Neural Radiance: From NeRFs to 3D Gaussian Splatting

Ying Shen
ying22@illinois.edu

Junkun Chen
junkun3@illinois.edu

Abstract

This survey studies Neural Radiance-based scene representations for 3D scene reconstruction and rendering. Since the emergence of Neural Radiance Fields (NeRFs) in 2020, NeRFs have revolutionized the 3D scene representations, by modeling volumetric radiance and density fields with learnable deep neural networks. Such a representation produces high-quality and high-fidelity reconstruction and rendering results with the simplest and most intuitive modeling, inspiring academia with diverse directions in improving, enhancing, and strengthening the NeRF modeling, aiming to address challenges, including rendering realism, training and/or inference efficiency, complicated visual effects support, etc, while opening up a new way for 4D (dynamic 3D) reconstruction and rendering. In 2023, 3D Gaussian Splatting (3DGS) introduced a brand-new paradigm to model the 3D scene as Gaussian splattings and render with rasterization, offering advantages in both high-efficiency training/inference and even better rendering quality. By providing an in-depth analysis of NeRFs and 3DGS about their development, variants, applications, etc. along with a comprehensive comparison, this survey aims to illuminate the current landscape of neural radiance techniques and identify potential directions for future research in 3D vision.

1. Introduction

In recent years, Neural Radiance Fields (NeRFs) [31] have emerged as a groundbreaking method for 3D scene reconstruction and rendering. Introduced in 2020, NeRFs utilize deep neural networks to model volumetric radiance and density fields, enabling the synthesis of photorealistic images from novel viewpoints. By representing scenes with a continuous function that maps spatial coordinates and viewing directions to color and density, NeRFs provide a simple yet powerful framework for capturing complex geometry and intricate visual effects. This approach not only achieves high-quality and high-fidelity 3D scene reconstruction but also shows the potential of such a simple, intuitive, yet effective modeling. The remarkable performance of NeRFs has inspired extensive follow-up research aiming at different aspects of improving, enhancing, and strengthening the vanilla NeRF, including improving rendering realism, enhancing computational efficiency during training and inference, and extending support for complicated visual effects, thereby broadening the scope and applicability of neural radiance-based representations.

Despite the effectiveness that is already shown by the vanilla NeRF, the original framework relies solely on simple multilayer perceptrons (MLPs) to model the radiance field, without leveraging explicit 3D structural information. To harness more specialized representations and achieve corresponding benefits, a popular research direction involves replacing the MLPs with alternative models or structures. This has led to one branch of the NeRF's development – underlying structures. The explicit underlying structures can be broadly categorized into voxel-based, tree-based, and point-based methods. Voxel-based methods [5, 34, 44, 50] organize the radiance field using either dense voxel grids or tri-plane decomposed voxel grids, enhancing rendering efficiency through simpler ray propagation and locating. Sparse voxel and Tree-based methods [27, 41, 60] employ hierarchical data structures like OCTrees to store the radiance field, improving rendering precision by allowing an adaptive level of detail. Point-based methods [9, 52] utilize explicit representations such as point clouds or meshes to directly model object surfaces, representing the radiance field through pairs of sample points and their associated values, which can also be regarded as a former work of 3D Gaussian splatting. These different underlying models introduce unique advantages – such as increased efficiency, precision, or fidelity – while expanding the range of applications for neural radiance representations.

Other than altering the whole underlying representation of the radiance field, another exciting branch of research is to augment the current underlying representation. As the vanilla NeRF models the color as simply as the function of a single position and the viewing direction, some of the visual effects might not be well modeled. Therefore, one main idea of this branch is to introduce a better modeling of color to explicitly or implicitly support more visual effects. These approaches include introducing a multi-scale color modeling [1] or applying some different or additional

color modeling with more detailed attributes [15,46]. These approaches expand the boundary of NeRF's modeling quality.

The above research directions are still mainly improving the basic rendering ability of NeRF, while there is also a branch of research aiming to add more functionalities or applications within the NeRF framework. As vanilla NeRF needs to be trained individually on each scene (as a 'per-scene' model), one hot direction is to introduce or improve cross-scene generalization [6, 47, 61], to make NeRFs able to fit on novel scenes in an efficient way and/or with only a few views with some prior knowledge. Another hot direction is to support scene editing which is not supported by vanilla NeRF, including object manipulation [21, 55], shape editing [9, 38, 53, 63], and further introduces diffusion for native or distillation-guided scene editing and generation [7, 8, 11, 16, 33, 40, 48]. Finally, there is also a popular direction to introduce NeRF to 4D (dynamic 3D) scenes [4, 14, 43] along with all the applications above.

Notably, in 2023, the emergence of 3D Gaussian splatting (3DGS) [20] introduced a brand-new paradigm of radiance fields. 3DGS can be regarded as an updated version of point-based NeRFs, where it offers an explicit representation by modeling complex scenes as a set of learnable 3D Gaussians instead of single points, which improves the multi-scale capability. Notably, 3DGS also introduces rasterization to take the place of standard volume rendering to render the scene in a similar way like "projecting" or "plotting" the Gaussians onto the view plane. There are also similar follow-ups and extensions of 3DGS, including memory efficiency [10, 12, 13, 23, 35, 37], reconstruction quality [18, 19, 24, 36, 54, 62], and extension to 4D scenes (dynamic 3D scenes) [22, 25, 26, 29, 49, 56, 57].

In this survey, we aim to provide a comprehensive overview of the development of radiance fields, from NeRFs to 3DGS, from plain rendering to various applications. We also provide a comparison between different models, and show some potential directions of future research in 3D scene reconstruction and various topics.

2. NeRF

2.1. Vanilla NeRF and NeRF Frameworks

Vanilla NeRF [31] was published in ECCV 2020 and nominated as one of the best paper candidates. The key idea of NeRF is to directly simulate the rendering formula for a ray $r(t) = o + t \cdot \mathbf{d}$, namely

$$C(r) = \int_{t_l}^{t_r} T(t) \cdot \sigma(r(t)) \cdot c(r(t), \mathbf{d}) \cdot \mathrm{d}t, \qquad (1)$$

where $C(r)$ is the final rendered pixel value as an RGB color, $\sigma(x)$ is the volume density of 3D point location x, $c(r(t), \mathbf{d})$ is the view-dependent color as a function of x and viewing direction $\mathbf{d}$, and $T(t) = \exp\left(-\int_{t_l}^{t} \sigma(r(s)) \cdot \mathrm{d}s\right)$ is the accumulated transmittance. Instead of calculating this integration by definition, NeRF applies a discrete approximation with sampled points $\{t_i\} \subset [t_l, t_r]$, so that

$$C(r) \approx \sum_i T(t_i)(1 - e^{-\sigma(r(t_i)) \cdot (t_{i+1} - t_i)} \cdot c(r(t_i), \mathbf{d}), \quad (2)$$

where $T(t_i)$ can also be approximated as

$$T(t_i) \approx \exp\left(-\sum_{j<i} \sigma(r(t_j)) \cdot (t_{j+1} - t_j)\right). \qquad (3)$$

With such approximation, NeRF converts the original rendering task to modeling and learning of the two functions $c(x, \mathbf{d})$ and $\sigma(x)$, where x is a 3D coordinate and $\mathbf{d}$ is a viewing direction, which is a standard machine learning task that can be solved with multi-layer perceptrons (MLPs).

One remaining task is how to efficiently sample the points $\{t_i\}$, so that it will not waste time in vacuum spaces and will focus more on the objects - this is called "density-adaptive rendering". This objective relies on the model of the volume density $\sigma(\cdot)$, as vacuum spaces imply low volume density and vice versa. NeRF proposes a coarse-to-fine strategy, which trains a coarse network of $\sigma(\cdot)$ with uniformly sampled $\{t_i\}$ to get an estimation of $\sigma(\cdot)$'s distribution over the ray, and then do the actual sampling accordingly on another fine network of $\sigma(\cdot)$. The former coarse network is also regarded as a "proposal network" by follow-up work [45] as we can query it to get several proposals of sampled points.

The vanilla NeRF proposes the paradigm of the NeRF framework: to model the functions $c(x, \mathbf{d})$ and $\sigma(x)$ in a learnable way to simulate the rendering formula. The common challenges or tasks of NeRF framework are threefolds:

- Task 1 (Modeling): How to design the learnable network for $c(x, \mathbf{d})$ and $\sigma(x)$;
- Task 2 (Density-Adaptivity): How to render in a density-adaptive way, that skip vacuum spaces and focuses on the objects; and
- Task 3 (Render): How to efficiently and effectively render images from, and optimize the whole network.

In the vanilla NeRF, Task 1 is completed by using a simple MLP, Task 2 is resolved by using a coarse-to-fine strategy and applies rendering and training by definition for Task 3. This approach is simple and intuitive but still suffers from many different disadvantages, including rendering quality and training efficiency, leading to many follow-up works aiming for improvement.

2.2. Various Underlying Models

The underlying MLP in vanilla NeRF is simple but limits many possible improvements and extensions, as it does not use any explicit 3D structures and lacks interpretability. Therefore, there is a branch of follow-up works that propose novel NeRF variants by replacing its underlying models with more tailored designs.

One popular alternative underlying representation is voxels, which divides the 3D space into $N \times N \times N$ equally-sized square voxel grids. In a typical method like VoxelNeRF [44], the learnable parameters are the features of each vertex of each voxel, and the $c(x, \mathbf{d})$ and $\sigma(x)$ can be computed by trilinearly interpolate the feature at x with the voxel grid containing it then use an MLP to decode these features into the colors and volume densities. As the coordinates (x, y, z) can be easily converted to the corresponding voxel indices (i_x, i_y, i_z) and each voxel already decreases the complexity in each grid, a shallow MLP can be used here instead of a large MLP in vanilla NeRF, leading to high computational efficiency. And the density adaptivity can also be simply and efficiently implemented by scanning all the intersected voxels (no more than $2N$ voxels, while N is small as the number of parameters is $O(N^3)$). Based on the idea of voxels, there are also several follow-ups. Observing that the rendering process actually covers the whole ray, and all sampled points' features are calculated with trilinear interpolation, DIVeR [50] replaced point sampling with deterministic integration with trilinear interpolated segment, highly improves the rendering speed to real-time inference. Instant-NGP [34] decomposes a single large voxel grids into several smaller voxels used as hash tables, and the final feature is the concatenation of the features modeled by each small voxels, which significantly improves the training efficiency. Tri-planes [5] is another idea of decomposition, which decomposes the voxel into three planes, namely xOy, yOz, and zOx planes with features at grid voxels, where each vertice's feature is the concatenation of the features of the projected points at each of the three planes, significantly decreases the memory consumption and allows a finer grid. However, all these methods suffered from some quality drop due to the limited granularity of the voxel grids.

As a follow-up of voxels, sparse-voxel-based and OCTree-based methods are proposed, to overcome the challenge of granularity of the dense voxels. NSVF [27] applies a similar idea to use sparse voxels instead of dense voxels through self-pruning and progressive training, achieving high-quality editing. As a more aggressive method, based on sparse voxel grids, [41] even completely get rid of the neural networks by using spherical harmonics as the color features, which can be directly converted to the view-dependent color as the function of the viewing direction. In another direction, the OCTree can be regarded as a set of multi-scale voxels as different layers of nodes, so that each node can be regarded as a "voxel". The top-down structure of OCTrees is a native top-down procedure to skip vacuum spaces by skipping a vacuum node with all its subnodes, and the multi-scale voxels also prevent the granularity issue of plain voxels. PlenOctree [60] is such a method that uses OCTrees to organize the NeRF and support real-time rendering.

The actual key idea behind sparse voxels is to focus more on the object, and especially the surface of objects. The point-based NeRFs are the more aggressive and tailored ones that directly models the surface of the objects with the point cloud, but in a learnable way instead of a traditional way. PointNeRF [52] is the first point-based NeRF, which models the radiance field as a point cloud with features. Each arbitrary point x's feature is calculated by the interpolation of x's KNNs in the point cloud, and the density-adaptive method is achieved with CAGQ [51] method, a specific method to divide the point cloud into several bounding boxes for ray intersection detection. In point-based NeRFs, the optimization in Task 3 is not an intuitive method, and therefore, they proposed the "pruning and growing" strategy to dynamically add or remove the points during training. As a follow-up, NeuralEditor [9] proposed an improved point-based NeRF that uses K-D Trees to organize the points and apply DIVeR-like spline integration for rendering, while models normal vectors for better color modeling, which improves the rendering quality and supports the specific editing task researched by their paper. These point-based NeRFs can be regarded as the previous work of 3D Gaussian splatting [20], as they share a similar idea of using points/Gaussians to model the surface with a specific optimization strategy.

2.3. More Visual Effects

The vanilla NeRF models the color as simply as the function of a single position and the viewing direction, which does not take every visual effect into account and limits the rendering quality. Therefore, another branch of research aims to explicitly introduce and model those visual effects in the model.

Observing that a pixel in the rendered image is not just a single ray but a cone, MipNeRF [1] introduces a multi-scale representation that integrates over a conical frustum along the ray through an integrated positional encoding, enabling scale-aware rendering and reduces the aliases. ZipNeRF [2] follows this idea and combines the voxel-based models with data structures, achieves comparable performance, and significantly improves the training speed.

On the other hand, there are also some papers focusing on the view-dependent effects. One better way to model the view-dependent color is to use spherical harmonics (SHs), which model a function $f(\mathbf{d})$ using SH bases. This color modeling is widely used in recent NeRFs and even Gaus-

sian splattings. RefNeRF [46] further studies the view-dependent color, by modeling normal vectors and model the view-dependent color as the function of the reflection of viewing direction by the normal vector (instead of the viewing direction), which is consistent with the Phong reflection model [39], and achieves high-quality specular effects. NeRFReN [15] specifically supports reflection effects by modeling the world in the mirror with another NeRF. By supporting more visual effects, the rendering quality can be improved with the correct color modeling.

As most of these approaches are agnostic to the underlying model, the two parts of improvements can be combined. For example, NeuralEditor [9] also models normal vectors but in an easier way to estimate them from KNNs on the point cloud, and then applies RefNeRF's reflection modeling.

2.4. Extensions and Applications

Beyond the basic rendering ability of NeRF, there is also a branch of research aiming to add more functionalities or applications within the NeRF framework.

As vanilla NeRF is a per-scene model, which needs to be trained from scratch for each scene, one direction is to improve the generalizability across different scenes. The key idea is to first pre-train some models on large scene datasets to obtain some prior, then use these priors to serve as a good initialization of the model. In this case, these models can not only converge fast on a novel scene, but also potentially support reconstructing a scene with only a few views (i.e. few-shot reconstruction). PixelNeRF [61] encodes the input views into features and uses it as an additional input to the MLP of the NeRF, so that such MLP can be trained to be scene-generalizable by reasoning the feature. IBRNet [47] follows the traditional idea of image-based rendering, which warps the information from other views and uses a scene-generalizable ray transformer to obtain the volume density and the view-dependent colors. MVSNeRF [6] follows another traditional idea of scene-generalizable pre-trained multi-view stereo (MVS) [58] models to generate the 3D cost volume, and convert such volume to the radiance field with another 3D convolutional network for volume rendering. Similarly, PointNeRF [52] also uses an MVS to initialize the point clouds at the first step of optimization. By utilizing scene-generalizable components, the training can be significantly more efficient with better initialization.

Another popular extension or application is to support scene editing, which includes, but is not limited to, changing the color or position of scene objects, deforming the shape, or changing the textures with different user interfaces. All these are not directly supported by vanilla NeRF, as it is hard to predict the abstract MLP parameters after the editing operation. DeformingNeRF [53], NeRF-Editing [63], and Cage-NeRF [38] are three papers that support scene deformation, which first convert the prime task of "render in the *deformed* scene (with *original* viewing rays)" to the dual task "render in the *original* scene with *deformed* viewing rays", and then record the deformation of rays using cages. NeuralEditor [9] is a follow-up work that proposes a better point-based NeRF that supports directly deforming the underlying point cloud to simply perform deformation and even supports more shape editing like cutting in the middle or morphing. DistillNeRF [21] aims to solve some object-level manipulation, by distilling 2D image features to NeRFs and render these features at novel views to perform editing.

Other than editing an existing NeRF, there are also works that aims to generate novel scenes, especially with the emerged diffusion models [17]. DiffRF [33] and SSDNeRF [7] trains a diffusion model to directly generate the underlying voxel or tri-plane features for the NeRF to generate a NeRF representation of a 3D scene directly. DreamFusion [40] proposes score distillation sampling (SDS) to distill the generation signal from 2D diffusion model to 3D to perform open-world text-guided 3D scene generation with 2D diffusion models, while ProlificDreamer [48] proposed an improved distillation method that further improves the results. This idea of distilling from 2D diffusion also affects the scene editing. Instruct-NeRF2NeRF [16] is one of these work to apply text-guided 3D scene editing with the image editing diffusion Instruct-Pix2Pix [3]. ConsistDreamer [8] further improves editing quality and fidelity by introducing several consistency-enforcing strategies.

Finally, there is also a direction that extends NeRF's approaches to supporting 4D (dynamic 3D) scenes. The main idea is still intuitive: model the volume density as $\sigma(x, t)$, and the color $c(x, \mathbf{d}, t)$, where the additional input t is the timestep. K-Planes [14] is a model that extends Tri-planes by adding three new planes: xOt, yOt, and zOt, and models the dynamic scene in a similar way as tri-planes. NeRFPlayer [43] decomposes the dynamic scene modeling into (1) stationary field, (2) deformation field, (3) newness field, and (4) decomposition field, to model the dynamic scene as the deformation and addition/deletion from the scene at the first step. Also, similar applications are also introduced to 4D, e.g., Instruct 4D-to-4D [32] introduces distillation-based text-guided scene editing to 4D scenes.

3. 3D Gaussian Splatting

3.1. Vanilla 3DGS

Recently, 3D Gaussian Splatting (3DGS) [20] has emerged as a powerful technique for achieving impressive novel view synthesis results, while achieving real-time rendering at high-definition resolutions. 3DGS employs an explicit radiance field-based scene representation that models

Model	Rendering Quality on NeRF Synthetic Dataset [31], PSNR ↑							
	Chair	Hotdog	Lego	Drums	Ficus	Materials	Mic	Ship
Traditional Methods (For Reference)								
SRN [42]	26.96	17.18	20.73	26.81	20.85	18.09	26.85	20.60
NV [28]	28.33	22.58	24.79	30.71	26.08	24.22	27.78	23.93
LLFF [30]	28.72	21.13	21.79	31.41	24.54	20.72	27.48	23.22
NeRFs								
NeRF [31]	33.00	25.01	30.13	36.18	32.54	29.62	32.91	28.65
PlenOctrees [60]	34.66	25.31	30.79	36.79	32.95	29.76	33.97	29.42
DIVeR [50]	34.34	25.39	31.77	36.83	35.52	29.63	34.58	30.50
Instant-NGP [34]	31.02	34.86	32.77	24.18	28.74	28.93	31.89	28.06
NSVF [27]	33.19	25.18	31.23	37.14	32.29	32.68	34.27	27.93
Plenoxels [41]	33.98	25.35	31.83	36.43	34.10	29.14	33.26	29.62
PlenOctree [60]	34.66	25.31	30.79	36.79	32.95	29.76	33.97	29.42
PointNeRF [52]	35.40	26.06	35.04	35.95	29.61	30.97	37.30	36.13
MipNeRF [1]	37.14	27.02	33.19	39.31	35.74	32.56	38.04	33.08
ZipNeRF [2]	34.84	25.84	33.90	37.14	34.84	31.66	35.15	31.38
RefNeRF [46]	35.83	36.25	35.41	36.76	37.72	33.91	25.79	30.28
IBRNet [47]	28.18	21.93	25.01	31.48	25.34	24.27	27.29	21.48
MVSNeRF [6]	26.80	22.48	26.24	32.65	26.62	25.28	29.78	26.73
3D Gaussian Splatting								
3DGS [20]	35.83	37.72	35.78	26.15	34.87	30.00	35.36	30.80
LightGaussian [12]	34.77	36.46	34.94	26.02	34.48	29.34	35.37	30.41
Compact3D [23]	34.91	37.38	35.48	26.18	35.44	29.97	35.81	31.51
Mip-Splatting [62]	35.69	26.50	32.99	36.18	32.76	30.01	31.66	29.98
2DGS [18]	35.05	37.36	35.10	26.05	35.57	29.74	35.09	30.60

Table 1. Comparison on rendering quality of NeRF synthetic dataset [31].

complex scenes as a set of learnable 3D Gaussians, which allows for flexible and expressive scene representation.

In 3DGS, each 3D Gaussian is defined by a position (mean) $\mu \in \mathbb{R}^3$, covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$, opacity $\alpha \in [0, 1]$, and spherical harmonic parameters $\mathcal{C} \in \mathbb{R}^k$, where k is the degrees of freedom, for modeling view-dependent color. The 3D Gaussian is mathematically represented as:

$$G(x) = e^{-\frac{1}{2}(x)^T\Sigma^{-1}(x)}. \tag{4}$$

To constrain Σ to the space of valid covariance matrices, it is further decomposed into a rotation matrix $R \in \mathbb{R}^{3\times3}$ and scaling matrix $S \in \mathbb{R}^{3\times3}$:

$$\Sigma = RSS^TR^T. \tag{5}$$

Rendering in 3DGS involves splatting these 3D Gaussians onto the image plane, followed by alpha compositing based on depth to create the final rendered image. Specifically, the projection of 3D Gaussian ellipsoids can formulated as follows:

$$\Sigma' = JW\Sigma W^TJ^T, \tag{6}$$

where Σ' is the projected 2D covariance matrix given the viewing transformation W and J is the Jacobian matrix for the projective transformation. To enable fast and parallel rendering, 3DGS introduces a tile-based rasterizer that divides the image into multiple non-overlapping tiles. Then, for each tile in the image, the projected Gaussians that intersect with the tile are sorted by depth, forming a sorted list of Gaussians $\mathcal{N}$. The color of each pixel in the tile is computed by alpha compositing, as follows:

$$C = \sum_{i\in\mathcal{N}} c_i\alpha_i\Pi_{j=1}^{i-1}(1-\alpha_j), \tag{7}$$

where α_i represents the opacity of this Gaussian splat multiplied by the density of the projected 2D Gaussian and c_i represents the color.

The optimization of 3DGS is based on successive iterations of differentiable rendering and comparing the resulting image to the training views via photometric loss. In each iteration, the system renders an image using the current parameters of the 3D Gaussians, including their positions, sizes, rotations, colors, and opacities, and then these learnable parameters can then be adjusted via gradient-based optimization, enabling the Gaussians to better represent the 3D scene given by a set of input images. The optimization of these parameters is interleaved with adaptive density con-

trol steps, where it adds and removes 3D Gaussians during optimization.

3.2. Advancements and Extensions of 3DGS

Although 3D Gaussian Splatting (3DGS) has shown remarkable results in novel-view synthesis with real-time rendering capabilities, there remains significant potential for improvement. One area for enhancement is **memory efficiency**, particular in optimizing memory usage in both model training and storage. While 3DGS excels in capturing high-quality scenes, this often comes at the cost of enormous memory consumption, which can limit its scalability. Recent works have mainly focused on two main directions to address this issue: (1) reducing the number of 3D Gaussians required to represent a scene without sacrificing quality via pruning or masking methods [12, 13, 37], and (2) compressing the memory usage of the Gaussians' properties, such as positions, colors, and covariance matrices [10, 23, 35].

Furthermore, improvements in reconstruction **quality** are still necessary to address the aliasing and artifacts issue for enhanced quality. [54] observe that 3DGS is susceptible to aliasing when dealing with varying resolutions, which leads to blurring or jagged edges. They, therefore, propose a multi-scale 3D Gaussian splatting algorithm, which maintains Gaussians at different scales to represent the same scene. Mip-Splatting [62] aims to eliminate the artifacts when changing sampling rates by introducing a 3D smoothing filter to constrain the size of the 3D Gaussian primitives based on the maximal sampling frequency induced by the input views. In addition, to avoid the dilation effects, it introduces another 2D mip filter to the projected Gaussian ellipsoids, replacing the original 2D dilation filter. Beyond improving rendering details, other works focus on enhancing reconstruction quality from different aspects [18, 19, 24, 36]. For example, Radsplat [36] proposes to leverage a trained neural radiance field as a prior and supervision signal for optimizing point-based scene representations, leading to improved quality and more robust optimization. 2DGS [18] collapses the 3D volume into a set of 2D oriented planar Gaussian disks, aiming to improve the quality of the reconstructions by leveraging the view-consistent geometry.

Moreover, 3DGS can be extended to general **4D (dynamic 3D) scenes**, enabling the representation of more complex environments over time. Dynamic 3D Gaussians [29] first proposes to adapt 3DGS to dynamic scenarios. It models the dynamic 3D scene with a set of moving 3D Gaussians, parameterized by 6-DOF rotation and translation. By incorporating local-rigidity constraints, Dynamic 3D Gaussians correctly model the same area of physical space over time, including the rotation of that space. Following this, other approaches have emerged to model scene dynamics via motion basis [22, 25, 26] or deformation fields [49, 56]. For instance, Deformable 3D Gaussians [56] is the first work that extends 3DGS for dynamic scenes through a deformation field. This method reconstructs scenes by learning Gaussians in canonical space, which are then transformed by a deformation field to represent monocular dynamic scenes. In addition, 4D Gaussian Splatting [57] designs a 4D Gaussian representation that models both the space and time dimensions for dynamic scenes. This approach optimizes a collection of 4D primitives to accurately fit the underlying spatiotemporal 4D volume of dynamic environments, enabling real-time high-fidelity video synthesis.

4. Discussion

4.1. Evaluation and Comparison

We provide a comprehensive comparison between most of the methods in Table 1 on the most widely-used NeRF synthetic [31] dataset, with the peak signal-to-noise (PSNR) metric to evaluate the reconstruction fidelity. All the numbers are taken from their original paper.

The papers on radiance fields also use some other datasets, including NeRF synthetic 360 [31], LLFF [30], ScanNet++ [59], etc. Along with PSNR, they also use structural similarity index measure (SSIM) and learned perceptual image patch similarity (LPIPS) metrics to quantify their rendering fidelity from different aspects.

4.2. Future Directions and Research Opportunities

According to our survey, there are many branches of research in the field of radiance field. For example, some good insights and improvements might be transferred from NeRF to 3DGS, including a more controllable, editable, and manipulation-supportive 3D Gaussian splatting, and scene-generalizable 3DGS. Also, to align with the trend of diffusion-guided generation, some other future research directions also includes be the generation or editing of scenes with video diffusion models and a feed-forward 3D Gaussian diffusion model. Another interesting direction for future work involves the integration of physics with NeRF or 3DGS, enabling the simulation of more complex and large-scale dynamic scenes. By merging physics with radiance fields, 3DGS could represent not only object appearance but also their physical behaviors, enabling more comprehensive scene representations. Another area of opportunities lie in the integration of semantic or geometric information into radiance field-based representations, facilitating applications that require semantic scene understanding, precise object recognition, segmentation, or structural consistency. For instance, embedding prior knowledge about object shapes or affordance could enhance the precision of scene reconstruction and improve interaction with dynamic objects. This

would benefit fields like robotics, where understanding object types and behaviors is essential for decision-making in real-world environments.

5. Conclusion

In this survey, we introduced the development and several major branches of research in radiance fields. There are still many interesting research topics that are under exploration in the field. We hope that our survey will inspire more future research in this direction.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021. 1, 3, 5

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields, 2023. 3, 5

[3] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Learning to follow image editing instructions. In *CVPR*, 2023. 4

[4] Ang Cao and Justin Johnson. HexPlane: A fast representation for dynamic scenes. In *CVPR*, 2023. 2

[5] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks, 2022. 1, 3

[6] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021. 2, 4, 5

[7] Hansheng Chen, Jiatao Gu, Anpei Chen, Wei Tian, Zhuowen Tu, Lingjie Liu, and Hao Su. Single-stage diffusion NeRF: A unified approach to 3D generation and reconstruction. In *ICCV*, 2023. 2, 4

[8] Jun-Kun Chen, Samuel Rota Bulò, Norman Müller, Lorenzo Porzi, Peter Kontschieder, and Yu-Xiong Wang. ConsistDreamer: 3d-consistent 2d diffusion for high-fidelity scene editing. In *CVPR*, 2024. 2, 4

[9] Jun-Kun Chen, Jipeng Lyu, and Yu-Xiong Wang. NeuralEditor: Editing neural radiance fields via manipulating point clouds. In *CVPR*, 2023. 1, 2, 3, 4

[10] Yihang Chen, Qianyi Wu, Jianfei Cai, Mehrtash Harandi, and Weiyao Lin. Hac: Hash-grid assisted context for 3d gaussian splatting compression. *arXiv preprint arXiv:2403.14530*, 2024. 2, 6

[11] Jiahua Dong and Yu-Xiong Wang. ViCA-NeRF: View-consistency-aware 3D editing of neural radiance fields. In *NeurIPS*, 2023. 2

[12] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023. 2, 5, 6

[13] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. *arXiv preprint arXiv:2403.14166*, 2024. 2, 6

[14] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 2, 4

[15] Yuan-Chen Guo, Di Kang, Linchao Bao, Yu He, and Song-Hai Zhang. NeRFReN: Neural radiance fields with reflections. In *CVPR*, 2022. 2, 4

[16] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D scenes with instructions. In *ICCV*, 2023. 2, 4

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 4

[18] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2, 5, 6

[19] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5322–5332, 2024. 2, 6

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *TOG*, 2023. 2, 3, 4, 5

[21] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing NeRF for editing via feature field distillation. In *NeurIPS*, 2022. 2, 4

[22] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. *arXiv preprint arXiv:2312.00112*, 2023. 2, 6

[23] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024. 2, 5, 6

[24] Yanyan Li, Chenyu Lyu, Yan Di, Guangyao Zhai, Gim Hee Lee, and Federico Tombari. Geogaussian: Geometry-aware gaussian splatting for scene rendering. *arXiv preprint arXiv:2403.11324*, 2024. 2, 6

[25] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8508–8520, 2024. 2, 6

[26] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21136–21145, 2024. 2, 6

[27] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 1, 3, 5

[28] Stephen Lombardi, Tomas Simon, Jason M. Saragih, Gabriel Schwartz, Andreas M. Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *CoRR*, abs/1906.07751, 2019. 5

[29] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 2, 6

[30] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4), July 2019. 5, 6

[31] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 5, 6

[32] Linzhan Mou, Jun-Kun Chen, and Yu-Xiong Wang. Instruct 4D-to-4D: Editing 4d scenes as pseudo-3d scenes using 2d diffusion. In *CVPR*, 2024. 4

[33] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. DiffRF: Rendering-guided 3D radiance field diffusion. In *CVPR*, 2023. 2, 4

[34] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 3, 5

[35] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10349–10358, June 2024. 2, 6

[36] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv preprint arXiv:2403.13806*, 2024. 2, 6

[37] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3d gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1):1–17, 2024. 2, 6

[38] Yicong Peng, Yichao Yan, Shengqi Liu, Yuhao Cheng, Shanyan Guan, Bowen Pan, Guangtao Zhai, and Xiaokang Yang. CageNeRF: Cage-based neural radiance field for generalized 3D deformation and animation. In *NeurIPS*, 2022. 2, 4

[39] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975. 4

[40] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *ICLR*, 2023. 2, 4

[41] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 1, 3, 5

[42] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019. 5

[43] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. NeRFPlayer: A streamable dynamic scene representation with decomposed neural radiance fields. *TVCG*, 2023. 2, 4

[44] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *CoRR*, abs/2111.11215, 2021. 1, 3

[45] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH*, 2023. 2

[46] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *CVPR*, 2022. 2, 4, 5

[47] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2, 4, 5

[48] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. In *NeurIPS*, 2023. 2, 4

[49] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, June 2024. 2, 6

[50] Liwen Wu, Jae Yong Lee, Anand Bhattad, Yu-Xiong Wang, and David Forsyth. DIVeR: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *CVPR*, 2022. 1, 3, 5

[51] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-GCN for fast and scalable point cloud learning. In *CVPR*, 2020. 3

[52] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *CVPR*, 2021. 1, 3, 4, 5

[53] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *ECCV*, 2022. 2, 4

[54] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20923–20931, 2024. 2, 6

[55] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *ICCV*, 2021. 2

[56] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024. 2, 6

[57] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *The Twelfth International Conference on Learning Representations*. 2, 6

[58] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018. 4

[59] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *ICCV*, 2023. 6

[60] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 1, 3, 5

[61] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2, 4

[62] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 2, 5, 6

[63] Yu-Jie Yuan, Yang tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. NeRF-Editing: Geometry editing of neural radiance fields. In *CVPR*, 2022. 2, 4