Editable Physically-based Reflections in Raytraced Gaussian Radiance Fields

YOHAN POIRIER-GINTER, Université Laval, Canada and Inria, Université Côte d'Azur, France
JEFFREY HU, Inria, Université Côte d'Azur, France
JEAN-FRANÇOIS LALONDE, Université Laval, Canada
GEORGE DRETTAKIS, Inria, Université Côte d'Azur, France

Fig. 1. Our Gaussian-based radiance-field method allows interactive editing of path traced reflections, with consistent updates. We develop our proof-of-concept method in synthetic scenes with known ground truth inputs and show its extension to a real scene with network predicted inputs.

Radiance fields such as 3D Gaussian Splatting allow real-time rendering of scenes captured from photos. They also reconstruct most specular reflections with high visual quality, but typically model them with "fake" reflected geometry, using primitives behind the reflector. Our goal is to correctly reconstruct the reflector and the reflected objects such as to make specular reflections editable; we present a proof of concept which exploits promising learning-based methods to extract diffuse and specular buffers from photos, as well as geometry and BRDF buffers. Our method builds on three key components. First, by using diffuse/specular buffers of input training views, we optimize a diffuse version of the scene and use path tracing to efficiently generate physically-based specular reflections. Second, we present a specialized training method that allows this process to converge. Finally, we present a fast ray tracing algorithm for 3D Gaussian primitives that enables efficient multi-bounce reflections. Our method reconstructs reflectors and reflected objects—including those not seen in the input images—in a *unique* scene representation. Our solution allows real-time, consistent editing of captured scenes with specular reflections, including multi-bounce effects, changing roughness etc. We mainly show results using ground truth buffers from synthetic scenes, and also preliminary results in real scenes with currently imperfect learning-based buffers. Code and data are available at: https://repo-sam.inria.fr/nerphys/editable-gaussian-reflections/.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Reconstruction**.

Additional Key Words and Phrases: Gaussian splatting, differentiable rendering, path tracing

ACM Reference Format:
Yohan Poirier-Ginter, Jeffrey Hu, Jean-François Lalonde, and George Drettakis. 2025. Editable Physically-based Reflections in Raytraced Gaussian Radiance Fields. In *SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25), December 15–18, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3757377.3763971

1 INTRODUCTION

Neural Radiance Fields (NeRFs) [Mildenhall et al. 2020] and 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] allow high-quality novel-view synthesis with only posed photographs as input. One of their major strengths is that they render view-dependent effects such as low- to medium-frequency specular reflections with high visual quality. However, the representation of reflections is entangled with the geometry: they are baked into a directional radiance component, and often are actually "fake mirror geometry" situated behind the reflective surfaces [Meng et al. 2024; Zhang et al. 2024]. This precludes consistent editing of scenes with reflections, i.e., where reflections will be correctly updated when changing geometry or materials. We propose the first method that allows interactive reflection-consistent editing of radiance fields (Fig. 1) by using distinct processes to reconstruct each of the diffuse and reflective components of the scene.

Authors' addresses: Yohan Poirier-Ginter, Université Laval, Quebec, Canada and Inria, Université Côte d'Azur, Nice, France, yohan.poirier-ginter.1@ulaval.ca; Jeffrey Hu, Inria, Université Côte d'Azur, Nice, France, hujh14@gmail.com; Jean-François Lalonde, Université Laval, Quebec, Canada, jean-francois.lalonde@gel.ulaval.ca; George Drettakis, Inria, Université Côte d'Azur, Nice, France, george.drettakis@inria.fr.

SA Conference Papers '25, December 15–18, 2025, Hong Kong, Hong Kong

Fig. 2. Our method takes as input several buffers for every training view (left), and reconstructs a unique gaussian-based scene where reflections in novel views are computed with cached diffuse (middle) and physics-based path tracing (right). This figure shows ground truth input buffers from a synthetic scene, from top to bottom: separated targets (diffuse and non-diffuse), BRDF parameters (base reflectance and roughness), and geometry (depth and normals).

Our method also allows reconstruction of reflected objects unseen in the input images.

In the original versions of NeRF and 3DGS, a directional radiance representation is modeled by a Multi-Layer Perceptron (MLP) and Spherical Harmonics (SH) respectively. However, these low-frequency representations cannot represent reflections accurately, and thus the optimization favors the creation of "fake mirror geometry" (see video). Several improvements exist for better reflection in NeRFs, e.g., by changing parameterizations or encodings [Ma et al. 2024; Verbin et al. 2022]. Similarly, several methods attempt to improve reflections in 3DGS, often for distant lighting [Jiang et al. 2024; Ye et al. 2024]. These methods still suffer from the entangled reflection representation. Recently, NeRFcasting [Verbin et al. 2024] and EnvGS [Xie et al. 2025] use ray tracing but require a separate scene representation for reflections to allow stable optimization. Representing reflectors and reflected objects as separate scenes prevents consistent editing of the radiance field.

To address these problems, our key intuition is to have a unique scene with two distinct—but concurrent—optimizations: one for the diffuse, and one for the specular components of the scene, supervised on diffuse and specular input buffers respectively. In this paper, we use the term *specular* to mean all non-diffuse reflections, from rough glossy to pure mirror. The diffuse buffer captures diffuse global illumination, including shadows and interreflections, while the specular buffer captures mirror and glossy reflections. We reconstruct the diffuse component of the scene using ray traced Gaussian primitives and compute specular reflections with path tracing while treating the diffuse component similarly to an irradiance cache [Ward et al. 1988]. Our reflections support surfaces of different roughness, and multi-bounce effects computed in a physically-based manner, i.e., exclusively with path tracing. They also allow interactive scene editing with physically-correct reflections. Note that transparency is more complicated, since buffers would have to deal with two rays—reflected and refracted; we thus leave it as future work.

First, we present a *proof-of-concept* of our method on synthetic scenes, where rendered images are used in the place of photos, and ground truth values are used for all buffers (diffuse, specular, materials etc.). This solution shows we can duplicate a reflective object in a scene, and all *real, captured* objects will be correctly reflected in the duplicated object interactively, even with multiple bounces (see Fig. 1). Second, we present an editable inverse rendering pipeline on synthetic *and real* images using buffers predicted by neural networks, albeit with lower quality for novel-view synthesis. In particular, recent intrinsic image decomposition methods can extract the required buffers from photographs [Roberts et al. 2021; Zeng et al. 2024], by training on synthetic data where light transport is separated into diffuse and reflective (or non-diffuse) components at the first bounce. That being said, while these methods are improving at outstanding rates, the current quality of predicted buffers from existing methods (e.g., [Ke et al. 2024; Liang et al. 2025a; Zeng et al. 2024]) is only partially sufficient for our goals. We thus focus mainly on our proof-of-concept using synthetic data with perfect buffers to develop our method, and present preliminary results for scenes processed with the best possible existing learning-based methods at the time of writing.

In short, our method leverages intrinsic decomposition to separately reconstruct the diffuse and reflected components of a scene (see Fig. 2) to make editable specular reflections possible (see video). We optimize a *unique* representation of the scene, i.e., both the reflectors and reflected objects are represented with the same Gaussian primitives, by carefully designing separate losses and a training schedule. This unique representation makes it possible to reconstruct objects only viewed indirectly through specular reflections, as shown in Fig. 3. Finally, we present an efficient hardware-accelerated Gaussian primitive ray tracer, which uses efficient spatial data structures and other techniques to optimize Gaussian primitive traversal.

In summary our contributions are:

- A reconstruction method for radiance fields with distinct optimization for diffuse and specular components, using path tracing for the latter.
- An efficient and accurate training method that reconstructs the diffuse and the specular components of the scene in a single representation.
- An efficient ray tracer for Gaussian primitives, that is fast enough to enable treatment of multiple bounces with minimal computational overhead.

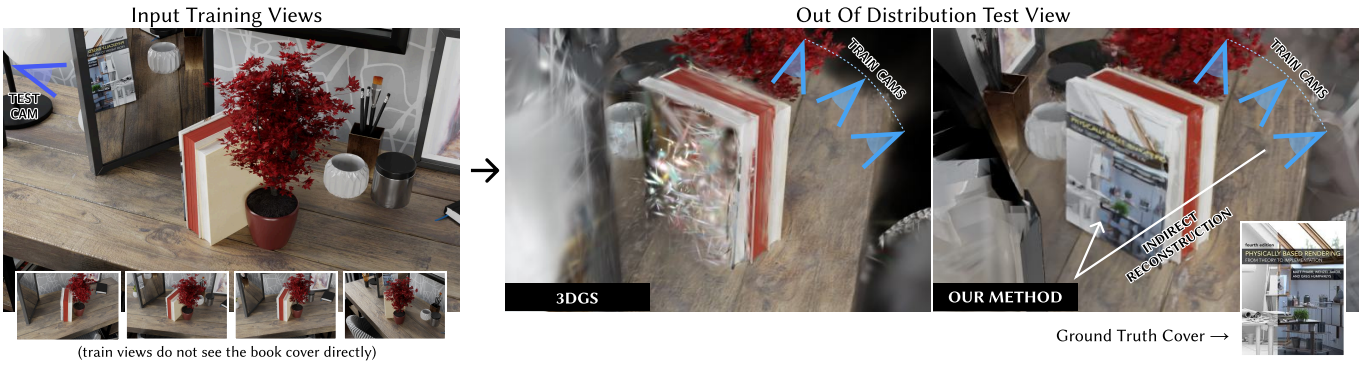

Fig. 3. We showcase reconstruction of objects viewed indirectly by recovering the cover of a book only visible indirectly through a mirror in the training views (left). From these views alone, our method accurately recovers the cover (right), unlike standard methods like 3DGS (middle). Note that in these results, our method uses ground truth input buffers.

Given our advantage of additional input buffers, and notably diffuse/specular image decomposition, we achieve state-of-the-art (SOTA) diffuse/specular disentanglement for novel-view synthesis, even though overall visual quality is lower that previous methods. Our disentanglement allows real-time reflection and material editing in both the proof-of-concept case with ground truth buffers and for real scenes. For the latter we present an experimental neural network that provides the buffers we require. While quality measured on our synthetic scenes given SOTA networks is low, it is sufficient for realistic, interactive scene manipulation in certain real scenes, showing the potential of the approach with future, better quality learning-based buffer predictions. Code and data are available at: https://repo-sam.inria.fr/nerphys/editable-gaussian-reflections/.

2 RELATED WORK

Our goal is to provide physically-based specular reflections for radiance fields that allow consistent real-time editing and reconstruction of the unseen reflected objects. We review most closely related methods.

Reflections in NeRF. Neural radiance fields, or NeRF [Mildenhall et al. 2020], learn an implicit scene representation, parametrized with a multi-layer perceptron (MLP), from multi-view posed images. NeRF typically represents specular reflections with a combination of view-dependent color in the MLP and "fake geometry" representing the reflected objects *behind* the reflector. Ref-NeRF [Verbin et al. 2022] improve on this by using the reflected view direction, NeRFReN [Guo et al. 2022] model the transmitted and reflected scene separately, similarly for MS-NeRF [Yin et al. 2025], and Mirror-NeRF [Zeng et al. 2023] assume planar mirrors. Others [Liang et al. 2023a; Liu et al. 2023; Wang et al. 2024] exploit signed distance fields [Li et al. 2023; Wang et al. 2021] to obtain more accurate scene geometries, or modify the directional encoding [Ma et al. 2024] to better capture the spatially-varying nature of near-field lighting. NeRF-casting [Verbin et al. 2024] integrates reflection features along reflection directions. However, the need for multiple MLP queries for each ray and the creation of a separate version of the scene makes it both impractical for real-time rendering and reflection-consistent scene editing.

Reflections in 3D Gaussian Splatting. 3DGS [Kerbl et al. 2023] introduces a primitive-based representation for radiance fields, and uses efficient rasterization to render Gaussian primitives, achieving fast training and real-time rendering. 3DGS-based approaches model reflections with a combination of view-dependent spherical harmonics (SH) and the creation of "fake geometry" representing reflected objects behind the reflected surface. More recent works [Jiang et al. 2024; Ye et al. 2024], enhance reflection modeling by incorporating additional environment maps. However, these methods focus only on distant lighting. 3iGS [Tang and Cham 2024] integrates an illumination field through tensorial factorization (as in [Chen et al. 2022; Jin et al. 2023]) and renders the final reflections with a neural renderer, but is constrained to bounded scenes. Ref-GS [Zhang et al. 2025] models near-field reflection using a tensorial factorization within a 2DGS framework [Huang et al. 2024], and models far-field illumination with a spherical feature grid. None of these approaches overcome the basic limitation of representing specular reflections with "fake geometry", making them unsuitable for consistent editing. Some approaches also assume knowledge of planar mirrors [Liu et al. 2024; Meng et al. 2024]. Most approaches are efficient for objects, but were not tested on full scenes [Lai et al. 2025]. A predecessor of 3DGS [Kopanas et al. 2022] models reflections with a separate point cloud rendered with an MLP and shows reflection editing, which is not truly consistent.

Ray Tracing with Gaussian Primitives. 3D Gaussian Ray Tracing (3DGRT) [Moenne-Loccoz et al. 2024] renders a 3D Gaussian representation using ray tracing instead of rasterization. They build a bounding volume hierarchy and cast a ray for each pixel using high-performance GPU ray tracing hardware [Parker et al. 2010]. 3DGUT also showed that reflection rays are possible in a splatting framework [Wu et al. 2025]. A similar proposal is made in RaySplats [Byrski et al. 2025a], more recently extended in REdiSplats [Byrski et al. 2025b] and in RayGauss [Blanc et al. 2025]. After training, 3DGRT shows they can render secondary ray effects such as reflections. Critically, only mesh-Gaussian reflections can be handled by their framework; in contrast, we model Gaussian-Gaussian reflections *during training*. We also improve over 3DGRT with several performance enhancements (see Sec. 3.3). Inter-Reflective

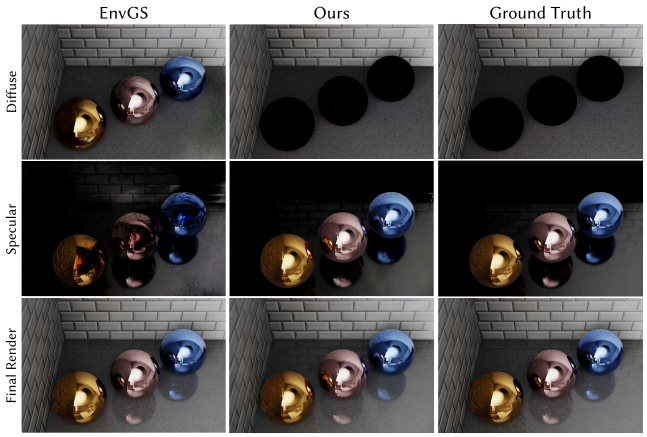

Fig. 4. Radiance fields, recovered for example with EnvGS [Xie et al. 2025] (left), often rely on duplicate geometry to represent specular reflections; notice the replicas of the environment placed within the chrome spheres (top-left). Our method (middle) represents reflections with physically-based light transport *only*, closely matching ground truth novel views (right). Note that in these results, our method uses ground truth input buffers.

Gaussian Splatting (IRGS) [Gu et al. 2024] attempts to address inter-reflection modeling by extending the 2DGS framework with a differentiable ray tracing approach. EnvGS [Xie et al. 2025] trains two versions of the scene: a base scene (with rasterization), and "environment Gaussians" (with raytracing) which represents scene elements only visible through reflections. While this approach improves the separation between the diffuse and specular component, there are still residual reflections in the diffuse buffer (as we show in Sec. 4) and the lack of a BRDF model makes it unsuitable for editing. In contrast, we handle inter-reflections, and also reconstruct scene elements only visible in specular reflections, thanks to our *unique* scene representation. This allows consistent interactive reflection editing.

Inverse Rendering and Relighting. Scene editing could be achieved using full inverse rendering [Li et al. 2018; Marschner 1998; Nimier-David et al. 2019]. However, full physically-based inverse rendering is ill-posed [Kouros et al. 2024] and expensive [Li et al. 2018; Nimier-David et al. 2019; Srinivasan et al. 2021; Zhang et al. 2021], and currently cannot handle complex scenes [Shi et al. 2025]. GS-IR [Liang et al. 2023b] leverages 3DGS to accelerate it. Relighting techniques [Bi et al. 2024; Gao et al. 2024; Kuang et al. 2024; Poirier-Ginter et al. 2024] allow the *lighting* to be edited, and are orthogonal to the kind of edits we show (moving reflective objects and changing materials). Combining them is interesting future work.

3 METHOD

At a high level, our method starts with BRDF parameters from input views, attaches these parameters as well as depth and normals to diffuse Gaussians using 3DGS-like optimization, and during optimization computes specular reflections by simulating full light transport of the specular component. At each intersection, accumulated parameters are used to perform proper ray tracing.

3.1 Diffuse/Specular Separation for Physics-Based Reflections

By computing and supervising diffuse and specular reflections separately, our method makes it much harder for the optimization to "cheat" and create "fake" geometry to represent reflections.

To achieve this we make a key design choice by having separate supervision signals: one for the diffuse and one for the specular component—both are computed with ray tracing. However, we create a *unique* scene representation for both. The Gaussian primitives in this reconstruction will represent all geometry visible in the input views, but also the reflected objects *at coherent positions* in space (see Fig. 13).

The RGB colors in the input images are the result of (real-world) physical light transport, expressed by the rendering equation:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}_o) = L_e(\boldsymbol{x}, \boldsymbol{\omega}_o) + \int_\Omega L_i(\boldsymbol{x}, \boldsymbol{\omega}_i) f(\boldsymbol{\omega}_i, \boldsymbol{n}, \boldsymbol{\omega}_o) \cos\theta_i \, d\boldsymbol{\omega}_i \,, \quad (1)$$

where L_o, L_e, L_i are the outgoing, emitted, incoming radiance respectively, $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ are the incoming and outgoing directions, $\boldsymbol{n}$ is the surface normal at point $\boldsymbol{x}$, $\cos\theta_i = \boldsymbol{\omega}_i \cdot \boldsymbol{n}$, and f is the BRDF. We use the approximation that $f = f_d + f_s$ with f_d and f_s the diffuse and specular components. Ignoring the emissive term, our method approximates light transport by the sum of two terms:

$$\begin{aligned} L_o(\boldsymbol{x}, \boldsymbol{\omega}_o) = &\int_\Omega L_i(\boldsymbol{x}, \boldsymbol{\omega}_i) f_d(\boldsymbol{\omega}_i, \boldsymbol{n}) \cos\theta_i \, d\boldsymbol{\omega}_i + \\ &\int_\Omega L_i(\boldsymbol{x}, \boldsymbol{\omega}_i) f_s(\boldsymbol{\omega}_i, \boldsymbol{n}, \omega_o) \cos\theta_i \, d\boldsymbol{\omega}_i \,. \end{aligned} \quad (2)$$

Using Heckberts light path notation [Heckbert 1990], the first term computes all $\mathcal{L}((\mathcal{D}|\mathcal{S})*)\mathcal{D}\mathcal{E}$ paths, where $\mathcal{L}$ are the lights, $\mathcal{D}$ the diffuse vertices in a path, $\mathcal{S}$ the specular vertices and $\mathcal{E}$ is the eye or camera, while the second term computes all $\mathcal{L}((\mathcal{D}|\mathcal{S})*)\mathcal{S}\mathcal{E}$ paths. When path tracing scenes, each term is computed with the same paths, but at the first intersection the first (diffuse) term is evaluated using only f_d, and the second (specular) term using only f_s. We reconstruct the diffuse component by supervising on the diffuse buffer, resulting in a "cached" version of diffuse lighting $\hat{L}_d$ which we use like an irradiance cache already multiplied with albedo i.e.

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}_o) = \hat{L}_d + \int_\Omega L_i(\boldsymbol{x}, \boldsymbol{\omega}_i) f_r(\boldsymbol{\omega}_i, \boldsymbol{n}, \omega_o) \cos\theta_i \, d\boldsymbol{\omega}_i \,. \quad (3)$$

We thus avoid a large part of the full path tracing costs, namely the first term in Eq. (2). This reconstruction contains "baked" shadows and diffuse color bleeding. Other quantities (depths, normals, material properties) are also supervised with corresponding buffers, while reflections (second term in Eq. (2)) are computed with path tracing. Diffuse and specular components are shown in Figs. 2 and 4.

To develop our approach and evaluate it quantitatively, we first use synthetic scenes, rendering RGB images that serve the same purpose as real photos for radiance field reconstruction. We render exact image buffers per input view for diffuse and specular component of Eq. (2). We also render buffers with all depths, normals, BRDF coefficients per pixel that are attached to Gaussians, and then used to perform proper ray traced reflections. For real images, we use a neural network trained on such path traced images (e.g., using the Hypersim [Roberts et al. 2021] and InteriorVerse [Zhu et al.

2022] datasets) to extract diffuse and specular buffers, and buffers for the other parameters (Sec. 3.4).

We use the Cook-Torrance BRDF [Cook and Torrance 1982] and specifically, a subset of the Disney BRDF formulation [Burley 2012]. The parameters of this model are the roughness ρ and the base reflectance or F_0 which is defined based on the metalness, specular and color parameters [Burley 2012].

Reconstructing the Diffuse Component. We reconstruct the diffuse component using a single diffuse RGB value per Gaussian primitive instead of spherical harmonics. This step could use the standard 3DGS splatting pipeline, but we prefer to use our ray tracing solution throughout to avoid the discrepancy induced by the affine approximation of splatting [Huang et al. 2025], ensuring a single consistent solution. Note that, like 3DGRT, our scenes are incompatible with 3DGS since we use per-pixel sorting to avoid popping artifacts.

The diffuse component is supervised directly with the diffuse buffer, and reconstructs a version of the scene with diffuse global illumination and shadows. For the other quantities (normals, roughness and base reflectance), we attach values to each Gaussian primitive, and optimize with supervision from the per-input-view buffers. We also optimize the expected termination depth [Kerbl et al. 2023] with the depth buffers.

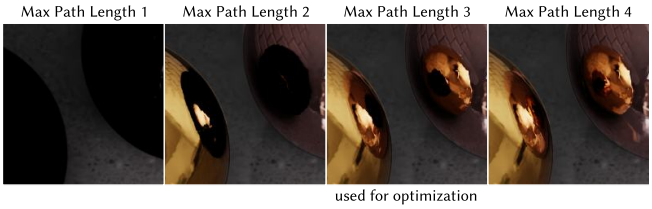

Fig. 5. We support multi-bounce reflections during optimization. The number of bounces can be adjusted, trading off performance vs. accuracy.

Ray Traced Physics-Based Specular Reflections for Radiance Fields. We compute the $\mathcal{L}((\mathcal{D}|\mathcal{S})*)\mathcal{SE}$ paths with path tracing of the Gaussian primitives and cached diffuse. Each primitive is initialized with a base reflectance value F_0 initialized at 0.04 and with roughness and normal values initialized at 0. Each ray traverses Gaussians while accumulating values with alpha-blending as in 3DGS, for all quantities, including normals, roughness and F_0 values. The accumulated "normal" value is then normalized to unit length. We compute the intersection at the expected termination depth and use accumulated normal, roughness and F_0 values to importance sample a random reflection ray with the Cook-Torrance BRDF. Our approach supports multiple ray bounces natively, see Fig. 5. In practice, we set the max path length to 3, resulting in good performance at the cost of some minor inaccuracies (i.e. the black spots in reflections in Fig. 1). During these bounces we query the diffuse component representation for color, which is the (diffuse component of) radiance extracted from the input images (i.e., real-world lighting for photos). We train at 1 sample per pixel, but for offline rendering use 128 samples and then apply the OptiX denoiser (see Fig. 6). The specular component is supervised with the specular buffer provided as input.

Fig. 6. We support physically-based reflections with different levels of roughness in the same scene. At inference, we sample several samples per pixel (column 1) and denoise with the OptiX denoiser (column 2).

Finally, we adapt dense initialization [Kotovenko et al. 2025] instead of densification to further accelerate convergence; we found ray tracing effective at handling the large number of small Gaussians that dense initialization provides. We do not densify further and aim for a number of gaussians comparable with the baselines; for details refer to Supplemental.

3.2 Optimizing a Unique Scene for Radiance Field Reflections

Optimizing a representation that uses Gaussian primitive ray tracing is hard: previous approaches [Verbin et al. 2024; Xie et al. 2025] showed that flowing gradients naively through reflections damages geometric reconstruction. This can create a feedback effect where bad geometry skews reflection rays, leading to more damage. These previous approaches side-step this problem by optimizing two separate radiance fields, one for the primary scene and one for reflections (e.g., [Xie et al. 2025]). However, such a choice precludes scene editing with multi-bounce reflections, and can hinder reconstruction of reflected objects unseen in the input views. We overcome this optimization instability and reconstruct a unique scene.

Diffuse/specular separation resolves the ambiguity between texture and reflected objects, letting us maintain a unique scene while keeping training stable. Our training proceeds with two different losses: one for diffuse and one for specular components. It also recovers normals and BRDF directly through input buffer supervision and not via inverse rendering: we do *not* differentiate normals and BRDF w.r.t. the RGB images, ensuring that the scene remains well-formed even with gradients flowing through reflections.

We cast a path through each pixel, and use the first segment of the path to query diffuse global illumination (see Sec. 3.1), represented by the diffuse component reconstruction. We denote this quantity d and we supervise with the value d^* in the diffuse buffer:

$$\mathcal{L}_d = \lambda_d \ell(\hat{d}, d^*)\,, \tag{4}$$

where $\hat{}$ denotes the accumulated value, $(^*)$ the corresponding ground truth value from the buffers, and ℓ the L1 loss. We then continue the path using the specular BRDF component f_s and match the contribution of the remaining path segments $c_i,\ i = 2 \ldots K$ to the specular buffer s:

$$\mathcal{L}_r = \lambda_r \ell(\sum_{i=2}^{K} \hat{c}_i, s^*)\,, \tag{5}$$

where K is the maximum path length. To this we add a loss matching the accumulated values of other attached properties namely depth $\hat{t}$, normal $\hat{n}$, roughness $\hat{\rho}$, and base reflectance $\hat{F}_0$ to their

corresponding target buffers:

$$\mathcal{L}_a = \lambda_t\, \ell(\hat{t}, t^*) + \lambda_n\, \ell(\hat{n}, n^*) + \lambda_\rho\, \ell(\hat{\rho}, \rho^*) + \lambda_{F_0}\, \ell(\hat{F_0}, F_0^*) \quad (6)$$

This loss is only applied to the first path segment since these buffers are only known for this segment. Our final objective is:

$$\min_{\theta,d} \mathbb{E}\left[\mathcal{L}_d + \mathcal{L}_r\right] + \min_{\theta,t,n,\rho,F_0} \mathbb{E}[\mathcal{L}_a]\,, \quad (7)$$

where θ contains the position, rotation, scale and alpha parameters for each Gaussian. Note that both subobjectives are optimized at the same time. We use values $\lambda_d = 5.0$, $\lambda_s = 3.0$, $\lambda_t = 2.5$, $\lambda_n = 2.5$, $\lambda_\rho = 1.0$ and $\lambda_{F_0} = 1.0$. We compute all losses in linear space.

Our training schedule starts with a warmup for 750 iterations with the first ray segment only, which reconstructs a first version of the diffuse component. We then enable all reflections and add $M = 75,000$ far-field Gaussians initialized around the scene origin with positions randomly sampled from a normal distribution with $\sigma = 4S$ truncated at 3σ; the scene extent S is estimated as in 3DGS by computing the radius of a sphere which bounds the input camera poses [Kerbl et al. 2023]. As optimization progresses, these primitives provide a (coarse) approximate reconstruction of the reflected objects in the environment (Fig. 13).

3.3 Efficient Ray Tracer for Multi-Bounce Radiance Field Reflections

To achieve our goals, we need *efficient* Gaussian ray tracing, both for optimization and rendering ("inference"), beyond the speed of existing methods (Sec. 2). To do this we focus on four components: 1) Using oriented bounding boxes (OBBs) to bound Gaussians 2) Avoiding multiple Bounding Volume Hierarchy (BVH) traversals, 3) Fused forward/backward pass and 4) Aggressive truncation. Numbers reported below use the configuration employed to compare with 3DGRT [Moenne-Loccoz et al. 2024] (Supplemental, Sec. 3.3).

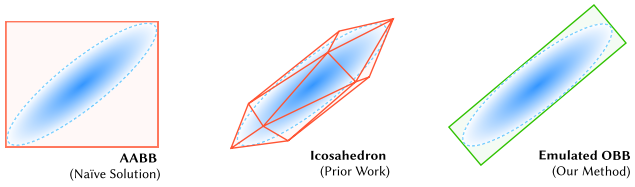

Fig. 7. We encapsulate Gaussians with oriented bounding boxes (OBB) using instancing with hardware-accelerated OptiX transforms.

Oriented Bounding Boxes. Prior methods [Condor et al. 2025; Moenne-Loccoz et al. 2024] use an icosahedron mesh to bound Gaussians, allowing fast intersections for rendering, but requiring costly BVH updates for the 20 triangles assigned to each Gaussian during optimization. To avoid this overhead, we exploit the hardware accelerated capabilities of OptiX, allowing efficient use of OBBs [1]. OptiX only supports Axis-Aligned Bounding Boxes (AABBs) natively, thus naively emulating OBBs would still result in slow traversal (Fig. 7).

[1]The method of Moenne-Loccoz et al. [2024] describes the icosahedron approach, however a recent code release associated with the paper also uses OBBs. [Lee et al. 2024] also proposed OBBs but for splatting.

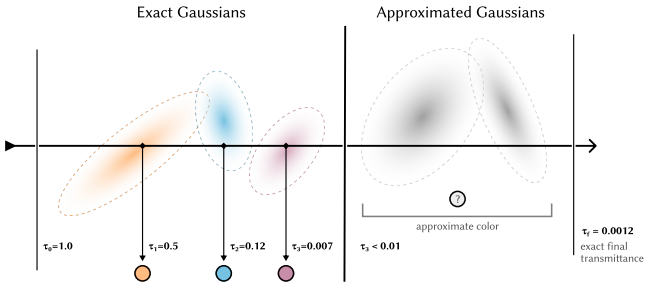

Fig. 8. We accelerate integration by approximating Gaussians of low contribution, enabling aggressive transmittance thresholding during optimization.

Instead, we exploit hardware accelerated transforms of AABBs provided by OptiX, by instancing an AABB in a two-level acceleration structure [Wald et al. 2020]. This has the added benefit of letting us simplify the Gaussian evaluation since we avoid computing covariance matrices explicitly and we do not invert the Gaussian transforms ourselves. On the Kitchen scene (Fig. 12), an incremental BVH update adds a minimal overhead of 9.5%.

Avoiding Multiple BVH Traversals per Ray. Per-pixel sorting a large number of Gaussian primitives is slow since it requires incoherent memory accesses. 3DGRT [Moenne-Loccoz et al. 2024] resolved this by collecting and sorting the nearest primitives in small (typically, 16) groups which fit in register memory, requiring multiple BVH traversals. To avoid this, we perform a single traversal and store all intersected Gaussians into a per-pixel linked-list (PPLL) [Yang et al. 2010] "replay buffer" which contains Gaussian id, alpha value, etc. We then loop over this buffer multiple times to collect, sort, and integrate the nearest 16 Gaussians. While this approach does increase memory consumption, our experiments show high-resolution scenes can still be handled on modern cards (e.g., RTX4090). Our experiments show this results in speedups of 10–50% depending on the scene and resolution.

Fused Forward/Backward Pass. To further accelerate optimization, we use a fused forward/backward pass where each pixel's backward pass starts immediately after the forward pass. We use a PPLL to store data for the backward pass: the intersected Gaussian ids, hit distances, precomputed alpha values, etc. Our backward pass is quite fast, a combined forward/backward is 2.23 times slower than just the forward pass while 3DGS is 3.31 times slower.

Aggressive Primitive Truncation. Since transmittance decays rapidly along a ray, all Gaussians past a certain point have minimal contribution to the final color. 3DGRT [Moenne-Loccoz et al. 2024] handled this by reducing the transmittance threshold from $\tau = 0.001$ to $\tau = 0.03$ at inference. Doing so during optimization fails since discarding even seemingly insignificant Gaussians biases gradients of the others in front, making them to grow to mask the background. To avoid this instability, we discard farthermost Gaussians, and scale the gradients of those remaining using an approximation of their color, similar to [Byrski et al. 2025a; Hahlbohm et al. 2025], see Fig. 8. See Supplemental for more details.

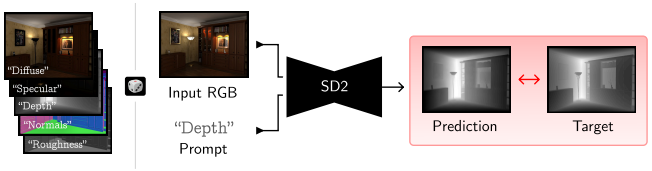

Fig. 9. We fine-tune a pretrained Stable Diffusion 2 network into a one-step inverse rendering network.

3.4 A Dedicated Network to Infer Disentangled Layers

Recent learning-based methods have made incredible advances in inferring different intrinsic quantities such as depth, normals or reflectance properties directly from images [Garcia et al. 2025; Ke et al. 2024; Li et al. 2025; Liang et al. 2025a; Xi et al. 2024; Zeng et al. 2024]. Many of these methods fine-tune a diffusion model (typically StableDiffusion2 (SD2) [Rombach et al. 2022]). Marigold [Ke et al. 2024] and RGB↔X [Zeng et al. 2024] require many denoising steps; more recent work [Garcia et al. 2025; Xu et al. 2025] showed that for strongly conditioned tasks, a single step model can be equally accurate and cheaper at inference. Additionally, concurrent work [Liang et al. 2025b] leverages video models to improve quality and temporal stability; we leave extension to video models as future work, and base our model on SD2.

Based on these recent results, and since none predict the non-diffuse buffer needed by our method, we fine-tune SD2 into a single step model that predicts all the buffers (diffuse, specular, normals, see Sec. 3.2) and switches between them with text prompts, using the approach of RGB↔X [Zeng et al. 2024] (Fig. 9). For training, we use a mix of the Hypersim [Roberts et al. 2021] and InteriorVerse [Zhu et al. 2022] datasets and provide as input the RGB image and a prompt designating the desired buffer. We use a latent mean squared error (MSE) loss for fast training. Fig. 10 shows that our network yields plausible predictions of all target buffers.

4 RESULTS AND EVALUATION

We first present results of real-time reflection editing of radiance fields. Since no previous method can perform such editing, we present our best-effort evaluation of diffuse/specular disentanglement, and compare our ray-tracer with that of 3DGRT [Moenne-Loccoz et al. 2024]. Finally, in supplemental we show that we perform better than 3DGS for a sparse set of views.

4.1 Real-Time Editing

We show results of real-time editing of radiance field with specular reflections in the supplemental video and in Figs. 1, 11 and 12. We can consistently move reconstructed reflectors: reflections are correctly updated in *other* objects in the radiance field. Similarly, we can modify material properties with consistent updates. These operations are only possible because of our full disentanglement and *multiple bounces* we enable. Note however that shadows are "baked" into the diffuse layer, and are not updated. In Figs. 3 and 13, we see that our method reconstructs the *unseen* reflected objects, with accuracy that is sufficient for specular reflection computation.

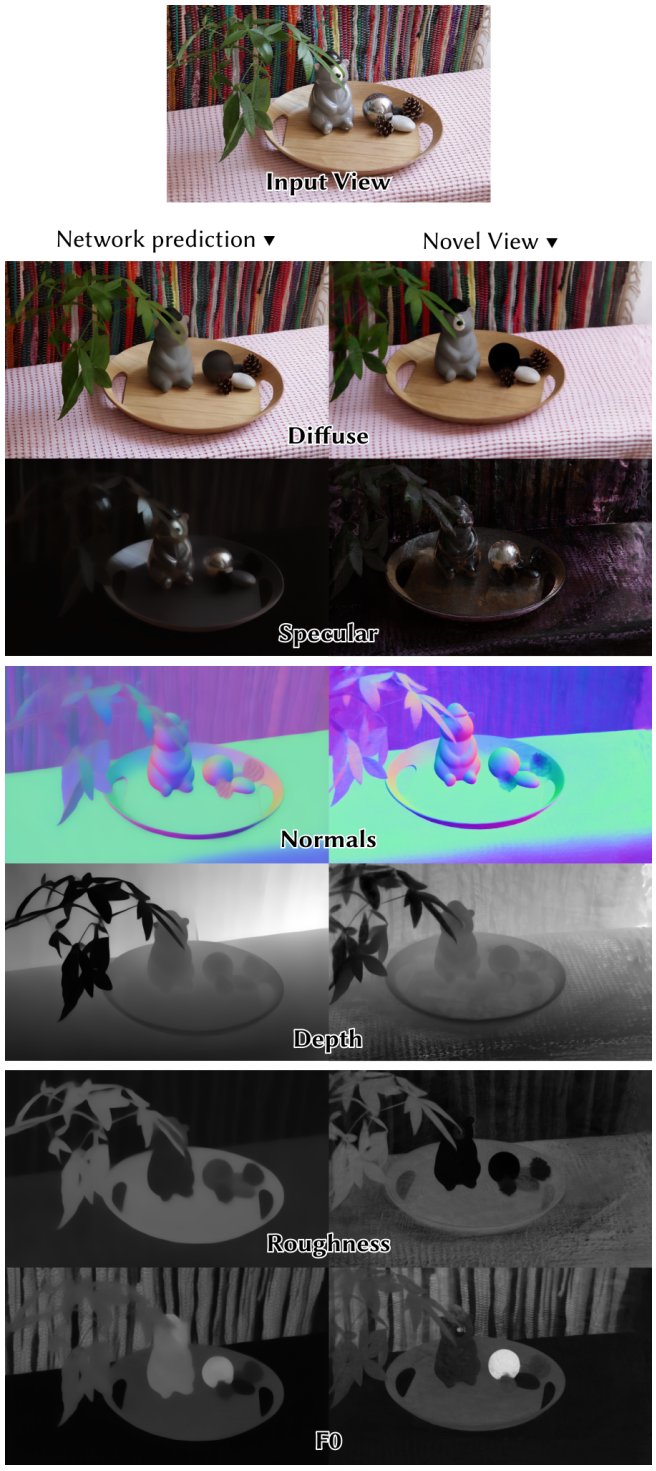

Fig. 10. Example network predictions (left) for a given input view (top), and novel view renderings (right) of the different layers used in our approach obtained after optimization.

Fig. 11. Editing two synthetic scenes: making a typewriter diffuse (first row), making a lamp metallic (second row), changing every object to be smooth or diffuse (third row), duplicating a reflective pot (fourth row). Observe how reflections are consistent with the scene edits.

4.2 Evaluation on Synthetic Scenes

We cannot directly compare our method with previous work, since no other solution allows consistent editing of specular reflections in radiance fields. As a best-effort evaluation, we compare quality of diffuse/specular disentanglement which is required for reflection editing. We show results both for the proof-of-concept case with ground truth (g.t.) buffers, as well as the inverse rendering approach using inferred buffers. In all cases our method has the advantage of using these additional buffers—in particular diffuse/specular image decomposition—that previous methods cannot exploit.

To allow precise evaluation, we use three synthetic scenes adapted from [Poirier-Ginter et al. 2024] at 1152×768 resolution and changed material properties to be shinier. We also removed all transparency from material properties of scene objects. We render RGB images and additional layers, used as input to our method.

We compare to methods based on 3DGS-based solutions that claim to model diffuse, in particular: Gaussian Shader [Jiang et al. 2024], 3DGS-DR [Ye et al. 2024], Reflective-GS [Yao et al. 2024] and the recent (to appear) EnvGS [Xie et al. 2025] which uses ray tracing and is the closest method to ours. Again, our method has the advantage of additional buffers that these previous approaches are not designed to use.

Fig. 12. Editing a synthetic scene: changing the base reflectance (F_0) of the cups (top), rotating the plate (middle), changing the roughness of the teapot (bottom). Observe how reflections are consistent with the scene edits.

Fig. 14 shows that, in complex scenes, existing approaches incorrectly separate the diffuse from specular components of the scene—reflections are represented as "fake mirror" geometry in the diffuse, making them unusable for editing. In contrast, assuming an optimal network (given *ground truth* inputs), our method reconstructs a clean reflection-free diffuse pass.

In Tab. 1 we show the PSNR for the 3 synthetic scenes for each method (more details in the supplemental). Metrics are computed on a test path separate from input views. The top part of the table does not use any ground truth values, only network predictions; our method ($\text{Ours}_{(\text{net. inputs})}$) achieves on average better disentanglement for diffuse/specular compared to all other methods, including the concurrent EnvGS, using network predicted normals ($\text{EnvGS}_{(\text{net. normals})}$), despite its lower overall quality. As a proof-of-concept, in the last two rows we show results for our method using all the ground truth layers ($\text{Ours}_{(\text{g.t. inputs})}$). We also show EnvGS using ground truth normals ($\text{EnvGS}_{(\text{g.t. normals})}$). Since our method does not create "fake" geometry to reproduce the input RGB images, reconstructing the final image is much harder, and thus PSNR is lower. Note however that all other methods have lower final PSNR than 3DGS and do not improve much on the diffuse nor specular layers, showing their inability to properly model specular reflections in complex scenes. In addition, qualitatively our image quality is sufficient for editing (see video and figures). Achieving higher PSNR requires solving the hard problem of accurate reconstruction of the reflected part of the scene, as well as improved optimization (see Sec. 5).

We also show optimization times (Tab. 2), which shows that our method optimizes significantly faster than all previous method and ~5× faster than the second best method EnvGS. In supplemental we show statistics of number of primitives and FPS: as expected, our method is slower but still interactive, since we trace multiple rays per pixel.

4.3 Evaluation on Real Scenes

On real scenes, we use the network (Sec. 3.4) to compute layers from input images. The prediction quality, seen for example in Fig. 10, is far from perfect, but we do manage to obtain sufficiently good results

Table 1. Disentanglement performance comparison. Note that our method has the advantage of using additional buffers unavailable to other methods. The second part of the table is shown as a proof-of-concept, using g.t. train view inputs where possible.

	Shiny Kitchen			Shiny Livingroom			Shiny Office		
	Diffuse	Spec.	Final	Diffuse	Spec.	Final	Diffuse	Spec.	Final
3DGS	13.11	13.17	**32.83**	17.01	17.35	**32.72**	19.10	15.74	**34.91**
Gauss. Shader	14.88	12.15	30.13	20.57	16.91	24.26	**21.77**	15.56	32.75
3DGS-DR	10.10	8.59	32.79	16.08	12.97	31.39	9.74	8.96	34.66
Refl. GS	13.50	13.45	32.17	20.43	20.46	29.85	20.63	17.24	33.68
EnvGS$_{\text{(net. normals)}}$	14.33	14.48	32.82	22.74	**21.22**	30.65	20.92	16.33	34.16
Ours$_{\text{(net. inputs)}}$	**20.36**	**16.95**	20.41	**23.77**	20.35	21.21	20.60	**17.40**	17.75
EnvGS$_{\text{(g.t. normals)}}$	14.51	15.15	**32.96**	22.64	21.46	**30.31**	20.81	15.90	**34.34**
Ours$_{\text{(g.t. inputs)}}$	**33.20**	**24.30**	26.96	**29.68**	**26.46**	26.96	**31.74**	**24.48**	27.54

to demonstrate similar editing examples (see video and Fig. 1). We compare to EnvGS in Fig. 15 where we show renderings of diffuse and specular layers, and the final novel view (left EnvGS, right ours). We can clearly see that the EnvGS diffuse layer contains specular content, especially visible in the Bear scene (far right).

In Supplemental (Sec. 3.5), we ablate the quality of network predictions, progressively replacing GT layers with predicted versions. The most important layers are diffuse/specular that have a significant impact on PSNR; all other layers are less important.

Table 2. Training times for different methods.

	Shiny Kitchen	Shiny Livingroom	Shiny Office
GShader	2:24:26	2:28:28	2:31:09
3DGS-DR	0:44:09	0:45:45	0:51:41
ReflGS	1:28:57	1:28:38	1:43:22
EnvGS$_{\text{(g.t. norm.)}}$	3:34:27	2:40:52	3:13:21
Ours$_{\text{(net. inputs)}}$	0:48:52	0:23:56	0:37:05
Ours$_{\text{(g.t. inputs)}}$	**0:23:16**	**0:20:48**	**0:24:13**

4.4 Comparing our Raytracer to 3DGRT

We compared our raytracer's performance to 3DGRT when used as a drop-in replacement for regular 3DGS, by swapping it for the splatting rasterizer while limiting to 1 ray per pixel and integrating RGB color only. We trained for 7k iterations and our method does not have spherical harmonics; for details of the exact configuration used, please see supplemental. Tab. 3 shows that our raytracer improves training times and FPS performance over 3DGRT.

Table 3. Performance comparison between our raytracer and 3DGRT when used as a drop-in replacement for 3DGS. Average across all MipNerf scenes at different resolutions. Our method was run without spherical harmonics.

Downsampl.	Training Time			FPS			PSNR (dB)	
	3DGRT	Ours	Speedup	3DGRT	Ours	Speedup	3DGRT	Ours
2	00:29:24	**00:06:51**	4.35×	26.77	**49.93**	2.34×	**25.44**	25.01
4	00:10:03	**00:02:27**	4.08×	75.49	**143.93**	2.43×	**25.83**	25.35
8	00:05:28	**00:01:08**	4.77×	139.09	**248.75**	2.24×	**26.27**	26.25

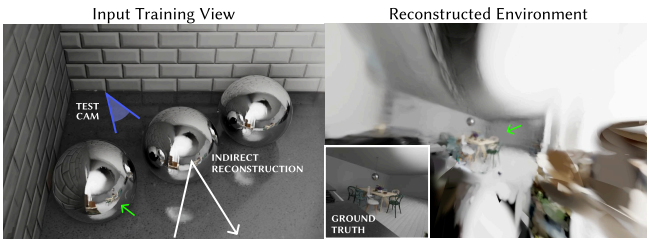

Fig. 13. Our method reconstructs the environment behind the camera with a high enough degree of fidelity to produce realistic novel views of parts of the scene *never observed in the input images*. This reconstruction is part of the same scene and can be observed by simply turning the camera around.

5 LIMITATIONS AND CONCLUSION

We make an important step forward to allow truly disentangled, physically-based reflections for radiance fields. The main limitation of our method is the performance of the network used to extract layers: we are confident that such approaches will keep improving and provide the quality needed. Another limitation is lack of support for transparency. This requires determining how to estimate transparency from images (including producing training data and corresponding networks), and an efficient rendering method. Full scene editing with shadows updates requires a solution with (at least partial) relighting. Finally, improving image quality even with perfect layers requires more accurate reconstruction of the reflected scene. These are all hard problems and exciting new research directions.

In conclusion, we presented a new approach that allows real-time, consistent specular reflection editing in radiance fields, enabled by our diffuse/specular disentanglement and our support for multi-bounce reflections. The key to our solution is the use of separate supervision of the diffuse and specular layers of images, building on learning-based predictors of such layers. We also show how to provide stable training and several performance improvements to Gaussian ray tracing that, taken together, allow real-time, consistent, physically-based editing of reflections, and reconstruction of unseen reflected objects.

ACKNOWLEDGMENTS

This research was co-funded by the European Union (EU) ERC Advanced grant FUNGRAPH No 788065 and ERC Advanced Grant NERPHYS No 101141721. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the EU or the European Research Council. Neither the EU nor the granting authority can be held responsible for them. This research was also supported by NSERC grant RGPIN-2020-04799 and the Digital Research Alliance Canada. The authors are grateful to Adobe and NVIDIA for generous donations, and the OPAL infrastructure from Université Côte d'Azur.

REFERENCES

Zoubin Bi, Yixin Zeng, Chong Zeng, Fan Pei, Xiang Feng, Kun Zhou, and Hongzhi Wu. 2024. GS^3: Efficient relighting with triple gaussian splatting. In *ACM SIGGRAPH Asia Conf.*

Hugo Blanc, Jean-Emmanuel Deschaud, and Alexis Paljic. 2025. Raygauss: Volumetric gaussian-based ray casting for photorealistic novel view synthesis. In *IEEE/CVF*

Winter Conf. App. Comput. Vis.

Brent Burley. 2012. Physically-Based Shading at Disney. https://api.semanticscholar.org/CorpusID:7260137

Krzysztof Byrski, Marcin Mazur, Jacek Tabor, Tadeusz Dziarmaga, Marcin Kadziolka, Dawid Baran, and Przemyslaw Spurek. 2025a. RaySplats: Ray Tracing based Gaussian Splatting. *arXiv preprint arXiv:2501.19196* (2025).

Krzysztof Byrski, Grzegorz Wilczyński, Weronika Smolak-Dyżewska, Piotr Borycki, Dawid Baran, Sławomir Tadeja, and Przemysław Spurek. 2025b. REdiSplats: Ray Tracing for Editable Gaussian Splatting. *arXiv preprint arXiv:2503.12284* (2025).

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. TensoRF: Tensorial radiance fields. In *Eur. Conf. Comput. Vis.*

Jorge Condor, Sebastien Speierer, Lukas Bode, Aljaz Bozic, Simon Green, Piotr Didyk, and Adrian Jarabo. 2025. Don't Splat your Gaussians: Volumetric Ray-Traced Primitives for Modeling and Rendering Scattering and Emissive Media. *ACM Trans. Graph.* (2025).

Robert L. Cook and Kenneth E. Torrance. 1982. A Reflectance Model for Computer Graphics. *ACM Trans. Graph.* 1, 1 (1982), 7–24.

Jian Gao, Chun Gu, Youtian Lin, Zhihao Li, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. 2024. Relightable 3D gaussians: Realistic point cloud relighting with brdf decomposition and ray tracing. In *Eur. Conf. Comput. Vis.*

Gonzalo Martin Garcia, Karim Abou Zeid, Christian Schmidt, Daan De Geus, Alexander Hermans, and Bastian Leibe. 2025. Fine-tuning image-conditional diffusion models is easier than you think. In *IEEE/CVF Winter Conf. App. Comput. Vis.*

Chun Gu, Xiaofei Wei, Zixuan Zeng, Yuxuan Yao, and Li Zhang. 2024. IRGS: Inter-reflective gaussian splatting with 2D gaussian ray tracing. *arXiv preprint arXiv:2412.15867* (2024).

Yuan-Chen Guo, Di Kang, Linchao Bao, Yu He, and Song-Hai Zhang. 2022. NeRFReN: Neural radiance fields with reflections. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Florian Hahlbohm, Fabian Friederichs, Tim Weyrich, Linus Franke, Moritz Kappel, Susana Castillo, Marc Stamminger, Martin Eisemann, and Marcus Magnor. 2025. Efficient Perspective-Correct 3D Gaussian Splatting Using Hybrid Transparency. *Comput. Graph. Forum* (2025).

Paul S Heckbert. 1990. Adaptive radiosity textures for bidirectional ray tracing. In *Proc. 17th Ann. Conf. Comp. Graph. Interac. Tech.* 145–154.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conf.*

Letian Huang, Jiayang Bai, Jie Guo, Yuanqi Li, and Yanwen Guo. 2025. On the Error Analysis of 3D Gaussian Splatting and an Optimal Projection Strategy. In *Computer Vision – ECCV 2024*. Springer Nature Switzerland, Cham, 247–263.

Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. 2024. GaussianShader: 3D gaussian splatting with shading functions for reflective surfaces. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. 2023. TensoIR: Tensorial inverse rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. 2024. Marigold: Repurposing diffusion-based image generators for monocular depth estimation. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. 2022. Neural Point Catacaustics for Novel-View Synthesis of Reflections. *ACM Transactions on Graphics* 41, 6 (2022), Article–201.

Dmytro Kotovenko, Olga Grebenkova, and Björn Ommer. 2025. EDGS: Eliminating Densification for Efficient Convergence of 3DGS. *arXiv preprint arXiv:2504.13204* (2025).

Georgios Kouros, Minye Wu, Sushruth Nagesh, Xianling Zhang, and Tinne Tuytelaars. 2024. Unveiling the Ambiguity in Neural Inverse Rendering: A Parameter Compensation Analysis. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Zhiyi Kuang, Yanchao Yang, Siyan Dong, Jiayue Ma, Hongbo Fu, and Youyi Zheng. 2024. OLAT Gaussians for Generic Relightable Appearance Acquisition. In *SIGGRAPH Asia 2024 Conference Papers*. 1–11.

Shuichang Lai, Letian Huang, Jie Guo, Kai Cheng, Bowen Pan, Xiaoxiao Long, Jiangjing Lyu, Chengfei Lv, and Yanwen Guo. 2025. GlossyGS: Inverse Rendering of Glossy Objects With 3D Gaussian Splatting. *IEEE Transactions on Visualization and Computer Graphics* (2025), 1–14. https://doi.org/10.1109/TVCG.2025.3547063

Junseo Lee, Seokwon Lee, Jungi Lee, Junyong Park, and Jaewoong Sim. 2024. GSCore: Efficient Radiance Field Rendering via Architectural Support for 3D Gaussian Splatting. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 497–511. https://doi.org/10.1145/3620666.3651385

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–11.

Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. 2023. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Zhibing Li, Tong Wu, Jing Tan, Mengchen Zhang, Jiaqi Wang, and Dahua Lin. 2025. IDArb: Intrinsic Decomposition for Arbitrary Number of Input Views and Illuminations. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=uuef1HP6X7

Ruofan Liang, Huiting Chen, Chunlin Li, Fan Chen, Selvakumar Panneer, and Nandita Vijaykumar. 2023a. EnvIDR: Implicit differentiable renderer with neural environment lighting. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Zhi-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, and Zian Wang. 2025a. DiffusionRenderer: Neural Inverse and Forward Rendering with Video Diffusion Models. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Zhi-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, and Zian Wang. 2025b. DiffusionRenderer: Neural Inverse and Forward Rendering with Video Diffusion Models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. 2023b. Gs-ir: 3d gaussian splatting for inverse rendering. *arXiv preprint arXiv:2311.16473* (2023).

Jiayue Liu, Xiao Tang, Freeman Cheng, Roy Yang, Zhihao Li, Jianzhuang Liu, Yi Huang, Jiaqi Lin, Shiyong Liu, Xiaofei Wu, et al. 2024. MirrorGaussian: Reflecting 3D gaussians for reconstructing mirror reflections. In *Eur. Conf. Comput. Vis.*

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

Li Ma, Vasu Agrawal, Haithem Turki, Changil Kim, Chen Gao, Pedro Sander, Michael Zollhöfer, and Christian Richardt. 2024. SpecNeRF: Gaussian directional encoding for specular reflections. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Stephen Robert Marschner. 1998. *Inverse rendering for computer graphics*. Cornell University.

Jiarui Meng, Haijie Li, Yanmin Wu, Qiankun Gao, Shuzhou Yang, Jian Zhang, and Siwei Ma. 2024. Mirror-3DGS: Incorporating mirror reflections into 3D gaussian splatting. In *IEEE Int. Conf. Vis. Comm. Image Proc.*

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Eur. Conf. Comput. Vis.*

Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 2024. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM Trans. Graph.* 43, 6 (2024), 1–19.

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (ToG)* 38, 6 (2019), 1–17.

Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. 2010. OptiX: A General Purpose Ray Tracing Engine. *ACM Trans. Graph.* (August 2010).

Yohan Poirier-Ginter, Alban Gauthier, Julien Phillip, J-F Lalonde, and George Drettakis. 2024. A Diffusion Approach to Radiance Field Relighting using Multi-Illumination Synthesis. *Comput. Graph. Forum* 43, 4 (2024).

Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. 2021. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Yahao Shi, Yanmin Wu, Chenming Wu, Xing Liu, Chen Zhao, Haocheng Feng, Jian Zhang, Bin Zhou, Errui Ding, and Jingdong Wang. 2025. GIR: 3D Gaussian Inverse Rendering for Relightable Scene Factorization. *IEEE Transactions on Transactions on Pattern Analysis and Machine Intelligence* (2025).

Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. 2021. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Zhe Jun Tang and Tat-Jen Cham. 2024. 3IGS: Factorised tensorial illumination for 3d gaussian splatting. In *Eur. Conf. Comput. Vis.*

Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. 2022. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Dor Verbin, Pratul P Srinivasan, Peter Hedman, Ben Mildenhall, Benjamin Attal, Richard Szeliski, and Jonathan T Barron. 2024. NeRF-casting: Improved view-dependent appearance with consistent reflections. In *ACM SIGGRAPH Asia Conf.*

Ingo Wald, Nate Morrical, Stefan Zellmann, Lei Ma, Will Usher, Tiejun Huang, and Valerio Pascucci. 2020. Using Hardware Ray Transforms to Accelerate Ray/Primitive

Intersections for Long, Thin Primitive Types. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2, Article 17 (Aug. 2020), 16 pages. https://doi.org/10.1145/3406179

Fangjinhua Wang, Marie-Julie Rakotosaona, Michael Niemeyer, Richard Szeliski, Marc Pollefeys, and Federico Tombari. 2024. UniSDF: Unifying neural representations for high-fidelity 3d reconstruction of complex scenes with reflections. In *Adv. Neural Inform. Process. Syst.*

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Adv. Neural Inform. Process. Syst.*

Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. 1988. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph.* 22, 4 (June 1988), 85–92. https://doi.org/10.1145/378456.378490

Qi Wu, Janick Martinez Esturo, Ashkan Mirzaei, Nicolas Moenne-Loccoz, and Zan Gojcic. 2025. 3dgut: Enabling distorted cameras and secondary rays in gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 26036–26046.

Chen Xi, Peng Sida, Yang Dongchen, Liu Yuan, Pan Bowen, Lv Chengfei, and Zhou. Xiaowei. 2024. IntrinsicAnything: Learning Diffusion Priors for Inverse Rendering Under Unknown Illumination. *arxiv: 2404.11593* (2024).

Tao Xie, Xi Chen, Zhen Xu, Yiman Xie, Yudong Jin, Yujun Shen, Sida Peng, Hujun Bao, and Xiaowei Zhou. 2025. EnvGS: Modeling view-dependent appearance with environment gaussian. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Guangkai Xu, Yongtao Ge, Mingyu Liu, Chengxiang Fan, Kangyang Xie, Zhiyue Zhao, Hao Chen, and Chunhua Shen. 2025. What Matters When Repurposing Diffusion Models for General Dense Perception Tasks?. In *Int. Conf. Learn. Represent.*

Jason C Yang, Justin Hensley, Holger Grün, and Nicolas Thibieroz. 2010. Real-time concurrent linked list construction on the GPU. In *Comput. Graph. Forum*, Vol. 29. 1297–1304.

Yuxuan Yao, Zixuan Zeng, Chun Gu, Xiatian Zhu, and Li Zhang. 2024. Reflective Gaussian Splatting. *arXiv preprint arXiv:2412.19282* (2024).

Keyang Ye, Qiming Hou, and Kun Zhou. 2024. 3D gaussian splatting with deferred reflection. In *ACM SIGGRAPH Conf.*

Ze-Xin Yin, Peng-Yi Jiao, Jiaxiong Qiu, Ming-Ming Cheng, and Bo Ren. 2025. MS-NeRF: Multi-Space Neural Radiance Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* (2025).

Junyi Zeng, Chong Bao, Rui Chen, Zilong Dong, Guofeng Zhang, Hujun Bao, and Zhaopeng Cui. 2023. Mirror-NeRF: Learning neural radiance fields for mirrors with whitted-style ray tracing. In *ACM Int. Conf. Multimedia.*

Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. 2024. RGB↔X: Image decomposition and synthesis using material- and lighting-aware diffusion models. In *ACM SIGGRAPH Conf.*

Rui Zhang, Tianyue Luo, Weidong Yang, Ben Fei, Jingyi Xu, Qingyuan Zhou, Keyi Liu, and Ying He. 2024. RefGaussian: Disentangling Reflections from 3D Gaussian Splatting for Realistic Rendering. *arXiv preprint arXiv:2406.05852* (2024).

Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. 2021. NeRFactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.* 40, 6 (2021), 1–18.

Youjia Zhang, Anpei Chen, Yumin Wan, Zikai Song, Junqing Yu, Yawei Luo, and Wei Yang. 2025. Ref-GS: Directional Factorization for 2D Gaussian Splatting. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*

Jingsen Zhu, Fujun Luan, Yuchi Huo, Zihao Lin, Zhihua Zhong, Dianbing Xi, Rui Wang, Hujun Bao, Jiaxiang Zheng, and Rui Tang. 2022. Learning-based inverse rendering of complex indoor scenes with differentiable Monte Carlo raytracing. In *ACM SIGGRAPH Asia Conf.*

Fig. 14. Disentanglement results of different methods on Shiny kitchen and Shiny Livingroom. Inlays show that our method does not mix reflections into diffuse. Note that in these results, our method uses ground truth input buffers and EnvGS uses ground truth normals.

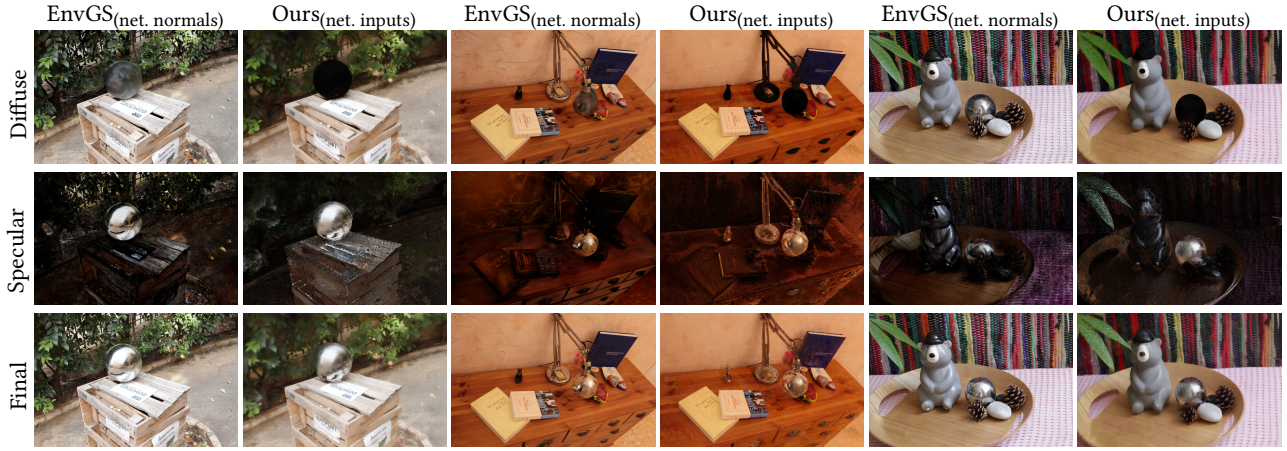

Fig. 15. Our method compared against EnvGS [Xie et al. 2025] on real scenes from the Neural Catacaustics dataset [Kopanas et al. 2022]. Qualitatively, we achieve better disentanglement at the cost of lower visual fidelity in the final render. Note that the denoiser was only applied to the final image.