3iGS: Factorised Tensorial Illumination for 3D Gaussian Splatting

Zhe Jun Tang[1] and Tat-Jen Cham[2]

[1] S-Lab, Nanyang Technological University
[2] College of Computing & Data Science, Nanyang Technological University
{zhejun001} at {e.ntu.edu.sg}

Abstract. The use of 3D Gaussians as representation of radiance fields has enabled high quality novel view synthesis at real-time rendering speed. However, the choice of optimising the outgoing radiance of each Gaussian independently as spherical harmonics results in unsatisfactory view dependent effects. In response to these limitations, our work, Factorised Tensorial Illumination for 3D Gaussian Splatting, or *3iGS*, improves upon 3D Gaussian Splatting (3DGS) rendering quality. Instead of optimising a single outgoing radiance parameter, 3iGS enhances 3DGS view-dependent effects by expressing the outgoing radiance as a function of a local illumination field and Bidirectional Reflectance Distribution Function (BRDF) features. We optimise a continuous incident illumination field through a Tensorial Factorisation representation, while separately fine-tuning the BRDF features of each 3D Gaussian relative to this illumination field. Our methodology significantly enhances the rendering quality of specular view-dependent effects of 3DGS, while maintaining rapid training and rendering speeds.

Keywords: Gaussian Splatting · Neural Radiance Field · Novel View Synthesis

1 Introduction

3D Gaussian Splatting (3DGS) has emerged as the standard method for representing 3D objects and scenes, trained from images, to render photorealistic novel views. Unlike the other popular method of Neural Radiance Field (NeRF) [22], which models a scene as an implicit continuous function, 3DGS represents surfaces with independent 3D Gaussians of different opacities, anisotropic covariances, and spherical harmonic coefficients. To render a pixel's colour, a fast, tile-based rasteriser performs alpha blending of anisotropic Gaussian splats, sorted in accordance with the visibility order.

Although 3DGS shows promising performance in synthesising novel views of a scene at real-time rendering speeds, its renderings fall short in more challenging scenarios that involve complex, view-dependent surface effects. When observing images with reflective and specular surfaces, the changes in surface colour across viewing angles remain consistent, rather than exhibiting the complex variations

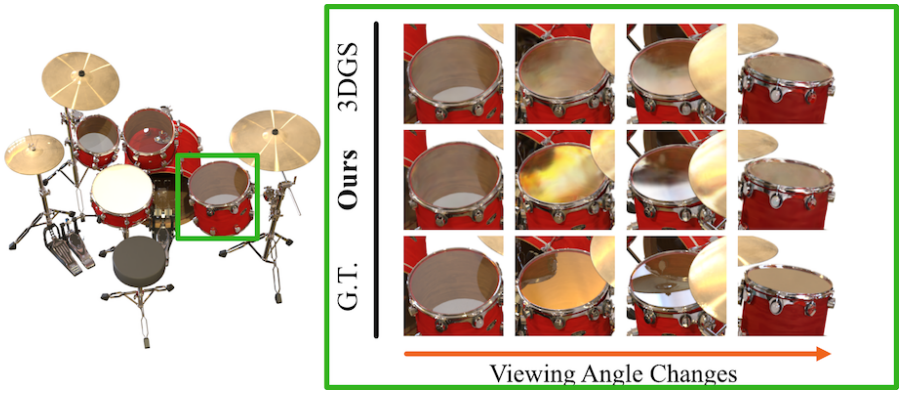

Fig. 1: We present test renderings from the "Drums" scene within the blender dataset [22], comparing our technique against Gaussian Splatting (3DGS) [17] and the ground truth (G.T). As the perspective shifts around the scene, the colour of the Floor Tom's top changes from translucent to reflective, showcasing intricate effects that depend on the viewpoint. These effects result from the specular reflection of incoming light and the reflections within the scene from elements like the Cymbals. Contrary to 3DGS, which struggles to capture these complex variations in light reflection, our method, 3iGS, aligns more accurately with the ground truth.

in reflections observed in the dataset shown in Fig. 1. A logical solution is to adopt the strategy of Physically Based Rendering (PBR), which involves explicitly modeling the surface characteristics and performing ray marching from surfaces to calculate illumination effects. As part of the process, the Bidirectional Reflectance Distribution Function (BRDF) of surfaces are predicted and a shading function is applied to simulate view-dependent effects [7,10]. Nonetheless, accurately determining these physical properties is an ill-posed challenge, making it difficult to infer and model all the intricate rendering effects correctly.

In this paper, we draw inspiration from graphics engines that utilise illumination volumes or light probes that summarise illumination information directed towards a surface. These methods compute illumination either directly from the local illumination volume surrounding the surface [12] or from the nearest light probes [28], rather then sampling numerous outward rays from the surface's upper hemisphere. Such approaches allow fast rendering speed at run time, as illumination information is pre-calculated and stored in the volumes or light probes.

Our work, named Factorised Tensorial Illumination for Gaussian Splatting (3iGS), enhances 3DGS rendering quality. We introduce a continuous local illumination field of 3D Gaussians represented by compact factorised tensors for fast evaluation. The means of the 3D Gaussians serve as the input to these factorised tensors to calculate illumination features. Subsequently, each 3D Gaussian is refined through an optimisation of its mean, opacity, anisotropic covariance, diffused colour, and BRDF features. A neural renderer then maps the incident

illumination neural field, Gaussian BRDF attributes, and viewing angle to the Gaussian's specular colour. Overall, our approach represents a Gaussian's outgoing radiance as *a function of both a continuous local illumination field and the individual Gaussian's BRDF attributes relative to it*. This is opposed to the conventional optimisation of the 3D Gaussians' outgoing radiance in isolation, without accounting for the effects of adjacent Gaussians or scene lighting conditions.

3iGS significantly enhances the accuracy of 3DGS, offering clear advantages in scenes with reflective surfaces where surface colours change dramatically across viewing angles as shown in Fig. 1. In synthetic datasets, such as the NeRF Blender dataset and the Shiny Blender dataset, 3iGS surpasses 3DGS both quantitatively and qualitatively. Similarly, 3iGS demonstrates superior performance over 3DGS in real-world scenarios on the Tanks and Temples dataset. In summary our technical contributions are:

1. a method to optimise the outgoing radiance as an incident continuous illumination field and Gaussian BRDF features with a neural renderer;
2. an approach to model a continuous illumination field with Tensorial Factorisation for compactness and fast evaluation; and
3. superior performance in rendering quality over baseline 3D Gaussian Splatting while maintaining real time performance.

2 Related Work

Our work falls into the category of learning scene representation from multi-view input images. Here we review prior work on NeRF-based representations and Gaussian splatting. We also discuss other relevant topics pertaining to inverse rendering which aims to recover scene geometry, material properties, and scene lighting conditions in Sec. 2.1.

Scene Representations for View Synthesis - One of the pioneering neural rendering techniques called Neural Radiance Fields (NeRF) [22] has achieved remarkable results in novel view synthesis from multi-view images. By sampling points along rays traced from the camera into the scene, NeRF reconstructs a scene as a continuous field of outgoing radiance. The technique employs volumetric rendering to determine the colour of each pixel. This method has inspired numerous developments of other scene representations [1, 2, 22, 31, 32]. However, the vanilla NeRF, which encodes the entire scene representation into a set of MLPs, requires multiple queries of points along rays during training and inference. This massively slows down the speed required for real time rendering. To address this, other neural scene representation techniques apply hash encoding [19, 23], triplanes or factorised tensors [9, 14], and gridding [3, 11, 27] to accelerate training and inference speeds.

Tensorial Factorisation - In TensoRF [9], a feature grid can be represented as a 4D tensor of which the first 3 represents the XYZ spatial grid and the last represents the feature channel dimension. To model a radiance field with grid

representation, [9] propose an extension of CANDECOMP/PARAFAC (CP)-Decomposition [8] to Vector-Matrix (VM) decomposition:

$$\begin{aligned}\mathcal{G}_c &= \sum_{r=1}^{R_c} \mathbf{v}^{\mathbf{X}}_{\mathbf{c,r}} \circ \mathbf{M}^{\mathbf{YZ}}_{\mathbf{c,r}} \circ \mathbf{b}_{\mathbf{3r-2}} + \mathbf{v}^{\mathbf{Y}}_{\mathbf{c,r}} \circ \mathbf{M}^{\mathbf{XZ}}_{\mathbf{c,r}} \circ \mathbf{b}_{\mathbf{3r-1}} + \mathbf{v}^{\mathbf{Z}}_{\mathbf{c,r}} \circ \mathbf{M}^{\mathbf{XY}}_{\mathbf{c,r}} \circ \mathbf{b}_{\mathbf{3r}} \\ &= \sum_{r=1}^{R_c} \mathbf{A}^{X}_{C,r} \circ \mathbf{b}_{3r-2} + \mathbf{A}^{Y}_{C,r} \circ \mathbf{b}_{3r-1} + \mathbf{A}^{Z}_{C,r} \circ \mathbf{b}_{3r}\end{aligned} \tag{1}$$

In Eq. (1), the inputs $\mathbf{v}$ and $\mathbf{M}$ corresponds to XYZ-mode vector and matrix factorisation and $\mathbf{b}$ denotes the appearance feature mode vectors. Separately, $\mathcal{G}_c$ and R_C refers to the outgoing radiance and the colour feature channels.

Gaussian Splatting - As opposed to ray marching, 3D Gaussian Splatting is a recent method for rendering scenes via rasterisation. To begin, Gaussians are fitted on a point cloud that are either initialised as a set of random points or bootstrapped with a sparse point cloud produced during the SfM process for free [17]. The Gaussians of the point cloud are defined by a function:

$$g(\mathbf{x}|\mu, \mathbf{\Sigma}) = e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)} \tag{2}$$

where each point $\mathbf{x}$ is centered at mean $\mu \in \mathbb{R}^3$ with an anisotropic covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{3x3}$. The mean of a Gaussian is parameterised by the coordinates $\mu = (\mu_x, \mu_y, \mu_z)$ that is scaled by the full 3D covariance matrix $\mathbf{\Sigma}$. As discussed in [17], these Gaussians have no physical meanings, given the difficulty of constraining $\mathbf{\Sigma}$ to a valid semi-positive definite matrix during the optimisation process. Instead, to derive $\mathbf{\Sigma}$, a scaling matrix S and a rotation matrix R is learned during the optimisation process to scale the Gaussians:

$$\mathbf{\Sigma} = \mathbf{RSS^TR^T} \tag{3}$$

With a viewing transformation $\mathbf{W}$ and an affine approximation of the projective transformation $\mathbf{J}$, the covariance matrix is then expressed in camera coordinates as:

$$\mathbf{\Sigma}^{'} = \mathbf{JW\Sigma W^TJ^T} \tag{4}$$

Furthermore, each Gaussian is coloured via a set of Spherical Harmonics (SH) coefficients that represent the view dependent colour c_i, also known as radiance field, multiplied by its opacity α. To colour a pixel u as $\hat{C}$, alpha blending of N ordered Gaussians is applied:

$$\hat{\mathbf{C}} = \sum_{i \in N} T_i g_i(\mathbf{u}|\mu^{'}, \mathbf{\Sigma}^{'}) \alpha_i \mathbf{c}_i, \quad T_i = \prod_{j=1}^{i-1} (1 - g_i(\mathbf{u}|\mu', \mathbf{\Sigma}')\alpha_i) \tag{5}$$

2.1 Preliminaries

As discussed, the direct optimisation of spherical harmonics to describe the outgoing radiance in individual Gaussians in 3DGS results in poor view-dependent

effects. A crucial reason is that these Gaussians do not fully model scene properties [13] and thus fail to capture the specular effects which changes drastically across viewing angles.

Therefore to account for the specular highlights, it is beneficial to model the underlying properties such as the BRDF and illumination effects of the scene. In conventional computer graphics, a rendering equation is commonly applied to simulate effects of specular and diffused shading [15]. For instance, rendering Eq. (6) describes an outgoing radiance of a surface point:

$$L_o(\mathbf{x}, \mathbf{v}) = \int_{\Omega} L_i(\mathbf{x}, \mathbf{l}) f_r(\mathbf{l}, \mathbf{v})(\mathbf{l} \cdot \mathbf{n}) d\mathbf{l}, \tag{6}$$

The radiance L_o emitted from a surface point $\mathbf{x}$, when observed from a viewing direction $\mathbf{v}$, is defined in Eq. (6). An integral is applied to accumulate the contribution of incident light at an incident angle $\mathbf{l}$ across the upper hemisphere $\mathbf{\Omega}$ of $\mathbf{x}$. The function f_r denotes the Bidirectional Radiance Distribution Function (BRDF), describing the reflection characteristics of incident radiance at $\mathbf{x}$ viewed in direction $\mathbf{v}$. Lastly the inclusion of the cosine law with the normal vector $\mathbf{n}$ ensures the energy conservation.

From a signal processing perspective, an alternative to Eq. (6) is expressed more generally in terms of spherical harmonic convolution [16, 21]:

$$B_{lm} = \Lambda_l \rho_l L_{lm} \tag{7}$$

In Eq. (7), B_{lm} defines the outgoing reflected light as the product of BRDF filter ρ_l, spherical harmonic coefficients of lighting signal L_{lm}, and the normalisation constant Λ_l.

Some studies [13, 25] enhance 3DGS by expressing BRDF f_r as a Cook-Torrance microfacet model [10] or the GGX Trowbridge-Reitz model [7]. In these approaches, physical attributes, including roughness r, albedo a, metallicity m, and the normal vector $\mathbf{n}$ are predicted and used in Eq. (6). Although these modifications marginally improve rendering quality metrics, they fail to accurately produce high-quality, view-dependent effects. This shortfall primarily stems from relying on estimated parameters for physical rendering within a simplified rendering equation [20]. Furthermore, these parameters are inherently challenging to be estimated accurately, due to the ill-posed nature of inverse rendering from multi-view images. Although numerous works [4–6, 14, 20, 26] also achieved success by exploring a neural representation of the rendering equation, these works either require prior information, such as known lighting conditions or a pre-trained model on a realistic dataset with known BRDF parameters. Furthermore these techniques are experimented with ray tracing based methods like NeRF. A work closest to ours in the area of rasterisation and Gaussian Splatting manner, is GaussianShader [13] which we compare against in Sec. 5.2.

3 Method

Instead of predicting the physical BRDF properties of materials in the scene, our goal is to express the outgoing radiance of a Gaussian as a more general

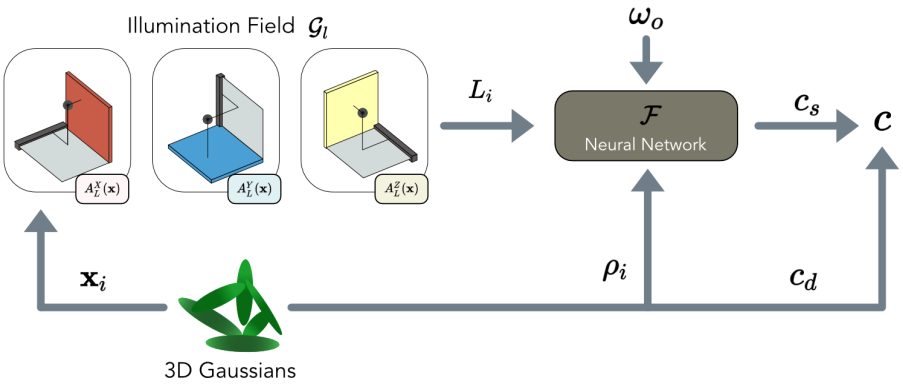

Fig. 2: A visualisation of 3iGS pipeline to render a single Gaussian's colour. We interpolate an incident illumination L_i from the factorised tensorial illumination field $\mathcal{G}_l$ using a Gaussian mean $\boldsymbol{x}_i$ as input. A neural network $\mathcal{F}$ maps the illumination field L_i, the Gaussian BRDF features ρ_i, and the viewing direction ω_o to Gaussian's specular colour c_s. Following, the diffused colour c_d and specular colour c_s are added linearly to produce the final outgoing radiance field c.

expression of BRDF, and the incoming illumination as neural features. This idea is based on a generalized version of Eq. (7), where BRDF features modify an incoming illumination field, without the need for decomposing down to intrinsic material properties [21].

Specifically for each 3D Gaussian in the scene, the outgoing radiance field is formed by:

$$\mathbf{c}(\omega_o) = \mathbf{c_d} + \mathbf{c_s}(\omega_o) \tag{8}$$

For viewing angle ω_o, a Gaussian is coloured by its constant diffused colour c_d and a view dependent specular colour c_s. At each Gaussian i, a small neural network $\mathcal{F}$ maps the Gaussian BRDF features ρ_i and the incoming illumination L_i to its specular colour viewed at an angle ω_o :

$$\mathcal{F} : \{\rho_i, L_i, \omega_o\} \mapsto \mathbf{c_s} \tag{9}$$

3.1 Illumination Grid by Tensorial Factorisation

Our work is largely inspired by conventional computer graphics engines for fast rendering of scene and objects in video games. The fundamental rendering equation highlights the role of multi-bounce lighting in achieving indirect illumination, wherein light bounces off one surface to illuminate another. However, the process of ray tracing from each Gaussian surface into the scene is notably resource-intensive, undermining the goal of quick rendering in 3D graphics systems. To facilitate real-time rendering, one strategy involves the use of baking techniques that employ irradiance volumes [12]. This method segments a scene into distinct volumes and pre-calculates irradiance data offline. An alternative

strategy places light probes [28,29] throughout the scene to gather lighting information at specific spatial locations. When rendering the colour of a surface, the system quickly interpolates lighting information from the nearest light probes, ensuring swift rendering times.

To maintain the fast rendering speed of 3DGS, our work describes a methodology of learning the illumination features of a Gaussian with a continuous grid based illumination field as:

$$\mathcal{G}_l : \{\mathbf{x_i}\} \mapsto L_i \tag{10}$$

Given a Gaussian's mean coordinate $\mathbf{x_i}$, we seek to compute an illumination field L_i by interpolating from learnable grid representation. The illumination tensors $\mathcal{G}_l$ is formulated similar to TensoRF [9] by a vector-matrix spatial factorisation as follows:

$$\mathcal{G}_l = \sum_{r=1}^{R_L} \mathbf{A}_{L,r}^{X} \circ \mathbf{b}_{3r-2} + \mathbf{A}_{L,r}^{Y} \circ \mathbf{b}_{3r-1} + \mathbf{A}_{L,r}^{Z} \circ \mathbf{b}_{3r} \tag{11}$$

In Eq. (11), R_L represents the feature channels of the illumination components, $\boldsymbol{A}$ as feature tensors and $\mathbf{b}$ as feature vectors. The illumination feature grid is jointly learned end to end in the optimisation process together with each Gaussian in the scene. Unlike 3DGS, where each Gaussian is optimised independently, the illumination field is modelled as a continuous grid function. A Gaussian mean serves as the input to query from the factorised tensor grid via interpolation. The inclusion of this continuous incoming illumination field directed at each Gaussian is the core component of producing accurate view-dependent effects, as we show in the ablation study of Sec. 5.4. Furthermore, by formulating this field as factorised tensors, it allows the network to achieve fast rendering speed. Our illumination field is coarse, using 87.5% less voxels compared to TensoRF on synthetic datasets. This compact representation is also low in memory footprint compared to the number of optimised Gaussians, which is often a magnitude order or more higher. We refer readers to [9], which provides a comprehensive overview to describe how the tensors are factorised and interpolated.

3.2 3D Gaussian Features

In 3DGS [17], Gaussians are optimised with a set of parameters: 3D positions, opacity α, anisotropic covariance, and spherical harmonics coefficients. In our work, instead of optimising spherical harmonics as an outgoing radiance, 3iGS characterises the Gaussians with a diffused colour and learnable BRDF features. Unlike [13, 25], we do not strictly enforce physically interpretable properties commonly used in shading techniques. Aforementioned, these techniques are often simplified, too ill-posed to be decomposed individually, and insufficient to encompass all complex rendering effects [20]. Rather, we loosely follow Eq. (7) and treat BRDF feature components as a set of weights that alter the incoming illumination field. Given a continuous illumination field obtained from Eq. (11), a Gaussian's BRDF is conditionally optimised against it. This is in contrast

to 3DGS where the Gaussians' outgoing radiance are individually optimised without modelling the interdependencies that should arise from a shared scene illumination, resulting in detrimental view-dependent effects.

3.3 Shading Gaussians

Following Eq. (9), we shade each Gaussian by mapping its viewing directions encoded with Integrated Directional Encoding (IDE) [30], Gaussian features (obtained in Sec. 3.2), and its illumination field, to the specular colour output. We linearly add the diffused and specular colours to create its radiance field as per Eq. (8). To render the final scene, we follow the rasterisation pipeline proposed in the original 3DGS work.

4 Optimisation

In the previous Sec. 3, we described the necessary components to model a scene with Gaussians and render it via rasterisation. To improve the stability of training and to enhance the final rendering quality, we first train the model with the diffused colour in the first 3,000 iterations. Following, specular colours are added to the Gaussians as in Eq. (8).

While training the tensorial illumination grid, an initial boundary which encapsulate the scene bounding box is defined. Midway through training, we shrink the illumination grid to fit the Gaussians and resample the grid with the same number of voxels. We adopt the same adaptive control of Gaussians of 3DGS [17] to limit the number of Gaussians and the units per volume. We propose to train our model with the same loss function as 3DGS for a fair evaluation:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} \tag{12}$$

where we combined the $\mathcal{L}_1$ term with a D-SSIM term with λ set to 0.2.

5 Experiments and Results

5.1 Datasets

Synthetic scenes - We show experimental results of 3iGS based on the Blender dataset released in [22]. This dataset contains challenging scenes of complex geometries with realistic non-Lambertian materials. Similarly, we evaluate our model on the Shiny Blender dataset presented in [30]. Unlike the Blender dataset, Shiny Blender contains a singular object with simple geometries in each scene with more glossy effects.

Real world complex scenes - To prove the effectiveness of our model in real world scenes, we evaluate our renderings on the Tanks and Temples dataset [18]. This dataset is obtained from video sequences of real world objects and environment.

5.2 Comparisons

To evaluate our model, we compared against methods that apply both ray-tracing methods like NeRF, or rasterisation methods with Gaussian Splatting. Out of all prior work, 3DGS and GaussianShader is the closest work which offers real time inference speed which we will mainly compare against. On comparing the qualitative result figures, we re-ran the experiments of 3DGS [17] and GaussianShader [13] using their original repository code under settings specified by the authors. **Ray-Tracing Methods** such as [20, 22, 30] represent a scene as a radiance field using MLPs. By performing multiple samplings on rays marched from the camera into the scene, the sampled points are queried with MLP to obtain the opacity and radiance values. Volume rendering is performed to obtain the final pixel colour. **Rasterisation Methods** such as Gaussian Splatting (3DGS) [17] and GaussianShader [13] apply a rasterisation pipeline as opposed to ray tracing methods. These models represents a scene as Gaussians with radiance properties based on Spherical Harmonics. In, [13], 3DGS is extended by modelling a scene with additional material characteristics and a shading function is applied, as opposed to ours which uses an MLP as neural renderer. Furthermore, [13] shades Gaussians with a global differentiable environment light stored in cube maps, and optimises independent Gaussians with spherical harmonic-based color for unaccounted illumination. In our work, we represent incident illumination *locally* with grid-based tensors and optimise Gaussian BRDF features relative to this field.

For a fair comparison, 3iGS is trained with the same loss function as 3DGS as described in Sec. 4 and the same number of iterations of 30,000 steps. We repurposed the 16×3 SH coefficients in 3DGS as BRDF feature channels and added 4 additional parameters of base colour and roughness for IDE view-directional encoding. The tensorial illumination field is set at a coarse resolution size of 150^3 voxels.

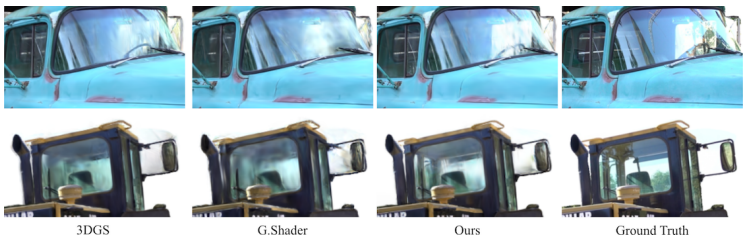

Fig. 3: Comparisons of test-set views of real world scenes. 3iGS enhances 3DGS renderings by producing clearer view dependent effects as shown.

Table 1: Our approach demonstrates superior quantitative performance over current methods when tested on Synthetic Datasets. Specifically, within the NeRF Synthetic dataset, our method surpasses all competitors across various image quality assessments (PSNR/SSIM/LPIPS). In the context of the Shiny Blender dataset, 3iGS matches the performance of existing rasterization techniques in terms of PSNR and SSIM but surpasses them in LPIPS for the majority of scenes. We encourage readers to examine the accompanying figure showcasing renderings of the Shiny Blender scene, where our method attains enhanced qualitative outcomes. Best results, benchmarked across *real time rendering methods*, are in bold.

NeRF Synthetic [22]									
	Chair	Drums	Lego	Mic	Mats.	Ship	Hotdog	Ficus	Avg.
PSNR↑									
NeRF [22]	33.00	25.01	32.54	32.91	29.62	28.65	36.18	30.13	31.01
Ref-NeRF [30]	33.98	25.43	35.10	33.65	27.10	29.24	37.04	28.74	31.29
ENVIDR [20]	31.22	22.99	29.55	32.17	29.52	21.57	31.44	26.60	28.13
3DGS [17]	35.82	26.17	35.69	35.34	30.00	30.87	37.67	34.83	33.30
G.Shader [13]	35.83	26.36	35.87	35.23	**30.07**	30.82	37.85	34.97	33.38
G.Shader(reproduced) [13]	33.70	25.50	32.99	34.07	28.87	28.37	35.29	33.05	31.48
Ours	**35.90**	**26.75**	**35.94**	**36.01**	30.00	**31.12**	**37.98**	**35.40**	**33.64**
SSIM↑									
NeRF [22]	0.967	0.925	0.961	0.980	0.949	0.856	0.974	0.964	0.947
Ref-NeRF [30]	0.974	0.929	0.975	0.983	0.921	0.864	0.979	0.954	0.947
ENVIDR [20]	0.976	0.930	0.961	0.984	0.968	0.855	0.963	0.987	0.956
3DGS [17]	0.987	0.954	0.983	0.991	0.960	0.907	0.985	0.987	0.969
G.Shader [13]	0.987	0.949	0.983	0.991	0.960	0.905	0.985	0.985	0.968
G.Shader(reproduced) [13]	0.980	0.945	0.972	0.989	0.951	0.881	0.980	0.982	0.960
Ours	**0.987**	**0.955**	**0.983**	**0.992**	**0.961**	**0.908**	**0.986**	**0.989**	**0.970**
LPIPS↓									
NeRF [22]	0.046	0.091	0.050	0.028	0.063	0.206	0.121	0.044	0.081
Ref-NeRF [30]	0.029	0.073	0.025	0.018	0.078	0.158	0.028	0.056	0.058
ENVIDR [20]	0.031	0.080	0.054	0.021	0.045	0.228	0.072	0.010	0.067
3DGS [17]	0.012	0.037	0.016	0.006	0.034	0.106	0.020	0.012	0.030
G.Shader [13]	0.012	0.040	**0.014**	0.006	**0.033**	**0.098**	0.019	0.013	0.029
G.Shader(reproduced) [13]	0.019	0.045	0.026	0.009	0.046	0.148	0.029	0.017	0.042
Ours	**0.012**	**0.036**	0.015	**0.005**	0.034	0.102	**0.019**	**0.010**	**0.029**

Shiny Blender [30]							
	Car	Ball	Helmet	Teapot	Toaster	Coffee	Avg.
PSNR↑							
NVDiffRec [24]	27.98	21.77	26.97	40.44	24.31	30.74	28.70
Ref-NeRF [30]	30.41	29.14	29.92	45.19	25.29	33.99	32.32
ENVIDR [20]	28.46	38.89	32.73	41.59	26.11	29.48	32.88
3DGS [17]	27.24	27.69	28.32	45.68	20.99	32.32	30.37
G.Shader [13]	**27.90**	**30.98**	28.32	45.86	**26.21**	32.39	**31.94**
G.Shader(reproduced) [13]	27.51	29.02	**28.73**	43.05	22.86	31.34	30.41
Ours	27.51	27.64	28.21	**46.04**	22.69	**32.58**	30.77
SSIM↑							
NVDiffRec [24]	0.963	0.858	0.951	0.996	0.928	0.973	0.945
Ref-NeRF [30]	0.949	0.956	0.955	0.995	0.910	0.972	0.956
ENVIDR [20]	0.961	0.991	0.980	0.996	0.939	0.949	0.969
3DGS [17]	0.930	0.937	0.951	0.996	0.895	0.971	0.947
G.Shader [13]	**0.931**	**0.965**	0.950	0.996	**0.929**	0.971	**0.957**
G.Shader(reproduced) [13]	0.930	0.954	**0.955**	0.995	0.900	0.969	0.950
Ours	0.930	0.938	0.951	**0.997**	0.908	**0.973**	0.949
LPIPS↓							
NVDiffRec [24]	0.045	0.297	0.118	0.011	0.169	0.076	0.119
Ref-NeRF [30]	0.051	0.307	0.087	0.013	0.118	0.082	0.109
ENVIDR [20]	0.049	0.067	0.051	0.011	0.116	0.139	0.072
3DGS [17]	0.047	0.161	0.079	0.007	0.126	0.078	0.083
G.Shader [13]	0.045	**0.121**	0.076	0.007	**0.079**	0.078	**0.068**
G.Shader(reproduced) [13]	0.045	0.148	0.088	0.012	0.111	0.085	0.099
Ours	**0.045**	0.156	**0.073**	**0.006**	0.099	**0.076**	0.075

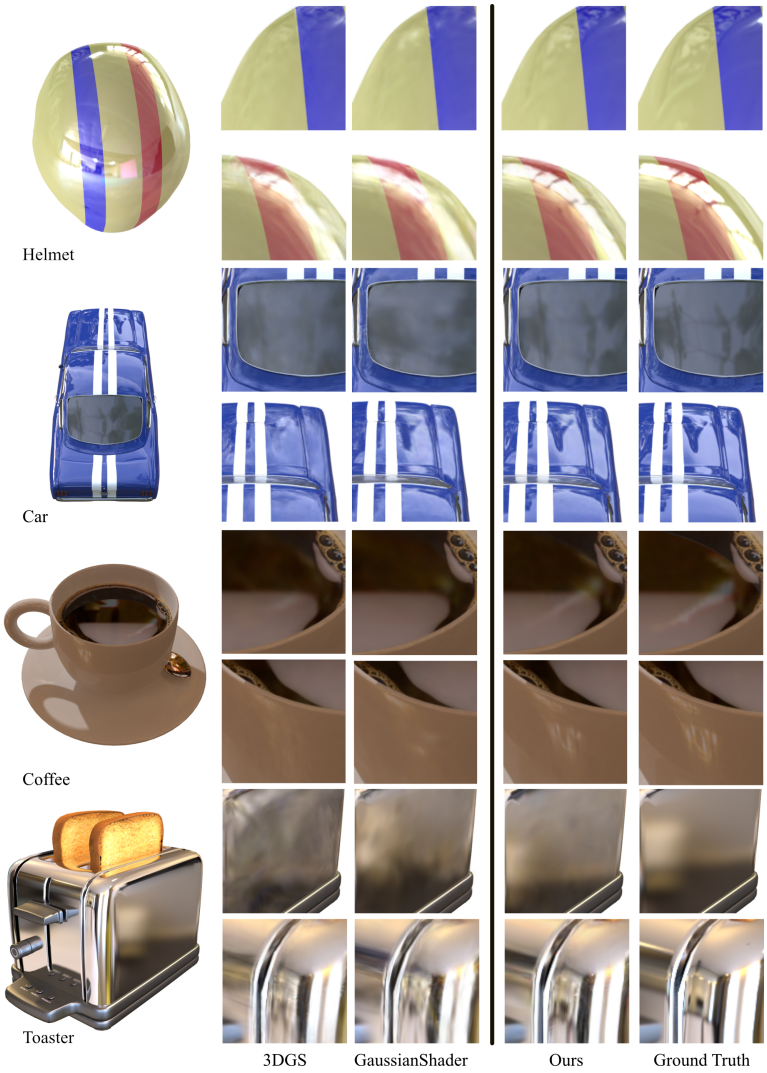

Fig. 4: In evaluating test-set views from the Shiny Blender dataset, we compared the performance of 3DGS [17], GaussianShader [13], and our work 3iGS. The standard 3DGS method generally yields the least satisfactory renderings, with images often appearing blurry in areas of specular reflection. GaussianShader shows a slight improvement by incorporating the GGX BRDF model, leading to marginally better results in rendering specular regions. In contrast, 3iGS stands out by employing a general rendering function that predicts neural features of illumination field and BRDF instead of relying on physical parameters. This approach allows 3iGS to surpass existing methods significantly, capturing the intricate details within specular highlights with remarkable precision.

Table 2: A quantitative comparisons (PSNR / SSIM / LPIPS) between 3DGS [17], GaussianShader [13], and our method on real world scenarios on Tanks and Temples Dataset [18]

	Tanks and Temples Dataset [18]					
	Barn	Caterpillar	Family	Ignatius	Truck	Avg.
			PSNR↑			
3DGS [17]	29.13	26.17	34.88	29.50	28.38	29.61
G.Shader(reproduced) [13]	27.67	25.23	33.52	28.28	27.61	28.46
Ours	**29.73**	**27.04**	**35.36**	**30.04**	**28.82**	**30.20**
			SSIM↑			
3DGS [17]	0.920	0.932	0.982	0.973	0.945	0.950
G.Shader(reproduced) [13]	0.897	0.915	0.977	0.968	0.935	0.938
Ours	**0.923**	**0.938**	**0.983**	**0.974**	**0.947**	**0.953**
			LPIPS↓			
3DGS [17]	0.113	0.074	0.023	0.032	0.059	0.060
G.Shader(reproduced) [13]	0.147	0.098	0.029	0.039	0.071	0.077
Ours	**0.112**	**0.071**	**0.022**	**0.031**	**0.057**	**0.058**

5.3 Discussion

In the comparisons detailed in Sec. 5.2, 3iGS demonstrates superior performance over the established baselines, delivering both quantitatively and qualitatively enhanced renderings in a majority of test cases on real time rendering rasterisation approaches. In the NeRF Synthetic dataset, 3iGS surpasses the prior 3DGS and GaussianShader. Although GaussianShader reportedly performs slightly better on the Shiny Blender dataset, we have included both reported and reproduced results based on the official code repository from the authors. We postulate that the Shiny Blender dataset scenes, which comprise single objects only, presents simpler geometries which facilitates an easier recovery of intrinsic material properties essential for rendering view-dependent effects. In addition, specular reflections in this dataset is primarily dominated by direct illumination from an external environment map. Thus GaussianShader which models direct lighting with a differentiable environment cube map performs well. However, when presented with a complex scene containing multiple objects, such as the NeRF Synthetic dataset shown in Fig. 1 with its intricate intra-scene interactions, GaussianShader struggles to accurately recover the physical rendering parameters. Furthermore these lighting scenarios are more complex due to indirect lighting. Therefore jointly modelling direct and indirect lighting using a continuous local incident field is crucial. NeRF based approaches reported above present competitive results. Yet, such methods are extremely slow to train, often requiring days, and are unable to perform real-time rendering needed for interactive applications.

Comparing across all methodologies, our 3iGS method presents an attractive and pragmatic alternative to achieve excellent rendering quality while balancing rendering speed, as discussed in Sec. 5.4.

Fig. 5: In contrast to 3DGS [17] and GaussianShader [13], our 3iGS method uniquely identifies both the golden specular highlights and the reflections on the Medium Tom as seen in the plastic surface of the Floor Tom (top row). Our approach successfully captures the detailed specular highlights on every cymbal within the drum setup from the Blender dataset, as presented in [22].

5.4 Ablation Studies

Table 3: An ablation study of our model on the Blender synthetic dataset. We experiment 3iGS under a variety of model parameters. In the first row, we directly an outgoing radiance field similar to NeRF based methods. The second row omits the prediction of a BRDF roughness parameter which encodes the viewing direction as IDE. Both experimental results are inferior compared to our complete model.

	PSNR	SSIM	LPIPS
Ours (outgoing radiance field)	32.38	0.965	0.035
Ours (no roughness parameter, i.e IDE)	33.26	0.967	0.031
Ours (complete model)	**33.64**	**0.970**	**0.029**

In Tab. 3, we study the effectiveness of our design choices and parameters for 3iGS. In the first row, we use the Gaussian mean and interpolate features from the factorised tensors and predict the outgoing specular colours directly. In this scenario, we predict the outgoing radiance field similar to a NeRF like manner for specular colours. In the second row, we abandon the BRDF roughness parameters from the Gaussian features and apply a standard Fourier positional encoding of viewing direction. Both cases led to inferior renderings as compared to our complete model.

In Tab. 4, we illustrate the training and rendering speed (test) of 3iGS against 3DGS and GaussianShader. We normalise the speed based on 3DGS. Our model performs competitively and achieve real time rendering speed although it is slower than 3DGS whereas GaussianShader performs much slower than the vanilla model. We attribute the efficient rendering speed to the use of factorised tensors for the illumination field.

Table 4: We evaluate the test and train speed of 3DGS [17] and GaussianShader [13] on a single Tesla V100 32Gb VRAM GPU with the original codebase and settings advocated by the authors. We then report the results normalised with these rendering speed of 3DGS.

	Test	Train
3DGS	1.0x	1.0x
GaussianShader	6.3x slower	12.1x slower
Ours	2.0x slower	3.2x Slower

6 Limitations and Weaknesses

3iGS inherits the main challenges of factorised tensors as [9]. Our model is limited to scenes that fit within a defined bounding box. Future works could explore this direction in warping unbounded scenes to fit a tensorial grid representation. Furthermore, 3iGS inherits the weaknesses of 3DGS; a large VRAM GPU is necessary to fit 3D Gaussians, and to evaluate the illumination field. A straightforward workaround is to reduce the number of Gaussians created by adding an upper bound on the number of produced Gaussians in the adaptive control step. Our work also inherits 3DGS's difficulty in producing accurate scene geometry.

7 Conclusion

We introduce our work, *Factorised Tensorial Illumination for 3D Gaussian Splatting (3iGS)*, to enhance the view-dependent effects in rendering Gaussian radiance fields. Our approach overcomes the constraints of previous methods, which relied on optimising an outgoing radiance field of independent Gaussians with Spherical Harmonics (SH) parameters. We illustrate that superior view-dependent effects in 3DGS can be attained by depicting an outgoing radiance field as a continuous illumination field and the Gaussian's BRDF characteristics in relation to this field. Distinct from other methods depending on over-simplified yet restrictive rendering equations that require prediction of physical attributes of scene surfaces for shading, our methodology proves to be more efficacious. Furthermore, we have shown that fast rendering speeds are attainable through the representation of an illumination field with factorised tensors. We demonstrated our claims across diverse datasets, from synthetic to real-world environments, and compared against prior art on both quantitative and qualitative metrics. We also evaluate the effectiveness of our model parameters and design choices through an ablation study. Finally we acknowledge the limitations of our research as a catalyst for future investigative directions. Our code is released here. **Acknowledgement** This study is supported under the RIE2020 Industry Alignment Fund - Industry Collaboration Projects (IAF-ICP) Funding initiative, as well as cash and in-kind collaboration from the industry partner(s). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore.

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. arXiv preprint arXiv:2304.06706 (2023)
4. Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Neural reflectance fields for appearance acquisition. arXiv preprint arXiv:2008.03824 (2020)
5. Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.: Nerd: Neural reflectance decomposition from image collections. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12684–12694 (2021)
6. Boss, M., Jampani, V., Braun, R., Liu, C., Barron, J., Lensch, H.: Neural-pil: Neural pre-integrated lighting for reflectance decomposition. Advances in Neural Information Processing Systems **34**, 10691–10704 (2021)
7. Burley, B., Studios, W.D.A.: Physically-based shading at disney. In: Acm Siggraph. vol. 2012, pp. 1–7. vol. 2012 (2012)
8. Carroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. Psychometrika **35**(3), 283–319 (1970)
9. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022)
10. Cook, R.L., Torrance, K.E.: A reflectance model for computer graphics. ACM Transactions on Graphics (ToG) **1**(1), 7–24 (1982)
11. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5501–5510 (2022)
12. Greger, G., Shirley, P., Hubbard, P.M., Greenberg, D.P.: The irradiance volume. IEEE Computer Graphics and Applications **18**(2), 32–43 (1998)
13. Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., Ma, Y.: Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces (2023)
14. Jin, H., Liu, I., Xu, P., Zhang, X., Han, S., Bi, S., Zhou, X., Xu, Z., Su, H.: Tensoir: Tensorial inverse rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2023)
15. Kajiya, J.T.: The rendering equation. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques. pp. 143–150 (1986)
16. Kautz, J., Snyder, J., Sloan, P.P.J.: Fast arbitrary brdf shading for low-frequency lighting using spherical harmonics. Rendering Techniques **2**(291-296), 1 (2002)
17. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics (ToG) **42**(4), 1–14 (2023)
18. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics **36**(4) (2017)

19. Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8456–8465 (2023)
20. Liang, R., Chen, H., Li, C., Chen, F., Panneer, S., Vijaykumar, N.: Envidr: Implicit differentiable renderer with neural environment lighting. arXiv preprint arXiv:2303.13022 (2023)
21. Mahajan, D., Ramamoorthi, R., Curless, B.: A theory of frequency domain invariants: Spherical harmonic identities for brdf/lighting transfer and image consistency. IEEE transactions on pattern analysis and machine intelligence **30**(2), 197–213 (2007)
22. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision. pp. 405–421 (2020)
23. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
24. Munkberg, J., Hasselgren, J., Shen, T., Gao, J., Chen, W., Evans, A., Müller, T., Fidler, S.: Extracting triangular 3d models, materials, and lighting from images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8280–8290 (2022)
25. Shi, Y., Wu, Y., Wu, C., Liu, X., Zhao, C., Feng, H., Liu, J., Zhang, L., Zhang, J., Zhou, B., Ding, E., Wang, J.: Gir: 3d gaussian inverse rendering for relightable scene factorization (2023)
26. Srinivasan, P.P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., Barron, J.T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7495–7504 (2021)
27. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5459–5469 (2022)
28. Technologies, U.: Light probes, `https://docs.unity3d.com/Manual/LightProbes.html`
29. Technologies, U.: Reflection probe, `https://docs.unity3d.com/Manual/class-ReflectionProbe.html`
30. Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-nerf: Structured view-dependent appearance for neural radiance fields. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5481–5490. IEEE (2022)
31. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
32. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020)