

Relightable 3D Gaussians: Realistic Point Cloud Relighting with BRDF Decomposition and Ray Tracing

Jian Gao^{1*}, Chun Gu^{2*}, Youtian Lin¹, Zhihao Li³, Hao Zhu¹, Xun Cao¹,
Li Zhang^{2✉}, and Yao Yao^{1✉}

¹ Nanjing University

² Fudan University

³ Huawei Noah’s Ark Lab

Abstract. In this paper, we present a novel differentiable point-based rendering framework to achieve photo-realistic relighting. To make the reconstructed scene relightable, we enhance vanilla 3D Gaussians by associating extra properties, including normal vectors, BRDF parameters, and incident lighting from various directions. From a collection of multi-view images, the 3D scene is optimized through 3D Gaussian Splatting while BRDF and lighting are decomposed by physically based differentiable rendering. To produce plausible shadow effects in photo-realistic relighting, we introduce an innovative point-based ray tracing with the bounding volume hierarchies for efficient visibility pre-computation. Extensive experiments demonstrate our improved BRDF estimation, novel view synthesis and relighting results compared to state-of-the-art approaches. The proposed framework showcases the potential to revolutionize the mesh-based graphics pipeline with a point-based pipeline enabling editing, tracing, and relighting.

Keywords: 3D Gaussian Splatting, Relighting, Point based rendering

1 Introduction

Reconstructing 3D scenes from multi-view images for photo-realistic rendering is a fundamental problem at the intersection of computer vision and graphics. Recently, 3D Gaussian Splatting (3DGS) [26] has been proposed and has gained significant attention from the community. The method employs a set of 3D Gaussian points to represent a 3D scene and projects these points onto a designated view through a tile-based rasterization. Attributes of each 3D Gaussian point are then optimized through the point-based differentiable rendering. Notably, 3DGS achieves real-time rendering with quality comparable to or even surpassing previous state-of-the-art methods (e.g., Mip-NeRF [3]), with a training speed on par with the most efficient Instant-NGP [32]. However, the current 3DGS is unable to reconstruct a scene that can be relighted under different lighting conditions,

*Equally contributed.

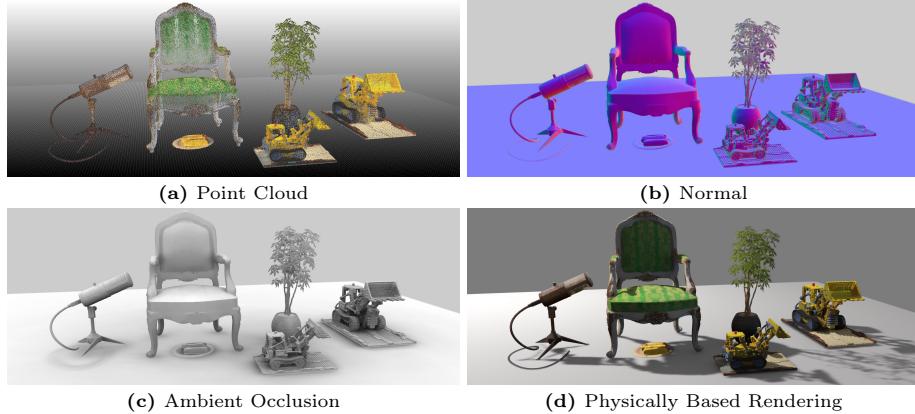


Fig. 1: Visual results of our pipeline on a multi-object composition scene. In our pipeline, we represent a scene as *Relightable 3D Gaussians*. From multi-view images, we recover the geometry and materials of individual objects with *inverse rendering* techniques (see Sec. 3). Then, objects are easily composited into a new scene, thanks to our explicit representation. After that, we solve the complex occlusions through *point-based ray tracing* (see Sec. 4) and re-light the new scene. Ultimately, we achieve high-fidelity relighting with remarkably **realistic shadow**.

making the method only applicable to the task of novel view synthesis. In addition, ray tracing, a crucial component in achieving realistic rendering, remains an unresolved challenge in point-based representation, limiting 3DGS to more dedicated rendering effects such as shadowing and light reflectance.

In this paper, we aim to reconstruct a relightable 3D Gaussian point cloud from multi-view images with a differentiable rendering framework and achieve realistic relighting. We make 3D Gaussian relightable by assigning it with additional normal, BRDF properties, and incident light information to model per-point light reflectance. In contrast to the plain alpha blending in the original 3DGS [26], we apply physically based rendering (PBR) to get a PBR color for each 3D Gaussian point, which is then alpha-composited to obtain a rendered color for the corresponding image pixel. For robust material and lighting estimation, we split the incident light into a global environment map and an indirect incident light field. To capture accurate visibility for each 3D Gaussian, we propose a novel ray tracing method based on the bounding volume hierarchy (BVH), which enables efficient visibility pre-computing for real-time rendering. Additionally, the proposed ray tracing method can handle complex occlusion relationships in a novel multi-object composition scene, thus realizing realistic shadow effects. Moreover, proper regularizations are introduced to enhance the geometry and mitigate the material-lighting ambiguity during the optimization, including constraints on depth distribution, smoothness priors, and a lighting regularization.

Extensive experiments conducted across diverse datasets demonstrate the improved material and lighting estimation results and novel view rendering quality. Additionally, we illustrate the relightable and editable capabilities of our system through multi-object composition in a novel lighting environment. To summarize, major contributions of the paper include:

- We propose a material and lighting decomposition scheme for 3D Gaussian Splatting, where a normal vector, BRDF values, and incident lights are assigned and optimized for each 3D Gaussian point.
- We introduce a novel point-based ray tracing approach based on bounding volume hierarchy, enabling efficient visibility pre-computing for each 3D Gaussian point and rendering of a 3D scene with realistic shadow effect.
- We demonstrate a comprehensive graphics pipeline solely based on a discretized point representation, supporting relighting, editing, and ray tracing of a reconstructed 3D point cloud.

2 Related Works

Neural Radiance Field. Differentiable rendering techniques, exemplified by Neural Radiance Field (NeRF) [31], have garnered significant attention [3,12,32]. NeRF utilizes an implicit Multi-Layer Perceptron (MLP) that takes 3D positions and viewing directions as inputs to generate density and view-dependent colors for differentiable volume rendering. Despite its powerful neural implicit representation, both speed and quality can be improved. Efficiency improvements mainly focus on optimizing queries on neural fields and minimizing queries [12, 13, 15, 16, 21, 22, 32, 39]. Quality enhancements involve anti-aliasing [3–5] or utilizing exterior supervision [14, 48], etc.

Differentiable Point Based Rendering. Using points as rendering primitives was first proposed in [29]. To address the issue of resulting holey images when rasterizing discrete points directly, solutions fall into two categories. One approach directly employs points as rendering primitives and encodes the geometric and photometric features near the point using SIFT descriptors [37] or neural descriptors [1,38]. This leads to a feature image with gaps, which is then decoded to recover a hole-free RGB image. The other category models a point as a primitive occupying a specific space, like a surfel [36,47] or a 3D Gaussian [26]. The rendered image is then generated using a splatting technique. This technique saw significant development in the 2000s [36, 57, 58], and in the era of differentiable rendering, PointRF [52], DSS [47] and 3DGS [26] serve as notable examples. We assert that the point serves as a promising rendering primitive owing to its inherent ease of editing, as also corroborated in [54].

Material and Lighting Estimation. Decomposing the materials and illumination of a scene from multi-view images presents a formidable challenge owing to its intrinsic high-dimensional complexity. Some methods simplified the problem under the assumption of a controllable light [2,6–8,18,34,35,40]. Subsequent research explores more complex lighting models to cope with realistic scenarios. NeRV [41] and PhySG [51] leverage an environmental map to manage arbitrary

lighting conditions. NeRD [9] addresses the challenge posed by images captured under varying illumination by attributing Spherical Gaussians to each image. IRON [50] introduces an innovative edge sampling algorithm tailored for neural SDFs. NeRFactor [53] exploits light visibility to achieve superior material and lighting decomposition. Further studies [19, 52] address the consideration of indirect lighting, leading to enhanced BRDF estimation quality. [20, 33] utilize differentiable marching tetrahedrons for direct optimization on mesh surfaces. [10, 28] deal with varying cameras, illumination, and backgrounds. Ref-NeRF [42] suggests the employment of integrated direction encoding to ameliorate the reconstruction fidelity of reflective objects. [30, 33] use split-sum approximation to approximate the shading effects. NeMF [55] represents the scene as a microflake volume. NeFII [44] integrates lights through path tracing with Monte Carlo sampling. InvRender [56] computes indirect illumination by directly leveraging the neural radiance field, rather than concurrently estimating it alongside the decomposition of direct lighting and materials. TensoIR [23] performs inverse rendering based on tensor factorization and neural fields. NeILF [45] and NeILF++ [49] expresses the incident lights as a neural incident light field, while NeILF++ integrates VolSDF [46] with NeILF through inter-reflection. However, existing schemes rarely delve into BRDF estimation for point clouds, and the estimation and rendering quality remain to be improved.

3 Relightable 3D Gaussians

In this section, we introduce a novel pipeline to decompose materials and lighting from multi-view images based on 3D Gaussian Splatting (3DGS) [26]. An overview of our pipeline is shown in Fig. 2.

3.1 Preliminary

Distinct from the widely adopted Neural Radiance Field, 3DGS [26] employs explicit 3D Gaussian points as its rendering primitives. A 3D Gaussian point is mathematically defined as:

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \quad (1)$$

where $\boldsymbol{\mu}$ and Σ denote the 3D spatial mean and covariance matrix, respectively. Each Gaussian is also equipped with an opacity o and a view-dependent color \mathbf{c} .

The rendering process in 3DGS is divided into two main steps. Firstly, 3D Gaussians are projected to 2D Gaussians on the image plane. The 2D means are determined by accurate projection of 3D means, while the 2D covariance matrices are approximated by: $\Sigma' = \mathbf{J}\mathbf{W}\Sigma\mathbf{W}^\top\mathbf{J}^\top$, where \mathbf{W} and \mathbf{J} denote the viewing transformation and the Jacobian of the affine approximation of perspective projection transformation [57]. Subsequently, the pixel color is derived by alpha blending N ordered 2D Gaussians from front to back:

$$\mathcal{C} = \sum_{i \in N} T_i \alpha_i \mathbf{c}_i, \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j). \quad (2)$$

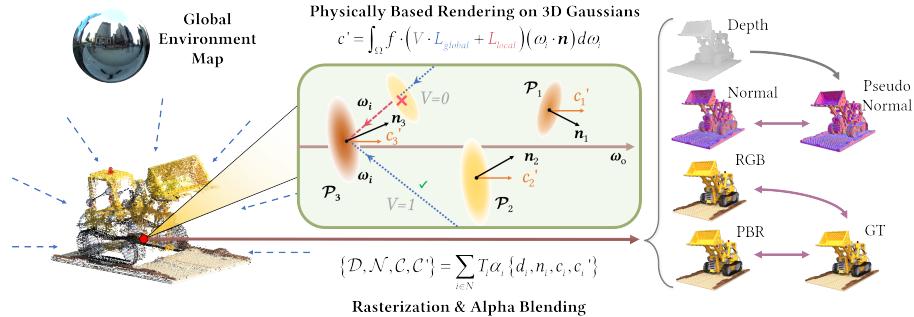


Fig. 2: The proposed differentiable rendering pipeline. Starting with a collection of 3D Gaussians that embody geometry, materials, and lighting attributes, we first execute the physically based rendering equation for each 3D Gaussian to determine its outgoing radiance, denoted as c' . Following this, we perform rasterization and alpha blending to obtain vanilla color map C , PBR color map C' , depth map D , normal map N , etc. To optimize relightable 3D Gaussians, we utilize the ground truth image C_{gt} and the pseudo normal map derived from D for supervision.

Here, α is obtained by multiplying the opacity o with the 2D covariance’s contribution computed from Σ' and pixel coordinate in image space [26]. In implementation details, the covariance matrix Σ is parameterized as a unit quaternion q and a 3D scaling vector s to maintain its meaningful interpretation throughout optimization. Additionally, view-dependent color c_i is represented through a set of Spherical Harmonics (SH).

In summation, a 3D scene is represented by a collection of 3D Gaussians, with the i^{th} Gaussian \mathcal{P}_i parameterized as $\{\mu_i, q_i, s_i, o_i, c_i\}$.

3.2 Geometry Enhancement

Scene geometry, specifically surface normal, is essential for realistic physically based rendering, which will be discussed in Sec. 3.3.

Normal Estimation. We incorporate a normal attribute n for each 3D Gaussian and try to solve it. A potential methodology treats the spatial mean of a 3D Gaussian as a conventional point, and executes normal estimation based on the local planar assumption. However, this approach cannot provide accurate normal estimation. The reasons are two fold: First, the reconstructed Gaussian point cloud is often sparse. More critically, the Gaussian points are naturally *soft*, which means these points are not precisely aligned with the object surface.

To address these limitations, we propose to optimize n from an initial random vector via back-propagation for each 3D Gaussian. We perceive the depth of all Gaussians along a ray as a distribution, and estimate the pixel depth as the expectation of this distribution. Similarly, we determine the normal for each

pixel. This process is described by:

$$\{\mathcal{D}, \mathcal{N}\} = \sum_{i \in N} w_i \{d_i, \mathbf{n}_i\}, \quad (3)$$

where d_i , \mathbf{n}_i and $w_i = T_i \alpha_i / \sum_{i \in N} T_i \alpha_i$ denote the depth, normal and weight of the point. We then encourage the consistency between the rendered normal \mathcal{N} and the pseudo normal $\tilde{\mathcal{N}}$, which is computed from the rendered depth \mathcal{D} under the local planarity assumption. The normal consistency is quantified as follows:

$$\mathcal{L}_n = \|\mathcal{N} - \tilde{\mathcal{N}}\|_2. \quad (4)$$

Normal Gradient Based Densification. To achieve superior rendering fidelity in detail regions, vanilla 3DGS [26] densifies 3D Gaussians through the gradient of view space points. Drawing upon this, to improve normal recovery in thin regions, we introduce an additional densification criterion on the gradient of normals. Specifically, we densify Gaussians whose normal gradient exceed a threshold T_n .

Constraint on Depth Distribution. Given our assumption of the object possessing an accurate surface, we enforce a constraint on the depth distribution by minimizing the uncertainty. The uncertainty is defined as:

$$\mathcal{L}_u = \mathcal{D}_{sq} - \mathcal{D}^2, \quad (5)$$

where $\mathcal{D}_{sq} = \sum_{i \in N} w_i d_i^2$, and \mathcal{D} is the depth estimation defined in Eq. 3. This constraint on the depth distribution drives Gaussian points to the object surface, thereby improving geometric reconstruction.

Object Mask Constraint. If there is a mask indicating the object, we can constrain the optimization by a binary cross entropy loss [43]:

$$\mathcal{L}_O = -M \log O - (1 - M) \log (1 - O), \quad (6)$$

where M is the object mask and $O = \sum_{i \in N} T_i \alpha_i$. With this constraint, O is forced to be aligned with the distribution of M , and so we get opaque surface.

3.3 BRDF and Light Modeling

Rendering Equation. We use the the rendering equation [24] to model light interaction with surfaces, accounting for their material properties and geometry. It is given by:

$$L_o(\omega_o, \mathbf{x}) = \int_{\Omega} f(\omega_o, \omega_i, \mathbf{x}) L_i(\omega_i, \mathbf{x})(\omega_i \cdot \mathbf{n}) d\omega_i, \quad (7)$$

where \mathbf{x} and \mathbf{n} are the surface point and its normal vector, f is the Bidirectional Reflectance Distribution Function (BRDF), and L_i and L_o denote the incoming and outgoing radiance in directions ω_i and ω_o . Ω signifies the hemispherical domain above the surface.

Prior methods [49, 51] typically begin with the acquisition of intersection points between rays and surface through differentiable rendering. Then they apply the rendering equation at these points to facilitate Physical Based Rendering (PBR). However, these approach present significant challenges in the point-based framework. Certainly, it is feasible to extract surfaces in 3DGS [17]; but, it necessitates a coordinate-based incident light field for inverse PBR. The querying of this field at millions of points during every iteration imposes a substantial computational burden. Moreover, it is pertinent to acknowledge that accurately extracting geometric surfaces presents considerable challenges in 3DGS system.

To tackle this issue, we propose to compute PBR color $\{c'_i\}_{i=0}^N$ for each 3D Gaussian, and then obtain the PBR image C' through alpha-blending, as shown in Fig. 2. This method is more efficient for two main reasons. First, PBR is performed on fewer 3D Gaussians rather than image pixels in all input views, as each Gaussian affects multiple pixels based on its scale. Second, by allocating discrete attributes for each Gaussian, we avoid the need for global neural fields. **BRDF Parameterization.** To make 3D Gaussians relightable, we assign BRDF properties to each Gaussian and adopt a simplified Disney BRDF model [11]. The BRDF properties include an albedo $b \in [0, 1]^3$ and a roughness $r \in [0, 1]$, and the BRDF f in Eq. 7 is divided into a diffuse term $f_d = \frac{b}{\pi}$ and a specular term:

$$f_s(\omega_o, \omega_i) = \frac{D(\mathbf{h}; r) \cdot F(\omega_o, \mathbf{h}) \cdot G(\omega_i, \omega_o, h; r)}{(\mathbf{n} \cdot \omega_i) \cdot (\mathbf{n} \cdot \omega_o)}, \quad (8)$$

where $\mathbf{h} = (\omega_i + \omega_o)/2$ is the half vector, D , F and G denote the normal distribution function, Fresnel term and geometry term.

Incident Light Modeling. Optimizing a NeILF for each Gaussian can be excessively unconstrained, leading to challenges in accurately decomposing incident lights from its appearance. To address this issue, we apply a prior by partitioning the incident light into globally shared direct component and individual per-Gaussian indirect components. The sampled incident light at a Gaussian from direction ω_i is represented as:

$$L_i(\omega_i) = V(\omega_i) \cdot L_{direct}(\omega_i) + L_{indirect}(\omega_i), \quad (9)$$

where $V(\omega_i)$ represents the visibility term which will be further discussed in Sec. 4, the indirect light term $L_{indirect}$ is parameterized by 3-level SH, denoted as \mathbf{l} , while the direct light term L_{direct} is parameterized as a globally shared 16x32 environment map, denoted as \mathbf{l}^{env} . Despite the utilization of relatively low-level Spherical Harmonics (SH), we can still capture some inter-reflection effects, as the rendering color of a given pixel arises from the collaboration of multiple relightable 3D Gaussians.

For each 3D Gaussian, we sample N_s incident directions over the hemisphere space through Fibonacci sampling [45] to provide numerical integration for Eq. 7. The PBR color of each Gaussian is then given by:

$$c'(\omega_o) = \sum_{i=0}^{N_s} (f_d + f_s(\omega_o, \omega_i)) L_i(\omega_i) (\omega_i \cdot \mathbf{n}) \Delta\omega_i, \quad (10)$$

where ω_i is the solid angle.

To summarize, our method represents a 3D scene as a set of relightable 3D Gaussians and a global environment light \mathbf{l}^{env} , where the i^{th} Gaussian \mathcal{P}_i is parameterized as $\{\boldsymbol{\mu}_i, \mathbf{q}_i, \mathbf{s}_i, \mathbf{o}_i, \mathbf{c}_i, \mathbf{n}_i, \mathbf{b}_i, r_i, \mathbf{l}_i\}$.

3.4 Regularizations

To mitigate the materials-lighting ambiguity [45], proper regularizations are utilized to facilitate their plausible decomposition.

Light Regularization. We apply a light regularization assuming a near-natural white incident light [30]:

$$\mathcal{L}_{light} = \sum_c (L_c - \frac{1}{3} \sum_c L_c), c \in \{R, G, B\}. \quad (11)$$

Smoothness Priors. We expect that the BRDF properties do not change drastically in homogeneous areas [45]. We define a smooth constraint on roughness as:

$$\mathcal{L}_{s,r} = \|\nabla R\| \exp(-\|\nabla C_{gt}\|), \quad (12)$$

where $R = \sum_{i \in N} w_i r_i$ is the rendered roughness map. Similarly, we also define smoothness constraints $\mathcal{L}_{s,n}$ and $\mathcal{L}_{s,b}$ on normal and albedo.

4 Point-based Ray Tracing

For real-time realistic rendering with plausible shadow effects on relightable 3D Gaussians, we introduce a novel point-based ray tracing approach in this section.

4.1 Ray Tracing on 3D Gaussians

Our proposed ray tracing technique on 3D Gaussians is built upon the Bounding Volume Hierarchy (BVH), enabling efficient visibility querying along a ray. Our method adopts the idea from [25], an in-place algorithm for constructing a binary radix tree that maximizes parallelism and facilitates real-time BVH construction. Specifically, we construct a binary radix tree from a given set of 3D Gaussians, where each leaf node represents the tight bounding box of a Gaussian, and each internal node denotes the bounding box of its two children.

Unlike ray tracing with opaque polygonal meshes, where only the closest intersection point is required, ray tracing with semi-transparent Gaussians necessitates accounting for all Gaussians potentially influencing the ray's transmittance. The process of ray tracing on 3D Gaussian points can be described as follows: the ray travels from the camera center and accumulates transmittance as it passes through 3D Gaussian points until the transmittance is zero. The first key issue to be addressed is how to define the contribution of a single 3D Gaussian to the transmittance. As discussed in Sec.3.1, transforming a 3D Gaussian to a 2D Gaussian involves an approximation, which complicating the

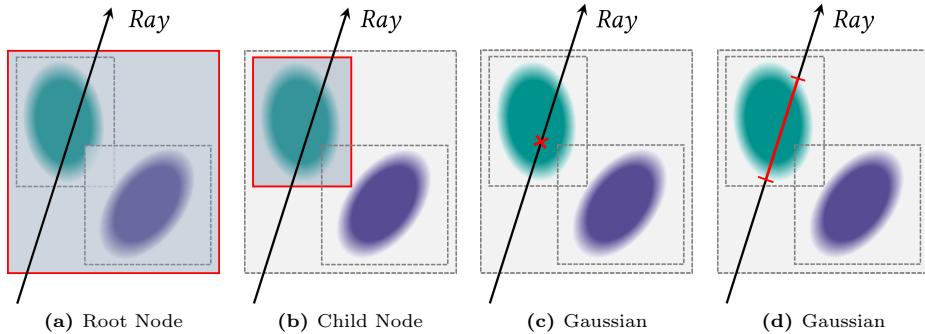


Fig. 3: Intersection tests in point-based ray tracing. Intersection point between ray and Gaussian is obtained by three steps: (a) intersect the BVH root node; (b) dive into the intersected child nodes recursively until the leaf node; (c) perform Eq. 13 to get the equivalent intersection point. (d) shows that a 3D Gaussian actually has non-negligible effect on a segment of a ray.

identification of accurate intersection between the Gaussian and a ray. As shown in Fig. 3d, a 3D Gaussian actually has non-negligible effect on a segment of a ray. Thus, drawing inspiration from [27], we approximate the intersection of ray with 3D Gaussian as a point where the 3D Gaussian’s contribution peaks, as showed in Fig. 3(3). The equivalent intersection point is defined as:

$$\mathbf{r}_x = \mathbf{r}_o + t_j \mathbf{r}_d, \quad (13)$$

where \mathbf{r}_o denotes the origin, \mathbf{r}_d is direction of the ray, corresponding to the previously mentioned ω_i , t_j is defined as:

$$t_j = \frac{(\boldsymbol{\mu} - \mathbf{r}_o)^T \boldsymbol{\Sigma} \mathbf{r}_d}{\mathbf{r}_d^T \boldsymbol{\Sigma} \mathbf{r}_d}. \quad (14)$$

Subsequently, we approximate the contribution of this 3D Gaussian to the ray as α_j at the equivalent intersection point \mathbf{r}_x .

Considering the transmittance along a ray: $T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$, it is evident that the order of α_j does not affect T_i . In other word, the order in which Gaussians are encountered along a ray does not impact the overall transmittance. As illustrated in Fig. 3, starting from the root node of the binary radix tree, intersection tests are recursively performed between the ray and the bounding volumes of each node’s children. Upon reaching a leaf node, the associated Gaussian is identified. Through this traversal, the transmittance T is progressively attenuated:

$$T_i = (1 - \alpha_{i-1}) T_{i-1}, \quad \text{for } i = 1, \dots, j-1 \text{ with } T_1 = 1. \quad (15)$$

To speed up ray tracing, the process is early terminated if a ray’s transmittance drops below a certain threshold T_{min} .

4.2 Visibility Pre-computation

In Sec. 3.3, a potential ambiguity in the decomposition of indirect and direct lights has been discerned. While the essential visibility term can be derived via the proposed ray tracing on 3D Gaussians (Sec. 4.1), querying the visibility term through ray tracing online proves challenging due to the computational complexity. Nevertheless, given our exclusive focus on static scenes, we can pre-compute the visibility term $V(\omega_i)$ across the hemispherical domain determined by \mathbf{n} for each Gaussian, and subsequently integrate it into the rendering equation.

Hence, we divide the optimization process into two stages. In the first stage, we optimize a vanilla 3D Gaussian point cloud, augmented with an additional normal vector \mathbf{n} for each Gaussian. Immediately afterwards, we pre-compute the per-Gaussian visibility through the proposed ray tracing. In the second stage, we lock the geometry of 3D Gaussians and focus solely on optimizing the material and lighting parameters using the pipeline described in Fig. 2.

4.3 Point Based Relighting Pipeline

Based on our relightable 3D Gaussians, we devise a point based graphics pipeline that seamlessly integrates **effortless scene editing** and **realistic relighting**. To our knowledge, there is currently no point-based rendering pipeline that effectively accomplishes both tasks. Although explicit point representations make scene editing easy, achieving photo-realistic relighting proved nearly insurmountable in existing point-based rendering pipeline. Conversely, in inverse rendering approaches based on implicit representations, while highly realistic relighting is feasible, scene editing presents a challenging endeavor.

As an illustration, we concentrate on relighting in a multi-object composition scene. Initially, our pipeline computes the visibility $V(\omega_i)$ for each Gaussian point via ray tracing (Sec. 4.1). Despite the challenging inter-object occlusions introduced by composition, our proposed point-based ray tracing method adeptly manages these complexities and ensures accurate occlusion updates within the novel scene. Subsequently, the rendering process unfolds (Sec. 3.3), starting with Gaussian-level PBR and ending with alpha blending. Throughout this process, the original indirect lighting of each object is discarded. Consequently, we achieve relighting with remarkably vivid shadow effects solely based on a discrete set of points, as illustrated in Fig. 1.

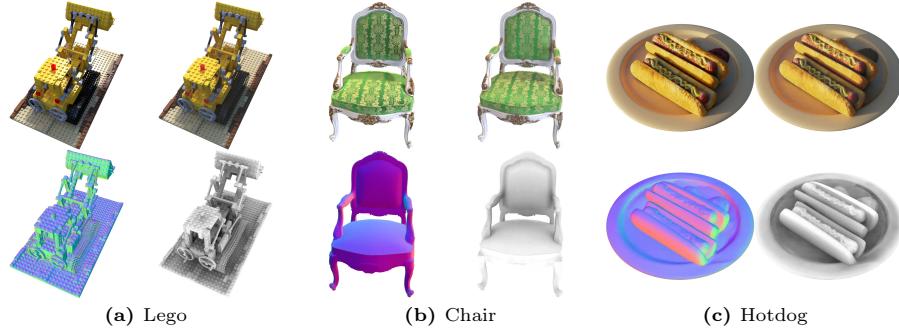
5 Experiments

5.1 Training Details

As pointed out in Sec. 4.2, the training procedure is divided into two stages. The first stage follows the setting of 3DGS [26], along with the proposed normal gradient-based densification (Sec. 3.2) where T_n is set as 2×10^{-9} . In the second stage, we sample $N_s = 64$ incident rays per Gaussian point for PBR. We train

Table 1: Quantitative results for novel view synthesis on NeRF synthetic dataset

	Geometry	Relightable	PSNR ↑	SSIM ↑	LPIPS ↓
NPBG [1]	point	✗	28.10	0.923	0.077
NPBG++ [38]	point	✗	28.12	0.928	0.076
FreqPCR [54]	point	✗	31.24	0.950	0.049
3DGS [26]	point	✗	33.88	0.970	0.031
PhySG [51]	neural	✓	18.91	0.847	0.182
NeILF++ [49]	neural	✓	26.37	0.911	0.091
Nvdiffrc [33]	mesh	✓	29.05	0.939	0.081
R3DG(Ours)	point	✓	31.22	0.959	0.039

**Fig. 4: Visualizations on NeRF synthetic dataset** [31]. Each scene is displayed in an order from left to right and from top to bottom: Ground Truth, PBR Image, Normal Map and Ambient Occlusion Map.

our model for 30,000 iterations in the initial stage and 10,000 iterations in the second stage. The loss in the first stage is calculated by:

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_{ssim} \mathcal{L}_{ssim} + \lambda_n \mathcal{L}_n + \lambda_{s,n} \mathcal{L}_{s,n} + \lambda_O \mathcal{L}_O + \lambda_u \mathcal{L}_u, \quad (16)$$

where $\{\lambda_1, \lambda_{ssim}, \lambda_n, \lambda_{s,n}, \lambda_O, \lambda_u\}$ are set to $\{0.8, 0.2, 0.01, 0.01, 0.01, 0.01\}$, respectively. The loss in the second stage is given by:

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_{ssim} \mathcal{L}_{ssim} + \lambda_l \mathcal{L}_l + \lambda_{s,b} \mathcal{L}_{s,b} + \lambda_{s,r} \mathcal{L}_{s,r}, \quad (17)$$

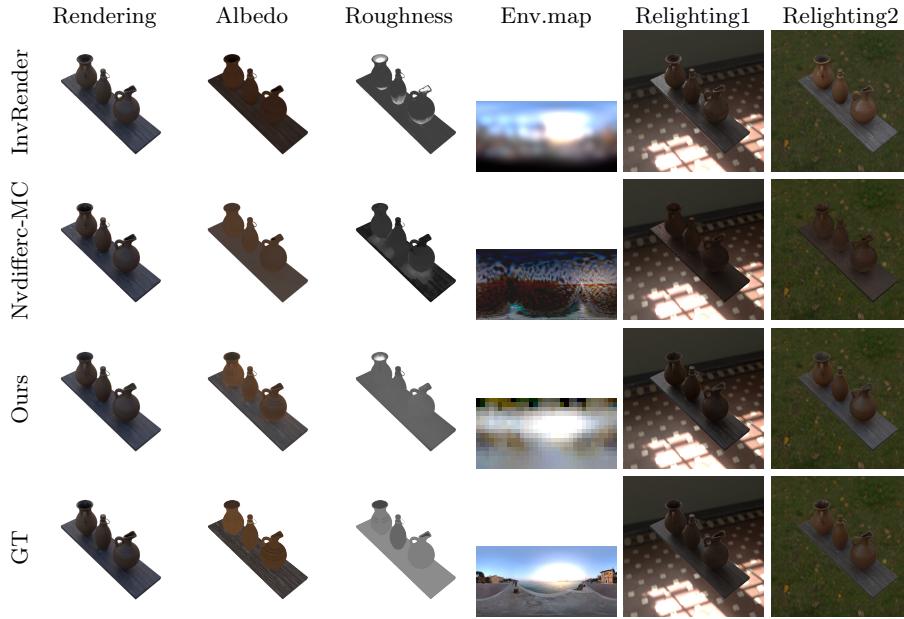
where $\{\lambda_1, \lambda_{ssim}, \lambda_l, \lambda_{s,b}, \lambda_{s,r}\}$ are set to $\{0.8, 0.2, 0.0001, 0.01, 0.01\}$, respectively.

5.2 Performance on Novel View Synthesis

We first evaluate the novel view synthesis (NVS) of the physically based rendering (PBR) on NeRF synthetic dataset [31]. We compare both *point-based* rendering methods and *relightable* rendering approaches. We report average metrics across all scenes in Tab. 1, including peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and learned perceptual image patch similarity (LPIPS). Compared with existing *point-based* rendering methods, our

Table 2: Quantitative results on Synthetic4Relight dataset [56]

	Novel View Synthesis			Relighting			Albedo		Roughness		Time (hours)
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	
NerFactor [53]	22.80	0.916	0.150	21.54	0.875	0.171	19.49	0.864	0.206	N/A	>48
Nvdiffer-MC [19]	34.29	0.967	0.068	24.22	0.943	0.078	29.61	0.945	0.075	0.009	4.17
InvRender [56]	30.74	0.953	0.086	28.67	0.950	0.091	28.28	0.935	0.072	0.008	14.3
TensorIR [23]	35.80	0.978	0.049	29.69	0.951	0.079	30.58	0.946	0.065	0.015	3.24
R3DG (Ours)	36.80	0.982	0.028	31.00	0.964	0.050	28.31	0.951	0.058	0.013	0.90

**Fig. 5:** Qualitative results on Synthetic4Relight dataset [56].

R3DG demonstrates superiority over most of them [1, 38, 54]. While our performance slightly marginally trails that of vanilla 3DGS [26], it is noteworthy that our approach excels in relighting capability, presenting a significant advantage. Furthermore, in comparison to other *relightable* methods [33, 49, 51], our method showcases significantly better NVS quality.

In Fig. 4, we visualize the our reconstruction results in Nerf synthetic dataset, including the PBR image, the normal map, and the pre-computed visibility illustrated in the form of the Ambient Occlusion (AO) map. As depicted in Fig. 4, our approach successfully achieves realistic PBR rendering, smooth normal estimation, and accurate visibility solving on discrete point clouds.

5.3 Performance on Relighting

To comprehensively assess the relighting capabilities of our pipeline, we further undertook experiments using the Synthetic4Relight dataset [56]. Recognizing



Fig. 6: Qualitative results of relighting on real-world scenes.

the inherent scale ambiguity between the estimated albedo and lighting, we standardized the scale of the estimated albedo against the ground truth to relighting, consistent with previous studies [51, 56].

Our evaluation encompasses the analysis of decomposed materials, the synthesis of novel views, and the relighting outcomes. The quantitative analysis is presented in Tab. 2. Our method outperforms existing approaches in terms of Novel View Synthesis (NVS) and relighting precision. Regarding material estimation, our method exhibits superior albedo accuracy in SSIM and LPIPS. Qualitatively, our method achieves visually pleasing material decomposition, facilitating a realistic relighting effect, which is shown in 5.

Additionally, we conduct relighting on real-world scenes in the Mip-NeRF 360 dataset [4], as shown in Fig. 6. It is worth noting that our relighting in the *kitchen* with the second environment map, exhibits pleasing shadow effects.

5.4 Ablation Study

We conduct ablation studies on three principal components of our method that significantly influence the quality of inverse rendering. The results are illustrated in Fig. 7. Initially, we investigate the impact of normal gradient densification (NGD). Fig. 7 (a, b) depict the rendering normal maps with and without the densification. It shows that the proposed densification markedly enhances the details of the normal map for *Lego*. In Fig. 7 (c, d, e), we examine the effects of our full light modeling. We simplify the full model to a global environment map in the optimization. The result demonstrates that the simplified model fails to generate shadow effects when relighting. Furthermore, we evaluate the significance of the proposed constraint on depth distribution, denoted as \mathcal{L}_u . The depth uncertainty is visualized in Fig. 7 (h, i), which illustrates the distribution

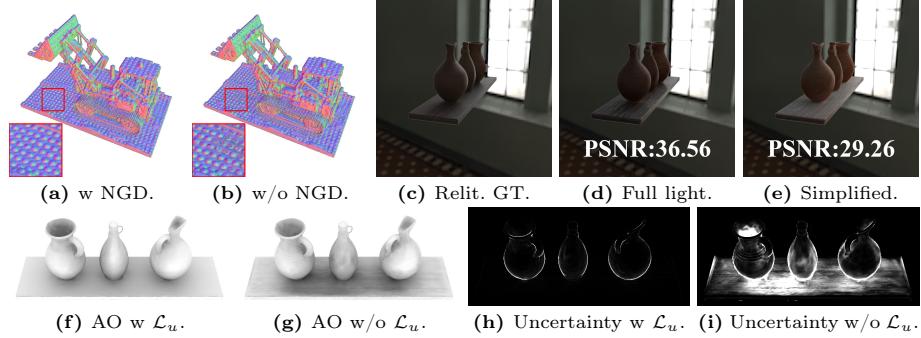


Fig. 7: Ablation studies on main components of our method. (a) and (b) show normal maps with and without the proposed normal gradient densification (NGD). (d) and (e) display re-lighting results using the proposed full lighting model and only a simplified light modeling, i.e. a global environment map. (f) and (g) visualize ambient occlusion (AO) maps with and without the proposed constraint on depth distribution \mathcal{L}_u . (h) and (i) illustrate the depth uncertainty with and without \mathcal{L}_u .

constraint contribute significantly to reducing the depth uncertainty. Additionally, Fig. 7 (f, g) reveals that incorporating \mathcal{L}_u leads to more plausible ambient occlusion (averaged visibility) which plays a crucial role in stage 2.

6 Conclusion

In this paper, we have introduced a differentiable point-based rendering pipeline enabling effort less editing, accuracy ray tracing, and photo-realistic relighting. We present the scene as *Relightable 3D Gaussians*, an extension of traditional 3D Gaussians enriched with supplementary normals, BRDFs and indirect lighting. To reconstruct a relightable scene from multi-view images, we build a novel differentiable rendering pipeline based on the inverse rendering techniques and 3D Gaussian Splatting. Additionally, we introduce an innovative ray tracing scheme specifically designed for scenes represented by discrete points, providing precise visibility for realistic relighting. Quantitative and qualitative experiments confirm that our pipeline successfully reconstructs reasonable normals, materials for discrete point clouds, and achieves commendable accuracy in both novel view synthesis and scene relighting tasks.

Limitations and Future work. Our present pipeline targets the reconstruction of static objects. Certain design choices pose challenges in maintaining its performance with large scale scenes. The high density of points in large scene can slow down optimization as we sample rays and perform PBR at each point. This can be mitigated through deferred rendering technique. Regarding geometry, integrating MVS shows promise for achieving more accurate representations.

Acknowledgements

This work was supported by National Key R&D Program of China (2023YFB3209702), and NSFC (62441204).

References

1. Aliev, K.A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural point-based graphics. In: ECCV (2020)
2. Azinovic, D., Li, T.M., Kaplanyan, A., Nießner, M.: Inverse path tracing for joint material and lighting estimation. In: CVPR (2019)
3. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: ICCV (2021)
4. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: CVPR (2022)
5. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. arXiv preprint (2023)
6. Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Neural reflectance fields for appearance acquisition. arXiv preprint (2020)
7. Bi, S., Xu, Z., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. In: ECCV (2020)
8. Bi, S., Xu, Z., Sunkavalli, K., Kriegman, D., Ramamoorthi, R.: Deep 3d capture: Geometry and reflectance from sparse multi-view images. In: CVPR (2020)
9. Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.: Nerd: Neural reflectance decomposition from image collections. In: ICCV (2021)
10. Boss, M., Engelhardt, A., Kar, A., Li, Y., Sun, D., Barron, J., Lensch, H., Jampani, V.: Samurai: Shape and material from unconstrained real-world arbitrary image collections. In: NeurIPS (2022)
11. Burley, B., Studios, W.D.A.: Physically-based shading at disney. In: SIGGRAPH (2012)
12. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: ECCV (2022)
13. Chen, Z., Funkhouser, T., Hedman, P., Tagliasacchi, A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In: CVPR (2023)
14. Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-supervised nerf: Fewer views and faster training for free. In: CVPR (2022)
15. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022)
16. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. In: ICCV (2021)
17. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. arXiv preprint arXiv:2311.12775 (2023)
18. Guo, K., Lincoln, P., Davidson, P., Busch, J., Yu, X., Whalen, M., Harvey, G., Orts-Escalano, S., Pandey, R., Dourgarian, J., et al.: The relightables: Volumetric performance capture of humans with realistic relighting. ACM TOG (2019)

19. Hasselgren, J., Hofmann, N., Munkberg, J.: Shape, light, and material decomposition from images using monte carlo rendering and denoising. In: NeurIPS (2022)
20. Hasselgren, J., Hofmann, N., Munkberg, J.: Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. arXiv:2206.03380 (2022)
21. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. In: ICCV (2021)
22. Hu, T., Liu, S., Chen, Y., Shen, T., Jia, J.: Efficientnerf efficient neural radiance fields. In: CVPR (2022)
23. Jin, H., Liu, I., Xu, P., Zhang, X., Han, S., Bi, S., Zhou, X., Xu, Z., Su, H.: Tensoir: Tensorial inverse rendering. In: CVPR (2023)
24. Kajiya, J.T.: The rendering equation. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques (1986)
25. Karras, T.: Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. In: Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics (2012)
26. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM TOG (2023)
27. Keselman, L., Hebert, M.: Flexible techniques for differentiable rendering with 3d gaussians. arXiv preprint (2023)
28. Kuang, Z., Olszewski, K., Chai, M., Huang, Z., Achlioptas, P., Tulyakov, S.: Neroic: Neural rendering of objects from online image collections. ACM TOG (2022)
29. Levoy, M., Whitted, T.: The use of points as a display primitive (1985)
30. Liu, Y., Wang, P., Lin, C., Long, X., Wang, J., Liu, L., Komura, T., Wang, W.: Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. In: SIGGRAPH (2023)
31. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
32. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM TOG (2022)
33. Munkberg, J., Hasselgren, J., Shen, T., Gao, J., Chen, W., Evans, A., Müller, T., Fidler, S.: Extracting triangular 3d models, materials, and lighting from images. In: CVPR (2022)
34. Nam, G., Lee, J.H., Gutierrez, D., Kim, M.H.: Practical svbrdf acquisition of 3d objects with unstructured flash photography. ACM TOG (2018)
35. Park, J.J., Holynski, A., Seitz, S.M.: Seeing the world in a bag of chips. In: CVPR (2020)
36. 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 (2000)
37. Pittaluga, F., Koppal, S.J., Kang, S.B., Sinha, S.N.: Revealing scenes by inverting structure from motion reconstructions. In: CVPR (2019)
38. Rakhimov, R., Ardelean, A.T., Lempitsky, V., Burnaev, E.: Npbg++: Accelerating neural point-based graphics. In: CVPR (2022)
39. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: ICCV (2021)
40. Schmitt, C., Donne, S., Riegler, G., Koltun, V., Geiger, A.: On joint estimation of pose, geometry and svbrdf from a handheld scanner. In: CVPR (2020)

41. Srinivasan, P.P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., Barron, J.T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In: CVPR (2021)
42. Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-nerf: Structured view-dependent appearance for neural radiance fields. In: CVPR (2022)
43. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint (2021)
44. Wu, H., Hu, Z., Li, L., Zhang, Y., Fan, C., Yu, X.: Nefii: Inverse rendering for reflectance decomposition with near-field indirect illumination. In: CVPR (2023)
45. Yao, Y., Zhang, J., Liu, J., Qu, Y., Fang, T., McKinnon, D., Tsin, Y., Quan, L.: Neilf: Neural incident light field for physically-based material estimation. In: ECCV (2022)
46. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. In: NeurIPS (2021)
47. Yifan, W., Serena, F., Wu, S., Öztireli, C., Sorkine-Hornung, O.: Differentiable surface splatting for point-based geometry processing. ACM TOG (2019)
48. Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A.: Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. In: NeurIPS (2022)
49. Zhang, J., Yao, Y., Li, S., Liu, J., Fang, T., McKinnon, D., Tsin, Y., Quan, L.: Neilf++: Inter-reflectable light fields for geometry and material estimation. In: ICCV (2023)
50. Zhang, K., Luan, F., Li, Z., Snavely, N.: Iron: Inverse rendering by optimizing neural sdf's and materials from photometric images. In: CVPR (2022)
51. Zhang, K., Luan, F., Wang, Q., Bala, K., Snavely, N.: Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In: CVPR (2021)
52. Zhang, Q., Baek, S.H., Rusinkiewicz, S., Heide, F.: Differentiable point-based radiance fields for efficient view synthesis. In: SIGGRAPH Asia (2022)
53. Zhang, X., Srinivasan, P.P., Deng, B., Debevec, P., Freeman, W.T., Barron, J.T.: Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. ACM TOG (2021)
54. Zhang, Y., Huang, X., Ni, B., Li, T., Zhang, W.: Frequency-modulated point cloud rendering with easy editing. In: CVPR (2023)
55. Zhang, Y., Xu, T., Yu, J., Ye, Y., Jing, Y., Wang, J., Yu, J., Yang, W.: Nemf: Inverse volume rendering with neural microflake field. In: ICCV (2023)
56. Zhang, Y., Sun, J., He, X., Fu, H., Jia, R., Zhou, X.: Modeling indirect illumination for inverse rendering. In: CVPR (2022)
57. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Surface splatting. In: Proceedings of the 28th annual conference on computer graphics and interactive techniques (2001)
58. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Ewa splatting. IEEE TVCG (2002)