RGS-DR: Deferred Reflections and Residual Shading in 2D Gaussian Splatting

Georgios Kouros Minye Wu Tinne Tuytelaars
Department of Electrical Engineering (ESAT), KU Leuven, Belgium
{georgios.kouros,minye.wu,tinne.tuytelaars}@esat.kuleuven.be

Abstract

In this work, we address specular appearance in inverse rendering using 2D Gaussian splatting with deferred shading and argue for a refinement stage to improve specular detail, thereby bridging the gap with reconstruction-only methods. Our pipeline estimates editable material properties and environment illumination while employing a directional residual pass that captures leftover view-dependent effects for further refining novel view synthesis. In contrast to per-Gaussian shading with shortest-axis normals and normal residuals, which tends to result in more noisy geometry and specular appearance, a pixel-deferred surfel formulation with specular residuals yields sharper highlights, cleaner materials, and improved editability. We evaluate our approach on rendering and reconstruction quality on three popular datasets featuring glossy objects, and also demonstrate high-quality relighting and material editing. The source code is available at https://github.com/gkouros/RGS-DR.

1. Introduction

Reconstructing and rendering glossy and reflective objects remain significant challenges in computer vision and graphics. These objects exhibit strong view-dependent effects, such as specular highlights and reflections, which undermine multi-view consistency and increase the difficulty of recovering accurate 3D geometry and appearance from multi-view images. In real-world scenarios, such objects are prevalent, including automobiles, metallic artifacts, and ceramic tableware. Achieving high-quality reconstruction and rendering of such surfaces is crucial for enhancing realism and user immersion in applications such as virtual reality (VR), augmented reality (AR), and gaming.

Recent advances in novel view synthesis (NVS) methods have demonstrated high-fidelity scene reconstruction and rendering quality when applied to objects with predominantly Lambertian reflectance. However, state-of-the-art techniques such as Neural Radiance Fields (NeRF) [24] and 3D Gaussian Splatting [15] struggle to achieve satisfactory

Figure 1. RGS-DR delivers high-quality inverse rendering by disentangling scene geometry, material properties, and illumination, enabling photorealistic rendering, relighting, and scene editing.

results on glossy and reflective surfaces. The primary limitation arises from their lack of explicit modeling of light transport. These methods rely heavily on multi-view consistency, which fails to capture sufficient cues for reconstructing complex specular effects. Consequently, artifacts such as blurring and floating structures (floaters) frequently occur.

To address these issues, inverse neural rendering aims to decompose scene properties and approximate light transport. Techniques in this area [12, 16, 18, 22, 35, 38, 43] have led to significant improvements in rendering shiny objects. A core component of these methods is the estimation of reflection directions based on surface normals, which is essential for modeling specular reflections under environmental lighting. The quality of reconstruction and rendering is highly dependent on the accuracy of these surface normals. To obtain them, Ref-NeRF [35] uses multilayer perceptrons to learn coordinate-based normals, while GaussianShader [12] employs learnable normal residuals conditioned on the viewing direction. However, both methods detach the normal estimation from the underlying geometry, which may result in noisy normals. Other methods, such as NMF [22] and ENVIDR [18], achieve better performance by using normals derived from the gradient of the geometry, while 3DGS-DR [16] relies on the shortest axis

of each Gaussian ellipsoid to estimate normals that may or may not align with the actual surface of the object. In contrast, Ref-GS [43] takes advantage of standard surfel modeling [26], adopting 2D-oriented disks as surface elements, providing an explicit normal representation that enhances the modeling of shiny objects. However, Ref-GS [43] encodes some of the object's material properties implicitly together with the ambient lighting, thus preventing relighting or other scene editing tasks.

To achieve relighting with inverse rendering, it is essential to explicitly model the environment lighting within the rendering equation. This explicit modeling enables the seamless replacement of the original lighting with the desired target lighting for effective relighting. Existing methods approximate the rendering equation using different formulations to enable this explicit modeling. A key challenge in this process is the accurate modeling of the integral of incident light from different directions. Spherical Harmonics [22] and Spherical Gaussians [35, 41] are commonly used to represent the distribution of environment lighting, as they provide closed-form solutions for approximating light integration. Other methods [12, 16, 38] utilize discrete grid textures to capture high-frequency signals in environment lighting. However, 3DGS-DR [16] fails to properly formulate the integral using only a single-level environment map and does not learn explicit material properties, which limit its expressive capability and editability. Meanwhile, GaussianShader [12], which samples from multi-level mipmaps to approximate the integral, struggles with geometric uncertainty, resulting in degraded results. Ref-Gaussian [38] reduces geometric uncertainty by utilizing the shortest axes of 2D Gaussians, but struggles to decompose diffuse from specular appearance and has to fit the diffuse color with spherical harmonics, disentangled from base color, leading to incorrect estimation of the latter.

In this paper we present *RGS-DR*, an inverse-rendering method for glossy objects explicitly scoped to direct image-based lighting (IBL). As in 2DGS [11] and related work [34, 38, 43], we use surfels to obtain reliable geometry and normals. Scene geometry and physically based rendering (PBR) parameters are optimized on each 2D Gaussian and rasterized into an image-space material buffer via a pixel-deferred pipeline. Per-pixel shading with split-sum IBL improves the discrete approximation of the specular integral and avoids per-Gaussian alpha-blend artifacts typical of forward shading [12]. To better capture view-dependent detail, we add a lightweight directional residual in image space (Ref-GS encoding [43]) that fits leftover specular effects such as glints and micro-geometry. Relighting and scene edits are then performed by swapping the estimated environment map and adjusting the recovered material maps (e.g., roughness, diffuse albedo). Empirically, RGS-DR achieves competitive reconstruction under training illumination while providing consistent environment and material estimates, and it supports high-quality relighting and editing on datasets with highly reflective objects. Our contributions can be summarized as follows:

- A 2DGS-based, inverse-rendering pipeline with deferred reflections that recovers editable materials and environment illumination.
- A refinement stage with a directional residual module that sharpens specular appearance for novel view synthesis and reduces the gap to reconstruction-only baselines.
- A detailed evaluation of rendering, reconstruction, relighting, and material editing, compared to (direct IBL) inverse rendering and reconstruction-only baselines.

2. Related Work

Novel View Synthesis. Neural Radiance Fields (NeRF) [23] offer an end-to-end framework for synthesizing photorealistic images from sets of 2D images using implicit neural representations. While NeRF offers high photorealism and compact representation, it is inefficient in training and rendering. To address this, subsequent approaches have incorporated hybrid implicit-explicit representations, such as voxel grids [19, 29, 31, 32] and hash encodings [25], which significantly enhance computational efficiency without compromising reconstruction quality. However, these methods face challenges when dealing with shiny objects due to geometric ambiguities arising from multi-view inconsistencies and their stochastic geometry modeling.

3D Gaussian Splatting (3DGS) [15] has recently emerged as a promising alternative to NeRF for 3D scene representation, delivering high-quality novel view synthesis and rapid rendering speeds. Advancements in 3DGS, such as the use of structured Gaussians [6, 21, 27], texture mapping [5], gradient cues [28, 44], and anti-aliasing [40], have further enhanced rendering quality. However, these methods often fall short in accurately reconstructing and rendering glossy or reflective surfaces due to their inability to precisely model surface-light interactions. To address these limitations, 2D Gaussian Splatting (2DGS) [11] projects Gaussian disks onto object surfaces and applies local smoothing. This approach ensures view-consistent geometry. Our method and other recent methods [34, 38, 43] leverage its explicit surface normals combined with a tailored shading function to accurately model light transport in scenes featuring shiny objects.

Inverse Neural Rendering. Inverse rendering is a challenging problem that seeks to reconstruct scene properties, including geometry, materials, and lighting, from images. Recent advances in inverse rendering have integrated differentiable volume rendering with implicit neural representations, drawing inspiration from NeRF [23], to learn scene representations encompassing both geometry and materials.

Early approaches in this area often made strong assumptions or required additional priors, such as a known illumination [2, 30], geometry [42], or varying illumination conditions [2, 10]. These limitations have been addressed in subsequent works [3, 4, 9, 13, 18, 20, 35, 41], which jointly estimate geometry, materials (e.g., BRDF), and lighting.

Gaussian-based neural rendering methods [12, 33, 36, 45] have emerged as faster, more specular-aware alternatives to field-based NeRFs by explicitly computing reflection directions and using BRDF-aware shading. Several recent works [16, 34, 38, 43] adopt a deferred formulation [8], rasterizing material buffers and shading in image space. We follow this pixel-deferred paradigm and explicitly estimate PBR material maps together with an HDR environment, performing split-sum IBL per pixel. This avoids per-primitive alpha-blended shading, which tends to smear speculars and couple errors to depth ordering (see Fig. 1). Closest to our setting, GaussianShader [12] attaches a per-Gaussian residual color modeled with low-order spherical harmonics to absorb unmodeled view dependence. In contrast, we employ a deferred residual pass with the more expressive spherical directional encoding of Ref-GS [43]. Residuals are predicted in image space (not stored per primitive), so we keep only a small per-Gaussian feature vector (much smaller than SH) for conditioning, yielding fewer parameters and lower memory at comparable cost. The residual is disabled at relight time to prevent training-illumination bias. Several methods [7, 20, 38, 39] extend split-sum IBL with additional components for indirect lighting, aiming to capture complex global illumination effects. However, these enhancements fall outside the scope of direct IBL, which is our focus. Other works [17, 34, 37] adopt fundamentally different assumptions, such as multi-light supervision or external geometric priors, making them incompatible with our setting. These directions target complementary problems and are not directly comparable to ours.

3. Preliminary

2D Gaussian Splatting [11] projects 3D Gaussians [15] onto 2D oriented planar disks, ensuring view-consistent geometry and explicit normals, defined as the direction of the steepest density change. Each planar disk serves as a 2D Gaussian primitive, characterized by a center point $\mathbf{x}$, two principal tangential vectors $\mathbf{t}_u$ and $\mathbf{t}_v$, and two variance-controlling scaling factors s_u and s_v that need to be optimized. We use the covariance matrix Σ to represent the rotation and scaling of each Gaussian. Instead of evaluating Gaussian values at the intersection of a pixel ray and a 3D Gaussian, 2DGS computes them directly on 2D disks, leveraging explicit ray-splat intersections for perspective-correct splatting. The Gaussian value is formulated as: $\mathcal{G}(\mathbf{u}(\mathbf{r})) = \exp(-(u^2+v^2)/2)$, where $\mathbf{u}(\mathbf{r}) = (u, v)$ is the intersection point between ray $\mathbf{r}$ and the disk in UV space.

For rendering, Gaussians along with their properties, such as color $\mathbf{c}_i$, are sorted by their centers and blended into pixels front to back, which is formulated by alpha blending:

$$c(\mathbf{r}) = \sum_{i=1}^{n} \mathbf{c}_i \, \alpha_i \, \mathcal{G}_i\big(\mathbf{u}(\mathbf{r})\big) \prod_{j=1}^{i-1} \Big(1 - \alpha_j \, \mathcal{G}_j\big(\mathbf{u}(\mathbf{r})\big)\Big), \quad (1)$$

where i and α_i denote the index of the Gaussian and its alpha value; $\mathbf{c}_i$ is the view-dependent appearance color for the i-th Gaussian. In our method, we need to render different object material properties (e.g. diffuse color, roughness, and normal) and feature vectors into the target view. This is achieved using alpha blending, where the principle is to replace the color $\mathbf{c}_i$ with the corresponding material properties or features, generating multi-channel pixel values.

Planar disks have well-defined surface normals that are perpendicular to their surfaces. Using the tangential vectors, we can compute the normal vectors in closed form:

$$\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v, \quad (2)$$

where $\mathbf{n}$ is the normal vector, and operator $\times$ stands for cross product. In our method, we use this explicit normal formulation in the light transport modeling, enabling accurate reflection calculations. This perspective-correct rendering improves rendering performance based on deferred shading.

4. Methodology

RGS-DR exploits surfel primitives that have accurate surface properties to compose the scene. In addition to the geometry-related variables mentioned above, each primitive has learnable properties, namely diffuse color $\mathbf{c}_d$, roughness ρ, and specular tint $\mathbf{s}$. We also assign a feature vector $\mathbf{f}$ to each primitive to capture inter-reflection.

Overview. Instead of first calculating the view-dependent color of primitives and then blending them, we employ a deferred rendering scheme. Our approach rasterizes the geometry and blends the properties into a Graphics Buffer (G-Buffer), which stores buffer images for each property (§ 4.1). The G-Buffer stores essential information for modeling light transport in reflections, including geometry, material, and lighting-related properties. It enables a representation of global illumination effects. Leveraging this data, shading functions are applied to approximate the rendering equation based on the material attributes in the G-Buffer. This process enables the computation of pixel-wise colors for the specular component while accounting for indirect lighting contributions from the surrounding environment (§ 4.2). In addition to environmental illumination, light undergoes multiple reflections between object surfaces, a phenomenon known as inter-reflection, which is not explicitly modeled in standard shading functions. To

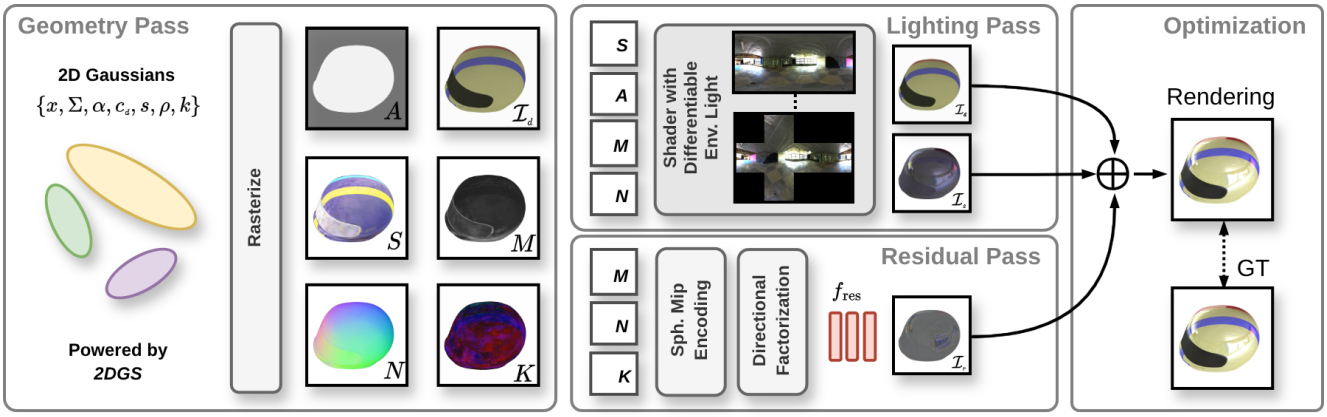

Figure 2. Our rendering pipeline consists of three passes. The geometry pass produces screen-space diffuse color $\mathcal{I}_d$, specular tint S, roughness M, normals N, and low-dimensional features K, which feed into the subsequent passes. The lighting pass employs a cube mipmap to model environmental light for shading as described in [12]. Meanwhile, the residual pass uses a spherical-mip-based directional encoding (inspired by [43]) along with a shallow MLP f_{res} to predict view-dependent effects not captured by the lighting pass.

address this limitation, we propose a residual rendering pass that integrates features from the G-Buffer with a spherical feature mip-map [43] to effectively encode inter-reflection effects (§ 4.3). We integrate these components into our proposed differentiable pipeline to generate the final rendered images. The complete pipeline is illustrated in Fig. 2.

Deferred Rendering. RGS-DR adopts the rendering equation approximation of GaussianShader [12] defined as:

$$\mathbf{c}(\boldsymbol{\omega}_o) = \mathbf{c}_d + \mathbf{s} \odot \mathbf{L}_s(\boldsymbol{\omega}_o, \mathbf{n}) + \mathbf{c}_r(\boldsymbol{\omega}_o), \tag{3}$$

where $\boldsymbol{\omega}_o \in \mathbb{R}^3$ is the viewing direction, $\mathbf{c}_d \in \mathbb{R}^3$ is the diffuse color, $\mathbf{s} \in \mathbb{R}^3$ is the specular tint, $\mathbf{L}_s \in \mathbb{R}^3$ is the specular light from the shading functions (§ 4.2), $\rho \in \mathbb{R}^1$ is the roughness, and $\mathbf{c}_r \in \mathbb{R}^3$ is a residual color term (§ 4.3) that accounts for inter-reflection effects. Note that deferred rendering computes on the image pixel level; all the input variables are fetched from the G-buffer at the corresponding pixel coordinates. The final rendered image $\mathcal{I}$ is calculated as

$$\mathcal{I} = \mathcal{I}_d + \mathcal{I}_s + \mathcal{I}_r, \tag{4}$$

where $\mathcal{I}_d$ and $\mathcal{I}_s$ are the diffuse and specular components of the scene, while $\mathcal{I}_r$ is the residual image term. The components of Eq. 3-4 will be introduced in the following subsections.

4.1. G-Buffer Rendering

We adapt the alpha blending formulation in Eq. 1 to generate buffer images by replacing $\mathbf{c}_i$ with the corresponding property vectors. Following the notation of Ref-GS[43], RGS-DR fills the G-buffer with the diffuse map $\mathcal{I}_d$, roughness map $\mathbf{M}$, specular tint map $\mathbf{S}$, feature map $\mathbf{K}$, and normal map $\mathbf{N}$, as well as the accumulated opacity $\mathbf{A}$. They have the same resolution as the render target but differ in the number of channels. Most of them are alpha-blended from corresponding properties stored in each Gaussian primitive except for the normal map $\mathbf{N}$, which is derived from the primitive geometry using Eq. 2.

4.2. Deferred Shading Functions

Shading functions are an important part of the approximation of the rendering equation [14] to model light-surface interactions. They have been used in [12], demonstrating their effectiveness in material decomposition and representation. In this work, we extend shading functions to the deferred rendering framework and apply them to the G-buffer.

We use a shading function to compute the specular component $\mathbf{L}_s$, which is formulated as:

$$\mathbf{L}_s(\boldsymbol{\omega}_o, \mathbf{n}) = \int_{\Omega} L(\boldsymbol{\omega}_i)\, f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, (\boldsymbol{\omega}_i \cdot \mathbf{n})\, d\boldsymbol{\omega}_i, \tag{5}$$

where $\boldsymbol{\omega}_i$ is the direction of the input radiance, Ω denotes the directions of the entire upper hemisphere above the current primitive disk, and f_r is the bidirectional reflectance distribution function.

We use the split-sum approximation of the integral to avoid computationally expensive explicit integral evaluations. Specifically, we achieve this using a pre-filtered cube mip-map $\mathbf{E}$ where each mip-map level stores progressively blurrier versions of the original environment map, simulating different levels of surface roughness for reflections.

In contrast to GaussianShader [12] which applies shading on each individual 3D Gaussian before splatting, we defer shading and apply it at the pixel level. This approach provides two benefits. First, we compute lighting only on the visible parts of the scene, and second, we leverage the smooth normals that are produced from 2D Gaussian primitive, which are far more accurate than the shortest normal

axes of 3D Gaussians. At the same time, we do not have to learn additional normal residuals as does GaussianShader to account for inconsistencies between surface normals and Gaussian normals. In the deferred rendering scheme, pixel shading is performed without alpha blending, resulting in high-frequency reflections.

4.3. Residual Rendering Pass

We incorporate a residual color term optimized in what we call the residual pass, as shown in Fig.2. Instead of using high-dimensional spherical harmonic coefficients per Gaussian, we employ low-dimensional features $\mathbf{k}$ and optimize a directional encoding based on the methodology of Ref-GS [43]. This involves a spherical mipmap-based directional encoding that enables roughness-aware photorealistic rendering on highly reflective scenes. However, the environment illumination in Ref-GS is baked in the mipmap, and thus learned scenes are not relightable. Instead, we leverage the high expressivity of such a directional encoding to learn scene-specific information and augment the reconstruction quality of our model. The resulting residual color term complements our explicit shading branch and accounts for missing complex view-dependent effects, such as interreflections, thus refining our method's reconstruction quality. However, relighting quality is not affected as the residual term is illumination-specific and is not included when performing relighting by rendering with new environment maps.

For a given screen-space point $\mathbf{u}(u, v)$, the spherical mip encoding can be calculated based on the viewing direction $\boldsymbol{\omega}_o$, normal $\mathbf{n}$, roughness ρ, and feature $\mathbf{k}$. The viewing direction and normal are used to estimate the reflected direction $\boldsymbol{\omega}_r$ which is then converted into polar coordinate (θ_r, ϕ_r). The spherical mip encoding is then computed by trilinear interpolation on the spherical mipmap $\mathcal{M}$ along the roughness dimension and given by $\mathbf{h} = \text{Sph-Mip}(\boldsymbol{\omega}_r, \rho, \mathcal{M})$. This feature is concatenated with the outer product $\mathbf{h} \otimes \mathbf{k}$ to form the directional encoding, which is used as input to a shallow MLP f_{res} to produce the residual color term $\mathbf{c}_r = f_{\text{res}}(\mathbf{h}, \mathbf{k} \otimes \mathbf{h})$ and the corresponding residual image

$$\mathcal{I}_r = f_{\text{res}}(\mathbf{H}, \mathbf{K} \otimes \mathbf{H}). \tag{6}$$

4.4. Training Strategy

To train our model, we follow the training strategy of 3DGS [15], which supervises the model with a photometric RGB reconstruction loss $\mathcal{L}_c$ that combines an $\mathcal{L}_1$ loss with a D-SSIM term into

$$\mathcal{L}_c = (1 - \lambda)\,\mathcal{L}_1 + \lambda\,\mathcal{L}_{\text{D-SSIM}}, \tag{7}$$

where λ is the balancing term between the two losses and is usually chosen as $\lambda = 0.2$.

However, the problem of estimating 3D geometry from a collection of posed 2D images using only a photometric loss is ill-posed. As a result, we additionally adopt the regularization methodology of 2DGS [11]. This regularization involves the normal consistency regularization from Ref-NeRF [35], given in Eq. 8 and the depth distortion regularization from MipNeRF360 [1], given in eq. 9. The former tries to enforce consistency between the predicted normal map and the normals computed by depth gradients and is formulated as

$$\mathcal{L}_n = \sum_i w_i \big(1 - \mathbf{n}_i^T \mathbf{N}\big), \tag{8}$$

where i, w_i, $\mathbf{n}_i$, and N correspond to a ray-splat intersection, its blending weight, its normal, and the local depth normal calculated using finite differences from nearby depth points. The latter regularizes the weight distribution of the splats to concentrate them around the surface by minimizing the surface-splat intersection. This is accomplished through

$$\mathcal{L}_d = \sum_{i,j} w_i\, w_j\, \big|z_i - z_j\big|, \tag{9}$$

where z_i is the depth of the i-th ray-splat intersection.

Furthermore, we use an alpha loss to prevent floaters caused by redundant Gaussians in synthetic scenes:

$$\mathcal{L}_\alpha = \big\|A - A_{gt}\big\|_1, \tag{10}$$

where A is the accumulated alpha map in the G-buffer and A_{gt} is the corresponding ground-truth alpha channel of the provided RGBA image.

The total loss is calculated as a weighted sum of the individual loss terms as

$$\mathcal{L} = \mathcal{L}_c + \lambda_d \mathcal{L}_d + \lambda_n \mathcal{L}_n + \lambda_\alpha \mathcal{L}_\alpha, \tag{11}$$

where λ_d, λ_n, and λ_α are the corresponding loss weights.

5. Experimental Results

5.1. Datasets

We compare our method against state-of-the-art baselines on two synthetic datasets and one real dataset with complex, view-dependent effects resulting from highly reflective surfaces. Specifically, we evaluate on the Shiny Synthetic [35], Glossy Synthetic [20], and Shiny Real [35]. Results are provided for all scenes in the Shiny Synthetic and Shiny Real datasets. For the Glossy Synthetic dataset, we select the same subset of scenes as in [16, 43].

5.2. Implementation Details

All our models are trained on an NVIDIA Tesla P100 GPU, completing 35k iterations in two hours on average. For

the lighting pass, we use an environment cubemap of size $6\times512\times512$ with 3 RGB channels, and for the residual pass an $8 \times 512 \times 512$ mipmap with 16 feature channels, along with a shallow MLP (two hidden layers of 256 units) plus one 4D feature per Gaussian. For the real scenes, we downsample the training images of the gardenspheres and toycar scenes by factors of 4 and the sedan scene by a factor of 8 to be consistent with [16, 43]. Furthermore, we employ a similar bounding volume as in [16, 43] to allow for reflective surfaces only in the foreground object. We achieve this by penalizing low roughness properties in Gaussians outside of the bounding volume. We use loss weights of $\lambda_n = 0.05$, $\lambda_d = 100$, and $\lambda_\alpha = 1.0$. Material properties are optimized with a learning rate of 2.5×10^{-3}, while opacity, scale, and rotation use 0.03, 3×10^{-3}, and 10^{-2}, respectively. The environment cube mipmap is updated with an exponentially decaying rate from 10^{-2} to 10^{-3}. Gaussian positions are trained as in 2DGS [11], with a decay from 1.6×10^{-4} to 1.6×10^{-6}. We train 30k iters without the residual, then enable it and freeze geometry/materials/lighting for a 5k-iter fine-tune. Only the residual branch is updated, benefiting NVS but not affecting editing.

5.3. Comparison

In Table 1 and Fig. 3, we present the quantitative and qualitative evaluation of our proposed methodology against state-of-the-art novel view synthesis and inverse rendering methods. As shown in the table, we outperform the competing methods on several scenes of the examined datasets. Specifically, our method appears superior on the most glossy objects such as car, helmet, and toaster, as well as the entire Glossy Synthetic dataset. Our results in Fig. 3 highlight the ability of our method to reconstruct sharper reflections with less blurring and distortions than the two main competing inverse rendering methods, while being competitive with the reconstruction-only Ref-GS.

In Fig. 4, we present the decomposition of our method's output, including the material properties, smooth geometry (visualized through surface normals), as well as the estimated specular components and residuals. As demonstrated, our method effectively disentangles the geometry, material properties, and illumination of a given scene, thus enabling photorealistic novel view synthesis, relighting, and scene editing. Unlike 3DGS-DR, our method learns explicit material properties and thus trained scenes can be easily ported to existing 3D rendering engines.

In Fig. 5, we present our method's estimated environment illumination for scenes from the Shiny Synthetic dataset [35]. Our method manages to effectively disentangle the illumination from the material properties and geometry of the scene. Compared to 3DGS-DR and GaussianShader, we obtain more refined environment maps with sharper details and fewer artifacts. A quantitative evaluation is provided in Table 2, containing the surface normal error and environment map quality for Shiny Synthetic.

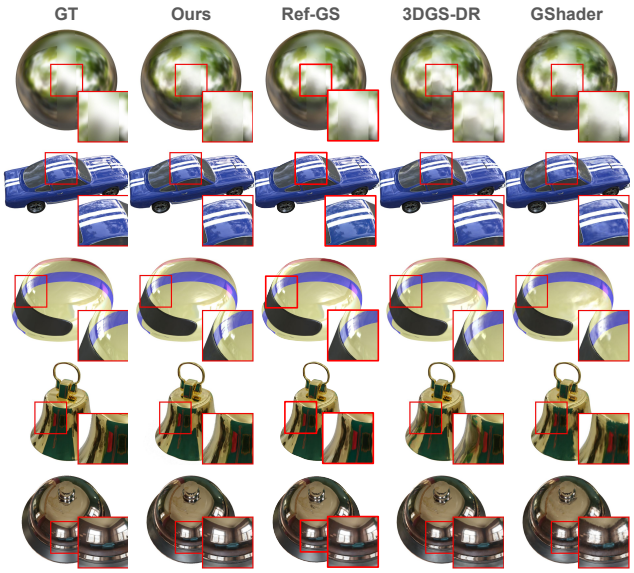

Figure 3. Comparison of the rendering quality of our method against Ref-GS [43], 3DGS-DR [16] and GaussianShader [12].

5.4. Ablation Study

In Table 3, we conduct an ablation study to validate our design choices. First, we replace the cube mipmap of the environment light with a single cubemap to compare with 3DGS-DR which uses a single level. We observe a considerable drop in performance, demonstrating the necessity of a multi-resolution cubemap for accurate estimates of environment illumination. Second, we ablate the residual pass and once again notice a drop in performance, which highlights its benefit to photorealistic NVS. Finally, we replace the residual pass with spherical-harmonics-based residuals like in [12], which improves performance but not as much as our proposed residual pass.

5.5. Relighting and Scene Editing

In Fig. 6, we present the relighting capabilities of our model for two scenes from the Shiny Synthetic dataset [35] and two scenes from the Glossy Synthetic dataset [20]. To relight the scenes, we replace the learned environment map and convert it to a cube mipmap before rendering. As demonstrated, our method can effectively relight learned scenes and achieve photorealistic results even under new illuminations. Additional relighting results can be found in the supplementary material. In Fig. 7, on the other hand, we demonstrate the scene editing capabilities of our model on three glossy objects by modifying their roughness to make them rougher or smoother, and by changing their diffuse color.

Table 1. Quantitative results comparing our method with state-of-the-art approaches on test views from the Shiny Synthetic, Shiny Real, and Glossy Synthetic datasets. Metrics from [16, 43] are used, with the top three methods for each scene and metric marked as first, second, and third. The final columns present the average scores for each dataset. The second column indicates whether a method supports relighting (✓) or is only reconstruction-capable (✗).

	Relight	Shiny Synthetic [35]							Glossy Synthetic [20]							Shiny Real [35]			
		ball	car	coffee	helmet	teapot	toaster	*avg*	bell	cat	luyu	potion	tbell	teapot	*avg*	garden	sedan	toycar	*avg*
PSNR ↑																			
Ref-NeRF	✗	33.16	30.44	33.99	29.94	45.12	26.12	33.13	30.02	29.76	25.42	30.11	26.91	22.77	27.50	22.01	25.21	23.65	23.62
3DGS	✗	27.65	27.26	32.30	28.22	45.71	20.99	30.36	25.11	31.36	26.97	30.16	23.88	21.51	26.50	21.75	26.03	23.78	23.85
GShader	✓	30.99	27.96	32.39	28.32	45.86	26.28	31.97	28.07	31.81	27.18	30.09	24.48	23.58	27.54	21.74	24.89	23.76	23.46
ENVIDR	✓	41.02	27.81	30.57	32.71	42.62	26.03	33.46	30.88	31.04	28.03	32.11	28.64	26.77	29.58	21.47	24.61	22.92	23.00
3DGS-DR	✓	33.66	30.39	34.65	31.69	47.12	27.02	34.09	31.65	33.86	28.71	32.79	28.94	25.36	30.22	21.82	26.32	23.83	23.99
Ref-GS	✗	36.10	30.94	34.38	33.40	46.69	27.28	34.80	31.70	33.15	29.46	32.64	30.08	26.47	30.58	22.48	26.63	24.20	24.44
Ours	✓	37.76	31.05	33.54	33.97	46.29	27.54	35.02	32.15	32.95	29.74	33.10	30.54	26.61	30.85	23.19	26.25	24.45	24.63
SSIM ↑																			
Ref-NeRF	✗	0.971	0.950	0.972	0.954	0.995	0.921	0.961	0.941	0.944	0.901	0.933	0.947	0.897	0.927	0.584	0.720	0.633	0.646
3DGS	✗	0.937	0.931	0.972	0.951	0.996	0.894	0.947	0.892	0.959	0.916	0.938	0.908	0.881	0.916	0.571	0.771	0.637	0.660
GShader	✓	0.966	0.932	0.971	0.951	0.996	0.929	0.958	0.919	0.961	0.914	0.938	0.898	0.901	0.922	0.576	0.728	0.637	0.647
ENVIDR	✓	0.997	0.943	0.962	0.987	0.995	0.990	0.979	0.954	0.965	0.931	0.960	0.947	0.957	0.952	0.561	0.707	0.549	0.606
3DGS-DR	✓	0.979	0.962	0.976	0.971	0.997	0.943	0.971	0.962	0.976	0.936	0.957	0.952	0.936	0.953	0.581	0.773	0.639	0.664
Ref-GS	✗	0.981	0.961	0.973	0.975	0.997	0.950	0.973	0.965	0.973	0.946	0.957	0.956	0.944	0.957	0.507	0.783	0.682	0.657
Ours	✓	0.988	0.967	0.971	0.977	0.997	0.946	0.974	0.964	0.973	0.952	0.964	0.969	0.951	0.962	0.638	0.769	0.687	0.698
LPIPS ↓																			
Ref-NeRF	✗	0.166	0.050	0.082	0.086	0.012	0.083	0.080	0.102	0.104	0.098	0.084	0.114	0.098	0.100	0.251	0.234	0.231	0.239
3DGS	✗	0.162	0.047	0.079	0.081	0.008	0.125	0.084	0.104	0.062	0.064	0.093	0.125	0.102	0.092	0.248	0.206	0.237	0.230
GShader	✓	0.121	0.044	0.078	0.074	0.007	0.079	0.067	0.098	0.056	0.064	0.088	0.122	0.091	0.087	0.274	0.259	0.239	0.257
ENVIDR	✓	0.020	0.046	0.083	0.036	0.009	0.081	0.046	0.054	0.049	0.059	0.072	0.069	0.041	0.057	0.263	0.387	0.345	0.332
3DGS-DR	✓	0.098	0.033	0.076	0.049	0.005	0.081	0.057	0.064	0.040	0.053	0.075	0.067	0.067	0.061	0.247	0.208	0.231	0.229
Ref-GS	✗	0.098	0.034	0.082	0.045	0.006	0.070	0.056	0.049	0.041	0.046	0.076	0.073	0.064	0.058	0.242	0.196	0.236	0.225
Ours	✓	0.080	0.032	0.084	0.050	0.007	0.097	0.059	0.060	0.038	0.042	0.057	0.049	0.050	0.049	0.264	0.225	0.263	0.251

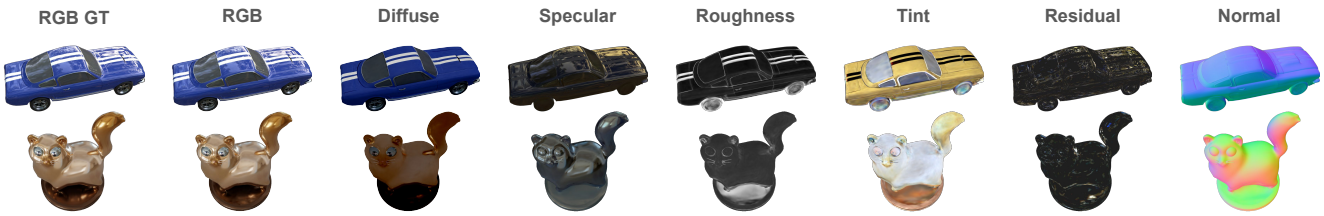

Figure 4. Outputs of our model including renderings, material properties, residuals and surface normals for scenes from the Shiny Synthetic and Glossy Synthetic datasets. The residuals are amplified for visualization purposes.

Table 2. Quantitative comparison of surface normals via MAE° and environment map recovery via E-PSNR, E-SSIM, and E-LPIPS on the Shiny Synthetic dataset [35].

	MAE° ↓	E-PSNR ↑	E-SSIM ↑	E-LPIPS ↓
GShader	6.735	11.06	0.142	0.644
3DGS-DR	2.266	11.89	0.339	0.599
Ref-GS	2.210	-	-	-
Ours	**1.917**	**13.28**	**0.403**	**0.563**

Table 3. Ablation study on the Shiny Synthetic dataset [35]. We ablate the environment map mipmapping, the residual pass, and also evaluate an alternative residual pass with spherical harmonics.

	PSNR↑	SSIM ↑	LPIPS↓
w/o mipmap	33.89	0.968	0.068
w/o residual	34.28	**0.974**	**0.058**
w/ SH-residual	34.68	0.973	0.059
Ours	**35.02**	**0.974**	0.059

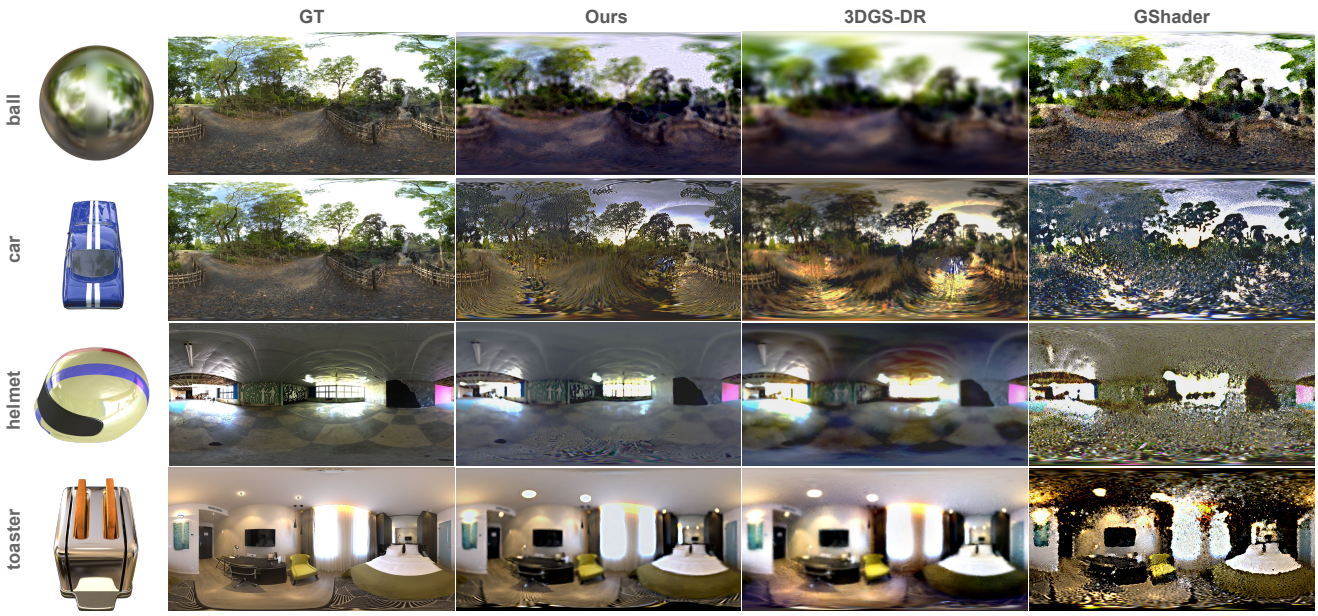

Figure 5. Comparison of estimated environment maps of synthetic scenes from the Shiny Synthetic dataset [35].

Figure 6. Relighted scenes with GT on the left and utilized environment map on the top.

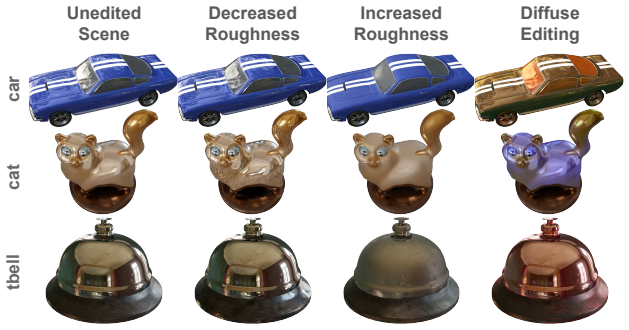

Figure 7. Demonstration of the material editing capabilities of our method.

6. Conclusion

This work tackles inverse rendering of glossy objects with complex view-dependent effects using a 2D Gaussian, pixel-deferred formulation. Our method estimates editable material properties and environment illumination via split-sum IBL, and adds a refinement stage with a lightweight directional residual that captures leftover specular detail, further enhancing novel view synthesis and bridging the gap with reconstruction-only methods. Across challenging reflective scenes, we observe sharper highlights, plausible material estimates, and higher relighting fidelity than per-Gaussian shading baselines or uniform roughness baselines. Our scope is limited to direct IBL since we do not model multi-bounce indirect lighting, transparency, or subsurface scattering. Scenes dominated by strong near-field interreflections are better addressed by global-illumination-enabled variants.

Acknowledgments

This work was supported by the Flanders AI Research Program and the KU Leuven Methusalem project ”Lifelines”. The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 5

[2] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. In *European Conference on Computer Vision (ECCV)*, page 294–311, Berlin, Heidelberg, 2020. Springer-Verlag. 3

[3] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 3

[4] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik P.A. Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[5] Brian Chao, Hung-Yu Tseng, Lorenzo Porzi, Chen Gao, Tuotuo Li, Qinbo Li, Ayush Saraf, Jia-Bin Huang, Johannes Kopf, Gordon Wetzstein, et al. Textured gaussians for enhanced 3d scene appearance modeling. *arXiv preprint arXiv:2411.18625*, 2024. 2

[6] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2

[7] Hongze Chen, Zehong Lin, and Jun Zhang. Gi-gs: Global illumination decomposition on gaussian splatting for inverse rendering. In *ICLR*, 2025. 3, 1

[8] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. *Acm siggraph computer graphics*, 22(4):21–30, 1988. 3

[9] Yue Fan, Ningjing Fan, Ivan Skorokhodov, Oleg Voynov, Savva Ignatyev, Evgeny Burnaev, Peter Wonka, and Yiqun Wang. Factored-neus: Reconstructing surfaces, illumination, and materials of possibly glossy objects. *arXiv preprint arXiv:2305.17929*, 2025. 3

[10] Duan Gao, Guojun Chen, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deferred neural lighting: free-viewpoint relighting from unstructured photographs. *ACM Transactions on Graphics (TOG)*, 39(6):258, 2020. 3

[11] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. 2, 3, 5, 6

[12] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5322–5332, 2024. 1, 2, 3, 4, 6

[13] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. Tensoir: Tensorial inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[14] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986. 4

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2, 3, 5

[16] Ye Keyang, Hou Qiming, and Zhou Kun. 3d gaussian splatting with deferred reflection. *ACM SIGGRAPH Conference Proceedings, Denver, CO, United States, July 28 - August 1, 2024*, 2024. 1, 2, 3, 5, 6, 7, 4

[17] Jingzhi Li, Zongwei Wu, Eduard Zamfir, and Radu Timofte. Recap: Better gaussian relighting with cross-environment captures. In *CVPR*, 2025. 3

[18] Ruofan Liang, Huiting Chen, Chunlin Li, Fan Chen, Selvakumar Panneer, and Nandita Vijaykumar. Envidr: Implicit differentiable renderer with neural environment lighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 79–89, 2023. 1, 3

[19] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 2

[20] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. *ACM Trans. Graph.*, 42(4), 2023. 3, 5, 6, 7, 1, 2, 4

[21] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 2

[22] Alexander Mai, Dor Verbin, Falko Kuester, and Sara Fridovich-Keil. Neural microfacet fields for inverse rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 408–418, 2023. 1, 2

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

[24] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1

[25] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 2

[26] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Com-*

puter graphics and interactive techniques, pages 335–342, 2000. 2

[27] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898*, 2024. 2

[28] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. In *European Conference on Computer Vision*, pages 347–362. Springer, 2024. 2

[29] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2

[30] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021. 3

[31] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2

[32] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Improved direct voxel grid optimization for radiance fields reconstruction. *arXiv preprint arXiv:2206.05085*, 2022. 2

[33] Zhe Jun Tang and Tat-Jen Cham. 3igs: Factorised tensorial illumination for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 143–159. Springer, 2024. 3

[34] Jinguang Tong, Xuesong Li, Fahira Afzal Maken, Sundaram Muthu, Lars Petersson, Chuong Nguyen, and Hongdong Li. Gs-2dgs: Geometrically supervised 2dgs for reflective object reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21547–21557, 2025. 2, 3

[35] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 1, 2, 3, 5, 6, 7, 8

[36] Tong Wu, Jia-Mu Sun, Yu-Kun Lai, Yuewen Ma, Leif Kobbelt, and Lin Gao. Deferredgs: Decoupled and editable gaussian splatting with deferred shading. *arXiv preprint arXiv:2404.09412*, 2024. 3

[37] Tao Xie, Xi Chen, Zhen Xu, Yiman Xie, Yudong Jin, Yujun Shen, Sida Peng, Hujun Bao, and Xiaowei Zhou. Envgs: Modeling view-dependent appearance with environment gaussian. *arXiv preprint arXiv:2412.15215*, 2024. 3

[38] Yuxuan Yao, Zixuan Zeng, Chun Gu, Xiatian Zhu, and Li Zhang. Reflective gaussian splatting. In *ICLR*, 2025. 1, 2, 3

[39] Mae Younes and Adnane Boukhayma. Texturesplat: Per-primitive texture mapping for reflective gaussian splatting, 2025. 3, 1

[40] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024. 2

[41] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5453–5462, 2021. 2, 3

[42] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. Nerfactor: neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.*, 40(6), 2021. 3

[43] Youjia Zhang, Anpei Chen, Yumin Wan, Zikai Song, Junqing Yu, Yawei Luo, and Wei Yang. Ref-gs: Directional factorization for 2d gaussian splatting. *arXiv preprint arXiv:2412.00905*, 2024. 1, 2, 3, 4, 5, 6, 7

[44] Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 326–342. Springer, 2024. 2

[45] Zuo-Liang Zhu, Beibei Wang, and Jian Yang. Gs-ror: 3d gaussian splatting for reflective object relighting via sdf priors. *arXiv preprint arXiv:2406.18544*, 2024. 3