Relightable 3D Gaussian: Real-time Point Cloud Relighting with BRDF Decomposition and Ray Tracing

Jian Gao[1*] Chun Gu[2*] Youtian Lin[1] Hao Zhu[1] Xun Cao[1] Li Zhang[2✉] Yao Yao[1✉]
[1] Nanjing University [2] Fudan University

Abstract

We present a novel differentiable point-based rendering framework for material and lighting decomposition from multi-view images, enabling editing, ray-tracing, and real-time relighting of the 3D point cloud. Specifically, a 3D scene is represented as a set of relightable 3D Gaussian points, where each point is additionally associated with a normal direction, BRDF parameters, and incident lights from different directions. To achieve robust lighting estimation, we further divide incident lights of each point into global and local components, as well as view-dependent visibilities. The 3D scene is optimized through the 3D Gaussian Splatting technique while BRDF and lighting are decomposed by physically-based differentiable rendering. Moreover, we introduce an innovative point-based ray-tracing approach based on the bounding volume hierarchy for efficient visibility baking, enabling real-time rendering and relighting of 3D Gaussian points with accurate shadow effects. Extensive experiments demonstrate improved BRDF estimation and novel view rendering results compared to state-of-the-art material estimation approaches. Our framework showcases the potential to revolutionize the mesh-based graphics pipeline with a relightable, traceable, and editable rendering pipeline solely based on point cloud. Project page:https://nju-3dv.github.io/projects/Relightable3DGaussian/.

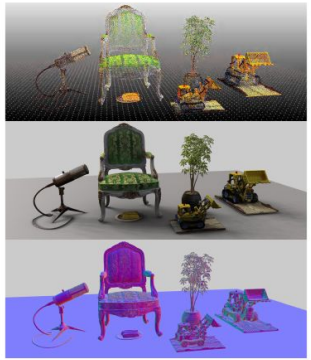


Figure 1. **Multi-object composition and realistic relighting** through our proposed relightable 3D Gaussian. From top to bottom: 1) optimized point cloud with BRDF attributes (colored with base color) 2) physically based relighting with shadow effects via point-based ray tracing, and 3) rendered normal map.

1. Introduction

Reconstructing 3D scenes from multi-view images for photo-realistic rendering is a fundamental problem at the intersection of computer vision and graphics. With the recent development of Neural Radiance Field (NeRF) [27], differentiable rendering techniques have gained tremendous popularity and have demonstrated extraordinary ability in image-based novel view synthesis. However, despite the prevalence of NeRF, both training and rendering of the neural implicit representation require substantial time investments, posing an insurmountable challenge for real-time rendering. Researchers have collectively acknowledged that the efficiency bottleneck lies in the query of the neural field. To tackle the slow sampling problem, different acceleration algorithms have been proposed, including the utilization of grid-based structures [10, 28] and pre-computation [17] for advanced baking.

Recently, 3D Gaussian Splatting (3DGS) [22] has been proposed and has gained significant attention from the community. The method employs a set of 3D Gaussian points to represent a 3D scene and projects these points onto a designated view through a title-based rasterization. Attributes

*Equally contributed.

Relightable 3D Gaussian: Real-time Point Cloud Relighting with BRDF Decomposition and Ray Tracing

Alejandro Sztrajman

December 19th, 2023

3D Gaussian Splatting: Brief Introduction

3D Gaussian Splatting for Real-Time Radiance Field Rendering

BERNHARD KERBL*, Inria, Université Côte d'Azur, France
GEORGIOS KOPANAS*, Inria, Université Côte d'Azur, France
THOMAS LEIMKÜHLER, Max-Planck-Institut für Informatik, Germany
GEORGE DRETTAKIS, Inria, Université Côte d'Azur, France

Fig. 1. Our method achieves real-time rendering of radiance fields with quality that equals the previous method with the best quality [Barron et al. 2022], while only requiring optimization times competitive with the fastest previous methods [Fridovich-Keil and Yu et al. 2022; Müller et al. 2022]. Key to this performance is a novel 3D Gaussian scene representation coupled with a real-time differentiable renderer, which offers significant speedup to both scene optimization and novel view synthesis. Note that for comparable training times to InstantNGP [Müller et al. 2022], we achieve similar quality to theirs; while this is the maximum quality they reach, by training for 51min we achieve state-of-the-art quality, even slightly better than Mip-NeRF360 [Barron et al. 2022].

Radiance Field methods have recently revolutionized novel-view synthesis of scenes captured with multiple photos or videos. However, achieving high visual quality still requires neural networks that are costly to train and render, while recent faster methods inevitably trade off speed for quality. For unbounded and complete scenes (rather than isolated objects) and 1080p resolution rendering, no current method can achieve real-time display rates. We introduce three key elements that allow us to achieve state-of-the-art visual quality while maintaining competitive training times and importantly allow high-quality real-time (≥ 30 fps) novel-view synthesis at 1080p resolution. First, starting from sparse points produced during camera calibration, we represent the scene with 3D Gaussians that preserve desirable properties of continuous volumetric radiance fields for scene optimization while avoiding unnecessary computation in empty space; Second, we perform interleaved optimization/density control of the 3D Gaussians, notably optimizing anisotropic covariance to achieve an accurate representation of the scene; Third, we develop a fast visibility-aware rendering algorithm that supports anisotropic splatting and both accelerates training and allows real-time rendering. We demonstrate state-of-the-art visual quality and real-time rendering on several established datasets.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Point-based models**; **Rasterization**; **Machine learning approaches**.

*Both authors contributed equally to the paper.

Authors' addresses: Bernhard Kerbl, bernhard.kerbl@inria.fr, Inria, Université Côte d'Azur, France; Georgios Kopanas, georgios.kopanas@inria.fr, Inria, Université Côte d'Azur, France; Thomas Leimkühler, thomas.leimkuehler@mpi-inf.mpg.de, Max-Planck-Institut für Informatik, Germany; George Drettakis, george.drettakis@inria.fr, Inria, Université Côte d'Azur, France.

0730-0301/2018/0-ART0 $15.00
https://doi.org/XXXXXXX.XXXXXXX

Additional Key Words and Phrases: novel view synthesis, radiance fields, 3D gaussians, real-time rendering

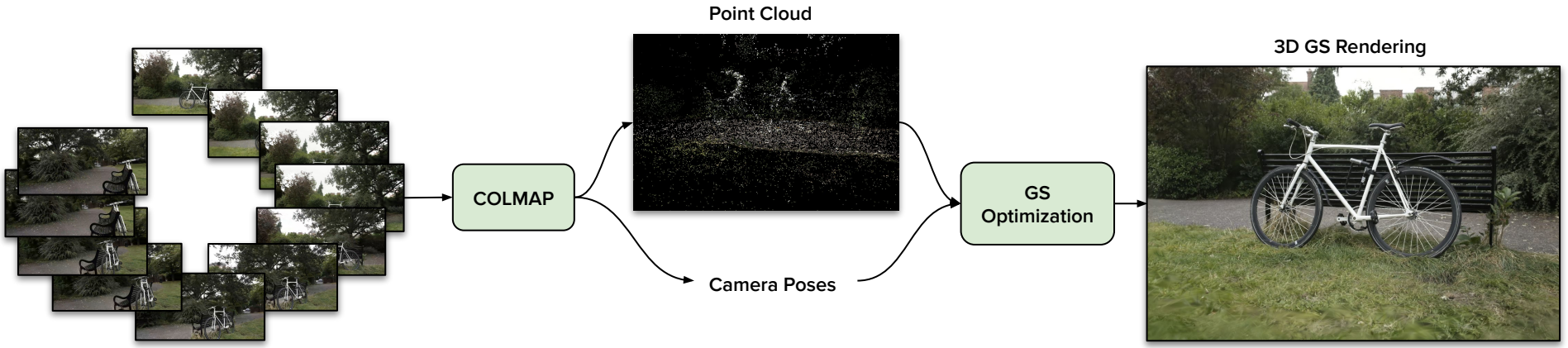
ACM Reference Format:
Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2018. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 0, 0, Article 0 (2018), 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

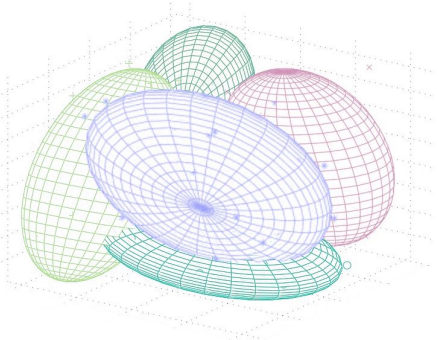
Meshes and points are the most common 3D scene representations because they are explicit and are a good fit for fast GPU/CUDA-based rasterization. In contrast, recent Neural Radiance Field (NeRF) methods build on continuous scene representations, typically optimizing a Multi-Layer Perceptron (MLP) using volumetric ray-marching for novel-view synthesis of captured scenes. Similarly, the most efficient radiance field solutions to date build on continuous representations by interpolating values stored in, e.g., voxel [Fridovich-Keil and Yu et al. 2022] or hash [Müller et al. 2022] grids or points [Xu et al. 2022]. While the continuous nature of these methods helps optimization, the stochastic sampling required for rendering is costly and can result in noise. We introduce a new approach that combines the best of both worlds: our 3D Gaussian representation allows optimization with state-of-the-art (SOTA) visual quality and competitive training times, while our tile-based splatting solution ensures real-time rendering at SOTA quality for 1080p resolution on several previously published datasets [Barron et al. 2022; Hedman et al. 2018; Knapitsch et al. 2017] (see Fig. 1).

Our goal is to allow real-time rendering for scenes captured with multiple photos, and create the representations with optimization times as fast as the most efficient previous methods for typical real scenes. Recent methods achieve fast training [Fridovich-Keil

3D Gaussian Splatting: Brief Introduction



$$G(\boldsymbol{x}) = exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))$$



For each 3D Gaussian we have parameters:

- Mean $\boldsymbol{\mu}$
- Covariance Σ
- Max Opacity **o**
- View-dependent colour ***c*** **(4th-order SHs).**

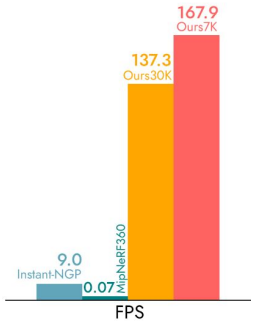
Alpha Blending:

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

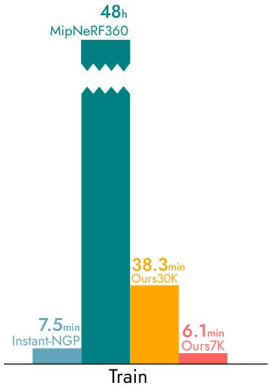
C: colour map. α: 3D Gaussian opacity.

Opacity = 1

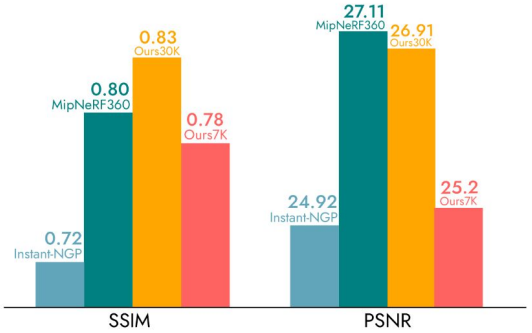
3D Gaussian Splatting: Brief Introduction



Fast rendering based on rasterization.

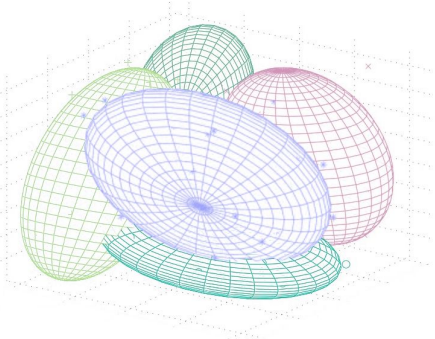


Fast training (10 - 30 min in single GPU) due to good initialization with point cloud.



High-quality (comparable to MipNeRF360).

Relightable 3D Gaussian Splatting



$$G(\boldsymbol{x}) = exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))$$

For each 3D Gaussian we have parameters:

- Mean μ
- Covariance Σ
- Max Opacity **o**
- View-dependent colour ***c*** **(4th-order SHs).**

New set of parameters per Gaussian:

- Mean: μ
- Covariance: Σ
- Max Opacity: ***o***
- View-dependent colour: **c** (4th-order SHs).
- Normal: ***n***] Geometry
- Global incident light visibility: **v** (SHs).] Light
- Local incident light: **l** (SHs).
- Base colour: ***b***] Material
- Roughness: ***r***
- Metallicness: ***m***

Relightable 3D Gaussian Splatting: Normal Estimation

A potential methodology involves treating the spatial means of 3D Gaussians as a conventional point cloud and executing normal estimation based on the local planar assumption. However, this approach is hindered by the often sparse density of the points and, more critically, their *soft* nature, signifying a non-precise alignment with the object surface, which can lead to an inaccurate estimate.

To address these limitations, we propose an optimization of $\boldsymbol{n}$ from initial random vectors via back-propagation. This process entails rendering the depth and normal map for a specified viewpoint:

$$\{\mathcal{D}, \mathcal{N}\} = \sum_{i \in N} T_i \alpha_i \{d_i, \boldsymbol{n}_i\}, \tag{3}$$

where, d_i and $\boldsymbol{n}_i$ denote the depth and normal of the point. We then encourage the consistency between the rendered normal N and the pseudo normal $\tilde{\mathcal{N}}$, computed from the rendered depth $\mathcal{D}$ under the local planarity assumption. The normal consistency is quantified as follows:

$$\mathcal{L}_n = \|\mathcal{N} - \tilde{\mathcal{N}}\|_2. \tag{4}$$

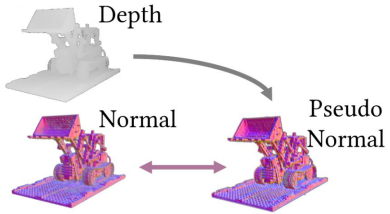
In regular point clouds, normals can be estimated by assuming planar surfaces between neighbouring points. But:

- Sparcity of points can lead to inaccurate normals.
- These are not regular point clouds.

1) For each camera pose, we can compute the depth $\mathbf{d}_i$ for each Gaussian.
2) Then we can compute depth and normal maps D and N, using alpha blending (as used for the colour map).

$$\mathcal{C} = \sum_{i \in N} T_i \alpha_i \boldsymbol{c}_i \longrightarrow \{\mathcal{D}, \mathcal{N}\} = \sum_{i \in N} T_i \alpha_i \{d_i, \boldsymbol{n}_i\},$$

3) With the depth map D we can compute a pseudo-normal map N'.
4) Finally we enforce consistency between N and N'.



Relightable 3D Gaussian Splatting: Light Representation

Incident Light Modeling Optimizing a NeILF for each Gaussian directly can be overly unconstrained, making it difficult to accurately decompose incident light from its appearance. We apply a prior by dividing the incident light into global and local components. The sampled incident light at a Gaussian from direction $\boldsymbol{\omega}_i$ is represented as:

$$L_i(\boldsymbol{\omega}_i) = V(\boldsymbol{\omega}_i) \cdot L_{global}(\boldsymbol{\omega}_i) + L_{local}(\boldsymbol{\omega}_i), \quad (8)$$

where the visibility term $V(\boldsymbol{\omega}_i)$ and the local light term $L_{local}(\boldsymbol{\omega}_i)$ are parameterized as Spherical Harmonics (SH) for each Gaussian, denoted as $\boldsymbol{v}$ and $\boldsymbol{l}$ respectively. The global light term is parameterized as a globally shared SH, denoted as $\boldsymbol{l}^{env}$. For each 3D Gaussian, we sample N_s

Not using NeILF.

Incident light is split between local and global:
$V(w_i)$ and $L_{Local}(w_i)$: different for each Gaussian.
$L_{Global}(w_i)$: same for all Gaussians.

All three terms are encoded as Spherical Harmonics.

Relightable 3D Gaussian Splatting: Material Representation

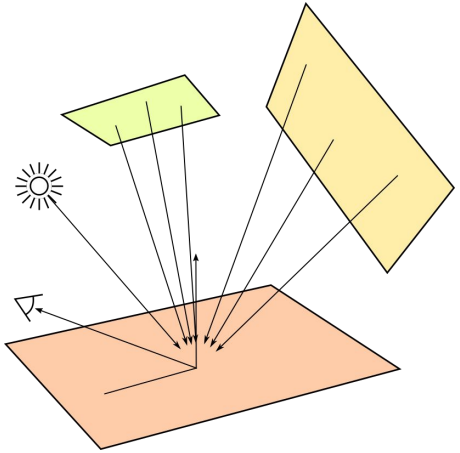
BRDF Parameterization As stated above, We assign additional BRDF properties to each Gaussian: a base color $\boldsymbol{b} \in [0,1]^3$, a roughness $r \in [0,1]$ and a metallic $m \in [0,1]$. We adopt the simplified Disney BRDF model as in [42]. The BRDF function f in Eq. 6 is divided into a diffuse term $f_d = \frac{1-m}{\pi} \cdot \boldsymbol{b}$ and a specular term:

$$f_s(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i) = \frac{D(\boldsymbol{h}; r) \cdot F(\boldsymbol{\omega}_o, \boldsymbol{h}; \boldsymbol{b}, m) \cdot G(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, h; r)}{(\boldsymbol{n} \cdot \boldsymbol{\omega}_i) \cdot (\boldsymbol{n} \cdot \boldsymbol{\omega}_o)}, \tag{7}$$

where $\boldsymbol{h}$ is the half vector, D, F and G represent the normal distribution function, Fresnel term and geometry term.

Analytic BRDF prior with metallic-roughness workflow.

Parameters are:
Base colour (b): RGB diffuse colour.
Metallicness (m): relative weight of specular term.
Roughness (r): approx. width of specular lobes.



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

Outgoing light

BRDF (material)

Incident light

Normal (geometry)

Integral over hemisphere of incident light

For each Gaussian, colour is computed with Monte Carlo integration (24 samples):

$$\boldsymbol{c}'(\boldsymbol{\omega_o}) = \sum_{i=0}^{N_s} (f_d + f_s(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)) L_i(\boldsymbol{\omega}_i)(\boldsymbol{\omega}_i \cdot \boldsymbol{n}) \Delta\boldsymbol{\omega}_i, \tag{9}$$

Alpha Blending

$$\mathcal{C}' = \sum_{i \in N} T_i \alpha_i \boldsymbol{c}'_i.$$

Relightable 3D Gaussian Splatting: Regularizations

Bilateral Smoothness We expect that the BRDF properties will not change drastically in areas with smooth color [42]. We define a smooth constraint on metallic as:

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

where M is the rendered metallic map, given by $M = \sum_{i \in N} T_i \alpha_i m_i$. Similarly, we also define smoothness constraints $L_{s,r}$ and $L_{s,b}$ on roughness and base color.

Variations of M can only be large in regions where variations of colour are large.

This regularization is applied to metallicness, roughness and base colour.

Light Regularization We apply light regularization assuming a near-natural white incident light [26].

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

Regularization term:
Forces light to be approx. white.

Base Colour Regularization: remove shadows and highlights from images and enforce this to be approx. base colour.

Relightable 3D Gaussian Splatting: Pipeline

5.1. Training Details

To ensure stable optimization, the training procedure is divided into two stages. Firstly, we optimize an 3DGS [22] model, augmented with an additional normal vector $\boldsymbol{n}$ (Sec. 3.2). We also add normal gradient condition for adaptive density control. Subsequently, with the already stable geometry from the first stage, we begin with the ray tracing method (Sec. 4.1) to bake the visibility term $\boldsymbol{v}$, and then optimize the entire parameter set using the comprehensive pipeline described in Sec. 3.3. During the second stage, we sample $N_s = 24$ rays per Gaussian for PBR. Tab. 1 provides a complete list of the used losses and their weights. We train our model for 30,000 iterations in the initial stage and 10,000 iterations in the latter. And all experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU. Please see the supplementary material for more details.

Training stage 1:

- Optimize 3DGS with normals.

Training stage 2:

- Apply ray-tracing to compute the per-gaussian visibility and bake it into SHs.
- Optimize entire set of parameters.

Speed and Memory To train a NeRF synthetic scene, our model typically requires approximately 16 minutes and 10 GB of memory. Notably, the visibility baking in the second stage is remarkably brief, lasting only a few seconds. Following this optimization, our model attains real-time rendering across different lighting conditions, achieving 120 frames per second.

Training and rendering speeds are similar to the original 3DGS method. For a synthetic scene:

- Training time: 16 min (single GeForce RTX 3090).
- Rendering: 120 FPS.
- GPU memory: 10 GB.

Relightable 3D Gaussian Splatting: Results on Synthetic

Table 2. Quantitative results for NVS on NeRF synthetic dataset.

	Geometry	Relighting	PSNR ↑	SSIM ↑	LPIPS ↓
NPBG [1]	point	✘	28.10	0.923	0.077
NPBG++ [34]	point	✘	28.12	0.928	0.076
FreqPCR [52]	point	✘	31.24	0.950	0.049
3DGS [22]	point	✘	**33.88**	**0.970**	**0.031**
PhySG [49]	neural	✔	18.91	0.847	0.182
Nvdiffrec [29]	mesh	✔	29.05	0.939	0.081
NeILF++ [47]	neural	✔	26.37	0.911	0.091
Ours	point	✔	**31.63**	**0.960**	**0.043**

Results of Novel View Synthesis (NVS) on synthetic scenes, in terms of 3 image quality metrics (PSNR, SSIM, LPIPS).

Quality is a bit lower than vanilla 3DGS, but higher than other inverse rendering methods.

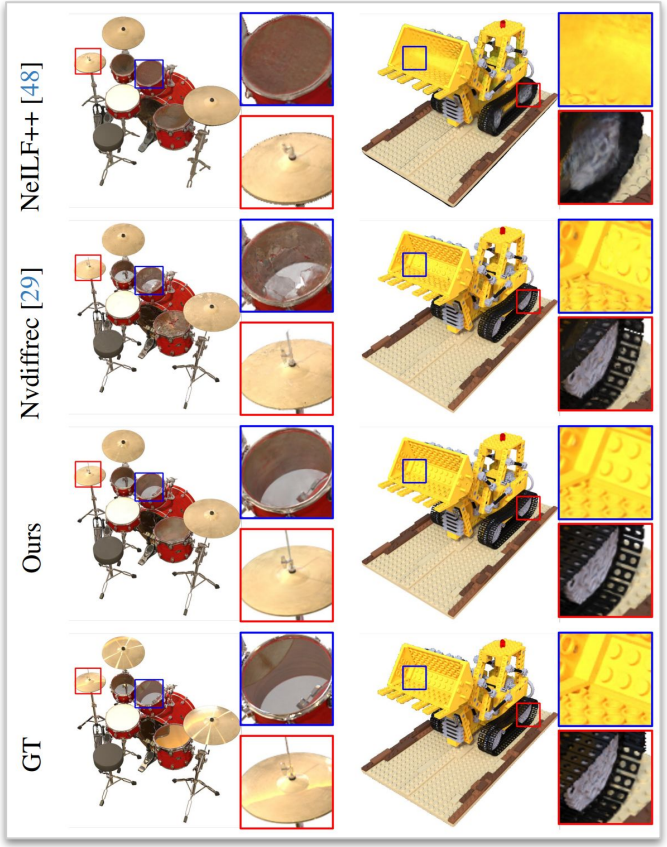


Figure 4. Qualitative results of novel view synthesis on NeRF synthetic dataset

Relightable 3D Gaussian Splatting: Results on Real

Table 3. Quantitative results on real-world DTU dataset.

DTU Scan	Mean
NerFactor [51]	24.28
PhySG [49]	19.11
Neual-PIL [9]	19.94
Nvdiffrec [29]	21.91
NeILF++ [47]	28.61
Ours	**30.95**

Table 6. Training time (time for optimization) and Rendering time (time per rendered image).

	Training Time	Rendering Time
PhySG [49]	9h	5s
Nvdiffrec []	1.5h	**5ms**
NeILF++ [47]	4h	14s
NerFactor [51]	days	-
Neual-PIL [9]	days	-
Ours	**16min**	8ms

Results of Novel View Synthesis (NVS) on real-world scenes, in terms of 3 image quality metrics (PSNR, SSIM, LPIPS).

Quality is better than other inverse rendering methods.

Training and rendering speeds are also much higher.

But: no comparison with NVS-only methods.

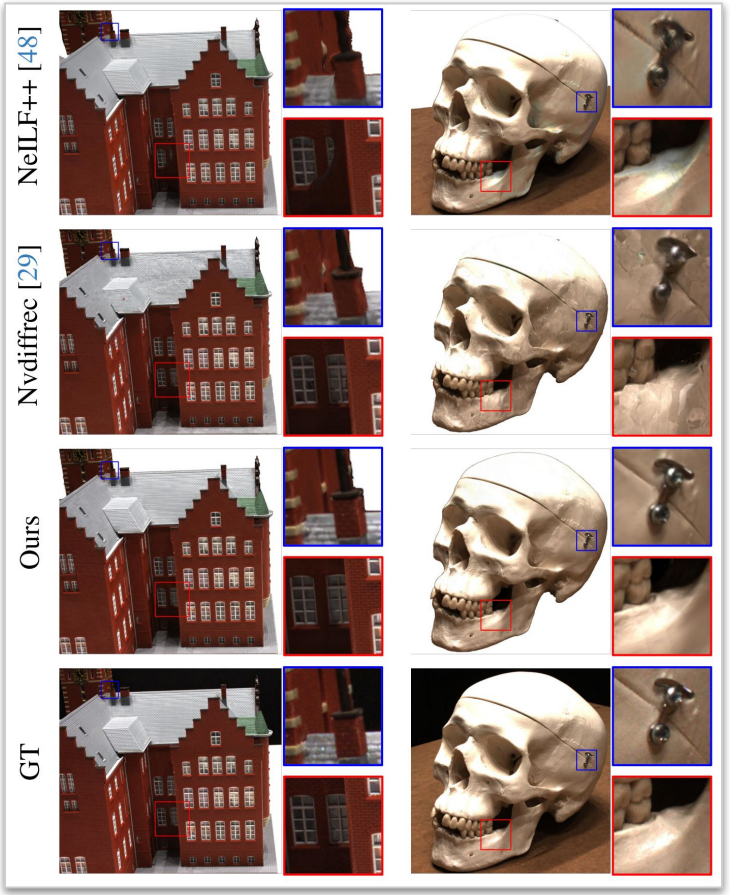


Figure 4. Qualitative results of novel view synthesis on DTU dataset

Relightable 3D Gaussian Splatting: Results of BRDF Estimation

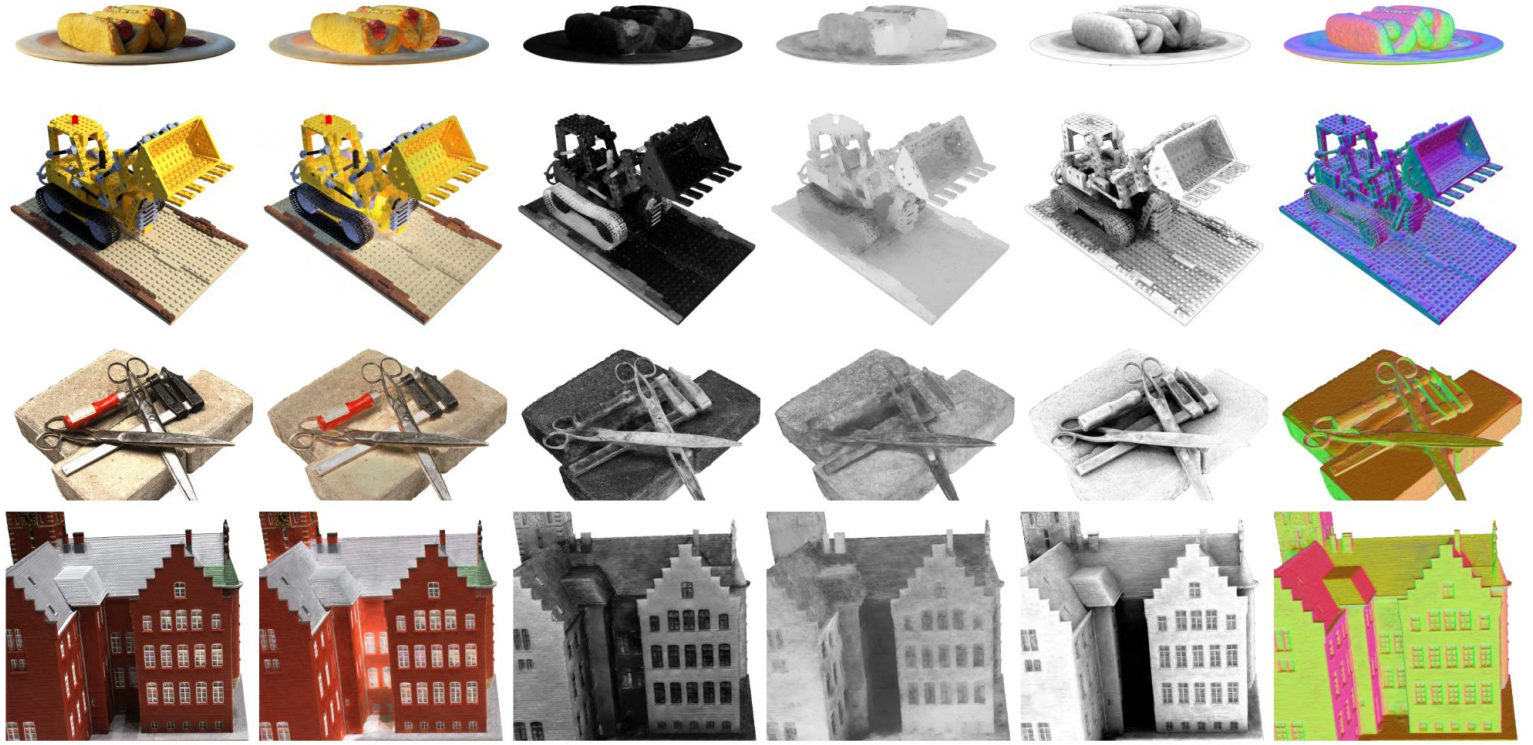
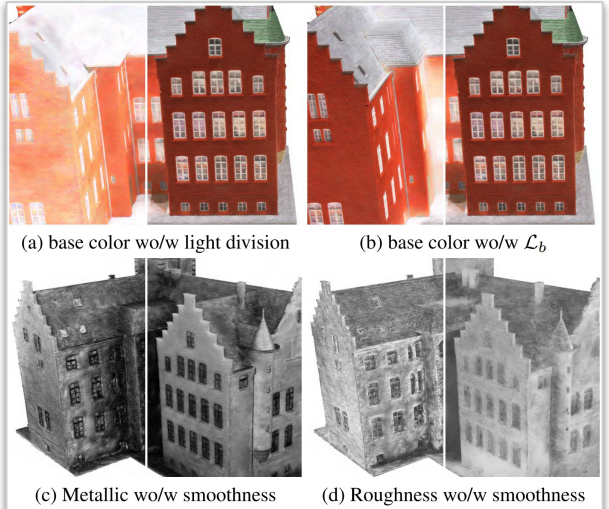


Figure 5. Qualitative results of BRDF estimation. Here we visualize the rendered average visibility (ambient occlusion) as well.

Relightable 3D Gaussian Splatting: Ablations



(a) base color wo/w light division (b) base color wo/w $\mathcal{L}_b$

(c) Metallic wo/w smoothness (d) Roughness wo/w smoothness

Figure 6. Ablation Studies on main components of our method.

Ablations:

- Light division
- Base colour regularization
- Bilateral smoothness regularizations

Table 4. Ablation on sample number. Average scores are reported.

	PSNR ↑	SSIM ↑	LPIPS ↓	Time (min)	FPS
N_s=12	31.42	0.959	0.043	**13.77**	**150**
N_s=24	31.63	0.960	0.043	15.68	120
N_s=48	**31.78**	**0.961**	**0.042**	19.52	87

Analysis of number of samples for BRDF estimation.

Relightable 3D Gaussian Splatting: Relighting

https://nju-3dv.github.io/projects/Relightable3DGaussian/

Relightable 3D Gaussian Splatting: Limitations

Limitations and Future work Our method comes with limitations. It does not handle well for unbordered scenes and requires the presence of object masks during the optimization process. We have noticed an adverse impact from the background on our optimization process, likely due to the unique characteristics of the point clouds generated by 3D Gaussian Splatting. These point clouds resemble particles with color and some transparency, diverging from conventional *surface* points. Although we've integrated off-the-shelf Multi-View Stereo (MVS) cues for geometry enhancement, the resulting geometry still falls short of our expectations. Considering the potential benefits, exploring the integration of MVS into our optimization process for more accurate geometry stands as a promising avenue for future research.