

Towards Efficient and Accurate Neural Geometry and Appearance Representations

Silong Yong

CMU-RI-TR-25-36

March 28, 2025



The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:

Prof. Katia Sycara, *chair*

Prof. Shubham Tulsiani

Tianyi Zhang

*Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Robotics.*

Copyright © 2025 Silong Yong. All rights reserved.

To all the people that supported me along the way.

Abstract

The challenge of developing scene representations for world modeling has persisted for a long time. The advent of neural scene representations, which leverage neural networks and volume rendering, has unlocked new possibilities for constructing highly realistic world models. However, volume rendering—the key mechanism behind these advances—remains computationally expensive due to the large number of MLP evaluations required for each sampled point along a ray. Prior research has tackled this inefficiency by introducing novel neural networks or data structures. In this thesis, we take a different approach by revisiting the mathematical formulation of volume rendering and introducing GL-NeRF, a new perspective that employs Gauss-Laguerre quadrature. GL-NeRF enables efficient querying of learned geometric representations without incurring additional computational overhead. Specifically, it dramatically reduces the number of MLP evaluations needed for volume rendering without relying on extra data structures or additional neural networks. Its straightforward formulation allows seamless integration into any NeRF-based model. We first establish the theoretical foundation for using Gauss-Laguerre quadrature and then demonstrate its versatility by incorporating it into three different NeRF frameworks. This approach results in a more efficient geometric representation, advancing towards a more generalized model of the world.

Another essential aspect of world modeling is appearance modeling, typically achieved through inverse rendering within the framework of neural scene representations. This process seeks to disentangle geometry, materials, and lighting from image data—a long-standing challenge in computer vision and graphics. Recent breakthroughs in neural rendering have enabled highly realistic and physically plausible inverse rendering results by integrating neural scene representations with inverse rendering techniques. The emergence of 3D Gaussian Splatting as a novel scene representation has further propelled this progress, demonstrating real-time rendering capabilities. However, these models often treat opacity and material properties as independent parameters, leading to theoretical inaccuracies. Drawing inspiration from radiative transfer theory, we enhance the opacity term by introducing a neural network that takes material properties as input, thereby enabling the modeling of cross-section—a material-dependent term in physics—along with a physically sound activation function. This approach allows the gradients of material properties to be influenced not only by color but also by opacity, enforcing an additional constraint that improves optimization. By integrating our method into three Gaussian

Splatting-based inverse rendering models, we achieve substantial improvements in novel view synthesis and material modeling. This formulation underscores the importance of leveraging correct physical priors to develop more accurate and generalizable scene representations across diverse scenarios.

This thesis marks an initial step toward enhancing neural scene representations in both geometry and appearance, with the ultimate goal of developing generalizable neural models that comprehensively capture both aspects.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Katia Sycara, for her invaluable guidance and support. Her kindness and dedication have been a constant source of inspiration, shaping my commitment to conducting meaningful and impactful research. I am deeply grateful to my family for their unwavering support—I couldn’t have made it this far without them. A special thanks to my girlfriend for her understanding and companionship throughout this journey. I would also like to express my appreciation to Dr. Yaqi Xie, Dr. Simon Stepputtis, Dr. Dana Hughes, Dr. Bernhard Kerbl, Zifu Wan, and Ce Zhang—amazing researchers whose deep insights, support, and encouragement have been invaluable along the way. I would also like to thank my committee members, Prof. Katia Sycara, Prof. Shubham Tulsiani and Tianyi Zhang, for their generous help.

Funding

This work has been funded in part by the Army Research Laboratory (ARL) award W911NF-23-2-0007 and W911QX-24-F-0049, DARPA award FA8750-23-2-1015, and ONR award N00014-23-1-2840.

Contents

1	Introduction	1
1.1	Outline	4
2	Background	7
2.1	Rendering	7
2.1.1	Volume rendering	7
2.1.2	Inverse Rendering	8
2.2	Neural Scene Representations	9
2.2.1	NeRF	9
2.2.2	3D Gaussian Splatting	9
3	Efficient Geometry Representation	11
3.1	Motivation	11
3.2	Quadrature Method	13
3.3	GL-NeRF: Sampling Strategy with Gauss-Laguerre Quadrature	15
3.3.1	Volume rendering and Gauss-Laguerre quadrature	15
3.4	Results	20
3.4.1	Comparison with baselines	23
3.4.2	Discussion on acceleration	23
3.4.3	Ablation studies	24
3.4.4	Discussion on GL-NeRF usage for training	24
3.5	Conclusion	25
4	Accurate Appearance Modeling	27
4.1	Motivation	27
4.2	The Bouguer-Beer-Lambert Law	29
4.3	OMG: Opacity Modeling in Gaussian Splatting using the Bouguer-Beer-Lambert Law	31
4.3.1	Physically Correct Activation	31
4.3.2	Neural Cross Section Modeling	33
4.3.3	Gradient Flow Analysis	33
4.3.4	Analysis	34
4.4	Results	35
4.4.1	Implementation Details	35

4.4.2	Results on Synthetic Data	36
4.4.3	Results on Real-World Data	37
4.5	Conclusion	39
5	Conclusions	41
5.1	Summary	41
5.2	Discussion	42
5.3	Limitations	42
5.4	Open Directions	43
A	Appendix	45
A.1	GL-NeRF	45
A.1.1	Implementation details	45
A.1.2	Gauss-Laguerre quadrature	46
A.1.3	Proof for Theorem 1	50
A.1.4	Quantitative results on NeRF-Synthetic and LLFF datasets	52
A.2	OMG	54
A.2.1	Hyperparameters	54
A.2.2	Full Derivation of Equations	54
A.2.3	Additional Results	54
A.2.4	Visualization of Cross Section	55
	Bibliography	61

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

3.1	Verification on using the Gauss-Laguerre quadrature for volume rendering. We plot the red channel of the color function w.r.t. the ray it corresponds to. The color function remains zero in most of the interval (bottom). We use a 7th-degree polynomial to approximate the non-zero region (top). As can be seen, the color function itself is similar to a polynomial, validating the use of our approach.	17
3.2	Point Selection strategy in GL-NeRF. We choose points along the ray that satisfy the integral from zero to the point of the volume density function should be equal to the roots of Laguerre polynomials. The points selected is then used for querying the color. In the figure above is an example of choosing 5 points using a 5-degree Laguerre polynomial. The number on the plot indicates the value of the integral from zero to the right boundary of the region.	19
3.3	Qualitative results on LLFF (top) and NeRF-Synthetic (bottom) datasets. We could tell from the comparisons that the drop in performances has minimal effect on the visual quality.	20
3.4	Comparison between GL-NeRF and vanilla NeRF in terms of render time and quantitative metrics. Each data point in the visualization represents an individual scene’s performance characteristics. Our results demonstrate that GL-NeRF achieves $1.2\times$ to $2\times$ speedup in rendering time per image compared to the baseline NeRF implementation, while maintaining comparable visual quality. The performance metrics show minimal degradation despite this substantial reduction in computational cost.	21
3.5	Effect of sample number. The first five columns correspond to the number of sampled points on top. Column 6 (Ori) shows TensorRF’s original sampling strategy (100+ points). Column 7 provides ground truth reference. Our method achieves visual fidelity comparable to the original strategy using just 4-8 samples. Note: The blur in columns 1-2 stems from inherent limitations in piecewise-constant density estimation.	22

4.1	This illustration demonstrates the material-dependent nature of opacity through a comparative study of two translucent media. Left: A glass exhibits minimal light attenuation, transmitting incident radiation with negligible intensity reduction. Right: In contrast, the same incident light undergoes complete absorption within the gaseous medium. This fundamental difference motivates our key insight: each 3D Gaussian primitive, when associated with material properties, should be modeled as an absorbing medium with a material-aware opacity coefficient. We therefore propose to explicitly incorporate material properties as governing factors in the absorption characteristics of Gaussian primitive, establishing a physically-grounded relationship between material and opacity.	28
4.2	An illustration of the Bouguer-Beer-Lambert law. Consider a ray passing through an absorbing body consists of particles of some type. The cross section depends on the area that each particle would affect (usually bigger than the size of the particle and equals the size of the particle if treated as a hard sphere). When traveling, the intensity of the light would be reduced by the areas that each particle affected. The reduction in intensity is therefore affected by the area around the particles and the number of particles, corresponding to cross section σ and number density n in the main paper.	30
4.3	Pipeline Overview. Our method extends existing 3DGS-based inverse rendering approaches [19, 28, 32, 37] through a novel architectural modification. Departing from prior works that treat opacity as an independent parameter, we condition it on material properties. Specifically, we introduce a neural network that takes material attributes as input and outputs cross-section values, which are then multiplied with the opacity to integrate the Bouguer-Beer-Lambert law into the framework. During optimization, material properties receive dual gradient signals: (1) color-derived gradients via differentiable PBR and (2) alpha-derived gradients enforced by our network. This dual-path supervision imposes an additional physical constraint, ensuring strict adherence to the Bouguer-Beer-Lambert law throughout the rendering pipeline.	32
4.4	Qualitative comparison of albedo, relighting and roughness on Synthetic4Relight dataset [70]. Our method is able to decouple specular effects from albedo compared to R3DG, the baseline we implemented our method on. It is also capable of providing roughness estimation more precisely. We add a gamma correction to the relighting results to make the visual differences clearer.	36

4.5	Qualitative comparison on Shiny Blender [56] and Glossy Synthetic [39] datasets. Our method is able to provide satisfactory normal estimation compared to the baseline. Notice that our method does not include any additional supervision or constraints for normal estimation. Our method is also able to provide much more accurate view synthesis results when doing physically-based rendering. These results suggest the significance of introducing the physically correct model for inverse rendering.	38
4.6	Qualitative Comparison of GS-IR on MIP-NeRF 360 [5]. Our method is able to provide much more accurate novel view synthesis results compared to the baseline GS-IR [37] in which we plugged our model. The GS-IR method produces a blurred prediction in the Bicycle scene (first row), which our approach successfully resolves. Additionally, as demonstrated in the second row, our method achieves significantly more precise and detailed normal estimation.	39
A.1	More qualitative example in comparison to GaussianShader on Glossy Synthetic dataset.	56
A.2	More qualitative example in comparison to GS-IR on MIP-NeRF 360 dataset.	57
A.3	More qualitative example in comparison to R3DG on Synthetic4Relight dataset.	57
A.4	More qualitative example in comparison to R3DG on Synthetic4Relight dataset. We add a gamma correction to the relighting results to make the visual differences clearer.	58
A.5	More qualitative example in comparison to GaussianShader on Glossy Synthetic and Shiny Blender datasets.	58
A.6	Visualization of the learned cross section on Mip-NeRF 360 dataset. Dark indicates the lower cross section (meaning it lets more light pass through) and bright stands for the higher cross section. The example shows that the windows as well as the plastic bag on the floor actually have a lower cross section assigned to them, which corresponds to human intuition.	59

List of Tables

3.1	Gauss-Laguerre quadrature look-up table when $n = 8$	18
3.2	Quantitative comparison. We demonstrate that our method has a minimal performance drop while significantly reducing the number of color MLP calls.	22
3.3	Quantitative comparison when training with GL-NeRF. Vanilla refers to the vanilla NeRF and its sampling strategy while ours refers to replacing the fine sample stage in vanilla NeRF with our sampling strategy, i.e. GL-NeRF. The result for Vanilla NeRF is produced by rendering using more than 100 points while GL-NeRF only uses 32 points.	22
3.4	Ablation study on the number of points sampled. The more points we have, the better the performance will be. With 8 points, our method is comparable to the original sampling strategy in TensorRF.	23
3.5	Per-scene results on Blender dataset between InstantNGP and ours. We demonstrate that GL-NeRF is able to be plugged into ANY NeRF models.	24
3.6	Comparison between our method and TensorRF on Lego scene using WebGL-based renderer. The result is collected from an AMD Ryzen 9 5900HS CPU. GL-NeRF is able to provide almost real-time rendering while remaining similar quality as TensorRF.	25
4.1	Quantitative results on Synthetic4Relight dataset [70]. Our method is able to provide significant improvements universally. The improvements in novel view synthesis and relighting can be attributed to the improved estimation of albedo and roughness thanks to the correct modeling we use for constraining material properties.	37
4.2	Quantitative results when combined with GaussianShader [28]. We compare with GaussianShader on Shiny Blender [56] and Glossy Synthetic datasets [39]. Our method is able to outperform GaussianShader on novel view synthesis in terms of all three standard image quality metrics.	37

4.3	Quantitative Comparison on Mip-NeRF 360 [5]. The results show that our approach, when combined with GS-IR, can significantly improve the novel view synthesis results of real-world scenes on MIP-NeRF 360 [5]. Thanks to the constraint we add to opacity, our method is able to produce ~ 0.5 db PSNR gains along with SSIM and LPIPS improvements.	40
A.1	Per-scene quantitative comparison between TensorRF and ours.	50
A.2	Per-scene quantitative comparison between Vanilla NeRF and ours.	53
A.3	Quantitative results when training on Blender and LLFF Datasets.	53
A.4	Per-scene results on Synthetic4Relight dataset. For albedo reconstruction results, we follow the standard practice [19, 69] and scale each RGB channel by a global scalar.	55
A.5	Per-scene results on Shiny Blender dataset.	55
A.6	Per-scene results on Glossy Synthetic dataset.	56

Chapter 1

Introduction

The emergence of Neural Radiance Fields (NeRFs) [45] revolutionized the traditional methods for rendering as well as 3D representations. By introducing a coordinate-based neural network and relying on the volume rendering technique, it is able to provide photo-realistic novel view synthesis results. Gradient descent is used to update the network’s parameters to make it compatible with the underlying scene geometry with only 2D supervision. The success of NeRF hinges on volume rendering, which approximates an integral by densely sampling points along camera rays and querying a neural network for density and radiance. However, this process is computationally expensive: each pixel requires numerous network evaluations, many of which may be redundant. Recent efforts [18, 20, 49, 61] address this inefficiency by introducing specialized architectures or data structures to enable real-time rendering. While effective, these methods often demand task-specific training and lack generalizability. Such limitation motivates us to introduce generalizable efficient rendering methods that can be applied to any existing methods at only inference time, where computation is expected to be less costly. That being said, we want to find a way to accelerate any NeRF models without any additional training or fine-tuning. Such a design choice is natural since a learned neural network should have all the information needed, *e.g.*, surface positions of the underlying scene, namely the geometry, stored in its parameters.

By revisiting the volume rendering integral, which serves as the most crucial component for NeRF, we have the following observation: the integral itself not only is a well-developed tool for integrating point-wise information but also can inform

the point selection strategy during the sampling stage.

Motivated by this insight, we present a lightweight solution that can be seamlessly integrated into existing NeRF architectures requiring volume rendering while eliminating the need for retraining. Unlike previous approaches that introduce new network components or data structures, our method preserves the original framework architecture. By fundamentally reformulating the volume rendering integral computation, we achieve significant sample efficiency with only minimal algorithmic modifications.

The fundamental contribution lies in transforming the volume rendering integral into an exponentially weighted radiance integral through a strategic variable substitution. This reformulation reveals that the Gauss-Laguerre quadrature - a numerical integration method specifically designed for such an integral - provides the theoretically optimal approximation for it. We leverage this insight to develop GL-NeRF, which directly computes the integral using this quadrature rule with modification in the sampling stage of NeRF rendering. The approach yields three key advantages:

- **Computational Efficiency:** By exploiting the mathematical properties of Gauss-Laguerre quadrature, we achieve substantial reductions in both time and memory requirements.
- **Theoretical Guarantees:** The method maintains high precision due to the optimal approximation properties of Gauss quadrature. It provides a different perspective for computing volume rendering.
- **Practical Implementation:** While the method requires density estimation in the space, this can be accelerated through modern techniques like tensor factorization [12].

In addition to better understanding the fundamentals of geometry modeling and querying, appearance modeling in neural scene representations is also important for an accurate 3D digital model of the world. Appearance modeling is usually formulated as an inverse rendering problem, where the goal is to recover physical properties such as materials and lighting conditions from a set of images [3, 14]. These properties are crucial for applications including material editing, relighting, and real-to-virtual scene transfer. However, inverse rendering remains fundamentally challenging due to its ill-posed nature - different configurations of lighting, materials,

and geometry can produce visually identical outputs. The advent of neural rendering techniques [4, 5, 6, 12, 45, 46] has enabled new approaches using implicit neural representations (NeRF variants) for inverse rendering [10, 29, 54, 63, 67, 69]. More recently, 3D Gaussian Splatting (3DGS) [32] has emerged as a powerful alternative, offering real-time performance through compact scene representations. This has led to 3DGS-based inverse rendering methods [19, 28, 37] that leverage its explicit structure to incorporate material parameters (albedo, roughness, etc.) within a physically-based rendering framework. Although 3DGS naturally enables efficient training and high-quality rendering, its direct application to inverse rendering presents notable limitations. A key challenge stems from insufficient modeling of the physical relationships between Gaussian attributes (e.g., the interdependence of opacity, albedo, and roughness). This lack of explicit material constraints leads to an under-constrained optimization problem, ultimately compromising the quality and physical accuracy of inverse rendering results. Consequently, current 3DGS-based approaches leave significant room for improvement in material decomposition and scene reconstruction.

The disentangled nature of 3DGS-based methods naturally brings up the question of whether there should be dependencies between these parameters for a more accurate representation.

On the other hand, the physical law in radiative transfer, namely the Bouguer-Beer-Lambert law reveals that the intensity of light should decrease exponentially when passing through an absorbing body, and the factor that affects the rate consists of the density and the material of the body.

The Bouguer-Beer-Lambert law reveals the fundamental missing correlation between when utilizing the disentangled 3DGS method for inverse rendering: the materials each Gaussian primitive contains should contribute to the opacity of itself, mimicking the underlying interactions between light and absorbing bodies.

Inspired by this, we can introduce constraints at the modeling level to align the model better with the physical world. Specifically, opacity for each Gaussian primitive in 3DGS-based inverse rendering methods should follow the form derived from the Bouguer-Beer-Lambert law, which consists of a number density term and a material-dependent term. Intuitively, this can also be observed from a macroscopic comparison of light transport through different translucent materials as in Fig. 4.1. As shown, glass and gas exhibit fundamentally different optical behaviors - while glass transmits

red light with minimal attenuation, the same wavelength is completely absorbed by gas. This demonstrates a critical physical principle: **material composition determines opacity**. Current 3DGS-based inverse rendering approaches [19, 28, 37] fail to capture this relationship because their disentangled representations treat opacity as an independent parameter, decoupled from material properties. In contrast, NeRF’s implicit alpha blending naturally encodes material-opacity correlations through shared network weights. To address this limitation in 3DGS and to facilitate more accurate neural scene representation for appearances, we have to introduce a physics-aware opacity model that combines: (1) a material-dependent multiplicative term and (2) a physically-grounded activation function. Such a formulation creates a bidirectional coupling where material parameters influence both PBR shading and alpha blending, while receiving gradients from both processes for physically consistent optimization.

To validate both the theoretical soundness and practical efficacy of our formulation, we perform rigorous analytical verification through Taylor expansion analysis, demonstrating that conventional opacity computation represents a first-order approximation of our physically-grounded approach. We extensively evaluate our method by integrating it with three state-of-the-art baselines and testing across four established benchmarks encompassing both synthetic and real-world datasets. The comprehensive experimental results reveal consistent performance gains across all tested scenarios, with particular improvements in material property estimation, translating to superior novel view synthesis and relighting quality. These findings underscore the critical importance of embedding physically accurate constraints in inverse rendering frameworks, as evidenced by the universal performance enhancement achieved through the principled opacity modeling.

All in all, this thesis aims to step towards accurate and efficient neural scene representations by utilizing correct physics priors and their fundamental math properties, with the hope that it would inspire the research in this domain as well as have an impact in downstream real-world applications.

1.1 Outline

Chapter 2 will cover the background for neural scene representations, starting from rendering methods to specific models.

Chapter 3 will discuss the approach for improving math properties for these rendering techniques, leading towards a more efficient geometry representation.

Chapter 4 will discuss the fundamental physics priors that can be incorporated into the neural scene representations, aiming for more accurate appearance modeling.

Chapter 5 will summarize the efforts being made throughout this thesis, as well as discuss the limitations and potential future directions.

Chapter A includes all the details for the experiments and derivations discussed throughout the thesis.

1. Introduction

Chapter 2

Background

2.1 Rendering

2.1.1 Volume rendering

Volume rendering is a foundational technique in computer graphics and vision, extensively employed to project 3D scenes onto 2D images [16, 42, 60]. Mathematically, this process can be expressed as a weighted integral along viewing rays, where the contribution of each point is determined by its color and opacity (volume density):

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) \mathbf{c}(\mathbf{r}(t), d) dt, \quad (2.1)$$

where

$$T(t) = \exp\left(-\int_{t_n}^t \sigma(\mathbf{r}(s)) ds\right). \quad (2.2)$$

The continuous integral is typically approximated via a discrete sum over sampled points along the ray, as formulated in [42]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^N w_i \mathbf{c}(\mathbf{r}(t_i)), \quad (2.3)$$

where

$$w_i = T_i(1 - \exp(-\sigma(\mathbf{r}(t_i))\delta_i)), \quad (2.4)$$

2. Background

$$T_i = \exp(-\sum_{j=0}^{i-1} \sigma(\mathbf{r}(t_j))\delta_j), \quad (2.5)$$

t_i represents the sampled position along the ray and $\delta_i = t_{i+1} - t_i$ is the distance between two nearby sampled points. Modern implicit scene representations, such as NeRF [45], Plenoxels [18], and 3D Gaussians [32], universally adopt this rendering pipeline.

A key challenge in volume rendering is the inefficiency of uniform or random sampling, as it may evaluate redundant points in empty or low-density regions, increasing computational overhead. To address this, recent works have proposed techniques to optimize the approximation of the integral components (e.g., volume density and radiance). For instance, PL-NeRF [55] employs a piecewise linear approximation of volume density, reducing the number of samples required in the fine-stage rendering of [45]. AutoInt [38] and DIVER [61] replace Monte Carlo sampling with a neural network to directly estimate the density integral. DONeRF [49] leverages a depth oracle to focus sampling near surfaces, where contributions to the rendered image are most significant. MCNeRF [24] introduces Monte Carlo denoising for sample-efficient rendering, albeit requiring per-scene training of a denoiser network.

2.1.2 Inverse Rendering

Inverse rendering seeks to disentangle observed images into underlying scene properties—geometry, materials, and lighting—enabling applications like material editing and relighting [31, 36, 62, 70]. This is typically achieved by coupling physically-based rendering (PBR) with differentiable renderers to optimize scene parameters through gradient-based methods [13, 30]. The rendering equation, which underpins PBR, is given by:

$$C_o(x, \mathbf{v}) = \int_{\Omega} L_i(x, \mathbf{l}) f(\mathbf{l}, \mathbf{v}, x) (\mathbf{l} \cdot \mathbf{n}) d\mathbf{l}, \quad (2.6)$$

where $L_i(x, \mathbf{l})$ is the incoming light from direction $\mathbf{l}$ at point x , $\mathbf{n}$ is the surface normal direction and f is the Bidirectional Reflectance Distribution Function (BRDF). The integral is computed in the upper hemisphere of the surface. However, inverse rendering is inherently ill-posed; to mitigate ambiguity, prior works impose constraints such as: known lighting conditions [9, 54, 70], fixed illumination [15], explicit shadow handling [57], or simplified shadow-free assumptions [67].

The integration of differentiable volume rendering with PBR has expanded the scope of tractable scenarios, leveraging neural scene representations to model complex geometry and lighting interactions [2, 10, 29, 39, 47, 56, 63, 69].

2.2 Neural Scene Representations

2.2.1 NeRF

Neural Radiance Fields (NeRFs) have revolutionized novel view synthesis by representing scenes as continuous volumetric functions via coordinate-based MLPs [45]. Each pixel is rendered through volumetric integration (Eq. 2.3) along camera rays, with density σ and color c evaluated at hundreds of sampled points using the MLP. To capture high-frequency details, NeRF employs positional encoding $\gamma(x)$ to map 3D coordinates x to higher-dimensional space. The success of NeRF has spurred innovations across multiple dimensions. Mip-NeRF [4] addresses aliasing by convolving rays with conical frustums. Unbounded Scenes: Techniques like NeRF++ [66] and Mip-NeRF 360 [5] use spatial partitioning or contraction to handle large-scale environments. Methods like D-NeRF [51] introduce time-conditioned MLPs for temporal modeling. Panoptic NeRF [33] and NeSF [58] augment radiance fields with semantic labels.

Despite these advances, NeRF’s reliance on MLP queries results in slow training and rendering. Efforts to improve efficiency fall into two categories. 1. Explicit Data Structures: PlenOctrees [65], Instant NGP [46], and TensorRF [12] accelerate queries via voxel grids or hash tables, often at the cost of discretization artifacts. 2. Sample Efficiency: AdaNeRF [34] and DOnNeRF [49] reduce sample counts by predicting surface locations or adaptive sampling.

2.2.2 3D Gaussian Splatting

More Recent trends in neural rendering highlight 3D Gaussian Splatting (3DGS) [32] as a breakthrough, combining real-time rendering with high fidelity. Unlike NeRF’s implicit MLP, 3DGS [32] represents scenes using a collection of 3D Gaussian primitives. Each primitive G_i is parameterized by: a mean position μ_i , a covariance matrix Σ_i , an opacity o_i and a view-dependent outgoing radiance c_i . Additionally, the covariance

2. Background

matrix is factorized into a quaternion q_i and a scale s_i following Eq. 2.7 and the outgoing radiance is treated as a set of spherical harmonics coefficients.

$$\Sigma_i = R_i \text{diag}(s_i)^2 R_i^T, \quad (2.7)$$

R_i is the rotation matrix corresponds to the quaternion q_i .

Research has focused on efficiency, training speed and its downstream applications. LightGaussian [17] reduces memory usage via pruning; Compact3D [48] employs quantization. Taming 3DGS [41] accelerates convergence with gradient-aware densification. LangSplat [52] integrates language embeddings; Dynamic3DGS [40] models temporal scenes.

For inverse rendering tasks, recent 3DGS-based methods [19, 28, 37] augment each Gaussian with material properties m_i and normals n_i to each Gaussian to support physically-based rendering. These sets of parameters are used to produce PBR color, which serves as c_i in the original formulation.

The rendering process involves: first Transform 3D Gaussians to 2D screen space via camera parameters θ :

$$\mu_i^{2D} = R(\mu_i; \theta), \quad \Sigma_i^{2D} = R_\theta(\Sigma_i) \quad (2.8)$$

Afterwards, 3DGS adopts α -blending for rendering the final color of a specific pixel location x by compositing N sorted Gaussians:

$$\hat{C}(x) = \sum_{i=1}^N c_i \alpha_i(x) T_i(x), \quad T_i(x) = \prod_{j=1}^{i-1} (1 - \alpha_j(x)), \quad (2.9)$$

where $\alpha_i(x)$ is the contribution of the i -th Gaussian:

$$\alpha_i(x) = o_i \exp \left(-\frac{1}{2} (x - \mu_i^{2D})^T (\Sigma_i^{2D})^{-1} (x - \mu_i^{2D}) \right). \quad (2.10)$$

This corresponds to the volume rendering equation that is adopted in NeRF.

Finally, the optimization goal is to minimize the photometric loss between rendered and ground truth images:

$$L = \|C - \hat{C}\|^2, \quad (2.11)$$

Chapter 3

Efficient Geometry Representation

3.1 Motivation

As discussed before, previous works that aims to make NeRF models more efficient tends to introduce new data structures or adopt adaptive sampling method to reduce the sample points. Our efficient geometry representation also falls into the second category. While previous works draw their intuition from the underlying physics perspective and thus need different formulations of the sampling strategy and different neural network architecture for predicting the surface position of the underlying scenes, we propose our method based on a mathematical observation while maintaining the overall pipeline. Benefiting from this, our work could be seamlessly incorporated into any existing NeRF-related works without further training. On the other hand, despite being derived from the mathematical perspective, our method still intuitively satisfies the underlying physical constraints.

We rewrite the relevant equations here for future reference.

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^N w_i \mathbf{c}(\mathbf{r}(t_i)), \quad (3.1)$$

where

$$w_i = T_i(1 - \exp(-\sigma(\mathbf{r}(t_i))\delta_i)), \quad (3.2)$$

Algorithm 1 Gauss Laguerre Quadrature for Volume Rendering

Input: ray direction d , ray origin o , step size Δt , sample number M , Gauss-Laguerre quadrature weight look-up table L_w , point look-up table L_p

```

1:  $t_{\min}, t_{\max} = \text{RayIntersectBoundingBox}(d, o)$ 
2: if  $t_{\min} > t_{\max}$  then return  $\text{bg\_color}$ 
3: end if
4: Initialize  $t = t_{\min}$ , transmittance  $T = 1.0$ , already sampled point number  $n = 0$ 
5: while  $t < t_{\max}$  do
6:   if  $(n == m)$  then break
7:   end if
8:    $\text{pos} = o + t * d$ 
9:    $\sigma = \text{GetVolumeDensity}(\text{pos})$ 
10:   $x = -\log(T)$ 
11:   $x_{\text{next}} = x + \Delta t * \sigma$ 
12:  if  $x < L_p[n]$  and  $x_{\text{next}} \geq L_p[n]$  then
13:     $t_{\text{Laguerre}} = (L_p[n] - x) / (x_{\text{next}} - x)$ 
14:     $\text{pos}_{\text{sample}} = o + (t + t_{\text{Laguerre}} * \Delta t - \Delta t) * d$ 
15:     $\text{ray\_color} += L_w[n] * \text{GetColor}(\text{pos}_{\text{sample}})$ 
16:     $n = n + 1$ 
17:  end if
18:   $t = t + \Delta t$ 
19:   $T = T * \exp(-\sigma * \Delta t)$ 
20: end while
21:  $\text{bg\_weight} = \text{sum}(L_w[n :])$ 
22:  $\text{ray\_color} = \text{ray\_color} + \text{bg\_weight} * \text{bg\_color}$ 
23: return  $\text{ray\_color}$ 

```

$$T_i = \exp\left(-\sum_{j=0}^{i-1} \sigma(\mathbf{r}(t_j))\delta_j\right), \quad (3.3)$$

t_i represents the sampled position along the ray and $\delta_i = t_{i+1} - t_i$ is the distance between two nearby sampled points.

3.2 Quadrature Method

An n -point Gaussian quadrature rule [21] provides a numerical integration technique that exactly computes integrals of polynomials up to degree $2n - 1$, achieving the maximal possible accuracy for quadrature-based integral approximation. Conceptually, this approach can be viewed as estimating an integral through quadrature as shown in Eq. 3.4.

$$\int_{-1}^1 p(x)dx = \sum_{i=1}^n w_i p(x_i), \quad (3.4)$$

where $p(x)$ is a polynomial of degree $2n - 1$, $w(x)$ is a weight function, and I is the interval of integration. We first define the orthogonality of two polynomials $p_m(x)$ and $p_n(x)$:

$$\int_{-1}^1 p_m(x)p_n(x)dx = 0, \quad (3.5)$$

where $p_m(x)$ is of degree m , $p_n(x)$ is of degree n and $m \neq n$. we can use long division for $p(x)$ to obtain

$$p(x) = q(x)L_n(x) + r(x), \quad (3.6)$$

where $L_n(x)$ is a polynomial of degree n that is orthogonal to any polynomial of degree less than n (*i.e.*, the n -th degree Legendre polynomial), and $q(x)$, $r(x)$ are both polynomials with degree less than n . Then

$$\int_{-1}^1 p(x)dx = \int_{-1}^1 q(x)L_n(x)dx + \int_{-1}^1 r(x)dx. \quad (3.7)$$

Since $L_n(x)$ is orthogonal to all polynomials of degree less than n , the first term on the right-hand side of Eq. 3.7 vanishes. As this term does not contribute to the integral computation, we may safely disregard it when constructing the quadrature rule.

Consequently, we select the nodes x_i as roots of $L_n(x)$, satisfying $L_n(x_i) = 0$ [26].

3. Efficient Geometry Representation

With this foundation, an appropriate choice of weights w_i enables exact computation of the quadrature. Specifically, with n such nodes, we can exactly evaluate the second term in Eq. 3.7 – which represents the integral of a polynomial of degree less than n – thereby providing precise computation of Eq. 3.4.

In general, given a function $f(x)$, Gauss quadrature computes its integral on $[-1, 1]$ using

$$\int_{-1}^1 f(x)dx \approx \sum_{i=1}^n w_i f(x_i), \quad (3.8)$$

where x_i ($i = 1, 2, \dots, n$) are the roots of orthogonal polynomials defined on the interval $[-1, 1]$. This quadrature rule is known as Gauss-Legendre quadrature, as the orthogonal polynomials on $[-1, 1]$ with weight function $g(x) \equiv 1$ are precisely the Legendre polynomials. The n -th degree Legendre polynomial can be expressed in the form [25, 27, 53]:

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n, \quad (3.9)$$

and w_i is computed using Eq. 3.10 as shown in [1].

$$w_i = \frac{2}{(1 - x_i^2)[P'_n(x_i)]^2}. \quad (3.10)$$

Gauss-Laguerre quadrature is an extension of Gauss quadrature for approximating integrals of the form

$$\int_0^\infty e^{-x} f(x) dx \approx \sum_{i=1}^n w_i f(x_i). \quad (3.11)$$

In this formulation, the weight function is given by $g(x) = e^{-x}$ and the integration domain is the semi-infinite interval $[0, \infty)$. The quadrature nodes $x_i \in [0, \infty)$ correspond to the roots of the n -th order Laguerre polynomial

$$L_n(x) = \frac{1}{n!} \left(\frac{d}{dx} - 1 \right)^n x^n, \quad (3.12)$$

a class of polynomials that are orthogonal over the interval $[0, \infty)$ with respect to the weight function $g(x) = e^{-x}$. The weight for computing the quadrature is computed as

$$w_i = \frac{x_i}{(n+1)^2 [L_{n+1}(x_i)]^2}. \quad (3.13)$$

Although the computation of the nodes x_i and weights w_i is complicated, in practice these values are typically precomputed and stored in lookup tables for various orders n .

3.3 GL-NeRF: Sampling Strategy with Gauss-Laguerre Quadrature

Our algorithm development was motivated by a fundamental observation about the volume rendering integral. The expression in Eq. 3.1 provides an approximation to the integral

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) \mathbf{c}(\mathbf{r}(t), d) dt, \quad (3.14)$$

where

$$T(t) = \exp\left(-\int_{t_n}^t \sigma(\mathbf{r}(s)) ds\right). \quad (3.15)$$

3.3.1 Volume rendering and Gauss-Laguerre quadrature

Let

$$x(t) = \int_{t_n}^t \sigma(\mathbf{r}(s)) ds, \quad (3.16)$$

we have

$$\frac{dx}{dt} = \sigma(\mathbf{r}(t)). \quad (3.17)$$

Since $\sigma(\mathbf{r}(t)) \geq 0$, the function $x(t)$ is monotonically non-decreasing with respect to t . This establishes a unique correspondence between x and t on any interval where $x(t)$ is increasing. Based on this observation, we perform a change of variables in

Eq. 3.14 to obtain

$$\begin{aligned}
 C(\mathbf{r}) &= \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) \mathbf{c}(\mathbf{r}(t), d) dt \\
 &= \int_{t_n}^{t_f} e^{-x} \mathbf{c}(\mathbf{r}(t), d) \frac{dx}{dt} dt \\
 &= \int_{x(t_n)}^{x(t_f)} e^{-x} \mathbf{c}(\mathbf{r}(x), d) dx.
 \end{aligned} \tag{3.18}$$

As shown in Eq. 3.18, the volume rendering integral represents a weighted integration of $\mathbf{c}(\mathbf{r}(x), d)$ with the exponential weight function $g(x) = e^{-x}$. We can extend the integration domain from $[x(t_n), x(t_f)]$ to $[0, \infty)$ since the integral between $[0, x(t_n))$ and $(x(t_f), \infty)$ are zero. Thus, we have

$$C(\mathbf{r}) = \int_0^\infty e^{-x} \mathbf{c}(\mathbf{r}(t(x)), d) dx, \tag{3.19}$$

a pure exponentially weighted integral with respect to the color function, which is of the exact same form as required by the Gauss-Laguerre quadrature.

Gauss-Laguerre quadrature for volume rendering

As discussed in Sec. 3.2, the Gauss-Laguerre quadrature guarantees the highest algebraic precision when computing integral over polynomials. When applying this method to volume rendering integrals, a critical consideration emerges: **is the color function a polynomial, or can it be approximated by a polynomial with a satisfactory error rate?**

To answer this question, we first analytically give out a fundamental theorem, and then empirically approximate the color function with polynomials.

Theorem 1 (Stone-Weierstrass theorem). *Suppose f is a continuous real-valued function defined on the real interval $[a, b]$. For every $\epsilon > 0$, there exists a polynomial p such that for all x in $[a, b]$, we have $|f(x) - p(x)| < \epsilon$.*

Since the pixel color is primarily contributed by points with density exceeding a certain threshold (*i.e.*, regions near surfaces), we can focus our analysis on these non-empty regions while disregarding empty space. In Fig. 3.1 we plot a representative of how the color function looks like. As can be seen from the figure, it has a major region with values greater than zero while the others remain zero. When approximating the

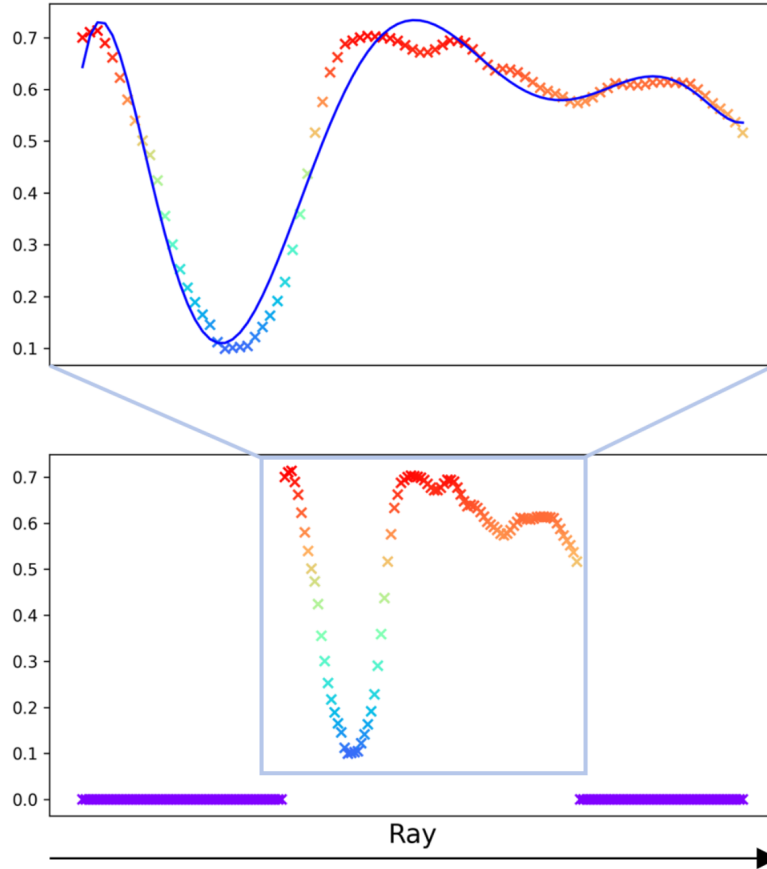


Figure 3.1: Verification on using the Gauss-Laguerre quadrature for volume rendering. We plot the red channel of the color function w.r.t. the ray it corresponds to. The color function remains zero in most of the interval (bottom). We use a 7th-degree polynomial to approximate the non-zero region (top). As can be seen, the color function itself is similar to a polynomial, validating the use of our approach.

non-zero region with a 7-th degree polynomial, we have a relative error rate lower than 6.5%. While the relative error is not sufficiently low, we argue that we can increase the degree to better approximate it since it’s continuous by nature. On the other hand, this specific landscape is fluctuated and for most of the cases, the error rate could be smaller than 1%. These observations suggest that Theorem 1 (Weierstrass approximation theorem) remains applicable in our context, thereby justifying the use of Gauss-Laguerre quadrature for computing the volume rendering integral.

Point selection in GL-NeRF

In contrast to NeRF’s stochastic sampling approach, our Gauss-Laguerre quadrature enables a *deterministic* point selection strategy for color sampling. Recall Eq. 3.16 is the integral variable for Eq. 3.19. To apply Gauss-Laguerre quadrature, we should first, Select points $\{x_i\}_{i=1}^n$ corresponding to the roots of the n -th degree Laguerre polynomial, then compute their associated $\{t_i\}_{i=1}^n$ via the inverse mapping for Eq. 3.16 Fig. 3.2 illustrates this process for $n = 5$. In the figure, colored regions represent distinct integration intervals and annotated values show the cumulative density $\int_0^{t_i} \sigma(\mathbf{r}(s))ds$ at each boundary. The complete rendering procedure is formalized in Algorithm 1.

x_i	w_i
0.17	3.69×10^{-1}
0.90	4.19×10^{-1}
2.25	1.76×10^{-1}
4.27	3.33×10^{-2}
7.05	2.79×10^{-3}
10.76	9.08×10^{-5}
15.74	8.49×10^{-7}
22.86	1.05×10^{-9}

Table 3.1: Gauss-Laguerre quadrature look-up table when $n = 8$.

Intuitive understanding of the points selected using the Gauss-Laguerre quadrature

Since the points near the surface contribute the most to the final color of the pixel as discussed in [34, 49, 50], the optimal point selection strategy should choose points near

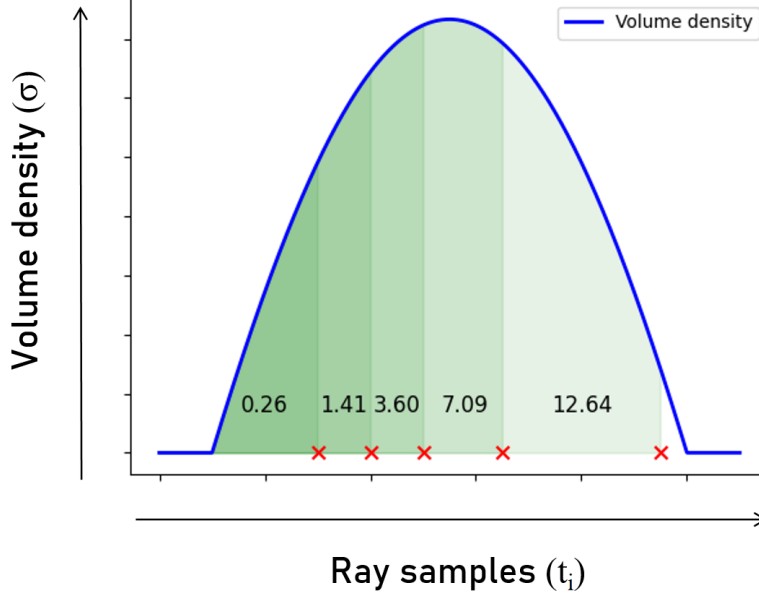


Figure 3.2: **Point Selection strategy in GL-NeRF.** We choose points along the ray that satisfy the integral from zero to the point of the volume density function should be equal to the roots of Laguerre polynomials. The points selected is then used for querying the color. In the figure above is an example of choosing 5 points using a 5-degree Laguerre polynomial. The number on the plot indicates the value of the integral from zero to the right boundary of the region.

the surface. The volume density, on the other hand, increases remarkably near the surface and remains close to zero at other areas. Therefore, the integral value of it Eq. 3.16 should also increase significantly near the surface and remain almost unchanged throughout the rest of the space. Therefore, most of the points chosen using GL-NeRF should lie around the surface of the underlying scene. Consider the case when $n = 8$, where we select points $\{t_i\}_{i=1}^8$ such that the integrated density $x(t_i)$ from Eq. 3.16 matches the precomputed roots $\{x_i\}_{i=1}^8$ in Tab. 3.1. From the table, we can see that the first few value for x_i (e.g., x_1, x_2, x_3) are small so that they could be reached by the integral of volume density near the surface easily. These values have relatively larger weights assigned to them. Evaluating the color of these points using a neural network and summing them up using the weights w_i given in Eq. 3.1 following Eq. 3.11 would contribute mostly to the pixel color. Notice that even though the last few x_i are quite large and may not be reached by Eq. 3.16 along the ray, their corresponding weights

are so small that they have minimal impact on the final result of the pixel color. Consequently, GL-NeRF’s selected sampling points naturally concentrate near surfaces, just as in previous works [34, 49, 50] that design different neural networks for estimating the surface position, but only without any additional neural networks. Therefore, thanks to the nice property of the Gauss quadrature, ideally we can select the optimal points for computing volume rendering integral if the volume density estimation is oracle.

3.4 Results

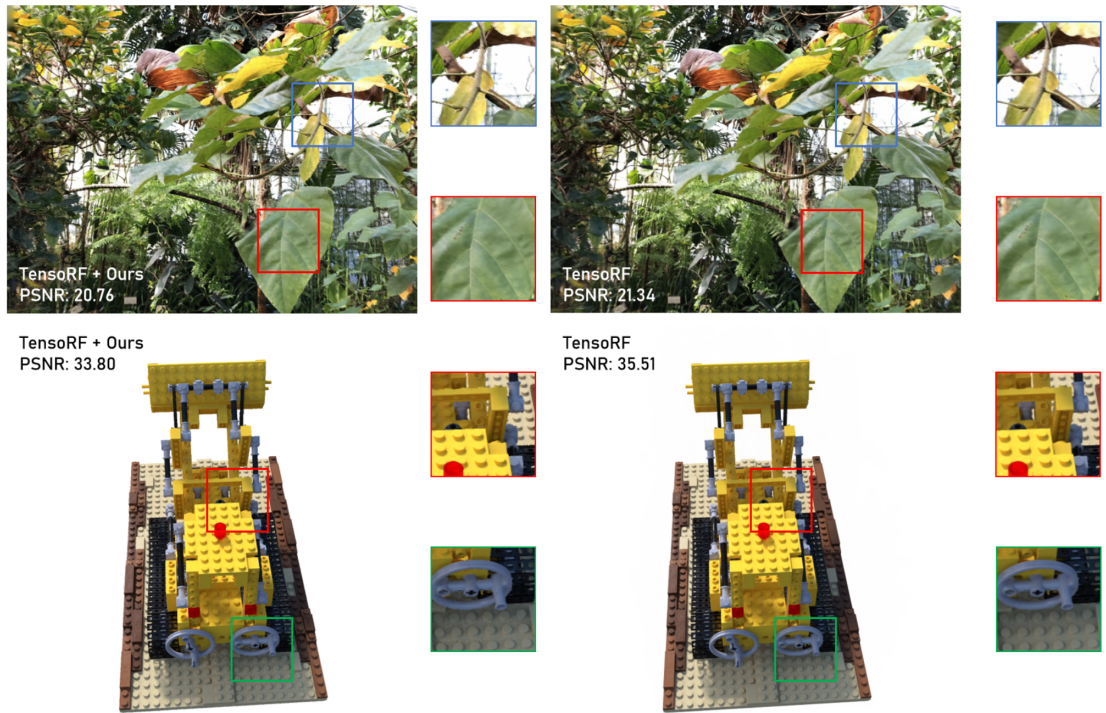


Figure 3.3: Qualitative results on LLFF (top) and NeRF-Synthetic (bottom) datasets. We could tell from the comparisons that the drop in performances has minimal effect on the visual quality.

Datasets and evaluation metrics. We evaluate our method on two standard benchmarks: NeRF-Synthetic and Real Forward Facing Dataset(LLFF) [44] as in [45] with three different models, *i.e.*, Vanilla NeRF [45], TensorRF [12] and InstantNGP [46].

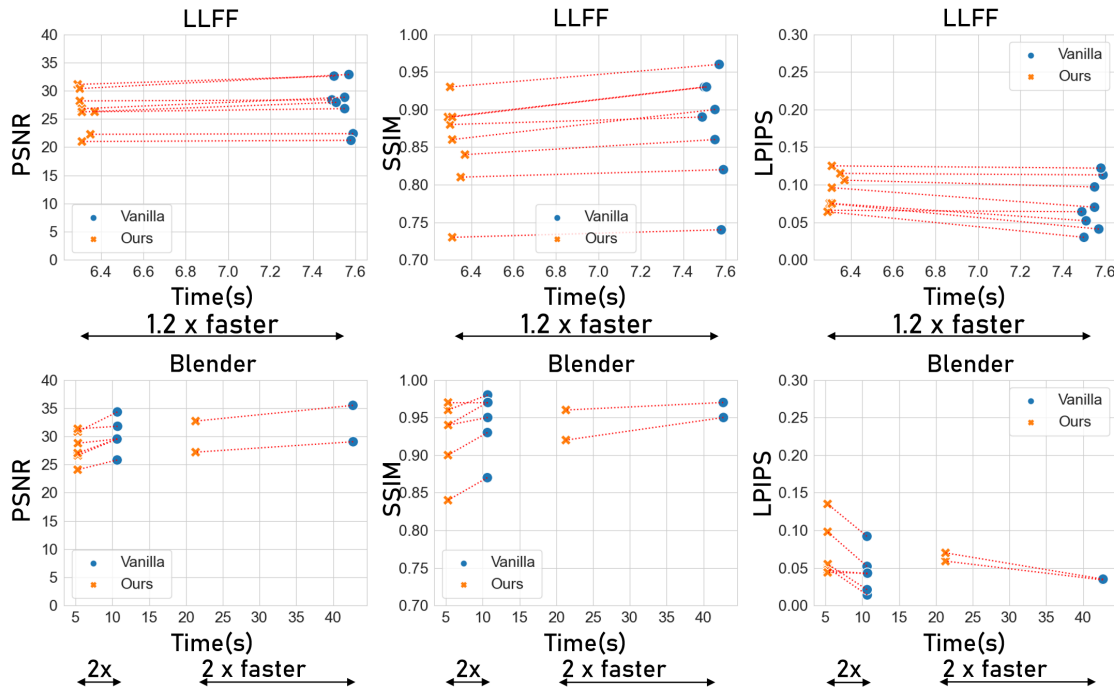


Figure 3.4: Comparison between GL-NeRF and vanilla NeRF in terms of render time and quantitative metrics. Each data point in the visualization represents an individual scene’s performance characteristics. Our results demonstrate that GL-NeRF achieves 1.2 $\times$ to 2 $\times$ speedup in rendering time per image compared to the baseline NeRF implementation, while maintaining comparable visual quality. The performance metrics show minimal degradation despite this substantial reduction in computational cost.

Since our method is training-free, we conduct render-only experiments with the vanilla volume rendering method and our method. We plot the standard render quality evaluation metrics PSNR, SSIM [59] and LPIPS [68] with respect to the average time needed for rendering one image for each scene in Vanilla NeRF. We also report the metrics with averaged color MLP calls for TensorRF and InstantNGP. For Vanilla NeRF, we use 32 points for our method while the network is trained with more than 100 points. For TensorRF and InstantNGP, the results are produced with 4 MLP calls if not otherwise mentioned. More details can be found in Sec. A.1.1.

Dataset	Methods	Avg. MLPs↓	PSNR↑	SSIM↑	LPIPS↓
LLFF	TensoRF	118.51	26.51	0.832	0.135
	ours	4	25.63	0.797	0.146
NeRF-Synthetic	TensoRF	31.08	32.39	0.957	0.032
	ours	4	30.99	0.945	0.048

Table 3.2: Quantitative comparison. We demonstrate that our method has a minimal performance drop while significantly reducing the number of color MLP calls.

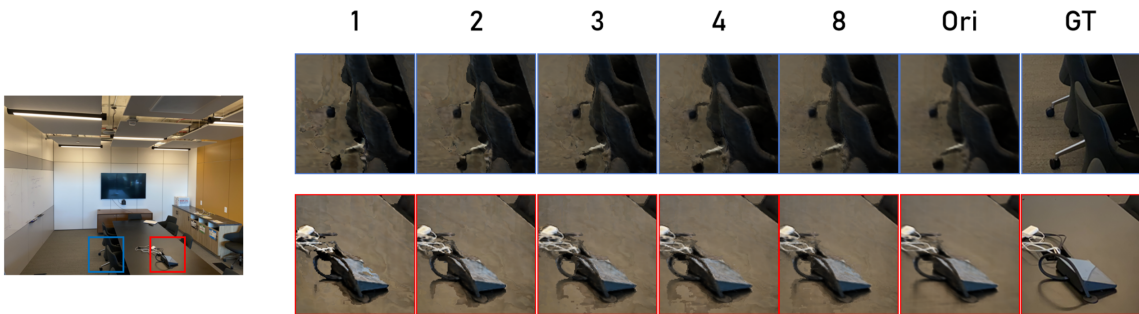


Figure 3.5: Effect of sample number. The first five columns correspond to the number of sampled points on top. Column 6 (Ori) shows TensorRF’s original sampling strategy (100+ points). Column 7 provides ground truth reference. Our method achieves visual fidelity comparable to the original strategy using just 4-8 samples. Note: The blur in columns 1-2 stems from inherent limitations in piecewise-constant density estimation.

Dataset	Methods	PSNR↑	SSIM↑	LPIPS↓
LLFF	Vanilla	27.62	0.88	0.073
	ours	27.21	0.87	0.087
NeRF-Synthetic	Vanilla	30.63	0.95	0.037
	ours	29.18	0.93	0.056

Table 3.3: Quantitative comparison when training with GL-NeRF. Vanilla refers to the vanilla NeRF and its sampling strategy while ours refers to replacing the fine sample stage in vanilla NeRF with our sampling strategy, i.e. GL-NeRF. The result for Vanilla NeRF is produced by rendering using more than 100 points while GL-NeRF only uses 32 points.

Table 3.4: Ablation study on the number of points sampled. The more points we have, the better the performance will be. With 8 points, our method is comparable to the original sampling strategy in TensorRF.

Point number	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
1	23.49	0.752	0.166
2	24.90	0.782	0.142
3	25.38	0.791	0.145
4	25.63	0.797	0.146
8	26.10	0.812	0.142
Ori	26.51	0.832	0.135

3.4.1 Comparison with baselines

Our method enables efficient novel view synthesis using pretrained NeRF models without additional training. Figure 3.4 presents a quantitative comparison between GL-NeRF and original NeRF, demonstrating comparable rendering quality with 1.2–2 $\times$ faster rendering speeds and reduced memory footprint due to fewer MLP evaluations. The minimal quality degradation is visually negligible. Integration with TensorRF [12] (Tab. 3.2) shows significant reduction in MLP calls with only marginal metric decreases. We observe that the minimal drop in the performance has little effect on the quality of the image. Integration with TensorRF [12] (Tab. 3.2) shows significant reduction in MLP calls with only marginal metric decreases. Furthermore, our seamless integration with InstantNGP [46] (Tab. 3.5) demonstrates the method’s plug-and-play capability while maintaining consistent performance on the Blender dataset.

3.4.2 Discussion on acceleration

The speed improvement in Vanilla NeRF fails to achieve real-time performance primarily because of its computationally intensive density estimation network. Our method’s more efficient density requirements are particularly compatible with recent NeRF acceleration approaches such as factorized tensors [12]. Therefore, reducing the number of color MLP calls needed could lead to real-time performance as shown by previous work [24]. We therefore follow MC-NeRF [24] and develop a real-time renderer based on WebGL and train a small variant of TensorRF with 8 channels for each

Blender		Avg.	Chair	Drums	Ficus	Hotdog	Lego	Mat.	Mic	Ship
PSNR $\uparrow$	InstantNGP	32.05	34.13	25.61	31.91	36.32	34.72	29.09	34.92	29.73
	Ours	30.35	33.08	25.07	30.13	34.78	33.05	26.54	33.02	27.15

Table 3.5: Per-scene results on Blender dataset between InstantNGP and ours. We demonstrate that GL-NeRF is able to be plugged into ANY NeRF models.

density component and color component and 32 as hidden size for the color MLP. The result on Lego in the Blender dataset is shown in Tab. 3.6. GL-NeRF is able to provide almost real-time performance in WebGL with similar quality as TensorRF running on an AMD Ryzen 9 5900HS CPU thanks to the reduced number of color MLP calls.

3.4.3 Ablation studies

We further study the effect of sampled points per ray. We conduct experiments using the TensorRF model on the LLFF dataset. We found that 8 points per ray already shows comparable results to the original sampling strategy that uses more than 100 points. Quantitative comparison can be found in Tab. 3.4. Qualitatively, in Fig. 3.5 we found that less number of points would lead to blurrier results. Since the points selected using GL-NeRF intuitively correspond to where the surface is, we argue that the blurriness comes from the inherent inaccuracy of piece-wise constant density estimation.

3.4.4 Discussion on GL-NeRF usage for training

Although our primary focus demonstrates GL-NeRF as a general test-time alternative for volume rendering sampling, the method also proves effective during training. We validate this by substituting Vanilla NeRF’s fine sampling stage with GL-NeRF, with quantitative results presented in Tab. 3.3 and Tab. A.3. Our approach achieves comparable quality to the original sampling strategy while using significantly fewer sample points, *i.e.*, 32, for GL-NeRF versus over 100 for Vanilla NeRF, demonstrating its efficiency throughout both training and inference phases.

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	FPS $\uparrow$
TensoRF	33.28	0.97	0.016	5.84
ours	33.09	0.97	0.016	22.34

Table 3.6: Comparison between our method and TensoRF on Lego scene using WebGL-based renderer. The result is collected from an AMD Ryzen 9 5900HS CPU. GL-NeRF is able to provide almost real-time rendering while remaining similar quality as TensoRF.

3.5 Conclusion

For efficient geometry representation, we propose GL-NeRF, a novel approach for calculating the volume rendering integral. We show that with a simple change of variable, the Gauss-Laguerre quadrature can be used for computing the volume rendering integral. The highest algebraic precision of the Gauss-Laguerre quadrature allows a dramatic reduction in the required MLP evaluations during rendering. Such a formulation enables us to query geometry from the learned 3D representation, *i.e.*, NeRF, in an efficient way. We provide both theoretical justification and empirical validation for our quadrature choice, demonstrating GL-NeRF’s plug-and-play compatibility across multiple NeRF architectures. Experimental results reveal GL-NeRF’s potential as a universal accelerator for existing NeRF models. Furthermore, we show successful integration during NeRF training, suggesting new research directions in efficient neural rendering and geometry representations.

3. *Efficient Geometry Representation*

Chapter 4

Accurate Appearance Modeling

4.1 Motivation

In addition to geometry, appearance is another crucial aspect when developing representations for 3D scenes. Inspired by previous success in neural scene representations and volume rendering, research has been focusing on modifying such representations for modeling different materials by extending to physically-based rendering [30] and different network structures and representations [31, 36, 62, 70]. It is also not surprised to see that 3D Gaussian Splatting (3DGS) [32], a much more efficient approach for neural scene representations, has been adopted for modeling materials, namely the inverse rendering task [19, 28, 37]. Due to the discretized nature of 3DGS, all the parameters are independent to each other in the formulation, *e.g.*, opacity, mean position, covariance, material attributes are all independent values. While it is reasonable to have standalone mean and covariance, it is not clear whether having independent opacity and materials is an optimal choice. We therefore study the problem whether we can have a more accurate appearance modeling. We draw our inspiration from a well-known physics law that will be discussed Sec. 4.2. Deriving from this law, we observed that it is better to have an entangled representations of opacity and material attributes, which obeys the underlying physics law [7, 11, 35, 43].

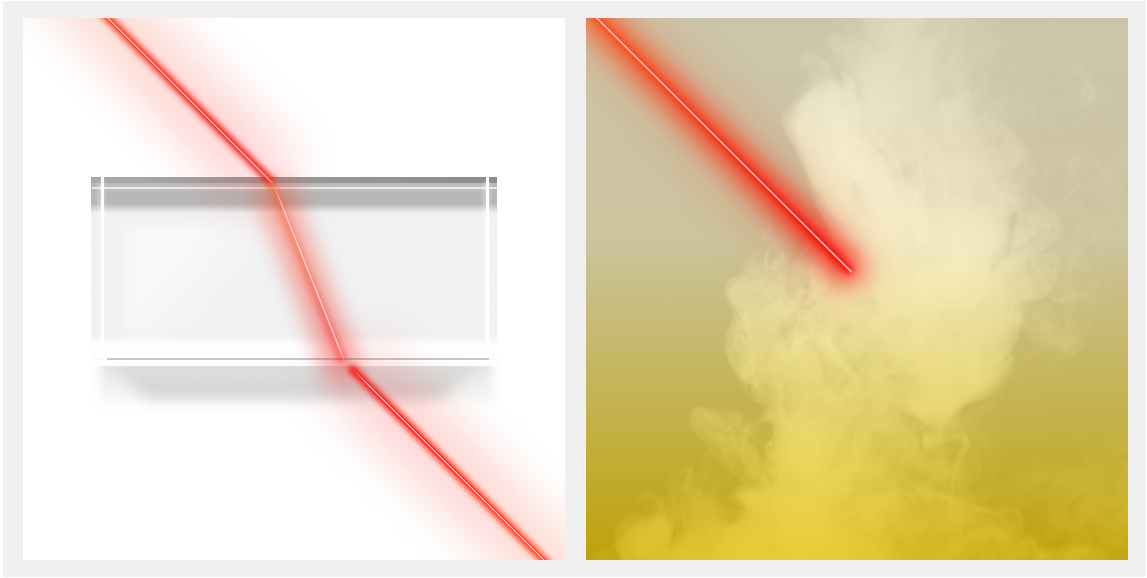


Figure 4.1: This illustration demonstrates the material-dependent nature of opacity through a comparative study of two translucent media. **Left:** A glass exhibits minimal light attenuation, transmitting incident radiation with negligible intensity reduction. **Right:** In contrast, the same incident light undergoes complete absorption within the gaseous medium. This fundamental difference motivates our key insight: each 3D Gaussian primitive, when associated with material properties, should be modeled as an absorbing medium with a material-aware opacity coefficient. We therefore propose to explicitly incorporate material properties as governing factors in the absorption characteristics of Gaussian primitive, establishing a physically-grounded relationship between material and opacity.

4.2 The Bouguer-Beer-Lambert Law

According to the Bouguer-Beer-Lambert law [7, 11, 35, 43], when radiation (such as light) with an initial intensity I traverses an absorbing medium, its intensity variation is described by the differential equation:

$$dI_\nu = -\alpha_\nu I_\nu ds, \quad (4.1)$$

where subscript ν means that the term depends on the frequency of light and α_ν is called the extinction coefficient, a property that governs the attenuation of energy within the absorbing medium.

Solving for intensity I_ν , we get

$$I_\nu(s) = I_\nu(0)e^{-\alpha_\nu s}, \quad (4.2)$$

here, $e^{-\alpha_\nu s}$ is the transmittance of region s , indicating that the light intensity decays exponentially as it propagates through the absorbing body. When light attenuation in the medium occurs purely through absorption (without scattering effects), the extinction coefficient can be calculated as

$$\alpha_\nu = n\sigma_\nu, \quad (4.3)$$

where n denotes the number density and σ_ν represents the cross section. These parameters are commonly used in chemical contexts: n quantifies the concentration of discrete entities (typically particles), and σ_ν models the probability that two particles will collide, which is determined by the types of the particles. Fig. 4.2 demonstrates the Bouguer-Beer-Lambert law, showing how light intensity diminishes as it traverses a particle-filled medium through interactions with particles (where the effective interaction area typically exceeds the physical particle size). Essentially, the reduction in intensity is determined by the number of particles in the region that the light hits and the affecting area of the specific particle type, which corresponds to Eq. 4.3 and Eq. 4.2.

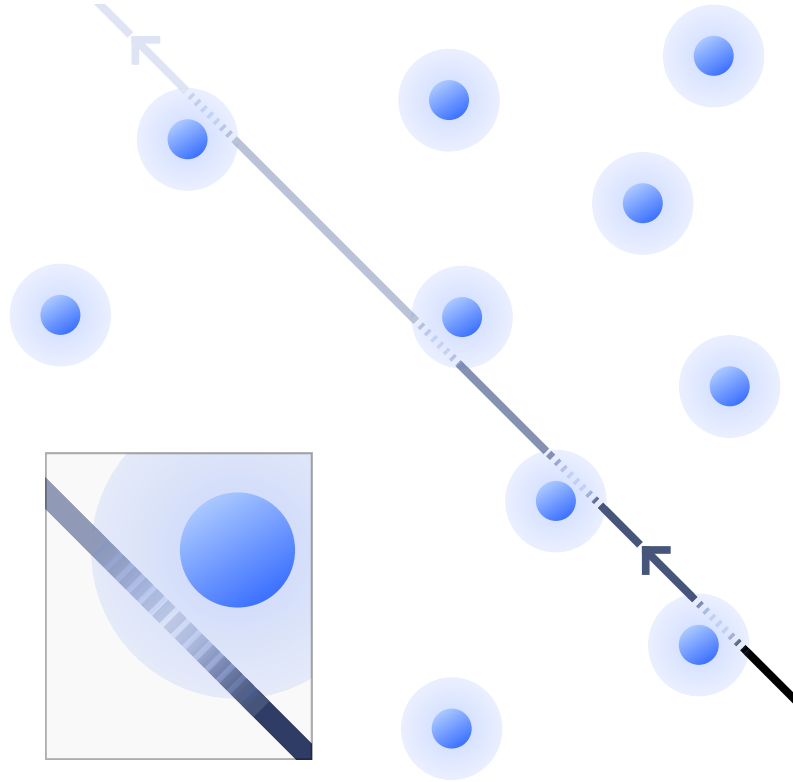


Figure 4.2: An illustration of the Bouguer-Beer-Lambert law. Consider a ray passing through an absorbing body consists of particles of some type. The cross section depends on the area that each particle would affect (usually bigger than the size of the particle and equals the size of the particle if treated as a hard sphere). When traveling, the intensity of the light would be reduced by the areas that each particle affected. The reduction in intensity is therefore affected by the area around the particles and the number of particles, corresponding to cross section σ and number density n in the main paper.

4.3 OMG: Opacity Modeling in Gaussian Splatting using the Bouguer-Beer-Lambert Law

We start by introducing the intuition behind our method. It is well-known that different materials react differently when hit by a beam of light, for instance, wood completely blocks and scatters incident light, whereas glass permits nearly unattenuated transmission. This fundamental behavior is quantitatively characterized by the Bouguer-Beer-Lambert law introduced earlier, which establishes a material-dependent relationship for optical opacity. Building upon this physical principle, we develop a theoretical framework that leads to our proposed modular solution: opacity should be modeled as a material-specific property, with transmittance calculations adhering to the exponential decay formulation prescribed by the Bouguer-Beer-Lambert law.

4.3.1 Physically Correct Activation

Our method is fundamentally grounded in the Bouguer-Beer-Lambert law. As defined in Eq. 2.9, T_i represents the cumulative transmittance through the first $(i - 1)$ Gaussians, which quantifies the attenuation of light reaching the i -th Gaussian from the viewpoint. For an individual Gaussian G_i , its transmittance is computed as $1 - \alpha_i(x)$. However, the Bouguer-Beer-Lambert law dictates that transmittance through an absorbing medium must exhibit the exponential form $e^{-\alpha_\nu s}$ or equivalently $e^{-n\sigma_\nu s}$ (Eq. 4.2). Therefore we have

$$\begin{aligned} 1 - \alpha_i(x) &= e^{-n\sigma_\nu s}, \\ \alpha_i(x) &= 1 - e^{-n\sigma_\nu s}. \end{aligned} \tag{4.4}$$

A natural question then arises: **what is n , σ_ν and s in the context of Gaussian Splatting?**

To address this, we revisit the definition of $\alpha_i(x)$ in Eq. 2.10. The coefficient α_i quantifies the Gaussian’s density concentration, while the exponential component dictates a radial density profile - maximized at the center and decaying outward as visualized in Fig. 4.3. Since each Gaussian blob is not solid, it is natural to think of it

4. Accurate Appearance Modeling

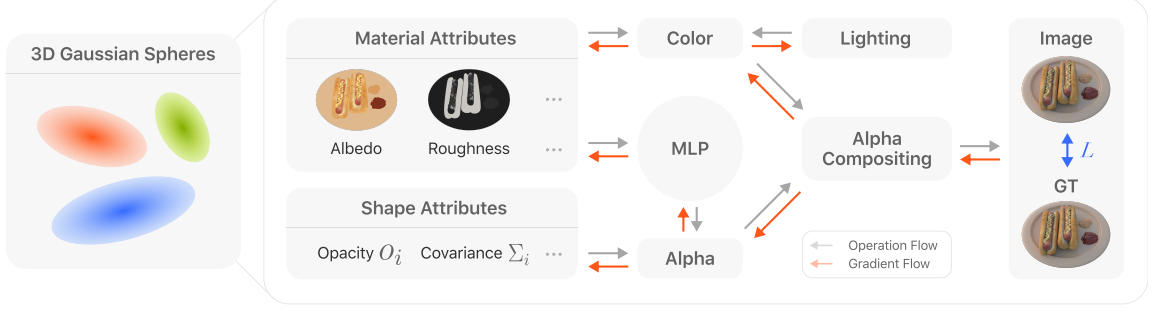


Figure 4.3: **Pipeline Overview.** Our method extends existing 3DGS-based inverse rendering approaches [19, 28, 32, 37] through a novel architectural modification. Departing from prior works that treat opacity as an independent parameter, we condition it on material properties. Specifically, we introduce a neural network that takes material attributes as input and outputs cross-section values, which are then multiplied with the opacity to integrate the Bouguer-Beer-Lambert law into the framework. During optimization, material properties receive dual gradient signals: (1) color-derived gradients via differentiable PBR and (2) alpha-derived gradients enforced by our network. This dual-path supervision imposes an additional physical constraint, ensuring strict adherence to the Bouguer-Beer-Lambert law throughout the rendering pipeline.

as a blob of gas with its density following a Gaussian distribution. Notice that in this formulation, the density corresponds to the number density n in Eq. 4.4. Formally, we have

$$n_i = o_i \exp\left(-\frac{1}{2}(x - R(\mu_i; \theta))^T R_\theta(\Sigma_i)^{-1}(x - R(\mu_i; \theta))\right). \quad (4.5)$$

Regarding the path length s , we note that each Gaussian is projected onto a 2D plane through splatting, i.e. density is marginalized along the viewing axis, the distance of the light passing through each Gaussian can be set as a constant 1. In fact, the blending algorithm omits the depth of the Gaussians into the formulation of the opacity value at the pixel location. We therefore observed that we can treat this opacity value as the density and since there is no concept of depth on the 2D plane, each Gaussian is then treated equally from this perspective, therefore it is reasonable to treat the path length to constant 1.

4.3.2 Neural Cross Section Modeling

After deciding on the correspondence of existing terms, we found that there is still one term missing from the Bouguer-Beer-Lambert law, namely the cross section σ_ν . Cross section, by definition, is determined by the type of particle that fills the region as illustrated in Fig. 4.2, namely the material of the region. On a macro level, cross section affects the color that one can perceive from the region. Since in inverse rendering, material properties, i.e., albedo, metalness, and roughness, are explicitly included in the model (for 3DGS-based method they're set as additional parameters for each Gaussian), we can directly use the parameter for representing the cross section. In practice, a neural network $f(\cdot)$ that takes as input the material properties m and outputs cross section σ_ν is introduced, namely

$$\sigma_\nu = f(m). \quad (4.6)$$

Our derivation has established all three components required by the Bouguer-Beer-Lambert law within the Gaussian Splatting framework. This enables us to reformulate $\alpha_i(x)$ in Eq. 2.10 differently as

$$\begin{aligned} \alpha_i(x) &= 1 - e^{-n_i(x)\sigma_{\nu i}} \\ &= 1 - e^{-o_i G_i(x) f(m_i)}, \end{aligned} \quad (4.7)$$

where o_i is the original opacity term for the i -th Gaussian, $G_i(x)$ is the normal distribution in Eq. 2.10, f is the neural network that produces cross section and m_i is the material property.

4.3.3 Gradient Flow Analysis

Consider c_i in Eq. 2.9, in inverse rendering: it is computed using the material properties in Eq. 2.6. Therefore, our formulation of material takes gradients not only from the color term itself, but also from the α term. In this case,

$$\frac{\partial L}{\partial m_i} = \frac{\partial c_i}{\partial m_i} \frac{\partial L}{\partial C} \alpha_i T_i + c_i \frac{\partial L}{\partial C} T_i \frac{\partial \alpha_i}{\partial m_i} - \sum_{j=i+1}^N c_j \frac{\partial L}{\partial C} T_j \alpha_j \frac{1}{1 - \alpha_i} \frac{\partial \alpha_i}{\partial m_i}, \quad (4.8)$$

where L stands for loss. Our formulation augments the gradient for material learning, allowing the gradient to flow from opacity to serve as an additional constraint in a physics-inspired way, as illustrated by Fig. 4.3. Consequently, the material representation becomes optimized under both photometric and physical constraints.

4.3.4 Analysis

Beyond its derivation from the Bouguer-Beer-Lambert law, our formulation possesses two independent interpretations that mutually reinforce its validity.

NeRF. The alpha-blending algorithm in NeRF [45] takes the form

$$\begin{aligned} C(r) &= \sum_{i=1}^N c_i (1 - \exp(-\sigma_i \delta_i)) T_i, \\ T_i &= \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \end{aligned} \tag{4.9}$$

in which r stands for the ray direction, c_i is a view-dependent color of the sampled points, σ_i is volume density and δ_i is the distance between two sampled points. This formulation yields $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$, precisely matching our Bouguer-Beer-Lambert law derivation. Notably, neural field approaches [9, 10] employing particle-style ray marching implicitly model cross section through shared MLP representations of density and material. The comparison reveals that explicit cross section incorporation becomes essential when transitioning from neural field representations to discrete Gaussian representations and suggests the necessity of introducing cross section into the **disentangled** representation of 3DGS-based methods.

Taylor Expansion. Through Taylor expansion w.r.t. the function $\alpha(t) = 1 - \exp(-t)$, we have

$$\begin{aligned} \alpha(t) &= 1 - e^{-t} \\ &= 1 - (1 - t + o(t)) \\ &= t + o(t), \end{aligned} \tag{4.10}$$

where $o(t)$ denotes the little-o notation. This relationship reveals that Eq. 2.10 serves as a first-order approximation to the exact form $\alpha_i(x) = 1 - \exp(-o_i G_i(x))$, which shares identical structure with Eq. 4.7 when excluding the cross section term $f(m_i)$. The mathematical consistency between these formulations provides rigorous justification for our novel activation function.

4.4 Results

We discuss the implementation details to show the experimental design and present experimental results on synthetic and real-world data to provide a comprehensive understanding of our method.

4.4.1 Implementation Details

The model-agnostic design of our method facilitates direct integration with state-of-the-art inverse rendering frameworks. We validate this through implementation with three representative baselines: GaussianShader [28], GS-IR [37], and R3DG [19]. All the experiments are conducted on a single NVIDIA RTX 6000 Ada Generation GPU and we report the reproduced baseline results for fair comparison. We use an MLP that has 2 hidden layers with size 128 for the hidden dimension and ReLU as activation function for cross section and the output activation is a sigmoid function. The input to the MLP is designed as follows: for GaussianShader, an end-to-end trained material modeling pipeline, we directly input the albedo, roughness and specular tint terms as in their original model for MLP input. For GS-IR and R3DG, since they follow a two-stage training pipeline, in which the first stage uses normal rendering for geometry reconstruction and second stage uses physically-based rendering for material modeling, we input the material properties as well as the SH coefficients used for color in the first stage to the MLP. The logic behind this design is to make the optimization of the MLP in the first stage meaningful. We follow the experimental settings as provided by the original baselines. GaussianShader is evaluated on Shiny Blender [56] and Glossy Synthetic [39] as in the original work. For R3DG, Synthetic4Relight dataset [70] is used for getting material modeling evaluation results. GS-IR is evaluated on MIP-NeRF 360 dataset [5] to provide insights on real-world data. For albedo and relighting evaluation, we follow

4. Accurate Appearance Modeling

the standard practice to standardize the albedo prediction against the ground truth.

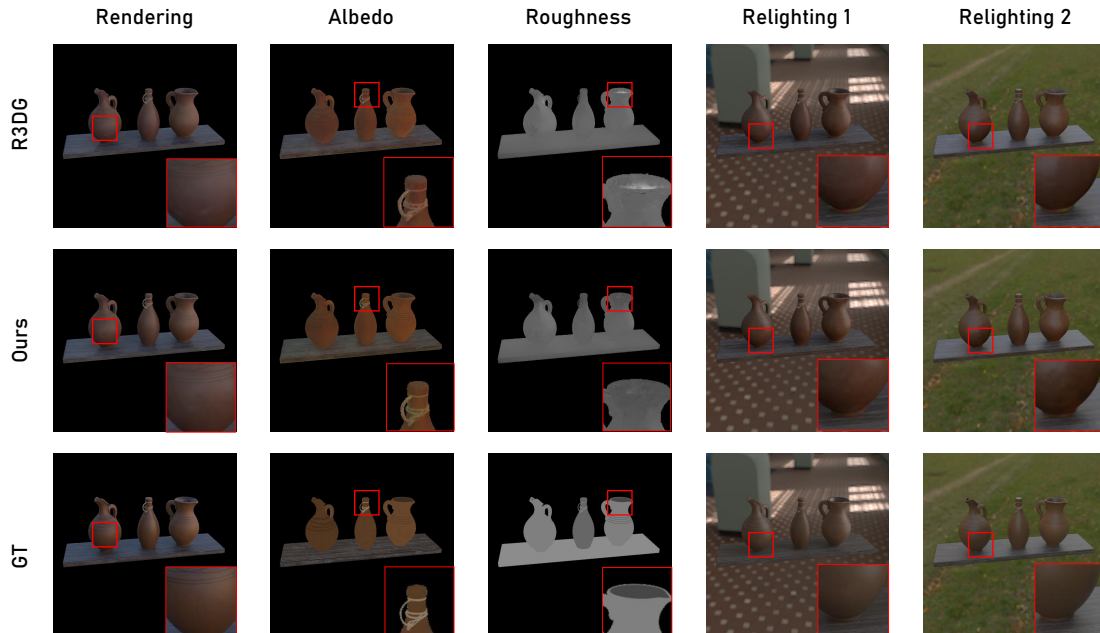


Figure 4.4: Qualitative comparison of albedo, relighting and roughness on Synthetic4Relight dataset [70]. Our method is able to decouple specular effects from albedo compared to R3DG, the baseline we implemented our method on. It is also capable of providing roughness estimation more precisely. We add a gamma correction to the relighting results to make the visual differences clearer.

4.4.2 Results on Synthetic Data

Material Modeling. We first evaluate our proposed module on top of R3DG [19] for material modeling evaluation. As can be seen from Tab. 4.1, with the help of our modeling, R3DG is able to produce much better results universally. Our method outperforms the baseline on novel view synthesis by 0.4db PSNR, as well as on two other metrics. The relighting results share similar performance boosts as novel view synthesis, which could be attributed to the significant improvements in albedo (~ 0.6 db PSNR boost) and roughness estimation. Qualitatively speaking, as can be seen from the visualization in Fig. 4.4, our method is capable of producing albedo estimation with less specular effects and roughness estimation that aligns better with

Table 4.1: Quantitative results on Synthetic4Relight dataset [70]. Our method is able to provide significant improvements universally. The improvements in novel view synthesis and relighting can be attributed to the improved estimation of albedo and roughness thanks to the correct modeling we use for constraining material properties.

	Novel View Synthesis			Relighting			Albedo			Roughness
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	MSE $\downarrow$
R3DG	34.07	0.975	0.047	32.87	0.967	0.052	28.70	0.952	0.064	0.011
R3DG + Ours	34.47	0.977	0.046	33.21	0.969	0.050	29.31	0.954	0.059	0.007

Table 4.2: Quantitative results when combined with GaussianShader [28]. We compare with GaussianShader on Shiny Blender [56] and Glossy Synthetic datasets [39]. Our method is able to outperform GaussianShader on novel view synthesis in terms of all three standard image quality metrics.

	Shiny Blender			Glossy Synthetic		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
GaussianShader	30.77	0.953	0.081	27.25	0.925	0.083
GaussianShader + Ours	31.07	0.957	0.076	27.68	0.929	0.081

the ground truth, resulting in pleasing relighting results.

Novel View Synthesis on Synthetic Data To further validate our approach, we integrated our method with GaussianShader [28], enabling comprehensive evaluation on additional synthetic datasets. Tab. 4.2 showcases the result on Shiny Blender [56] and Glossy Synthetic [39]. Our method is able to produce significant improvement in terms of novel view synthesis when built on top of GaussianShader (specifically ~ 0.3 db PSNR boost on Shiny Blender and ~ 0.4 db boost on Glossy Synthetic). Fig. 4.5 showcases some qualitative comparisons of our model with the GaussianShader baseline. Remarkably, without explicit normal constraints, our method achieves smoother normal estimation compared to the baseline as well as better modeling of light reflection.

4.4.3 Results on Real-World Data

Apart from the results on synthetic data, we also analyze the effect of our method when applied to real-world data. To verify the plug-and-play attribute of our method, we apply it to GS-IR [37] and conduct experiments on the MIP-NeRF 360 dataset [5].

4. Accurate Appearance Modeling

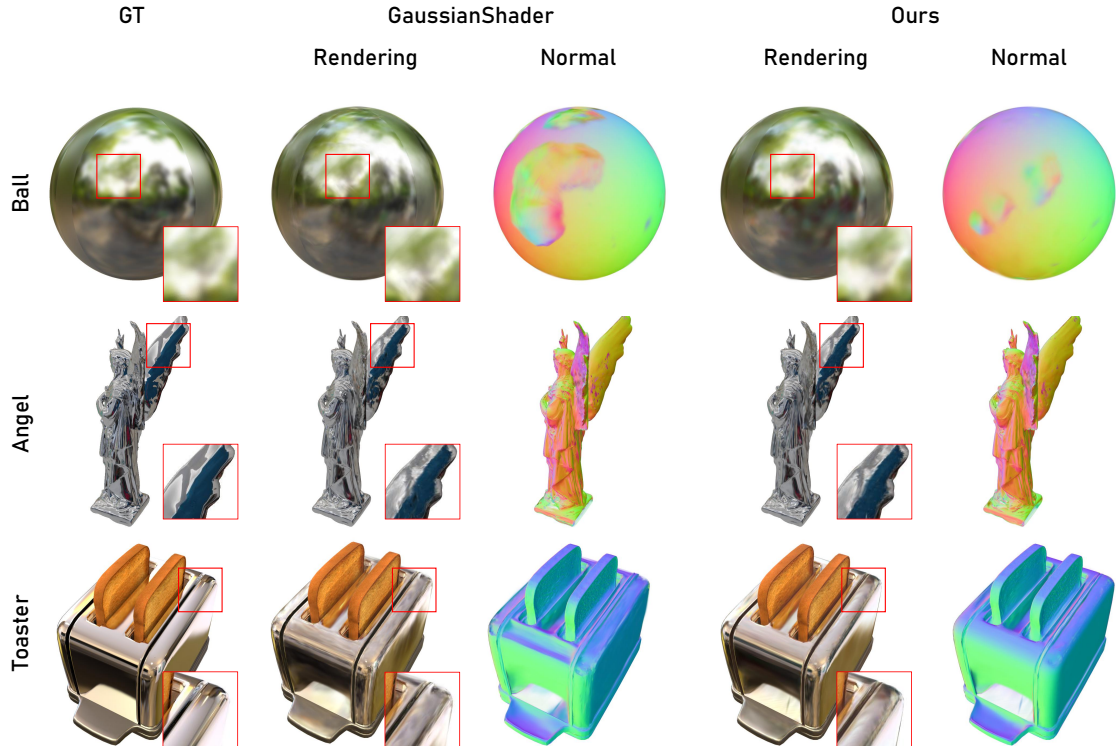


Figure 4.5: **Qualitative comparison on Shiny Blender [56] and Glossy Synthetic [39] datasets.** Our method is able to provide satisfactory normal estimation compared to the baseline. Notice that our method does not include any additional supervision or constraints for normal estimation. Our method is also able to provide much more accurate view synthesis results when doing physically-based rendering. These results suggest the significance of introducing the physically correct model for inverse rendering.

The experimental results shown in Tab. 4.3 indicate a significant universal improvement in terms of novel view synthesis thanks to the correct modeling and better estimation of materials as indicated by the results in Sec. 4.4.2. Qualitatively speaking, our method is able to produce less blurry images compared to the GS-IR baseline as shown by the bicycle scene in the first row in Fig. 4.6. This real-world performance confirms our approach’s plug-and-play adaptability and the importance of physical constraints across data domains, which imposes more accurate material modeling.

Analysis. Since our method augments opacity with a material-dependent term, it turns the opacity from a pure geometry indicator to a combined representative of

geometry and material. That being said, the optimization of geometry is also affected by our method. This is obvious if we consider the gradient of α_i w.r.t. o_i :

$$\frac{\partial \alpha_i}{\partial o_i} = G_i f(m_i)(1 - \alpha_i), \quad (4.11)$$

pixel location x is omitted for simplicity. We hypothesize that since MIP-NeRF 360 supports SfM initialization, it helps our method by easing the burden for geometry reconstruction, therefore leading to more improvements compared to synthetic data which uses random initialization.

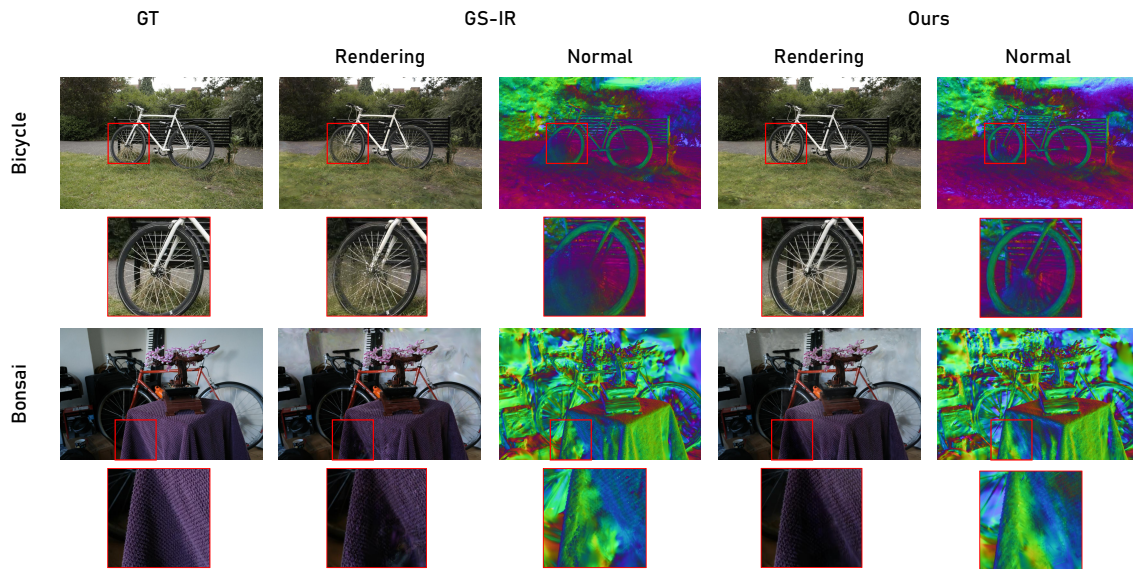


Figure 4.6: **Qualitative Comparison of GS-IR on MIP-NeRF 360** [5]. Our method is able to provide much more accurate novel view synthesis results compared to the baseline GS-IR [37] in which we plugged our model. The GS-IR method produces a blurred prediction in the Bicycle scene (first row), which our approach successfully resolves. Additionally, as demonstrated in the second row, our method achieves significantly more precise and detailed normal estimation.

4.5 Conclusion

In this chapter, we study the problem of how to have a more accurate representation for appearances of the underlying scenes, namely, how to have an accurate modeling

Mip-NeRF 360		Avg.	Bicycle	Flowers	Garden	Stump	Treehill	Room	Counter	Kitchen	Bonsai
PSNR $\uparrow$	GS-IR	25.38	23.31	20.43	25.76	25.46	21.75	28.32	26.31	28.66	28.44
	Ours	25.85	24.09	20.16	26.24	25.54	21.97	29.64	26.85	28.92	29.27
SSIM $\uparrow$	GS-IR	0.760	0.688	0.542	0.808	0.717	0.583	0.874	0.847	0.877	0.901
	Ours	0.767	0.720	0.510	0.824	0.720	0.586	0.897	0.859	0.884	0.905
LPIPS $\downarrow$	GS-IR	0.265	0.269	0.368	0.154	0.256	0.367	0.281	0.255	0.179	0.254
	Ours	0.254	0.230	0.401	0.140	0.245	0.368	0.247	0.244	0.168	0.243

Table 4.3: **Quantitative Comparison on Mip-NeRF 360** [5]. The results show that our approach, when combined with GS-IR, can significantly improve the novel view synthesis results of real-world scenes on MIP-NeRF 360 [5]. Thanks to the constraint we add to opacity, our method is able to produce ~ 0.5 db PSNR gains along with SSIM and LPIPS improvements.

of the materials. We introduce **OMG**, a plug-and-play module designed for material modeling within the Gaussian Splatting framework for better appearance modeling. Drawing inspiration from the Bouguer-Beer-Lambert law, we mathematically derive the correct opacity formulation for particle-based models. We compare our proposed formulation with NeRF-based approaches and provide theoretical analysis, offering deeper understanding of inverse rendering modeling. Our analysis further highlights the underlying connections between NeRF-based techniques and 3D Gaussian Splatting.

To demonstrate the versatility and effectiveness of our modeling approach, we integrate it with three different state-of-the-art baseline methods and perform extensive experiments on both synthetic and real-world datasets. The substantial performance gains observed in these experiments confirm the validity of our modeling approach and emphasize the importance of proper priors and physically accurate modeling in visual computing. We hope this chapter encourages the research community to focus more on developing physically grounded models and incorporating appropriate priors for 3D world representation.

Chapter 5

Conclusions

5.1 Summary

This thesis steps towards solving the problem of how to develop a more efficient and accurate scene representation in terms of geometry and appearances. The methods proposed and discussed in Chapter 3 and Chapter 4 mainly considers methods that are derived from a fundamental math and physics perspective, making them inherently generalizable. The method discussed for efficient geometry representation, *i.e.*, GL-NeRF, in Chapter 3, is derived from a pure math perspective. It studies the problem of how to utilize existing knowledge stored in a learned representation in an efficient way, making it possible to distill information from the learned representation. while GL-NeRF mainly focuses on extracting geometry information from the learned representations, in Chapter 4 we cover another crucial aspect for world representation learning, namely appearance. The proposed method, OMG, stems from an important observation that the base model, 3D Gaussian Splatting, relies on disentangled representation, whereas appearances from a physics perspective are entangled and more precise. Therefore, starting from the Bouguer-Beer-Lambert law, we revisit the formulation of such representation and derived a better one that strictly follows the physical law. It's also generalizable to all the variants of 3D Gaussian Splatting.

In conclusion, we hope this thesis can serve as a foundation for further research in how to build a generalizable world representation with efficiency and accuracy.

5.2 Discussion

Correct prior in modeling. Both Chapter 3 and Chapter 4 emphasize the importance of introducing correct priors into the modeling itself. Despite the dominance of scaling law where a large amount of data seems to solve it all, adding correct priors into the modeling and representation learning stage is always desired as the amount of data might come to a limit.

Disentangled representation. While it is common to use disentangled representations for machine learning for efficiency and handy purposes, it is not always the best. This thesis steps towards understanding when disentanglement matters. The hypothesis derived is that semi-explicit dependency should be modeled, whereas implicit dependency can be ignored and modeled as independent representations.

5.3 Limitations

In Chapter 3, although GL-NeRF demonstrates significant reductions in MLP evaluations while maintaining reasonable rendering quality, our experiments reveal a persistent gap between theoretical guarantees and empirical results. This discrepancy suggests several promising research directions:

1. **Density Field Regularization:** The polynomial approximation error may be reduced by enforcing smoother density distributions during training. Also, a better density estimation could lead to better point positions.
2. **Adaptive Quadrature:** A hybrid approach could combine GL-NeRF’s efficiency with importance sampling, dynamically adjusting point distributions based on local density variations.

These potential improvements aim to better align the practical performance with the theoretical precision guarantees of Gauss-Laguerre quadrature.

In Chapter 4, the current formulation does not account for the frequency-dependent nature of cross sections in light interaction. A promising direction for future research would be to incorporate this spectral relationship to achieve more physically accurate world modeling. As demonstrated in Sec. A.2.4 and Fig. A.6, the currently learned

cross sections show limitations when handling specular surfaces under intense illumination. These observations motivate further investigation into developing better regularization strategies for material and cross-section estimation to achieve higher precision in the learning process.

5.4 Open Directions

In addition to the above-mentioned directions that aim to close the gap between theoretical results and practical observations, how to bridge efficiency and accuracy together is a big direction that is interesting and should be explored. Eventually, we want to build a generalizable representation that accurately and efficiently reflects the real world. A straightforward problem from this thesis is how to combine the Gauss-Laguerre quadrature in Chapter 3 with the 3D Gaussian Splatting representation in Chapter 4 for an efficient and accurate representation of geometry and appearance.

5. *Conclusions*

Appendix A

Appendix

A.1 GL-NeRF

Here, we introduce the implementation details, give a more thorough introduction of the Gauss-Laguerre quadrature and the full quantitative results on NeRF-Synthetic and LLFF datasets.

A.1.1 Implementation details

For Vanilla NeRF, our experiments are conducted based upon NeRF-PyTorch [64], a reproducible PyTorch implementation of the original NeRF [45]. We implement our method by changing the hierarchical sampling strategy into our point selection method using the Gauss-Laguerre quadrature. We follow the standard setting as done in [45] to train a “coarse” and a “fine” network for evaluation. We use a learning rate of 5×10^{-4} that exponentially decays to 5×10^{-5} over the course of optimization. Each scene is trained for 200k iterations using a single NVIDIA RTX 6000 GPU. We use 128 coarse samples and 32 fine samples to test our method. Since the density estimation for “coarse” and “fine” network are not aligned, we test our method using only the “fine” network by first using “coarse” samples to query it for an estimation of density, then use GL-NeRF to select 32 points for final rendering as discussed in Sec. 3.3.1. LLFF scenes are trained and tested with 64 coarse samples and 64 fine samples for baseline method, while NeRF-Synthetic scenes require 128 coarse samples

and 64 fine samples. For TensorRF, we directly use the pretrained checkpoints in the folder VM48 provided by the authors. The qualitative results are produced by 4 neural network calls if not otherwise mentioned. For InstantNGP, we build our code on top of the public PyTorch implementation [23] and train it with the default setting. The final results for GL-NeRF are also produced by 4 neural network calls.

A.1.2 Gauss-Laguerre quadrature

The Gauss-Laguerre quadrature is an approximation formula for computing integrals over the semi-infinite interval $[0, +\infty)$ with the weight function e^{-x} and reads as

$$\int_0^{+\infty} e^{-x} f(x) dx \approx \sum_{k=0}^n w_k f(x_k). \quad (\text{A.1})$$

Here $x_0, x_1, \dots, x_n \in [0, +\infty)$ are the zeros of the Laguerre polynomial $L_{n+1} = L_{n+1}(x)$ of degree $(n+1)$:

$$L_{n+1}(x) = \frac{1}{(n+1)!} e^x \frac{d^{n+1}}{dx^{n+1}} (x^{n+1} e^{-x}),$$

for $n = -1, 0, 1, \dots$, and the coefficients

$$w_k = \frac{1}{x_k [L'_{n+1}(x_k)]^2}, \quad k = 0, 1, 2, \dots, n. \quad (\text{A.2})$$

From the Leibniz formula, it is easy to see that $L_n(x)$ is a polynomial of degree n and the coefficient of x^n is $\frac{(-1)^n}{n!}$. In particular, we have

$$L_0 = 1, \quad L_1 = 1 - x, \quad L_2 = \frac{1}{2}x^2 - 2x + 1, \dots$$

The fundamental property of the Laguerre polynomials is

Theorem 2. *The Laguerre polynomials $L_n = L_n(x)$ are orthogonal with respect to the weight function e^{-x} , that is,*

$$\int_0^{+\infty} e^{-x} L_n(x) L_m(x) dx = \begin{cases} 0, & n \neq m, \\ 1, & n = m. \end{cases}$$

Proof. Assume $m \leq n$ and set $g_k(x) = x^k e^{-x}$. From the Leibniz formula it follows that, for $j < k$, $g_k^{(j)}(x)$ is a product of $x e^{-x}$ and a polynomial of degree $(k-1)$ and thereby $g_k^{(j)}(0) = 0 = g_k^{(j)}(+\infty)$ for $j < k$. Thus, we deduce that

$$\begin{aligned}
& n!m! \int_0^{+\infty} e^{-x} L_n(x) L_m(x) dx \\
&= \int_0^{+\infty} e^{-x} e^x g_n^{(n)}(x) e^x g_m^{(m)}(x) dx \\
&= \int_0^{+\infty} e^x g_m^{(m)}(x) d g_n^{(n-1)}(x) \\
&= g_n^{(n-1)}(x) [e^x g_m^{(m)}(x)]|_0^{+\infty} \\
&\quad - \int_0^{+\infty} [e^x g_m^{(m)}(x)]' d g_n^{(n-2)}(x) \\
&= -g_n^{(n-2)}(x) [e^x g_m^{(m)}(x)]'|_0^{+\infty} \\
&\quad + \int_0^{+\infty} [e^x g_m^{(m)}(x)]'' d g_n^{(n-3)}(x) \\
&= \dots\dots\dots \\
&= (-1)^n \int_0^{+\infty} g_n(x) [e^x g_m^{(m)}(x)]^{(n)}(x) dx
\end{aligned} \tag{A.3}$$

By the Leibniz formula, we have

$$\begin{aligned}
& [e^x g_m^{(m)}(x)]^{(n)} = \sum_{j=0}^n \frac{n!}{(n-j)!j!} (e^x)^{(n-j)} g_m^{(m+j)}(x) \\
&= e^x \sum_{j=0}^n \frac{n!}{(n-j)!j!} g_m^{(m+j)}(x)
\end{aligned}$$

and thereby

$$\begin{aligned}
& n!m! \int_0^{+\infty} e^{-x} L_n(x) L_m(x) dx \\
&= (-1)^n \sum_{j=0}^n \frac{n!}{(n-j)!j!} \int_0^{+\infty} x^n g_m^{(m+j)}(x) dx \\
&= \sum_{j=0}^{n-m} \frac{n!(-1)^{n+m+j}}{(n-j)!j!} \int_0^{+\infty} \frac{n!x^{n-m-j}}{(n-m-j)!} g_m(x) dx \\
&= \sum_{j=0}^{n-m} \frac{n!(-1)^{n+m+j}}{(n-j)!j!} \int_0^{+\infty} \frac{n!}{(n-m-j)!} x^{n-j} e^{-x} dx \\
&= (-1)^n \sum_{j=0}^{n-m} \frac{n!}{(n-j)!j!} (-1)^{m+j} \frac{n!(n-j)!}{(n-m-j)!} \\
&= (-1)^{n+m} \frac{(n!)^2}{(n-m)!} \sum_{j=0}^{n-m} \frac{(n-m)!}{j!(n-m-j)!} (-1)^j \\
&= (-1)^{n+m} \frac{(n!)^2}{(n-m)!} (1-1)^{n-m}.
\end{aligned}$$

Here the second equality is similar to that in [A.3](#) and the fourth uses

$$\int_0^{+\infty} x^{n-j} e^{-x} dx = (n-j)!.$$

This completes the proof. □

The orthogonality of the Laguerre polynomials ensures that the $L_n(x)$'s are linearly independent and $L_{n+1}(x)$ has $(n+1)$ distinct zeros $x_0, x_1, \dots, x_n$ in $[0, +\infty)$ [22]. With the zeros, the coefficients w_k are chosen so that the following $(n+1)$ equalities

$$\int_0^{+\infty} e^{-x} x^j dx = \sum_{k=0}^n w_k x_k^j, \tag{A.4}$$

hold for $j = 0, 1, \dots, n$. This leads to a system of $(n+1)$ linear algebraic equations for the unknowns w_k and the corresponding coefficient matrix is the Vandermonde matrix $[x_k^j]_{(n+1) \times (n+1)}$. The latter is invertible since the zeros are distinct and therefore the w_k 's are uniquely determined. The specific expressions of the w_k 's are given in [A.2](#) [22].

It is remarkable that all the coefficients w_k are non-negative. This important property ensures the stability and convergence of the Gauss-Laguerre quadrature [22]. Moreover, we have

Theorem 3. *The algebraic precision of the Gauss-Laguerre quadrature A.1 is $(2n+1)$ exactly. Namely, " $\approx$ " in A.1 is " $=$ " if $f(x)$ is a polynomial of degree $(2n+1)$ and is not " $=$ " if $f(x)$ is a polynomial with degree higher than $(2n+1)$.*

Proof. Notice that

$$(n+1)!L_{n+1}(x) = (-1)^{n+1}\prod_{k=0}^n(x-x_k)$$

is a polynomial of degree $(n+1)$. Since

$$0 < \int_0^{+\infty} e^{-x} L_{n+1}^2(x) dx \neq 0 = \sum_{k=0}^n w_k L_{n+1}^2(x_k),$$

the precision is less than $2(n+1)$. On the other hand, for any polynomial $p = p(x)$ of degree $(2n+1)$ there are two polynomials of degree n such that $p(x) = q(x)L_{n+1}(x) + r(x)$. Notice that $q(x)$ can be written as a linear combination of $L_0(x), L_1(x), \dots, L_n(x)$. Compute

$$\begin{aligned} \sum_{k=0}^n w_k p(x_k) &= \sum_{k=0}^n w_k [q(x_k)L_{n+1}(x_k) + r(x_k)] \\ &= \sum_{k=0}^n w_k r(x_k) \\ &= \int_0^{+\infty} e^{-x} r(x) dx \\ &= \int_0^{+\infty} e^{-x} [q(x)L_{n+1}(x) + r(x)] dx \\ &= \int_0^{+\infty} e^{-x} p(x) dx. \end{aligned}$$

Here the third equality is due to A.4 (the choice of w_k) and the fourth is due to the orthogonality of $L_{n+1}(x)$ and $q(x)$ with respect to the weight function. Hence the proof is complete. $\square$

For further details on the Gauss-Laguerre quadrature and for other Gauss quadratures, the interested reader is referred to the book [22].

NeRF-Synthetic		Avg.	Chair	Drums	Ficus	Hotdog	Lego	Mat.	Mic	Ship
PSNR $\uparrow$	TensorRF	32.39	34.68	25.58	33.37	36.81	35.51	29.54	33.59	30.12
	Ours	30.99	33.98	25.15	30.41	35.75	33.80	27.32	32.52	30.12
SSIM $\uparrow$	TensorRF	0.96	0.98	0.93	0.98	0.98	0.98	0.94	0.98	0.88
	Ours	0.94	0.98	0.92	0.96	0.97	0.97	0.91	0.98	0.87
LPIPS $\downarrow$	TensorRF	0.032	0.014	0.059	0.015	0.017	0.009	0.036	0.012	0.098
	Ours	0.048	0.019	0.068	0.043	0.031	0.015	0.088	0.029	0.095
LLFF		Avg.	Fern	Flower	Fortress	Horns	Leaves	Orchid	Room	Trex
PSNR $\uparrow$	TensorRF	26.51	25.31	28.22	31.14	27.64	21.34	20.02	31.80	26.61
	Ours	25.63	24.11	27.24	30.41	26.86	20.76	18.91	30.82	25.94
SSIM $\uparrow$	TensorRF	0.83	0.82	0.86	0.89	0.86	0.75	0.66	0.95	0.89
	Ours	0.80	0.76	0.82	0.87	0.83	0.72	0.59	0.92	0.87
LPIPS $\downarrow$	TensorRF	0.135	0.161	0.121	0.084	0.146	0.167	0.204	0.093	0.108
	Ours	0.146	0.181	0.115	0.089	0.146	0.146	0.255	0.122	0.118

Table A.1: Per-scene quantitative comparison between TensorRF and ours.

A.1.3 Proof for Theorem 1

Here we present a proof of the well-known Stone-Weierstrass theorem, basically from [8]. We rephrase the theorem here for readability.

Theorem 4. *Suppose $f = f(x) : [0, 1] \rightarrow (-\infty, \infty)$ is continuous. Then for any $\epsilon > 0$ there is a polynomial $p = p(x)$ satisfying*

$$\sup_{x \in [0, 1]} |f(x) - p(x)| < \epsilon.$$

Namely, polynomials are dense in the Banach space $C[0, 1]$.

Proof. Fix $f = f(x) \in C[0, 1]$ and $\epsilon > 0$. Since $f = f(x)$ is continuous on the bounded closed interval $[0, 1]$, it is bounded and uniformly continuous, meaning that there are positive numbers $M > 0$ and $\delta = \delta(\epsilon) > 0$ such that

$$|f(x)| \leq M, \quad |f(x) - f(y)| < \frac{\epsilon}{2} \quad (\text{A.5})$$

for any $x, y \in [0, 1]$ satisfying $|x - y| < \delta$.

With δ fixed, let $n \geq \frac{M}{\delta^2 \epsilon}$ be a positive integer. Consider the n th-order Bernstein

polynomial [8]

$$p(x) = \sum_{k=0}^n f\left(\frac{k}{n}\right) C_n^k x^k (1-x)^{n-k}$$

with $C_n^k = \frac{n!}{k!(n-k)!}$. Notice that, for $x \in [0, 1]$,

$$C_n^k x^k (1-x)^{n-k} \geq 0, \quad \sum_{k=0}^n C_n^k x^k (1-x)^{n-k} = (x + 1 - x)^n = 1, \quad (\text{A.6})$$

$$\begin{aligned} \sum_{k=0}^n k C_n^k x^k (1-x)^{n-k} &= \sum_{k=1}^n k \frac{n!}{k!(n-k)!} x^k (1-x)^{n-k} \\ &= nx \sum_{k=1}^n \frac{(n-1)!}{(k-1)!(n-1-(k-1))!} x^{k-1} (1-x)^{n-1-(k-1)} \quad (\text{A.7}) \\ &= nx(x + 1 - x)^{n-1} = nx, \end{aligned}$$

$$\begin{aligned} \sum_{k=0}^n k^2 C_n^k x^k (1-x)^{n-k} &= \sum_{k=0}^n k C_n^k x^k (1-x)^{n-k} + \sum_{k=0}^n k(k-1) C_n^k x^k (1-x)^{n-k} \\ &= nx + \sum_{k=2}^n k(k-1) \frac{n!}{k!(n-k)!} x^k (1-x)^{n-k} \\ &= nx + n(n-1)x^2 \sum_{k=2}^n \frac{(n-2)!}{(k-2)!(n-k)!} x^{k-2} (1-x)^{n-k} \\ &= nx + n(n-1)x^2. \end{aligned} \quad (\text{A.8})$$

A. Appendix

Then, for any $x \in [0, 1]$, we deduce from (A.5)–(A.8) that

$$\begin{aligned}
|f(x) - p(x)| &= \left| \sum_{k=0}^n [f(x) - f(\frac{k}{n})] C_n^k x^k (1-x)^{n-k} \right| \\
&\leq \sum_{k=0}^n |f(x) - f(\frac{k}{n})| C_n^k x^k (1-x)^{n-k} \\
&\leq \left(\sum_{k: |x-k/n| < \delta} + \sum_{k: |x-k/n| \geq \delta} \right) |f(x) - f(\frac{k}{n})| C_n^k x^k (1-x)^{n-k} \\
&< \left(\frac{\epsilon}{2} \sum_{k: |x-k/n| < \delta} + 2M \sum_{k: |x-k/n| \geq \delta} \right) C_n^k x^k (1-x)^{n-k} \\
&\leq \frac{\epsilon}{2} + 2M \sum_{k: |x-k/n| \geq \delta} \frac{(nx - k)^2}{n^2 \delta^2} C_n^k x^k (1-x)^{n-k} \\
&\leq \frac{\epsilon}{2} + \frac{2M}{n^2 \delta^2} \sum_{k=0}^n (nx - k)^2 C_n^k x^k (1-x)^{n-k} \\
&\leq \frac{\epsilon}{2} + \frac{2M}{n^2 \delta^2} (n^2 x^2 - 2n x n x + n x + n(n-1)x^2) = \frac{\epsilon}{2} + \frac{2M}{n \delta^2} x(1-x) \\
&\leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.
\end{aligned}$$

This completes the proof. □

A.1.4 Quantitative results on NeRF-Synthetic and LLFF datasets

In this part, we present our per-scene quantitative results on NeRF-Synthetic and LLFF datasets in Tab. A.2 for Vanilla NeRF and in Tab. A.1 for TensorRF. The full result for training using GL-NeRF is presented in Tab. A.3

NeRF-Synthetic		Avg.	Chair	Drums	Ficus	Hotdog	Lego	Mat.	Mic	Ship
PSNR $\uparrow$	Vanilla	30.63	34.32	25.80	29.54	35.49	29.53	29.04	31.78	29.52
	Ours	28.56	30.82	24.08	26.62	32.70	28.78	27.19	31.34	27.03
SSIM $\uparrow$	Vanilla	0.95	0.98	0.93	0.97	0.97	0.95	0.95	0.97	0.87
	Ours	0.93	0.96	0.90	0.94	0.96	0.94	0.92	0.97	0.84
LPIPS $\downarrow$	Vanilla	0.042	0.014	0.052	0.021	0.034	0.042	0.035	0.044	0.092
	Ours	0.070	0.050	0.098	0.055	0.059	0.047	0.070	0.044	0.135
LLFF		Avg.	Fern	Flower	Fortress	Horns	Leaves	Orchid	Room	Trex
PSNR $\uparrow$	Vanilla	27.62	26.82	28.37	32.59	28.83	22.38	21.20	32.87	27.93
	Ours	26.53	26.27	28.19	31.12	26.81	22.27	20.99	30.38	26.24
SSIM $\uparrow$	Vanilla	0.88	0.86	0.89	0.93	0.90	0.82	0.74	0.96	0.92
	Ours	0.85	0.84	0.88	0.89	0.86	0.81	0.73	0.93	0.89
LPIPS $\downarrow$	Vanilla	0.074	0.097	0.064	0.030	0.070	0.113	0.122	0.041	0.052
	Ours	0.090	0.106	0.066	0.064	0.096	0.115	0.125	0.075	0.075

Table A.2: Per-scene quantitative comparison between Vanilla NeRF and ours.

Blender		Avg.	Chair	Drums	Ficus	Hotdog	Lego	Mat.	Mic	Ship
PSNR $\uparrow$	Vanilla	30.63	34.32	25.80	29.54	35.49	29.53	29.04	31.78	29.52
	Ours	29.18	32.43	24.38	26.92	33.91	29.49	27.27	31.55	27.47
SSIM $\uparrow$	Vanilla	0.95	0.98	0.93	0.97	0.97	0.95	0.95	0.97	0.87
	Ours	0.93	0.97	0.91	0.94	0.96	0.95	0.92	0.97	0.84
LPIPS $\downarrow$	Vanilla	0.037	0.014	0.052	0.021	0.034	0.042	0.035	0.044	0.092
	Ours	0.056	0.029	0.087	0.050	0.052	0.038	0.065	0.046	0.122
LLFF		Avg.	Fern	Flower	Fortress	Horns	Leaves	Orchid	Room	Trex
PSNR $\uparrow$	Vanilla	27.62	26.82	28.37	32.59	28.83	22.38	21.20	32.87	27.93
	Ours	27.21	26.63	28.05	31.93	28.05	22.35	21.12	32.51	27.01
SSIM $\uparrow$	Vanilla	0.88	0.86	0.89	0.93	0.90	0.82	0.74	0.96	0.92
	Ours	0.87	0.85	0.88	0.91	0.88	0.81	0.74	0.95	0.90
LPIPS $\downarrow$	Vanilla	0.073	0.097	0.064	0.030	0.070	0.113	0.122	0.041	0.052
	Ours	0.087	0.121	0.075	0.043	0.089	0.117	0.131	0.053	0.069

Table A.3: Quantitative results when training on Blender and LLFF Datasets.

A.2 OMG

A.2.1 Hyperparameters

The neural network introduced uses 0.001, 0.0001, 0.0005 and 0.007 as learning rate respectively for Synthetic4Relight, Shiny Blender, Glossy Synthetic and MIP-NeRF 360 datasets. The neural network is implemented as a fully-connected MLP with two hidden layers, 128 as hidden dimension and ReLU as activation function. We directly feed the material properties into the MLP without any encoding such as positional embedding.

A.2.2 Full Derivation of Equations

Here we give the full derivation of Eq. 4.8.

$$\begin{aligned}
\frac{\partial L}{\partial m_i} &= \frac{\partial L}{\partial C} \frac{\partial C}{\partial m_i} = \frac{\partial L}{\partial C} \sum_{j=1}^N \frac{\partial(c_j \alpha_j T_j)}{\partial m_i} \\
&= \frac{\partial L}{\partial C} \sum_{j=1}^N \left[\frac{c_j}{m_i} \alpha_j T_j + c_j \frac{\partial \alpha_j}{\partial m_i} T_j + c_j \alpha_j \frac{\partial T_j}{\partial m_i} \right] \\
&= \frac{\partial L}{\partial C} \left[\frac{\partial c_i}{\partial m_i} \alpha_i T_i + c_i T_i \frac{\partial \alpha_i}{\partial m_i} + \sum_{j=i+1}^N c_j \alpha_j \frac{\partial}{\partial m_i} \prod_{k=1}^{j-1} (1 - \alpha_k) \right] \tag{A.1} \\
&= \frac{\partial L}{\partial C} \left[\frac{\partial c_i}{\partial m_i} \alpha_i T_i + c_i T_i \frac{\partial \alpha_i}{\partial m_i} + \sum_{j=i+1}^N c_j \alpha_j \frac{\partial}{\partial m_i} \exp\left(\sum_{k=1}^{j-1} (1 - \alpha_k)\right) \right] \\
&= \frac{\partial c_i}{\partial m_i} \frac{\partial L}{\partial C} \alpha_i T_i + c_i \frac{\partial L}{\partial C} T_i \frac{\partial \alpha_i}{\partial m_i} - \sum_{j=i+1}^N c_j \frac{\partial L}{\partial C} T_j \alpha_j \frac{1}{1 - \alpha_i} \frac{\partial \alpha_i}{\partial m_i},
\end{aligned}$$

A.2.3 Additional Results

We present the full results on Synthetic4Relight [70], Shiny Blender [56] and Glossy Synthetic [39]. All the baseline results are reproduced by us as in Chapter 4. More visualization results can be found in Fig. A.1, Fig. A.2, Fig. A.3, Fig. A.4 and Fig. A.5.

Scene	Method	Novel View Synthesis			Albedo			Relight			Roughness
		PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	MAE $\downarrow$
Balloons	R3DG	31.90	0.967	0.072	24.82	0.923	0.070	31.38	0.965	0.078	0.0009
	Ours	32.11	0.968	0.072	25.84	0.926	0.055	32.50	0.967	0.071	0.0011
Hotdog	R3DG	32.61	0.972	0.050	26.53	0.956	0.089	29.04	0.954	0.067	0.0357
	Ours	33.59	0.975	0.047	26.95	0.957	0.085	28.73	0.955	0.066	0.0217
Chair	R3DG	34.72	0.978	0.039	30.55	0.963	0.046	33.07	0.966	0.041	0.0036
	Ours	35.00	0.979	0.040	30.82	0.964	0.045	33.46	0.968	0.041	0.0039
Jugs	R3DG	37.03	0.984	0.027	32.88	0.967	0.064	37.98	0.984	0.023	0.0032
	Ours	37.17	0.984	0.027	33.64	0.968	0.049	38.16	0.984	0.023	0.0074

Table A.4: Per-scene results on Synthetic4Relight dataset. For albedo reconstruction results, we follow the standard practice [19, 69] and scale each RGB channel by a global scalar.

Shiny Blender		Avg.	Ball	Car	Coffee	Helmet	Teapot	Toaster
PSNR $\uparrow$	GaussianShader	30.77	29.05	28.51	31.55	28.91	43.57	23.05
	Ours	31.07	29.65	28.14	31.29	28.79	43.48	25.07
SSIM $\uparrow$	GaussianShader	0.953	0.955	0.940	0.970	0.956	0.996	0.902
	Ours	0.957	0.961	0.939	0.969	0.955	0.996	0.924
LPIPS $\downarrow$	GaussianShader	0.081	0.146	0.047	0.085	0.086	0.011	0.110
	Ours	0.076	0.137	0.050	0.086	0.085	0.011	0.089

Table A.5: Per-scene results on Shiny Blender dataset.

A.2.4 Visualization of Cross Section

To understand what the cross section network is learning, we visualize some examples in Fig. A.6. In the figure, the darker region indicates a lower cross section, meaning that it lets more light pass through, whereas the brighter regions stand for the regions that have a higher cross section. The example shows that the windows as well as the plastic bag on the floor actually have a lower cross section assigned to them, which corresponds to human intuition.

Glossy Synthetic		Avg.	Angel	Bell	Cat	Horse	Luyu	Potion	Tbell	Teapot
PSNR $\uparrow$	GaussianShader	27.25	26.92	29.73	31.43	26.38	27.31	29.54	23.41	23.25
	Ours	27.68	28.24	29.91	31.00	26.62	27.32	29.40	24.67	24.31
SSIM $\uparrow$	GaussianShader	0.925	0.923	0.939	0.961	0.934	0.917	0.936	0.897	0.895
	Ours	0.929	0.929	0.940	0.959	0.934	0.916	0.935	0.907	0.909
LPIPS $\downarrow$	GaussianShader	0.083	0.071	0.085	0.058	0.056	0.066	0.096	0.136	0.098
	Ours	0.081	0.066	0.084	0.059	0.057	0.067	0.097	0.125	0.091

Table A.6: Per-scene results on Glossy Synthetic dataset.



Figure A.1: More qualitative example in comparison to GaussianShader on Glossy Synthetic dataset.

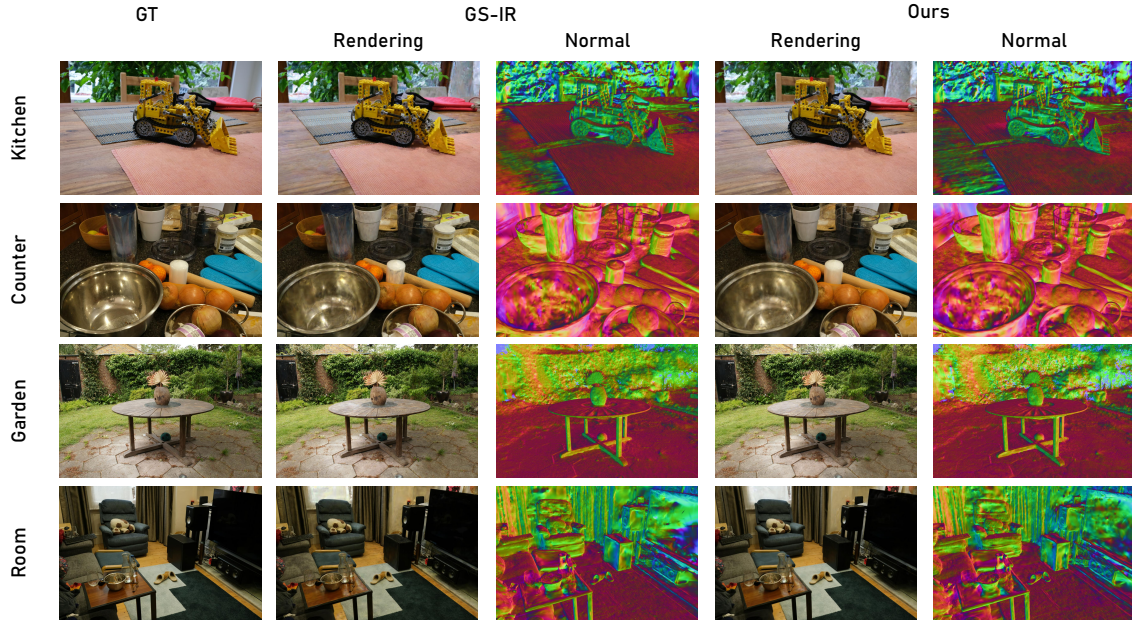


Figure A.2: More qualitative example in comparison to GS-IR on MIP-NeRF 360 dataset.

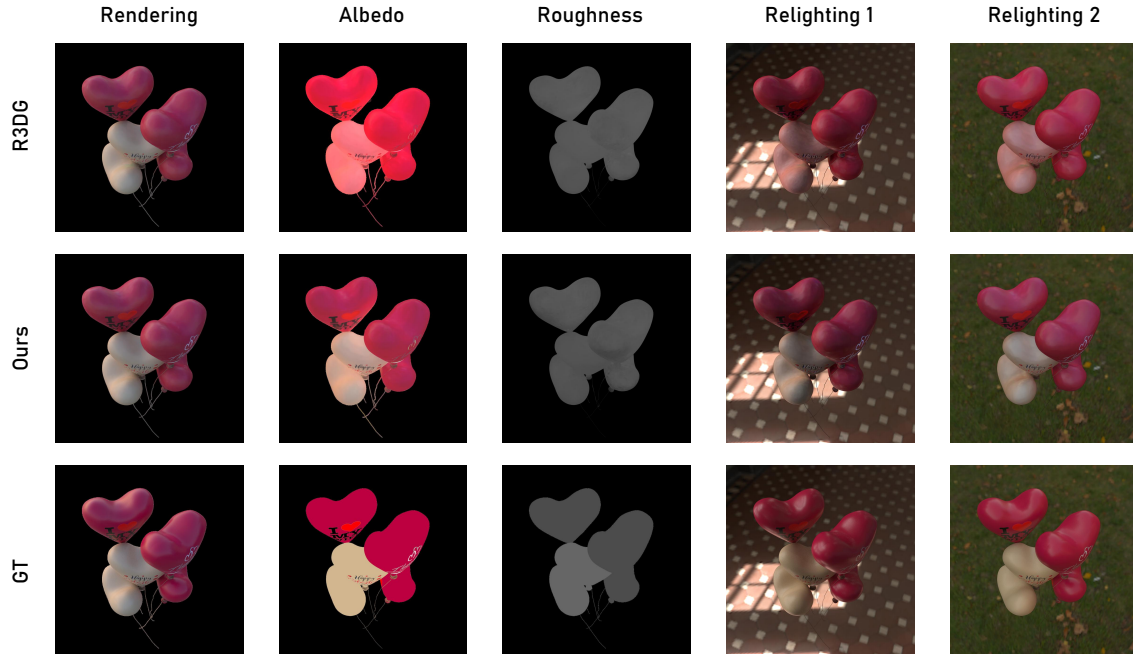


Figure A.3: More qualitative example in comparison to R3DG on Synthetic4Relight dataset.

A. Appendix

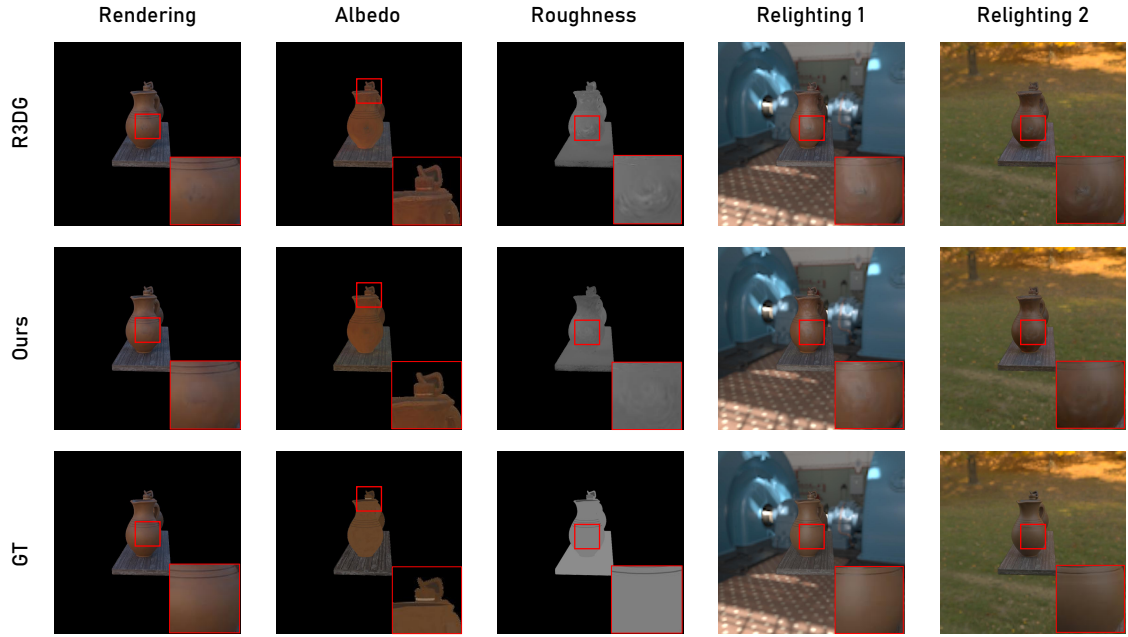


Figure A.4: More qualitative example in comparison to R3DG on Synthetic4Relight dataset. We add a gamma correction to the relighting results to make the visual differences clearer.

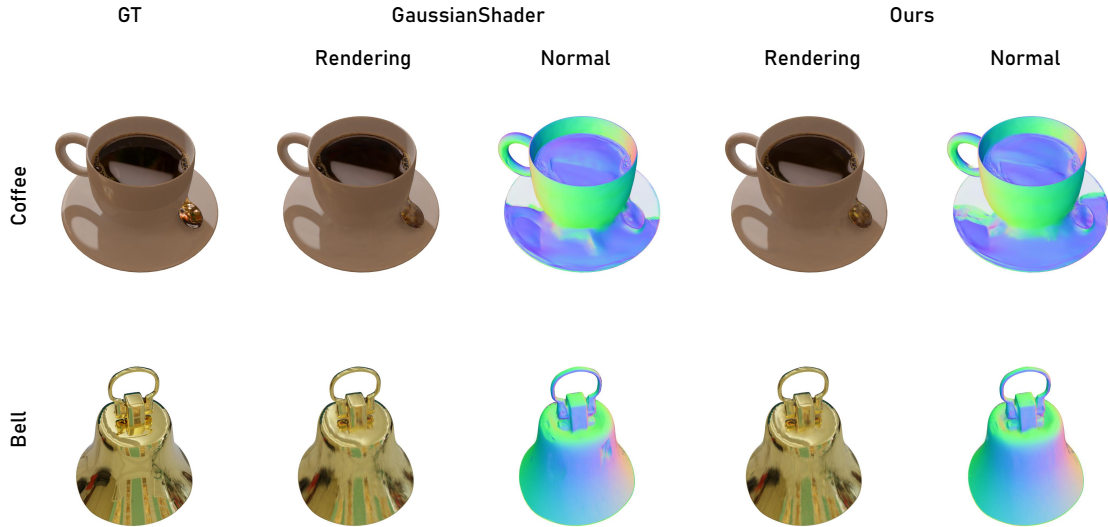


Figure A.5: More qualitative example in comparison to GaussianShader on Glossy Synthetic and Shiny Blender datasets.



Figure A.6: Visualization of the learned cross section on Mip-NeRF 360 dataset. Dark indicates the lower cross section (meaning it lets more light pass through) and bright stands for the higher cross section. The example shows that the windows as well as the plastic bag on the floor actually have a lower cross section assigned to them, which corresponds to human intuition.

A. Appendix

Bibliography

- [1] Milton Abramowitz, Irene A Stegun, and Robert H Romer. Handbook of mathematical functions with formulas, graphs, and mathematical tables, 1988. [3.2](#)
- [2] Benjamin Attal, Dor Verbin, Ben Mildenhall, Peter Hedman, Jonathan T Barron, Matthew O’Toole, and Pratul P Srinivasan. Flash cache: Reducing bias in radiance cache based inverse rendering. *arXiv preprint arXiv:2409.05867*, 2024. [2.1.2](#)
- [3] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 17–24, 2013. [1](#)
- [4] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. [1](#), [2.2.1](#)
- [5] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. ([document](#)), [1](#), [2.2.1](#), [4.4.1](#), [4.4.3](#), [4.6](#), [4.3](#)
- [6] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023. [1](#)
- [7] Beer. Bestimmung der absorption des rothen lichts in farbigen flüssigkeiten. *Annalen der Physik*, 162(5):78–88, 1852. [4.1](#), [4.2](#)
- [8] Serge Bernstein. Démonstration du théorème de weierstrass fondée sur le calcul des probabilités. *Commun. Soc. Math. Kharkow*, 13(1):1–2, 1912. [A.1.3](#), [A.1.3](#)
- [9] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*,

2020. [2.1.2](#), [4.3.4](#)
- [10] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021. [1](#), [2.1.2](#), [4.3.4](#)
 - [11] Pierre Bouguer. *Essai d’optique sur la gradation de la lumière*. Claude Jombert, 1729. [4.1](#), [4.2](#)
 - [12] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. [1](#), [2.2.1](#), [3.4](#), [3.4.1](#), [3.4.2](#)
 - [13] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in neural information processing systems*, 32, 2019. [2.1.2](#)
 - [14] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Acm siggraph 2008 classes*, pages 1–10. 2008. [1](#)
 - [15] Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Transactions on Graphics (TOG)*, 33(6):1–12, 2014. [2.1.2](#)
 - [16] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4):65–74, 1988. [2.1.1](#)
 - [17] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023. [2.2.2](#)
 - [18] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. [1](#), [2.1.1](#)
 - [19] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023. [\(document\)](#), [1](#), [2.2.2](#), [4.1](#), [4.3](#), [4.4.1](#), [4.4.2](#), [A.4](#)
 - [20] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14346–14355, 2021. [1](#)

- [21] Carl Friedrich Gauss. *Methodus nova integralium valores per approximationem inveniendi*. Dieterich, 1815. [3.2](#)
- [22] Walter Gautschi. *Numerical analysis*. Springer Science & Business Media, 2011. [A.1.2](#), [A.1.2](#), [A.1.2](#)
- [23] Yuan-Chen Guo. Instant neural surface reconstruction, 2022. <https://github.com/bennyguo/instant-nsr-pl>. [A.1.1](#)
- [24] Kunal Gupta, Milos Hasan, Zexiang Xu, Fujun Luan, Kalyan Sunkavalli, Xin Sun, Manmohan Chandraker, and Sai Bi. Mcnerf: Monte carlo rendering and denoising for real-time nerfs. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023. [2.1.1](#), [3.4.2](#)
- [25] James Ivory. V. on the figure requisite to maintain the equilibrium of a homogeneous fluid mass that revolves upon an axis. *Philosophical Transactions of the Royal Society of London*, (114):85–150, 1824. [3.2](#)
- [26] Carl Gustav Jakob Jacobi. Ueber gauss neue methode, die werthe der integrale näherungsweise zu finden. 1826. [3.2](#)
- [27] JDG Jacobi. Ueber eine besondere gattung algebraischer functionen, die aus der entwicklung der function $(1-2xz+z^2)^{1/2}$ entstehen. 1827. [3.2](#)
- [28] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5322–5332, 2024. ([document](#)), [1](#), [2.2.2](#), [4.1](#), [4.3](#), [4.4.1](#), [4.2](#), [4.4.2](#)
- [29] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. Tensorir: Tensorial inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2023. [1](#), [2.1.2](#)
- [30] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986. [2.1.2](#), [4.1](#)
- [31] Yoshihiro Kanamori and Yuki Endo. Relighting humans: occlusion-aware inverse rendering for full-body human images. *arXiv preprint arXiv:1908.02714*, 2019. [2.1.2](#), [4.1](#)
- [32] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. ([document](#)), [1](#), [2.1.1](#), [2.2.2](#), [4.1](#), [4.3](#)
- [33] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser.

- Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12871–12881, 2022. [2.2.1](#)
- [34] Andreas Kurz, Thomas Neff, Zhaoyang Lv, Michael Zollhöfer, and Markus Steinberger. Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. In *European Conference on Computer Vision*, pages 254–270. Springer, 2022. [2.2.1](#), [3.3.1](#)
- [35] Jean-Henri Lambert. *Photometria sive de mensura et gradibus luminis, colorum et umbrae*. Sumptibus viduae Eberhardi Klett, typis Christophori Petri Detleffsen, 1760. [4.1](#), [4.2](#)
- [36] Zhen Li, Lingli Wang, Mofang Cheng, Cihui Pan, and Jiaqi Yang. Multi-view inverse rendering for large-scale real-world indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12499–12509, 2023. [2.1.2](#), [4.1](#)
- [37] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21644–21653, 2024. ([document](#)), [1](#), [2.2.2](#), [4.1](#), [4.3](#), [4.4.1](#), [4.4.3](#), [4.6](#)
- [38] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14556–14565, 2021. [2.1.1](#)
- [39] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. *ACM Transactions on Graphics (TOG)*, 42(4):1–22, 2023. ([document](#)), [2.1.2](#), [4.4.1](#), [4.2](#), [4.4.2](#), [4.5](#), [A.2.3](#)
- [40] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. [2.2.2](#)
- [41] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Francisco Vicente Carrasco, Markus Steinberger, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. *arXiv preprint arXiv:2406.15643*, 2024. [2.2.2](#)
- [42] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. [2.1.1](#), [2.1.1](#)
- [43] Thomas G Mayerhöfer, Susanne Pahlow, and Jürgen Popp. The bouguer-beer-lambert law: Shining light on the obscure. *ChemPhysChem*, 21(18):2029–2046,

2020. [4.1](#), [4.2](#)
- [44] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. [3.4](#)
 - [45] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. [1](#), [2.1.1](#), [2.2.1](#), [3.4](#), [4.3.4](#), [A.1.1](#)
 - [46] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. [1](#), [2.2.1](#), [3.4](#), [3.4.1](#)
 - [47] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8280–8290, 2022. [2.1.2](#)
 - [48] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koochpayegani, and Hamed Pirsiavash. Compact3d: Compressing gaussian splat radiance field models with vector quantization. *arXiv preprint arXiv:2311.18159*, 2023. [2.2.2](#)
 - [49] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *Computer Graphics Forum*, volume 40, pages 45–59. Wiley Online Library, 2021. [1](#), [2.1.1](#), [2.2.1](#), [3.3.1](#)
 - [50] Martin Píala and Ronald Clark. Terminerf: Ray termination prediction for efficient neural rendering. In *2021 International Conference on 3D Vision (3DV)*, pages 1106–1114. IEEE, 2021. [3.3.1](#)
 - [51] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. [2.2.1](#)
 - [52] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024. [2.2.2](#)
 - [53] Olinde Rodrigues. *De l’attraction des sphéroïdes, Correspondence sur l’École Impériale Polytechnique*. PhD thesis, PhD thesis, Thesis for the Faculty of Science of the University of Paris, 1816. [3.2](#)

- [54] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021. [1](#), [2.1.2](#)
- [55] Mikaela Angelina Uy, Kiyohiro Nakayama, Guandao Yang, Rahul Krishna Thomas, Leonidas Guibas, and Ke Li. Nerf revisited: Fixing quadrature instability in volume rendering. *arXiv preprint arXiv:2310.20685*, 2023. [2.1.1](#)
- [56] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022. ([document](#)), [2.1.2](#), [4.4.1](#), [4.2](#), [4.4.2](#), [4.5](#), [A.2.3](#)
- [57] Dor Verbin, Ben Mildenhall, Peter Hedman, Jonathan T Barron, Todd Zickler, and Pratul P Srinivasan. Eclipse: Disambiguating illumination and materials using unintended shadows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 77–86, 2024. [2.1.2](#)
- [58] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *arXiv preprint arXiv:2111.13260*, 2021. [2.2.1](#)
- [59] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. [3.4](#)
- [60] Lee Westover. Interactive volume rendering. In *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, pages 9–16, 1989. [2.1.1](#)
- [61] Liwen Wu, Jae Yong Lee, Anand Bhattad, Yu-Xiong Wang, and David Forsyth. Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16200–16209, 2022. [1](#), [2.1.1](#)
- [62] Wenqi Yang, Guanying Chen, Chaofeng Chen, Zhenfang Chen, and Kwan-Yee K Wong. Ps-nerf: Neural inverse rendering for multi-view photometric stereo. In *European Conference on Computer Vision*, pages 266–284. Springer, 2022. [2.1.2](#), [4.1](#)
- [63] Yao Yao, Jingyang Zhang, Jingbo Liu, Yihang Qu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Neilf: Neural incident light field for physically-based material estimation. In *European Conference on Computer Vision*, pages 700–716. Springer, 2022. [1](#), [2.1.2](#)
- [64] Lin Yen-Chen. Nerf-pytorch. <https://github.com/yenchenlin/>

- [nerf-pytorch/](#), 2020. [A.1.1](#)
- [65] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenotrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. [2.2.1](#)
 - [66] Kai Zhang, Gernot Riegler, Noah Snaveley, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. [2.2.1](#)
 - [67] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snaveley. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5453–5462, 2021. [1](#), [2.1.2](#)
 - [68] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. [3.4](#)
 - [69] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (ToG)*, 40(6):1–18, 2021. [\(document\)](#), [1](#), [2.1.2](#), [A.4](#)
 - [70] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18643–18652, 2022. [\(document\)](#), [2.1.2](#), [2.1.2](#), [4.1](#), [4.4.1](#), [4.4](#), [4.1](#), [A.2.3](#)