Advances in Neural Rendering

A. Tewari[1⋆] J. Thies[2⋆] B. Mildenhall[3⋆] P. Srinivasan[3⋆] E. Tretschk[1] Y. Wang[4] C. Lassner[5] V. Sitzmann[6] R. Martin-Brualla[3]
S. Lombardi[5] T. Simon[5] C. Theobalt[1] M. Nießner[7] J. T. Barron[3] G. Wetzstein[8] M. Zollhöfer[5] V. Golyanik[1]

[1]MPI for Informatics [2]MPI for Intelligent Systems [3]Google Research [4]ETH Zurich [5]Reality Labs Research
[6]MIT [7]Technical University of Munich [8]Stanford University [⋆]Equal contribution.

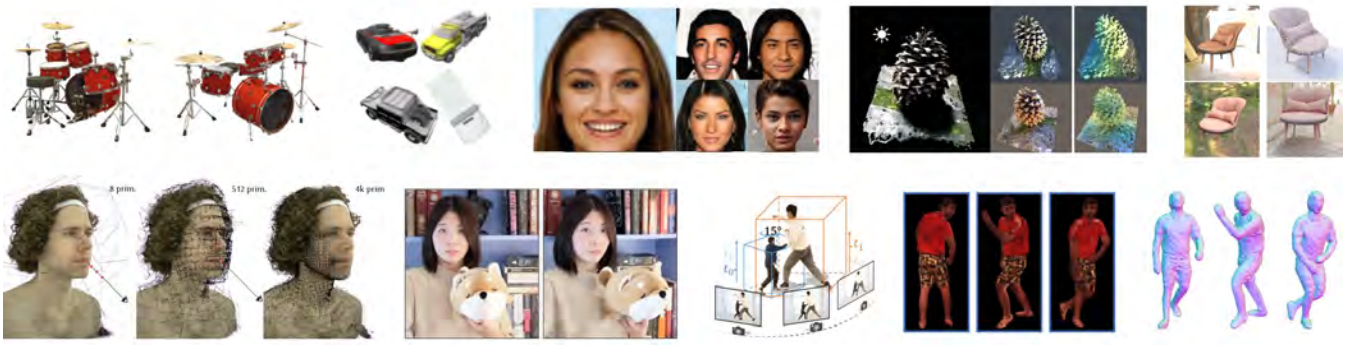


Figure 1: This state-of-the-art report discusses a large variety of neural rendering methods which enable applications such as novel-view synthesis of static and dynamic scenes, generative modeling of objects, and scene relighting. See Section 4 for more details on the various methods. Images adapted from [MST*20, TY20, CMK*21, ZSD*21, BBJ*21, LSS*21, PSB*21, JXX*21, PDW*21].

Abstract

Synthesizing photo-realistic images and videos is at the heart of computer graphics and has been the focus of decades of research. Traditionally, synthetic images of a scene are generated using rendering algorithms such as rasterization or ray tracing, which take specifically defined representations of geometry and material properties as input. Collectively, these inputs define the actual scene and what is rendered, and are referred to as the scene representation *(where a scene consists of one or more objects). Example scene representations are triangle meshes with accompanied textures (e.g., created by an artist), point clouds (e.g., from a depth sensor), volumetric grids (e.g., from a CT scan), or implicit surface functions (e.g., truncated signed distance fields). The reconstruction of such a scene representation from observations using differentiable rendering losses is known as* inverse graphics *or* inverse rendering*. Neural rendering is closely related, and combines ideas from classical computer graphics and machine learning to create algorithms for synthesizing images from real-world observations. Neural rendering is a leap forward towards the goal of synthesizing photo-realistic image and video content. In recent years, we have seen immense progress in this field through hundreds of publications that show different ways to inject learnable components into the rendering pipeline. This state-of-the-art report on advances in neural rendering focuses on methods that combine classical rendering principles with learned 3D scene representations, often now referred to as neural scene representations. A key advantage of these methods is that they are 3D-consistent by design, enabling applications such as novel viewpoint synthesis of a captured scene. In addition to methods that handle static scenes, we cover neural scene representations for modeling non-rigidly deforming objects and scene editing and composition. While most of these approaches are scene-specific, we also discuss techniques that generalize across object classes and can be used for generative tasks. In addition to reviewing these state-of-the-art methods, we provide an overview of fundamental concepts and definitions used in the current literature. We conclude with a discussion on open challenges and social implications.*

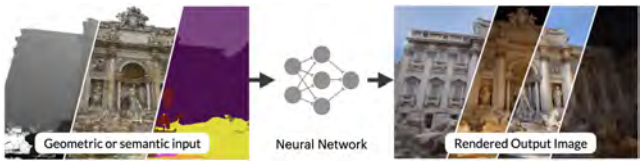
1. Introduction

Synthesis of controllable and photo-realistic images and videos is one of the fundamental goals of computer graphics. During the last decades, methods and representations have been developed to mimic the image formation model of real cameras, including the handling of complex materials and global illumination. These methods are based on the laws of physics and simulate the light

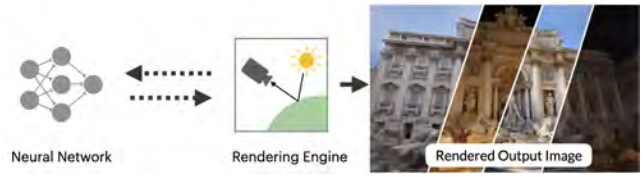
transport from light sources to the virtual camera for synthesis. To this end, all physical parameters of the scene have to be known for the rendering process. These parameters, for example, contain information about the scene geometry and material properties such as reflectivity or opacity. Given this information, modern ray tracing techniques can generate photo-real imagery. Besides the physics-based rendering methods, there is a variety of techniques that approximate the real-world image formation model. These methods are based on mathematical approximations (e.g., a piece-wise linear approximation of the surface; i.e., triangular meshes) and heuristics (e.g., Phong shading) to improve the applicability (e.g., for real-time applications). While these methods require fewer parameters to represent a scene, the achieved realism is also reduced.

While traditional computer graphics allows us to generate high-quality controllable imagery of a scene, all physical parameters of the scene, for example, camera parameters, illumination and materials of the objects need to be provided as inputs. If we want to generate controllable imagery of a real-world scene, we would need to estimate these physical properties from existing observations such as images and videos. This estimation task is referred to as inverse rendering and is extremely challenging, especially when the goal is photo-realistic synthesis. In contrast, neural rendering is a rapidly emerging field which allows the compact representation of scenes, and rendering can be learned from existing observations by utilizing neural networks (see Figure 1). The main idea of neural rendering is to combine insights from classical (physics-based) computer graphics and recent advances in deep learning. Similar to classical computer graphics, the goal of neural rendering is to generate photo-realistic imagery in a controllable way (c.f. definition of neural rendering in [TFT*20]). This, for example, includes novel viewpoint synthesis, relighting, deformation of the scene, and compositing.

Early neural rendering approaches (covered in [TFT*20]) used neural networks to convert scene parameters into the output images. The scene parameters are either directly given as one-dimensional inputs, or a classical computer graphics pipeline is used to generate two-dimensional inputs. The deep neural networks are trained on observations of real-world scenes and learn to model as well as render these scenes. A deep neural network can be seen as a universal function approximator. Specifically, a network defines a family of functions based on its input arguments, model architecture, and trainable parameters. Stochastic gradient descent is employed to find the function from this space that best explains the training set as measured by the training loss. From this viewpoint, neural rendering aims to find the mapping $\mathbf{I} = \mathcal{M}(\mathbf{c})$ between control parameters $\mathbf{c} \in \mathbb{R}^{d_{in}}$ and the corresponding output image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, with H and W being image height and width. This can be interpreted as a complex and challenging sparse data interpolation problem. Thus, neural rendering, similar to classical function fitting, has to navigate the trade-off between under- and over-fitting, i.e., representing the training set well vs. generalization to unobserved inputs. If the representational power of the network is insufficient, the quality of the resulting images will be low, e.g., results are often blurry. On the other hand, if the representational power is too large, the network overfits to the training set and does not generalize to unseen inputs at test time. Finding the right network architecture is an art in itself. In the context of neural rendering, designing the right physically motivated inductive biases often requires a strong graphics background. These physically motivated inductive biases act as regularizers and ensure that the found function is close to how 3D space and/or image formation works in our real world, thus leading to better generalization at test time. Inductive biases can be added to the network in different ways. For example, in terms of the employed layers, at what point in the network and in which form inputs are provided, or even via the integration of non-trainable (but differentiable) components from classical computer graphics. One great example for this are recent neural rendering techniques that try to disentangle the modeling and rendering processes by only learning the 3D scene representation and relying on a rendering function from computer graphics for supervision. For example, Neural Radiance Fields (NeRF) [MST*20] uses a multi-layer perceptron (MLP) to approximate the radiance and density field of a 3D scene. This learned volumetric representation can be rendered from any virtual camera using analytic differentiable rendering (i.e., volumetric integration). For training, observations of the scene from several camera viewpoints are assumed. The network is trained on these observations by rendering the estimated



(a) 2D Neural Rendering, also known as neural refinement, neural re-rendering, or deferred neural rendering is based on 2D inputs that are generated for example using a classical renderer and *learns to render a scene in 2D*.



(b) 3D Neural Rendering *learns to represent a scene in 3D* and uses fixed differentiable rendering schemes from computer graphics which are motivated by physics.

Figure 2: The term "Neural Rendering" is often applied to what are two distinct concepts. The previous STAR report on neural rendering [TFT*20] primarily focused on the paradigm shown in (a), in which a neural network is trained to map from some 2D input signal (such as a semantic label or a rasterized proxy geometry) directly to the output image — the neural network is trained to render. This report focuses on a newer emerging paradigm for neural rendering, shown in (b) and well-exemplified by NeRF [MST*20]. Here, a neural network is supervised so as to represent the shape or appearance of a particular scene, and that neural representation is rendered using a somewhat conventional graphics "engine" that is defined analytically, instead of being learned. Unlike the previous paradigm, here the neural network does not learn how to render — it instead learns to represent a scene *in 3D*, and that scene is then rendered according to the physics of image formation. Image adapted from [MGK*19].

3D scene from these training viewpoints, and minimizing the difference between the rendered and observed images. Once trained, the 3D scene approximated by the neural network can be rendered from a novel viewpoint, enabling controllable synthesis. In contrast to approaches that use the neural network to learn the rendering function as well [TFT*20], NeRF uses knowledge from computer graphics more explicitly in the method, enabling better generalization to novel views due to the (physical) inductive bias: an intermediate 3D-structured representation of the density and radiance of the scene. As a result, NeRF learns physically meaningful color and density values in 3D space, which physics-inspired ray casting and volume integration can then render consistently into novel views.

The achieved quality, as well as the simplicity of the method, led to an 'explosion' of developments in the field. Several advances have been made which improve the applicability, enable controllability, the capture of dynamically changing scenes as well as the training and inference times. Within this report, we cover these recent advances in the field. To foster a deep understanding of these methods, we discuss the fundamentals of neural rendering by describing the different components and design choices in detail in Section 3. Specifically, we clarify the definition of the different scene representations used in the current literature (surfaces and volumetric approaches), and describe ways to approximate them using deep neural networks. We also present the fundamental rendering functions from computer graphics that are used to train these representations. Since neural rendering is a very fast evolving field, with significant progress along many different dimensions, we develop a taxonomy of the recent approaches w.r.t. their application field to provide a concise overview of the developments. Based on this taxonomy and the different application areas, we present the state-of-the-art methods in Section 4. The report is concluded with Section 5 discussing the open challenges and Section 6 discussing social implications of photo-realistic synthetic media.

2. Scope of This STAR

In this state-of-the-art report, we focus on advanced neural rendering approaches that combine classical rendering with learnable 3D representations (see Figure 2). The underlying neural 3D representations are 3D-consistent by design and enable control over different scene parameters. Within this report, we give a comprehensive overview of the different scene representations and detail the fundamentals of the components that are lent from classical rendering pipelines as well as machine learning. We further focus on approaches that use Neural Radiance Fields [MST*20] and volumetric rendering. However, we do not focus on neural rendering methods that reason mostly in 2D screen space; we refer to [TFT*20] for a discussion on such approaches. We also do not cover neural super-sampling and denoising methods for ray-traced imagery [CKS*17, KBS15].

3. Fundamentals of Neural Rendering

Neural rendering, and especially 3D neural rendering is based on classical concepts of computer graphics (see Figure 2). A neural rendering pipeline learns to render and/or represent a scene from real-world imagery, which can be an unordered set of images, or structured, multi-view images or videos. It does so by mimicking the physical process of a camera that captures a scene. A key property of 3D neural rendering is the disentanglement of the camera capturing process (i.e., the projection and image formation) and the 3D scene representation during this training. This disentanglement has several advantages and leads especially to a high level of 3D consistency during the synthesis of images (e.g., for novel viewpoint synthesis). To disentangle the projection and other physical processes from the 3D scene representation, 3D neural rendering methods rely on known image formation models from computer graphics (e.g., rasterization, point splatting, or volumetric integration). These models are motivated by physics, especially the interaction of the light of an emitter with the scene as well as the camera itself. This light transport is formulated using the rendering equation [Kaj86].

The computer graphics field offers a variety of approximations to this rendering equation. These approximations are dependent on the used scene representation and range from classical rasterization to path tracing and volumetric integration. 3D neural rendering exploits these rendering methods. In the following, we will detail the scene representations (Section 3.1) as well as the rendering methods (Section 3.2) used in common neural rendering methods. Note that both the scene representation as well as the rendering method itself have to be differentiable in order to learn from real images (Section 3.3).

3.1. Scene Representations

For decades, the computer graphics community has explored various primitives, including point clouds, implicit and parametric surfaces, meshes, and volumes (see Figure 3). While these representations have clear definitions in the computer graphics field, there is often a confusion in the current literature of neural rendering, especially when it is about implicit and explicit surface representations and volumetric representations. In general, volumetric representations can represent surfaces, but not vice versa. Volumetric representations store volumetric properties such as densities, opacities or occupancies, but they can also store multidimensional features such as colors or radiance. In contrast to volumetric representations, surface representations store properties w.r.t. the surface of an object. They cannot be used to model volumetric matter, such as smoke (unless it is a coarse approximation). For both surface and volumetric representations, there are continuous and discretized counterparts (see Figure 3). The continuous representations are particularly interesting for neural rendering approaches since they can provide analytic gradients.

For surface representations, there are two different ways to represent the surface – explicitly or implicitly. The surface using an explicit surface function $f_{explicit}(.) \in \mathbb{R}$ in the Euclidean space is

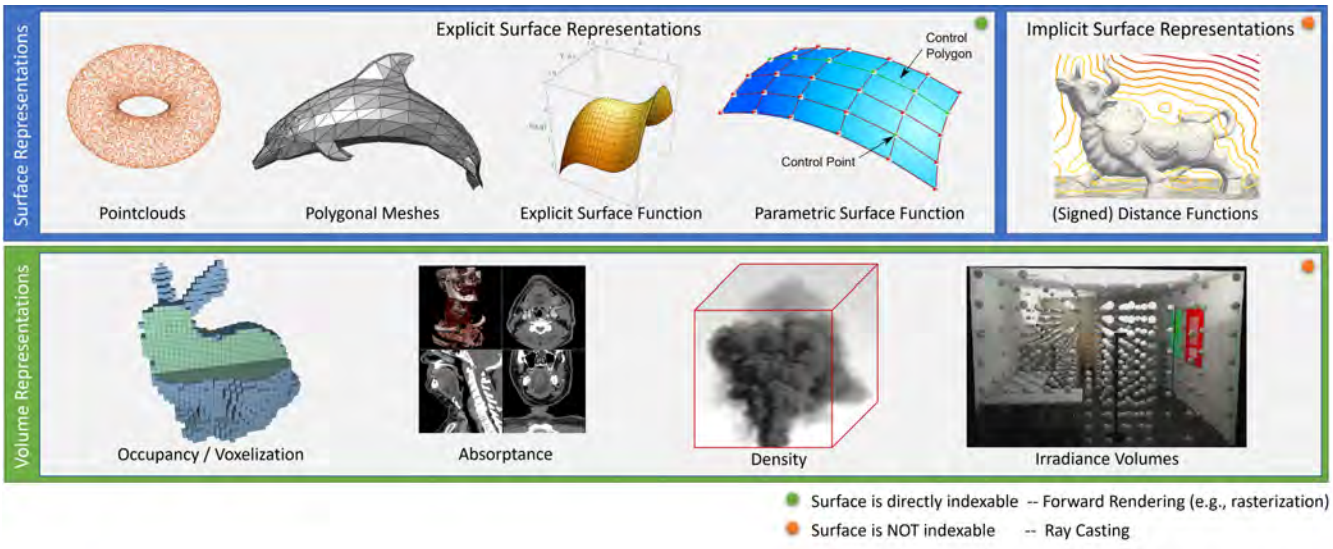


Figure 3: An overview of classical surface and volume representations. Images adapted from [GSHG98, SS10, YGKL21a, Vla09, Chu06].

defined as:

$$S_{explicit} = \left\{ \begin{pmatrix} x \\ y \\ f_{explicit}(x,y) \end{pmatrix} \middle| \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2 \right\}. \quad (1)$$

Note that an explicit surface can also be represented as a parametric function $f_{parametric}(.) \in \mathbb{R}^3$, which generalizes $S_{explicit}$:

$$S^*_{explicit} = \left\{ f_{parametric}(u,v) \middle| \begin{pmatrix} u \\ v \end{pmatrix} \in \mathbb{R}^2 \right\}. \quad (2)$$

The surface using an implicit surface function $f_{implicit}(\cdot) \in \mathbb{R}$ is defined as the zero-level set:

$$S_{implicit} = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 \middle| f_{implicit}(x,y,z) = 0 \right\}. \quad (3)$$

Whereas a volume representation defines properties in the entire space:

$$V = \left\{ f_{vol}(x,y,z) \middle| \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 \right\}. \quad (4)$$

Note that the respective function domain can be restricted for all these representations.

In general, for all three scene representations, the underlying function can be any function that is capable to approximate the respective content. For simple surfaces like a plane, the functions $f_{implicit}$, $f_{explicit}$ can be linear functions. To handle more complex surfaces or volumes, polynomials (for example from a Taylor series) or multivariate Gaussians can be used. To increase the expressiveness further, these functions can be spatially localized and then combined into a mixture, for example multiple Gaussians can form a Gaussian mixture. Radial basis function networks are such mixture models and can be used as an approximator for both, implicit surface and volume functions [CBC*01a]. Note that these radial basis function networks can be interpreted as a single layer of a neural network.

Since neural networks and, especially, multi-layer perceptrons (MLPs) are universal function approximators, they can be used to 'learn' the underlying functions ($f_{implicit}$, $f_{explicit}$, $f_{parametric}$, or f_{vol}). (Similar to a Gaussian mixture, multiple localized, weaker MLPs can be combined into a mixture as well, e.g., [RPLG21].) In the context of neural rendering, a scene representation that is using a neural network to approximate the surface or volumetric representation function is called *neural scene representation*. Note that both surface and volumetric representations can be extended to store additional information, like color or view-dependent radiance.

In the following, we will discuss the different MLP-based function approximators that build the foundation of the recent neural surface and volumetric representations.

3.1.1. Multi-Layer Perceptron as a Universal Function Approximator

Multi-Layer Perceptrons (MLPs) are known to act as Universal Function Approximators [HSW89]. Specifically, we use MLPs to represent surface or volumetric properties. A multi-layer perceptron is a conventional fully-connected neural network. In the context of scene repesentations, the MLP takes as input a *coordinate* in space, and produces as output some value corresponding to that coordinate. This type of network is also known as *coordinate-based neural network* (and the resulting representation is called coordinate-based scene representation). Note that the input coordinate space can be aligned with the Euclidean space, but it can also be embedded for example in the uv-space of a mesh (resulting in a neural parametric surface).

A key finding to use ReLU-based MLPs for neural representation and rendering tasks is the usage of positional encoding. Inspired by the positional encoding used in natural language processing (e.g.,

in Transformers [VSP*17]), the input coordinates are positionally encoded using a set of basis functions. These basis functions can be fixed [MST*20] or they can be learned [TSM*20]. These spatial embeddings simplify the task of the MLP to learn the mapping from a location to a specific value, since through the spatial embedding, the input space is partitioned. As an example, the positional encoding used in NeRF [MST*20] is defined as:

$$\mathbf{x} \mapsto [\cos(\mathbf{M}\mathbf{x}), \sin(\mathbf{M}\mathbf{x})] \tag{5}$$

$$\text{where } \mathbf{M} = \begin{bmatrix} \mathbf{I} & 2\mathbf{I} & 2^2\mathbf{I} & \dots & 2^{p-1}\mathbf{I} \end{bmatrix}^\top. \tag{6}$$

Here, $\mathbf{x}$ is the input coordinate and p is a hyperparameter controlling the frequencies used (dependent on the target signal resolution). This "soft" binary encoding of the input coordinates makes it easier for the network to access higher frequencies of the input.

As mentioned above, MLP-based function approximators can be used to represent a surface or volume (i.e., $f_{implicit}$, $f_{explicit}$, $f_{parametric}$, or f_{vol}), but they can also be used to store other attributes like color. For instance, there are hybrid representations composed of classical surface representations like point clouds or meshes with an MLP to store the surface appearance (e.g., texture field [OMN*19]).

3.1.2. Representing Surfaces

Point Clouds. A point cloud is a set of elements of the Euclidean space. A continuous surface can be discretized by a point cloud - each element of the point cloud represents a sample point (x, y, z) on the surface. For each point, additional attributes can be stored such as normals or colors. A point cloud that features normals is also referred to as oriented point cloud. Besides simple points that can be seen as infinitesimally small surface patches, oriented point clouds with a radius can be used (representing a 2D disk that lies on the tangent plane of the underlying surface). This representation is called surface elements, alias surfels [PZvBG00]. They are often used in computer graphics to render point clouds or particles from simulations. The rendering of such surfels is called splatting, and recent work shows that it is differentiable [YSW*19a]. Using such a differentiable rendering pipeline, it is possible to directly back-propagate to the point cloud locations as well as the accompanied features (e.g., radius or color). In Neural Point-based Graphics [ASK*20a] and SynSin [WGSJ20], learnable features are attached to the points that can store rich information about the appearance and shape of the actual surface. In ADOP [RFS21a] these learnable features are interpreted by an MLP which can account for view-dependent effects. Note that instead of storing explicitly features for specific points, one can also use an MLP to predict the features for the discrete positions.

As mentioned above, a point cloud is a set of elements of the Euclidean space, thus, besides surfaces, they can also represent volumes (e.g., storing additional opacity or density values). Using a radius for each point naturally leads to a full sphere-based formulation [LZ21].

Meshes. Polygonal meshes represent a piece-wise linear approximation of a surface. Especially, triangle and quad meshes are used in computer graphics as de facto standard representation for surfaces. The graphics pipeline and graphic accelerators (GPUs) are optimized to process and rasterize billions of triangles per second. The majority of graphics editing tools work with triangle meshes which makes this representation important for any content creation pipeline. To be directly compatible with these pipelines, many 'classical' inverse graphics and neural rendering methods use this basic surface representation. Using a differentiable renderer, the vertex positions as well as the vertex attributes (e.g., colors) can be optimized for to reproduce an image. Neural networks can be trained to predict the vertex locations, e.g., to predict dynamically changing surfaces [BNT21]. Instead of using vertex attributes, a common strategy to store surface attributes within the triangles are texture maps. 2D texture coordinates are attached to the vertices of the mesh which reference a location in the texture image. Using barycentric interpolation, texture coordinates can be computed for any point in a triangle and the attribute can be retrieved from the texture using bilinear interpolation. The concept of textures is also integrated into the standard graphics pipeline, with additional features such as mip-mapping which is needed to properly handle the sampling of the texture (c.f., sampling theorem). Deferred Neural Rendering [TZN19], uses textures that contain learnable view-dependent features, so-called neural textures. Specifically, a coarse mesh is used as underlying 3D representation, to rasterize these neural textures. A neural network interprets these rasterized features in image space. Note that the network can for example be a pixel-wise MLP, then the neural texture represents the surface radiance.

In contrast to using discrete textures, continuous textures can be used. The authors of texture fields [OMN*19] propose the usage of an MLP that predicts color values for each surface point. In neural reflectance field textures (NeRF-Tex) [BGP*21] the idea of NeRF [MST*20] is combined with the idea of using a 2D neural texture and an underlying 3D mesh. NeRF-Tex is conditioned on user-defined parameters that control the appearance, thus, being editable by artists.

Implicit Surfaces. Implicit surfaces define the surface as the zero level-set of a function, see Eq. 3. The most commonly used implicit surface representation is a signed distance function (SDF). These SDF representations are used in numerous 3D scanning techniques that use volumetric fusion [CL96] to incrementally reconstruct the surface of a static [IKH*11, NZIS13] or dynamic object [NFS15]. Implicit surface representations offer many advantages as they avoid the requirement of defining a mesh template, thus, being able to represent objects with unknown topology or changing topology in a dynamic scenario. The volumetric fusion approaches mentioned above use a discretized (truncated) signed distance function, i.e., using a 3D grid containing signed distance values. Hoppe et al. [HDD*92] propose piece-wise linear functions to model the signed distance function w.r.t. input surface point samples. The seminal work of Carr et al. [CBC*01b] uses a radial basis function network instead. This radial basis function network represent a continuous implicit surface function and can be seen as the first 'neural' implicit surface representation. Recent neural implicit surfaces representations are based on coordinate-based multi-layer perceptrons (MLPs), covered in Section 3.1.1. Such representations have been gaining widespread popularity in neural scene representation and rendering. They were proposed

concurrently in [PFS*19, CZ19] for shape modeling, where MLP architectures were used to map continuous coordinates to signed distance values. The fidelity of signals represented by such coordinate networks, or neural implicit representation, is primarily limited by the capacity of the network. Thus, compared to other aforementioned representations, implicit surfaces offer potential advantages in memory efficiency and, as a continuous representation, they can theoretically represent geometries at infinite resolution. The initial proposals was ensued, with broad enthusiasm, by a variety of improvements of different focuses, including improving the training schemes [XFYS20,DZW*20,YAK*20], leveraging global-local context [XWC*19, EGO*20], adopting specific parameterizations [GCV*19, DGY*20, CTZ20, KJJ*21, YRSH21] or spatial partitions [GCS*20,TTG*20,CLI*20,TLY*21,MLL*21]. As there is no requirement of pre-defining the mesh template or the object topology, neural implicit surfaces are well suited for modeling objects of varying topologies [PFS*19, CZ19]. Analytic gradients of the output with respect to the input coordinates can be computed using backpropagation. This makes it possible to implement regularization terms on the gradients [GYH*20], in addition to other geometrically motivated regularizers [GYH*20, PFAK20, YAK*20]. These respresentations can be extended to also encode the radiance of the scene [KJJ*21, YTB*21, SHN*19]. This is useful for neural rendering, where we want the scene representation to encode both the geometry and appearance of the scenes.

3.1.3. Representing Volumes

Voxel Grids. As the pixel-equivalent in $\mathbb{R}^3$, voxels are commonly used to represent volumes. They can store the geometry occupancy, or store the density values for a scene with volumetric effects such as transparency. In addition, the appearance of the scene can be stored [GSHG98]. Using trilinear interpolation these volume attributes can be accessed at any point within the voxel grid. T his interpolation is especially used for sample-based rendering methods like ray casting. While the stored attributes can have a specific semantic meaning (e.g., occupancy), the attributes can also be learned. Sitzmann et al. propose the use of DeepVoxels [STH*19], where features are stored in a voxel grid. The accumulation and interpretation of the features after the ray-casting rendering procedure is done using a deep neural network. These DeepVoxels can be seen as volumetric neural textures, which can be directly optimized using backpropagation. While dense voxel-based representations are fast to query, they are memory inefficient and 3D CNNs, potentially operating on these volumes, are computationally heavy. Octree data structures [LK10] can be used to represent the volume in a sparse manner. Sparse 3D convolution on octrees [WLG*17, ROUG17] can help mitigate some problems, but these compact data structures cannot be easily updated on the fly. Thus, they are difficult to integrate into learning frameworks. Other approaches to mitigating the memory challenges of dense voxel grids include using object-specific shape templates [KTEM18], multi-plane [ZTF*18, MSOC*19, FBD*19, TS20, WPYS21] or multi-sphere [BFO*20, ALG*20] images, which all aim at representing the voxel grid using a sparse approximation.

Neural Volumetric Representations. Instead of storing features or other quantities of interest using a voxel grid, these quantities can also be defined using a neural network, similar to neural implicit surfaces (see Section 3.1.2). MLP network architectures can be used to parameterize volumes, potentially in a more memory efficient manner than explicit voxel grids. Still, these representations can be expensive to sample depending on the underlying network size because for each sample, an entire feedforward pass through the network has to be computed. Most methods can be roughly classified as using global or local networks [GCV*19, GCS*20, CZ19, MPJ*19, AL20, SHN*19, SZW19, OMN*19, GYH*20, YKM*20, DNJ20, SMB*20, NMOG20, LGL*20, JJHZ20, LZP*20, KSW20]. Hybrid representations that use both grids and neural networks make a trade-off between computational and memory efficiency [PNM*20, JSM*20, CLI*20, MLL*21]. Similar to neural implicit surfaces, neural volumetric representations allow for the computation of analytic gradients, which has been used to define regularization terms in [SMB*20, TTG*21, PSB*21].

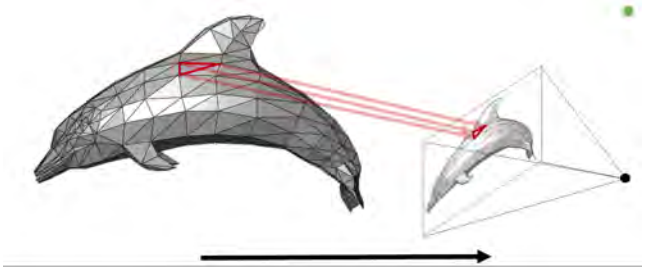
General remark: The use of coordinate-based neural networks to model scenes volumetrically (as in NeRF) superficially resembles the use of coordinate networks to model surfaces implicitly (as in neural implicit surfaces). However, NeRF-like volumetric representations are not necessarily implicit — because the output of the network is density and color, the geometry of the scene is parameterized by the network *explicitly*, not implicitly. Despite this, it is common in the literature for these models to still be called "implicit", perhaps in reference to the fact that the geometry of the scene is defined "implicitly" by the weights of a neural network (a different definition of "implicit" than is used by the SDF literature). Also note that this is a distinct definition of "implicit" than what is commonly used by the deep learning and statistic communities, where "implicit" usually refers to models whose outputs are implicitly defined as fixed points of dynamic systems, and whose gradients are computed using the implicit function theorem [BKK19].

3.2. Differentiable Image Formation

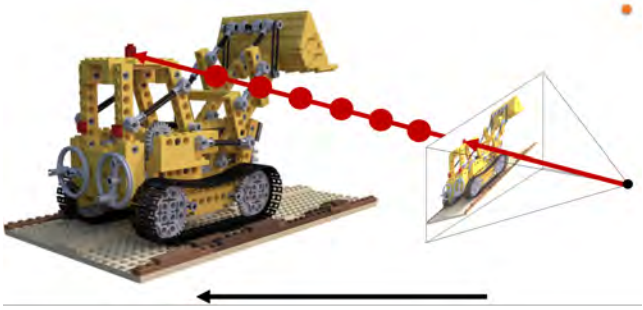
The scene representations in the previous sections allow us to represent the 3D geometry and appearance of the scene. As a next step, we describe how images can be generated from such scene representations through rendering. There are two general approaches to rendering a 3D scene into a 2D image plane: ray casting and rasterization, see also Figure 4. A rendered image of the scene can be computed by also defining the camera in the scene. Most methods use a pinhole camera, where all camera rays pass through a single point in space (focal point). With a given camera, rays from the camera origin can be cast towards the scene in order to calculate the rendered image.

Ray Casting. In the pinhole model, the basic intercept theorem can be used to describe how a point $\mathbf{p} \in \mathbb{R}^3$ in 3D is projected to the correct position $\mathbf{q} \in \mathbb{R}^2$ in the image plane. It is by definition a non-injective function and hard to invert—this puts it at the heart of the 3D reconstruction problem.

The Pinhole model has a single parameter matrix for this projection: the intrinsic matrix $\mathbf{K}$ contains the focal lengths normalized by pixel size $\mathbf{f} = [\alpha_x, \alpha_y]$, axis skew γ and center point $\mathbf{c} = [c_x, c_y]$. Using the intercept theorem and assuming homogeneous coordinates $\mathbf{p}' = [x, y, z, 1]$, we find that the projected coordinates are $\mathbf{q}' = \mathbf{K} \cdot \mathbf{p}'$,



(a) Forward Rendering (e.g., rasterization) – the image is generated by projecting the 3D representation to the image plane.



(b) Ray Casting – the image is generated by casting viewing rays, sampling the 3D representation and accumulating them. Image adapted from [MST*20].

Figure 4: For explicit surfaces representations, the surface is directly indexable. This allows us to use forward rendering methods that project the surface to the image plane and to set a pixel accordingly (e.g., using rasterization or point splatting). Implicit surface representations and volumetric representations, do not provide direct information of the surface that would allow for forward rendering, instead, the 3D space seen from the virtual camera has to be sampled to generate an image (e.g., using ray marching).

with

$$\mathbf{K} = \begin{bmatrix} \alpha_x & \gamma & c_x & 0 \\ 0 & \alpha_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

This assumes that the center of the projection is at the coordinate origin and that the camera is axis-aligned. To generalize this for arbitrary camera positions, an extrinsic matrix $\mathbf{R}$ can be used. This homogeneous 4×4 matrix $\mathbf{E}$ is composed of

$$\mathbf{E} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{t}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix},$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ is a translation vector, such that $\mathbf{R} \cdot \mathbf{p_w} + \mathbf{t} = \mathbf{p_c}$, where we use $\mathbf{p_w}$ to denote a point in world coordinates and $\mathbf{p_c}$ to denote it in camera coordinates. This definition of $\mathbf{R}$ and $\mathbf{t}$ is common in Computer Vision (for example, used by OpenCV) and referred to as 'world-to-cam' mapping, whereas in Computer Graphics (for example, in OpenGL) a similar inverse 'cam-to-world' mapping is more prevalent. Assuming the 'world-to-cam' convention and using homogeneous coordinates, we can write the full projection of $\mathbf{p_w}$ to $\mathbf{q_p}$ as:

$$\mathbf{q_p}' = \mathbf{K} \cdot \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \cdot \mathbf{p_w}'.$$

If the 'cam-to-world' convention is used, the ray casting is similarly convenient. Whereas these equations are non-injective due to the depth ambiguity, they lend themselves very well for automatic differentiation and can be optimized end-to-end in image formation models.

To model current cameras correctly, there is one more component that has to be taken into account: the lens. Leaving aside effects such as depth-of-field or motion blur, which must be modeled in the image formation process, they add distortion effects to the projection function. Unfortunately, there is no single, simple model to capture all different lens effects. Calibration packages, such as the one in OpenCV, usually implement models with up to 12 distortion parameters. They are modeled through polynomials up to degree five, hence are not trivially invertible (which is required for raycasting as opposed to point projection). More modern approaches to camera calibration use many more parameters and achieve a higher accuracy [SLPS20] and could be made invertible and differentiable.

Rasterization. An alternative to ray casting is rasterization of geometric primitives. This technique does not try to emulate the image formation process of the real world, but instead exploits the geometric properties of objects to quickly create an image. It is mostly used with meshes, which are described by a set of vertices $\mathbf{v}$ and faces $\mathbf{f}$, connecting triples or quadruplets of vertices to define surfaces. One fundamental insight is that the geometric operations in 3D can solely work with the vertices: for example, we can use the same extrinsic matrix $\mathbf{E}$ to transform each point from the world to the camera coordinate system. After this transformation, the points outside of the view frustum or points with wrong normal orientation can be culled to reduce the amount of points and faces to be processed in the next steps. The location of the remaining points projected to image coordinates can again trivially be found by using the intrinsic matrix $\mathbf{K}$ as outlined above. The face information can be used to interpolate the depth on face primitives, and the topmost faces can be stored in a z-buffer.

This way of implementing the projection is often faster than ray casting: it mainly scales with the number of visible vertices in a scene, whereas ray-casting scales with the number of pixels and the numbers of primitives to intersect with. However, it is harder to capture certain effects using it (e.g., lighting effects, shadows, reflections). It can be made differentiable through 'soft' rasterization. This has been implemented, for example, in [LLCL19, RRN*20].

3.2.1. Surface Rendering

Point Cloud Rendering. In the computer graphics literature, point cloud rendering techniques are extensively used [KB04, SP04]. As mentioned before, point clouds are discrete samples of continuous surfaces or volumes. Point cloud rendering corresponds to reconstructing the continuous signal, e.g., the appearance of a continu-

ous surface, from irregularly distributed discrete samples then resampling the reconstructed signal in the image space at each pixel location.

This process can be done in two different ways. The first approach is based on the theory of classic signal processing and can be seen as a 'soft' point splatting (similar to the soft rasterizer in the mesh rendering section below). It first constructs the continuous signal using continuous local reconstruction kernels $r(\cdot)$, i.e., $\mathbf{f} = \sum \mathbf{f}_i r(\mathbf{p}_i)$. Essentially, this approach amounts to blending the discrete samples with some local deterministic blurring kernels [LKL18, ID18, RROG18], such as EWA splatting [ZPVBG01, ZPVBG02], which is a spatially-variant reconstruction kernel that is designed to minimize aliasing. In neural rendering, the discrete samples can store some learnable features. Correspondingly, this aforementioned step effectively projects and blends the individual features into a 2D feature map. If the features have a predefined semantic meaning (e.g., colors, normals), a fixed shading function or BRDF can be used to generate the final image. If the features are learned neural descriptors, a 2D neural network is deployed to transform the 2D feature map to an RGB image. Recent neural point rendering methods that adopt this approach include SinSyn and Pulsar [WGSJ20, LZ21]. They use spatially-invariant and isotropic kernels in the blending step for performance reasons. While these simplified kernels can result in rendering artifacts such as holes, blurred edges and aliasing, these artifacts can be compensated in the neural shading step.

Alternative to the soft point splatting approach, one can use a conventional point renderer from OpenGL or DirectX. Here, each point is projected to a single pixel (or a small area of pixels) resulting in a sparse feature map. One can use a deep neural networks to reconstruct the signal directly in the image space [ASK*20b]. Note that this naive rendering approach does not provide gradients with respect to the point positions $\mathbf{p}$, and only allows to differentiate the rendering function w.r.t. the (neural) features. In contrast, the soft point splatting approaches provide point position gradients via the reconstruction kernel $r(\mathbf{p})$.

However, even in this case, the gradient is confined spatially within the support of the local reconstruction. [YSW*19b] addressed this issue by approximating the gradient using finite difference, and successfully applied the renderer to surface denoising, stylization, and multiview shape reconstruction. This idea was adopted in [RFS21b] to optimize the geometry and camera poses jointly for novel view synthesis.

Mesh Rendering. There are a number of general-purpose renderers that allow meshes to be rasterized or otherwise rendered in a differentiable manner. Among differentiable mesh rasterizers, Loper and Black [LB14] developed a differentiable rendering framework called OpenDR that approximates a primary renderer and computes the gradients via automatic differentiation. Neural mesh renderer (NMR) [KUH18] approximates the backward gradient for the rasterization operation using a handcrafted function for visibility changes. [LTJ18] proposed Paparazzi, an analytic differentiable renderer for mesh geometry processing using image filters. Petersen et al. [PBDCO19] presented *Pix2Vex*, a C^∞ differentiable renderer via soft blending schemes of nearby triangles, and [LLCL19] introduced *Soft Rasterizer*, which renders and aggregates the probabilistic maps of mesh triangles, allowing gradient flow from the rendered pixels to the occluded and far-range vertices. While most rasterizers only support rendering based on direct illumination, [LHL*21] also supports differentiable rendering of soft shadows. In the domain of physics-based rendering, [LADL18a] and [ALKN19] introduced a differentiable ray tracer to implement the differentiability of physics-based rendering effects, handling camera position, lighting and texture. In addition, Mitsuba 2 [NDVZJ19] and Taichi [HLA*19, HAL*20] are general-purpose physically based renderers that support differentiable mesh rendering via automatic differentiation, among many other graphics techniques.

Neural Implicit Surface Rendering. When the input observations are in the form of 2D images, the network which implements the implicit surface is extended to not only produce geometry-related quantities, i.e., signed distance values, but also appearance-related quantities. An implicit differentiable renderer [SZW19, NMOG20, LZP*20, LSCL19, YKM*20, KJJ*21, BKW21, TLY*21] can be implemented by first finding the intersection between a viewing ray and the surface using the geometric branch of the neural implicit function, and then obtaining the RGB value of this point from the appearance branch. The search of surface intersection is typically based on some variant of the sphere tracing algorithm [Har96]. Sphere tracing iteratively samples the 3D space from the camera center in the direction of the view ray until the surface is reached. Sphere tracing is an optimized ray marching approach that adjusts the step size by the SDF value sampled at the previous location, but still this iterative strategy can be computationally expensive. Takikawa et al. [TLY*21] improved the rendering performance by adapting the ray-tracing algorithm to the sparse octree data structure. A common problem for implicit surface rendering for joint geometry and appearance estimation from 2D supervision is the ambiguity of geometry and appearance. In [NMOG20, YKM*20, KJJ*21, BKW21], foreground masks were extracted from the 2D images to provide additional supervision signals for the geometry branch. Recently, [OPG21] and [YGKL21b] addressed this issue by formulating the surface function into the volumetric rendering formulation (introduced below); on the other hand [ZYQ21] use off-the-shelf depth estimation methods to generate pseudo ground truth signed distance values to assist the training of the geometry branch.

3.2.2. Volumetric Rendering

Volumetric rendering is based on ray casting and has proven to be effective in neural rendering and, especially, in learning a scene representation from multi-view input data. Specifically, the scene is represented as a continuous field of volume density or occupancy rather than a collection of hard surfaces.

This means that rays have some probability of interacting with the scene content at each point in space, rather than a binary intersection event. This continuous model works well as a differentiable rendering framework for machine learning pipelines that rely heavily on the existence of well-behaved gradients for optimization.

Though fully general volumetric rendering does account for

"scattering" events where rays can be reflected off of a volumetric particles [Jar08], we will limit this summary to the basic model commonly used by neural volumetric rendering methods for view synthesis [LH96, Max95], which only accounts for "emission" and "absorption" events, where light is emitted or blocked by a volumetric particle.

Given a set of pixel coordinates, we can use the camera model previously described to calculate the corresponding ray through 3D space with origin $\mathbf{p}$ and direction ω_o. The incoming light along this ray can be defined using a simple emission/absorption model as

$$L(\mathbf{p},\omega_o) = \int_{t_0}^{t_1} T(\mathbf{p},\omega_o,t_0,t)\sigma(\mathbf{p}+t\omega_o)L_e(\mathbf{p}+t\omega_o,-\omega_o)\,dt\,, \quad (7)$$

where σ is volume density at a point, L_e is emitted light at a point and direction, and transmittance T is a nested integral expression

$$T(\mathbf{p},\,\omega_o,\,t_0,\,t) = \exp\left(-\int_{t_0}^{t}\sigma(\mathbf{p}+s\omega_o)\,ds\right). \quad (8)$$

Density denotes the differential probability that a ray interacts with the volumetric "medium" of the scene at a particular point, whereas transmittance describes how much light will be attenuated as it travels back toward the camera from point $p+t\omega_o$.

These expression can only be evaluated analytically for simple density and color fields. In practice, we typically use quadrature to approximate the integrals, where σ and L_e are assumed to be piecewise-constant within a set of N intervals $\{[t_{i-1},t_i)\}_{i=1}^{N}$ that partition the length of the ray:

$$L(\mathbf{p},\,\omega_o) \approx \sum_{i=1}^{N} T_i\alpha_i L_e^{(i)}\,, \quad (9)$$

$$T_i = \exp\left(-\sum_{j=1}^{i-1}\Delta_j\sigma_j\right), \quad (10)$$

$$\alpha_i = 1-\exp(-\Delta_i\sigma_i)\,, \quad (11)$$

$$\Delta_i = t_i - t_{i-1}\,. \quad (12)$$

For a full derivation of this approximation, we refer the reader to Max and Chen [MC10]. Note that when written in this form, the expression for approximating L exactly corresponds to alpha compositing the colors $L_e^{(i)}$ from back to front [PD84].

NeRF [MST*20] and related methods (e.g., [MBRS*21, NG21b, PCPMMN21, SDZ*21, ZRSK20, NSP*21]) use differentiable volume rendering to project the scene representations into 2D images. This allows these methods to be used in an "inverse rendering" framework, where a three- or higher-dimensional scene representation is estimated from 2D images. Volume rendering requires many samples to be processed along a ray, each requiring a full forward pass through the network. Recent work has proposed enhanced data structures [YLT*21, HSM*21, GKJ*21], pruning [LGL*20], importance sampling [NSP*21], fast integration [LMW21], and other strategies to accelerate the rendering speed, although training times of these methods are still slow. Adaptive coordinate networks accelerate training using a multi-resolution network architecture that is optimized during the training phase by allocating available network capacity in an optimal and efficient manner [MLL*21].

3.3. Optimization

At the heart of training neural networks lies a non-linear optimization which aims to apply the constraints of the training set in order to obtain a set of neural network weights. As a result, the function which is approximated by the neural network is fit to the given training data. Typically, optimization of the neural networks is gradient-based; more specifically SGD variants such as Momentum or Adam [KB14] are utilized, where the gradients are obtained by leveraging the backpropagation algorithm. In the context of neural rendering, the neural network implements the 3D scene representation, and the training data consists of 2D observations of the scene. The renderings obtained using differentiable rendering of the neural scene representations is compared with the given observation using various loss functions. These reconstruction losses can be realized with per-pixel L1 or L2 terms, but also using perceptual [JAFF16] or even discriminator-based loss formulations [GPAM*14]. However, key is that the losses are directly coupled with the respective differentiable rendering formulation in order to update the scene representations, cf. Section 3.1.

4. Applications

In this section, we discuss the specific applications of neural rendering and the underlying neural scene representations. We first discuss improvements to novel view synthesis of static content in Section 4.1. We then give an overview over methods that generalize across objects and scenes in Section 4.2. After that, Section 4.3 discusses non-static, dynamic scenes. We next turn to editing and composing scenes in Section 4.4. Then we provide an overview over relighting and material editing in Section 4.5. Finally, we discuss several engineering frameworks in Section 4.6. We also develop a taxonomy of the different methods for each application. These are presented in Table 1, Table 2, Table 3, Table 4, and Table 5, respectively.

4.1. Novel View Synthesis of Static Content

Novel view synthesis is the task of rendering a given scene from new camera positions, given a set of images and their camera poses as input. Most of the applications presented later in this section generalize the task of view synthesis in some way: in addition to being able to move the camera, they might allow moving or deforming objects within the scene, changing the lighting, and so on.

View synthesis methods are evaluated on a few salient criteria. Clearly, output images should look as realistic as possible. However, this is not the whole story — perhaps even more important is *multiview 3D consistency*. Rendered video sequences must appear to portray consistent 3D content as the camera moves through the scene, without flickering or warping. As the field of neural rendering has matured, most methods have moved in the direction of producing a fixed 3D representation as output that can be used to render new 2D views, as explained in the scope. This approach automatically lends a degree of multiview consistency that has historically been hard to achieve when relying too heavily on black-box 2D convolutional networks as image generators or renderers.

Method	Required Data	Requires Pre-trained NeRF	3D Representation	Persistent 3D	Network Inputs	Code
Mildenhall et al. [MST*20]	I+P	✗	V	F	PE(P)+PE(V)	
Sitzmann et al. [SZW19]	I+P	✗	S	P	P	
Niemeyer et al. [NMOG20]	I+P+M	✗	O	F	P	
Chen et al. [CZ19]	S	✗	O	F	P	
Gu et al. [LGL*20]	I+P	✗	G+V	F	PE(P)+PE(V)	
Lindell et al. [LMW21]	I+P	✗	V	P	PE(P)+PE(V)	
Reiser et al. [RPLG21]	I+P	✓	G+V	F	PE(P)+PE(V)	
Garbin et al. [GKJ*21]	I+P	✓	G	F	P+V	✗
Hedman et al. [HSM*21]	I+P	✓	G	F	P+PE(V)	
Yu et al. [YLT*21]	I+P	✓	G	F	P+V	
Neff et al. [NSP*21]	I+P+D	✗	V	F	PE(P)+PE(V)	
Sitzmann et al. [SRF*21]	I+P	✗	✗	N	L	

Table 1: Selected methods for static scene view synthesis presented in Section 4.1. Although some of these representations are used for applications beyond static scene view synthesis, in this table we only classify such methods based on the use of their underlying 3D scene representation for static scene view synthesis. I: Images, P: Camera poses (exact or approximate), S: 3D shape, M: Object masks, D: Depth. G: Grid, V: Neural volumetric, S: Neural SDF, O: Neural occupancy. F: Fully. P: Partial, N: Not guaranteed. P: 3D position, V: 2D viewing direction, L: Light field ray coordinates. PE(): Positinal encoding of argument.

In Table 1, we give an overview over the discussed methods.

4.1.1. View Synthesis from a 3D Voxel Grid Representation

We will briefly review the recent history of view synthesis using 3D voxel grids and a volumetric rendering model.

DeepStereo [FNPS16] presented the first end-to-end deep learning pipeline for view synthesis. This work included many concepts that have now become commonplace. A convolutional neural network is presented with input images in the form of a plane sweep volume (PSV), where each nearby input is reprojected to a set of candidate depth planes, requiring the network to simply evaluate how well the reprojections match for each pixel at each candidate depth. The CNN's outputs are converted into a probability distribution over depths using a softmax, which is then used to combine a stack of proposed color images (one per depth plane). The final loss is only enforced on the pixel-wise difference between the rendered output and a heldout target image, with no intermediate heuristic losses required.

A major drawback of DeepStereo is that it requires running a CNN to estimate depth probabilities and produce each output frame independently, resulting in slow runtime and a lack of multiview 3D consistency. Stereo Magnification [ZTF*18] directly addresses this issue, using a CNN to process a plane sweep volume directly into an output persistent 3D voxel grid representation

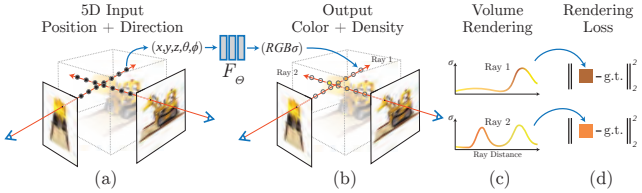


Figure 5: An overview of the neural radiance field (NeRF) scene representation and volume rendering procedure. NeRF synthesizes images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce color and volume density (b), and using volume rendering to composite these values into an image (c). Since this rendering function is differentiable, the NeRF scene representation MLP can be optimized by minimizing the residual between synthesized and ground truth observed images (d). Digital zoom recommended. Image adapted from [MST*20].

named a "multiplane image," or MPI. Rendering new views simply requires using an alpha compositing to render the RGB-alpha grid from a new location. In order to achieve high image quality, Stereo Magnification heavily distorts the parameterization of its 3D grid to bias it to the frame of reference of one of the two input views. This significantly decreases storage requirements for the dense grid but means that new views can only be rendered in the direct neighborhood of the input stereo pair. This shortcoming was later addressed by improving the training procedure for a single MPI [STB*19], providing many more than two input images to the network [FBD*19], or combining multiple MPIs together to represent a single scene [MSOC*19].

All methods mentioned above use a feed-forward neural network to map from a limited set of input images to an output image or 3D representation and must be trained on a large dataset of pairs of input/output views. In contrast, DeepVoxels [STH*19] optimizes a 3D voxel grid of features jointly with a learned renderer using images of a *single* scene, without requiring any external training data. Similarly, Neural Volumes [LSS*19] optimizes a 3D CNN to produce an output volumetric representation for a single scene of multiview video data This single-scene training paradigm has greatly increased in popularity recently, leveraging the unique "self-supervised" aspect of view synthesis: any input images can also be used as supervision via a rerendering loss. In comparison to MPI-based methods, DeepVoxels and Neural Volumes also use a 3D voxel grid parameterization that is not heavily skewed to one particular viewing direction, allowing novel views to be rendered observing the reconstructed scene from any direction.

It is worth mentioning that a number of computer vision papers focused primarily on 3D shape reconstruction (rather than realistic image synthesis) adopted an alpha compositing volumetric rendering model in parallel with this view synthesis research [HRRR18, KHM17, TZEM17]; however, these results were heavily constrained by the memory limitations of 3D CNNs and could not produce voxel grid outputs exceeding 128^3 resolution.

4.1.2. View Synthesis from a Neural Network Representation

To address the resolution and memory limitations of voxel grids, Scene Representation Networks (SRNs) [SZW19] combined a sphere-tracing based neural renderer with a multilayer perceptron (MLP) as a scene representation, focusing mainly on generalization across scenes to enable few-shot reconstruction. Differentiable Volumetric Rendering (DVR) [NMOG20] similarly leveraged a surface rendering approach, but demonstrated that overfitting on single scenes enables reconstruction of more complex appearance and geometry.

Neural radiance fields (NeRF [MST*20] signified a breakthrough in the application of MLP-based scene representations to single-scene, photorealistic novel view synthesis, see Figure 5. Instead of a surface-based approach, NeRF directly applies the volume rendering model described in Section 3.2.2 to synthesize images from an MLP that maps from an input position and viewing direction to output volume density and color. A different set of MLP weights are optimized to represent each new input scene based on pixelwise rendering loss against the input images.

This overall framework shares many similarities with the work described in the previous section. However, MLP-based scene representations can achieve higher resolution than discrete 3D volumes by virtue of effectively *differentiably* compressing the scene during optimization. For example, a NeRF representation capable of rendering 800×800 resolution output images only required 5MB of network weights. In comparison, an 800^3 RGBA voxel grid would consume close to 2GB of storage.

This ability can be attributed to NeRF's use of a *positional encoding* applied to the input spatial coordinates before passing through the MLP. In comparison to the previous work on using neural networks to represent implicit surfaces [PFS*19, CZ19] or volumes [MON*19], this allows NeRF's MLP to represent much higher frequency signals without increasing its capacity (in terms of number of network weights).

The main drawback of switching from a discrete 3D grid to an MLP-based representation is rendering speed. Rather than directly querying a simple data structure, calculating the color and density value for a *single point* in space now requires evaluating an entire neural network (hundreds of thousands of floating point operations). On a typical desktop GPU, an implementation of NeRF in a standard deep learning framework takes tens of seconds to render a single high resolution image.

4.1.3. Improving Rendering Speed

Several different methods have been proposed for speeding up volumetric rendering of MLP-based representations. Neural Sparse Voxel Fields [LGL*20] builds and dynamically updates an octree structure while the MLP is optimized, allowing for aggressive empty space skipping and early ray termination (when the transmittance along the ray approaches zero). KiloNeRF [RPLG21] combines empty space skipping and early termination with a dense 3D grid of MLPs, each with a much smaller number of weights than a standard NeRF network.

Three concurrent works recently proposed methods for caching the values various quantities learned by the NeRF MLP on a sparse 3D grid, allowing for realtime rendering once training is complete. Each method modifies the way in which view-dependent colors are predicted in order to facilitate faster rendering and smaller memory requirements for the cached representations. SNeRG [HSM*21] stores volume density and a small spatially-varying feature vector in a sparse 3D texture atlas, using fast shader for compositing these values along a ray and running a tiny MLP decoder to produce view-dependent color for each ray. FastNeRF [GKJ*21] caches volume density along with weights for combining a set of learned spherical basis functions that produce view-varying colors at each point in 3D. PlenOctrees [YLT*21] queries the MLP to produce a sparse voxel octree of volume density and spherical harmonic coefficients and further finetunes this octree representation using a rendering loss to improve its output image quality.

NeX-MPI [WPYS21] combines the multiplane image parameterization with an MLP scene representation, with view dependent effects parameterized as a linear combination of globally learned basis functions. Because the model is supervised directly on a 3D MPI grid of coordinates, this grid can be easily cached to render new views in real time once optimization is complete.

An alternative approach for accelerating rendering is to train the MLP representation itself to effectively precompute part or all of the volume integral along the ray. AutoInt [LMW21] trains a network to "automatically integrate" the output color value along ray segments by supervising the *gradient* of the network to behave like a standard NeRF MLP. This allows the rendering step to break the integral along a ray into an order of magnitude fewer segments than the standard quadrature estimate (down to as few as 2 or 4 samples), trading off speed for a minor loss in quality. Light Field Networks [SRF*21] takes this a step further, optimizing an MLP to directly encode the mapping from an input ray to an output color (the scene's light field). This enables rendering with only a *single* evaluation of the MLP per ray, in contrast to hundreds of evaluations for volume- and surface-based renderers, and enables real-time novel view synthesis. These methods present a tradeoff between rendering speed and multiview consistency: reparameterizing the MLP representation as a function of rays rather than 3D points means that the scene is no longer guaranteed to appear consistent when viewed from different angles. In this case multiview consistency must be enforced through supervision, either by providing a very large number of input images or learning this property via generalization across a dataset of 3D scenes.

4.1.4. Miscellaneous Improvements

A variety of papers have augmented the rendering model, supervision data, or robustness of volumetric MLP scene representations.

Depth Supervision. DONeRF [NSP*21] trains an "depth oracle" network to predict sample locations along each ray, drastically reducing the number of samples sent through the NeRF MLP and allowing interactive rate rendering. However, this method is supervised with dense depths maps, which are challenging to obtain for real data. Depth-supervised NeRF [DLZR21] directly supervises the output depths from NeRF (in the form of expected termination depth along each ray) using the sparse point cloud output which is a byproduct of estimating camera poses using structure-from-motion.

NerfingMVS [WLR*21] uses a multistage pipeline for depth supervision, first finetuning a single-view depth estimation network on sparse structure-from-motion depth estimates, then uses the resulting dense depth maps to guide NeRF optimization.

Optimizing Camera Poses. NeRF-- [WWX*21] and Self-Calibrating Neural Radiance Fields [JAC*21] jointly optimize the NeRF MLP and input camera poses, bypassing the need for structure-from-motion preprocessing for forward facing scenes. Bundle-Adjusting Neural Radiance Fields (BARF) [LMTL21] extends this idea by applying a coarse-to-fine annealing schedule to each frequency component of the positional encoding function, providing a smoother optimization trajectory for joint reconstruction and camera registration. However, neither of these methods can optimize poses from scratch for wide-baseline 360 degree captures. GNeRF [MCL*21] achieves this by training a set of cycle consistent networks (a generative NeRF and a pose classifier) that map from pose to image patches and back to pose, optimizing until the classified pose of real patches matches that of sampled patches. They alternate this GAN training phase with a standard NeRF optimization phase until the result converges.

Hybrid Surface/Volume Representations. The Implicit Differentiable Renderer (IDR) from Yariv et al. [YKM*20] combines a DVR-like implicit surface MLP with a NeRF-like view dependent branch which takes viewing direction, implicit surface normal, and the 3D surface point as inputs and predicts the view-varying output color. This work shows that including the normal vector as input to the color branch helps the representation disentangle geometry and appearance more effectively. It also demonstrates that camera pose can be jointly optimized along with the shape representation to recover from small miscalibration errors.

UNISURF [OPG21] proposes a hybrid MLP representation that unifies surface and volume rendering. To render a ray, UNISURF uses root finding to get a "surface" intersection point, treating the volume as an occupancy field, then distributes volume rendering samples only within an interval around that point. The width of this interval monotonically decreases over the course of optimization, allowing early iterations to supervise the whole training volume and later stages to more efficiently refine the surface with tightly spaced samples. Azinovic et al. [AMBG*21] propose to use an SDF representation instead of volume densities to reconstruct scenes from RGB-D data. They convert the sdf values to densities that can be used in the NeRF formulation. NeuS [WLL*21] ties the volume density to an signed distance field and reparameterizes the transmittance function such that it achieves its maximal slope precisely at the zero-crossing of this SDF, allowing an unbiased estimate of the corresponding surface. VolSDF [YGKL21b] uses an alternate mapping from SDF to volume density, which allows them to devise a new resampling strategy to achieve provably bounded error on the approximated opacity in the volume rendering quadrature equation. In [LFS*21], the authors propose a method called MINE which is a hybrid between multi-plane images (MPI) and NeRF. They are able to reconstruct dense 3D reconstructions from single color images which they demonstrate on RealEstate10K, KITTI and Flowers Light Fields.

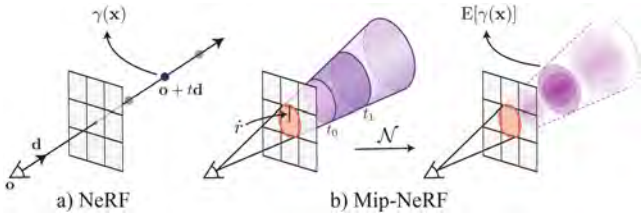


Figure 6: Instead of sampling points **x** along the rays traced from the camera projection center (a), MipNeRF [BMT*21] reasons about 3D canonical frustum per camera pixel (b). Image adapted from [BMT*21].

Robustness and Quality. NeRF++ [ZRSK20] provides a "inverted sphere" parameterization of space that can allow NeRF to large-scale, unbounded 3D scenes. Points outside the unit sphere are inverted back into the unit sphere and passed through a separate MLP.

NeRF in the Wild [MBRS*21] adds additional modules to the MLP representation to account for inconsistent lighting and objects across different images. They apply their robust model to the Photo-Tourism dataset [SSS06] (consisting of internet images of famous landmarks across the world) and are able to remove transient objects such as people and cars and capture time-varying appearance through use of a latent code embedding associated with each input image.

MipNeRF [BMT*21] modifies the positional encoding applied to 3D points to incorporate the pixel footprint, see Figure 6. By pre-integrating the positional encoding over a conical frustum corresponding to each quadrature segment sampled along the ray, MipNeRF can be trained to encode a representation of the scene at multiple different scales (analogously to a mipmap of a 2D texture), preventing aliasing when rendering the scene from dramatically varying positions or resolutions.

4.2. Generalization over Object and Scene Classes

While a significant amount of prior work addresses generalization over multiple scenes and object categories for voxel-based, mesh-based, or non-3D structured neural scene representations, we focus this discussion on recent progress in generalization leveraging MLP-based scene representations. Where approaches that overfit a single MLP on a single scene [MST*20, YKM*20] require a large number of image observations, the core objective of generalizing across scene representations is novel view synthesis given few or potentially only a single input view. In Table 2, we give an overview over the discussed methods, classified by whether they leverage local or global conditioning, whether they can be used as unconditional generative models or not, what kind of 3D representation they leverage (volumetric, SDF, or occupancy), what kind of training data they require, and how inference is performed (amortized with an encoder, via the auto-decoder framework, or via gradient-based meta-learning).

We may differentiate two key approaches in generalizing across scenes. One line of work follows an approach reminiscent of image-based rendering [CW93, SK00], where multiple input views are

Method	Conditioning	Required Data	3D Representation	Class Specific Prior	Generative Model	Inference Type	Code
Yu et al. [YYTK21]	L	G	V	✗	✗	A	↗
Raj et al. [RZS*20]	L	F	V	✗	✗	A	✗
Trevithick et al. [TY20]	L	G	V	✗	✗	A	↗
Wang et al. [WWG*21]	L	G	V	✗	✗	A	✗
Reizenstein et al. [RSH*21]	L	G	V	✗	✗	A	✗
Sitzmann et al. [SZW19]	G	G	S	✓	✓	D	↗
Kosiorek et al. [KSZ*21]	G	G	V	✗	✓	A	✗
Rematas et al. [RMBF21]	G	G	V	✓	✓	D	↗
Xie et al. [XPMBB21]	G	G	V	✗	✗	A	✗
Tancik et al. [TMW*21]	G	G	V	✗	✗	GB	↗
Gao et al. [GSL*20]	G	F	V	✗	✗	GB	✗
Nguyen-Phuoc et al. [NPLT*19]	G	G	V	✓	✓	✗	↗
Schwarz et al. [SLNG20]	G	G	V	✓	✓	✗	✗
Chan et al. [CMK*21]	G	G	V	✓	✓	✗	↗
Anonymous [Ano22]	G	G	V	✓	✓	✗	✗
Niemeyer et al. [NG21a]	G	G	V	✓	✓	✗	✗

Table 2: Selected methods for generalization presented in Section 4.2. G: Global, L: Local. I: Implicitly, E: Explicitly. G: General, B: Body, F: Face. A: Amortized/Encoder, D: Auto-Decoder, GB: Gradient-based. V: Neural volumetric, S: Neural SDF.

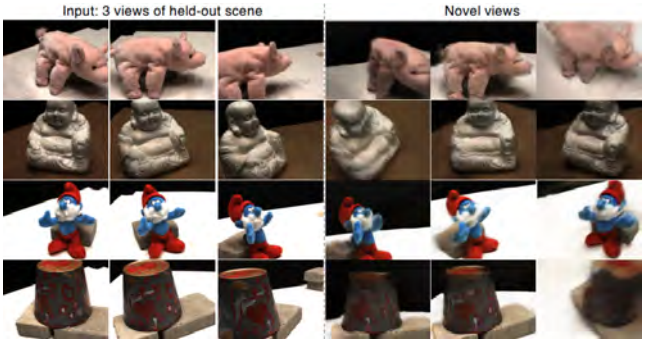


Figure 7: Input images from DTU MVS dataset [JDV*14] and novel views obtained by PixelNeRF [YYTK21] with no test-time optimization. Furthermore, training and test sets do not share the same scenes. Image adapted from [YYTK21].

warped and blended to synthesize a novel viewpoint. In the context of MLP-based scene representations, this is often implemented via *local conditioning*, where the coordinate input to the scene representation MLP is concatenated with a locally varying feature vector, stored in a discrete scene representation, such as a voxel grid [PNM*20]. PiFU [SHN*19] uses an image encoder to compute features on the input image and conditions a 3D MLP on these features via projecting 3D coordinates on the image plane - however, PiFU did not feature a differentiable renderer, and so required ground-truth 3D supervision. PixelNeRF [YYTK21] (see Figure 7) and Pixel-Aligned Avatars [RZS*20] leverage this approach in a volume rendering framework where these features are aggregated over multiple views, and a MLP produces color and density fields that are rendered as in NeRF. When trained on multiple scenes, they learn scene priors for reconstruction, that enable high fidelity reconstruction of scenes from a few views. PixelNeRF can also be trained on specific object categories, enabling object instance 3D reconstruction from one or multiple posed images. GRF [TY20] uses a similar framework, with an additional attention module that reasons about the visibility of the 3D point in the different sampled input images. Stereo Radiance Fields [CBLPM21] similarly extracts features from several context views, but leverages learned correspondence matching between pairwise features across context images to aggregate features across context images instead of a simple mean aggregation. Finally, IBRNet [WWG*21] and NeRFormer [RSH*21] introduce transformer networks across the ray samples that reason about visibility.

An alternative to such image-based approaches aims to learn a monolithic, global representation of a scene instead of relying on images or other discrete spatial data structures. This is accomplished by inferring a set of weights for the scene representation MLP that describes the whole scene, given a set of observations. One line of work accomplishes this by encoding a scene in a single, low-dimensional latent code that is then used to condition the scene representation MLP. Scene Representation Networks (SRNs) [SZW19] map low-dimensional latent codes to the parameters of a MLP scene representation via a hypernetwork, and subsequently render the resulting 3D MLP via ray-marching. To reconstruct an instance given a posed view, SRNs optimize the latent code so that its rendering matches the input view(s). Differentiable Volumetric Rendering [NG20] similarly uses surface rendering, but computes its gradients analytically and performs inference via a CNN encoder. Light Field Networks [SRF*21] leverage low-dimensional latent codes to directly parameterize the 4D light field of the 3D scene, enabling single-evaluation rendering. NeRF-VAE embeds a NeRF in a variational auto-encoder, similarly representing the whole scene in a single latent code, but learning a generative model that enables sampling [KSZ*21]. Sharf [RMBF21] uses a generative model of voxelized shapes of objects in a category, which in turn condition a higher resolution neural radiance field that is rendered using volume rendering for higher novel view synthesis fidelity. Fig-NeRF [XPMBB21] models an object category as a template shape conditioned on a latent code, that undergoes a deformation that is also conditioned on the same latent variable. This enables the network to explain certain shape variations as more intuitive deformations. Fig-NeRF focuses on retrieving an object category from real object scans, and also proposes using a learn background model to segment the object from its background. An alternative to representing the scene as a low-dimensional latent code is to quickly optimize the weights of an MLP scene representation in a few optimization steps via gradient-based meta-learning [SCT*20]. This can be used to enable fast reconstruction of neural radiance fields from few images [TMW*21]. The pre-trained models converge faster when trained on a novel scene, and require fewer views compared to standard neural radiance field training. Portrait-

NeRF [GSL*20] proposes a meta-learning approach to recover a NeRF from a single frontal image of a person. To account for differences in pose between the subjects, it models the 3D portraits in a pose-agnostic canonical reference frame, that is warped for each subject using 3D keypoints. Bergman et al. [BKW21] leverage gradient-based meta-learning and local conditioning on image features to quickly recover a NeRF of a scene.

Instead of *inferring* a low-dimensional latent code conditioned on a set of observations of the sought-after 3D scene, a similar approach can be leveraged to learn *unconditional* generative models. Here, a 3D scene representation equipped with a neural renderer is embedded in a generative adversarial network. Instead of inferring low-dimensional latent codes from a set of observations, we define a distribution over latent codes. In a forward pass, we sample a latent from that distribution, condition the MLP scene representation on that latent, and render an image via the neural renderer. This image can then be used in an adversarial loss. This enables learning of a *3D* generative model of shape & appearance of 3D scenes given only *2D* images. This approach was first proposed with 3D scene representations parameterized via voxelgrids [NPLT*19]. GRAF [SLNG20] first leveraged a conditional NeRF in this framework and achieved significant improvements in photorealism. Pi-GAN [CMK*21] further improved on this architecture with a FiLM-based conditioning scheme [PSDV*18] of a SIREN architecture [SMB*20]. StyleNeRF [Ano22] further improves image quality by adopting the condition mechanism of StyleGAN, and improves runtime (enabling higher-resolution image generation) by only leveraging NeRF to generate low-resolution feature maps and subsequent upsampling with a carefully designed upsampling network that avoids aliasing to ensure multi-view consistency. While these approaches do not require more than observation per 3D scene and thus, also no ground-truth camera poses, they still require knowledge of the *distribution* of camera poses (i.e., for portrait images, the distribution over camera poses must produce plausible portrait angles). CAMPARI [NG21a] addresses this constraint by jointly learning camera pose distribution and generative model. GIRAFFE [NG21b] proposes to learn a generative model of scenes composed of several objects by parameterizing a scene as a composition of several foreground (object) NeRFs and a single background NeRF. Latent codes are sampled for each NeRF separately, and a volume renderer composes them to a coherent 2D image.

4.3. Learning to Represent and Render Non-static Content

While the original neural radiance fields [MST*20] are used to represent static scenes and objects, there are approaches that can additionally handle dynamically changing content. In Table 3, we give an overview over the discussed methods.

These approaches can be categorized in time-varying representations that allow to do novel viewpoint synthesis of a dynamically changing scene as an unmodified playback (e.g., to produce a bullet-time effect), or in techniques that also give control over the deformation state, thus, allowing for novel-view point synthesis and editing of the content. The deforming neural radiance field can be achieved implicitly or explicitly, see Figure 8:

Method	Data	Deformation	Class-Specific Prior	Controllable Parameters	Code
Lombardi et al. [LSS*21]	MV	I	G	V,R	↗
Li et al. [LNSW21]	MV	I	G	V,R	✗
Xian et al. [XHKK21]	Mo	I	G	V,R	✗
Gao et al. [GSKH21]	Mo	I	G	V,R	✗
Du et al. [DZY*21]	Mo	I	G	V,R	↗
Pumarola et al. [PCPMMN21]	Mo	E	G	V,R	↗
Park et al. [PSB*21]	Mo	E	G	V,R	↗
Tretschk et al. [TTG*21]	Mo	E	G	V,R	↗
Park et al. [PSH*21]	Mo	I+E	G	V,R	↗
Attal et al. [ALG*21]	Mo+D	I	G	V,R	↗
Li et al. [LSZ*21]	MV	I	G	V,R	✗
Gafni et al. [GTZN21]	Mo	E	F	V,R,E	↗
Wang et al. [WBL*20]	MV	I	F	V,R	✗
Guo et al. [GCL*21]	Mo	I	F	V,R,E	↗
Noguchi et al. [NSLH21]	Mo+3D	E	B	V,R,E	↗
Su et al. [SYZR21]	Mo	E	B	V,R,E	✗
Peng et al. [PDW*21]	MV	E	B	V,R,E	↗
Peng et al. [PZX*21]	MV	I+ E	B	V,R	↗
Liu et al. [LHR*21]	MV	E	B	V,R,E	✗
Xu et al. [XAS21]	MV+Mo	I	B	V,R,E	✗

Table 3: Selected methods for non-static, dynamic scenes presented in Section 4.3. MV: Multi-View, Mo: Monocular, D: Depth, 3D: 3D pose. I: Implicitly, E: Explicitly. G: General, B: Body, F: Face/Head. V: Viewpoint, R: Replay, E: Editing.

- Implicitly, by conditioning the NeRF on a representation of the deformation state (e.g., a time input)
- Explicitly, by using a separate deformation field that can map from the deformed space to a canonical space where the NeRF is embedded.

4.3.1. Time-varying Neural Radiance Fields

Time-varying neural radiance fields allow to playback a video with novel view points, see Figure 9. Since they forego control, these methods do not rely on a specific motion model and can thus handle general objects and scenes.

Several extensions of NeRF for non-rigid scenes were proposed concurrently. We first discuss methods that model deformations implicitly [LNSW21, XHKK21, GSKH21, DZY*21]. While the original NeRF is static and takes as input only a 3D spatial point, it can be extended in a straightforward manner to become time-varying: the volume can additionally be conditioned on a vector that represents the deformed state. In current methods, this conditioning takes the form of a time input (potentially positionally encoded)

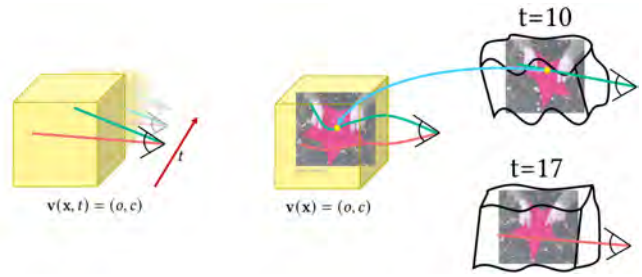


Figure 8: Current methods for modelling deformations with neural radiance fields fall into two categories. Left: Implicitly, by conditioning the radiance field, **v**, on the deformation (here: time t). Right: Explicitly, by warping space with a separate deformation MLP that regresses offsets (blue arrow) from the deformed space (black) into static canonical space (yellow). This bends straight rays into the canonical radiance field. Image adapted from [TFT*21].

Figure 9: An input monocular video of a general deformable scene and novel view rendering thereof by space-time neural irradiance fields of Xian et al. [XHKK21]. Image adapted from [XHKK21].

[XHKK21,LNSW21,GSKH21,DZY*21,PCPMMN21] or an auto-decoded latent code per time step [PSB*21,TTG*21,PSH*21].

Since handling non-rigid scenes without prior knowledge of object type or 3D shape is an ill-posed problem, methods of this class adopt various geometric regularizers and condition learning on additional data modalities. To encourage consistency of reflectance and opacity across time, several methods learn scene-flow mappings between temporally neighboring time steps [LNSW21, XHKK21, GSKH21, DZY*21]. Since this is restricted to small temporal neighborhoods, artifact-free novel-view synthesis is predominantly demonstrated on spatio-temporal camera trajectories that are close to the spatio-temporal input camera trajectories. The scene-flow mapping can be trained with reconstruction losses that warp the scene from other time steps into the current time step [LNSW21, DZY*21], by encouraging consistency between estimated optical flow and the 2D projection of the scene flow [LNSW21,GSKH21], or by tracking backprojected keypoints in 3D [DZY*21]. The scene flow is often constrained with additional regularization losses [LNSW21, XHKK21, GSKH21, DZY*21], e.g., to encourage spatial or temporal smoothness or forward-backward cycle consistency. Unlike the other methods mentioned, Neural Radiance FLow (NeRFlow) of Du et al. [DZY*21] models deformations with infinitesimal displacements that need to be integrated with Neural ODE [CRBD18] to obtain offsets.

In addition, several methods use estimated depth maps to supervise the geometry estimation [LNSW21, XHKK21, GSKH21, DZY*21]. One limitation of this regularization is that the accuracy of the reconstruction depends on the accuracy of monocular depth estimation methods. As a result, artefacts of monocular depth estimation methods are recognizable in the novel views [XHKK21].

Finally, the static background is often handled separately, allowing it to exploit multi-view clues from monocular input recordings across time. To that end, some methods estimate a second static volume that is not conditioned on the deformation [LNSW21, GSKH21] or introduce soft regularization losses to constrain static scene content [XHKK21]. Gao *et al.* [GSKH21], a follow-up to Xian *et al.*'s work [XHKK21], train the static NeRF on observations that do not contain moving and deforming parts with the help of a binary segmentation mask (one of the inputs to the model and user-provided).

One advantage of Guo et al.'s method is that it produces the most accurate quantitative and qualitative results on the challenging dataset of Yoon *et al.* [YKG*20] (compared to Tretschk et al. [TTG*21] and Li et al. [LNSW21]). The latter dataset was initially introduced for novel-view synthesis from a comparably sparse set of input monocular views of dynamic scenes with moderate changes in the camera poses. Limitations of the method include strong reliance on optical flow and handling of arbitrary non-rigid deformations (in contrast to scenes composed of independent rigidly moving objects).

Finally, NeRFlow can be used to de-noise and super-resolve views of pre-trained scenes. Limitations of NeRFlow, which the authors mention, include difficulty in preserving static backgrounds, handling complex scenes (non-piecewise-rigid deformations and motions) and rendering novel views at substantially different camera trajectories compared to the input ones.

The methods discussed so far model deformations implicitly by conditioning the scene representation on the deformation. This makes controllability of the deformation cumbersome and difficult. Other works instead disentangle the deformations from the geometry and appearance: they factor out the deformations into a separate function on top of a static canonical scene, a crucial step towards controllability. The deformations are accomplished by shooting straight rays into deformed space and then bending them into the canonical scene, usually by regressing per-point offsets for points on the straight ray using a coordinate-based MLP that is conditioned on the deformation. This can be thought of as space warping or scene flow. In contrast to implicit modelling, these methods share geometry and appearance information across time by construction via the static canonical scene, thereby providing hard correspondences, which do not drift. Due to that hard constraint, unlike implicit methods, current methods with explicit deformations cannot handle topological changes and only demonstrate results on scenes with significantly smaller motion than implicit methods.

D-NeRF [PCPMMN21] uses an unregularized ray-bending MLP to model deformations of a single or multiple synthetic objects segmented from the background and observed by virtual cameras. It assumes a pre-defined set of multi-view images given, though, at training time, only a single view chosen arbitrarily is used for supervision at any time. Thus, D-NeRF can be considered an inter-

mediate step between techniques with multi-view supervision and truly monocular approaches.

Several works show results on real-world scenes observed by a moving monocular camera. The core application of Deformable NeRF of Park *et al.* [PSB*21] is the creation of Nerfies, i.e., free-viewpoint selfies. Deformable NeRF conditions deformations and appearance with an auto-decoded latent code per input view. The bent rays are regularized using an as-rigid-as-possible term (also known as elastic energy term) that penalizes deviations from piece-wise rigid scene configurations. Thus, Deformable NeRF works well on articulated scenes (e.g., a hand holding a tennis racket) and scenes such as human heads (where the head is moving w.r.t. the torso). Still, small non-rigid deformations are handled well (such as smiling), as the regularizers are soft. Another important innovation of this work is using a coarse-to-fine scheme which allows learning low-frequency components first and avoiding local minima due to overfitting to high-frequency details.

HyperNeRF [PSH*21] is an extension of Deformable NeRF [PSB*21] using a canonical hyperspace instead of a single canonical frame. This allows tackling scenes with topological changes such as opening and closing the mouth. In HyperNeRF, the bending network (MLP) of Deformable NeRF is augmented with an ambient slicing surface network (likewise an MLP) that selects a canonical subspace for every input RGB view by indirectly conditioning the canonical scene on the deformation. As such it is a hybrid that combines both explicit and implicit deformation modelling, which allows it to handle topological changes by sacrificing hard correspondences.

Non-rigid NeRF (NR-NeRF) [TTG*21] models a time-varying scene appearance using a per-scene canonical volume, per-scene rigidity flag (an MLP) and per-frame ray bending operator (an MLP). NR-NeRF shows that no additional supervisory cues such as depth maps or scene flows are required to handle scenes with small non-rigid deformations and motions, in contrast to [PSB*21, XHKK21, LNSW21]. Moreover, the observed deformations are regularized by a divergence operator, which imposes a volume-preserving constraint and stabilizes occluded areas with respect to supervising monocular input views. In this regard, it has similarities with the elastic regularizer of Nerfies penalizing deviations from piece-wise rigid deformations. This regularization makes it possible for the camera trajectory of novel views to differ significantly from the input camera trajectory. While controllability is still severely limited, NR-NeRF demonstrates several simple edits of the learned deformation field, such as motion exaggeration or removal of dynamic scene content.

Other works do not restrict themselves to the case of monocular RGB input video, but instead consider other inputs.

Time-of-Flight Radiance Fields (TöRF) method [ALG*21] replaces data-driven priors for reconstructing dynamic contents with depth maps from a depth sensor. In contrast to the vast majority of computer vision works, TöRF uses raw ToF sensor measurements (so-called phasors), which brings advantages when handling weakly-reflecting regions and other limitations of modern depth sensors (e.g., restricted working depth range). Integration of measured scene depths in the learning of NeRF reduces the requirement on the number of input views leading to sharp and detailed models.

Figure 10: Dynamic Neural Radiance Field to synthesize novel views and expressions of humans. Image adapted from [GTZN21].

The depth cue also enables superior accuracy compared to NSFF [LNSW21] and space-time neural irradiance fields [XHKK21].

Neural 3D Video Synthesis [LSZ*21] uses a multi-view RGB setup and models deformations implicitly. The method exploits temporal smoothness by first training on keyframes. It also exploits that the cameras remain static and that the scene content is predominantly static by sampling rays in a biased manner for training. The results are sharp even for dynamic content that is small.

4.3.2. Controllable Dynamic Neural Radiance Fields

To allow controllability of the deformation of the neural radiance field, method use class specific motion models as underlying representation of the deformation state (e.g., a morphable model for the human face or a skeletal deformation graph for the human body).

NeRFace [GTZN21] is the first approach that uses a morphable model to implicitly control a neural radiance field (see Figure 10). They use a face tracker [TZS*16] to reconstruct the face blendshape parameters as well as the camera pose in the training views (monocular video). The MLP is trained on these views with the blendshape parameters and a learnable per-frame latent codes as conditioning. In addition, they assume a known static background such that the radiance field only stores the information about the face. The latent codes are used to compensate missing tracking information (i.e., the shoulders of a person) as well as errors in the tracking). Once trained the radiance field can be controlled via the blendshape parameters, thus, allowing reenactment and expression editing. While NeRFace uses a global deformation code based on a morphable model, Wang et al. [WBL*20] generate local animation codes. Specifically, they extract a global animation code from multi-view inputs which is mapped to local codes using 3D convolutional neural network. These are used to condition the fine-level radiance field which are represented as MLPs. In contrast to NeRFace, the method does not allow direct control over expressions of the face, but an encoder has to be trained that for example can generate the animation codes from facial keypoints. Guo et al. [GCL*21] propose an audio driven neural radiance field (AD-NeRF) which is inspired by NeRFace. But instead of using expression coefficients, they map audio features extracted using DeepSpeech [HCC*14, TET*20] to a feature which serves as a conditioning to the MLP that represents the radiance field. While the expression is controlled implicitly via an audio signal, they provide explicit control over the rigid pose of the head. To synthesize the portrait view of a person, they employ two separate radiance fields, one for the head and one for the torso.

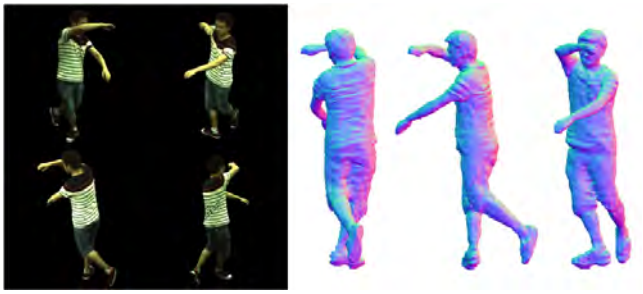


Figure 11: Neural Body [PZX*21] approach recovers 3D models of humans with fine appearance details from sparse multi-view video.

While the afore mentioned approaches show promising results in a portrait scenario, they are not applicable to highly non-rigid deformations, especially, for articulated motion of a human body captured from a single view. Therefore, methods leverage the human skeleton embedding explicitly. Neural Articulated Radiance Field (NARF) [NSLH21] is trained via pose-annotated images. An articulated object is decomposed into several rigid object parts with their local coordinate systems and global shape variations on top. The converged NARF can be used to render novel views by manipulating the poses, estimate depth maps and perform body parts segmentation. In contrast to NARF, A-NeRF [SYZR21] learns actor-specific volumetric neural body models from monocular footage in a self-supervised manner. The method combines a dynamic NeRF volume with the explicit controllability of an articulated human skeleton embedding and reconstructs both the pose and radiance field in an analysis-by-synthesis way. Once trained, the radiance field can be used for novel view point synthesis as well as motion retargeting. On the Human3.6M dataset [IPOS14], they show the benefits of using a learned surface-free model which improves the accuracy of human pose estimation from monocular videos with the help of a photometric reconstruction loss. While A-NeRF is trained on monocular videos, Animatable Neural Radiance Fields (ANRF) [PDW*21] is a skeleton-driven approach for human model reconstruction from multi-view videos. Its core component is a new motion representation, i.e., the neural blend weight field, that is combined with 3D human skeletons for deformation field generation. Similarly to several general non-rigid NeRFs [PSB*21, TTG*21], ANRF maintains a canonical space and estimates two-way correspondences between the multi-view inputs and the canonical frame. The reconstructed animatable human models can be used for free-viewpoint rendering and re-rendering under novel poses. Human meshes can also be extracted from ANRF by running marching cubes on volume densities at the discretized canonical space points. The method achieves high visual accuracy for the learned human models, and the authors suggest that handling complex non-rigid deformations on the observed surfaces (such as those due to loose clothes) can be improved in future work.

The Neural Body approach of Peng and colleagues [PZX*21] enables novel view synthesis of human performances from sparse multi-view videos (e.g., only four synchronized views), see Figure 11 for exemplary inputs and the result. Their method uses conditioning by the parametric human shape model SMPL [LMR*15] as a shape proxy prior. It assumes that the recovered neural representation at different frames has the same set of latent codes anchored to a deformable mesh. General-purpose baselines such as rigid NeRF [MST*20] (applied per timestamp) or NeuralVolumes [LSS*19] assume much denser input image sets and, consequently, cannot compete with Neural Body in its ability to render novel views of moving humans from a few synchronized input images. The method also favourably compares to human mesh reconstruction techniques such as PIFuHD [SSSJ20], which strongly depends on training 3D data when it comes to the 3D reconstruction of fine appearance details (e.g., rarely-worn or unique garments). Similar to the Neural Body approach, Neural Actor (NA) [LHR*21] uses the SMPL model to represent the deformation states. They leverage the proxy to explicitly unwarp the surrounding 3D space into a canonical pose, where the NeRF is embedded. To improve the recovery of high fidelity details in geometry and appearance, they use additional 2D texture maps defined on the SMPL surface, which are used as an additional conditioning to the NeRF MLP. H-NeRF [XAS21] is another technique for temporal 3D reconstructions of humans with conditioning using an human body model. Similarly to Neural Body [PZX*21], they require a sparse set of videos from synchronized and calibrated cameras. In contrast to it, H-NeRF uses a structured implicit body model with signed distance fields [AXS21], which results in sharper renderings and more complete geometry for challenging subjects.

Mixture of Volumetric Primitives [LSS*21] is a model for rendering dynamic, animatable virtual humans in real time. The main idea is to model a scene or object with a set of volumetric primitives that can dynamically change position and content. These primitives model components of the scene like a parts-based model. Each volumetric primitive is a voxel grid produced by a decoder network from a latent code. The code defines the configuration of the scene (e.g., a facial expression, in the case of human faces) which is used by the decoder network to produce primitive locations and voxel values (which contain RGB color and opacity). To render, a ray-marching procedure is used to accumulate color and opacity values along the rays corresponding to each pixel. Similar to other dynamic NeRF methods, multi-view video is used as training data. The method is capable of creating extremely high-quality realtime renderings that look realistic even on challenging materials, like hair and clothing.

4.4. Compositionality and Editing

The methods discussed so far allow reconstructing volumetric representations of static or dynamic scenes and render novel views of them, perhaps from a few input images. They keep the observed scene unchanged, except for comparably straightforward modifications (e.g., foreground removal). Several recent methods also allow editing the reconstructed 3D scenes, i.e., rearranging and affine-transforming the objects and altering their structure and appearance. In Table 4, we give an overview of the discussed methods.

Conditional NeRF [LZZ*21] can alter the color and shape of rigid objects observed in 2D images from manual user edits (e.g., it is possible to remove some object parts). This functionality is enabled by a single NeRF trained on multiple object instances of the same category. During editing, the network parameters are adjusted

Method	Required Data	3D Representation	Controllable Parameters	Generative Model	Code
Nguyen-Phuoc et al. [NPLT*19]	MVI+UIC	V	P,S,T	✓	
Liu et al. [LZZ*21]	MVI	V	S,C	✗	
Jang and Agapito [JA21]	UIC	V	P,S,T	✗	
Ost et al. [OMT*21]	VID	V-O	P,S,T,O	✓	
Zhang et al. [JXX*21]	MV-VID	V-O	S,C	✓	
Niemeyer and Geiger [NG21b]	UIC	NFF	P,S,T	✓	

Table 4: Selected methods for editing and scene compositionality presented in Section 4.4. VID: Monocular videos, MV-VID: Synchronised multi-view videos, MVI: Multi-view image collections, UIC: Unstructured 2D image collections. V: Neural volumetric, V-O: Per-object+background neural volumetric, NFF: Neural feature fields. P: Camera pose, S: Shape, T: Texture/appearance, O: Opacity, C: Color.

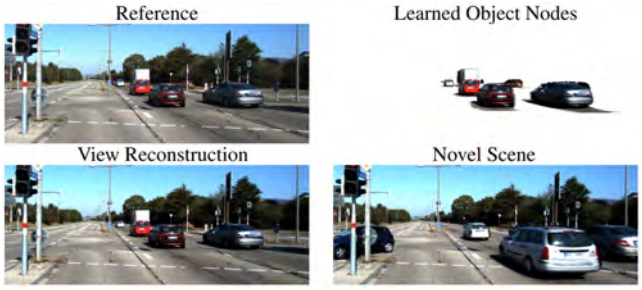


Figure 12: A reconstructed view, learned object nodes and novel scene renderings by the Neural Scene Graphs approach [OMT*21].

to match the shape and color of a newly observed instance. One of the contributions of this work is finding a subset of tunable parameters which can successfully propagate user edits for novel view generation. This avoids expensive modifications of the entire network. CodeNeRF [JA21] represents shape and texture variations across an object class. Similar to pixelNeRF, CodeNeRF can synthesize novel views of unseen objects. It learns two different embeddings for the shape and texture. At test time, it estimates a camera pose, 3D shape and texture of the object from a single image, and both can be continuously modified by altering their latent codes. CodeNeRF achieves comparable performance to previous methods for single-image 3D reconstruction, while not assuming known camera poses.

Neural Scene Graphs (NSG) [OMT*21] is a recent method for novel view synthesis from monocular videos recorded while driving (ego-vehicle views). This technique decomposes a dynamic scene with multiple independent rigidly moving objects into a learned scene graph that encodes individual object transformations and radiances. Thus, each object and the background are encoded by different neural networks. In addition, the sampling of the static node is restricted to layered planes (which are parallel to the image plane) for increased efficiency, i.e., a 2.5D representation. NSG requires annotated tracking data for each rigidly moving object of interest over the set of input frames, and each object class (e.g., a car or bus) shares a single volumetric prior. The neural scene graph can then be used to render novel views of the same (i.e., observed) or edited (i.e., by rearranging the objects) scene. Applications of NSG include background-foreground decomposition, enriching training datasets for automotive perception, and improved object detection and scene understanding (see Figure 12).

Another layered representation for editable free-viewpoint videos is introduced in Zhang *et al.* [JXX*21]. Their spatially and temporally consistent NeRF (ST-NeRF) relies on bounding boxes for all independently moving and articulated objects—resulting in multiple layers—and disentangles their positions, deformations and appearance. The input to ST-NeRF is a set of 16 synchronized videos from the cameras placed at regular intervals in a half-circle, along with human-background segmentation masks. The method's name suggests that space-time coherence constraints are reflected in its architecture, i.e., as a space-time deformation module and a NeRF module of the canonical space. ST-NeRF also accepts timestamps to account for the appearance evolving in time. While rendering novel views, the sampling rays are cast through multiple scene layers, which results in accumulated densities and colors. ST-NeRF can be used for neural scene editing such as rescaling, shifting, duplication or removing of the performers, and temporal rearrangements. As promising directions for future work, the authors name reducing the number of input views and enabling non-rigid scene editing.

Note that some of the methods [NG21b, NPLT*19] discussed in Section 4.2 can be used for scene editing as well. E.g, GIFARRE can rotate an object of a known class observed in a single monocular image, change its appearance and translate it along the depth channel. See Table 4 for a comparison of the methods discussed in this section.

4.5. Relighting and Material Editing

The applications we have presented so far are based on the simplified absorption-emission volumetric rendering model discussed in Section 3.2.2, in which the scene is modeled as a volume of particles that block and emit light. While this model is sufficient for rendering images of the scene from novel viewpoints, it is unable to render images of the scene under different lighting conditions. Enabling relighting requires a scene representation that can simulate the transport of light through the volume, including the scattering of light by particles with various material properties. In Table 5, we give an overview over the discussed methods.

Neural Reflectance Fields [BXS*20] proposed the first extension of NeRF to enable relighting. Instead of representing a scene as a field of volume density and view-dependent emitted radiance, as in NeRF, Neural Reflectance Fields represent a scene as a field of volume density, surface normals, and bi-directional reflectance distribution functions (BRDFs). This allows for rendering the scene under arbitrary lighting conditions by using the predicted surface normals and BRDFs at each 3D location to evaluate how much incoming light is reflected off particles at that location towards the

Method	Required Data	3D Representation	Controllable Parameters	Models Light Visibility	Models Indirect Illumination	Code
Bi et al. [BXS*20]	I+L	V	L+M	✓	✗	✗
Zhang et al. [ZLW*21]	I+M	S	L+M	✗	✗	↗
Boss et al. [BBJ*21]	I+M	V	L+M	✗	✗	↗
Srinivasan et al. [SDZ*21]	I+L	V	L+M	✓	✓	✗
Zhang et al. [ZSD*21]	I	V	L+M	✓	✗	↗
Xiang et al. [XXH*21]	I+M	V	T	N/A	N/A	✗

Table 5: Selected methods for relighting presented in Section 4.5. I: Images, L: Lighting parameters for input images, M: Object masks. V: Neural volumetric, S: Neural SDF. L: Lighting, M: Materials, T: Texture map.

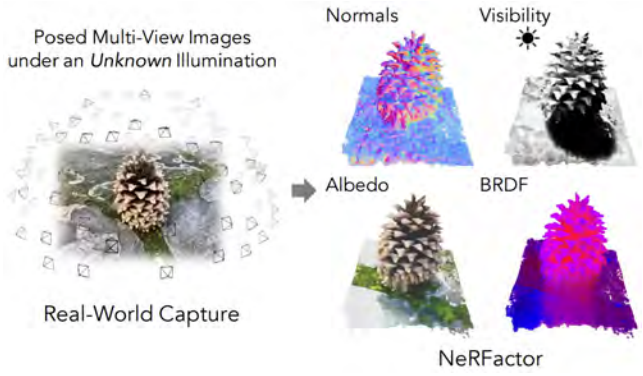


Figure 13: NeRFactor [ZSD*21] decomposes a scene captured under an unknown illumination into 3D neural fields of surface normals, albedo, BRDF and shading. This enables free-viewpoint relighting and material editing. Image adapted from [ZSD*21].

camera. However, evaluating the visibility from each point along the camera ray to each light source is extremely computationally intensive for neural volumetric rendering models. Even when just considering direct lighting, the MLP must be evaluated at densely-sampled locations between each point along the camera ray and each light source in order to compute the incident lighting to render that ray. Neural Reflectance Fields sidesteps this issue by only training with images of objects illuminated by a single point light that is co-located with the camera, so the MLP only needs to be evaluated along the camera ray.

Other recent works that recover relightable models have avoided the difficulty of computing light source visibility by simply ignoring self-occlusions and assuming that all light sources in the upper hemisphere above any surface are fully visible. Both PhySG [ZLW*21] and NeRD [BBJ*21] assume full light source visibility, and further accelerate rendering by representing the environment lighting and scene BRDFs as mixtures of spherical Gaussians, which enables the hemispherical integral of the incoming light multiplied by the BRDF to be computed in closed form. Assuming full light source visibility can work well for objects that are mostly convex, but this strategy is unable to simulate effects such as cast shadows that are due to the occlusion of light sources by scene geometry.

Neural Reflectance and Visibility Fields [SDZ*21] (NeRV) trains an MLP to approximate the light source visibility for any input 3D location and 2D incoming light direction. Instead of querying an MLP at densely-sampled points along each light ray, the visiblity MLP only needs to be queried a single time for each incoming light direction. This enables NeRV to recover relightable models of scenes from images with significant shadows and self-occlusion effects.

Instead of optimizing a relightable representation from scratch, as done in the previously discussed methods, NeRFactor [ZSD*21] starts with a pre-trained NeRF model. NeRFactor then recovers a relightable model by simplifying the pre-trained NeRF's volumetric geometry into a surface model, optimizing MLPs to represent the light source visibility and surface normals at any point on the surface, and finally optimizing a representation of the environment lighting and the BRDF at any surface point; see Figure 13 for an example decomposition. This results in a relightable model that is more efficient when rendering images, since the volumetric geometry has been simplified into a single surface and light-source visibility at any point can be computed by a single MLP query.

The relightable models described above represent scene materials as a continuous 3D field of BRDFs. This enables some basic amount of material editing since the recovered BRDFs can be changed before rendering. NeuTex [XXH*21] enables more intuitive material editing by introducing a surface parameterization network that learns a mapping from 3D coordinates in the volume to 2D texture coordinates. After a NeuTex model of a scene is recovered, the 2D texture can easily be edited or replaced.

4.6. Engineering Frameworks

For practitioners, working with neural rendering models poses engineering challenges: large amounts of image and video data must be processed in a highly non-sequential manner, and the models often require differentiation of large and complex computational graphs. In this section, we will discuss recent advances in tools that can help to overcome these problems.

4.6.1. Storage

Saturating a GPU with data in particular for neural rendering is challenging: often, each pixel of images or videos is treated as a separate data point. Methods require random iteration over the entire pool of pixels in the dataset, in case of temporal reconstruction often across the entire sequence for a single batch. Flexible storage solutions should take this into account.

NVIDIA AIStore [AMB19] is a general purpose storage solution that allows to monitor throughput per drive and implements tiered

architectures for loading and shuffling, while abstracting these layers away from the user. Independent of the storage backend, sharding has tremendous benefits through 1) allowing to shuffle data in memory while 2) using mostly sequential reads within the shards. Tensorflow [AAB*15] has built-in support sharded storage through the `tfrecord` file format, whereas `webdataset` offers similar convenient features for PyTorch [PGM*19].

4.6.2. Hyperparameter Search and Experiments

With long runtimes and complex configuration hierarchies, neural rendering experiments require good techniques for experiment management and hyperparameter search. Hydra [Yad19] excels at configuring even the most complex experiments and offers integrated support for hyperparameter search, for example using the AX adaptive experimentation framework. However, running all experiments for a sweep until convergence, even for smartly picked parameters using Bayesian hyperparameter search, might be too time consuming. Ray tune [LLN*18] has implementations of algorithms like ASHA [LJR*20] and Hyperband [LJRT18], which can dynamically assign computational and time budgets to experiments for a faster hyperparameter search.

4.6.3. Differentiable Rendering and Autodiff

Neural Rendering has high demands towards differentiability: complex computational graphs need to be built and, depending on the application, be executed either on large inputs vectorized (macro AD—for brevity we refer to auto-differentiation as AD throughout this section) or on large amounts of small inputs (micro AD). Depending on the application, the AD package might have to be used low level (e.g., in CUDA), or high level (e.g., in Python). A powerful AD library for C++ is STAN [CHB*15]. We refer to the accompanying paper for a comprehensive overview and evaluation of AD libraries until its publication in 2015, which is beyond the scope of this article. A noteworthy more recent AD package for C++17 is the `autodiff` package. Enzyme AD [MC] is taking a particularly versatile approach for low-level AD: it leverages the LLVM ecosystem as a whole. This is particularly powerful, because of the concept of frontends, the LLVM IR and backends. In broad strokes, LLVM frontends translate a language, for example C++, to the LLVM *intermediate representation* (IR). This representation is an abstract, language-agnostic representation of low-level commands, and it is the same for all frontends. This is where Enzyme comes in: it is an extension that can create derivatives of functions in this IR. That means that it works for all languages that LLVM supports. LLVM backends emit code from the IR: this could be for x86, ARM or GPU processors. This means, that Enzyme supports a variety of processors, including GPUs. Another C++ package specifically for processing images and graphics is Halide [LGA*18]. Its standout feature is flexible scheduling for parallel processing of pixels.

Difftaichi [HAL*20] offers differentiable programming in Python for physical simulation with applications in rendering. Enoki [Jak19] is a very versatile and high performance AD component for physically-based differentiable rendering and is the core component of the Mitsuba 2 renderer [NDVZJ19]. Jax [BFH*18] is a Python framework for differentiable and accelerated linear algebra with compilation options for GPUs and TPUs. JaxNeRF is a reference implementation for NeRF using Jax. The Swift programming language provides AD as a first class use case, and was heavily used for developing a Tensorflow integration.

4.6.4. Raycasting and Rendering

Several packages exist for providing high-level rendering and aggregation primitives. NVIDIA OptiX is a high performance library for ray-casting and ray-intersection and provides to date the only possibility to use the hardware acceleration on NVIDIA RTX hardware for ray intersection. Teg [BMM*21] is a differentiable programming language which provides primitives for optimizing integrals with discontinuous integrands, as frequently found in rendering. Redner [LADL18b] is a framework for differentiable ray tracing; Mitsuba 2 [NDVZJ19] provides an even more general framework for physically based differentiable rendering and path tracing. psdr-cuda [LZBD21] improves over Redner by using better gradient calculation techniques and sampling strategies. PyTorch3D [RRN*20] offers a broad suite of tools around differentiable rendering and graphics, tightly integrated with PyTorch. Tensorflow Graphics [VKP*19] has a similar goal for Tensorflow.

5. Open Challenges

After covering a wide variety of computer graphics and vision problems to which neural volumetric representations can be successfully applied, we now take a look at problems where only classical representations have been used. Thus, there are various avenues for future research. We further discuss multiple open challenges in the field. Many of the points discussed in the following are related to each other.

Seamless Integration and Usage. Most computer graphics algorithms and techniques developed over more than half a century assume meshes or point clouds as 3D scene representations for rendering and editing. In contrast, neural rendering is such a young field that this notion was used for the first time just a few years ago in 2018 [ERB*18]. Thus, inevitably, there is still a gap between the spectrum of available methods that can operate on classical 3D representations and those that are applicable to neural representations. Furthermore, many methods exist to edit classical representations, e.g., widely-used tools such as Blender [Com18] and Maya [Aut] support meshes and texture maps, whereas their counterparts for neural representations have to be developed from scratch. On the other hand, it is foreseeable that this gap will decrease with further improvement in the field and more and more widespread adoption and integration of neural representations. Moreover, modern hardware accelerators are designed for classical computer graphics and could in the future be similarly tailored to neural representations.

Another related challenge is interpretability of the learned representations, which concerns deep learning in general. Thus, learned neural network weights are notoriously hard to interpret in terms of the target quantities (e.g., point colors and opacities in the 3D space). At the same time, they aim to replace the graphics pipeline, which is well understood and relies on analytically derived steps.

Figure 14: 3D reconstruction for a room with keyframes (shown in red) obtained by iMAP [SLOD21]. iMAP is a real-time SLAM system for a single handheld RGB-D camera that can efficiently fill in occluded regions. Despite the first successful steps, learning neural representations for large-scale scenes has still many open challenges. Image adapted from [SLOD21].

Ultimately, to improve controllability and enable seamless integration of learned volumetric models in computer graphics tools, we would like to be able to modify the scene parametrization to change the scene in a desired direction. While this is likely not tractable for arbitrary scenes parametrized by global MLPs, composing a full scene out of local neural representations might make it tractable by opening up the intriguing possibility of re-introducing aspects of classical graphics.

Scalability. Most of the works on volumetric neural rendering focus on single objects and relatively simple composite scenes (e.g., a human and a background, several humans in the same environment, a street with moving cars) with or without background. Learning neural representations for large-scale scenes—which can only be partially observed in each input frame—is still challenging. Although the first successful and impressive steps in this direction have been made (we refer here to Nerf in the Wild [MBRS*21] and the neural SLAM system iMAP [SLOD21], see Figure 14), many open challenges remain. For instance, the approaches for scene editing, relighting, and compositionality developed for single objects cannot be straightforwardly extended to handle large-scale scenes. Moreover, a global representation for large-scale environments becomes unfeasible starting from some scene size, even when applying space partitioning policies such as those used in PlenOctrees [YLT*21]. Thus, a new generation of storage and retrieval techniques need to be developed for efficient neural models for large-scale scenes, along the lines of VoxelHashing [NZIS13] for TSDFs. First, they should make the scene completion more efficient (i.e., without the need to constantly recompute the entire model from scratch) and, second, enable easy retrieval of partial contents. Both these points are related to the open challenge of interpretability discussed above.

Generalizability. Only a few initial but promising methods exist for generalizable and instantiable volumetric neural representations. For example, StereoNeRF [CBLPM21] uses only a dozen spread-out views to generate novel views of a rigid scene with the visual accuracy comparable to the original NeRF [MST*20] after fine-tuning, while pixelNeRF [YYTK21] can infer volumetric models of rigid scenes unseen at training time just from a single image. This class of approaches is data-driven and requires large-scale multi-view datasets with a sufficiently wide baseline. Consequently, these methods can produce views at arbitrary novel viewpoints if the datasets provide sufficient viewpoint coverage. Reducing this strong dependency is an exciting direction for future work. Another open challenge is the generalizability of instantiable approaches to scenes with non-rigid deformations. The inputs can be sparse sets of spatiotemporal observations or even single images at the extreme (in this case, the task becomes scene animation from a single image). One straightforward direction towards such techniques would be relying on multi-view datasets of deformable scenes, which would likely increase required dataset sizes by a multitude. Another possible way would be to disentangle deformation modes and scene shapes and appearances at rest. Furthermore, while there exists some work on generating neural scene representations (e.g., using hypernets [SRF*21]), there is less progress on designing neural operators that take neural scene representations as input to work on them, for example to complete a partial scene or to regress semantic labels for an existing representation. No operator analogous to mesh convolutions for meshes or 3D convolutions for voxel grids exists. Such an operator would ideally be trained only once and then be generally applicable.

Multi-Modal Learning. Multi-modal learning means going beyond visual signals and incorporating other data types such as semantics, textual descriptions and sound. For example, telepresence and augmented reality would highly benefit from a method that can not only render novel views of dynamically interacting and talking humans but also synthesize the corresponding novel sounds; existing work can, for instance, synthesize stereo audio from mono audio inputs [RMG*21]. Synthesizing textual descriptions and semantics of the scene (e.g., semantic segmentation labels) can be very useful for downstream applications based on volumetric neural representations. While some prior work addresses this goal [KSW20,ZLLD21], this remains an open challenge. More detailed and scenario-specific modeling could take into account such information as the camera capture system (e.g., as already shown in TöRF [ALG*21] for depth cameras), whether the camera is using rolling or global shutter, or if there is motion blur in the input images. Other sensors like IMUs, Lidar, or event streams could all potentially be modeled in a continuous fashion. (Ultrasound and x-rays could be continuously modeled at arbitrary resolution for medical imaging.) It is also conceivable to optimize for certain capture properties that are not trivial to measure like color calibrations of a multi-view capture setup. This extends to physical simulation in general, where neural scene representations offer an exciting venue to “learn less and know more" by incorporating differentiable physics simulators; e.g., for physically motivated deformation models or physically correct light transport.

Other Questions. Can we increase quality? Reconstructing objects with many high-frequency details, shading, and view-dependent appearance remains a largely unsolved problem. Can we decrease training time? Although there has been progress on very fast inference for novel view synthesis at test time, improving the training time remains a big challenge. Are fewer input images sufficient?

Fewer input views might be sufficient to reach a similar visual fidelity as fully converged models requiring hundreds of views. Currently, partial observations (e.g., parts of the scene observed only in a subset of images) tend to be blurrier than the rest of the scene.

Beyond the immediate use case of AR/VR, there is little research on using neural scene representations in other contexts like robotics, with the notable exception of the real-time SLAM system iMAP [SLOD21] (see Figure 14). How can we obtain, incorporate, and predict object affordances or other annotations like temperature? Are there advantages to using neural scene representations for motion prediction or planning?

The list of future directions discussed in this section does not aim for completeness. We expect to see many improvements on more aspects of coordinate-based neural volumetric representations already in the near future.

6. Social Implications

Neural approaches discussed in this state-of-the-art report achieve a very high degree of realism for synthesized novel views. Rapid developments in the field already influence and will continue influencing society in many positive and potentially negative ways which we discuss in this section.

Research and Industry. The fields which are starkly impacted by the new volumetric neural representations are computer vision, computer graphics as well as augmented and virtual reality, which can benefit from increased photo-realism of rendered environments. The fact that the state-of-the-art volumetric models rely on well-understood and elegant principles lowers the barrier to entry for research on photogrammetry and 3D reconstruction. Moreover, this effect is magnified by the ease of use of the methods and publicly available codebases and datasets. Since neural rendering is still not mature and well understood, end-user tools like Blender do not yet exist, putting these novel methods out of reach for both 3D hobbyists and industry as of now. However, more widespread understanding of the technology inevitably impacts developed products and applications. With that, we foresee decreased effort in content creation for games and special effects for movies. The possibility to render photo-realistic novel views of a scene from a few input images is a significant advantage compared to existing technology. This can potentially reshape the entire established pipeline for content design used in the visual effects (VFX) industry.

Trustworthiness. However, at the same time, photo-realism creates the possibility to misuse the technology and create synthetic content that malicious actors may falsely claim to be real, in particular, when neural rendering approaches focus on human faces [TZS*16, TZN19, GTZN21]. In response to these potential misuses, methods to automatically detect such fake content are being developed by the research community [CRT*21, RCV*19] and security measures including encryption and block chain measures are being explored. And there are a number of other mitigations that could be explored to minimize these risks. For example, while there are cases where we expect users not to object to seeing synthetic photo-real content (e.g., when watching movies), synthetic content could be labelled or otherwise identified as such to inform users. Further user studies could investigate people's judgement of the need to label synthetic content in different contexts. On the collection side, people could provide explicit and informed consent that their identity can be used for creating synthetic content in a specified context.

Environment. Since current neural volumetric scene representations are deep-learning-based, the GPUs used for training them consume a sizable amount of energy. Since more and more laboratories are working on neural rendering, the use of high-end and multi-GPU systems increases accordingly. If the resources for manufacturing and the electricity for operating the GPU clusters are not taken predominantly from renewable sources, training volumetric neural representations can negatively influence the environment and global climate in the long term. In an attempt to soften the need for computational resources and hence electricity and hardware, there are many architectures that require less compute power for training than NeRF-based methods. Last but not least, high GPU demand potentially implies that not all groups can afford to contribute on equal footing as experimenting with volumetric representations is not the most lightweight task.

7. Conclusion

In this state-of-the-art report, we have reviewed the recent trends on neural rendering techniques. The methods covered learn 3D neural scene representations based on 2D observations as inputs for training, and enable synthesis of photo-realistic imagery with control over different scene parameters. The field of neural rendering has seen rapid progress during the last few years and continues to grow fast. Its applications range from free-viewpoint videos of rigid and non-rigid scenes to shape and material editing, relighting, and human avatar generation, among many others. These applications have been discussed in detail in this report.

At the same time, we believe that neural rendering is still an emerging field with many open challenges that can be addressed. To this end, we identify and discuss multiple directions for future research. In addition, we discuss social implications, which arise from the democratization of neural rendering along with its capability to synthesize photo-realistic image content. Overall, we conclude that neural rendering is an exciting field, which is inspiring thousands of researchers across many communities to tackle some of computer graphics' hardest problems, and we look forward to seeing further developments on the topic.

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G. S., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., YANGQING J., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/, 2015. 20

[AL20] ATZMON M., LIPMAN Y.: Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 2565–2574. 6

[ALG*20] ATTAL B., LING S., GOKASLAN A., RICHARDT C., TOMPKIN J.: MatryODShka: Real-time 6DoF video view synthesis using multi-sphere images. In *Proc. ECCV* (Aug. 2020). URL: https://visual.cs.brown.edu/matryodshka. 6

[ALG*21] 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. In *Neural Information Processing Systems (NeurIPS)* (2021). 14, 16, 21

[ALKN19] AZINOVIC D., LI T.-M., KAPLANYAN A., NIESSNER M.: Inverse path tracing for joint material and lighting estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 2447–2456. 8

[AMB19] AIZMAN A., MALTBY G., BREUEL T.: High Performance I/O For Large Scale Deep Learning. *IEEE International Conference on Big Data* (2019), 5965–5967. 19

[AMBG*21] AZINOVIC D., MARTIN-BRUALLA R., GOLDMAN D. B., NIESSNER M., THIES J.: Neural rgb-d surface reconstruction. 12

[Ano22] ANONYMOUS: StyleneRF: A style-based 3d aware generator for high-resolution image synthesis. In *Submitted to The Tenth International Conference on Learning Representations* (2022). under review. URL: https://openreview.net/forum?id=iUuzzTMUw9K. 13, 14

[ASK*20a] ALIEV K.-A., SEVASTOPOLSKY A., KOLOS M., ULYANOV D., LEMPITSKY V.: Neural point-based graphics. arXiv:2110.06635. 5

[ASK*20b] 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* (2020), Springer, pp. 696–712. 8

[Aut] AUTODESK, INC.: Maya. URL: https://autodesk.com/maya. 20

[AXS21] ALLDIECK T., XU H., SMINCHISESCU C.: imghum: Implicit generative models of 3d human shape and articulated pose. In *International Conference on Computer Vision (ICCV)* (2021). 17

[BBJ*21] BOSS M., BRAUN R., JAMPANI V., BARRON J. T., LIU C., LENSCH H. P. A.: NeRD: Neural reflectance decomposition from image collections. *ICCV* (2021). 1, 19

[BFH*18] BRADBURY J., FROSTIG R., HAWKINS P., JOHNSON M. J., LEARY C., MACLAURIN D., NECULA G., PASZKE A., VANDERPLAS J., WANDERMAN-MILNE S., ZHANG Q.: JAX: composable transformations of Python+NumPy programs, 2018. URL: http://github.com/google/jax. 20

[BFO*20] BROXTON M., FLYNN J., OVERBECK R., ERICKSON D., HEDMAN P., DUVALL M., DOURGARIAN J., BUSCH J., WHALEN M., DEBEVEC P.: Immersive light field video with a layered mesh representation. *ACM Trans. Graph. (SIGGRAPH) 39*, 4 (2020). 6

[BGP*21] BAATZ H., GRANSKOG J., PAPAS M., ROUSSELLE F., NOVÁK J.: Nerf-tex: Neural reflectance field textures. In *Eurographics Symposium on Rendering* (June 2021), The Eurographics Association. 5

[BKK19] BAI S., KOLTER J. Z., KOLTUN V.: Deep equilibrium models. *NeurIPS* (2019). 6

[BKW21] BERGMAN A. W., KELLNHOFER P., WETZSTEIN G.: Fast training of neural lumigraph representations using meta learning. In *Proceedings of the IEEE International Conference on Neural Information Processing Systems (NeurIPS)* (2021). 8, 14

[BMM*21] BANGARU S., MICHEL J., MU K., BERNSTEIN G., LI T.-M., RAGAN-KELLEY J.: Systematically differentiating parametric discontinuities. *ACM Trans. Graph. 40*, 107 (2021), 107:1–107:17. 20

[BMT*21] 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. *ICCV* (2021). 12

[BNT21] BUROV A., NIESSNER M., THIES J.: Dynamic surface function networks for clothed human bodies. 5

[BXS*20] BI S., XU Z., SRINIVASAN P. P., MILDENHALL B., SUNKAVALLI K., HAŠAN M., HOLD-GEOFFROY Y., KRIEGMAN D., RAMAMOORTHI R.: Neural reflectance fields for appearance acquisition. arXiv:2008.03824. 18, 19

[CBC*01a] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, Association for Computing Machinery, p. 67–76. URL: https://doi.org/10.1145/383259.383266, doi:10.1145/383259.383266. 4

[CBC*01b] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 67–76. 5

[CBLPM21] CHIBANE J., BANSAL A., LAZOVA V., PONS-MOLL G.: Stereo radiance fields (srf): Learning view synthesis from sparse views of novel scenes. In *Computer Vision and Pattern Recognition (CVPR)* (2021). 13, 21

[CHB*15] CARPENTER B., HOFFMAN M. D., BRUBAKER M., LEE D., LI P., BETANCOURT M.: The Stan Math Library: Reverse-Mode Automatic Differentiation in C++. URL: http://arxiv.org/abs/1509.07164, arXiv:1509.07164. 20

[Chu06] CHUMPUSREX: Craniale computertomographie, 2006. URL: https://de.wikipedia.org/wiki/Computertomographie#/media/Datei:Ct-workstation-neck.jpg. 4

[CKS*17] CHAITANYA C. R. A., KAPLANYAN A. S., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZAHRAI D., AILA T.: Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph. 36*, 4 (July 2017), 98:1–98:12. URL: http://doi.acm.org/10.1145/3072959.3073601, doi:10.1145/3072959.3073601. 3

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 303–312. 5

[CLI*20] CHABRA R., LENSSEN J. E., ILG E., SCHMIDT T., STRAUB J., LOVEGROVE S., NEWCOMBE R.: Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision (Proceedings of the European Conference on Computer Vision)* (2020). 6

[CMK*21] CHAN E., MONTEIRO M., KELLNHOFER P., WU J., WETZSTEIN G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR* (2021). 1, 13, 14

[Com18] COMMUNITY B. O.: *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org. 20

[CRBD18] CHEN R. T. Q., RUBANOVA Y., BETTENCOURT J., DUVENAUD D. K.: Neural ordinary differential equations. In *Advances in Neural Information Processing Systems* (2018), vol. 31. 15

[CRT*21] COZZOLINO D., ROSSLER A., THIES J., NIESSNER M., VERDOLIVA L.: Id-reveal: Identity-aware deepfake video detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 15108–15117. 22

[CTZ20] CHEN Z., TAGLIASACCHI A., ZHANG H.: Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 45–54. 6

[CW93] CHEN S. E., WILLIAMS L.: View interpolation for image synthesis. In *SIGGRAPH* (1993), pp. 279–288. 12

[CZ19] CHEN Z., ZHANG H.: Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 5939–5948. 6, 10, 11

[DGY*20] DENG B., GENOVA K., YAZDANI S., BOUAZIZ S., HINTON G., TAGLIASACCHI A.: Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 31–44. 6

[DLZR21] DENG K., LIU A., ZHU J.-Y., RAMANAN D.: Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791* (2021). 11

[DNJ20] DAVIES T., NOWROUZEZAHRAI D., JACOBSON A.: Overfit neural networks as a compact shape representation, 2020. arXiv:2009.09808. 6

[DZW*20] DUAN Y., ZHU H., WANG H., YI L., NEVATIA R., GUIBAS L. J.: Curriculum deepsdf. In *European Conference on Computer Vision* (2020), Springer, pp. 51–67. 6

[DZY*21] DU Y., ZHANG Y., YU H.-X., TENENBAUM J. B., WU J.: Neural radiance flow for 4d view synthesis and video processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021). 14, 15

[EGO*20] ERLER P., GUERRERO P., OHRHALLINGER S., MITRA N. J., WIMMER M.: Points2surf learning implicit surfaces from point clouds. In *Proceedings of the European Conference on Computer Vision* (2020), Springer, pp. 108–124. 6

[ERB*18] ESLAMI S. A., REZENDE D. J., 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 (2018), 1204–1210. 20

[FBD*19] FLYNN J., BROXTON M., DEBEVEC P., DUVALL M., FYFFE G., OVERBECK R., SNAVELY N., TUCKER R.: Deepview: View synthesis with learned gradient descent. In *Proc. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 2367–2376. 6, 10

[FNPS16] FLYNN J., NEULANDER I., PHILBIN J., SNAVELY N.: Deep stereo: Learning to predict new views from the world's imagery. In *Proc. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2016). 10

[GCL*21] GUO Y., CHEN K., LIANG S., LIU Y., BAO H., ZHANG J.: Ad-nerf: Audio driven neural radiance fields for talking head synthesis. In *IEEE/CVF International Conference on Computer Vision (ICCV)* (2021). 14, 16

[GCS*20] GENOVA K., COLE F., SUD A., SARNA A., FUNKHOUSER T.: Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 4857–4866. 6

[GCV*19] GENOVA K., COLE F., VLASIC D., SARNA A., FREEMAN W. T., FUNKHOUSER T.: Learning shape templates with structured implicit functions. In *Proceedings of the International Conference on Computer Vision* (2019), pp. 7154–7164. 6

[GKJ*21] GARBIN S. J., KOWALSKI M., JOHNSON M., SHOTTON J., VALENTIN J.: Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380* (2021). 9, 10, 11

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. In *Advances in Neural Information Processing Systems* (2014), Ghahramani Z., Welling M., Cortes C., Lawrence N., Weinberger K. Q., (Eds.), vol. 27, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf. 9

[GSHG98] GREGER G., SHIRLEY P., HUBBARD P. M., GREENBERG D. P.: The irradiance volume. *IEEE Computer Graphics and Applications 18*, 2 (1998), 32–43. 4, 6

[GSKH21] GAO C., SARAF A., KOPF J., HUANG J.-B.: Dynamic view synthesis from dynamic monocular video. *arXiv preprint arXiv:2105.06468* (2021). 14, 15

[GSL*20] 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). 13, 14

[GTZN21] GAFNI G., THIES J., ZOLLHÖFER 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 (CVPR)* (June 2021), pp. 8649–8658. 14, 16, 22

[GYH*20] GROPP A., YARIV L., HAIM N., ATZMON M., LIPMAN Y.: Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099* (2020). 6

[HAL*20] HU Y., ANDERSON L., LI T.-M., SUN Q., CARR N., RAGAN-KELLEY J., DURAND F.: Difftaichi: Differentiable programming for physical simulation. *ICLR* (2020). 8, 20

[Har96] HART J. C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer 12*, 10 (1996), 527–545. 8

[HCC*14] HANNUN A., CASE C., CASPER J., CATANZARO B., DIAMOS G., ELSEN E., PRENGER R., SATHEESH S., SENGUPTA S., COATES A., Y. NG A.: DeepSpeech: Scaling up end-to-end speech recognition. 16

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *SIGGRAPH* (1992). 5

[HLA*19] HU Y., LI T.-M., ANDERSON L., RAGAN-KELLEY J., DURAND F.: Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 201. 8

[HRRR18] HENZLER P., RASCHE V., ROPINSKI T., RITSCHEL T.: Single-image tomography: 3d volumes from 2d cranial x-rays. In *Eurographics* (2018). 10

[HSM*21] HEDMAN P., SRINIVASAN P. P., MILDENHALL B., BARRON J. T., DEBEVEC P.: Baking neural radiance fields for real-time view synthesis. *arXiv* (2021). 9, 10, 11

[HSW89] HORNIK K., STINCHCOMBE M., WHITE H.: Multilayer feedforward networks are universal approximators. *Neural Networks 2*, 5 (1989), 359–366. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208, doi:https://doi.org/10.1016/0893-6080(89)90020-8. 4

[ID18] INSAFUTDINOV E., DOSOVITSKIY A.: Unsupervised learning of shape and pose with differentiable point clouds. In *Proceedings of the IEEE International Conference on Neural Information Processing Systems (NeurIPS)* (2018), pp. 2802–2812. 8

[IKH*11] IZADI S., KIM D., HILLIGES O., MOLYNEAUX D., NEWCOMBE R., KOHLI P., SHOTTON J., HODGES S., FREEMAN D., DAVISON A., FITZGIBBON A.: Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *UIST '11 Proceedings*

of the 24th annual ACM symposium on User interface software and technology (October 2011), ACM, pp. 559–568. 5

[IPOS14] IONESCU C., PAPAVA D., OLARU V., SMINCHISESCU C.: Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 36*, 7 (2014), 1325–1339. 17

[JA21] JANG W., AGAPITO L.: Codenerf: Disentangled neural radiance fields for object categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2021), pp. 12949–12958. 18

[JAC*21] 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 (ICCV)* (October 2021), pp. 5846–5854. 12

[JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision – ECCV 2016* (Cham, 2016), Leibe B., Matas J., Sebe N., Welling M., (Eds.), Springer International Publishing, pp. 694–711. 9

[Jak19] JAKOB W.: Enoki: structured vectorization and differentiation on modern processor architectures, 2019. https://github.com/mitsuba-renderer/enoki. 20

[Jar08] JAROSZ W.: *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, September 2008. 9

[JDV*14] JENSEN R., DAHL A., VOGIATZIS G., TOLA E., AANÆS H.: Large scale multi-view stereopsis evaluation. In *Computer Vision and Pattern Recognition (CVPR)* (2014). 13

[JJHZ20] JIANG Y., JI D., HAN Z., ZWICKER M.: Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020). 6

[JSM*20] JIANG C. M., SUD A., MAKADIA A., HUANG J., NIESSNER M., FUNKHOUSER T.: Local implicit grid representations for 3d scenes. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition)* (2020). 6

[JXX*21] JIAKAI Z., XINHANG L., XINYI Y., FUQIANG Z., YANSHUN Z., MINYE W., YINGLIANG Z., LAN X., JINGYI Y.: Editable free-viewpoint video using a layered neural representation. In *ACM SIGGRAPH* (2021). 1, 18

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (1986), pp. 143–150. 3

[KB04] KOBBELT L., BOTSCH M.: A survey of point-based techniques in computer graphics. *Computers and Graphics 28*, 6 (2004), 801–814. URL: https://www.sciencedirect.com/science/article/pii/S0097849304001487, doi:https://doi.org/10.1016/j.cag.2004.08.009. 7

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). URL: http://arxiv.org/abs/1412.6980, arXiv:1412.6980. 9

[KBS15] KALANTARI N. K., BAKO S., SEN P.: A Machine Learning Approach for Filtering Monte Carlo Noise. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2015) 34*, 4 (2015). 3

[KHM17] KAR A., HÄNE C., MALIK J.: Learning a multi-view stereo machine. In *NeurIPS* (2017). 10

[KJJ*21] KELLNHOFER P., JEBE L., JONES A., SPICER R., PULLI K., WETZSTEIN G.: Neural lumigraph rendering. In *CVPR* (2021). 6, 8

[KSW20] KOHLI A., SITZMANN V., WETZSTEIN G.: Semantic Implicit Neural Scene Representations with Semi-supervised Training. In *International Conference on 3D Vision (3DV)* (2020). 6, 21

[KSZ*21] 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. URL: http://arxiv.org/abs/2104.00587, arXiv:2104.00587. 13

[KTEM18] KANAZAWA A., TULSIANI S., EFROS A. A., MALIK J.: Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision* (2018), pp. 371–386. 6

[KUH18] KATO H., USHIKU Y., HARADA T.: Neural 3D mesh renderer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 3907–3916. 8

[LADL18a] LI T.-M., AITTALA M., DURAND F., LEHTINEN J.: Differentiable monte carlo ray tracing through edge sampling. In *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)* (2018), ACM, p. 222. 8

[LADL18b] LI T.-M., AITTALA M., DURAND F., LEHTINEN J.: Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 37*, 6 (2018), 222:1–222:11. 20

[LB14] LOPER M. M., BLACK M. J.: Opendr: An approximate differentiable renderer. In *Proceedings of the European Conference on Computer Vision* (2014), Springer, pp. 154–169. 8

[LFS*21] 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 *International Conference on Computer Vision (ICCV)* (2021). 12

[LGA*18] LI T.-M., GHARBI M., ADAMS A., DURAND F., RAGAN-KELLEY J.: Differentiable programming for image processing and deep learning in Halide. *ACM Trans. Graph. (Proc. SIGGRAPH) 37*, 4 (2018), 139:1–139:13. 20

[LGL*20] LIU L., GU J., LIN K. Z., CHUA T.-S., THEOBALT C.: Neural sparse voxel fields. *Proceedings of the IEEE International Conference on Neural Information Processing Systems (NeurIPS)* (2020). 6, 9, 10, 11

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, Association for Computing Machinery, p. 31–42. URL: https://doi.org/10.1145/237170.237199, doi:10.1145/237170.237199. 9

[LHL*21] LYU L., HABERMANN M., LIU L., TEWARI A., THEOBALT C., ET AL.: Efficient and differentiable shadow computation for inverse problems. *arXiv preprint arXiv:2104.00359* (2021). 8

[LHR*21] 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 Trans. Graph.(ACM SIGGRAPH Asia)* (2021). 14, 17

[LJR*20] LI L., JAMIESON K., ROSTAMIZADEH A., GONINA E., BEN-TZUR J., HARDT M., RECHT B., TALWALKAR A.: A SYSTEM FOR MASSIVELY PARALLEL HYPERPARAMETER TUNING. *MLSys 2* (2020). arXiv:1810.05934v5. 20

[LJRT18] LI L., JAMIESON K., ROSTAMIZADEH A., TALWALKAR A.: Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research 18* (2018), 1–52. URL: http://jmlr.org/papers/v18/16-558.html., arXiv:1603.06560v4. 20

[LK10] LAINE S., KARRAS T.: Efficient sparse voxel octrees–analysis, extensions, and implementation. *NVIDIA Corporation 2* (2010). 6

[LKL18] LIN C.-H., KONG C., LUCEY S.: Learning efficient point cloud generation for dense 3d object reconstruction. In *AAAI Conference on Artificial Intelligence* (2018). 8

[LLCL19] LIU S., LI T., CHEN W., LI H.: Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *Proceedings of the International Conference on Computer Vision* (2019), pp. 7708–7717. 7, 8

[LLN*18] LIAW R., LIANG E., NISHIHARA R., MORITZ P., GONZALEZ J. E., STOICA I.: Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118* (2018). 20

[LMR*15] LOPER M., MAHMOOD N., ROMERO J., PONS-MOLL G., BLACK M. J.: SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 34*, 6 (2015), 248:1–248:16. 17

[LMTL21] LIN C.-H., MA W.-C., TORRALBA A., LUCEY S.: Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)* (2021). 12

[LMW21] LINDELL D. B., MARTEL J. N., WETZSTEIN G.: Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021). 9, 10, 11

[LNSW21] 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* (2021), pp. 6498–6508. 14, 15, 16

[LSCL19] LIU S., SAITO S., CHEN W., LI H.: Learning to infer implicit surfaces without supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 8295–8306. 8

[LSS*19] LOMBARDI S., SIMON T., SARAGIH J., SCHWARTZ G., LEHRMANN A., SHEIKH Y.: Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph. 38*, 4 (July 2019), 65:1–65:14. 10, 17

[LSS*21] LOMBARDI S., SIMON T., SCHWARTZ G., ZOLLHOEFER M., SHEIKH Y., SARAGIH J.: Mixture of volumetric primitives for efficient neural rendering. *ACM Trans. Graph. 40*, 4 (July 2021). URL: https://doi.org/10.1145/3450626.3459863, doi:10.1145/3450626.3459863. 1, 14, 17

[LSZ*21] LI T., SLAVCHEVA M., ZOLLHOEFER M., GREEN S., LASSNER C., KIM C., SCHMIDT T., LOVEGROVE S., GOESELE M., LV Z.: Neural 3D Video Synthesis. URL: http://arxiv.org/abs/2103.02597, arXiv:2103.02597. 14, 16

[LTJ18] LIU H.-T. D., TAO M., JACOBSON A.: Paparazzi: surface editing by way of multi-view image processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA) 37*, 6 (2018), 221–1. 8

[LZ21] LASSNER C., ZOLLHÖFER M.: Pulsar: Efficient sphere-based neural rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021). 5, 8

[LZBD21] LUAN F., ZHAO S., BALA K., DONG Z.: Unified Shape and SVBRDF Recovery using Differentiable Monte Carlo Rendering. *Computer Graphics Forum 40*, 4 (2021), 101–113. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14344, doi:https://doi.org/10.1111/cgf.14344. 20

[LZP*20] LIU S., ZHANG Y., PENG S., SHI B., POLLEFEYS M., CUI Z.: Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020). 6, 8

[LZZ*21] 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 (ICCV)* (2021). 17, 18

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108. doi:10.1109/2945.468400. 9

[MBRS*21] MARTIN-BRUALLA R., RADWAN N., SAJJADI M. S. M., 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* (2021). 9, 12, 21

[MC] MOSES W. S., CHURAVY V.: Instead of Rewriting Foreign Code for Machine Learning, Automatically Synthesize Fast Gradients. URL: https://enzyme.mit.edu. 20

[MC10] MAX N. L., CHEN M. S.: Local and global illumination in the volume rendering integral. In *Scientific Visualization: Advanced Concepts* (2010). 9

[MCL*21] 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 (ICCV)* (October 2021), pp. 6351–6361. 12

[MGK*19] 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* (2019), pp. 6878–6887. 2

[MLL*21] MARTEL J. N., LINDELL D. B., LIN C. Z., CHAN E. R., MONTEIRO M., WETZSTEIN G.: Acorn: Adaptive coordinate networks for neural representation. *ACM Trans. Graph. (SIGGRAPH)* (2021). 6, 9

[MON*19] MESCHEDER L., OECHSLE M., NIEMEYER M., NOWOZIN S., GEIGER A.: Occupancy networks: Learning 3d reconstruction in function space. In *CVPR* (2019). 11

[MPJ*19] MICHALKIEWICZ M., PONTES J. K., JACK D., BAKTASHMOTLAGH M., ERIKSSON A.: Implicit surface representations as layers in neural networks. In *Proceedings of the International Conference on Computer Vision* (2019), pp. 4743–4752. 6

[MSOC*19] 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 Trans. Graph. (SIGGRAPH) 38*, 4 (2019). 6, 10

[MST*20] 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 *ECCV* (2020). 1, 2, 3, 5, 7, 9, 10, 11, 12, 14, 17, 21

[NDVZJ19] NIMIER-DAVID M., VICINI D., ZELTNER T., JAKOB W.: Mitsuba 2: A retargetable forward and inverse renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia) 38*, 6 (Dec. 2019). doi:10.1145/3355089.3356498. 8, 20

[NFS15] 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 (CVPR)* (2015), pp. 343–352. 5

[NG20] NIEMEYER M., GEIGER A.: GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. URL: http://arxiv.org/abs/2011.12100, arXiv:2011.12100. 13

[NG21a] NIEMEYER M., GEIGER A.: CAMPARI: Camera-Aware Decomposed Generative Neural Radiance Fields. 46–48. URL: http://arxiv.org/abs/2103.17269, arXiv:2103.17269. 13, 14

[NG21b] NIEMEYER M., GEIGER A.: Giraffe: Representing scenes as compositional generative neural feature fields. In *Computer Vision and Pattern Recognition (CVPR)* (2021). 9, 14, 18

[NMOG20] NIEMEYER M., MESCHEDER L., OECHSLE M., GEIGER A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR* (2020). 6, 8, 10, 11

[NPLT*19] NGUYEN-PHUOC T., LI C., THEIS L., RICHARDT C., YANG Y.-L.: Hologan: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 7588–7597. 13, 14, 18

[NSLH21] NOGUCHI A., SUN X., LIN S., HARADA T.: Neural articulated radiance field. In *International Conference on Computer Vision (ICCV)* (2021). 14, 17

[NSP*21] NEFF T., STADLBAUER P., PARGER M., KURZ A., MUELLER J. H., CHAITANYA C. R., KAPLANYAN A., STEINBERGER M.: DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum 40*, 4 (2021), 45–59. arXiv:2103.03231, doi:10.1111/cgf.14340. 9, 10, 11

[NZIS13] NIESSNER M., ZOLLHÖFER M., IZADI S., STAMMINGER M.: Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)* (2013). 5, 21

[OMN*19] OECHSLE M., MESCHEDER L., NIEMEYER M., STRAUSS T., GEIGER A.: Texture fields: Learning texture representations in function space. In *ICCV* (2019). 5, 6

[OMT*21] OST J., MANNAN F., THUEREY N., KNODT J., HEIDE F.: Neural Scene Graphs for Dynamic Scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2021). 18

[OPG21] OECHSLE M., PENG S., GEIGER A.: Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. *arXiv preprint arXiv:2104.10078* (2021). 8, 12

[PBDCO19] PETERSEN F., BERMANO A. H., DEUSSEN O., COHEN-OR D.: Pix2vex: Image-to-geometry reconstruction using a smooth differentiable renderer. *arXiv preprint arXiv:1903.11149* (2019). 8

[PCPMMN21] 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* (2021). 9, 14, 15

[PD84] PORTER T., DUFF T.: Compositing digital images. *SIGGRAPH Comput. Graph. 18*, 3 (Jan. 1984), 253–259. URL: https://doi.org/10.1145/964965.808606, doi:10.1145/964965.808606. 9

[PDW*21] PENG S., DONG J., WANG Q., ZHANG S., SHUAI Q., BAO H., ZHOU X.: Animatable neural radiance fields for human body modeling. *arXiv preprint arXiv:2105.02872* (2021). 1, 14, 17

[PFAK20] POURSAEED O., FISHER M., AIGERMAN N., KIM V. G.: Coupling explicit and implicit surface representations for generative 3d modeling. In *European Conference on Computer Vision* (2020), Springer, pp. 667–683. 6

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: Deepsdf: Learning continuous signed distance functions for shape representation. *CVPR* (2019). 6, 11

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., DESMAISON A., KOPF A., YANG E., DEVITO Z., RAISON M., TEJANI A., CHILAMKURTHY S., STEINER B., FANG L., BAI J., CHINTALA S.: Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* (2019), vol. 32, Curran Associates, Inc. 20

[PNM*20] PENG S., NIEMEYER M., MESCHEDER L., POLLEFEYS M., GEIGER A.: Convolutional occupancy networks. In *European Conference on Computer Vision (Proceedings of the European Conference on Computer Vision)* (2020). 6, 13

[PSB*21] PARK K., SINHA U., BARRON J. T., BOUAZIZ S., GOLDMAN D. B., SEITZ S. M., MARTIN-BRUALLA R.: Nerfies: Deformable neural radiance fields. *ICCV* (2021). 1, 6, 14, 15, 16, 17

[PSDV*18] PEREZ E., STRUB F., DE VRIES H., DUMOULIN V., COURVILLE A.: Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2018), vol. 32. 14

[PSH*21] 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). 14, 15, 16

[PZvBG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels-surface elements as rendering primitives. In *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)* (7/2000 2000), pp. 335–342. 5

[PZX*21] 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 (CVPR)* (June 2021), pp. 9054–9063. 14, 17

[RCV*19] RÖSSLER A., COZZOLINO D., VERDOLIVA L., RIESS C., THIES J., NIESSNER M.: Faceforensics++: Learning to detect manipulated facial images. In *ICCV 2019* (2019). 22

[RFS21a] RÜCKERT D., FRANKE L., STAMMINGER M.: Adop: Approximate differentiable one-pixel point rendering. arXiv:2110.06635. 5

[RFS21b] RÜCKERT D., FRANKE L., STAMMINGER M.: Adop: Approximate differentiable one-pixel point rendering. *arXiv preprint arXiv:2110.06635* (2021). 8

[RMBF21] REMATAS K., MARTIN-BRUALLA R., FERRARI V.: ShaRF: Shape-conditioned Radiance Fields from a Single View. URL: http://arxiv.org/abs/2102.08860, arXiv:2102.08860. 13

[RMG*21] RICHARD A., MARKOVIC D., GEBRU I. D., KRENN S., BUTLER G., DE LA TORRE F., SHEIKH Y.: Neural synthesis of binaural speech from mono audio. In *International Conference on Learning Representations (ICLR)* (2021). 21

[ROUG17] 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* (2017), pp. 3577–3586. 6

[RPLG21] REISER C., PENG S., LIAO Y., GEIGER A.: KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. URL: http://arxiv.org/abs/2103.13744, arXiv:2103.13744. 4, 10, 11

[RRN*20] RAVI N., REIZENSTEIN J., NOVOTNY D., GORDON T., LO W.-Y., JOHNSON J., GKIOXARI G.: Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501* (2020). 7, 20

[RROG18] ROVERI R., RAHMANN L., OZTIRELI C., GROSS M.: A network architecture for point cloud classification via automatic depth images generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 4176–4184. 8

[RSH*21] REIZENSTEIN J., SHAPOVALOV R., HENZLER P., SBORDONE L., LABATUT P., NOVOTNY D.: Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *International Conference on Computer Vision* (2021). 13

[RZS*20] RAJ A., ZOLLHOEFER M., SIMON T., SARAGIH J., SAITO S., HAYS J., LOMBARDI S.: Pva: Pixel-aligned volumetric avatars. In *arXiv:2101.02697* (2020). 13

[SCT*20] SITZMANN V., CHAN E. R., TUCKER R., SNAVELY N., WETZSTEIN G.: Metasdf: Meta-learning signed distance functions. In *NeurIPS* (2020). 13

[SDZ*21] 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. *CVPR* (2021). 9, 19

[SHN*19] SAITO S., HUANG Z., NATSUME R., MORISHIMA S., KANAZAWA A., LI H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the International Conference on Computer Vision* (2019), pp. 2304–2314. 6, 13

[SK00] SHUM H., KANG S. B.: Review of image-based rendering techniques. In *Visual Communications and Image Processing 2000* (2000), vol. 4067, International Society for Optics and Photonics, pp. 2–13. 12

[SLNG20] SCHWARZ K., LIAO Y., NIEMEYER M., GEIGER A.: GRAF: Generative radiance fields for 3D-aware image synthesis. *Advances in Neural Information Processing Systems 2020-December*, NeurIPS (2020), 1–13. arXiv:2007.02442. 13, 14

[SLOD21] SUCAR E., LIU S., ORTIZ J., DAVISON A. J.: iMAP: Implicit Mapping and Positioning in Real-Time. URL: http://arxiv.org/abs/2103.12352, arXiv:2103.12352. 21, 22

[SLPS20] SCHOPS T., LARSSON V., POLLEFEYS M., SATTLER T.: Why having 10,000 parameters in your camera model is better than twelve. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020). 7

[SMB*20] SITZMANN V., MARTEL J. N., BERGMAN A. W., LINDELL D. B., WETZSTEIN G.: Implicit neural representations with periodic activation functions. In *Conference on Neural Information Processing Systems (NeurIPS)* (2020). 6, 14

[SP04] SAINZ M., PAJAROLA R.: Point-based rendering techniques. *Computers and Graphics 28*, 6 (2004), 869–879. URL: https://www.sciencedirect.com/science/article/pii/S0097849304001530, doi:https://doi.org/10.1016/j.cag.2004.08.014. 7

[SRF*21] SITZMANN V., REZCHIKOV S., FREEMAN W. T., TENENBAUM J. B., DURAND F.: Light field networks: Neural scene representations with single-evaluation rendering. In *arXiv* (2021). 10, 11, 13, 21

[SS10] SCHWARZ M., SEIDEL H.-P.: Fast parallel surface and solid voxelization on gpus. *ACM Trans. Graph. 29*, 6 (Dec. 2010). URL: https://doi.org/10.1145/1882261.1866201, doi:10.1145/1882261.1866201. 4

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings* (New York, NY, USA, 2006), ACM Press, pp. 835–846. 12

[SSSJ20] SAITO S., SIMON T., SARAGIH J., JOO H.: Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Computer Vision and Pattern Recognition (CVPR)* (2020). 17

[STB*19] SRINIVASAN P. P., TUCKER R., BARRON J. T., RAMAMOORTHI R., NG R., SNAVELY N.: Pushing the boundaries of view extrapolation with multiplane images. In *CVPR* (2019). 10

[STH*19] SITZMANN V., THIES J., HEIDE F., NIESSNER M., WETZSTEIN G., ZOLLHÖFER M.: Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR* (2019). 6, 10

[SYZR21] SU S.-Y., YU F., ZOLLHOEFER M., RHODIN H.: A-nerf: Surface-free human 3d pose refinement via neural rendering. In *Conference on Neural Information Processing Systems (NeurIPS)* (2021). 14, 17

[SZW19] SITZMANN V., ZOLLHÖFER M., WETZSTEIN G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS* (2019). 6, 8, 10, 11, 13

[TET*20] THIES J., ELGHARIB M., TEWARI A., THEOBALT C., NIESSNER M.: Neural voice puppetry: Audio-driven facial reenactment. *ECCV 2020* (2020). 16

[TFT*20] TEWARI A., FRIED O., THIES J., SITZMANN V., LOMBARDI S., SUNKAVALLI K., MARTIN-BRUALLA R., SIMON T., SARAGIH J., NIESSNER M., PANDEY R., FANELLO S., WETZSTEIN G., ZHU J.-Y., THEOBALT C., AGRAWALA M., SHECHTMAN E., GOLDMAN D. B., ZOLLHÖFER M.: State of the art on neural rendering. *EG* (2020). 2, 3

[TFT*21] TEWARI A., FRIED O., THIES J., SITZMANN V., LOMBARDI S., XU Z., SIMON T., NIESSNER M., TRETSCHK E., LIU L., MILDENHALL B., SRINIVASAN P., PANDEY R., ORTS-ESCOLANO S., FANELLO S., GUO M., WETZSTEIN G., ZHU J.-Y., THEOBALT C., AGRAWALA M., GOLDMAN D. B., ZOLLHÖFER M.: Advances in neural rendering. In *ACM SIGGRAPH 2021 Courses* (New York, NY, USA, 2021), SIGGRAPH '21, Association for Computing Machinery. URL: https://doi.org/10.1145/3450508.3464573, doi:10.1145/3450508.3464573. 15

[TLY*21] TAKIKAWA T., LITALIEN J., YIN K., KREIS K., LOOP C., NOWROUZEZAHRAI D., JACOBSON A., MCGUIRE M., FIDLER S.: Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021). 6, 8

[TMW*21] TANCIK M., MILDENHALL B., WANG T., SCHMIDT D., SRINIVASAN P. P., BARRON J. T., NG R.: Learned initializations for optimizing coordinate-based neural representations. In *CVPR* (2021). 13

[TS20] TUCKER R., SNAVELY N.: Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 551–560. 6

[TSM*20] TANCIK M., SRINIVASAN P. P., MILDENHALL B., FRIDOVICH-KEIL S., RAGHAVAN N., SINGHAL U., RAMAMOORTHI R., BARRON J. T., NG R.: Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS* (2020). 5

[TTG*20] TRETSCHK E., TEWARI A., GOLYANIK V., ZOLLHÖFER M., STOLL C., THEOBALT C.: Patchnets: Patch-based generalizable deep implicit 3d shape representations. In *European Conference on Computer Vision* (2020), Springer, Springer International Publishing, pp. 293–309. 6

[TTG*21] 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 *IEEE International Conference on Computer Vision (ICCV)* (2021), IEEE. 6, 14, 15, 16, 17

[TY20] TREVITHICK A., YANG B.: GRF: Learning a General Radiance Field for 3D Representation and Rendering. URL: http://arxiv.org/abs/2010.04595, arXiv:2010.04595. 1, 13

[TZEM17] TULSIANI S., ZHOU T., EFROS A. A., MALIK J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR* (2017). 10

[TZN19] THIES J., ZOLLHÖFER M., NIESSNER M.: Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph. 38*, 4 (2019), 1–12. 5, 22

[TZS*16] THIES J., ZOLLHÖFER M., STAMMINGER M., THEOBALT C., NIESSNER M.: Face2face: Real-time face capture and reenactment of rgb videos. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2016). 16, 22

[VKP*19] VALENTIN J., KESKIN C., PIDLYPENSKYI P., MAKADIA A., SUD A., BOUAZIZ S.: Tensorflow graphics: Computer graphics meets deep learning. 20

[Vla09] VLADSINGER: Surface control point diagram used in freeform modeling, 2009. URL: https://en.wikipedia.org/wiki/B-spline#/media/File:Surface_modelling.svg. 4

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER L. U., POLOSUKHIN I.: Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), Guyon I., Luxburg U. V., Bengio S., Wallach H., Fergus R., Vishwanathan S., Garnett R., (Eds.), vol. 30, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. 5

[WBL*20] WANG Z., BAGAUTDINOV T., LOMBARDI S., SIMON T., SARAGIH J., HODGINS J., ZOLLHÃ¶FER M.: Learning compositional radiance fields of dynamic human heads, 2020. arXiv:2012.09955. 14, 16

[WGSJ20] WILES O., GKIOXARI G., SZELISKI R., JOHNSON J.: Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (6 2020). 5, 8

[WLG*17] WANG P.-S., LIU Y., GUO Y.-X., SUN C.-Y., TONG X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG) 36*, 4 (2017), 1–11. 6

[WLL*21] WANG P., LIU L., LIU Y., THEOBALT C., KOMURA T., WANG W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS* (2021). 12

[WLR*21] 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 *ICCV* (2021). 12

[WPYS21] WIZADWONGSA S., PHONGTHAWEE P., YENPHRAPHAI J., SUWAJANAKORN S.: Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021). 6, 11

[WWG*21] WANG Q., WANG Z., GENOVA K., SRINIVASAN P., ZHOU

H., BARRON J. T., NOAH R. M.-B., FUNKHOUSER T., TECH C.: IBRNet : Learning Multi-View Image-Based Rendering. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), 4690—-4699. 13

[WWX*21] 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). 12

[XAS21] XU H., ALLDIECK T., SMINCHISESCU C.: H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. In *Advances in Neural Information Processing Systems (NeurIPS)* (2021). 14, 17

[XFYS20] XU Y., FAN T., YUAN Y., SINGH G.: Ladybird: Quasi-Monte Carlo sampling for deep implicit field based 3D reconstruction with symmetry. arXiv preprint arXiv:2007.13393, 2020. 6

[XHKK21] 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 (CVPR)* (2021), pp. 9421–9431. 14, 15, 16

[XPMBB21] XIE C., PARK K., MARTIN-BRUALLA R., BROWN M.: Fig-nerf: Figure-ground neural radiance fields for 3d object category modelling. *arXiv preprint arXiv:2104.08418* (2021). 13

[XWC*19] XU Q., WANG W., CEYLAN D., MECH R., NEUMANN U.: Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Proceedings of the IEEE International Conference on Neural Information Processing Systems (NeurIPS)* (2019), vol. 32, Curran Associates, Inc. 6

[XXH*21] XIANG F., XU Z., HAŠAN M., HOLD-GEOFFROY Y., SUNKAVALLI K., SU H.: NeuTex: Neural texture mapping for volumetric neural rendering. *CVPR* (2021). 19

[Yad19] YADAN O.: Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL: https://github.com/facebookresearch/hydra. 20

[YAK*20] YIFAN W., AIGERMAN N., KIM V. G., CHAUDHURI S., SORKINE-HORNUNG O.: Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition)* (6 2020). 6

[YGKL21a] YARIV L., GU J., KASTEN Y., LIPMAN Y.: Volume rendering of neural implicit surfaces, 2021. arXiv:2106.12052. 4

[YGKL21b] YARIV L., GU J., KASTEN Y., LIPMAN Y.: Volume rendering of neural implicit surfaces. *arXiv preprint arXiv:2106.12052* (2021). 8, 12

[YKG*20] YOON J. S., KIM K., GALLO O., PARK H. S., KAUTZ J.: Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *Computer Vision and Pattern Recognition (CVPR)* (2020). 15

[YKM*20] YARIV L., KASTEN Y., MORAN D., GALUN M., ATZMON M., BASRI R., LIPMAN Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS* (2020). 6, 8, 12

[YLT*21] YU A., LI R., TANCIK M., LI H., NG R., KANAZAWA A.: PlenOctrees for real-time rendering of neural radiance fields. In *arXiv* (2021). 9, 10, 11, 21

[YRSH21] YIFAN W., RAHMANN L., SORKINE-HORNUNG O.: Geometry-consistent neural shape representation with implicit displacement fields, 2021. arXiv:2106.05187. 6

[YSW*19a] YIFAN W., SERENA F., WU S., ÖZTIRELI C., SORKINE-HORNUNG O.: Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA) 38*, 6 (2019). 5

[YSW*19b] YIFAN W., SERENA F., WU S., ÖZTIRELI C., SORKINE-HORNUNG O.: Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA) 38*, 6 (2019). 8

[YTB*21] YENAMANDRA T., TEWARI A., BERNARD F., SEIDEL H.-P., ELGHARIB M., CREMERS D., THEOBALT C.: i3dmm: Deep implicit 3d morphable model of human heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 12803–12813. 6

[YYTK21] 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* (2021). 13, 21

[ZLLD21] ZHI S., LAIDLOW T., LEUTENEGGER S., DAVISON A. J.: In-place scene labelling and understanding with implicit scene representation. *Proc. ICCV* (2021). 21

[ZLW*21] ZHANG K., LUAN F., WANG Q., BALA K., SNAVELY N.: PhySG: Inverse rendering with spherical gaussians for physics-based material editing and relighting. *CVPR* (2021). 19

[ZPVBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *Proc. Conf. on Computer Graphics and Interactive techniques* (2001), ACM, pp. 371–378. 8

[ZPVBG02] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 223–238. 8

[ZRSK20] ZHANG K., RIEGLER G., SNAVELY N., KOLTUN V.: Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492* (2020). 9, 12

[ZSD*21] 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. *SIGGRAPH Asia* (2021). 1, 19

[ZTF*18] ZHOU T., TUCKER R., FLYNN J., FYFFE G., SNAVELY N.: Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph. (SIGGRAPH)* (2018). 6, 10

[ZYQ21] ZHANG J., YAO Y., QUAN L.: Learning signed distance field for multi-view surface reconstruction. *arXiv preprint arXiv:2108.09964* (2021). 8