Inverse Rendering using 2D-Gaussian Splatting

Chuhao Chen
chc091@ucsd.edu
University of California San Diego

Ziyang Fu
zifu@ucsd.edu
University of California San Diego

Sudhansh Peddabomma
speddabomma@ucsd.edu
University of California San Diego

Zejia Wu
zew024@ucsd.edu
University of California San Diego

ABSTRACT

In this work, we present our method 2D-GSIR, an inverse-rendering approach based on the seminal works GS-IR and 2DGS. Approaches using 3D Gaussian Splatting for inverse rendering use various heuristics to estimate surface normals as 3D Gaussians do not have an associated normal. Addressing this limitation, recent work in 2D Gaussian Splatting proposed reconstructing scenes using 2D Gaussians to represent surface robustly while having well-defined normals. We build our approach leveraging ideas from previous works - using baking to accurately map the occlusion dependencies, correcting for depth distortion using appropriate regularizer, and ensuring the normals are smooth and consistent. We are able to produce state-of-the-art results while reducing the training time significantly as compared to the previous approaches using 3D Gaussian Splatting.

1 INTRODUCTION

The concept of "vision as inverse graphics" has been captivating for some time. The goal has been to solve inverse rendering problems, which involve recovering shape, material, and lighting from images. Recently, neural rendering methods [2, 3, 5] have gained significant attention for their impressive success in various tasks, including shape reconstruction, novel view synthesis, non-physically-based relighting, and surface reflectance map estimation. These methods use scene representations that can be physical, neural, or a mix of both, combined with a neural-network-based renderer. While methods that reconstruct textures or radiance fields excel at generating new views, they do not separate appearance into lighting and materials. This limitation prevents physically-based appearance manipulation, such as material editing or relighting.

3D Gaussian Splatting (3DGS) [5] has recently become a promising technique for modeling 3D static scenes, significantly improving rendering speed to real-time levels. It makes scene representation more compact and achieves fast, high-quality novel view synthesis. Integrating 3DGS into the inverse rendering process, including geometry reconstruction, material decomposition, and illumination estimation, is both natural and essential. Unlike NeRF's ray tracing, 3DGS generates a set of 3D Gaussians around sparse points. However, the adaptive control of Gaussian density during 3DGS optimization can lead to loose geometry, making accurate normal estimation challenging. Therefore, a well-designed strategy is needed to regularize 3DGS's normal estimation. We choose 2D Gaussian Splatting (2DGS) [3], a new method for modeling and reconstructing geometrically accurate radiance fields from multi-view images. The main idea is to reduce the 3D volume to a set of 2D oriented planar Gaussian disks. Unlike 3D Gaussians, 2D Gaussians provide consistent geometry while modeling surfaces directly. To accurately capture thin surfaces and ensure stable optimization, we use a perspective-accurate 2D splatting process that involves ray-splat intersection and rasterization. This approach allows for detailed, noise-free geometry reconstruction while maintaining high-quality appearance, fast training speed, and real-time rendering.

In this work, we follow a novel inverse rendering based on Gaussian Splatting called GS-IR [6] and replace the 3DGS scene representation with 2DGS. . Our method accurately reconstructs the geometry and materials of complex real scenes under unknown natural illumination, enabling state-of-the-art novel view synthesis and additional applications like relighting. Our key technical contributions are as follows:

(1) **2DGS Scene Representation**: We employ a 2D Gaussian Splatting representation to achieve more precise geometry reconstruction, which significantly improves the quality of relighting results. By collapsing the 3D volume into 2D oriented planar Gaussian disks, we ensure view-consistent geometry and intrinsic surface modeling.
(2) **Hyperparameter Fine-Tuning**: We fine-tune the hyperparameters for 2DGS across various scenes, ensuring optimal performance on both synthetic and real-world datasets. This fine-tuning process allows our approach to adapt to different types of data, resulting in robust and high-fidelity reconstructions.

These contributions enable our method to deliver superior results in geometry and material reconstruction, novel view synthesis, and relighting, showcasing its effectiveness and versatility in various applications.

2 RELATED WORKS

2.1 Neural Representation

Recently, neural rendering techniques like Neural Radiance Field (NeRF) [7]have achieved impressive success in visual computing, leading to many neural representations tailored for different tasks. NeRF models a continuous radiance field using MLPs, which requires numerous repeated queries during training and inference. To address these computational inefficiencies, various neural scene representations have been proposed with more discretized geometry proxies such as hash grids [8], tri-planes [2]. These neural features are stored in a structured manner, allowing for efficient storage and retrieval. Although interpolation techniques can significantly reduce computational costs, they often result in some loss of image quality.

3D Gaussians have been introduced as an unstructured scene representation to balance efficiency and quality. With a specially designed tile-based rasterizer for Gaussian splats, this method achieves real-time rendering with high quality for novel view synthesis.

2.2 Inverse Rendering

Inverse rendering aims to decompose an image's appearance into its geometry, material, and lighting conditions. Due to the inherent ambiguity between observed images and underlying properties, many methods [4, 9] have been proposed with different constraints, such as capturing images with fixed lighting and rotating objects, or capturing with a moving camera and colocated lighting.

Combining neural representations, inverse rendering models simulate how light interacts with the neural volume with various material properties and estimate the lighting and material parameters during optimization. For instance, Neural Reflectance Fields assume a known point light source and represent the scene with volume density, surface normals, and bi-directional reflectance distribution functions (BRDFs) with one-bounce direct illumination. Other methods extend to arbitrary known lighting conditions and train additional models to account for light visibility. Some approaches assume full light source visibility without shadow simulation and use spherical Gaussians for acceleration. Efficient representations enable computation of visibility and indirect lighting by ray tracing, though they may be limited to object-level scenes.

For modeling surface geometry using point clouds, GS-IR [6] proposes a 3DGS-based pipeline to recover the geometry, material, and lighting for both objects and unbounded scenes. However, it struggles to handle intricate geometry (e.g. Lego and Ficus) let alone complex scenes in Mip-NeRF 360 dataset [1].

3 PRELIMINARIES

In this section, we provide the necessary technical background and mathematical definitions foundational to the proposed methods discussed in the subsequent sections.

3.1 3D Gaussian Splatting

3D Gaussian Splatting [5] is an explicit representation of 3D scenes as Gaussian primitives. Each primitive is defined by the Gaussian function

$$\mathcal{G}(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p})^\top \Sigma^{-1}(\mathbf{x}-\mathbf{p})} \tag{1}$$

where $\mathbf{p}$ is the center point and the covariance matrix $\Sigma = \mathbf{RSS}^\top\mathbf{R}^\top$ is facterized into a rotation matrix $\mathbf{R}$ and a scaling matrix $\mathbf{S}$.

To render 3D Gaussian Splatting in the image space, the two learnable parameters $\mathbf{p}$ and Σ are transformed into the camera coordinates, denoted by $\mathbf{p}'$ and Σ', which can be formulated as

$$\mathbf{p}' = \mathbf{KW}[\mathbf{p}, 1]^\top \quad \Sigma' = \mathbf{JW}\Sigma\mathbf{W}^\top\mathbf{J}^\top \tag{2}$$

where $\mathbf{K}$ is the intrinsic matrix of the camera, $\mathbf{W}$ is the extrinsic matrix of the camera for world-to-camera transformation and $\mathbf{J}$ is the Jacobian matrix of the affine approximation of the perspective projection. The Gaussian function in the image space is then transformed as

$$\mathcal{G}'(\mathbf{x}') = e^{-\frac{1}{2}(\mathbf{x}'-\mathbf{p}')^\top \Sigma'^{-1}(\mathbf{x}'-\mathbf{p}')} \tag{3}$$

where $\mathbf{x}'$ is the pixel position of the image transformed in the same way as $\mathbf{p} \mapsto \mathbf{p}'$.

In each 3D Gaussian primitive, we use $\mathbf{c}$ and α to model the appearance. $\mathbf{c}$ represents the view-dependent color, which is parameterized with spherical harmonics. α represents the opacity. The image's pixel color $\mathbf{C}$ at pixel $\mathbf{x}'$ is then obtained from a volumetric alpha-blending process:

$$\mathbf{C}(\mathbf{x}') = \sum_{i=1}^{N} T_i \alpha_i \mathcal{G}'_i(\mathbf{x}')\mathbf{c}_i \quad T_i = \prod_{j=1}^{i-1}(1 - \alpha_j \mathcal{G}'_i(\mathbf{x}')) \tag{4}$$

where $\mathcal{G}'(\mathbf{x}')$ is the Gaussian function with transformed $\mathbf{p}'$ and Σ' mentioned above and T_i is the accumulated transmittance that measures the probability density of i-th Gaussian primitive along the ray.

3.2 2D Gaussian Splatting

Similar to 3D Gaussian Splatting, 2D Gaussian Splatting [3] is also an explicit representation of 3D scenes as Gaussian primitives with some learnable parameters. Unlike 3D Gaussian Splatting, which models each Gaussian primitive with an ellipsoid, 2D Gaussian Splatting uses elliptical disks characterized by the center point $\mathbf{p}$, two principal tangential vectors $\mathbf{t}_u$ and $\mathbf{t}_v$ and a scaling vector $\mathbf{s} = (s_u, s_v)$ to control the mean and variance of the 2D Gaussian function. The normal of the tangent plane for the 2D Gaussian primitive is denoted by $\mathbf{t}_w = \mathbf{t}_u \times \mathbf{t}_v$.

Given the local position (u, v) of a 2D Gaussian primitive, its geometry is represented by a 4D homogeneous transformation matrix $\mathbf{H} \in \mathbb{R}^{4\times4}$, which can be formulated as

$$\mathbf{H} = \begin{bmatrix} s_u\mathbf{t}_u & s_v\mathbf{t}_v & \mathbf{0} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{RS} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \tag{5}$$

where $\mathbf{R} = [\mathbf{t}_u, \mathbf{t}_v, \mathbf{t}_w]$ can be considered as a 3×3 rotation matrix and the diagonal matrix $\mathbf{S}$ is constructed from the scaling vector $\mathbf{s}$. With the local position (u, v), we can also obtain the Gaussian function used in 2D Gaussian Splatting from a standard Gaussian function:

$$\mathcal{G}(u, v) = e^{-\frac{1}{2}(u^2+v^2)} \tag{6}$$

To render 2D Gaussian Splatting in the image space, transformation is applied from the pixel position $\mathbf{x} = (x, y)$ to the local position (u, v). Consider the mapped $\mathbf{x}'$ in the 4D homogeneous coordinates $\mathbf{x}' = (xz, yz, z, 1)^\top$ in which z is the depth, the transformation can then be expressed by

$$\mathbf{x}' = (xz, yz, z, 1)^\top = \mathbf{WH}(u, v, 1, 1)^\top \tag{7}$$

where $\mathbf{W}$ is the extrinsic matrix of the camera and $\mathbf{H}$ is the 4D homogeneous transformation matrix mentioned above. With this form of transformation, we can construct two 4D homogeneous planes $\mathbf{h}_x = (-1, 0, 0, x)^\top$ and $\mathbf{h}_y = (0, -1, 0, y)^\top$, then the corresponding 4D homogeneous planes for u and v can be represented as

$$\mathbf{h}_u = (\mathbf{WH})^\top\mathbf{h}_x \quad \mathbf{h}_v = (\mathbf{WH})^\top\mathbf{h}_y \tag{8}$$

Then the local position (u, v) can be formulated as:

$$u(\mathbf{x}) = \frac{\mathbf{h}_u^{(2)}\mathbf{h}_v^{(4)} - \mathbf{h}_u^{(4)}\mathbf{h}_v^{(2)}}{\mathbf{h}_u^{(1)}\mathbf{h}_v^{(2)} - \mathbf{h}_u^{(2)}\mathbf{h}_v^{(1)}} \quad v(\mathbf{x}) = \frac{\mathbf{h}_u^{(4)}\mathbf{h}_v^{(1)} - \mathbf{h}_u^{(1)}\mathbf{h}_v^{(4)}}{\mathbf{h}_u^{(1)}\mathbf{h}_v^{(2)} - \mathbf{h}_u^{(2)}\mathbf{h}_v^{(1)}} \tag{9}$$

where $\mathbf{h}_u^{(i)}$ and $\mathbf{h}_v^{(i)}$ are the i-th parameter of thee 4D homogeneous plane parameters.

2D Gaussian Splatting also employs an object-space low-pass filter to prevent the elliptical disks from degenerating into line segments when transformed into the image space. With the filter, the new Gaussian function can be formulated as

$$\hat{\mathcal{G}}(\mathbf{u}(\mathbf{x})) = \max\{\mathcal{G}(\mathbf{u}(\mathbf{x})), \mathcal{G}(\frac{\mathbf{x}-\mathbf{p}'}{\sigma})\} \tag{10}$$

where $\mathbf{p}'$ is the projection of the center point $\mathbf{p}$ and $\sigma = \frac{\sqrt{2}}{2}$ is the radius of the filter. This solution ensures sufficient pixels are used during rendering.

2D Gaussian primitives use $\mathbf{c}$ and α to model the appearance similar to those in the 3D Gaussian primitives, and the image color is obtained from the same volumetric alpha-blending process, which can be expressed as

$$\mathbf{C}(\mathbf{x}') = \sum_{i=1}^{N} T_i \alpha_i \hat{\mathcal{G}}_i(\mathbf{u}(\mathbf{x}))\mathbf{c}_i \quad T_i = \prod_{j=1}^{i-1}(1-\alpha_j \hat{\mathcal{G}}_i(\mathbf{u}(\mathbf{x}))) \tag{11}$$

where everything is the same as those in Eq. (4).

3.3 Physically-based Rendering Equation

In the classic Physically-based Rendering Equation, the outgoing radiance of a surface point $\mathbf{x}$ for the direction ω_o is defined as:

$$L_o(\mathbf{x}, \omega_o) = \int_\Omega L_i(\omega_i) f_r(\omega_i, \omega_o)(\omega_i \cdot \mathbf{n}) d\omega_i \tag{12}$$

where $L(\omega_i)$ is the incident radiance and ω_i is the corresponding direction, f_r is the bidirectional reflectance distribution function (BRDF) and $\mathbf{n}$ is the normal at point $\mathbf{x}$. The radiance is an integral in the upper hemisphere denoted by Ω.

We follow the GS-IR [6] and use the Cook-Torrance microfacet model to formulate the BRDF:

$$f_r(\omega_i, \omega_o) = (1-m)\frac{\mathbf{a}}{\pi} + \frac{DFG}{4(\omega_i \cdot \mathbf{n})(\omega_o \cdot \mathbf{n})} \tag{13}$$

where the first term models the diffuse property and the second term models the specular property. The material is determined by three parameters: albedo $a \in [0,1]^3$, metallic $m \in [0,1]$ and roughness $\rho \in [0,1]$. The microfacet distribution function D, Fresnel reflection F and geometric shadowing factor G are all related to ρ.

4 METHOD

In this section, we introduce the method we use to optimize the 2D Gaussian primitives for inverse rendering. The process can be decomposed into three stages: Geometry Reconstruction, Occlusion and Indirect Illumination Bake, and PBR Property Decomposition. Our implementation is released at https://github.com/CzzzzH/GS-IR

4.1 Geometry Reconstruction

4.1.1 Depth Estimation. The depth z_i for the i-th 2D Gaussian primitive at pixel $\mathbf{x} = (x, y)$ can be directly obtained according to the Eqs. (7) to (9). The pixel depth can be approximated either by the mean depth or median depth:

$$z_{mean} = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i + \epsilon} \quad z_{median} = \max\{z_i | T_i > 0.5\} \tag{14}$$

where $w_i = T_i \alpha_i \hat{\mathcal{G}}_i(\mathbf{u}(\mathbf{x}))$ is the weight contribution of the i-th Gaussian primitive and T_i is the accumulated transmittance in Eq. (11). The normalization in mean depth ensures that a 2D Gaussian can be rendered as a planar 2D disk in the depth visualization.

4.1.2 Normal Estimation. The main advantage of using 2DGS over 3DGS is that we can directly use the normal of the elliptical disk as the normal of the Gaussian primitive instead of creating a new property stored in the primitive and optimizing it, which is used in GS-IR. The pixel normal can then be defined as

$$\mathbf{N}(\mathbf{x}) = \sum_{i=1}^{N} T_i \alpha_i \hat{\mathcal{G}}_i(\mathbf{u}(\mathbf{x}))\mathbf{n}_i \tag{15}$$

4.1.3 Loss function. In this stage, we don't incorporate PBR properties and only optimize the Gaussian primitives according to the color reconstruction and geometry consistency. The loss we use follows the original 2DGS.

For color reconstruction, we simply apply $\mathcal{L}_1$ loss between the ground-truth image $\mathbf{I}_{gt}$ and the RGB image $\mathbf{I}_c$ containing the pixel color rendered with Gaussian primitives:

$$\mathcal{L}_c = ||\mathbf{I}_{gt} - \mathbf{I}_c||_1 \tag{16}$$

For geometry consistency, we apply the depth distortion loss and the normal loss:

$$\mathcal{L}_d = \sum_{i,j} w_i w_j |z_i - z_j| \quad \mathcal{L}_n = \sum_{i=1}^{n} w_i (1 - \mathbf{n}_i^\top \mathbf{N}') \tag{17}$$

where $\mathbf{N}'$ is the pseudo normal obtained from the gradient of the depth map, and w_i is the same weigth mentioned in Section 4.1.1. The final loss can then be expressed by

$$\mathcal{L} = \mathcal{L}_c + \lambda_d \mathcal{L}_d + \lambda_n \mathcal{L}_n \tag{18}$$

where λ_d and λ_n are hyperparameters we tuned for different scenes.

4.2 Occlusion and Indirect Illumination Bake

We simply follow the pipeline proposed by GS-IR to bake the occlusion map and the indirect illumination map after the stage of geometry reconstruction. In this stage, the Gaussian primitives are frozen and not be optimized.

4.2.1 Occlusion Volumes. A lot of small occlusion volumes v_i^{occl} are regularly placed in the scene. The occlusion value of each volume is represented by spherical harmonics (SH) and can be formulated as

$$O(\theta, \phi) = \sum_{l=0}^{def} \sum_{m=-l}^{l} \mathbf{f}^o_{i(lm)} Y_{lm}(\theta, \phi) \tag{19}$$

where (θ, ϕ) is the view direction toward the volume, deg is the degree of SH, $\mathbf{f}^o_{i(lm)}$ is the SH coefficients calculated from the occlusion cubemap, which is derived from the depth value projected to the volume and a manually set distance threshold. These occlusion volumes are then cached and can be used for recovering the occlusion map $O(\mathbf{x})$ in the image space given the camera pose.

4.2.2 Indirect Illumination Volumes. Similarly, the indirect illumination volume $v_i^{illu}$is represented by spherical harmonics (SH). The only difference between it and the occlusion volume is that the occlusion cubemap is derived from the depth value while the indirect illumination cubemap is derived from the illumination of the environment map. The indirect illumination volumes are used for recovering the indirect illumination map $I_d^{indir}(\mathbf{x})$. Note that, unlike the fixed occlusion volumes since the geometry has been optimized and would be fixed then, the indirect volumes would be optimized in the next stage since the estimated environment map has not been optimized and would be optimized then.

4.3 PBR Property Decomposition

In this last stage, we unfreeze the Gaussian primitives and optimize the PBR properties. We follow the decomposition rules proposed by GS-IR.

4.3.1 Shading. We apply the Rendering Equation in Eq. (12) for shading. Specifically, the outgoing radiance can be further decomposed into the diffusion component $L_d(\mathbf{x}, \omega_o)$ and the specular component $L_s(\mathbf{x}, \omega_o)$.

For the diffuse component, according to the microfacet BRDF in Eq. (13), $L_d(\mathbf{x}, \omega_o)$ can be formulated as

$$\begin{aligned} L_d(\mathbf{x}, \omega_o) =& (1-m)\frac{\mathbf{a}}{\pi} I_d \\ \approx& (1-m)\frac{\mathbf{a}}{\pi}[(1-O(\mathbf{x}))I_d^{dir} + O(\mathbf{x})I_d^{indir}] \end{aligned} \tag{20}$$

where I_d refers to the integral of diffuse incident radiance and can be further decomposed into I_d^{dir} and I_d^{indir}. I_d^{dir} is calculated from the estimated environment map, $O(x)$ and I_d^{dir} is queried from the occlusion and indirect illumination map mentioned in Section 4.2. We use the split-sum approximation to obtain the close-form solution of the integrals. The metallic m and albedo $\mathbf{a}$ are similar to the color $\mathbf{C}(\mathbf{x})$ that is blended by the primary properties stored in the 2D Gaussian primitive.

For the specular component, $L_s(\mathbf{x}, \omega_o)$ can simply be written as

$$L_s(\mathbf{x}, \omega_o) = RI_s \tag{21}$$

where R and I_s are two integrals decomposed from Eq. (13) and can also be calculated using split-sum approximation. The final outgoing radiance for shading can then be written as

$$L_o(\mathbf{x}, \omega_o) = L_d(\mathbf{x}, \omega_o) + L_s(\mathbf{x}, \omega_o) \tag{22}$$

4.3.2 Loss Function. Since the geometry is fixed at this stage, we do not use loss for geometry consistency in the first stage. Instead, we add regularizations for the material and light. The loss function can be formulated as

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{shading} + \mathcal{L}_{material} + \mathcal{L}_{light} \\ &= ||\mathbf{I}_{gt} + \mathbf{I}_{shading}||_1 + \lambda_{mat} TV_{mat} + \lambda_{light} TV_{light} \end{aligned} \tag{23}$$

where $\mathbf{I}_{shading}$ is obtained from the PBR shading process described in Section 4.3.1, and the latter two terms are TV loss regarding the material and light parameters. We follow the GS-IR to calculate these two losses.

5 EXPERIMENTS

We performed experiments for novel-view synthesis, normal map reconstruction and relighting on TensoIR Synthetic [4] and Mip-NeRF 360 [1] datasets. The TensoIR dataset consists of images of four objects (lego, armadillo, ficus, and hotdog) with their corresponding normal maps, physics-based rendering properties and relighting results. We evaluate the normal-map reconstruction using the Mean Angular Error (MAE) and use Peak-Signal to Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) for novel-view synthesis and albedo quality.

On the other hand, the Mip-NeRF 360 dataset consists of seven scenes (garden, bicycle, room, stump, counter, kitchen, and bonsai) with required data for novel-view synthesis and relighting results, and we compared the algorithms for these both tasks.

5.1 Setup

In our method, we use alpha-weighted mean depth estimation in contrast to median depth estimation used by the original 2D Gaussian splatting method. We empirically observed that this improves the results significantly for normal estimation. The results are summarised below in Table 1. The normals used for rendering are depicted under the column "derived" and the normals estimated from the depth directly are described under "from_depth".

The hyper-parameters for our method are chosen similar to that of GS-IR. For our method we use $\lambda\_normal = 0.05$ for the TensoIR dataset and $\lambda\_normal = 0.03$ for the MipNeRF dataset. The Gaussian densification threshold is set to 5e-5 and 2e-4 for the TensoIR and MipNeRF datasets respectively. We run the method for 30000 iterations and an additional 5000 iterations for PBR properties. For the Mip-NeRF dataset, we run the training for 40000 iterations in total.

5.2 Results

Table 2 summarises the results of GS-IR and our method for normal reconstruction, novel view synthesis, albedo estimation and relighting tasks across the objects in the TensoIR dataset. For the relighting task, we perform relighting under five different lighting conditions - bridge, city, fireplace, forest, and night.

For the novel-view synthesis task, our model performs equivalently to GS-IR. When looked closely, the reconstruction from our has more accurate geometry as seen in images Fig. 1 and Fig. 2. In ficus, GS-IR method does not have smooth depth varation which causes edge artifacts around the leaves. For the lego model, both have similar performance except for minor inconsistencies in the studs in the background.

Our method provides better estimates of normals across most of the scenes. In the Lego Fig. 4, it is seen that 2D Gaussians capture the geometry of the curvature of the track better than GS-IR. However, it does have an over-smoothing problem with studs in the background.

The advantage of depth-distortion technique introduced in 2D-Gaussian splatting is clearly seen in the Ficus Fig. 3. The normals across the leaves smoothly vary based on where the light intersects the flat surface. In contrast, with 3D Gaussians, different Gaussians are used to render different parts of the leaf increasing the number

Table 1: Mean Angular Error (MAE) for TensoIR Synthetic Data

Method	lego		armadillo		ficus	
	derived	from_depth	derived	from_depth	derived	from_depth
Alpha-weighted mean depth estimation	6.853	8.651	2.153	2.639	2.705	4.894
Median depth estimation	8.496	10.577	3.889	4.587	2.726	4.318

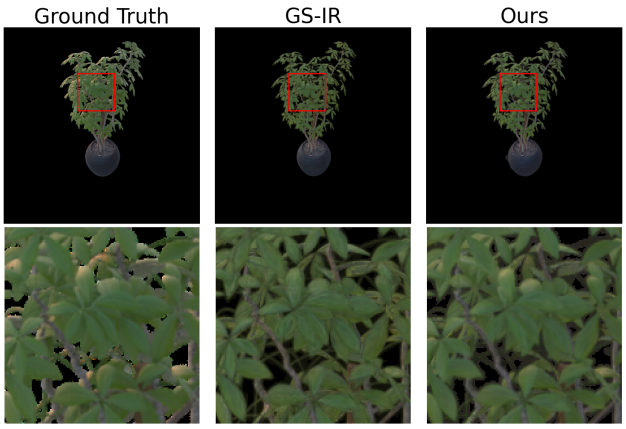

Figure 1: Ficus novel-view synthesis

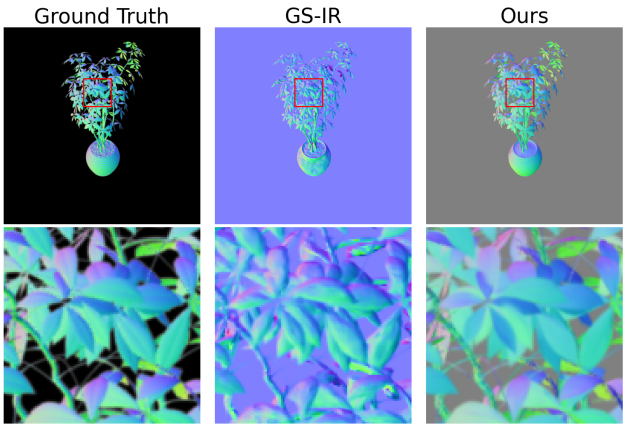

Figure 3: Ficus normals visualisation

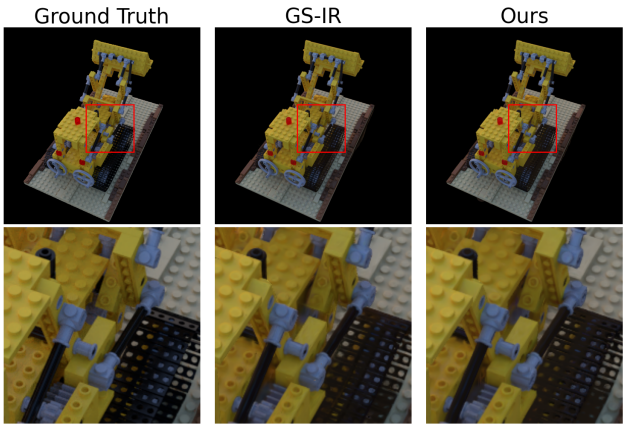

Figure 2: Lego novel-view synthesis

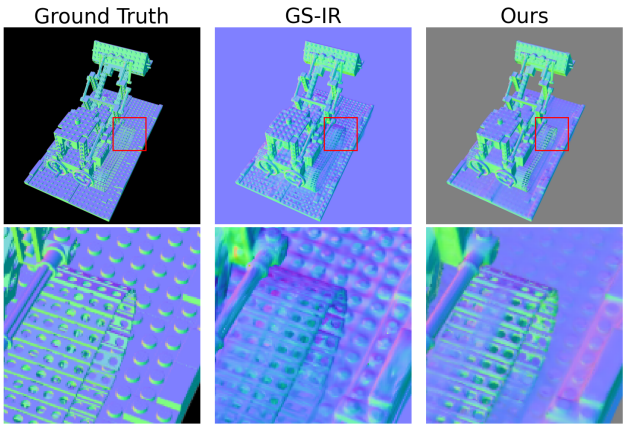

Figure 4: Lego normals visualisation

of Gaussians for the scene significantly. This is also apparent for the Mip-NeRF scenes as seen in figures Fig. 5, Fig. 6 and Fig. 7. The geometry estimated by our method is significantly better than that of GS-IR.

For the counter scene Fig. 5, the normals estimated by our method are smooth whereas the estimation by GS-IR is very noisy. In the stump scene Fig. 6, GS-IR is unable to capture the geometry of the thin stems behind the stump, whereas our algorithm is able to capture these detials. Finally, in the kitchen scene Fig. 7, the texture of the cloth is not captured by GS-IR and the normals for the track of the lego toy do not capture the details. Our method addresses these problems effectively.

As a result much better normal estimation, our method has improved relighting results across all objects in the dataset. The results for this task are visualized in Fig. 8 and Fig. 9 for the TensoIR dataset. The relighting for armadillo shows that the geometry estimated by our method is much less noisy as compared to that of GS-IR.

The Mip-NeRF dataset has ground-truth only for novel-view synthesis, and the results for both the methods are summarised in Table 3. Our method is at par with GS-IR across most of the scenes with slightly higher PSNR and slightly lower SSIM and LPIPS values. The results are visualised in Fig. 10 and Fig. 11, and there are no differences between the images to the naked eye.

Our relighting results in Mip-NeRF dataset are much better than GS-IR. For example, in the bicycle image shown in Fig. 12, the colors

Table 2: Performance Metrics for TensoIR dataset

Scene	Method	Normal	Novel View Synthesis			Albedo			Relight		
		MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Lego	GS-IR	8.355	**33.753**	0.954	0.043	20.447	**0.836**	**0.167**	22.767	0.833	0.120
	2D-GS-IR (ours)	**6.853**	33.597	**0.957**	**0.041**	**19.613**	0.831	0.169	**23.132**	**0.855**	**0.109**
Hotdog	GS-IR	**4.767**	**35.623**	0.972	**0.046**	**23.968**	**0.915**	**0.121**	**21.566**	0.887	0.140
	2D-GS-IR (ours)	5.080	34.893	**0.973**	0.047	22.934	0.908	0.124	21.544	**0.901**	**0.128**
Armadillo	GS-IR	3.114	**39.151**	**0.975**	0.038	42.210	0.972	0.07	27.757	0.909	0.082
	2D-GS-IR (ours)	**2.153**	37.843	0.973	**0.037**	**43.174**	**0.975**	**0.066**	**29.205**	**0.927**	**0.072**
Ficus	GS-IR	5.020	**33.507**	**0.961**	**0.037**	33.163	0.963	0.044	23.680	0.867	0.095
	2D-GS-IR (ours)	**2.705**	31.138	0.949	0.051	**33.733**	**0.966**	**0.040**	**24.387**	**0.879**	**0.092**

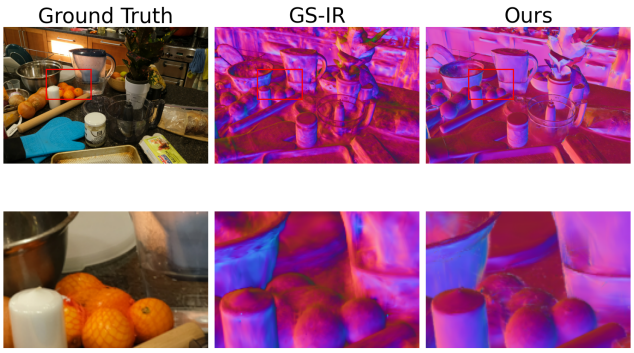

Figure 5: Counter normals

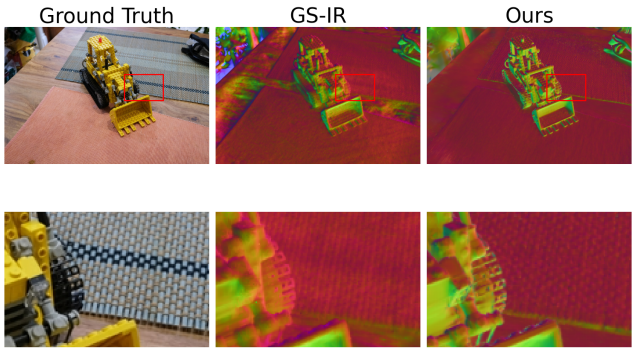

Figure 7: Kitchen normals

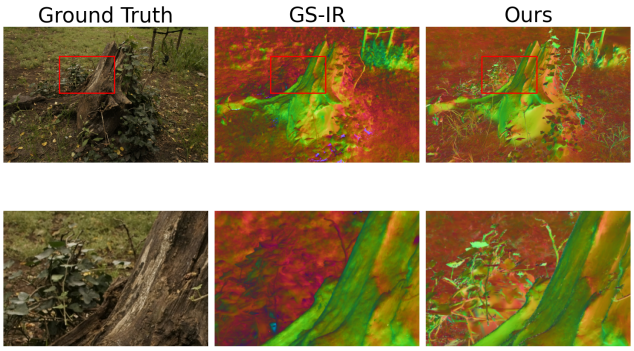

Figure 6: Stump normals

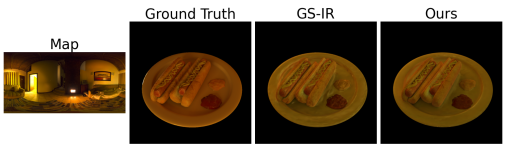

Figure 8: Hotdog Relighting with fireplace

Figure 9: Armadillo Relighting with city

are over-saturated in GS-IR and the detail is lost. This can be seen from the bushes in the background, and a similar result is seen for the garden relighting as well. The normals of GS-IR are noisy in the background, causing artifacts between the leaves on relighting.

Furthermore, the albedo estimation in our method is at par with that of GS-IR as well. The results for lego and hotdog are visualized in Fig. 14 and Fig. 15 respectively. Other PBR properties such as roughness and metallic are visualized in Fig. 16 and Fig. 17. These results show that indirect illumination using baking and physics-based rendering equations have been correctly implemented in our method.

Table 3: Performance Metrics for novel view synthesis on Mip-NeRF 360 dataset

Metric	Method	Garden	Bicycle	Stump	Kitchen	Bonsai	Counter	Room
PSNR	GS-IR	**27.307**	23.753	**26.934**	28.476	28.446	**26.416**	27.246
	2D-GS-IR (ours)	26.675	**24.129**	26.540	**28.478**	**28.665**	26.412	**29.873**
SSIM	GS-IR	**0.849**	0.699	**0.802**	**0.901**	0.906	**0.857**	0.897
	2D-GS-IR (ours)	0.835	**0.741**	0.780	0.888	0.906	0.847	**0.899**
LPIPS	GS-IR	**0.137**	0.262	**0.211**	**0.116**	**0.150**	**0.170**	**0.181**
	2D-GS-IR (ours)	0.161	**0.261**	0.246	0.132	0.168	0.188	0.187

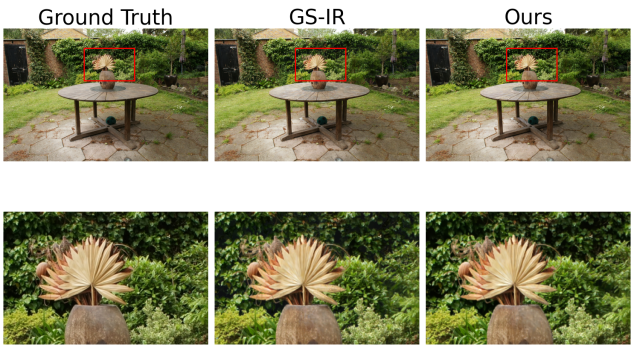

Figure 10: Garden novel-view synthesis

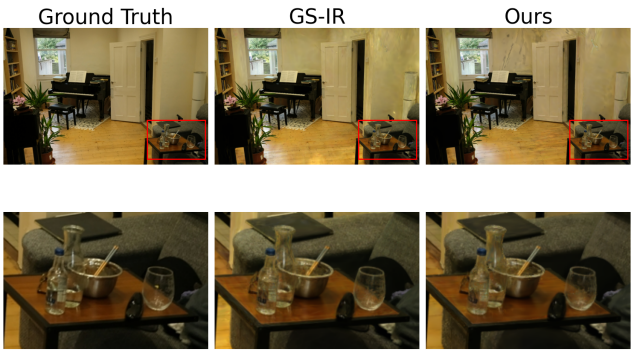

Figure 11: Room novel-view synthesis

Figure 12: Bicycle relighting with snow

Figure 13: Garden relighting with courtyard

6 LIMITATIONS

As with any other Gaussian Splatting based inverse rendering approach, our method is not suitable to model translucent/transparent objects due to the full-opacity surfaces assumption. The baking approach used in GS-IR is implemented via spherical harmonics and it cannot describe high-dimensional indirect illumination properties for specular surfaces. It can be addressed by future works utilising techniques such as screen-space global illumination (SSGI).

Furthermore, the geometry obtained by 2DGS is often over-smooth and struggles with geometry-rich regions. The current densification strategy prefers texture-rich regions over geometry-rich regions.

7 CONCLUSION

Using 2D Gaussian Splatting for scene representation indeed provides better geometry, and consequently a better normal map for inverse rendering tasks. This is quite useful in downstream applications such as relighting and augmented-reality scene construction. This framework provides the advantage of better reconstruction all while having a significantly lower training time. Through our

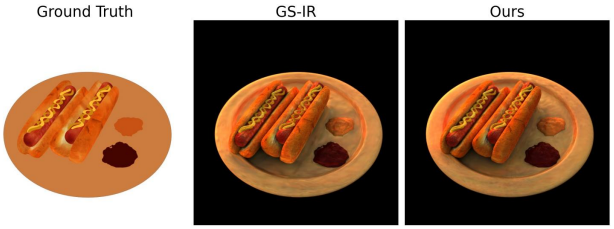

Figure 14: Albedo estimation for hotdog

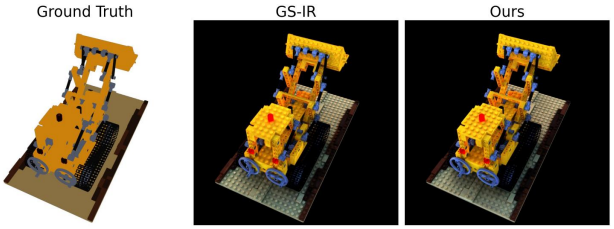

Figure 15: Albedo estimation for lego

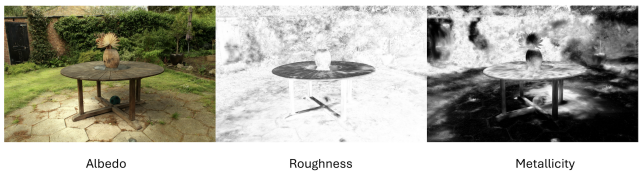

Figure 16: Garden PBR estimates

Figure 17: Bonsai PBR estimates

experiments, we have shown that this direction is indeed worth pursuing, and further work with 2D Gaussian splatting will be very useful for these tasks.

REFERENCES

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5470–5479.

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*. Springer, 333–350.

[3] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2d gaussian splatting for geometrically accurate radiance fields. *arXiv preprint arXiv:2403.17888* (2024).

[4] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 165–174.

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

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

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

[8] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.

[9] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. 2021. 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*. 5453–5462.

A STATEMENT OF CONTRIBUTION

Chuhao Chen: Implemented the modified 2DGS rasterizer containing PBR properties. Conducted part of the experiments. Wrote the Section 3 and Section 4 of the report.

Sudhansh: Set up the environment for running the scripts, experimented with different normal loss functions, compiled and contrasted the results, and wrote the abstract, Section 5, Section 6 and Section 7 of the report.

Ziyang Fu: Proposed the idea of using 2DGS in GSIR. Implemented baking and the PBR relighting using 2DGS in GSIR pipeline. Wrote Section 1, Section 2 of the report.

Zejia Wu: Conducted all the experiments and obtianed all the comprehensive results in the report, implemented edge filtering in depth maps to prevent artifacts near the edges of objects.