electronics

MDPI

Review

A Brief Review on Differentiable Rendering: Recent Advances and Challenges

Ruicheng Gao [1] and Yue Qi [2,1,3,*]

[1] State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[2] Jingdezhen Research Institute of Beihang University, Jingdezhen 333000, China
[3] Qingdao Research Institute of Beihang University, Qingdao 266104, China
* Correspondence: qy@buaa.edu.cn

Abstract: Differentiable rendering techniques have received significant attention from both industry and academia for novel view synthesis or for reconstructing shapes and materials from one or multiple input photographs. These techniques are used to propagate gradients from image pixel colors back to scene parameters. The obtained gradients can then be used in various optimization algorithms to reconstruct the scene representation or can be further propagated into a neural network to learn the scene's neural representations. In this work, we provide a brief taxonomy of existing popular differentiable rendering methods, categorizing them based on the primary rendering algorithms employed: physics-based differentiable rendering (PBDR), methods based on neural radiance fields (NeRFs), and methods based on 3D Gaussian splatting (3DGS). Since there are already several reviews for NeRF-based or 3DGS-based differentiable rendering methods but almost zero for physics-based differentiable rendering, we place our main focus on PBDR and, for completeness, only review several improvements made for NeRF and 3DGS in this survey. Specifically, we provide introductions to the theories behind all three categories of methods, a benchmark comparison of the performance of influential works across different aspects, and a summary of the current state and open research problems. With this survey, we seek to welcome new researchers to the field of differentiable rendering, offer a useful reference for key influential works, and inspire future research through our concluding section.

Keywords: differentiable rendering; analysis-by-synthesis; global illumination; neural radiance field; 3D Gaussian splatting

Citation: Gao, R.; Qi, Y. A Brief Review on Differentiable Rendering: Recent Advances and Challenges. *Electronics* **2024**, *13*, 3546. https://doi.org/10.3390/electronics13173546

Academic Editor: Beiwen Li

Received: 19 August 2024
Revised: 1 September 2024
Accepted: 3 September 2024
Published: 6 September 2024

1. Introduction

Differentiable rendering techniques have emerged as a popular research topic in the areas of virtual/augmented reality, computer vision, and computer graphics. The rendering process can be understood as a function that maps the parameters of geometries, materials, lights, and cameras to the image pixels' intensities; i.e., the data flows from scene parameters to image pixels. On the contrary, differentiable rendering techniques aim to propagate the gradients of image pixels to the scene parameters in the opposite direction; i.e., the data flows from image pixels to scene parameters.

The gradients obtained through differentiable rendering techniques can be paired with various optimization algorithms to solve inverse-rendering (also known as analysis-by-synthesis) problems such as reflectance and lighting estimation [1] and 3D reconstruction from one or multiple input photographs [2,3]. The gradients can also be propagated further into neural networks to learn the scene's neural representations [4], enabling applications such as novel view synthesis [5,6], scene relighting [7], and scene editing [8].

When we have challenging inverse rendering problems at hand, high-quality gradients are required to ensure that the corresponding optimization process converges to a valid

result. Note that the differentiable rendering techniques we use to obtain gradients are largely determined by the primary rendering process. Therefore, high-quality gradients come from a high-quality primary rendering process, and the rendering algorithms of the highest quality are undoubtedly global illumination algorithms.

Global illumination algorithms, obeying the rules of geometric optics, use Monte Carlo methods to solve the path integral and then render photorealistic images. However, the differentiable rendering techniques corresponding to global illumination algorithms, known as physics-based differentiable rendering methods [9–11], are much more complicated due to the high-order discontinuities contained in the integrand of the path integral, which combines the parameters of geometries, materials, lights, and cameras into a single entity.

Directly applying automatic differentiation techniques to global illumination algorithms will not yield physically correct gradients. Just like the Leibniz integral rule in calculus, the derivative of the path integral contains an additional boundary term due to the movement of discontinuities when the scene's geometric parameters change.

There are two main methods to handle the movement of geometric discontinuities in the physics-based differentiable rendering community: boundary sampling methods [9,11–14] and reparameterization methods [10,15,16]. Boundary sampling methods aim to explicitly calculate the boundary term by importance sampling paths in the boundary sample space. Reparameterization methods, on the other hand, aim to avoid explicitly sampling boundary paths. They reparameterize the integral domain by tracing auxiliary rays such that the geometric discontinuities remain static when scene parameters change. Then, automatic differentiation techniques can be applied to obtain the physically correct gradients.

The main drawbacks of physics-based differentiable rendering techniques are the relatively long time required to obtain a low-variance estimate of the gradients and the significantly more complex theory compared to other local illumination-based techniques.

The volume-rendering-based method, NeRF [5], is such a technique that ignores scattering in participating media, retaining only emission and absorption. NeRF-like methods represent scenes using a volume function in the form of neural networks and render this volume using traditional volume rendering methods, except that they now obtain samples through neural networks. The original NeRF exhibits a series of inefficiencies, such as slow rendering speeds that also hinder the training process [17] and poor generalization to new scenes [18]. Many algorithms have been proposed to address these inefficiencies.

Inspired by the idea of representing scenes using volumes, 3D Gaussian splatting (3DGS) [6] has been proposed, which uses a discrete set of 3D Gaussians as base primitives. By avoiding queries to a neural network and leveraging the advantageous projection properties of 3D Gaussians, this technique achieves much higher rendering frame rates, which also accelerates the training process a lot. The main shortcomings of 3DGS include the large storage space required [19] and the difficulty in converting the discrete scene representation into more editable meshes [20].

This paper is organized as follows: In Section 2, classification based on the primary rendering algorithms employed is presented. Next, the physics-based differentiable rendering methods, NeRF-like methods, and methods based on 3DGS are introduced and analyzed in Section 3, Section 4 and Section 5, respectively. In Section 7, the conclusions and open research problems are presented.

2. Algorithms

From the perspective of the primary rendering algorithms used, we provide a brief taxonomy of existing differentiable rendering methods, as illustrated in Figure 1. Note that we provide only a coarse taxonomy here, which is sufficient for the topics covered in this paper, i.e., physics-based, NeRF-based, and 3DGS-based differentiable rendering. Moreover, we list the representative works in these categories in Figure 2.

Besides NeRF-like methods, there are many other approaches included in the "neural-network-based" branch in Figure 1, such as [21–24], which are based on voxel or point

cloud representations. Under the "rasterization-based" branch, there are also many other techniques that are based on traditional rasterizers, such as [25,26]. A survey of these methods goes beyond the scope of this paper, as well.

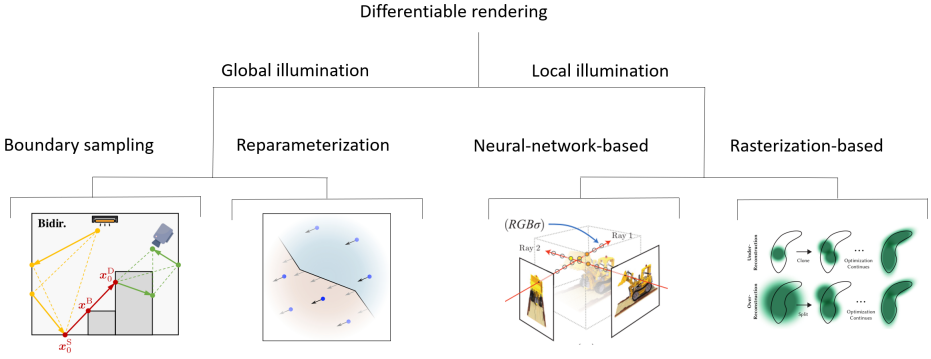


Figure 1. A brief taxonomy of existing differentiable rendering methods. Figures come from [5,6,10,11].

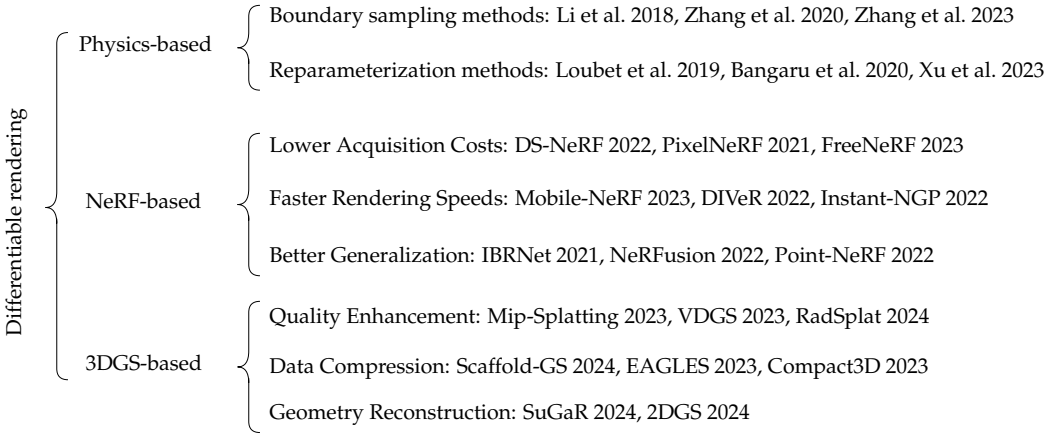


Figure 2. Selected representative works in the area of differentiable rendering [9–11,14–16,18–20,27–40].

We start with the most elaborate formula used in primary rendering algorithms and classify the review topics covered in this paper based on how they relate to this formula. The formula is the equation of transfer that governs the behavior of light in participating media containing surfaces:

$$L_i(p,\omega) = \int_0^D T_r(p' \to p)\sigma_s(p') \int_{S^2} P(p',\omega',\omega)L_i(p',\omega')\,d\omega' dt' + L_s(p,\omega) \tag{1}$$

where $L_i(p,\omega)$ is the incoming radiance at point p in direction ω, T_r is the beam transmittance, which is the fraction of radiance transmitted between two points, D is the distance from point p to the medium's boundary in the direction of $-\omega$, σ_s is the medium's scattering coefficient, P is the phase function, which describes the angular distribution of scattered radiation at a point, and L_s is the source term defined by:

$$\begin{aligned} L_s(p,\omega) = \int_0^D T_r(p' \to p)\sigma_a(p')L_e^v(p',\omega)\,dt' + \\ T_r(p_0 \to p)[\int_{S^2} f(p_0,\omega',\omega)L_i(p_0,\omega')\,d\omega' + L_e^s(p_0,\omega)] \end{aligned} \tag{2}$$

where σ_a is the absorption coefficient, L_e^v is the medium's radiant emission, p_0 is the intersection point where the ray originating from point p and traveling in direction $-\omega$ meets the medium's boundary, f is the cosine-weighted BSDF, and L_e^s is the interfacially emitted radiance.

The formulas above take both volume scattering and surface interactions into account and are the basis of the physics-based differentiable rendering techniques that will be reviewed in Section 3. If we ignore the contributions from surface interactions and scattering in participating media, the resulting volume rendering formula is:

$$L_i(p,\omega) = \int T_r(p' \to p) L_e(p', -\omega)\, dt' \tag{3}$$

which forms the basis of NeRF-like methods and will be reviewed in Section 4. If the scene representation we use is no longer volumetric but is instead based on discrete points, then the aforementioned formula can be rewritten as follows:

$$L_i(p,\omega) = \sum_{i=1}^{n} T_r(p_i \to p) L_e(p_i, -\omega) \tag{4}$$

We can further rewrite the above formula by utilizing the property of the beam transmittance term T_r:

$$T_r(p \to p'') = T_r(p \to p') T_r(p' \to p'') \tag{5}$$

Then we can obtain:

$$\begin{aligned} L_i(p,\omega) &= \sum_{i=1}^{n} \prod_{j=1}^{i} T_r(p_j \to p_{j-1}) L_e(p_i, -\omega) \\ &= \alpha_1 \cdot (L_e(p_1, -\omega) + \alpha_2 \cdot (L_e(p_2, -\omega) + \cdots + \alpha_n \cdot L_e(p_n, -\omega))) \end{aligned} \tag{6}$$

where we replace the transmittance term $T_r(p_j \to p_{j-1})$ with the alpha blending coefficients α_j. Then we obtain the formula utilized by 3DGS methods, which will be reviewed in Section 5.

3. Physics-Based Differentiable Rendering

Physics-based forward rendering aims to synthesize photorealistic images by simulating light transport in a manner consistent with physical laws using Equation (1). In contrast, physics-based differentiable rendering aims to propagate the derivatives of image pixels back to the scene parameters, which is a challenging problem due to the high-order discontinuities inherent in the scene function. With the gradients of these scene parameters, one of the most important applications of PBDR is inverse rendering, i.e., reconstructing geometries, materials, or emitters from a set of input photographs, as illustrated in Figure 3.

Note that there are several works [12,13] using the full form of Equation (1), but the main difficulties encountered in PBDR arise from the rendering part that involves geometric surfaces. The rendering part involving participating media is fully continuous and thus can be differentiated easily. Therefore, we will focus on scenes comprised entirely of surfaces. In the following subsections, we will first briefly introduce the theory behind physics-based rendering. Then, we will discuss two categories of methods in PBDR, which are classified based on their approach to handling discontinuities: boundary sampling methods and reparameterization methods.

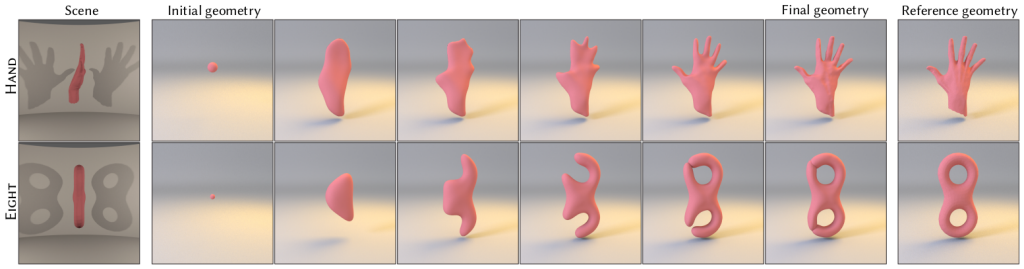


Figure 3. Reconstruction of geometry from one photograph by [14].

3.1. Physics-Based Rendering Preliminaries

Since the main difficulties in PBDR arise from geometric surfaces rather than the participating media, we will focus exclusively on scenes comprised entirely of surfaces. Adding participating media back is straightforward [12,13], as the corresponding rendering part is continuous. Ignoring the participating media term in Equation (1) gives the rendering equation:

$$L_o(p,\omega) = \int_{S^2} f(p,\omega',\omega) L_i(p,\omega')\, d\omega' + L_e^s(p,\omega) \tag{7}$$

which describes the local behavior of light, such as the reflection or transmission, at a point on the geometric surface. The above equation can be solved by iteratively replacing $L_i(p,\omega')$ in the right-hand side with the function in the left-hand side. The final solution can be expressed as a Neumann series: $L_i = \sum_{N=1}^{\infty} P_N$ where P_N stand for the radiance contribution from paths of length N:

$$P_N(p,\omega_0) = \int_{S^2\times\cdots\times S^2} F(p,\omega_0,...,\omega_{N-1}) d\omega_{N-1}\cdots d\omega_1 \tag{8}$$

The resulting radiance function L_i is then convolved with a pixel reconstruction filter function to obtain the final intensity of pixel j:

$$I_j = \iint h_j(x,y) L_i(x,y) dxdy \tag{9}$$

The rendering equation can also be expressed as a surface form or an integral on object surfaces:

$$L(p,p') = L_e(p,p') + \int_A f(p,p',p'') L(p',p'') G(p',p'') dA(p'') \tag{10}$$

where $G(\cdot,\cdot)$ is the geometric term. Expanding the formula again, we obtain the widely used path integral:

$$P_N(p,p_0) = \int_{A\times\cdots\times A} F(p,p_0,...,p_{N-1}) dA_{p_{N-1}}\cdots dA_{p_1} \tag{11}$$

Given the physics-based rendering formulas above, we still need a numerical integration method to calculate the corresponding integrals. Since these integrals tend to be very high-dimensional, Monte Carlo methods are preferred over other numerical integration methods. Specifically, Monte Carlo methods use random numbers to evaluate the integral:

$$I = \int_\Omega f(x)\, d\mu(x) \tag{12}$$

Employing the Monte Carlo estimator, we have:

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)} \tag{13}$$

Note that the points $X_1,...,X_N$ need to be sampled according to the probability density function $p(x)$. In practice, to make the Monte Carlo estimator converge as quickly as possible, we need to match the shape of $p(x)$ with that of $f(x)$ as closely as possible.

With these tools at hand, the next task is to calculate the corresponding differentials. However, automatic differentiation techniques cannot be applied directly due to the discontinuities in the integrand of Equations (8), (9) and (11). Based on how the discontinuities in these high-dimensional integrals are handled, existing PBDR methods can be classified into two categories: boundary sampling methods and reparameterization methods, which will be reviewed in the following subsections.

When comparing the performance of two PBDR methods, the commonly used metrics are the variance of the Monte Carlo estimators and the rendering time in seconds; notably,

there are generally no common datasets used in the field of PBDR, as the scenes for comparison are typically built by the authors themselves.

3.2. Boundary Sampling Methods

Li et al. [9] presented the first physics-based differentiable rendering technique for scenes composed of triangle meshes using an edge sampling method. They realized that the automatic differentiation technique cannot be used directly to compute the gradient of pixel intensity resulting from global illumination techniques such as path tracing due to the discontinuity of the integrand. The Heaviside step function is utilized to partition the discontinuous integrand into many small parts, ensuring that the corresponding function is continuous within each small domain. Then, the pixel intensity I can be written as:

$$I = \iint \sum_i \theta(\alpha_i(x,y)) f_i(x,y)\, dx\, dy \tag{14}$$

where $f_i(x,y)$ is the scene function, $\theta(\cdot)$ is the Heaviside step function, $\alpha_i(x,y)$ is the edge equation corresponding to the discontinuity's location in the scene function, and the summation is over the subdivided small domains.

Recall that the derivative of the Heaviside step function is the Dirac delta function, so the derivative of pixel intensity I with respect to scene parameter π can be written as:

$$\frac{\partial I}{\partial \pi} = \iint \sum_i \theta_i(x,y) \frac{\partial}{\partial \pi} f_i(x,y) + \delta(\alpha_i(x,y)) \nabla \alpha_i(x,y) f_i(x,y)\, dx\, dy \tag{15}$$

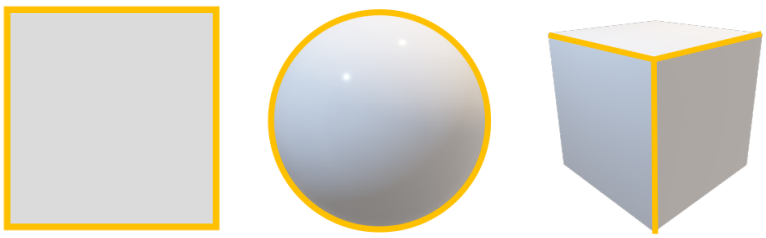
where $\delta(\cdot)$ is the Dirac delta function.

A 2D integral containing the Dirac delta function in the integrand is actually a 1D integral over the domains where the Dirac delta function has non-zero values, so the first physically correct formula for the derivative of the pixel intensity is obtained as follows:

$$\frac{\partial I}{\partial \pi} = \iint \frac{\partial}{\partial \pi} f(x,y)\, dx\, dy + \sum_i \int \frac{\nabla \alpha_i(x,y)}{\|\nabla_{x,y} \alpha_i(x,y)\|} f_i(x,y)\, dt \tag{16}$$

where the second integral is over the triangle edges corresponding to the discontinuity's location in the scene function.

For scenes composed of triangle meshes, the edges that cause discontinuities in the scene function come from three sources, as shown in Figure 4. The boundary edges belong to the topological boundary of the triangle mesh. If the mesh has no topological boundary, i.e., it is closed, then it has no boundary edges that may contribute to the discontinuities of the scene function. The silhouette edges correspond to the occlusion of one mesh over another mesh or self-occlusion, causing the shading to change suddenly from one side of the edge to the other. And the sharp edges result from the discontinuous face normals on the mesh if smooth shading using interpolated normals is disabled.



(**a**) Boundary edges (**b**) Silhouette edges (**c**) Sharp edges

Figure 4. Three sources of triangle edges that cause discontinuities in the scene function by Zhang et al. [12].

Given the physically correct formula Equation (16) for obtaining gradients of pixel colors rendered using global illumination algorithms, the next step is to design a Monte Carlo estimator and an efficient importance sampling scheme to put the theory into practice.

To reduce the variance of the corresponding Monte Carlo estimator as much as possible, the shape of the probability density function needs to closely match the shape of the contribution function.

For the former part of Equation (16), traditional importance sampling methods such as next-event estimation and multiple importance sampling techniques can be applied directly. Recall that arbitrary long light paths need to be evaluated for global illumination algorithms. Therefore, the contribution of the latter part of Equation (16) can be further divided into two parts: one corresponding to the primary visibility, i.e., the first segment of light paths, and the other corresponding to higher-order visibility, i.e., the subsequent segments of light paths.

We can precompute all the triangle edges that may contribute to geometric discontinuities and importance sample them for the primary visibility case. The main challenge comes from the higher-order visibility case.

In this case, we need to importance sample geometric discontinuities viewed from arbitrary shading points in the scene, which is a much more complicated task than the primary visibility case.

Li et al. [9] employs a 6D Hough tree, which takes both vertex positions and normals into account, for the importance sampling task. However, this pioneering importance sampling scheme does not scale well to scenes with high complexity and does not match the shape of the contribution function closely. This results in a Monte Carlo estimator with relatively large variance.

A series of methods have been proposed to improve upon the above technique. Zhang et al. [12] extends the edge sampling method to scenes that contain participating media, supporting arbitrary surface and volumetric configurations. However, their approach remains confined to the framework of sampling edges from given shading points, which is considered inefficient from a modern viewpoint.

The next milestone in the development of boundary sampling techniques is attributed to the work of Zhang et al. [11]. They utilize a so-called transport relation that originated in fluid mechanics to establish a mathematical framework that is used in modern techniques. They also introduce a multi-directional form of the boundary integral, allowing for silhouette paths to be generated "from the middle", which significantly reduces the variance of the gradient estimator with less computation time, as illustrated in Figure 5.

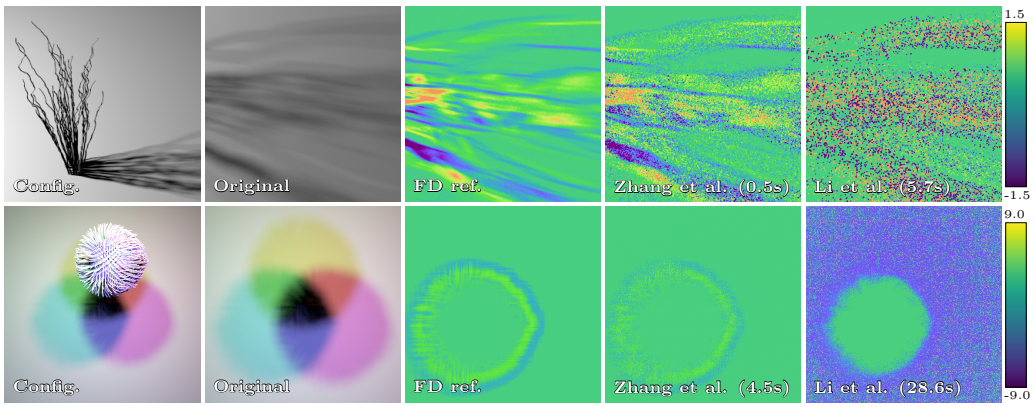


Figure 5. Comparison of the effectiveness between algorithms from Zhang et al. [11] and Li et al. [9] under an equal-sample configuration. Images in the left column visualize the overall scene configurations. The method of Li et al. achieves lower variance with less computation time. The figures come from [11].

Zhang et al. [13] extends the aforementioned method to scenes containing participating media. Yan et al. [41] employ adaptive data structures to guide the sampling process within the boundary sample space. They also propose an edge-sorting algorithm to reorganize the boundary sample space to further improve the sampling efficiency, as illustrated in Figure 6. Zhang et al. [14] re-derive the local formulation of the perimeter and propose the first local formulation of the interior. They also find that the calculation of the boundary term can greatly benefit from information gathered during the interior term's simulation. Specifically, they project the rays that intersect the shape that is being differentiated onto the corresponding geometry boundary. Then, they calculate these projected rays' contributions to the boundary integral to initialize the guiding structure, thereby reducing the overall variance, as illustrated in Figure 7.

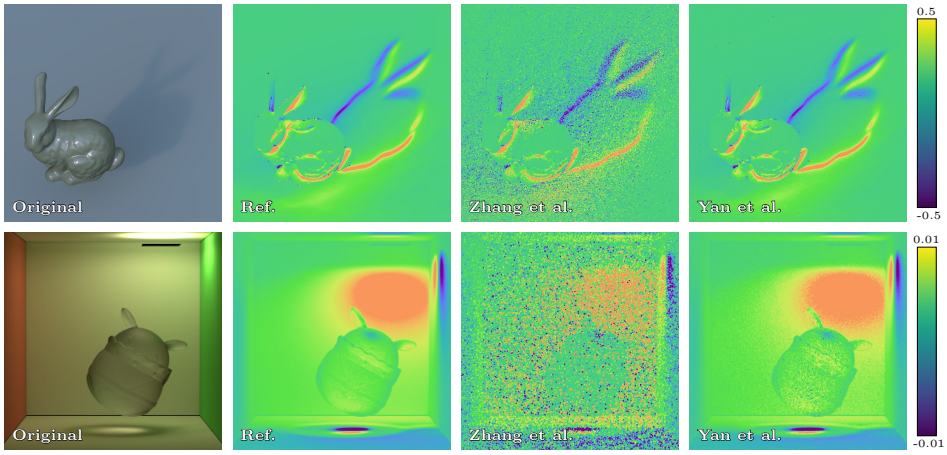


Figure 6. Comparison of the effectiveness between algorithms from Yan et al. [41] and Zhang et al. [11] under an equal-time configuration. The adaptive data structures employed by Yan et al. [41] provide better exploration of the boundary sample space, resulting in lower variance for the gradient estimates. Figures come from [41].

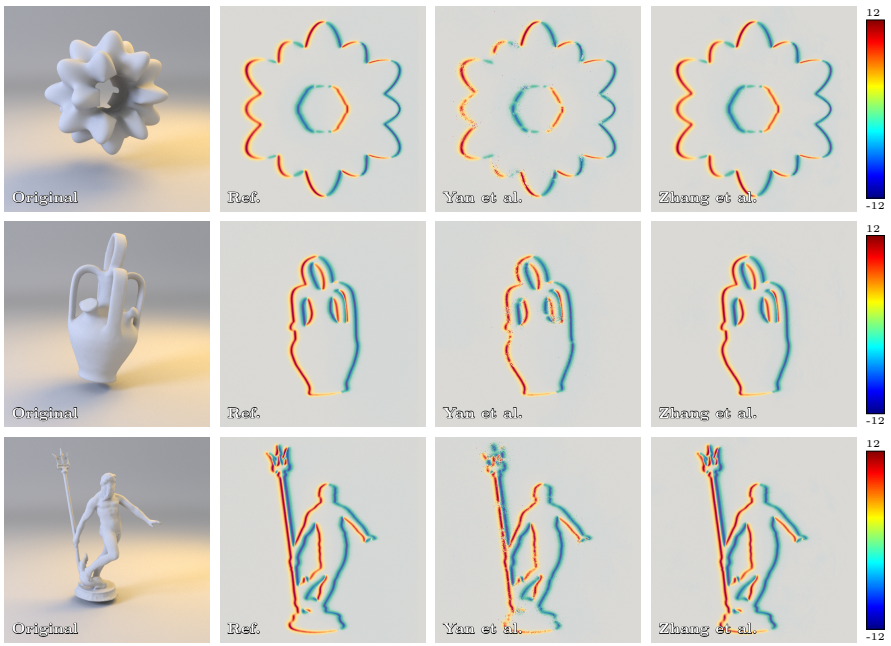


Figure 7. Comparison of the effectiveness between algorithms from Zhang et al. [14] and Yan et al. [41] under an equal-time configuration. The results show that the calculation of the boundary integral can greatly benefit from the information gathered during the simulation of the interior term. Figures come from [14].

3.3. Reparameterization Methods

The main inefficiency of the edge sampling method proposed by Li et al. [9] arises from the challenging task of importance sampling of geometric discontinuities viewed from arbitrary shading points in the scene. Reparameterization methods instead try to avoid sampling these discontinuities explicitly by reparameterizing the integral domain.

Recall that the pixel intensity I is the result of an integral that typically contains discontinuities due to the discontinuous visibility function. And the geometric discontinuities move in the integral domain when the geometric parameter π changes.

The core idea of reparameterization methods can be understood as subdividing the integral domain into small parts such that the scene function is continuous over each part and then reparameterizing the scene function over each subdomain. Note that the boundary of each small integral domain needs to match the moving geometric discontinuities to ensure the validity of the reparameterization. Then, we can rewrite formula as follows:

$$I(\pi) = \sum_i \iint_{\Omega_i(\pi)} f(x, y, \pi)\, dx\, dy \tag{17}$$

where $\Omega_i(\pi)$ represents the subdivided subdomains whose boundaries match the movement of the geometric discontinuities.

Suppose we have a change of the variables' schemes at hand:

$$F_i(\pi) : \Omega_i(\pi_0) \to \Omega_i(\pi); \quad (x_0, y_0) \overset{\pi}{\mapsto} (x, y) \tag{18}$$

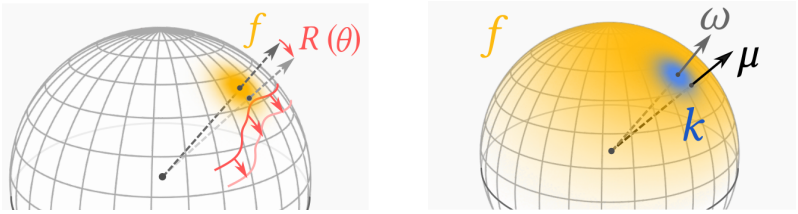
then we have:

$$I(\pi) = \sum_i \iint_{\Omega_i(\pi_0)} f(x(x_0, y_0, \pi), y(x_0, y_0, \pi), \pi) \left| \frac{\partial(x, y)}{\partial(x_0, y_0)} \right| dx_0\, dy_0 \tag{19}$$

Although the discontinuities still exist in the integrand, they no longer move when the geometric parameter π changes. Therefore, we can now apply automatic differentiation techniques to compute the derivative of pixel intensities.

In practice, we need to trace many rays of arbitrary lengths to estimate the derivative. This would result in a huge computational overhead if we explicitly subdivide the integral domain for each ray segment. Therefore, all existing reparameterization methods propose their own reparameterization schemes without knowing the boundaries of the subdomains in Equation (19).

The first work in a series of reparameterization studies was proposed by Loubet et al. [15]. They found that the movement of the geometric discontinuities in a small spherical integral domain can be well-approximated with a simple spherical rotation transform, as illustrated in Figure 8. For scene functions with large supports, they convolved the function with a convolution kernel and used spherical rotations to approximate the movement of geometric discontinuities within the small support of the kernel function at the cost of increased variance.



(**a**) Spherical rotation of directions (**b**) Convolutions if large support

Figure 8. (**a**) Use of spherical rotation to approximate the movement of geometric discontinuities within a small spherical domain. (**b**) Use of spherical convolution to transform the large function support case into a small support case by Loubet et al. [15].

Given the small domain to reparameterize the convolved scene function, auxiliary rays need to be emitted to determine the suitable changes of variables. The raw results obtained through their reparameterization scheme exhibit high variance. Control variates and partially correlated pairs of paths are used to reduce the overall variance.

The main drawback of this reparameterization method is that the final Monte Carlo estimator is biased due to the approximation the authors used.

The first unbiased reparameterization method for physics-based differentiable rendering was proposed by Bangaru et al. [10]. To better understand their work, we need to go further with Equation (19):

$$\frac{\partial I(\pi)}{\partial \pi} = \sum_i \iint_{\Omega_i(\pi_0)} \frac{\partial f(x_0, y_0, \pi)}{\partial \pi} + \frac{\partial f(x, y, \pi_0)}{\partial (x, y)} \cdot \frac{\partial (x, y)}{\partial \pi} + f(x_0, y_0, \pi_0) \frac{\partial}{\partial \pi} \left| \frac{\partial (x, y)}{\partial (x_0, y_0)} \right| dx_0\, dy_0 \tag{20}$$

where we assume that the absolute value of the Jacobian determinant in Equation (19) is 1 when evaluated at π_0, which is true in existing reparameterization methods.

By using the result in Magnus et al. [42], we can rewrite the derivative of the above Jacobian determinant as the divergence of the underlying mapping:

$$\frac{\partial I(\pi)}{\partial \pi} = \sum_i \iint_{\Omega_i(\pi_0)} \frac{\partial f(x_0, y_0, \pi)}{\partial \pi} + \nabla_{x,y} f(x, y, \pi_0) \cdot \frac{\partial (x, y)}{\partial \pi} + f(x_0, y_0, \pi_0) \nabla \cdot \frac{\partial (x, y)}{\partial \pi}\, dx_0\, dy_0 \tag{21}$$

where $\boldsymbol{V} := \frac{\partial (x,y)}{\partial \pi}$ is referred to as the warp field in the work of Bangaru et al. [10].

Instead of directly seeking suitable changes of variables, they choose to estimate the warp field using a Monte Carlo estimator. Recall that to ensure the validity of the reparameterization, the boundaries of subdomains $\Omega_i(\pi)$ need to match the movement of the geometric discontinuities. The same requirement applies to the induced warp field $\boldsymbol{V}$.

Applying automatic differentiation techniques directly to the ray tracing process can generate a warp field $\boldsymbol{V}^{(direct)}$. However, this warp field does not satisfy the above requirement in the case of silhouette edges. Imagine one object blocking another; the velocities of the movements of the projections of the silhouette edges are actually determined by the occluder. However, the warp field $\boldsymbol{V}^{(direct)}$ obtained through the ray tracing process is determined entirely by the object being blocked.

To achieve valid reparameterization, an additional operation is needed to transform $\boldsymbol{V}^{(direct)}$ into a warp field that matches the movement of the silhouette edges.

Bangaru et al. [10] convolves the warp field $\boldsymbol{V}^{(direct)}$ obtained through the ray tracing process with a weight function w, which is estimated by tracing additional rays, to obtain the final warp field:

$$\boldsymbol{V}^{(filtered)}(\omega) = \frac{\int_{\Omega'} w(\omega, \omega') V^{(direct)}(\omega')\, d\omega'}{\int_{\Omega'} w(\omega, \omega')\, d\omega'} \tag{22}$$

where the weight function w has the property:

$$\lim_{\omega^{(b)} \to \partial\Omega} \frac{w(\omega^{(b)}, \omega)}{\int_{\Omega'} w(\omega^{(b)}, \omega')\, d\omega'} = \delta(|\omega^{(b)} - \omega|) \tag{23}$$

where $\delta(\cdot)$ is the Dirac delta function. After the convolution, we obtain a valid warp field, and Equation (21) can be used to calculate the derivatives of scene parameters in an unbiased way, as illustrated in Figure 9. Xu et al. [16] extend the above method to more advanced primary Monte Carlo rendering techniques, such as bidirectional path

tracing, based on a new formulation for reparameterized differential path integrals. They also introduce a new distance function to further reduce the variance of the final gradient estimator, as illustrated in Figure 10.

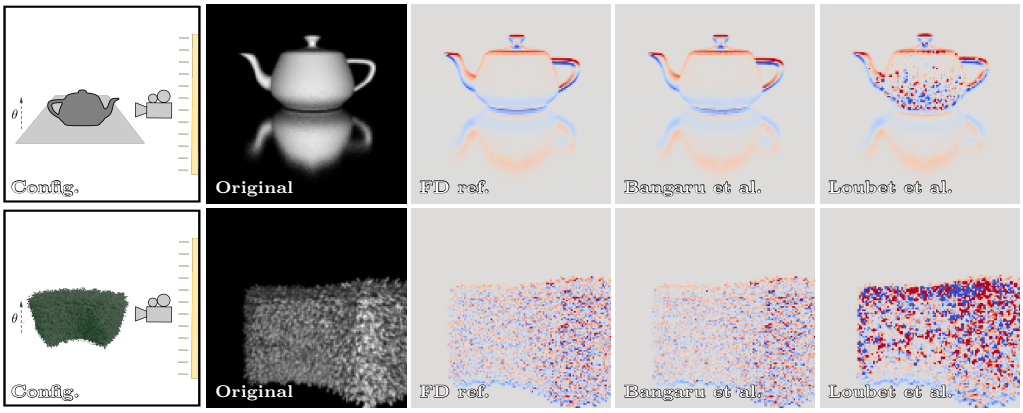


Figure 9. Comparison of the effectiveness between algorithms from Bangaru et al. [10] and Loubet et al. [15]. Note that the results from Loubet et al. [15] exhibit high bias, while those from Bangaru et al. [10] closely match the reference. Figures come from [10].

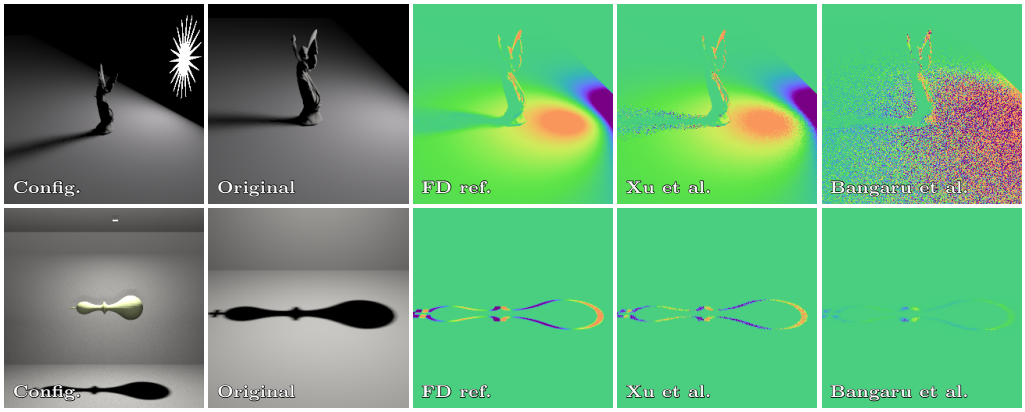


Figure 10. Comparison of the effectiveness between algorithms from Xu et al. [16] and Bangaru et al. [10] under an equal-sample configuration. The results from Xu et al. [16] exhibit lower variance due to the new formulation for reparameterized differential path integrals and the new distance function proposed. Figures come from [16].

4. Neural Radiance Field

Compared to physics-based differentiable rendering techniques, which simulate light transport between scene surfaces for every detail, neural radiance fields (NeRFs) [5] use an approximate method to model the real world. Specifically, NeRF-like methods model scenes primarily composed of 2D surfaces with participating media only. In these methods, the rendering integral is completely continuous, so the hard-to-handle discontinuities encountered in PBDR methods disappear. Due to the approximations used in NeRF-like methods, the single-object reconstruction results do not achieve the same level of precision as PBDR methods. However, thanks to these approximations, NeRF-like methods can reconstruct scenes with many complex geometries or even wild scenes using a few photographs—something that current PBDR methods cannot accomplish.

The core idea of NeRFs is to employ a neural network to implicitly represent the radiance field of a three-dimensional scene. Specifically, NeRFs utilize a multilayer perceptron (MLP) to map a pair of a position and direction to a color and volume density corresponding to the emission term and absorption term in the participating media. Points along the ray that originate from sensors are then sampled to calculate the rendering integral, resulting in the final pixel intensity.

NeRFs have successfully modeled scenes as radiance fields, enabling the synthesis of high-quality images from new perspectives for scenes with complex geometries and appearances. These representations have been rapidly extended and have been applied to numerous graphics and vision tasks, including generative modeling, surface reconstruction, appearance editing, and motion capture. These advancements facilitate applications in various fields such as robotics, autonomous navigation, scene inpainting [43], and virtual/augmented reality.

In the following subsections, we will first briefly introduce the theory behind NeRFs. Then, we will successively discuss the improvements made since the original NeRF work. Note that there is an enormous amount of research focusing on improving NeRF in various aspects such as view synthesis under fuzzy input conditions [44], thin structures [45], reconstruction of outdoor scenes [46], drone-captured scenes [47], large-scale scenes [48], and RGB-D-captured scenes [49]. Since there are already thorough surveys of NeRF-based methods [50–53], we will focus primarily on PBDR and only briefly review NeRF-based methods from three aspects for completeness: lowering acquisition costs, increasing rendering speeds, and enhancing generalization capabilities.

4.1. NeRF Preliminaries

The original NeRF represents a scene as a 5D vector-valued function described by the following mapping:

$$(c, \sigma) = F_\Theta(\boldsymbol{x}, \boldsymbol{d}) \tag{24}$$

Here, $\boldsymbol{x} = (x, y, z)$ represents the 3D coordinates of a point in the scene, and $\boldsymbol{d} = (\theta, \phi)$ is the camera's viewing direction. The function outputs $\boldsymbol{c} = (r, g, b)$, the color emitted by the point $\boldsymbol{x}$ in direction $\boldsymbol{d}$, and σ, which is the volume density and is a quantity related to absorption in the participating media. The function F_Θ is defined by a deep, fully connected neural network known as multilayer perceptron (MLP). Specifically, the volume density σ is only related to the position $\boldsymbol{x}$, whereas the color $\boldsymbol{c}$ is affected by both the position $\boldsymbol{x}$ and the direction $\boldsymbol{d}$. To help the neural network better capture and represent high-frequency details in the scene, the position encoding technique [54] is used to map low-frequency input coordinates to a high-dimensional space. In this way, NeRF is able to generate high-quality and realistic images across different viewing angles by using the following rendering process.

Given a ray passing through an image pixel in the world coordinate space, we use the following integral to calculate the corresponding intensity:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(\boldsymbol{r}(t))c(\boldsymbol{r}(t), \boldsymbol{d})dt \tag{25}$$

where

$$T(t) = exp\left(-\int_{t_n}^{t} \sigma(\boldsymbol{r}(s))ds\right) \tag{26}$$

represents the fraction of energy retained when a photon travels from t_n to t. Then, points along the ray are sampled, and the corresponding color and volume density are obtained through the neural network. The final pixel intensity is calculated using Equation (25). The whole pipeline is illustrated in Figure 11.

When comparing the performance of two NeRF-based methods, commonly used datasets include the DTU dataset [55] (carefully calibrated camera poses), LLFF dataset [56] (handheld cellphone images), NeRF Synthetic dataset [5] (Blender-generated), Redwood dataset [57] (over ten-thousand 3D scans), Mip-NeRF 360 dataset [58] (scenes containing a complex central object), Tanks and Temples dataset [59] (high-quality laser-scanned scenes), ICL-NUIM dataset [60] (synthetic indoor scenes), and ScanNet dataset [61] (over 2.5 million indoor scene views with semantic labels), while commonly used visual quality assessment metrics include PSNR (an approximation of human perception of reconstruction

quality), SSIM [62] (a measure of structural similarity), and LPIPS [63] (a measure based on deep features).

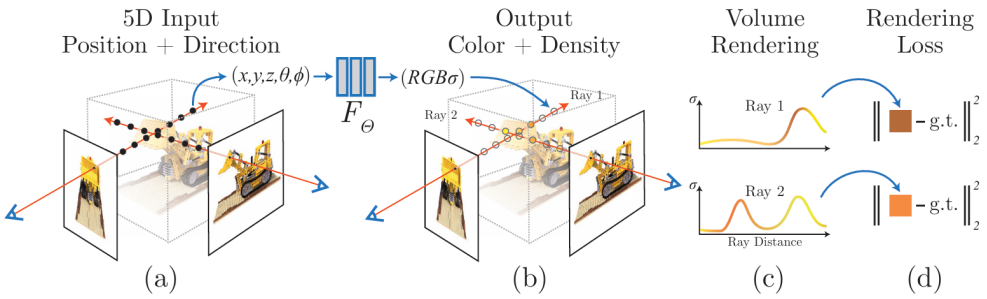


Figure 11. Overview of the NeRF rendering process: (**a**) select a series of sampling points along camera rays, (**b**) output the color and volume density of the sampling points using the underlying neural network, (**c**) calculate individual pixel colors via Equation (25), and (**d**) compare the predicted image with the reference image and optimize the parameters of the underlying neural network. Figures come from [5].

4.2. NeRF with Lower Acquisition Costs

Although a NeRF can synthesize high-quality images, it requires a large number of images to learn the 3D structure and lighting information of a scene. If the number of input views is insufficient, the NeRF may fail to fully capture the complex details and variations of the scene, resulting in a decrease in the quality of the synthesized images.

DS-NeRF [27] achieves enhanced training efficiency and improved rendering quality in sparse view settings by introducing a novel loss function for learning radiance fields that takes advantage of readily available depth supervision. This approach utilizes the sparse 3D points generated by structure-from-motion (SFM) as a source of "free" depth information. During training, this additional supervision signal helps the model learn more accurate scene geometries. A novel loss function is introduced to align the distribution of a ray's terminating depth with specified 3D keypoints, incorporating depth uncertainty to anchor the learning process to the 3D geometry. The comparison results in Figures 12 and 13 and Table 1 demonstrate that DS-NeRF delivers higher-quality view synthesis with fewer input views. Note that the compared methods, MetaNeRF [64] and PixelNeRF [28], both use data-driven priors recovered from a domain of training scenes to fill in missing information from test scenes, allowing them to use fewer images to recover the scene.

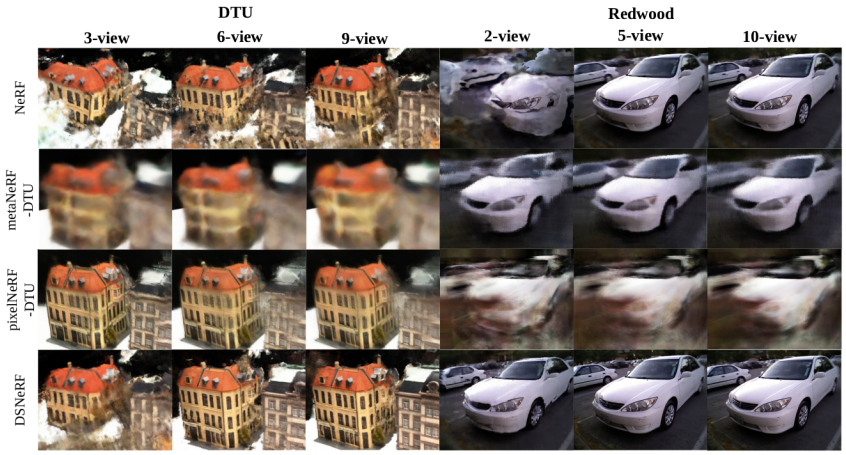


Figure 12. Comparison results of four methods—NeRF [5], MetaNeRF [64], PixelNeRF [28], and DS-NeRF [27]—on the DTU [55] and Redwood [57] datasets with sparse input views. Note that DS-NeRF performs the best, except on DTU when given 3 views. Figures come from [27].

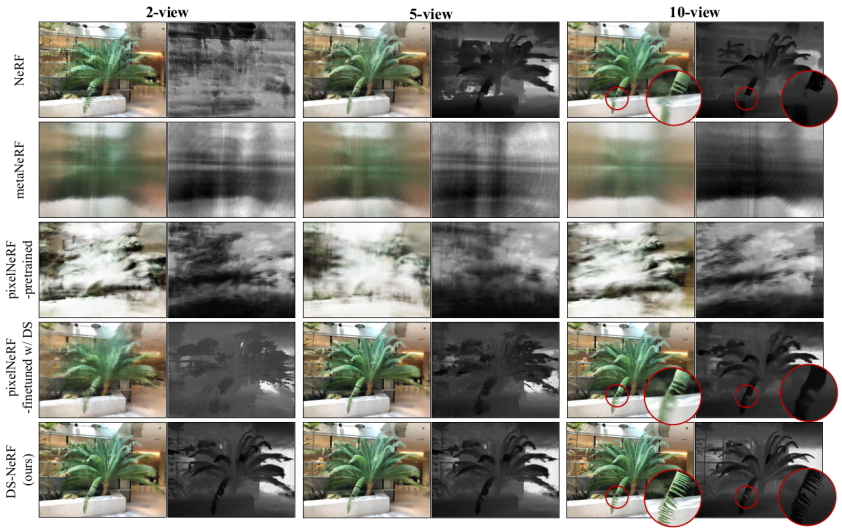


Figure 13. Comparison results of four methods—NeRF [5], MetaNeRF [64], PixelNeRF [28], and DS-NeRF [27]—on the NeRF Synthetic dataset [5] with sparse input views. Note that DS-NeRF performs the best in all cases. Figures come from [27].

RegNeRF [65] finds that errors in estimated scene geometry and the divergent behavior at the start of training lead to lower rendering quality with sparse input views. The authors then propose to regularize the geometry and appearance of patches rendered from unobserved viewpoints to improve rendering quality. Additionally, a normalizing flow model is used to regularize the color of unobserved viewpoints, which also plays an important role in the final rendering results.

FreeNeRF [29] finds that frequency plays an important role in NeRF's training under the few-shot setting. This paper introduces an innovative frequency regularization technique that significantly improves NeRF's performance with fewer training views. The core of the technique lies in two key regularization strategies: First, frequency regularization controls the range of visible frequencies in NeRF's input through a simple, linearly increasing frequency mask. This prevents overfitting to high-frequency signals during the early stages of training. Second, occlusion regularization mitigates the "floating artifacts" in novel view synthesis by penalizing high-density regions near the camera, effectively reducing these artifacts. The comparison results in Figures 14 and 15 and Table 2 demonstrate that FreeNeRF consistently outperforms state-of-the-art methods across multiple datasets. Moreover, some methods, like NeuralLift-360 [66], Sherf [67], and Dip-NeRF [68], combine NeRF with additional information, such as depths, to reconstruct scenes with few input views.

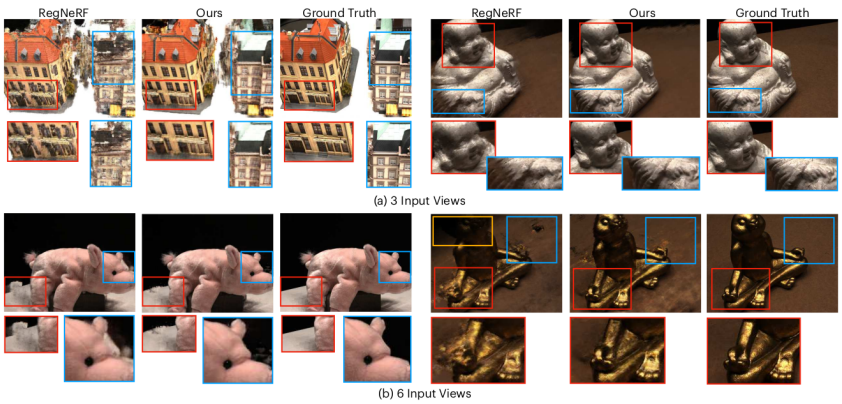


Figure 14. Comparison results between FreeNeRF [29] and RegNeRF [65] on the DTU [55] dataset. Note that FreeNeRF performs better than RegNeRF on fine-grained details. Figures come from [29].

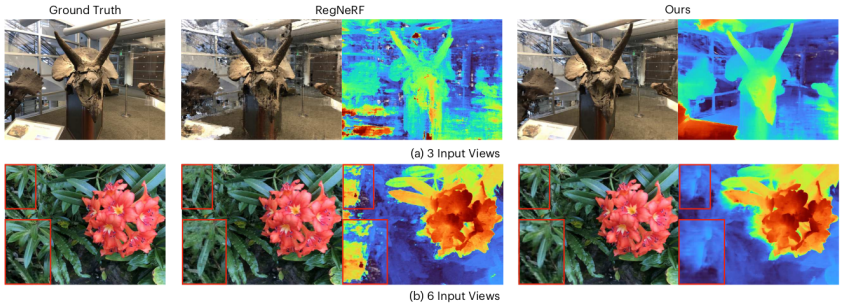


Figure 15. Comparison results between FreeNeRF [29] and RegNeRF [65] on the LLFF [56] dataset. Note that FreeNeRF reconstructs less-noisy occupancy fields with fewer floaters. Figures come from [29].

Table 1. Quantitative comparison of DS-NeRF [27] with previous methods.

Number of Input Views	PSNR ↑			SSIM [62] ↑			LPIPS [63] ↓		
	2	5	10	2	5	10	2	5	10
DTU [55]									
NeRF	9.9	18.6	22.1	0.37	0.72	**0.82**	0.62	0.35	0.26
MetaNeRF	18.2	18.8	20.2	0.60	0.61	0.67	0.40	0.41	0.35
PixelNeRF	**19.3**	20.4	21.1	**0.70**	0.73	0.76	**0.39**	0.36	0.34
DS-NeRF	16.9	**20.6**	**22.3**	0.57	**0.75**	0.81	0.45	**0.29**	**0.24**
Redwood-3dscan [57]									
NeRF	10.5	22.4	23.4	0.38	0.75	0.82	0.51	0.45	0.45
MetaNeRF	14.3	14.6	15.1	0.37	0.39	0.40	0.76	0.76	0.75
PixelNeRF	12.7	12.9	12.8	0.43	0.47	0.50	0.76	0.75	0.70
DS-NeRF	**20.3**	**23.4**	**23.9**	**0.73**	**0.77**	**0.84**	**0.36**	**0.35**	**0.28**
NeRF Synthetic [5]									
NeRF	13.5	18.2	22.5	0.39	0.57	0.67	0.56	0.50	0.52
MetaNeRF	13.1	13.8	14.3	0.43	0.45	0.46	0.89	0.88	0.87
PixelNeRF	18.2	22.0	24.1	0.56	0.59	0.63	0.53	0.53	0.41
DS-NeRF	**20.0**	**22.6**	**24.9**	**0.67**	**0.69**	**0.72**	**0.39**	**0.35**	**0.34**

Table 2. Quantitative comparison of FreeNeRF [29] with previous methods.

Number of Input Views	Full-Image PSNR ↑			Full-Image SSIM [62] ↑			LPIPS [63] ↓		
	3	6	9	3	6	9	3	6	9
DTU [55]									
PixelNeRF	17.38	21.52	21.67	0.548	0.670	0.680	-	-	-
RegNeRF	15.33	19.10	22.30	0.621	0.757	0.823	-	-	-
FreeNeRF	**18.02**	**22.39**	**24.20**	**0.680**	**0.779**	**0.833**	-	-	-
LLFF [56]									
PixelNeRF	16.17	17.03	18.92	0.438	0.473	0.535	0.512	0.477	0.430
RegNeRF	19.08	23.10	24.86	0.587	0.760	0.820	0.336	0.206	0.161
FreeNeRF	**19.63**	**23.73**	**25.13**	**0.612**	**0.779**	**0.827**	**0.308**	**0.195**	**0.160**

4.3. NeRF with Faster Rendering Speeds

Although NeRF can achieve photorealistic view synthesis, it requires frequent evaluation of the neural network at all point samples along each ray at runtime, which limits its capability for real-time rendering applications.

SNeRG [69] precomputes and stores the trained NeRF in a sparse voxel grid with learned feature vectors, enabling real-time rendering on commodity hardware. This method

is 3000 times faster than the original implementation, significantly enhancing its performance. Mobile-NeRF [30] tries to combine NeRF with the traditional polygon rasterization pipeline to increase rendering speed. The method utilizes a set of polygons with textures representing binary opacities and feature vectors to model the scene. The output of the rasterization pipeline is pixels representing features, which are then interpreted by a lightweight MLP running in a GLSL fragment shader to render images. This approach not only maintains high-quality image output but also significantly increases rendering speed and reduces memory requirements, enabling real-time rendering on a wide range of computing platforms, including mobile phones. The comparison results in Tables 3 and 4 demonstrate that Mobile-NeRF is around 10 times faster than SNeRG.

DIVeR [31] tries to accelerate the rendering process by using deterministic rather than stochastic estimates of the volume rendering integral. This method involves jointly optimizing a feature voxel grid and a decoder MLP to reconstruct the scene. Each ray from the camera is segmented into intervals corresponding to each voxel. Components of the volume rendering integral are decoded by an MLP for each interval to generate density and color. As a result, DIVeR outperforms previous methods in terms of quality, especially for thin translucent structures, while maintaining comparable rendering speed. Moreover, Instant-NGP [32] proposes a learned parametric multiresolution hash encoding to greatly reduce training time and uses an occupancy grid to accelerate inference speed.

Table 3. Comparison of frames per second (FPS) between Mobile-NeRF [30] and SNeRG [69].

Dataset	**NeRF Synthetic [5]**		**LLFF [56]**		**Mip-NeRF 360 [58]**
Method	**Mobile-NeRF**	**SNeRG**	**Mobile-NeRF**	**SNeRG**	**Mobile-NeRF**
iPhoneXS	**55.89**	-	**27.19**	-	22.20
Pixel 3	**37.14**	-	**12.40**	-	9.24
Surface Pro 6	**77.40**	-	**21.51**	-	19.44
Chromebook	**53.67**	22.62	**19.44**	7.85	15.28
Gaming laptop	**178.26**	8.30	**57.72**	3.63	55.32

Table 4. Quantitative comparison between Mobile-NeRF [30] and SNeRG [69].

	NeRF Synthetic [5]			**LLFF [56]**			**Mip-NeRF 360 [58]**		
	PSNR ↑	**SSIM** ↑	**LPIPS** ↓	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓
SNeRG	30.38	**0.950**	**0.050**	25.63	0.818	0.183	-	-	-
Mobile-NeRF	**30.90**	0.947	0.062	**25.91**	**0.825**	**0.183**	21.95	0.470	0.470

4.4. NeRF with Better Generalization

The original NeRF requires several hours of training for each new scene. In contrast, a generalizable NeRF can render multiple scenes directly using a pre-trained neural network without the need for retraining.

IBRNet [18] synthesizes novel views of complex scenes by interpolating a sparse set of nearby views: a common strategy in the field of image-based rendering. It employs a neural network to learn a generic view interpolation function that generalizes to new scenes. MVSNeRF [70] utilizes a higher-dimensional cost volume to represent the scene, resulting in faster processing and improved generalization. NeRFusion [33] combines the advantages of NeRF- and TSDF-based fusion, achieving state-of-the-art quality for both large-scale indoor scenes and small-scale object scenes. It predicts per-frame local radiance fields from an input image sequence via direct network inference and fuses these into a global sparse scene representation in real-time, which can be further fine-tuned. NeRFusion outperforms baseline NeRF [5], IBRNet [18], and MVSNeRF [70] on the NeRF Synthetic [5] and DTU [55] datasets, as shown in Table 5.

Point-NeRF [34] combines volumetric neural rendering and deep multi-view stereo by using neural 3D point clouds to make NeRFs generalizable to new scenes. Point-NeRF

can use the result of direct inference from a deep network pre-trained across scenes to produce an initial neural point cloud. This neural point cloud can then be rendered with a ray-marching-based pipeline or further fine-tuned to improve visual quality. Point-NeRF outperforms IBRNet [18] and MVSNeRF [70] on the NeRF Synthetic [5] and DTU [55] datasets, as shown in Figure 16.

Table 5. Quantitative comparison between NeRFusion [33] and previous methods.

Method	Settings	PSNR ↑	SSIM ↑	LPIPS ↓
IBRNet	NeRF Synthetic [5]	**25.51**	0.916	0.100
NeRFusion	No per-scene optimization	25.47	**0.922**	**0.093**
NeRF	NeRF Synthetic [5]	31.01	0.947	0.081
IBRNet	Per-scene optimization	28.19	0.943	0.072
NeRFusion		**31.25**	**0.953**	**0.069**
PixelNeRF		19.31	0.789	0.382
IBRNet	DTU [55]	26.04	0.917	0.190
MVSNeRF	No per-scene optimization	**26.63**	**0.931**	**0.168**
NeRFusion		26.19	0.922	0.177
NeRF		27.01	0.902	0.263
IBRNet	DTU [55]	31.35	0.956	0.131
MVSNeRF	Per-scene optimization	28.50	0.933	0.179
NeRFusion		**31.79**	**0.962**	**0.119**

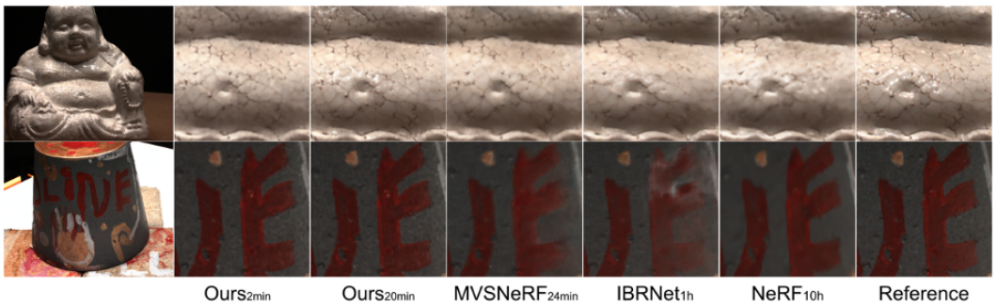


Figure 16. Comparison results for per-scene optimization between Point-NeRF [34] and previous methods on the DTU [55] dataset. Note that Point-NeRF recovers texture details and geometrical structures more accurately than other methods. Figures come from [34].

5. 3D Gaussian Splattting

Despite various improvements made by researchers to address the shortcomings of NeRFs, the neural-network-based implicit representation still struggles to balance training and rendering efficiency with scene reconstruction quality, limiting its further development. To address this issue, 3D Gaussian splatting (3DGS) [6] has recently emerged; 3DGS represents a scene as a collection of 3D Gaussians, each with its own attributes such as position, rotation, scale, opacity, and color. Given a camera ray, the corresponding intensity is calculated as the alpha blending result of the 3D Gaussians intersecting with the ray. The scene is reconstructed by optimizing the attributes of 3D Gaussians so that the rendered images match the input images. Unlike coordinate-based implicit 3D scene representation methods like NeRF, 3DGS draws inspiration from point-based rendering methods [71] and is a purely explicit representation. It is highly parallelized, is capable of converging in approximately 30 min, and achieves real-time rendering at over 30 FPS at 1080p resolution. Similar to NeRF, 3DGS is used in diverse applications such as robotics, autonomous navigation, urban mapping, and virtual/augmented reality. Since there are already thorough survey of 3DGS-based methods [72], we will focus primarily on PBDR and only briefly review 3DGS-based methods from three aspects for completeness. In the following subsections, we will first briefly introduce the theory behind 3DGS. Then,

we will successively discuss the improvements in 3DGS regarding quality enhancement, compression and regularization, and 3D geometry reconstruction.

5.1. 3DGS Preliminaries

A scene is represented as millions of 3D anisotropic balls in 3DGS, with each modeled using a 3D Gaussian distribution:

$$G(X) = e^{-\frac{1}{2}M^T\Sigma^{-1}M} \tag{27}$$

where $M \in \mathbf{R}^3$ is the mean position of the anisotropic ball and Σ is the corresponding covariance. Further, each 3D Gaussian has an opacity α and spherical harmonics parameters c^k (k is the degrees of freedom) for modeling density and view-dependent radiance for each anisotropic ball. For regularizing optimization, the covariance matrix is further decomposed into rotation matrix R and scaling matrix S:

$$\Sigma = RSS^TR^T \tag{28}$$

Given the camera pose, novel view rendering is performed via point splatting to project onto the 2D image plane. At last, for every pixel, the final pixel color can be computed by alpha compositing the opacity and color of all the 3D Gaussians by depth order.

When comparing the performance of two 3DGS-based methods, the commonly used datasets and visual quality assessment metrics are almost the same as those used by NeRF-based methods, as listed in Section 4.1.

5.2. 3DGS with Quality Enhancement

Although 3DGS can render realistic images, there is still room for improving the rendering quality. Due to the mismatch between the signal frequency and the sampling rate, 3DGS produces aliasing when zooming in. To address this issue, Mip-Splatting [35] first adopts a 2D Mip filter inspired by EWA-Splatting [73] to alleviate aliasing. Additionally, Mip-Splatting also limits the sampling frequency. MS 3DGS [74] also addresses the issue of aliasing. It proposes a multi-scale Gaussian splatting representation that selects different scales of 3D Gaussians based on the rendering resolution level. From another perspective, SA-GS [75] proposes a training-free method and maintains scale consistency using a 2D scale-adaptive filter to improve the anti-aliasing performance of 3DGS; it outperforms Mip-Splatting [35] and the original 3DGS [6], as illustrated in Figure 17.

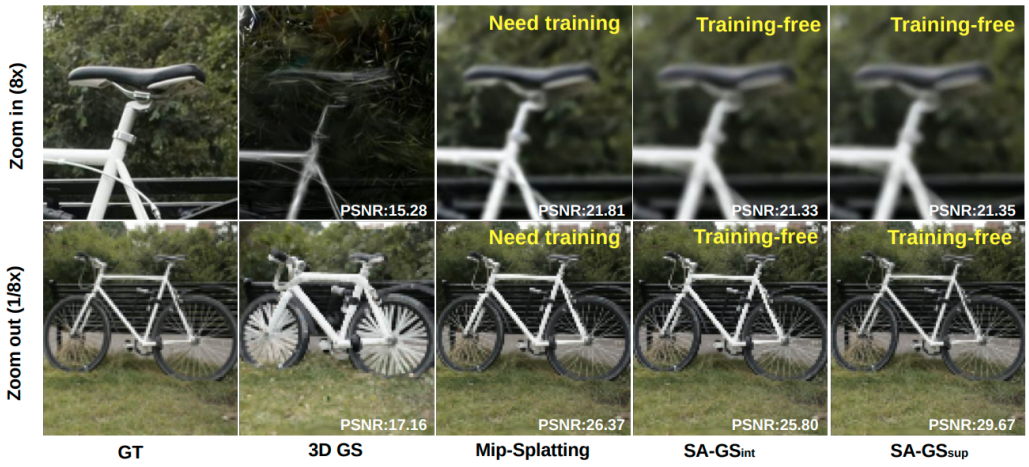


Figure 17. Comparison results between SA-GS [75] and previous methods under zoom-in and zoom-out settings. Note that the rendering results of SA-GS exhibit better anti-aliasing performance and scale consistency compared to other methods. Figures come from [75].

Due to the poor performance of spherical harmonics for fitting high-frequency radiance distributions, some works have proposed improvements based on the intrinsic properties and rendering formula of 3DGS. GaussianShader [76] proposes a simplified shading function on 3D Gaussians to enhance novel view synthesis results in scenes with reflective surfaces. VDGS [36] proposes using a neural network, similar to NeRF, to model the view-dependent radiance and opacity, thereby enhancing 3DGS's capability to model high-frequency information. A visual comparison between VDGS [36] and the original 3DGS [6] using the Tanks and Temples [59] and Mip-NeRF 360 [58] datasets is shown in Figure 18.

Figure 18. Comparison results between VDGS [36] and original 3DGS [6] on Tanks and Temples [59] and Mip-NeRF 360 [58] datasets. Note that VDGS renders fewer artifacts in both datasets compared to original 3DGS. Figures come from [36].

Researchers find that the spatial distribution of 3D Gaussians significantly impacts the final rendering quality. Therefore, some works have improved the mechanisms of 3DGS growing and pruning in vanilla 3DGS to enhance rendering quality. GaussianPro [77] guides the densification of the 3D Gaussians with a proposed progressive propagation strategy. RadSplat [37] develops a ray-contribution-based pruning technique to reduce the overall point count while maintaining photo-realistic rendering quality. LightGS [78] detects Gaussians that have minimal impact on scene reconstruction and employs a process of pruning and recovery, thereby reducing the number of redundant Gaussians while maintaining visual quality. To enhance the rendering quality in non-textured regions such as walls, ceilings, and furniture surfaces, GeoGaussian [79] introduces a novel pipeline to initialize thin Gaussians aligned with the surfaces, as shown in Figure 19, and achieves better novel view synthesis results, as shown in Figure 20.

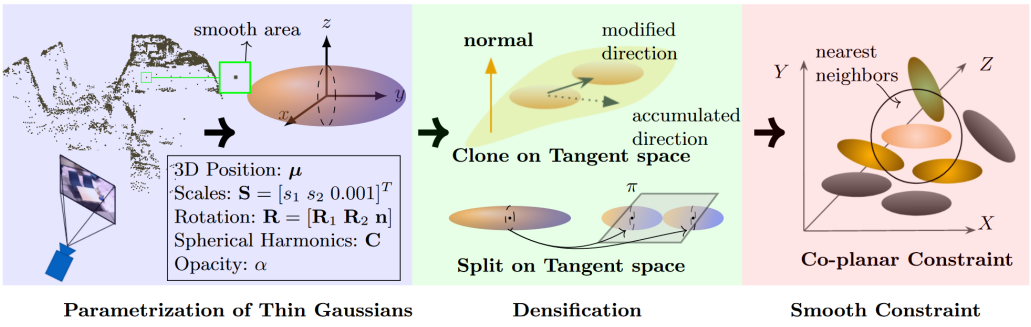


Figure 19. Pipeline from GeoGaussian [79]. A geometry-aware 3D Gaussian initialization strategy is proposed.

(a) 3DGS **(b)** LightGS **(c)** GeoGaussian **(d)** Reference

Figure 20. Comparison results for novel view synthesis between GeoGaussian [79] and previous methods on the ICL-NUIM [60] dataset. Note that the artifacts present in the results of 3DGS and LightGS disappear in GeoGaussian. Figures come from [79].

5.3. 3DGS with Data Compression

Despite the absolute advantages of 3DGS over NeRF-based methods in terms of training speed and rendering speed, 3DGS requires significantly more storage space. A vanilla NeRF representation of a typical scene requires only about 5MB, while 3DGS often requires an order of magnitude more. Scaffold-GS [19] uses anchor points to distribute local 3D Gaussians and generates a large number of 3D Gaussians around these anchor points to reduce the storage requirements significantly, as shown in Table 6. Many works have made improvements based on Scaffold-GS, including the introduction of level-of-detail strategies [80] and the use of adaptive quantization modules for further compression [81].

Table 6. Rendering FPS and storage size of Scaffold-GS [19].

Dataset	Mip-NeRF 360 [58]		Tanks and Temples [59]		Deep Blending [82]	
	FPS	Mem (MB)	FPS	Mem (MB)	FPS	Mem (MB)
3DGS	97	721	**123**	411	109	676
Scaffold-GS	**102**	**171** (4.2× ↓)	110	**87** (4.7× ↓)	**139**	**66** (10.2× ↓)

Others have combined 3DGS with vector quantization: a method widely used in the field of signal processing. EAGLES [38] quantifies implicit attributes to reduce the storage memory of 3D Gaussians and uses a coarse-to-fine rendering resolution approach during training to ensure rendering quality. Compact3D [39] performs vector quantization of Gaussian parameters during the training process. By treating each Gaussian as a vector, K-means clustering is executed to achieve compression. The resulting reduction in the number of Gaussians is illustrated in Table 7.

Table 7. Reducing number of Gaussians using Compact3D [39].

Dataset	Mip-NeRF 360 [58]				Tanks and Temples [59]				Deep Blending [82]			
	SSIM	PSNR	LPIPS	#Gauss	SSIM	PSNR	LPIPS	#Gauss	SSIM	PSNR	LPIPS	#Gauss
3DGS	0.813	27.42	0.217	3.30 M	0.844	23.68	0.178	1.83 M	0.899	29.49	0.246	2.80 M
Compact3D	0.813	27.42	0.227	**845 K**	0.844	23.71	0.188	**520 K**	0.905	29.73	0.249	**554 K**

5.4. 3DGS with Geometry Reconstruction

Since 3DGS represents scenes using discrete 3D anisotropic balls, reconstructing explicit geometries such as meshes is not as straightforward as with physics-based differentiable rendering methods. Few works have conducted preliminary explorations on how to utilize 3DGS for geometry reconstruction. SuGaR [20] proposes a constraint term on the scene surface to improve the geometric reconstruction effect of 3DGS and uses Poisson

reconstruction to extract meshes. Figure 21 shows the reconstruction and scene editing results of SuGaR.

Figure 21. Reconstruction and editing result of SuGaR [20].

Most recently, 2DGS [40] introduces 2D Gaussians to replace 3D Gaussians for scene representation and proposes a low-pass filter to prevent 2D Gaussians from generating line projections. However, these methods still fall short in reconstruction accuracy compared to NeRF-based implicit methods, let alone physics-based differentiable rendering methods.

6. Discussion

Differentiable rendering acts as a crucial link between image creation and analysis, delivering effective solutions for diverse applications in computer graphics and computer vision due to its capability of supplying gradients for optimization. This paper categorizes existing differentiable rendering methods based on the primary rendering algorithms they employ. Specifically, this work reviews physics-based, NeRF-based, and 3DGS-based differentiable rendering methods. It is noteworthy that while there are several existing reviews for NeRF-based and 3DGS-based methods, almost no reviews for PBDR exist. Therefore, our primary focus is on PBDR, with NeRF-based and 3DGS-based methods covered from several aspects to ensure completeness.

In this section, we summarize advancements and suggest potential directions for future research in physics-based, NeRF-based, and 3DGS-based differentiable rendering methods. A comparison of and conclusions related to these three categories of methods are presented in Section 7.

Physics-based differentiable rendering: The ultimate goal of PBDR is to reduce the variance of the gradient estimator as much as possible within the same computation time, allowing the inverse rendering algorithms to converge more quickly and efficiently.

For boundary sampling methods, state-of-the-art techniques have abandoned the strategy of explicitly finding silhouette edges for a given shading point. Instead, they first generate a path segment tangent to the geometry being differentiated, then complete it to a full path in a bidirectional manner. To importance sample these tangent segments, a guiding structure is built during the precomputation process. Currently, only a small portion of the seed rays used to build this guiding structure has a positive contribution. Thus, a new strategy is needed to build this guiding structure more efficiently. Moreover, the parameterization used for the boundary sampling space for triangle meshes exhibits high-frequency and sparse features, which hinder the importance sampling process. A new parameterization scheme promising a smoother distribution needs to be developed.

For reparameterization methods, state-of-the-art techniques use a warp field to reparameterize the rendering integral, ensuring that the discontinuities remain fixed in the newly reparameterized domain as the scene parameters change. The Monte Carlo technique is used to estimate the warp field at a specific point, which requires auxiliary rays to be emitted. This Monte Carlo technique introduces additional variance in the interior region, and the ray tracing process for auxiliary rays is also time-consuming. To reduce the variance and computation time for estimating the warp field, a new reparameterization scheme needs to be developed. Ideally, this new reparameterization scheme should be analytic so that the variance in the reparameterization step is reduced to zero.

Boundary sampling methods have advantages over reparameterization methods, as they produce a cleaner gradient in the interior region. This is because the Monte Carlo process used by reparameterization methods to estimate the warp field introduces additional variance in the interior region. The advantages of reparameterization methods over boundary sampling methods are that they neither require precomputation nor a guiding process before the main differentiable rendering process. Additionally, reparameterization methods can still produce good results in scenarios where boundary sampling methods struggle to build an effective guiding structure, such as when an area emitter is encapsulated within a transparent surface with low roughness.

NeRF-based differentiable rendering: Numerous technical improvements focusing on different aspects of the original NeRF have been proposed. Since our main focus is on PBDR, we will only review NeRF-based methods from several aspects.

The core problem for sparse-view and generalizable NeRF models is how to deduce the missing information while avoiding overfitting to individual views. Some methods first extract neural features from the input views and then aggregate them into the target novel view, optionally incorporating depth or geometry regularization during the training process. Others aim to infer a low-dimensional latent code for the scene, which is then decoded by a hypernetwork for final shading. Combining NeRF models with large vision models that are capable of introducing stronger scene priors, which are helpful for deducing the missing scene information, is a promising research direction.

One major challenge for NeRF-based methods is their prolonged training and inference time. Some methods embed neural features or small MLPs into voxels or on mesh surfaces then pair them with additional small MLPs to mitigate the catastrophic forgetting phenomenon of MLPs, resulting in faster convergence during training. Others cache the outputs of pre-trained networks into grids or tree structures, thereby eliminating the need for costly network queries and accelerating the inference process. Techniques such as early ray termination and empty space skipping are also usually adopted by these methods. Looking ahead, an avenue worth exploring is the use of more sophisticated data structures or more expressive neural features to enhance inference speed.

3DGS-based differentiable rendering: Like NeRF, various technical enhancements have been proposed for the original 3DGS. Given that our primary focus is on PBDR, we will limit our review of 3DGS-based methods to several specific aspects.

There are still several artifacts in the rendering results of 3DGS-based methods that need to be reduced. These include aliasing, which is typically mitigated by multi-scale approaches, and floater artifacts, which are usually addressed through filtering and pruning. Additionally, while surface normal regularization can enhance rendering quality for scenes with high-frequency reflections, there is still significant room for improvement.

Reducing memory usage without sacrificing too much rendering quality is also an essential improvement for 3DGS-based methods. This will not only benefit in terms of faster rendering speeds but will also enable quicker transmission and deployment on mobile devices. Existing methods usually adopt the strategy of filtering or pruning insignificant Gaussians, often followed by clustering or vector quantization to further reduce memory storage. However, extending these methods to dynamic scenes remains an under-explored area of research.

The problem of mesh reconstruction for 3DGS-based scene representation remains unresolved. Existing methods often employ a regularization term to align Gaussians with geometric surfaces, but the results still lack the precision found in NeRF-based geometry reconstruction, let alone the accuracy achieved by physics-based geometry reconstruction techniques.

7. Conclusions

As sub-branches of differentiable rendering, all these three categories of methods (physics-based, NeRF-based, and 3DGS-based) aim to propagate gradients from image pixel intensities to explicit or neural scene parameters. These gradients can then be used in

optimization algorithms to reconstruct the scene representation. The difference between these categories of methods lies in the primary rendering algorithms used, which in turn determine the differentiable rendering process.

The biggest difference between PBDR and the other two categories of methods is that its primary rendering process rigorously obeys the physical laws of light transport in the real world, whereas the processes used in NeRF-based and 3DGS-based methods are only approximations. Thus, the inverse rendering results of PBDR are more precise than those of the other two categories, but this comes at the cost of increased computation time. However, current PBDR theory is still underdeveloped. Existing methods mainly focus on the reconstruction of single objects with known emitters in purely virtual environments. An interesting research direction is to explore how to apply PBDR to scenes with various geometries in the real world or even in the wild, similar to how NeRF-based or 3DGS-based methods are used.

While both NeRF-based and 3DGS-based methods use rendering algorithms based on approximations of geometric optics, they differ significantly in form: NeRF is based on neural representations, whereas 3DGS is based on explicit representations. Further, 3DGS-based methods offer faster training and inference speeds compared to NeRF-based methods while maintaining competitive rendering quality thanks to their explicit representations. However, these explicit representations also result in a much larger memory footprint. Surface extraction results from NeRF-based methods are smoother than those from 3DGS-based methods due to a NeRF's underlying continuous representation. Moreover, a promising research direction is to explore how to incorporate more physics into NeRF-based and 3DGS-based methods to make them more physically correct and, consequently, achieve better reconstruction quality.

Author Contributions: Conceptualization, R.G. and Y.Q.; methodology, R.G. and Y.Q.; software, R.G.; validation, R.G. and Y.Q.; formal analysis, R.G.; investigation, R.G.; resources, Y.Q.; data curation, Y.Q.; writing—original draft preparation, R.G.; writing—review and editing, Y.Q.; visualization, R.G.; supervision, Y.Q.; project administration, Y.Q.; funding acquisition,Y.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is supported by the National Natural Science Foundation of China (No. 62072020) and Leading Talents in Innovation and Entrepreneurship of Qingdao, China (19-3-2-21-zhc).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: We thank the anonymous reviewers for their valuable suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. 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 (CVPR), Long Beach, CA, USA, 15–20 June 2019.
2. Nicolet, B.; Jacobson, A.; Jakob, W. Large steps in inverse rendering of geometry. *ACM Trans. Graph.* **2021**, *40*, 248. [CrossRef]
3. Jensen, J.N.; Hannemose, M.; Bærentzen, J.A.; Wilm, J.; Frisvad, J.R.; Dahl, A.B. Surface Reconstruction from Structured Light Images Using Differentiable Rendering. *Sensors* **2021**, *21*, 1068. [CrossRef]
4. Kuldashboy, A.; Umirzakova, S.; Allaberdiev, S.; Nasimov, R.; Abdusalomov, A.; Cho, Y.I. Efficient image classification through collaborative knowledge distillation: A novel AlexNet modification approach. *Heliyon* **2024**, *10*, e34376. [CrossRef] [PubMed]
5. 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 Proceedings of the ECCV, Glasgow, UK, 23–28 August 2020.
6. Kerbl, B.; Kopanas, G.; Leimkühler, T.; Drettakis, G. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* **2023**, *42*, 1–14. [CrossRef]
7. Chen, Z.; Chen, A.; Zhang, G.; Wang, C.; Ji, Y.; Kutulakos, K.N.; Yu, J. A Neural Rendering Framework for Free-Viewpoint Relighting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 5599–5610.
8. Ye, W.; Chen, S.; Bao, C.; Bao, H.; Pollefeys, M.; Cui, Z.; Zhang, G. IntrinsicNeRF: Learning Intrinsic Neural Radiance Fields for Editable Novel View Synthesis. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–3 October 2023.

9. Li, T.M.; Aittala, M.; Durand, F.; Lehtinen, J. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans. Graph.* **2018**, *37*, 222. [CrossRef]
10. Bangaru, S.P.; Li, T.M.; Durand, F. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.* **2020**, *39*, 245. [CrossRef]
11. Zhang, C.; Miller, B.; Yan, K.; Gkioulekas, I.; Zhao, S. Path-space differentiable rendering. *ACM Trans. Graph.* **2020**, *39*, 143. [CrossRef]
12. Zhang, C.; Wu, L.; Zheng, C.; Gkioulekas, I.; Ramamoorthi, R.; Zhao, S. A differential theory of radiative transfer. *ACM Trans. Graph.* **2019**, *38*, 227. [CrossRef]
13. Zhang, C.; Yu, Z.; Zhao, S. Path-space differentiable rendering of participating media. *ACM Trans. Graph.* **2021**, *40*, 76. [CrossRef]
14. Zhang, Z.; Roussel, N.; Jakob, W. Projective Sampling for Differentiable Rendering of Geometry. *ACM Trans. Graph.* **2023**, *42*, 212. [CrossRef]
15. Loubet, G.; Holzschuch, N.; Jakob, W. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.* **2019**, *38*, 228. [CrossRef]
16. Xu, P.; Bangaru, S.; Li, T.M.; Zhao, S. Warped-Area Reparameterization of Differential Path Integrals. *ACM Trans. Graph.* **2023**, *42*, 213. [CrossRef]
17. Yang, G.W.; Zhou, W.Y.; Peng, H.Y.; Liang, D.; Mu, T.J.; Hu, S.M. Recursive-nerf: An efficient and dynamically growing nerf. *IEEE Trans. Vis. Comput. Graph.* **2022**, *29*, 5124–5136. [CrossRef]
18. Wang, Q.; Wang, Z.; Genova, K.; Srinivasan, P.P.; Zhou, H.; Barron, J.T.; Martin-Brualla, R.; Snavely, N.; Funkhouser, T. Ibrnet: Learning multi-view image-based rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 4690–4699.
19. Lu, T.; Yu, M.; Xu, L.; Xiangli, Y.; Wang, L.; Lin, D.; Dai, B. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 17–21 June 2024; pp. 20654–20664.
20. Guédon, A.; Lepetit, V. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 17–21 June 2024; pp. 5354–5363.
21. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv* **2016**, arXiv:1612.00593.
22. Zhou, Y.; Tuzel, O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
23. Roynard, X.; Deschaud, J.E.; Goulette, F. Classification of Point Cloud Scenes with Multiscale Voxel Deep Network. *arXiv* **2018**, arXiv:1804.03583.
24. Godard, C.; Mac Aodha, O.; Brostow, G.J. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
25. Loper, M.M.; Black, M.J. OpenDR: An Approximate Differentiable Renderer. In Proceedings of the Computer Vision–ECCV 2014, Zurich, Switzerland, 6–12 September 2014; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 154–169.
26. Kato, H.; Harada, T. Learning View Priors for Single-view 3D Reconstruction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
27. Deng, K.; Liu, A.; Zhu, J.Y.; Ramanan, D. Depth-supervised nerf: Fewer views and faster training for free. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12882–12891.
28. 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 (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 4578–4587.
29. Yang, J.; Pavone, M.; Wang, Y. Freenerf: Improving few-shot neural rendering with free frequency regularization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 8254–8263.
30. Chen, Z.; Funkhouser, T.; Hedman, P.; Tagliasacchi, A. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 16569–16578.
31. Wu, L.; Lee, J.Y.; Bhattad, A.; Wang, Y.X.; Forsyth, D. DIVeR: Real-Time and Accurate Neural Radiance Fields with Deterministic Integration for Volume Rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 16200–16209.
32. Müller, T.; Evans, A.; Schied, C.; Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* **2022**, *41*, 102. [CrossRef]
33. Zhang, X.; Bi, S.; Sunkavalli, K.; Su, H.; Xu, Z. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 5449–5458.
34. Xu, Q.; Xu, Z.; Philip, J.; Bi, S.; Shu, Z.; Sunkavalli, K.; Neumann, U. Point-nerf: Point-based neural radiance fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 5438–5448.

35. Yu, Z.; Chen, A.; Huang, B.; Sattler, T.; Geiger, A. Mip-Splatting: Alias-free 3D Gaussian Splatting. *arXiv* **2023**, arXiv:2311.16493.
36. Malarz, D.; Smolak, W.; Tabor, J.; Tadeja, S.; Spurek, P. Gaussian Splitting Algorithm with Color and Opacity Depended on Viewing Direction. *arXiv* **2023**, arXiv:2312.13729.
37. Niemeyer, M.; Manhardt, F.; Rakotosaona, M.J.; Oechsle, M.; Duckworth, D.; Gosula, R.; Tateno, K.; Bates, J.; Kaeser, D.; Tombari, F. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv* **2024**, arXiv:2403.13806.
38. Girish, S.; Gupta, K.; Shrivastava, A. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. *arXiv* **2023**, arXiv:2312.04564.
39. Navaneet, K.; Meibodi, K.P.; Koohpayegani, S.A.; Pirsiavash, H. Compact3d: Compressing gaussian splat radiance field models with vector quantization. *arXiv* **2023**, arXiv:2311.18159.
40. Huang, B.; Yu, Z.; Chen, A.; Geiger, A.; Gao, S. 2d gaussian splatting for geometrically accurate radiance fields. *arXiv* **2024**, arXiv:2403.17888.
41. Yan, K.; Lassner, C.; Budge, B.; Dong, Z.; Zhao, S. Efficient estimation of boundary integrals for path-space differentiable rendering. *ACM Trans. Graph.* **2022**, *41*, 123. [CrossRef]
42. Magnus, J.R.; Neudecker, H. *Matrix Differential Calculus with Applications in Statistics and Econometrics*; John Wiley & Sons: Hoboken, NJ, USA, 2019.
43. Wang, M.; Yu, Q.; Liu, H. Three-Dimensional-Consistent Scene Inpainting via Uncertainty-Aware Neural Radiance Field. *Electronics* **2024**, *13*, 448. [CrossRef]
44. Ma, L.; Li, X.; Liao, J.; Zhang, Q.; Wang, X.; Wang, J.; Sander, P.V. Deblur-nerf: Neural radiance fields from blurry images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12861–12870.
45. Zeng, Y.; Lei, J.; Feng, T.; Qin, X.; Li, B.; Wang, Y.; Wang, D.; Song, J. Neural Radiance Fields-Based 3D Reconstruction of Power Transmission Lines Using Progressive Motion Sequence Images. *Sensors* **2023**, *23*, 9537. [CrossRef] [PubMed]
46. Song, L.; Wang, G.; Liu, J.; Fu, Z.; Miao, Y. SC-NeRF: Self-Correcting Neural Radiance Field with Sparse Views. *arXiv* **2023**, arXiv:2309.05028.
47. Jin, P.; Yu, Z. Research on 3D Visualization of Drone Scenes Based on Neural Radiance Fields. *Electronics* **2024**, *13*, 1682. [CrossRef]
48. Dong, B.; Chen, K.; Wang, Z.; Yan, M.; Gu, J.; Sun, X. MM-NeRF: Large-Scale Scene Representation with Multi-Resolution Hash Grid and Multi-View Priors Features. *Electronics* **2024**, *13*, 844. [CrossRef]
49. Wang, B.; Zhang, D.; Su, Y.; Zhang, H. Enhancing View Synthesis with Depth-Guided Neural Radiance Fields and Improved Depth Completion. *Sensors* **2024**, *24*, 1919. [CrossRef]
50. Dellaert, F.; Lin, Y. Neural Volume Rendering: NeRF and Beyond. *arXiv* **2021**, arXiv:2101.05204. Available online: http://arxiv.org/abs/2101.05204 (accessed on 2 August 2024).
51. Xie, Y.; Takikawa, T.; Saito, S.; Litany, O.; Yan, S.; Khan, N.; Tombari, F.; Tompkin, J.; sitzmann, V.; Sridhar, S. Neural Fields in Visual Computing and beyond. *Comput. Graph. Forum* **2022**, *41*, 641–676. [CrossRef]
52. Tewari, A.; Thies, J.; Mildenhall, B.; Srinivasan, P.; Tretschk, E.; Yifan, W.; Lassner, C.; Sitzmann, V.; Martin-Brualla, R.; Lombardi, S.; et al. Advances in Neural Rendering. *Comput. Graph. Forum* **2022**, *41*, 703–735. [CrossRef]
53. Gao, K.; Gao, Y.; He, H.; Lu, D.; Xu, L.; Li, J. NeRF: Neural Radiance Field in 3D Vision, a Comprehensive Review. *arXiv* **2023**, arXiv:2210.00379. Available online: http://arxiv.org/abs/2210.00379 (accessed on 2 August 2024).
54. Tancik, M.; Srinivasan, P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.; Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7537–7547.
55. Jensen, R.; Dahl, A.; Vogiatzis, G.; Tola, E.; Aanaes, H. Large Scale Multi-view Stereopsis Evaluation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 24–27 June 2014.
56. 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.* **2019**, *38*, 29. [CrossRef]
57. Choi, S.; Zhou, Q.; Miller, S.; Koltun, V. A Large Dataset of Object Scans. *arXiv* **2016**, arXiv:1602.02481. Available online: http://arxiv.org/abs/1602.02481 (accessed on 2 August 2024).
58. Barron, J.T.; Mildenhall, B.; Verbin, D.; Srinivasan, P.P.; Hedman, P. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 5470–5479.
59. Knapitsch, A.; Park, J.; Zhou, Q.Y.; Koltun, V. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph.* **2017**, *36*, 78. [CrossRef]
60. Handa, A.; Whelan, T.; McDonald, J.; Davison, A.J. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 1524–1531. [CrossRef]
61. Dai, A.; Chang, A.X.; Savva, M.; Halber, M.; Funkhouser, T.; Niessner, M. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
62. Wang, Z.; Bovik, A.; Sheikh, H.; Simoncelli, E. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef]

63. Zhang, R.; Isola, P.; Efros, A.A.; Shechtman, E.; Wang, O. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
64. 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 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 2846–2855.
65. Niemeyer, M.; Barron, J.T.; Mildenhall, B.; Sajjadi, M.S.M.; Geiger, A.; Radwan, N. RegNeRF: Regularizing Neural Radiance Fields for View Synthesis From Sparse Inputs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 5480–5490.
66. Xu, D.; Jiang, Y.; Wang, P.; Fan, Z.; Wang, Y.; Wang, Z. Neurallift-360: Lifting an in-the-wild 2d photo to a 3d object with 360deg views. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 4479–4489.
67. Hu, S.; Hong, F.; Pan, L.; Mei, H.; Yang, L.; Liu, Z. Sherf: Generalizable human nerf from a single image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–6 October 2023; pp. 9352–9364.
68. Qin, S.; Xiao, J.; Ge, J. Dip-NeRF: Depth-Based Anti-Aliased Neural Radiance Fields. *Electronics* **2024**, *13*, 1527. [CrossRef]
69. Hedman, P.; Srinivasan, P.P.; Mildenhall, B.; Barron, J.T.; Debevec, P. Baking Neural Radiance Fields for Real-Time View Synthesis. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 5875–5884.
70. Chen, A.; Xu, Z.; Zhao, F.; Zhang, X.; Xiang, F.; Yu, J.; Su, H. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 14124–14133.
71. Pfister, H.; Zwicker, M.; Van Baar, J.; Gross, M. Surfels: Surface elements as rendering primitives. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 23–28 July 2000; pp. 335–342.
72. Fei, B.; Xu, J.; Zhang, R.; Zhou, Q.; Yang, W.; He, Y. 3D Gaussian Splatting as New Era: A Survey. *IEEE Trans. Vis. Comput. Graph.* 2024, *early access*. [CrossRef]
73. Ren, L.; Pfister, H.; Zwicker, M. Object Space EWA Surface Splatting: A Hardware Accelerated Approach to High Quality Point Rendering. *Comput. Graph. Forum* **2002**, *21*, 461–470. [CrossRef]
74. Yan, Z.; Low, W.F.; Chen, Y.; Lee, G.H. Multi-scale 3d gaussian splatting for anti-aliased rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 17–21 June 2024; pp. 20923–20931.
75. Song, X.; Zheng, J.; Yuan, S.; Gao, H.a.; Zhao, J.; He, X.; Gu, W.; Zhao, H. SA-GS: Scale-Adaptive Gaussian Splatting for Training-Free Anti-Aliasing. *arXiv* **2024**, arXiv:2403.19615.
76. Jiang, Y.; Tu, J.; Liu, Y.; Gao, X.; Long, X.; Wang, W.; Ma, Y. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 17–21 June 2024; pp. 5322–5332.
77. Cheng, K.; Long, X.; Yang, K.; Yao, Y.; Yin, W.; Ma, Y.; Wang, W.; Chen, X. GaussianPro: 3D Gaussian Splatting with Progressive Propagation. *arXiv* **2024**, arXiv:2402.14650.
78. Fan, Z.; Wang, K.; Wen, K.; Zhu, Z.; Xu, D.; Wang, Z. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. *arXiv* **2024**, arXiv:2311.17245. Available online: http://arxiv.org/abs/2311.17245 (accessed on 2 August 2024).
79. Li, Y.; Lyu, C.; Di, Y.; Zhai, G.; Lee, G.H.; Tombari, F. Geogaussian: Geometry-aware gaussian splatting for scene rendering. *arXiv* **2024**, arXiv:2403.11324.
80. Ren, K.; Jiang, L.; Lu, T.; Yu, M.; Xu, L.; Ni, Z.; Dai, B. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv* **2024**, arXiv:2403.17898.
81. Chen, Y.; Wu, Q.; Cai, J.; Harandi, M.; Lin, W. HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression. *arXiv* **2024**, arXiv:2403.14530.
82. Hedman, P.; Philip, J.; Price, T.; Frahm, J.M.; Drettakis, G.; Brostow, G. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.* **2018**, *37*, 257. [CrossRef]