to change the perceived gaze direction. This technique is useful in video conferencing, eye-tracking studies, and virtual reality. Approaches for gaze redirection include geometric transformations, image warping, and generative models (e.g., GANs).

(Work in Progress...)

9 Neural Radiance Fields in Future

(Work in Progress...)

10 Conclusion

(Work in Progress...)

References

1. Adamkiewicz, M., Chen, T., Caccavale, A., Gardner, R., Culbertson, P., Bohg, J., Schwager, M.: Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters* **7**(2), 4606–4613 (2022)
2. Aliev, K.A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural point-based graphics. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16. pp. 696–712. Springer (2020)

3. Arandjelović, R., Zisserman, A.: Nerf in detail: Learning to sample for view synthesis. arXiv preprint arXiv:2106.05264 (2021)
4. Atcheson, B., Ihrke, I., Heidrich, W., Tevs, A., Bradley, D., Magnor, M., Seidel, H.P.: Time-resolved 3d capture of non-stationary gas flows. ACM transactions on graphics (TOG) **27**(5), 1–9 (2008)
5. Athar, S., Shu, Z., Samaras, D.: Flame-in-nerf: Neural control of radiance fields for free view face animation. arXiv preprint arXiv:2108.04913 (2021)
6. Attal, B., Huang, J.B., Zollhöfer, M., Kopf, J., Kim, C.: Learning neural light fields with ray-space embedding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19819–19829 (2022)
7. Attal, B., Laidlaw, E., Gokaslan, A., Kim, C., Richardt, C., Tompkin, J., O’Toole, M.: Törf: Time-of-flight radiance fields for dynamic scene view synthesis. Advances in neural information processing systems **34**, 26289–26301 (2021)
8. Azinović, D., Martin-Brualla, R., Goldman, D.B., Nießner, M., Thies, J.: Neural rgb-d surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6290–6301 (2022)
9. Bahat, Y., Zhang, Y., Sommerhoff, H., Kolb, A., Heide, F.: Neural volume super-resolution. arXiv preprint arXiv:2212.04666 (2022)
10. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
11. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
12. Beier, T., Neely, S.: Feature-based image metamorphosis. ACM SIGGRAPH computer graphics **26**(2), 35–42 (1992)
13. Bergman, A., Kellnhofer, P., Wetzstein, G.: Fast training of neural lumigraph representations using meta learning. Advances in Neural Information Processing Systems **34**, 172–186 (2021)
14. Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Neural reflectance fields for appearance acquisition. arXiv preprint arXiv:2008.03824 (2020)
15. Blinn, J.F.: Simulation of wrinkled surfaces. ACM SIGGRAPH computer graphics **12**(3), 286–292 (1978)
16. Blinn, J.F., Newell, M.E.: Texture and reflection in computer generated images. Communications of the ACM **19**(10), 542–547 (1976)
17. Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.: Nerd: Neural reflectance decomposition from image collections. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12684–12694 (2021)
18. Boss, M., Jampani, V., Braun, R., Liu, C., Barron, J., Lensch, H.: Neural-pil: Neural pre-integrated lighting for reflectance decomposition. Advances in Neural Information Processing Systems **34**, 10691–10704 (2021)
19. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 425–432 (2001)
20. Bui, G., Le, T., Morago, B., Duan, Y.: Point-based rendering enhancement via deep learning. The Visual Computer **34**, 829–841 (2018)

21. Carranza, J., Theobalt, C., Magnor, M.A., Seidel, H.P.: Free-viewpoint video of human actors. *ACM transactions on graphics (TOG)* **22**(3), 569–577 (2003)
22. Chai, J.X., Tong, X., Chan, S.C., Shum, H.Y.: Plenoptic sampling. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. pp. 307–318 (2000)
23. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015)
24. Chen, A., Wu, M., Zhang, Y., Li, N., Lu, J., Gao, S., Yu, J.: Deep surface light fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* **1**(1), 1–17 (2018)
25. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14124–14133 (2021)
26. Chen, J., Zhang, Y., Kang, D., Zhe, X., Bao, L., Jia, X., Lu, H.: Animatable neural radiance fields from monocular rgb videos. *arXiv preprint arXiv:2106.13629* (2021)
27. Chen, J., Bautembach, D., Izadi, S.: Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (ToG)* **32**(4), 1–16 (2013)
28. Chen, M., Zhang, J., Xu, X., Liu, L., Feng, J., Yan, S.: Geometry-guided progressive nerf for generalizable and efficient neural human rendering. *arXiv preprint arXiv:2112.04312* (2021)
29. Chen, S.E., Williams, L.: View interpolation for image synthesis. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques. pp. 279–288 (1993)
30. Chen, S., Niu, S., Lan, T., Liu, B.: Pct: Large-scale 3d point cloud representations via graph inception networks with applications to autonomous driving. In: 2019 IEEE international conference on image processing (ICIP). pp. 4395–4399. IEEE (2019)
31. Chen, X., Zhang, Q., Li, X., Chen, Y., Feng, Y., Wang, X., Wang, J.: Hallucinated neural radiance fields in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12943–12952 (2022)
32. Chiang, P.Z., Tsai, M.S., Tseng, H.Y., Lai, W.S., Chiu, W.C.: Styling 3d scene via implicit representation and hypernetwork. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1475–1484 (2022)
33. Chibane, J., Bansal, A., Lazova, V., Pons-Moll, G.: Stereo radiance fields (srdf): Learning view synthesis for sparse views of novel scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7911–7920 (2021)
34. Chmielewski, S., Tompalski, P.: Estimating outdoor advertising media visibility with voxel-based approach. *Applied Geography* **87**, 1–13 (2017)
35. Choi, S., Zhou, Q.Y., Koltun, V.: Robust reconstruction of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5556–5565 (2015)
36. Clark, J.H.: Hierarchical geometric models for visible surface algorithms. *Communications of the ACM* **19**(10), 547–554 (1976)
37. Cole, F., Genova, K., Sud, A., Vlasic, D., Zhang, Z.: Differentiable surface rendering via non-differentiable sampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6088–6097 (2021)

38. Cook, R.L., Porter, T., Carpenter, L.: Distributed ray tracing. In: Proceedings of the 11th annual conference on Computer graphics and interactive techniques. pp. 137–145 (1984)
39. Cortinhal, T., Tzelepis, G., Aksoy, E.E.: Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving. arXiv preprint arXiv:2003.03653 (2020)
40. Cruz, R.S., Lebrat, L., Bourgeat, P., Fookes, C., Fripp, J., Salvado, O.: DeepCSR: A 3d deep learning approach for cortical surface reconstruction. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 806–815 (2021)
41. Cui, Y., Chen, R., Chu, W., Chen, L., Tian, D., Li, Y., Cao, D.: Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems* **23**(2), 722–739 (2021)
42. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5828–5839 (2017)
43. Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)* **36**(4), 1 (2017)
44. Daras, G., Chu, W.S., Kumar, A., Lagun, D., Dimakis, A.G.: Solving inverse problems with nerfngans. arXiv preprint arXiv:2112.09061 (2021)
45. Debevec, P., Gortler, S., McMillan, L., Szeliski, R., Bregler, C.: Image-based modeling and rendering. *SIGGRAPH 98 Course Notes for Course 15* (1998)
46. Debevec, P., Yu, Y., Borshukov, G.: Efficient view-dependent image-based rendering with projective texture-mapping. In: *Rendering Techniques' 98: Proceedings of the Eurographics Workshop in Vienna, Austria, June 29—July 1, 1998* 9. pp. 105–116. Springer (1998)
47. Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-supervised nerf: Fewer views and faster training for free. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12882–12891 (2022)
48. Deng, Y., Yang, J., Xiang, J., Tong, X.: Gram: Generative radiance manifolds for 3d-aware image generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10673–10683 (2022)
49. Derksen, D., Izzo, D.: Shadow neural radiance fields for multi-view satellite photogrammetry. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1152–1161 (2021)
50. DeVries, T., Bautista, M.A., Srivastava, N., Taylor, G.W., Susskind, J.M.: Unconstrained scene generation with locally conditioned radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14304–14313 (2021)
51. Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems* **29** (2016)
52. Dou, M., Khamis, S., Degtyarev, Y., Davidson, P., Fanello, S.R., Kowdle, A., Escolano, S.O., Rhemann, C., Kim, D., Taylor, J., et al.: Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (ToG)* **35**(4), 1–13 (2016)
53. Drebin, R.A., Carpenter, L., Hanrahan, P.: Volume rendering. *ACM Siggraph Computer Graphics* **22**(4), 65–74 (1988)

54. Du, Y., Zhang, Y., Yu, H.X., Tenenbaum, J.B., Wu, J.: Neural radiance flow for 4d view synthesis and video processing. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14304–14314. IEEE Computer Society (2021)
55. Eslami, S.A., Jimenez Rezende, D., Besse, F., Viola, F., Morcos, A.S., Garnelo, M., Ruderman, A., Rusu, A.A., Danihelka, I., Gregor, K., et al.: Neural scene representation and rendering. *Science* **360**(6394), 1204–1210 (2018)
56. Fang, J., Xie, L., Wang, X., Zhang, X., Liu, W., Tian, Q.: Neusample: Neural sample field for efficient view synthesis. arXiv preprint arXiv:2111.15552 (2021)
57. Faugeras, O., Faugeras, O.A.: Three-dimensional computer vision: a geometric viewpoint. MIT press (1993)
58. Finkel, R.A., Bentley, J.L.: Quad trees a data structure for retrieval on composite keys. *Acta informatica* **4**, 1–9 (1974)
59. Gafni, G., Thies, J., Zollhofer, M., Nießner, M.: Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8649–8658 (2021)
60. Gallussera, F., Casaroli, G., Bassani, T.: Artificial intelligence and machine learning in spine research. *JOR spine* **2**(1), e1044 (2019)
61. Gao, C., Saraf, A., Kopf, J., Huang, J.B.: Dynamic view synthesis from dynamic monocular video. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5712–5721 (2021)
62. Gao, C., Shih, Y., Lai, W.S., Liang, C.K., Huang, J.B.: Portrait neural radiance fields from a single image. arXiv preprint arXiv:2012.05903 (2020)
63. Gao, K., Gao, Y., He, H., Lu, D., Xu, L., Li, J.: Nerf: Neural radiance field in 3d vision, a comprehensive review. arXiv preprint arXiv:2210.00379 (2022)
64. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14346–14355 (2021)
65. Gillmann, C., Peter, L., Schmidt, C., Saur, D., Scheuermann, G.: Visualizing multimodal deep learning for lesion prediction. *IEEE Computer Graphics and Applications* **41**(5), 90–98 (2021)
66. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 43–54 (1996)
67. Grassal, P.W., Prinzler, M., Leistner, T., Rother, C., Nießner, M., Thies, J.: Neural head avatars from monocular rgb videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18653–18664 (2022)
68. Gregory, A., State, A., Lin, M.C., Manocha, D., Livingston, M.A.: Interactive surface decomposition for polyhedral morphing. *The Visual Computer* **15**, 453–470 (1999)
69. Gross, M., Pfister, H., Zwicker, M., Pauly, M., Stamminger, M., Alexa, M.: Point based computer graphics. In: Eurographics (Tutorials) (2002)
70. Grossman, J.P., Dally, W.J.: Point sample rendering. In: Rendering Techniques' 98: Proceedings of the Eurographics Workshop in Vienna, Austria, June 29—July 1, 1998 9. pp. 181–192. Springer (1998)
71. Gu, J., Liu, L., Wang, P., Theobalt, C.: Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. arXiv preprint arXiv:2110.08985 (2021)
72. Guo, J., Yang, Z., Lin, X., Zhang, Q.: Template nerf: Towards modeling dense shape correspondences from category-specific object images. arXiv preprint arXiv:2111.04237 (2021)

73. Guo, M., Fathi, A., Wu, J., Funkhouser, T.: Object-centric neural scene rendering. arXiv preprint arXiv:2012.08503 (2020)
74. Guo, Y.C., Kang, D., Bao, L., He, Y., Zhang, S.H.: Nerfren: Neural radiance fields with reflections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18409–18418 (2022)
75. Ha, J.S., Driess, D., Toussaint, M.: Deep visual constraints: Neural implicit models for manipulation planning from visual input. IEEE Robotics and Automation Letters **7**(4), 10857–10864 (2022)
76. Habtegebrial, T., Varanasi, K., Bailer, C., Stricker, D.: Fast view synthesis with deep stereo vision. arXiv preprint arXiv:1804.09690 (2018)
77. Hart, J.C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer **12**(10), 527–545 (1996)
78. Hawkins, T., Einarsson, P., Debevec, P.: Acquisition of time-varying participating media. ACM Transactions on Graphics (ToG) **24**(3), 812–815 (2005)
79. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
80. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (TOG) **37**(6), 1–15 (2018)
81. Hedman, P., Ritschel, T., Drettakis, G., Brostow, G.: Scalable inside-out image-based rendering. ACM Transactions on Graphics (TOG) **35**(6), 1–11 (2016)
82. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5875–5884 (2021)
83. Heigl, B., Koch, R., Pollefeys, M., Denzler, J., Van Gool, L.: Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In: Mustererkennung 1999: 21. DAGM-Symposium Bonn, 15.–17. September 1999. pp. 94–101. Springer (1999)
84. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)
85. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
86. Hong, Y., Peng, B., Xiao, H., Liu, L., Zhang, J.: Headnerf: A real-time nerf-based parametric head model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20374–20384 (2022)
87. Huang, J., Dai, A., Guibas, L.J., Nießner, M.: 3dlite: towards commodity 3d scanning for content creation. ACM Trans. Graph. **36**(6), 203–1 (2017)
88. Huang, X., Zhang, Q., Feng, Y., Li, H., Wang, X., Wang, Q.: Hdr-nerf: High dynamic range neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18398–18408 (2022)
89. Ichnowski, J., Avigal, Y., Kerr, J., Goldberg, K.: Dex-nerf: Using a neural radiance field to grasp transparent objects. arXiv preprint arXiv:2110.14217 (2021)
90. Innmann, M., Zollhöfer, M., Nießner, M., Theobalt, C., Stamminger, M.: Volumedeform: Real-time volumetric non-rigid reconstruction. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VIII 14. pp. 362–379. Springer (2016)
91. Insafutdinov, E., Dosovitskiy, A.: Unsupervised learning of shape and pose with differentiable point clouds. Advances in neural information processing systems **31** (2018)

92. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1125–1134 (2017)
93. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: Proceedings of the 24th annual ACM symposium on User interface software and technology. pp. 559–568 (2011)
94. Jain, A., Mildenhall, B., Barron, J.T., Abbeel, P., Poole, B.: Zero-shot text-guided object generation with dream fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 867–876 (2022)
95. Jain, A., Tancik, M., Abbeel, P.: Putting nerf on a diet: Semantically consistent few-shot view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5885–5894 (2021)
96. Jang, W., Agapito, L.: Codenerf: Disentangled neural radiance fields for object categories. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12949–12958 (2021)
97. Jeong, Y., Ahn, S., Choy, C., Anandkumar, A., Cho, M., Park, J.: Self-calibrating neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5846–5854 (2021)
98. Ji, D., Kwon, J., McFarland, M., Savarese, S.: Deep view morphing. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2155–2163 (2017)
99. Jiang, H., Jiang, Z., Grauman, K., Zhu, Y.: Few-view object reconstruction with unknown categories and camera poses. arXiv preprint arXiv:2212.04492 (2022)
100. Jo, K., Shim, G., Jung, S., Yang, S., Choo, J.: Cg-nerf: Conditional generative neural radiance fields. arXiv preprint arXiv:2112.03517 (2021)
101. Johari, M.M., Lepoittevin, Y., Fleuret, F.: Geonerf: Generalizing nerf with geometry priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18365–18375 (2022)
102. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part II 14. pp. 694–711. Springer (2016)
103. Kajiya, J.T.: The rendering equation. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques. pp. 143–150 (1986)
104. Kaltenbrunner, M., Sekitani, T., Reeder, J., Yokota, T., Kuribara, K., Tokuhara, T., Drack, M., Schwödiauer, R., Graz, I., Bauer-Gogonea, S., et al.: An ultra-lightweight design for imperceptible plastic electronics. *Nature* **499**(7459), 458–463 (2013)
105. Kar, A., Häne, C., Malik, J.: Learning a multi-view stereo machine. Advances in neural information processing systems **30** (2017)
106. Kaya, B., Kumar, S., Sarno, F., Ferrari, V., Van Gool, L.: Neural radiance fields approach to deep multi-view photometric stereo. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1965–1977 (2022)
107. Kellnhofer, P., Jebe, L.C., Jones, A., Spicer, R., Pulli, K., Wetzstein, G.: Neural lumigraph rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4287–4297 (2021)

108. Kim, M., Seo, S., Han, B.: Infonerf: Ray entropy minimization for few-shot neural volume rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12912–12921 (2022)
109. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
110. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017)
111. Kobbelt, L., Botsch, M.: A survey of point-based techniques in computer graphics. Computers & Graphics **28**(6), 801–814 (2004)
112. Kondo, N., Ikeda, Y., Tagliasacchi, A., Matsuo, Y., Ochiai, Y., Gu, S.S.: Vaxnerf: Revisiting the classic for voxel-accelerated neural radiance field. arXiv preprint arXiv:2111.13112 (2021)
113. Kosiorek, A.R., Strathmann, H., Zoran, D., Moreno, P., Schneider, R., Mokrá, S., Rezende, D.J.: Nerf-vae: A geometry aware 3d scene generative model. In: International Conference on Machine Learning. pp. 5742–5752. PMLR (2021)
114. Krafcik, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., Torralba, A.: Eye tracking for everyone. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2176–2184 (2016)
115. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**(6), 84–90 (2017)
116. Kumar, R., Anandan, P., Irani, M., Bergen, J., Hanna, K.: Representation of scenes from collections of images. In: Proceedings IEEE Workshop on Representation of Visual Scenes (In Conjunction with ICCV'95). pp. 10–17. IEEE (1995)
117. Kwon, Y., Kim, D., Ceylan, D., Fuchs, H.: Neural human performer: Learning generalizable radiance fields for human performance rendering. Advances in Neural Information Processing Systems **34**, 24741–24752 (2021)
118. Lacroute, P., Levoy, M.: Fast volume rendering using a shear-warp factorization of the viewing transformation. In: Proceedings of the 21st annual conference on Computer graphics and interactive techniques. pp. 451–458 (1994)
119. Lanman, D., Raskar, R., Agrawal, A., Taubin, G.: Shield fields: modeling and capturing 3d occluders. ACM Transactions on Graphics (TOG) **27**(5), 1–10 (2008)
120. Lee, S.Y., Chwa, K.Y., Shin, S.Y.: Image metamorphosis using snakes and free-form deformations. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. pp. 439–448 (1995)
121. Levoy, M.: Volume rendering. IEEE Computer graphics and Applications **8**(3), 29–37 (1988)
122. Levoy, M., Hanrahan, P.: Light field rendering. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 31–42 (1996)
123. Levoy, M., Whitted, T.: The use of points as a display primitive. Dept of Computer Science, University of North Carolina (1985)
124. Li, J., Feng, Z., She, Q., Ding, H., Wang, C., Lee, G.H.: Mine: Towards continuous depth mpi with nerf for novel view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12578–12588 (2021)
125. Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M.A., Cao, D., Li, J.: Deep learning for lidar point clouds in autonomous driving: A review. IEEE Transactions on Neural Networks and Learning Systems **32**(8), 3412–3432 (2020)
126. Li, Y., Li, S., Sitzmann, V., Agrawal, P., Torralba, A.: 3d neural scene representations for visuomotor control. In: Conference on Robot Learning. pp. 112–123. PMLR (2022)

127. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6498–6508 (2021)
128. Lin, C.H., Kong, C., Lucey, S.: Learning efficient point cloud generation for dense 3d object reconstruction. In: proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
129. Lin, C.H., Ma, W.C., Torralba, A., Lucey, S.: Barf: Bundle-adjusting neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5741–5751 (2021)
130. Lin, H., Peng, S., Xu, Z., Bao, H., Zhou, X.: Efficient neural radiance fields with learned depth-guided sampling. arXiv preprint arXiv:2112.01517 (2021)
131. Liu, L., Gu, J., Zaw Lin, K., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. Advances in Neural Information Processing Systems **33**, 15651–15663 (2020)
132. Liu, L., Habermann, M., Rudnev, V., Sarkar, K., Gu, J., Theobalt, C.: Neural actor: Neural free-view synthesis of human actors with pose control. ACM Transactions on Graphics (TOG) **40**(6), 1–16 (2021)
133. Liu, S., Zhang, X., Zhang, Z., Zhang, R., Zhu, J.Y., Russell, B.: Editing conditional radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5773–5783 (2021)
134. Liu, Y., Peng, S., Liu, L., Wang, Q., Wang, P., Theobalt, C., Zhou, X., Wang, W.: Neural rays for occlusion-aware image-based rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7824–7833 (2022)
135. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. arXiv preprint arXiv:1906.07751 (2019)
136. Lombardi, S., Simon, T., Schwartz, G., Zollhoefer, M., Sheikh, Y., Saragih, J.: Mixture of volumetric primitives for efficient neural rendering. ACM Transactions on Graphics (ToG) **40**(4), 1–13 (2021)
137. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**, 91–110 (2004)
138. Luo, H., Chen, A., Zhang, Q., Pang, B., Wu, M., Xu, L., Yu, J.: Convolutional neural opacity radiance fields. In: 2021 IEEE International Conference on Computational Photography (ICCP). pp. 1–12. IEEE (2021)
139. Ma, L., Li, X., Liao, J., Zhang, Q., Wang, X., Wang, J., Sander, P.V.: Deblurnerf: Neural radiance fields from blurry images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12861–12870 (2022)
140. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7210–7219 (2021)
141. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). pp. 922–928. IEEE (2015)
142. McMillan, L., Bishop, G.: Plenoptic modeling: An image-based rendering system. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. pp. 39–46 (1995)
143. Meagher, D.J.: Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer. Electrical and Systems Engineering Department Rensselaer Polytechnic ... (1980)

144. Meng, Q., Chen, A., Luo, H., Wu, M., Su, H., Xu, L., He, X., Yu, J.: Gnerf: Gan-based neural radiance field without posed camera. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6351–6361 (2021)
145. Meshry, M., Goldman, D.B., Khamis, S., Hoppe, H., Pandey, R., Snavely, N., Martin-Brualla, R.: Neural rerendering in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6878–6887 (2019)
146. Mildenhall, B., Hedman, P., Martin-Brualla, R., Srinivasan, P.P., Barron, J.T.: 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. pp. 16190–16199 (2022)
147. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019)
148. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European conference on computer vision. pp. 405–421. Springer (2020)
149. Mittal, A., Kumar, D.: Aicnns (artificially-integrated convolutional neural networks) for brain tumor prediction. EAI Endorsed Transactions on Pervasive Health and Technology **5**(17) (2019)
150. Moreau, A., Piasco, N., Tsishkou, D., Stanciulescu, B., de La Fortelle, A.: Lens: Localization enhanced by nerf synthesis. In: Conference on Robot Learning. pp. 1347–1356. PMLR (2022)
151. Mullen, T.: Mastering blender. John Wiley & Sons (2011)
152. Murdock, K.L.: 3DS max 2009 bible, vol. 560. John Wiley & Sons (2008)
153. Nalbach, O., Arabadzhyska, E., Mehta, D., Seidel, H.P., Ritschel, T.: Deep shading: convolutional neural networks for screen space shading. In: Computer graphics forum. vol. 36, pp. 65–78. Wiley Online Library (2017)
154. Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J.H., Chaitanya, C.R.A., Kaplanyan, A., Steinberger, M.: Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In: Computer Graphics Forum. vol. 40, pp. 45–59. Wiley Online Library (2021)
155. Newcombe, R.A., Fox, D., Seitz, S.M.: Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 343–352 (2015)
156. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: 2011 10th IEEE international symposium on mixed and augmented reality. pp. 127–136. Ieee (2011)
157. Nguyen-Phuoc, T.H., Li, C., Balaban, S., Yang, Y.: Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. Advances in neural information processing systems **31** (2018)
158. Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S., Geiger, A., Radwan, N.: Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5480–5490 (2022)
159. Niemeyer, M., Geiger, A.: Campari: Camera-aware decomposed generative neural radiance fields. In: 2021 International Conference on 3D Vision (3DV). pp. 951–961. IEEE (2021)

160. Niemeyer, M., Geiger, A.: Giraffe: Representing scenes as compositional generative neural feature fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11453–11464 (2021)
161. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3d reconstruction at scale using voxel hashing. ACM Transactions on Graphics (ToG) **32**(6), 1–11 (2013)
162. Noguchi, A., Sun, X., Lin, S., Harada, T.: Neural articulated radiance field. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5762–5772 (2021)
163. Oechsle, M., Peng, S., Geiger, A.: Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5589–5599 (2021)
164. Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al.: Conditional image generation with pixelcnn decoders. Advances in neural information processing systems **29** (2016)
165. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural scene graphs for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2856–2865 (2021)
166. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021)
167. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. arXiv preprint arXiv:2106.13228 (2021)
168. Parker, S., Shirley, P., Livnat, Y., Hansen, C., Sloan, P.P.: Interactive ray tracing for isosurface rendering. In: Proceedings Visualization'98 (Cat. No. 98CB36276). pp. 233–238. IEEE (1998)
169. Peng, S., Dong, J., Wang, Q., Zhang, S., Shuai, Q., Zhou, X., Bao, H.: Animatable neural radiance fields for modeling dynamic human bodies. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14314–14323 (2021)
170. Peng, S., Zhang, Y., Xu, Y., Wang, Q., Shuai, Q., Bao, H., Zhou, X.: 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. pp. 9054–9063 (2021)
171. Pharr, M., Jakob, W., Humphreys, G.: Physically based rendering: From theory to implementation. Morgan Kaufmann (2016)
172. Porter, T., Duff, T.: Compositing digital images. In: Proceedings of the 11th annual conference on Computer graphics and interactive techniques. pp. 253–259 (1984)
173. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10318–10327 (2021)
174. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
175. Qiu, W., Zhong, F., Zhang, Y., Qiao, S., Xiao, Z., Kim, T.S., Wang, Y.: Unrealcv: Virtual worlds for computer vision. In: Proceedings of the 25th ACM international conference on Multimedia. pp. 1221–1224 (2017)

176. Raj, A., Zollhofer, M., Simon, T., Saragih, J., Saito, S., Hays, J., Lombardi, S.: Pixel-aligned volumetric avatars. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11733–11742 (2021)
177. Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K.M., Tagliasacchi, A.: Derf: Decomposed radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14153–14161 (2021)
178. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14335–14345 (2021)
179. Rematas, K., Liu, A., Srinivasan, P.P., Barron, J.T., Tagliasacchi, A., Funkhouser, T., Ferrari, V.: Urban radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12932–12942 (2022)
180. Rematas, K., Martin-Brualla, R., Ferrari, V.: Sharf: Shape-conditioned radiance fields from a single view. arXiv preprint arXiv:2102.08860 (2021)
181. Riegler, G., Osman Ulusoy, A., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3577–3586 (2017)
182. Roessler, B., Barron, J.T., Mildenhall, B., Srinivasan, P.P., Nießner, M.: Dense depth priors for neural radiance fields from sparse input views. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12892–12901 (2022)
183. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
184. Roth, S., Black, M.J.: Specular flow and the recovery of surface structure. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06). vol. 2, pp. 1869–1876. IEEE (2006)
185. Rudnev, V., Elgarib, M., Smith, W., Liu, L., Golyanik, V., Theobalt, C.: Neural radiance fields for outdoor scene relighting. arXiv preprint arXiv:2112.05140 (2021)
186. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
187. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (ECCV) (2016)
188. Schwarz, K., Liao, Y., Niemeyer, M., Geiger, A.: Graf: Generative radiance fields for 3d-aware image synthesis. Advances in Neural Information Processing Systems **33**, 20154–20166 (2020)
189. Seitz, S.M., Dyer, C.R.: Physically-valid view synthesis by image interpolation. In: Proceedings IEEE Workshop on Representation of Visual Scenes (In Conjunction with ICCV’95). pp. 18–25. IEEE (1995)
190. Seitz, S.M., Dyer, C.R.: View morphing. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 21–30 (1996)
191. Shen, J., Ruiz, A., Agudo, A., Moreno-Noguer, F.: Stochastic neural radiance fields: Quantifying uncertainty in implicit 3d representations. In: 2021 International Conference on 3D Vision (3DV). pp. 972–981. IEEE (2021)
192. Shen, S., Wang, Z., Liu, P., Pan, Z., Li, R., Gao, T., Li, S., Yu, J.: Non-line-of-sight imaging via neural transient fields. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(7), 2257–2268 (2021)

193. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
194. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Advances in Neural Information Processing Systems **33**, 7462–7473 (2020)
195. Sitzmann, V., Rezchikov, S., Freeman, B., Tenenbaum, J., Durand, F.: Light field networks: Neural scene representations with single-evaluation rendering. Advances in Neural Information Processing Systems **34**, 19313–19325 (2021)
196. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3d feature embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2437–2446 (2019)
197. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. Advances in Neural Information Processing Systems **32** (2019)
198. Smith, B.A., Yin, Q., Feiner, S.K., Nayar, S.K.: Gaze locking: passive eye contact detection for human-object interaction. In: Proceedings of the 26th annual ACM symposium on User interface software and technology. pp. 271–280 (2013)
199. Smiti, A.: When machine learning meets medical world: Current status and future challenges. Computer Science Review **37**, 100280 (2020)
200. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic scene completion from a single depth image. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1746–1754 (2017)
201. Sonka, M., Hlavac, V., Boyle, R.: Image processing, analysis, and machine vision. Cengage Learning (2014)
202. Srinivasan, P.P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., Barron, J.T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7495–7504 (2021)
203. Stelzner, K., Kersting, K., Kosiorek, A.R.: Decomposing 3d scenes into objects via unsupervised volume segmentation. arXiv preprint arXiv:2104.01148 (2021)
204. Su, S.Y., Yu, F., Zollhöfer, M., Rhodin, H.: A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. Advances in Neural Information Processing Systems **34**, 12278–12291 (2021)
205. Sucar, E., Liu, S., Ortiz, J., Davison, A.J.: imap: Implicit mapping and positioning in real-time. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6229–6238 (2021)
206. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5459–5469 (2022)
207. Swaminathan, R., Kang, S.B., Szeliski, R., Criminisi, A., Nayar, S.K.: On the motion and appearance of specularities in image sequences. In: Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7. pp. 508–523. Springer (2002)
208. Taira, H., Okutomi, M., Sattler, T., Cimpoi, M., Pollefeys, M., Sivic, J., Pajdla, T., Torii, A.: Inloc: Indoor visual localization with dense matching and view synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7199–7209 (2018)
209. Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., Ng, R.: Fourier features let networks learn

- high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems* **33**, 7537–7547 (2020)
210. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Single-view to multi-view: Reconstructing unseen views with a convolutional network. *CoRR abs/1511.06702* **1**(2), 2 (2015)
 211. Tewari, A., Oh, T.H., Weyrich, T., Bickel, B., Seidel, H.P., Pfister, H., Matusik, W., Elgharib, M., Theobalt, C., et al.: Monocular reconstruction of neural face reflectance fields. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 4791–4800 (2021)
 212. Thies, J., Zollhöfer, M., Nießner, M.: Deferred neural rendering: Image synthesis using neural textures. *Acm Transactions on Graphics (TOG)* **38**(4), 1–12 (2019)
 213. Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., Theobalt, C.: Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 12959–12970 (2021)
 214. Trevithick, A., Yang, B.: Grf: Learning a general radiance field for 3d representation and rendering. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 15182–15192 (2021)
 215. Tung, H.Y.F., Cheng, R., Fragkiadaki, K.: Learning spatial common sense with geometry-aware recurrent networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 2595–2603 (2019)
 216. Turki, H., Ramanan, D., Satyanarayanan, M.: 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*. pp. 12922–12931 (2022)
 217. Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-nerf: Structured view-dependent appearance for neural radiance fields. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 5481–5490. IEEE (2022)
 218. Wallace, J.R., Elmquist, K.A., Haines, E.A.: A ray tracing algorithm for progressive radiosity. In: *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. pp. 315–324 (1989)
 219. Wang, C., Chai, M., He, M., Chen, D., Liao, J.: Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 3835–3844 (2022)
 220. Wang, C., Wu, X., Guo, Y.C., Zhang, S.H., Tai, Y.W., Hu, S.M.: Nerf-sr: High quality neural radiance fields using supersampling. In: *Proceedings of the 30th ACM International Conference on Multimedia*. pp. 6445–6454 (2022)
 221. Wang, L., Wang, Z., Lin, P., Jiang, Y., Suo, X., Wu, M., Xu, L., Yu, J.: ibutter: Neural interactive bullet time generator for human free-viewpoint rendering. In: *Proceedings of the 29th ACM International Conference on Multimedia*. pp. 4641–4650 (2021)
 222. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689* (2021)
 223. Wang, Z., Wu, S., Xie, W., Chen, M., Prisacariu, V.A.: Nerf-: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064* (2021)
 224. Wang, Z., Bagautdinov, T., Lombardi, S., Simon, T., Saragih, J., Hodgins, J., Zollhofer, M.: Learning compositional radiance fields of dynamic human heads. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 5704–5713 (2021)

225. Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., Zhou, J.: Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5610–5619 (2021)
226. Wetzstein, G., Lanman, D., Heidrich, W., Raskar, R.: Layered 3d: tomographic image synthesis for attenuation-based light field and high dynamic range displays. In: ACM SIGGRAPH 2011 papers, pp. 1–12. ACM New York, NY, USA (2011)
227. Wetzstein, G., Lanman, D.R., Hirsch, M.W., Raskar, R.: Tensor displays: compressive light field synthesis using multilayer displays with directional backlighting. ACM Transactions on Graphics (TOG) (2012)
228. Whelan, T., Salas-Moreno, R.F., Glocker, B., Davison, A.J., Leutenegger, S.: Elasticfusion: Real-time dense slam and light source estimation. The International Journal of Robotics Research **35**(14), 1697–1716 (2016)
229. Wijmans, E., Furukawa, Y.: Exploiting 2d floorplan for building-scale panorama rgbd alignment. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 308–316 (2017)
230. Wizadwongsa, S., Phongthawee, P., Yenphraphai, J., Suwananakorn, S.: Nex: Real-time view synthesis with neural basis expansion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8534–8543 (2021)
231. Wolberg, G.: Digital image warping, vol. 10662. IEEE computer society press Los Alamitos, CA (1990)
232. Wood, D.N., Azuma, D.I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D.H., Stuetzle, W.: Surface light fields for 3d photography. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. pp. 287–296 (2000)
233. Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Interpretable transformations with encoder-decoder networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5726–5735 (2017)
234. Wu, L., Lee, J.Y., Bhattacharjee, A., Wang, Y.X., Forsyth, D.: 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. pp. 16200–16209 (2022)
235. Xian, W., Huang, J.B., Kopf, J., Kim, C.: Space-time neural irradiance fields for free-viewpoint video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9421–9431 (2021)
236. Xiang, F., Xu, Z., Hasan, M., Hold-Geoffroy, Y., Sunkavalli, K., Su, H.: Neutex: Neural texture mapping for volumetric neural rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7119–7128 (2021)
237. Xiangli, Y., Xu, L., Pan, X., Zhao, N., Rao, A., Theobalt, C., Dai, B., Lin, D.: Citynerf: Building nerf at city scale. arXiv preprint arXiv:2112.05504 (2021)
238. Xie, C., Park, K., Martin-Brualla, R., Brown, M.: Fig-nerf: Figure-ground neural radiance fields for 3d object category modelling. In: 2021 International Conference on 3D Vision (3DV). pp. 962–971. IEEE (2021)
239. Xu, H., Alldieck, T., Sminchisescu, C.: H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. Advances in Neural Information Processing Systems **34**, 14955–14966 (2021)
240. Xu, Y., Peng, S., Yang, C., Shen, Y., Zhou, B.: 3d-aware image synthesis via learning structural and textural representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18430–18439 (2022)

241. Xu, Z., Wu, H.T., Wang, L., Zheng, C., Tong, X., Qi, Y.: Dynamic hair capture using spacetime optimization. To appear in ACM TOG **33**, 6 (2014)
242. Yang, B., Zhang, Y., Xu, Y., Li, Y., Zhou, H., Bao, H., Zhang, G., Cui, Z.: Learning object-compositional neural radiance field for editable scene rendering. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13779–13788 (2021)
243. Yang, G., Vo, M., Neverova, N., Ramanan, D., Vedaldi, A., Joo, H.: Banmo: Building animatable 3d neural models from many casual videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2863–2873 (2022)
244. Yang, G.W., Zhou, W.Y., Peng, H.Y., Liang, D., Mu, T.J., Hu, S.M.: Recursive-nerf: An efficient and dynamically growing nerf. arXiv preprint arXiv:2105.09103 (2021)
245. Yao, G., Wu, H., Yuan, Y., Li, L., Zhou, K., Yu, X.: Learning implicit body representations from double diffusion based neural radiance fields. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22. International Joint Conferences on Artificial Intelligence Organization. pp. 1566–1572 (2022)
246. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. Advances in Neural Information Processing Systems **34**, 4805–4815 (2021)
247. Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., Lipman, Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. Advances in Neural Information Processing Systems **33**, 2492–2502 (2020)
248. Yen-Chen, L., Florence, P., Barron, J.T., Rodriguez, A., Isola, P., Lin, T.Y.: inferf: Inverting neural radiance fields for pose estimation. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1323–1330. IEEE (2021)
249. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5752–5761 (2021)
250. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4578–4587 (2021)
251. Yu, H.X., Guibas, L.J., Wu, J.: Unsupervised discovery of object radiance fields. arXiv preprint arXiv:2107.07905 (2021)
252. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4471–4480 (2019)
253. Zeng, M., Zhao, F., Zheng, J., Liu, X.: Octree-based fusion for realtime 3d reconstruction. Graphical Models **75**(3), 126–136 (2013)
254. Zhang, C., Chen, T.: Spectral analysis for sampling image-based rendering data. IEEE Transactions on Circuits and Systems for Video Technology **13**(11), 1038–1050 (2003)
255. Zhang, J., Yang, G., Tulsiani, S., Ramanan, D.: Ners: neural reflectance surfaces for sparse-view 3d reconstruction in the wild. Advances in Neural Information Processing Systems **34**, 29835–29847 (2021)
256. Zhang, J., Liu, X., Ye, X., Zhao, F., Zhang, Y., Wu, M., Zhang, Y., Xu, L., Yu, J.: Editable free-viewpoint video using a layered neural representation. ACM Transactions on Graphics (TOG) **40**(4), 1–18 (2021)
257. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020)

- 258. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
- 259. Zhang, X., Srinivasan, P.P., Deng, B.,Debevec, P., Freeman, W.T., Barron, J.T.: Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. ACM Transactions on Graphics (TOG) **40**(6), 1–18 (2021)
- 260. Zhang, X., Park, S., Beeler, T., Bradley, D., Tang, S., Hilliges, O.: Eth-xgaze: A large scale dataset for gaze estimation under extreme head pose and gaze variation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16. pp. 365–381. Springer (2020)
- 261. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: Appearance-based gaze estimation in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4511–4520 (2015)
- 262. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: It’s written all over your face: Full-face appearance-based gaze estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 51–60 (2017)
- 263. Zhao, F., Yang, W., Zhang, J., Lin, P., Zhang, Y., Yu, J., Xu, L.: Humannerf: Generalizable neural human radiance field from sparse inputs. arXiv preprint arXiv:2112.02789 (2021)
- 264. Zheng, K.C., Colburn, A., Agarwala, A., Agrawala, M., Salesin, D., Curless, B., Cohen, M.F.: Parallax photography: creating 3d cinematic effects from stills. In: Proceedings of Graphics Interface 2009, pp. 111–118. CIPS (2009)
- 265. Zhi, S., Laidlow, T., Leutenegger, S., Davison, A.J.: In-place scene labelling and understanding with implicit scene representation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15838–15847 (2021)
- 266. Zhi, S., Sucar, E., Mouton, A., Haughton, I., Laidlow, T., Davison, A.J.: ilabel: Interactive neural scene labelling. arXiv preprint arXiv:2111.14637 (2021)
- 267. Zhou, P., Xie, L., Ni, B., Tian, Q.: Cips-3d: A 3d-aware generator of gans based on conditionally-independent pixel synthesis. arXiv preprint arXiv:2110.09788 (2021)
- 268. Zhou, Q.Y., Koltun, V.: Color map optimization for 3d reconstruction with consumer depth cameras. ACM Transactions on Graphics (ToG) **33**(4), 1–10 (2014)
- 269. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. arXiv preprint arXiv:1805.09817 (2018)
- 270. Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A.: View synthesis by appearance flow. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. pp. 286–301. Springer (2016)
- 271. Zhu, J.Y., Zhang, Z., Zhang, C., Wu, J., Torralba, A., Tenenbaum, J., Freeman, B.: Visual object networks: Image generation with disentangled 3d representations. Advances in neural information processing systems **31** (2018)
- 272. Zhuang, Y., Zhu, H., Sun, X., Cao, X.: Mofanerf: Morphable facial neural radiance field. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III. pp. 268–285. Springer (2022)