ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Computer Vision And Image Processing

RELIGHTING NEURAL RADIANCE FIELDS LEVERAGING SHADOW MAPPING

CANDIDATE

Mazeyar Moeini Feizabadi

SUPERVISOR

Prof. Luigi Di Stefano

CO-SUPERVISOR

Riccardo Spezialetti, PhD.

Academic year 2022-2023

Session Six

Dedicated to my Mother who loved me unconditionally, my enduring Father who taught me perseverance, and my compassionate Sisters who always stood beside me

Acknowledgements

I'm very grateful to eyecan for the time and resources they entrusted me with to complete this project. I'd also like to acknowledge my advisor, Prof. Luigi Di Stefano, for his wisdom and advice while supervising me. This work could not have been completed without the patience and expertise of Riccardo Spezialetti, who kindly guided me through every step. To the friends and family who have supported me over the past two years, your love and care will never be forgotten.

Contents

List of Figures

List of Tables

Abstract

NeRF (Neural Radiance Fields) belongs to the category of modern implicit 3D image reconstruction algorithms. NeRF achieves state-of-the-art novel view synthesis of complex scenes by optimizing a continuous volumetric rendering scene function. However, it cannot synthesize under unobserved light conditions. Several relighting methods have been proposed but have shown to be prohibitively expensive to train, failing to gain traction in real-world applications. ReNe (Relighting NeRF) from eyecan.ai presented a novel data set of various objects with complex geometry and a new lightweight architecture that can render real-world objects under one-light-at-time (OLAT) conditions. Any method that aims to generate a realistic scene must have geometrically accurate shadows. ReNe's lightweight architecture solves this by estimating global geometry through its visibility. A common theme of relighting is that approximations about geometry must be made to avoid expensive geometric queries. In this thesis, I propose using a classical computer graphics technique called shadow mapping to create low-cost yet convincing shadows. Shadow mapping utilizes precalculated distance maps from the viewpoint of the light to obtain an understanding of geometry. Through shadow mapping, we can create shadow hints, which are structured and geometrical scalars, to better advise shadow predictions at every point. With no architectural change, the injection of shadow hints produced more accurate shadows. We empirically tested our hybrid approach on the entire ReNe dataset, where we set new state-of-the-art results.

Chapter 1

Introduction

1.1 Introduction

Inverse rendering aims to solve the problem of estimating the properties of a scene given a set of images. As opposed to traditional rendering, which starts with a 3D scene and produces a 2D image, inverse rendering does the opposite. With many or sometimes even one image, it attempts to understand geometry reconstruction, lighting estimation, material estimation, and camera pose estimation. The task is challenging due to the inherent ambiguity that can arise in complex scenes, further complicated with dynamic lighting. The ability to inverse render under unobserved lighting conditions would greatly benefit many applications.

Given recent advances in 3D computer vision algorithms such as NeRF [11] (Neural Radiance Fields), the landscape of 3D model representations is vastly changing. NeRF represents a paradigm change from classical computer graphic methods utilizing modern deep learning frameworks to model and represent 3D scenes implicitly. Traditional computer graphics rely on explicit representations of geometry such as manual algorithms; NeRF learns to represent from data-driven training, making it more flexible for complex scene modeling and rendering. [9]

Relighting is considered a critical technique to achieve realism within classic computer graphics. Relighting allows for dynamic control over a scene's appearance and can enhance immersiveness. While NeRF is a state of the art method for novel view synthesis, it fails to change the appearance of lights. The problem of moving light within a scene is known as Relighting. In many applications of 3D asset creation, the lighting condition is baked in, making the asset unfeasible in other environments. Even within the same environment, the lighting conditions can be dynamic; consider a naturally lit factory with the sun's movement. Given how effective NeRFs are in scene representation, their limitations arise from data-driven training and fixed formulation, and this requires

a whole pipeline change to add new capabilities. While the geometry of NeRF is implicit, so are its lighting representations. Since NeRF learns a black-box function that maps 3D points and viewing directions to colors and densities, it is difficult to manipulate the scene's lighting conditions or material properties. NeRF relies on volume rendering to integrate the radiance along camera rays. However, simulating the attenuation and reflection of light by particles in the volume is very challenging, as it requires querying the neural network for the density at densely sampled points along many light rays. The computational cost becomes prohibitively expensive for complex lighting scenarios, such as environment maps or global illumination. NeRF assumes that the lighting conditions are fixed and known for the input images and uses them to compute the loss function during training. However, in outdoor scenes or uncontrolled settings, the lighting may vary significantly across the images and may not be accurately measured or estimated. Measurements that do not align can lead to errors or artifacts in the learned representation.

Several methods have been proposed in recent years to enable relighting in NeRF. These include techniques from Neural Reflectance Fields for Appearance Acquisition [1], NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis [14], NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination [20], NeRD: Neural Reflectance Decomposition From Image Collections [3]. The following papers were published after NeRF and contain collective insights on relighting a NeRF. Their main insights show that the dataset must be structured to have convincing relighting, and computational resources must be plentiful. NeRFactor, which was the fastest, still utilizes four GTX TITANs for 6-8 Hours, while NeRV, being the slowest, uses 128 TPU cores for one Day. Recently, there have been new papers that provide novel techniques and datasets. The "ReNeRF: Relightable Neural

Radiance Fields with Nearfield Lighting" [18] paper method builds relightable neural radiance fields (ReNeRFs) from images captured under a few area lights without requiring a dense light stage. This approach is inspired by image-based relighting, a data-driven approach that captures global light transport without explicit simulations or assumptions. The method models a continuous OLAT basis for each 3D point using a spherical codebook and an OLAT MLP and disentangles diffuse and specular radiance using a NeRF MLP. The method achieves photorealistic relighting of complex scenes with arbitrary materials and geometry under novel views and lighting, including nearfield and environment lighting. The paper "OpenIllumination: A Multi-Illumination Dataset for Inverse Rendering Evaluation on Real Objects" [8] introduces OpenIllumination, a dataset of more than 108,000 images of 64 different objects. The objects in the dataset have diverse materials and are captured under 72 camera views with various illuminations. Accurate camera parameters, illumination ground truth, and foreground segmentation masks accompany each image in the dataset. This dataset enables the quantitative evaluation of most inverse rendering and material decomposition methods for natural objects. The paper "Relighting Neural Radiance Fields with Shadow and Highlight Hints" [19] introduces a novel neural implicit radiance representation for free viewpoint relighting from a small set of unstructured photographs. The paper also introduces shadow and highlights hints to the radiance MLP to help model high-frequency light transport effects, such as occlusions and specular reflections, by leveraging signed distance functions (SDF). This representation's effectiveness and robustness are demonstrated on various challenging synthetic and real-world objects containing various materials, shape complexity, and global light transport effects.

The problem I intended to solve as a part of this thesis was to illuminate a scene's representation using NeRF with more accurately cast

shadows. The starting point for this thesis work is the paper ReLight My NeRF: A Dataset for Novel View Synthesis and Relighting of Real World Objects, a paper proposed at CVPR 2023 from eyecan.ai. It aims to set a new benchmark for relighting. They offer a custom, highly structured dataset and a much faster algorithm. The dataset comprises 2000 images acquired from 50 different points of view under 40 different OLAT conditions. It was the first to have multiple categories: real-world, background shadows, public, and light supervision. Their architecture coined as V5, was developed after iterations of the Instant NGP [12]. Their findings show that relighting is as much an architectural problem as an information injection problem, and the purpose should be to inject highly fruitful information.

In the ReNe paper, the authors attempted to model an MLP to predict a scalar value that allows efficient query of the point-to-light visibility, given a position and a normalized position to the light source vector. The MLP, however, acts as a memorizer that creates a shadow based on its location and does not generalize two new unseen lighting illuminations well. Fundamentally, cast shadows are not a local illumination problem but a global geometry problem. It is possible to mistakenly assume that shadows are solely governed by local illumination since the BRDF correlates light color to position. However, to achieve accurate generalization, one must consider the global geometry, precisely the path between the light and the point. This path can be prohibitively expensive as the number of queried points becomes squared. The critical insight of this thesis is to utilize a traditional graphics technique known as shadow mapping within NeRF. It works by rendering the scene from the viewpoint of a light source to create a *shadow map* that represents the depths of objects from that light's perspective (pre-computed). Then, during the regular scene rendering, this shadow map is used to determine which parts of the

scene are in shadow and which are not, allowing for the accurate rendering of shadows. Using Shadow Mapping, we could create a high-quality input named *shadow hints*, which considers geometry for cast shadows. In the results for never-before-seen lighting conditions, it is clear that the V5 cannot render geometrically accurate cast shadows, while the version with shadow hints does. Overall, through empirical testing on the benchmarks, we showed that shadow mapping does produce superior results. Given the variability of quality given NeRF initialization and that we only improved a small part of the image, these results are rather satisfactory.

This thesis is organized as follows: Chapter Two *Foundations of Radiance and Illumination* introduces the background concepts such as NeRF and foundations of Radiance and Illumination. With a solid understanding of the radiometric properties of light and the volumetric rendering algorithm of NeRF, it is then finally possible to delve into the different NeRF-based relighting algorithms. In Chapter Three *Related Work*, an overview of NeRF-based relighting works literature is presented. Chapter Four *Methodology* reviews the methodology and results to show the difference shadow mapping can make.

Chapter 2

Foundations of NeRF, Radiance, and Illumination

2.1 Radiance and Illumination

2.1.1 Modeling Assumptions

The assumption of light that we make is in its simplest terms. We avoid the more complicated models of light, such as polarization and wave optics. We model light as individual rays that travel in straight lines with infinite speed. [6] This assumption would quickly break down in some real-world scenes outside of studio conditions, such as being underwater as shown by SeaThru-NeRF. [7] Lighting effects such as diffraction and interference would require an extra layer of modeling.

2.1.2 Radiometric Quantities

To model the amount of light radiating from a scene, we must model the light transport as a physical term, such as light as a measure of energy. We know that the energy going outwards from a surface depends on the energy flowing inwards over time. Denoted by Φ and expressed in terms of watts $[W = J \cdot s^{-1}]$ **flux** is a unit of energy described measured Jules.

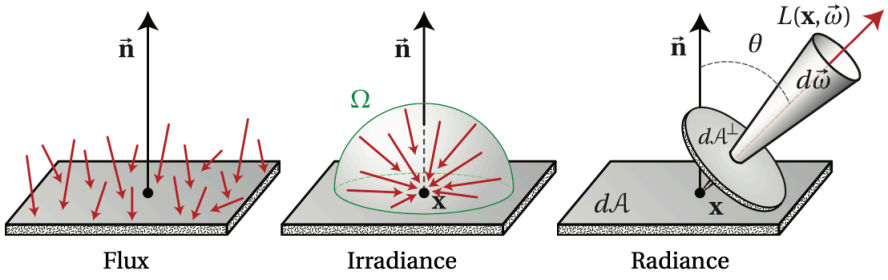

Figure 2.1: Flux, Irradiance, Radiance
Flux measures the amount of light that impacts the surface in a specified finite area. **Irradiance** is the integral around the hemisphere Ω arrive at the point **x**. **Radiance** measures the quantity of light arriving at a single point from a differential solid angle. Figure from [6]

Irradiance is a crucial parameter in radiative transfer, denoting the

quantity of incident power impacting a surface per unit area. It is quantified in scientific units as $[W \cdot m^{-2}]$ and can be formulated with regard to flux:

$$E(\mathbf{x}) = \frac{d\Phi(\mathbf{x})}{d\mathcal{A}(\mathbf{x})} \tag{2.1}$$

Irradiance has to be measured on a surface at a point with a surface normal $\vec{\mathbf{n}}$. This is different from the term *radiant exitance* (M) or *radiosity* (B), which is the flux departing the surface that will be defined later.

Radiance formulate the amount of flux that arrives from a hypothetical disk with a differential area $d\mathcal{A}^{\perp}$ that is perpendicular to a differential direction $d\vec{w}$. Radiance is quantified in the units of $[W \cdot sr^{-1} \cdot m^{-2}]$ and can be expressed as follows:

$$L(\mathbf{X}, \vec{w}) = \frac{d^2\Phi(\mathbf{x}, \vec{w})}{d\vec{w}d\mathcal{A}^{\perp}(\mathbf{x})} \tag{2.2}$$

We note that it may be more useful to describe radiance as the measurement of light at the surface rather than at a hypothetical surface perpendicular to the incident angle of $\vec{w}$. We can therefore consider $d\vec{A}^{\perp} = (\vec{\mathbf{n}} \cdot \vec{w})d\mathcal{A}$ to obtain:

$$L(\mathbf{X}, \vec{w}) = \frac{d^2\Phi(\mathbf{x}, \vec{w})}{(\vec{\mathbf{n}} \cdot \vec{w})d\vec{w}d\mathcal{A}(\mathbf{x})} \tag{2.3}$$

With another substitution of $d\vec{w}^{\perp} = (\vec{\mathbf{n}} \cdot \vec{w})d\vec{w}$ we can also derive the following expression for radiance:

$$L(\mathbf{X}, \vec{w}) = \frac{d^2\Phi(\mathbf{x}, \vec{w})}{d\vec{w}^{\perp}d\mathcal{A}(\mathbf{x})} \tag{2.4}$$

This formula can be more pragmatic in describing radiance since our area at the point $\mathbf{x}$ is the surface rather than an area perpendicular to the direction.

The radiance measurement interests us the most as its value is what is perceived by human eyes and camera sensors.

So far, we have defined the following radiometric quantities [6].

Symbol	Units	Description
Φ	W	Flux
E	$W \cdot m^{-2}$	Irradiance
M	$W \cdot m^{-2}$	Radiant Exitance (outgoing)
B	$W \cdot m^{-2}$	Radiosity (Outgoing)
L	$W \cdot m^{-2} \cdot sr^{-1}$	Radiance

Table 2.1: Radiometric Quantities

2.1.3 Radiometric Relationships and Properties

Now that radiance is quantified, it is important to distinguish the difference between *incident radiance* and *exitant radiance*. Incident radiance is expressed as $L(\mathbf{x} \leftarrow \vec{w})$ representing the radiance impacting the surface at the point $\mathbf{x}$ from direction $\vec{w}$. $L(\mathbf{x} \rightarrow \vec{w})$ represents the exitant radiance exiting the surface at the point $\mathbf{x}$ to direction $\vec{w}$. The **BRDF (Bidirectional Reflectance Distribution Function)** is one of the most fundamental and central functions in computer graphics and radiance fields. It describes the relationship between the two quantities of *incident radiance* and *exitant radiance*. Its flexibility allows us to accurately represent different materials from various viewing and illumination conditions, a key point in relighting.

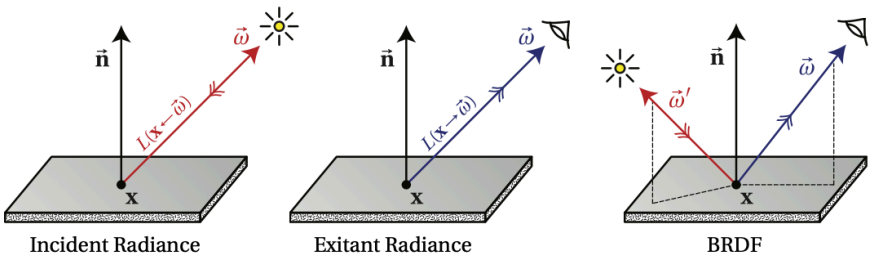

Figure 2.2: Incident Radiance, Exitant Radiance, BRDF
Figure from [6]

Using the $L(\mathbf{x} \leftarrow \vec{w})$ representation of incident radiance we can express flux; this requires the double integral over the hemisphere Ω at each point of the area $\mathcal{A}$ as follows:

$$\Phi = \int_{\mathcal{A}} \int_{\Omega} L(\mathbf{x} \leftarrow \vec{w})(\vec{\mathbf{n}} \cdot \vec{w}) d\vec{w} d\mathcal{A}(\mathbf{x}) \tag{2.5}$$

Furthermore, we can also express Irradiance (E), Radiant (M) exitance, and Radiosity (B) as integrals.

$$E(\mathbf{x}) = \int_{\Omega} L(\mathbf{x} \leftarrow \vec{w})(\vec{\mathbf{n}} \cdot \vec{w}) d\vec{w} \tag{2.6}$$

$$M(\mathbf{x}) = B(\mathbf{x}) = \int_{\Omega} L(\mathbf{x} \rightarrow \vec{w})(\vec{\mathbf{n}} \cdot \vec{w}) d\vec{w} \tag{2.7}$$

Thus far, when referring to incident radiance $L(\mathbf{x} \leftarrow \vec{w})$, and exitant radiance $L(\mathbf{x} \rightarrow \vec{w})$, the direction of the vector $\vec{w}$ was point away from the surface, even if $\vec{w}$ were equal in both cases, the measurement would be different since the incident measures the radiance arriving at the surface and the exitant measures the radiance leaving, these can only be measured at different times. Therefore:

$$L(\mathbf{x} \leftarrow \vec{w}) \neq L(\mathbf{x} \rightarrow \vec{w}) \tag{2.8}$$

However, due to the conservation of energy, we can derive that the incident radiance at point $\mathbf{x}$ coming from the direction $\vec{w}$ will continue off as the exitant radiance at point $\mathbf{x}$ in the direction $-\vec{w}$. (Not correct wording yet)

$$L(\mathbf{x} \leftarrow \vec{w}) = L(\mathbf{x} \rightarrow -\vec{w}) \tag{2.9}$$

We can also derive two distinct points' incident and exitant radiance functions. In our modeling assumptions, we modeled light as individual rays that travel in straight lines. In a typical scene, the observer does not view most of the reflected light but impacts another surface. This

realization makes the basis of the rendering equation as defined later. Therefore, it is vital to state the property that the incident radiance at the point $\mathbf{x}$ from direction $\vec{w}$ is equal to the outgoing radiance from the closest visible point in that direction [6]. To demonstrate this property we can define *ray casting function* which when given a point $\mathbf{x}$ and a direction $\vec{w}$: $\mathbf{r}(\mathbf{x}, \vec{w}) = \mathbf{x}'$, it returns the point $\mathbf{x}'$ which is the closest point along the casted ray. In the figure below, we can quickly see how one point's exitance is another point's incidence.

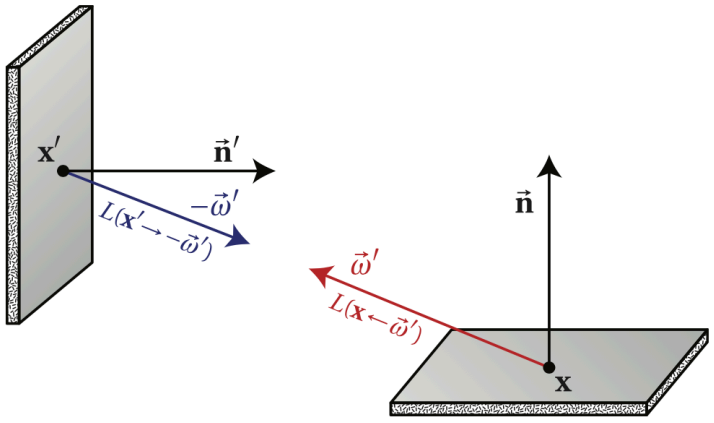

Figure 2.3: Incident and Exitant Radiance Relationship
Figure from [6]

The following equation can also express this.

$$L(\mathbf{x} \leftarrow \vec{w}) = L(\mathbf{x}' \rightarrow -\vec{w}) \tag{2.10}$$

2.1.4 The BRDF

The **BRDF (Bidirectional Reflectance Distribution Function)** is a six-dimensional (2 two angular components for $\vec{w}'$, $\vec{w}$ and $\vec{\mathbf{n}}$) it simply describes the radiance or "brightness" as we perceive it from viewing angle $\vec{w}$ as a function from the illumination angle $\vec{w}'$. In most illustrations, $\mathbf{x}$ and $\vec{w}$ are fixed as $\vec{w}'$ are set free to show how the BRDF operates as a distribution function.

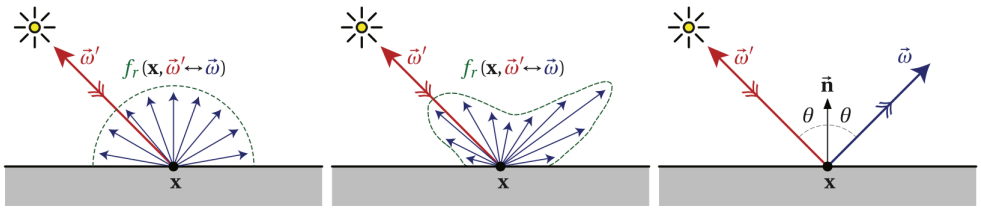

Figure 2.4: Diffusion, Lambertian and Specular BRDFs
Figure from [6]

This is the ratio of the infinitesimal change in outgoing radiance $dL(\mathbf{x} \rightarrow \vec{w})$ to the infinitesimal change in incident irradiance $dE(\mathbf{x} \leftarrow \vec{w}')$ The formula can be written as:

$$f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w}) = \frac{dL(\mathbf{x} \rightarrow \vec{w})}{dE(\mathbf{x} \leftarrow \vec{w}')} = \frac{dL(\mathbf{x} \rightarrow \vec{w})}{L(\mathbf{x} \leftarrow \vec{w})(\vec{\mathbf{n}} \cdot \vec{w}')d\vec{w}'} \tag{2.11}$$

Different Types of BRDFs

When given as expressions, it is easier to comprehend the different types of BRDFs.

1. **Diffuse BRDF:** Diffuse reflection imparts a matte, non-shiny appearance to surfaces, where light striking a diffuse surface scatters uniformly in all directions in the simplest cases. The Lambertian reflectance model is widely employed to model this phenomenon. The BRDF is expressed as $f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w}) = \frac{\rho_d}{\pi}$ where ρ_d denotes the diffuse reflectance of the material. Unsurprisingly, π appears in the formula as a normalization factor and is essential for ensuring the total reflected energy remains conserved over all possible directions. This mathematical representation encapsulates the idealized behavior of perfectly diffuse reflection, implying that the surface exhibits uniform brightness when viewed from any direction.

2. **Specular BRDF:** Specular reflection, indicative of glossy surfaces

with mirror-like characteristics, produces distinct highlights that mirror the light source. Various models, such as the Phong and Cook-Torrance (microfacet) models, have been developed to describe the intricate behavior of specular reflection. The Phong BRDF is defined as $f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w}) = \frac{\rho_s(\vec{h}\cdot\vec{w})^n}{(\vec{n}\cdot\vec{h})(\vec{n}\cdot\vec{w}')}$ where ρ_s is specular reflectance, $\vec{h}$ is the half-vector, and n si the shininess exponent. On the other hand, the Cook-Torrance BRDF integrates microfacet theory, conceptualizing surfaces as collections of tiny, reflective facets. While more physically accurate, this model is more complex, offering a detailed and sophisticated representation of specular reflection in computer graphics rendering.

3. **Diffuse + Specular(Phong)BRDF:** Combining diffuse and specular reflection components results in a more comprehensive model that effectively captures the diverse appearance of various materials. The Phong BRDF with both diffuse and specular components is expressed as: $f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w}) = \rho_d\frac{\vec{n}\cdot\vec{w}'}{\pi} + \rho_s\frac{(\vec{h}\cdot\vec{w})^n}{(\vec{n}\cdot\vec{h})(\vec{n}\cdot\vec{w}')}$ The first term corresponds to the Lambertian (diffuse) reflection. In contrast, using the Phong model, the second term captures the specular reflection. This combined BRDF offers a versatile approach for rendering almost all real-life materials.

2.1.5 Properties of the BRDF

1. **Positivity**

   From a physical and practical interpretation of the BRDF, it makes sense for the function to remain positive, as it can not emit negative amounts of light.

$$0 \leq f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w}) \tag{2.12}$$

2. **Reciprocity**

Helmholtz's law of reciprocity states that the transfer of radiance of a point in a wave field is the same, regardless of the direction of the angle of the vectors. Thus:

$$f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w}) = f_r(\mathbf{x}, \vec{w} \rightarrow \vec{w}') \tag{2.13}$$

This means that light can be traced forward or backward. In a physical sense, while measuring the BRDF of a point, the light source and camera could be swapped without a noticeable difference in the value of radiance. Another notion for writing the BRDF follows.

$$f_r(\mathbf{x}, \vec{w}' \leftrightarrow \vec{w}) \tag{2.14}$$

3. **Incident and Reflected Radiance**

By manipulating the BRDF, we can find an exciting relationship between them.

$$f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w}) = \frac{dL(\mathbf{x} \rightarrow \vec{w})}{L(\mathbf{x} \leftarrow \vec{w})(\vec{\mathbf{n}} \cdot \vec{w}')d\vec{w}'} \tag{2.15}$$

First, we multiply both sides by the $L(\mathbf{x} \leftarrow \vec{w})(\vec{\mathbf{n}} \cdot \vec{w}')d\vec{w}'$ term, we end up with the following.

$$dL(\mathbf{x} \rightarrow \vec{w}) = f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w})L(\mathbf{x} \leftarrow \vec{w})(\vec{\mathbf{n}} \cdot \vec{w}')d\vec{w}' \tag{2.16}$$

By taking the integral on both sides, we see a simple formula for the relationship between the incident and reflected radiance.

$$L(\mathbf{x} \rightarrow \vec{w}) = \int_{\Omega} f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w})L(\mathbf{x} \leftarrow \vec{w})(\vec{\mathbf{n}} \cdot \vec{w}')d\vec{w}' \tag{2.17}$$

The equation describes the relationship between incoming and outgoing radiance at the point $\mathbf{x}$ on the surface. Stated it signifies the reflected radiance at $\mathbf{x}$ in the direction $\vec{w}$ is determined by considering all incident radiance arriving at that point. While the BRDF only accounts for the exitant radiance departing along the angle $\vec{w}$. This is also known as *local illumination* as it implies that local points can act as light sources, although relatively small light sources.

3. **Conservation of Energy**

   Another practical and physical interpretation of the BRDF is that the conservation of energy means the surface cannot emit more light than it receives, it is stated as.

$$\int_{\Omega} f_r(\mathbf{x}, \vec{w}' \rightarrow \vec{w})(\vec{\mathbf{n}} \cdot \vec{w}')d\vec{w}' \leq 1, \forall \vec{w} \tag{2.18}$$

2.1.6 The Rendering Equation

Having previously defined local illumination, its primary use comes in the form of the rendering equation. Within a lit room exists an ongoing equilibrium where the outgoing light of every point is equal to the emitted light. This is expressed as the outgoing radiance, $L(x \rightarrow \vec{w})$, L_e as emitted radiance, L_r as reflected radiance.

$$L(\mathbf{x} \rightarrow \vec{w}) = L_e(\mathbf{x} \rightarrow \vec{w}) + L_r(\mathbf{x} \rightarrow \vec{w}) \tag{2.19}$$

We can express the $L_r(\mathbf{x} \rightarrow \vec{w})$ in the form of the spherical integral.

$$L(\mathbf{x} \rightarrow \vec{w}) = L_e(\mathbf{x} \rightarrow \vec{w}) + \int_{\Omega} f_r(\mathbf{x}, \vec{w}' \leftrightarrow \vec{w}) L(\mathbf{x} \leftarrow \vec{w}')(\vec{\mathbf{n}} \cdot \vec{w}')d\vec{w}' \tag{2.20}$$

This expression shows that the equilibrium is recursive. This requires tracing the path of the light for several bounces in a scene, which is computationally very expensive. Fortunately, with NeRF for each point, we calculate the total reflectance from each point, and we do not have to run the recursive step. However, in some cases, which we will see later, a recursive method is used in relighting to achieve better results.

2.2 Neural Radiance Fields

2.2.1 Plenoptic Function

In 1845, Michael Faraday was accredited with proposing light as a part of the electromagnetic field. Later on, in 1936, Arun Gershun proposed the term light field in his theory of the *Plenoptic Function*. The Plenoptic Function is a 5-dimensional function comprising of 3 coordinate positions (x, y, z) and two viewing spherical coordinates (θ, ϕ). The original Plenoptic Function was 7-dimensional with three coordinate positions (x, y, z) and two viewing spherical coordinates (θ, ϕ) with additional parameters for time and wavelength (t, λ). This function could theoretically render all possible scenes that can be composed. For this thesis, we will only use the 5D plenoptic function express as an implicit MLP.

2.2.2 Neural Fields

A recent phenomenon that is the main focus of this thesis is the use of MLPs to represent fields implicitly. A field is a region where all points are well-defined, and images can be considered discrete fields. For example, we can consider the function $x \in \mathbb{R}^2 \rightarrow c \in \mathbb{R}^3$,

which takes a pixel coordinate and outputs RGB color. We can extend this idea into higher dimensions, $x \in \mathbb{R}^3 \rightarrow c \in \mathbb{R}^3, \alpha \in \mathbb{R}^1$ where alpha is added to describe the opacity of a 3D point.

2.2.3 Coordinate-based Multilayer Perceptrons

Coordinate-based Multilayer Perceptrons are the best way to encode a field into an MLP, but some drawbacks will be discussed later. In some cases, it requires less memory to store the weights of the MLP than to store it in an explicit representation. That can unlock the potential of neural networks shown in the monumental Instant Neural Graphics Primitives with a Multiresolution Hash Encoding paper. [12] The paper also covers Neural gigapixel images, Neural volumes, and Neural SDF (Signed Distance Functions). Signed Distance Functions where the distance d is 0 on the surface, greater than one outside the surface, and less than 0 inside. Although they are not used in this thesis, their importance should be remembered, as finding the distance to the surface is a crucial factor in relighting.

2.2.4 Neural Radiance Fields: NeRF

Proposed by Mildenhall *et al* NeRF: representing scenes as neural radiance fields for view synthesis, was pixel breaking in our abilities to reconstruct 3D scenes. [11] NeRF is an overfitting method where each MLP fits a scene from scratch. This has the added benefit of high fidelity but requires new training if the scene changes, even in the slightest. As stated in by Mark Boss in his PhD thesis:

Each training step selects a stochastic batch of pixels and the corresponding camera rays. Samples are then placed along the rays.

The sample positions are evaluated in the current field and aggregated with volume rendering. The resulting color for each ray can be compared with the input. This process is repeated over several hundred thousand steps, and the neural field starts to replicate the input scene slowly. This process is similar to the volumetric reconstruction of computer tomography. [2]

More specifically, the method is training a five-dimension plenoptic function. This function takes in a position vector and viewing direction $x, d \in \mathbb{R}^{3,2} \rightarrow c, \alpha \in \mathbb{R}^{3,1}$ and outputs color and density.

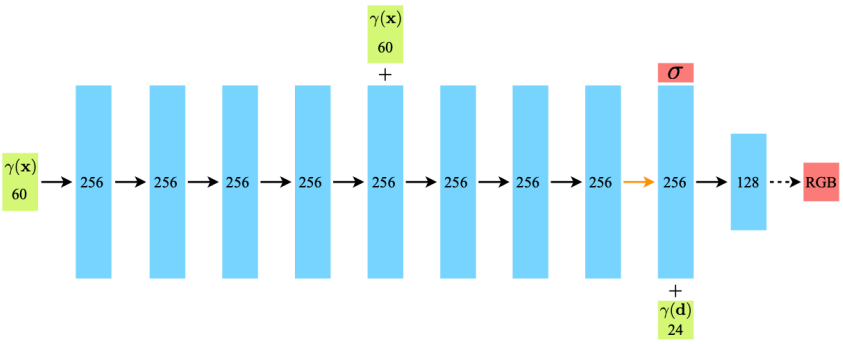

Figure 2.5: NeRF Architecture Model
Figure from [11]

The $\gamma(x)$ and $\gamma(d)$ stand for the positional encoding of the vectors. The transformer paper inspired this positional encoding. [17] This encoding is also recognized as the Fourier Encoding it encodes over a sinusoidal function. It is rather simple:

$$\gamma(x) = (x, \Gamma_1, ..., \Gamma_{L-1}) \tag{2.21}$$

$$\Gamma_k(x) = [sin(2^k x), cos(2^k x)] \tag{2.22}$$

Despite the low-frequency nature of this encoding, the network

demonstrates an enhanced ability to acquire high-frequency information. Tancik et al. [15] explore this characteristic by employing techniques derived from the neural tangent kernel (NTK) literature to show how spatial bias can be overcome. [5] An overlooked aspect of the architecture, a common theme later for relighting, is the input and output of different variables. Notably, the input of the viewing direction is usually in the last layers due to specular reflections having a minor impact on the total image, as seen in figure 2.6.

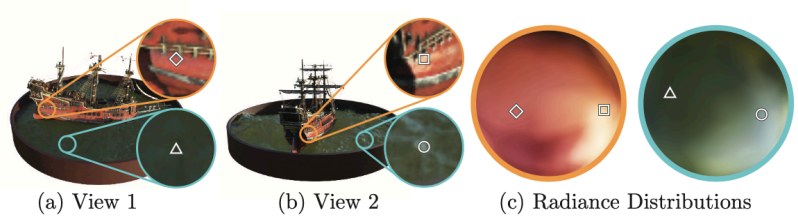

Figure 2.6: View Dependent Lighting
Figure from [11]

By leveraging view-dependent lighting, NeRF has a significant advantage over other photogrammetry techniques. Figure 2.6 demonstrates how a color difference can occur at the same point when looking at viewing directions (a) and (b).

NeRF renders each scene using volumetric rendering techniques to combine the different colors and densities. Each ray has around 100 to 200 point values that need to be evaluated and summed; however, sampling each point is immensely costly. Therefore, new sampling techniques are continuously being developed; the original paper uses hierarchical sampling, with two different neural networks trained simultaneously. Along the ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ the samples of t_i are picked from the following distribution. Where t_m and t_f are the far and near bounds, N is the number of samples to be taken.

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right] \tag{2.23}$$

For each pixel in a photo, a ray is shot out, and the colors are aggregated; this process must remain differentiable, as seen in Figure 2.6. The first step (a) is figuring out which points need to be queried based on the camera model, and this would result in the set of points of type (x, y, z, θ, ϕ). In step (b), these locations are fed into the MLP, as shown in Figure 2.5. In step (c), the differentiable volume rendering happens, which will be discussed later. Finally, in step (d), the color produced by volume rendering is compared in a rendering loss. Unlike most other Deep Learning (DL) solutions, our goal here is actually **overfit** the MLP to produce a perfect replica of the scene. It should also be stated that MLP is a continuous volumetric scene function; many points will never be fed into the MLP, and it also has to function as a regressor.

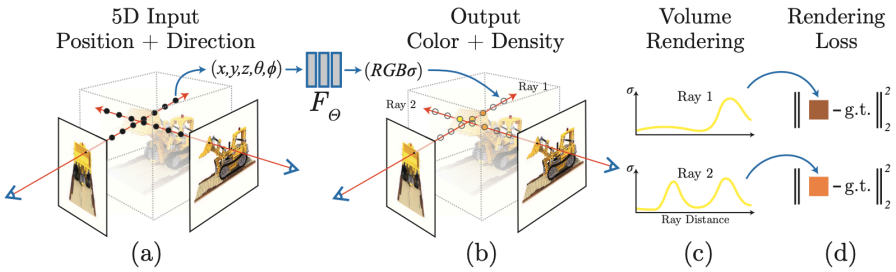

Figure 2.7: Four Step NeRF Pipeline
Figure from [11]

The crux of the NeRF algorithm is the volume rendering equation. At first glance, the equation is complicated to comprehend, but it comprises three components, each with its function. The simplest is the $\sigma(r(t))$, which is the density scalar value from MLP at the point $r(t)$. The second is $c(r(t), d)$, which is the RGB color value from the MLP at the point $r(t)$ and viewing directions d. It is no

surprise that these values are multiplied together as density should determine the brightness of color. Nevertheless, this method treats occluded terms with the same significance along the ray, posing an issue as our primary interest lies in the color values that are visibly present in front of the scene. The function $T(t)$ denotes the accumulated transmittance, which is the probability that the ray travels from t_n to t without hitting any other particle. [11] The parameter of this function is only t which is also its integral limit. The function can also be interpreted as $1/e^{\text{sum of densities}}$, which means $T(t)$ is exponentially monotonically decreasing. Ideally, this function would remain one but drop to zero after the first occurrence of a surface is reached. Then, the occluded colors and densities will not be visible.

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)dt, \text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))ds\right) ds \tag{2.24}$$

The next important idea is the discretization of this formula. The term $\delta_i = t_{i+1} - t_i$ which is a discrete delta. Secondly, the term $1 - exp(-\sigma_i\delta_i)$ is added to replace σ. Known as alpha compositing, this formula also has a minimum and maximum of 0 and 1, which is more suitable.

$$\hat{C}(r) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i\delta_i))c_i, \text{where } T_i = exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right) \tag{2.25}$$

However realistic NeRF might be, it still overfits one scene with baked-in lighting. This functionality dramatically limits its capabilities and requires a complete architectural change. One benefit

is that the geometry is consistently stored within the neural field; view-dependent colors can be interpolated, but relighting requires further thought. Nevertheless, NeRF was a giant leap forward regarding realistic image quality when compared to previous methods such as Local Light Field Fusion (LLFF) [10] and Scene Representation Networks (SRN) [13].

Figure 2.8: NeRF Compared to LLFF and SRN
Figure from [11]

Chapter 3

Related Work

The literature review comprises four fundamental papers that initiated the NeRF relighting revolution. In no order of importance, the papers that will be discussed in this chapter include Neural Reflectance Fields for Appearance Acquisition [1], NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis [14], NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination [20], NeRD: Neural Reflectance Decomposition From Image Collections [3].

3.1 Neural Reflectance Fields for Appearance Acquisition

3.1.1 Introduction

Compared to traditional methods, Neural Reflectance Fields' novel aspect lies in their ability to encode volume density, normal, and reflectance properties at any 3D point in a scene using a fully connected neural network. This representation can accurately model the appearance of real-world scenes with complex geometry and reflectance and render photo-realistic images under novel viewpoints and non-collocated lighting conditions. It can be estimated from images captured with a simple collocated camera-light setup, allowing for the rendering of images from new viewpoints and lighting conditions that were never captured. Neural reflectance fields also enable relighting and other image synthesis applications and can model complete scene appearance, including challenging effects like specularities, shadows, and occlusions.

3.1.2 Reflectance Equation

A problematic aspect of these papers is they tend to be similar but slightly different notations. Standardizing their notation would be too

tricky, and a limited summary would be given when required. The volume rendering equation generally computes the radiance $L(c, \omega_o)$, c is typically the camera point, and ω_o is a direction.

$$L(c, \omega_o) = \int_0^\infty \tau_c(x)\sigma(x)L_s(c, \omega_o)dt \tag{3.1}$$

$L_S(x, \omega_o)$ represents the scattered light at x along ω_o, σ is the extinction coefficient that indicates the probability density of medium particles (volume density), $\tau_c(x)$ represents the transmittance factor which determines the loss of light along the ray from c to x.

Considering single-bounce direct illumination under a single point light source to approximate L_s is essential in the context of neural reflectance fields because it allows for the accurate modeling of the incident radiance. By incorporating the explicit reflectance term and considering the loss of light due to extinction through the volume, the rendering equation can accurately capture the interaction of light with the scene. This approach enables the neural reflectance fields to represent geometry and reflectance, producing high-quality rendering under different lighting conditions. By accounting for single-bounce direct illumination, the rendering equation becomes reflectance-aware, allowing for a more comprehensive understanding of the light transport in the scene. This is done so in the following equations. First, they have to modify the L_s to Eqn. 2.1

$$L_S(x, \omega_o) = \int_{\mathcal{S}} f_p(x, \omega_o, \omega_i)L_i(x, \omega_i)d\omega_i \tag{3.2}$$

This is once again a sphere integral over hemisphere $\mathcal{S}$, f_p is a phase function that governs light scattering, and $L_i(x, \omega_i)$ is the incident radiance arriving at x from direction ω_i.

Since L_s is baked in NeRF their approach, therefore, the new L_s considers single-bounce direct illumination under a single-point light source

where f_r represents a differentiable reflectance model with parameters $R(x)$, n is the local surface shading normal, and L_i represents the incident radiance.

$$L_S(x, \omega_o) = f_r(x, \omega_o, \omega_i, n(x), R(x)) L_i(x, \omega_i) \tag{3.3}$$

$$L_i(x, \omega_o) = \tau_l(x) L_l(x) \tag{3.4}$$

Where τ_l is the transmittance from the light to the shading point, and L_l represents the light intensity with the consideration of distance attenuation. Here, l denotes the position of the point light source.

The original NeRF equation $C(r) = \int T(t)\sigma(x)c(x,d)dt$ does not encode the light transmittance along an additional ray toward the light (yellow ellipsoid). However, it does encode the $L_S(x, \omega_o)$, which does not allow access to reflectance properties, making relighting impossible.

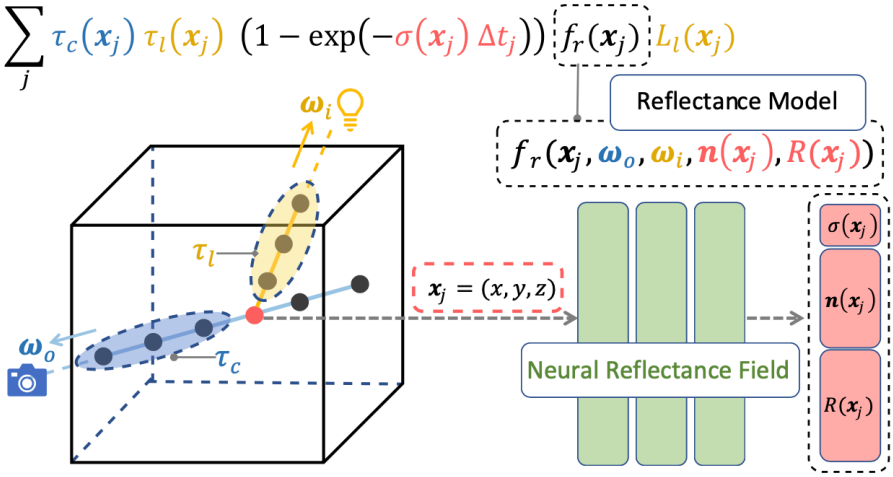

Figure 3.1: One Bounce Reflection Diagram
Figure from [1]

3.1.3 Limitations

One of the main limitations is the long training time. The authors mention that training each reflectance field network takes about two days to train using four NVIDIA RTX 2080Ti GPUs. Although the network takes about 30 seconds to render a 512 x 512 image at inference time, the training time is still a significant limitation.

3.2 NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis

3.2.1 Introduction

The NeRV method, developed by Google, MIT, and UC Berkeley researchers, introduces a novel approach to scene representation and rendering. It can synthesize 3D representations of scenes from images illuminated by unconstrained known lighting. NeRV can produce output that enables rendering from novel viewpoints under arbitrary lighting conditions, including indirect illumination effects. This capability sets it apart from prior methods and allows relightable 3D scene representations. The predicted visibility and surface intersection fields play a critical role in simulating direct and indirect illumination during training, particularly in complex lighting settings. NeRV's ability to handle such difficult lighting conditions sets it apart from alternative approaches.

3.2.2 Redering Equations

NeRV requires a set of images with known lighting conditions and recovers a 3D representation that can be rendered from novel viewpoints and multiple lighting conditions. Their method requires four MLPs: a *shape*

that outputs volume density σ, a "reflectance" that outputs BRDF parameters (3D diffuse albedo a and 1D roughness γ), a "visibility" for visibility field approximations and "distance" for surface distance approximation. The surface distance approximation is perhaps the most essential, especially indirect illumination. Sampling the direct visibility of light source l of each n point along the ray is in $\mathcal{O}(n^2 l)$ while using an approximation MLP makes it only $\mathcal{O}(nl)$, sampling a one-bounce indirect light source is in $\mathcal{O}(n^3 dl)$ while using approximation methods is $\mathcal{O}(n + dl)$. Their crucial insight is approximating distance and visibility fields via MLPs to reduce render times significantly under new lighting conditions, as seen in the figure 3.2.

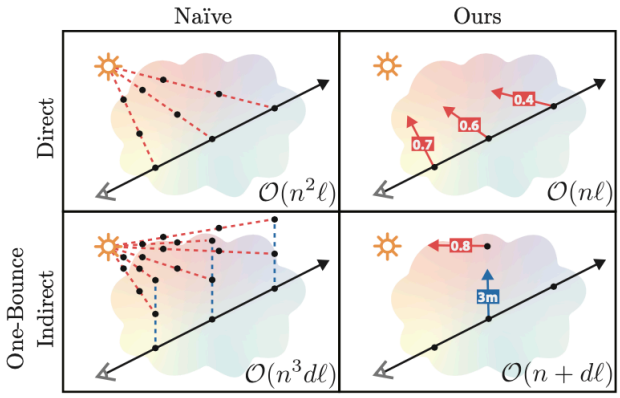

Figure 3.2: Runtime Relation of Different Approximation Methods
Figure from [14]

The two essential MLPs are the visibility and distance approximators ($\tilde{V}_\phi$, $\tilde{D}_\phi$), for a given (x, ω) they estimate the equations below to reduce heavy line integral calculations. E is the known light source and can be queried efficiently.

$$V(x, \omega) = exp\Big(- \int_0^\infty \sigma(x + s\omega) ds \Big) \tag{3.5}$$

$$D(x, \omega) = \int_0^\infty exp\Big(- \int_0^t \sigma(x + s\omega) ds \Big) t\sigma(x + s\omega) dt \tag{3.6}$$

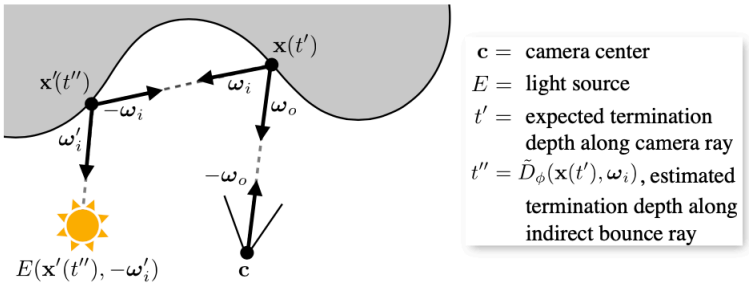

Figure 3.3: Indirect Illumination Path of NeRV
Figure from [14]

Instead of calculating the distance to each surface and its visibility of the known light source, the reflections also use the normals of each point, which are analytically computed.

In addition to the $\tilde{V}_\phi$, $\tilde{D}_\phi$ MLPs they require the shape $MLP_\theta(x(t)) \rightarrow \sigma$ and "reflectance" $MLP_\psi(x(t)) \rightarrow (a, \gamma)$. The normal is calculated s.t. $n = \nabla_x MLP_\theta(x(t))$, and with these, we can render an image with single bounce indirect light with the following steps.

1. Sample each ray across $x(t) = c - t\omega_c$ and query the shape and reflectance MLPs for the volume densities, surface normals, and BRDF parameters.

2. Shade each point along the ray with **direct illumination**. The visibility computes this, Light source E, and BRDF values at each point.

3. Shade each point along the ray with **indirect illumination**. Use the predicted endpoint $x(t")$ and compute its illumination effects, this time in the direction of $x(t')$. Then, calculate the illumination effects from $x(t')$ in the camera's direction.

4. The total reflected radiance at each point along the camera ray is the sum of all the direct and indirect illumination.

A single equation $L(c, \omega_o)$ can also describe these steps. This would be a near-perfect illumination model; however, it's too costly. When looking

at the more significant equation, it's understandable why approximations are necessary.

$$L(c,\omega_o) = \int_0^\infty V(x(t),c)\sigma(x(t)) \int_S \tilde{V}_\phi(x(t),\omega_i)E(x(t),-\omega_i)R(x(t),\omega_i,\omega_o)\,d\omega_i\,dt$$
$$+ \int \int_S \tilde{V}_\phi(x'(t''),\omega_i')E(x'(t''),-\omega_i')R(x'(t''),\omega_i',-\omega_i)\,d\omega_i' R(x(t'),\omega_i,\omega_o)\,d\omega_i \tag{3.7}$$

It's better to abstract each equation into the function that it operates in. The indirect illumination path has very discrete steps. Finding the distance to a point is integral, which is why the equation is so large and might cause confusion. However, it is still possible to define each function as such.

- $\tilde{V}_\phi(x(t),\omega_i)E(x(t),\omega_i)R(x(t),\omega_i,\omega_o)d\omega_i$: Is the direct illumination from a point into the camera.
- $\tilde{V}_\phi(x'(t"),\omega_i')E(x'(t"),-\omega_i')R(x'(t"),\omega_i',-\omega_i)d\omega_i'$: Is the indirect illumination of the light that is expected to reach t' (expected termination depth)
- $R(x(t'),\omega_i,\omega_o)d\omega_i)$: Takes the indirect light arriving at t' and directs it to the camera according to the BRDF.

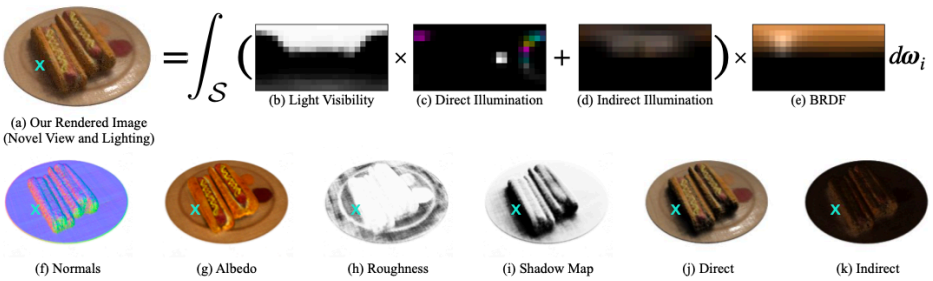

Figure 3.4: Indirect Illumination Path of NeRV
Figure from [14]

In figure 3.4 the authors visualize the visibility to a spherical environment map by sampling $\tilde{V}_\phi$; direct illumination is given, and they

estimated indirect illumination at that point to determine the full incident illumination; this is then multiplied by the BRDF to calculate the light is the direction of the camera. Given any continuous 3D location as input, such as the point at the cyan "x".

3.2.3 Limitations

The most significant limitation of NeRV is that it requires training for 1 million iterations using 128 TPU cores. It also has a complicated training recipe that requires the $\tilde{V}_\phi$, $\tilde{D}_\phi$ MLP to be trained extensively on their own. The unique ability to calculate the measurement of indirect illumination is commendable. However, the computational costs associated with it are immense.

3.3 NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination

3.3.1 Introduction

NeRFactor is a method for factorizing images of an object under an unknown lighting condition into shape, reflectance, and illumination, thereby supporting free-viewpoint relighting and material editing. Its main technical contributions include a method for joint estimation of surface normals, light visibility, albedo, spatially-varying BRDFs, and environment lighting from multi-view images of an object. NeRFactor also introduces a strategy to distill volume density into surface geometry for normals. Also, a novel data-driven BRDF prior learned from actual measured BRDFs. The unique features that set NeRFactor apart from other methods include its ability to handle one unknown illumination

condition, successfully separate shadows from albedo, model spatially-varying BRDFs, and use data-driven priors to disambiguate the most likely factorization of the scene. NeRFactor enables realistic relighting and material editing, even under challenging lighting conditions.

NeRFactor's architecture is relatively simple and intuitive. It consists of different blocks with their respective function; this function enforces careful selection of inputs and outputs in the pipeline and, more importantly, through clever loss functions. The first half of a standard NeRF is initialized to predict the surface volume. Additional visibility, BRDFs, albedo, and normal MPLs are included too. NeRF-estimated volume density into surface geometry (with normals and light visibility) is used as an initialization when improving the geometry and recovering reflectance, and then smoothing is jointly optimized in the loss function. A novel data-driven BRDF prior learned from training a latent code model on real measured BRDFs is also utilized.

3.3.2 Architecture and Training

The MLPs can be grouped into three training methods: frozen, pretrained, pretrained, jointly finetuned, and trained from scratch. The Normal and Light Visibility MLPs can be trained independently to give a high-quality estimate and initialize their weights. Still, they will be jointly finetuned for smoothing and final rendering. The BRDF MLP is trained upon the MERL dataset, while the Albedo and BRDF Identity MLP output a latent variable that can be edited later for material editing.

Shape

Normal and Light Visibility MLPs can be trained separately to provide high-quality estimates. After training them independently to reproduce

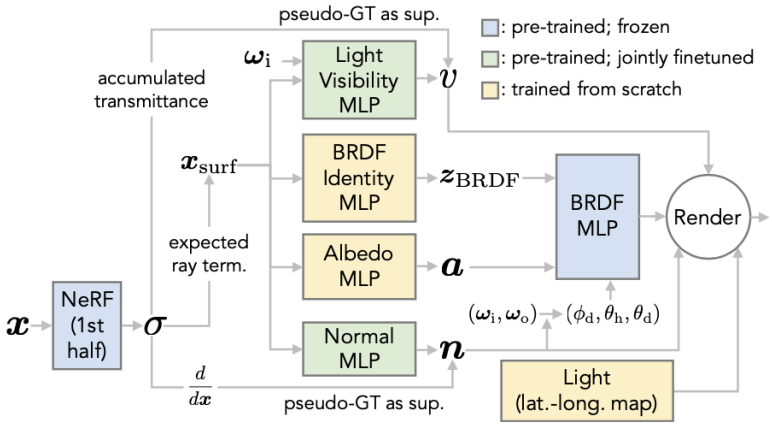

Figure 3.5: Architecture of NeRFactor
Figure from [20]

the values from the NeRF geometry without any smoothness regularization or re-rendering loss, their weights are finetuned jointly for smoothing and final rendering. This process reasonably initializes the visibility maps, preventing albedo or Bidirectional Reflectance Distribution Function (BRDF) estimation artifacts. Below n_a and v_a are the so-called analytical normal and visibility functions derived from the standard NeRF. f_n and f_v are initialized as n_a and v_a at first then finetuned on this loss. The second term in each loss can be referred to as the smoothing loss, while the first term ensures that the functions don't drift too far away from their main purpose.

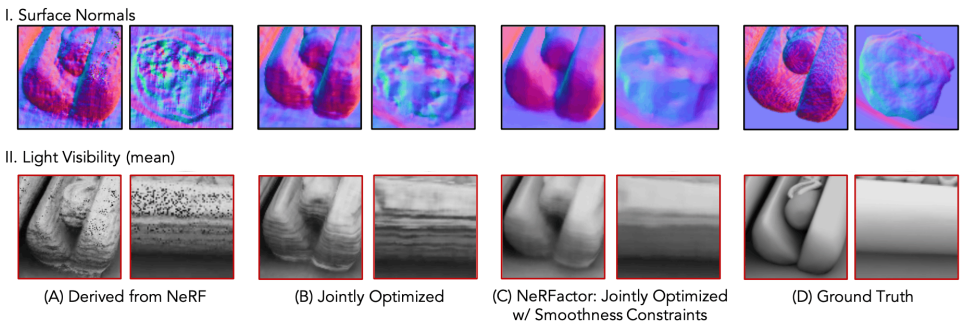

Figure 3.6: Surface Normals and Topology with Smoothing Loss
Figure from [20]

$$l_n = \sum_{x_{surf}} \left(\frac{\lambda_1}{3} ||f_n(x_{surf}) - n_a(x_{surf})||_2^2 + \frac{\lambda_2}{3} ||f_n(x_{surf}) - f_n(x_{surf} + \epsilon)||_1 \right) \tag{3.8}$$

$$l_v = \sum_{x_{surf}} \sum_{\omega_i} \left(\lambda_3 (f_v(x_{surf}, \omega_i) - v_a(x_{surf}, \omega_i))^2 + \lambda_4 |f_v(x_{surf}, \omega_i) - f_v(x_{surf} + \epsilon, \omega_i)| \right) \tag{3.9}$$

Reflectance

f'_r is parameterized BRDF trained on the MERL Dataset to learn specular components. It is trained to learn the latent space of real-world BRDFs but is kept frozen. f_z is then trained from scratch to predict the real world z_{BRDF} and can be changed later to material editing. f_a is responsible for generating the diffuse color component. While l_a and l_z ensure smooth functions, most gradient updates should come with reconstruction loss.

$$l_a = \lambda_5 \sum_{x_{surf}} \frac{1}{3} ||f_a(x_{surf}) - f_a(x_{surf} + \epsilon)||_1 \tag{3.10}$$

$$l_z = \lambda_6 \sum_{x_{surf}} \frac{||f_z(x_{surf}) - f_z(x_{surf} + \epsilon)||_1}{dim(z_{BRDF})} \tag{3.11}$$

$$R(x, \omega_i, \omega_o) = \frac{f_a(x)}{\pi} + f'_r(f_z(x), g(f_n(x), \omega_i, \omega_o)) \tag{3.12}$$

3.3.3 Limitations

Dealing with one unknown light source is impressive but not what's required in a controlled environment. This also creates one of its most significant limitations in Light Probe Image Resolution: The resolution

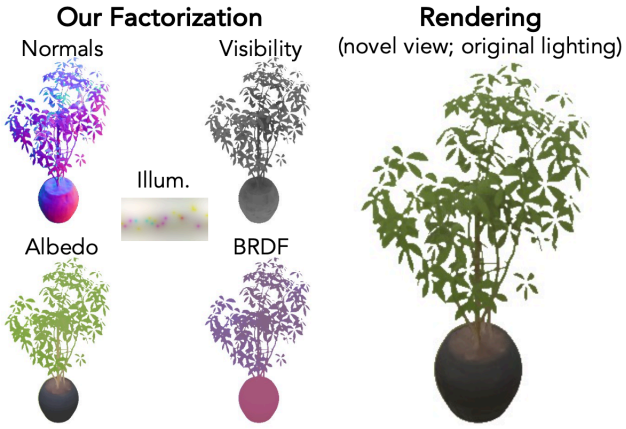

Figure 3.7: Factorized Plant from NeRFactor
Figure from [20]

of the light probe images is limited to 16×32, which may not be sufficient for generating tricky shadows or recovering very high-frequency BRDFs. This can lead to specularity or shadow residuals in albedo estimation under specific illumination conditions.

3.4 NeRD: Neural Reflectance Decomposition From Image Collections

3.4.1 Introduction

NeRD is a collaboration between the University of Tubingen and Google Research. It introduces a novel approach to scene representation and rendering. It can decompose a scene into shape, color, metallic, roughness, and normals, all view and illumination dependant. NeRD can produce output that enables rendering from novel viewpoints under arbitrary lighting conditions, including indirect illumination effects. It can also extract a relightable textured mesh from the learned neural volume, enabling fast real-time rendering with novel illuminations. The effectiveness of the proposed approach is demonstrated on both synthetic and

real datasets, where it can produce high-quality, relightable 3D assets from image collections.

NeRF and NeRD have similar course networks for estimating where the fine network should query. The Fine Network or Decomposition network is what does the actual rendering with an extra m samples around σ. Lighting comprises 24 lobes of spherical Gaussian mixtures ($24 \times 7 = 168$ parameters). These are differentiable and efficient to render. An encoder-decoder network computes the BRDFs to compress the latent space severely. This comes from the assumption that the number of materials in a scene is few. The scenes are decomposed into base color, metallic, roughness, and normals. The analytical calculation of normals here was later used in NeRFactor.

3.4.2 Sampling Network

The main task of the coarse sampling network is to generate a finer distribution for sampling in the decomposition network. N_{Θ_2} shown in figure 3.8 is a Network that reduces the dimensionality of the Gaussian illumination spheres Γ^j. This does not give the final light of the scenes determined by the decomposition network. These are then concatenated with the N_{Θ_1} network values, which produce color with illumination. N_{Θ_3} has the final color on the point x_i along the ray, composed of the other points to give the pixel color.

3.4.3 Composition Network

The intermediate decomposition step is responsible for the BRDF parameters, base color, and normals. Note that all of these are view and illumination-independent. The N_{Φ_1} as seen in figure 3.9 outputs (RGB+σ) which is also passed to N_{Φ_2} a [36, 16, 2, 16, 16, 5] MLP no activation network. The final five dimensions are for the analytical Cook

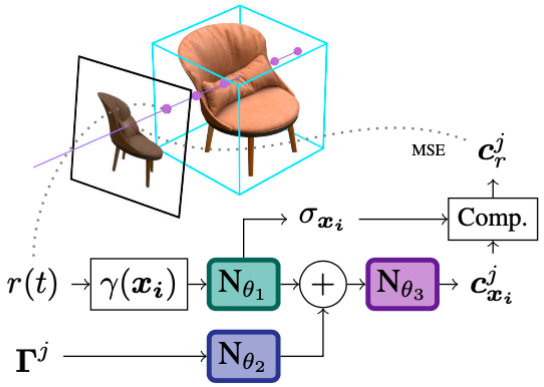

Figure 3.8: Course Sampling Network from NeRD
Figure from [3]

Torrance BRDF model [4]. After the points n_{x_i}, d_{x_i} and b_{x_i} are composed into n_r, d_r and b_r and are rendered with illumination and view direction being considered. To keep the lighting of spherical Gaussians Γ^j differentiable, the reflectance due to diffuse and specular lobes is separately evaluated by ρ_d and ρ_s, from the paper *All-frequency rendering of dynamic, spatially- varying reflectance.*

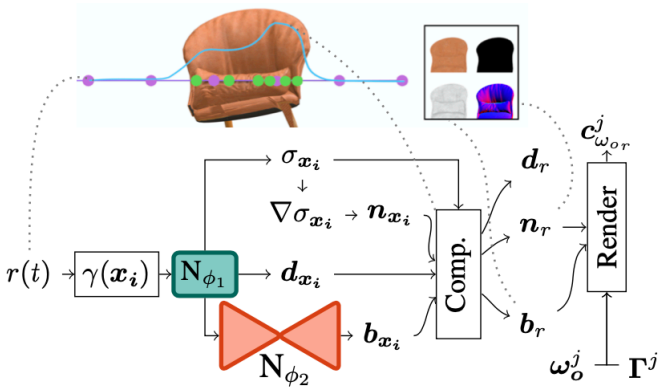

Figure 3.9: Fine Decomposition Network from NeRV
Figure from [3]

3.4.4 Limitations

On 4 NVIDIA 2080 Ti, the training takes about 1.5 days. The final mesh extraction takes approximately 90 minutes. It struggles to accurately recover detailed illumination, which can limit the realism of the results. Additionally, evaluating an object's radiance fields, a crucial part of the technique, can be resource and time-intensive. Despite these challenges, NeRD has shown potential in decomposing scenes into their shape, reflectance, and illumination and enabling fast real-time rendering with novel illuminations. This method requires no particular illumination setup and can have multiple light sources. Spherical Gaussians are much simpler to calculate than the lighting equation and don't require an approximation formula. Since the lighting is done in a physics engine, this also allows for hard shadows.

Chapter 4

Methodology

4.1 ReNe: ReLight My NeRF

ReNe aims to solve the problem of rending under unobserved lighting conditions. During their research era, this was only possible by creating a new dataset dubbed ReNE (Relighting NeRF), which frames real-world objects under one-light-at-time (OLAT) conditions alongside accurate ground-truth camera and light poses. The ground truths are essential for supervised learning, and the shift from synthetic to real-world allows us to test the actual merit of our rendering algorithms.

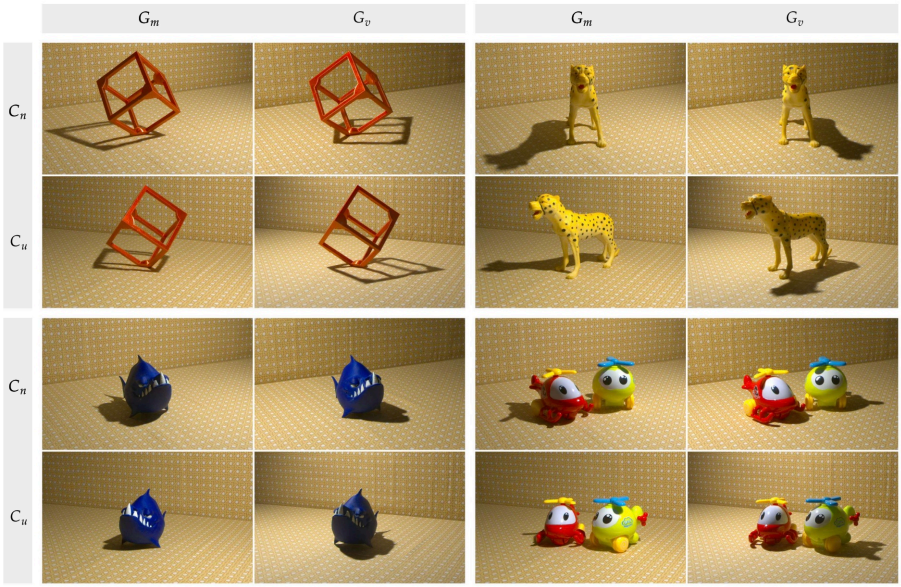

Figure 4.1: The ReNe Dataset with Varying Poses and Lights
Figure from [16]

Their data acquisition frame consists of a LightBot and CameraBot, which move the light and camera, respectively. Their trajectories must also be non-intersecting in the sections of the hemisphere of which they traverse. This can then be visualized in a grid of images. Each row depicts the same viewpoint as the light conditions change, and each column depicts the same light conditions as the viewpoint changes. Each column usually represents a NeRF trained since lighting is a necessary constant. Rene is a dataset of 40,000 images acquired from 50 different points of

view under 40 different OLAT conditions of 20 unique objects. It was the first to have multiple categories: real-world, background shadows, public, and light supervision. Each object has a total of 2000 images, which is also partly why the training takes so long; however, as relighting demands, there are no shortcuts to quality data.

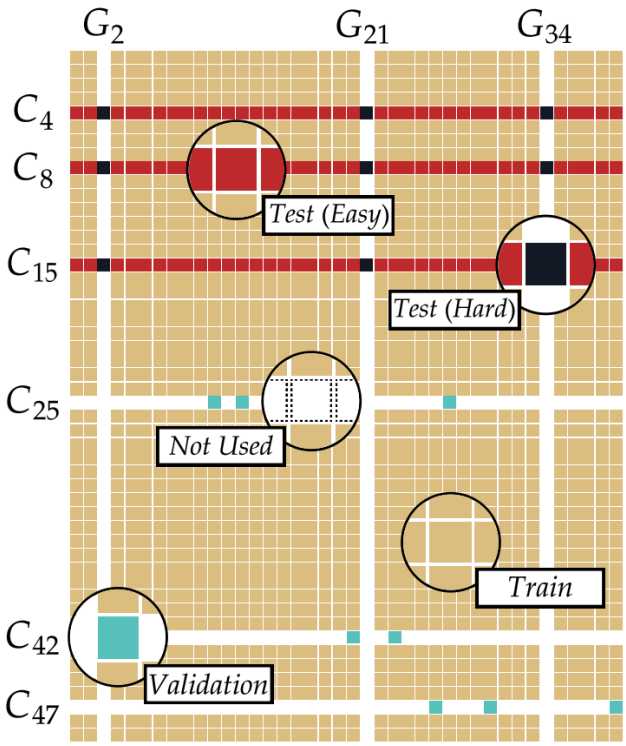

Figure 3. **Dataset splits.** Each scene $\mathcal{I}_s$ is divided in 4 different splits: ■ Samples used for training ■ Samples used for validation ■ Samples used for Easy Test ■ Samples used for Hard Test □ Samples never used.

Figure 4.2: Grid Sample of the ReNe dataset
Figure from [16]

Following the dataset creation, the authors developed a new architecture inspired by the existing NGP [12] model. This was done effectively in an iterative approach to address variations in lighting conditions. They start from the most naive approach, V0, and go to their most advanced, V5. Their architecture is comprised of Ψ_{geo}, Ψ_{rgb} and Ψ_{vis}. Ψ_{geo} takes in an encoded x using a multiresolution hash encoding called $h(x)$ it gives

a density σ and normal e. Ψ_{rgb} takes in $h(x)$ and e with the addition of spherical harmonically encoded $\zeta(d)$ of viewing directions and $\zeta(l)$ of direction to the light and give the RGB color. Meanwhile, the Ψ_{vis} takes in the $\zeta(l)$ and $h(x)$ and gives a scalar value to create a shadow. It is a clever way to create an independent Ψ_{vis}, which acts like a shadow network and can be used to generate shadow maps. What does this function do? It tries to predict visibility, which is a function of geometry. To make things worse, MLP does so along an entire ray, which we know is hard to generalize. More importantly, that is only the described function.

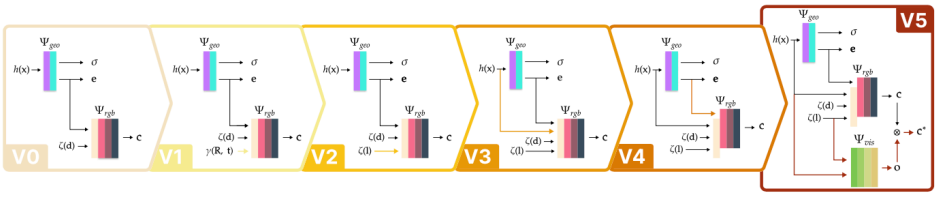

Figure 4.3: The Evolutions of ReNe Architectures
Figure from [16]

The V5 visibility MLP takes in $\zeta(l)$ and $h(x)$. The first red flag should be $h(x)$. Shadows should be based on geometry, not on position; injecting positional data with no geometric data will cause the shadow network to learn patterns of general darkness, i.e., a checkerboard. While their intention is commendable, they also incorporate $\zeta(l)$, aiming to convey information about the specific location of the light source for each point. So what makes this so hard to generalize to unobserved lighting conditions, and why does V5 perform poorly on the hard test set? To do its task correctly, the Ψ_{vis} network has become to memorize global geometry. Due to visibility being a mighty difficult task, we only expect to predict the σ of that point at every point; instead, for visibility, we expect it to predict it at all points along an entire ray. The visibility function does remarkably well in observed light conditions. It can memorize the lighting condition thanks to $\zeta(l)$ and then use the position x to

predict the shadow. Otherwise, it should perform equally when given a new unobserved lighting condition, as seen in Figure 4.4.

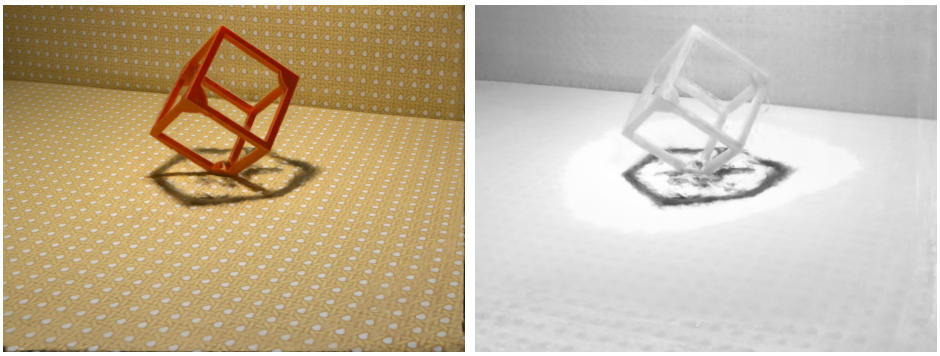

Figure 4.4: Struggles of V5 Network
RGB image on the left and the visibility MLP shadow predictions on the right.

4.2 ReNe with Geometric Shadow Hints

After reviewing ReNe's results and comparing them with the literature review's results, it became evident that any method that relights the shadows properly must consider the global geometry. As we have also witnessed, this is prohibitively expensive, so a hybrid solution is needed.

For a given OLAT light position, it is possible to conclude that all the rays point in the same direction. To calculate any occlusion for visibility, it is impossible to cast a ray from the point to the light source but from the light source to the point. Shadow mapping is a technique used to create realistic, low-cost shadows. It works by rendering the scene from the viewpoint of a light source to create a *shadow map* that represents the depths of objects from that light's perspective (pre-computed). Then, during the regular scene rendering, this shadow map is used to determine which parts of the scene are in shadow and which are not, allowing for the accurate rendering of shadows. This can create a radical speed-up in performance as we can recalculate a set of rays in the form of a distance map.

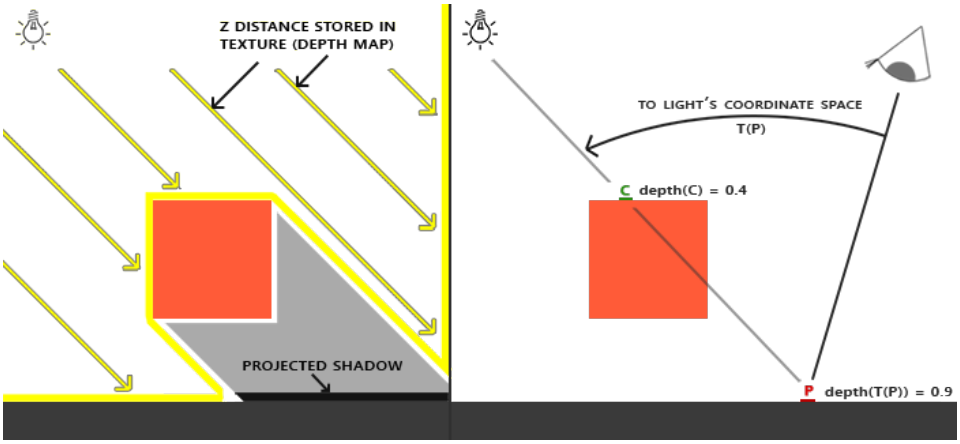

Figure 4.5: Shadow Mapping Diagram
Figure from learnopengl.com

In classic computer graphics, shadow mapping comes with its own set of problems. Aliasing artifacts, a limitation of shadow mapping, are shown as jagged and pixelated edges in rendered shadows due to insufficient resolution in the shadow map. This issue arises when the discrete nature of the shadow map's depth values fails to accurately represent the intricate details of shadow-casting geometry, leading to visual distortions. Light bleeding is another obstacle, causing shadows to appear lighter than expected due to inaccurate stored depth values, resulting in unrealistic illumination in shadowed areas. The limited resolution further compounds these problems; creating resolutions that are too high would require additional memory. Lastly, viewpoint dependency presents challenges when the camera perspective changes; this requires each light source to have its depth map; for dynamic light sources, this is almost impossible to create in real time. These limitations collectively underscore the complexity and ongoing efforts to enhance the realism and accuracy of shadow mapping in computer graphics.

All of these problems result from having explicitly defined what makes a shadow. However, with implicit models, we can let the neural network deal with these aspects in training. It is still vital to give the neural network accurate measurements concerning where a shadow might be.

As a precaution, we did many sanity checks to ensure our measurements were correct. The first test was to see if the distance map from the view of the light was geometrically accurate. The actual distance was relatively easy to calculate as it was the sum of the already calculated $T(t)\sigma(r(t))$ values. We are visualizing the distance map from the viewpoint of the light source on the right. If we closely examine the cube's cast shadow, we can notice that it is precisely the cube's silhouette from the light's viewpoint.

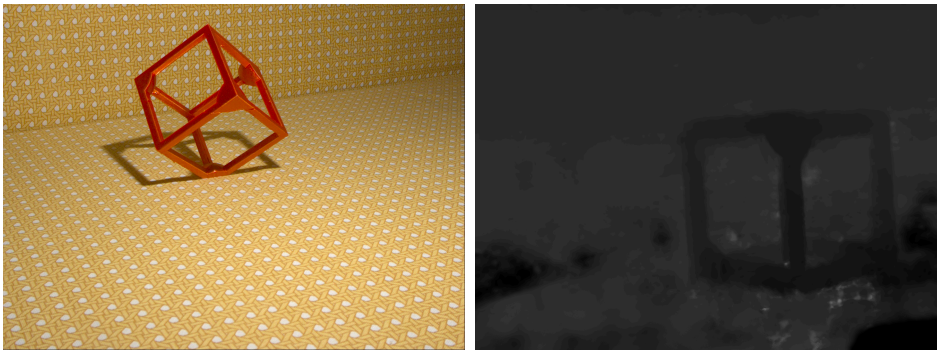

Figure 4.6: Distance Map From Light Source

Since the ReNe dataset lacks a 360 view around the object, it struggles with highly accurate distance maps from light sources that are remote from the camera positions. Initially, we believed this would cause significant damage to the quality of the shadow maps, but the MLP was remarkably able to learn past this. To see what the MLP is learning, we can visualize the shadow hints by setting a threshold for which pixels should remain the same and which should turn black. There is much noise included in the shadow hints.

The images on the left show ground truth, while those on the right show where a cast shadow should fall according to the shadow mapping algorithm and provided distance map. To speed up training time for this sanity check, we trained on one of the OLAT scenes, which is why, for all examples, there is a baked-in shadow. Regardless, it can pick up an idea about geometry using simple linear algebra rather than trying to

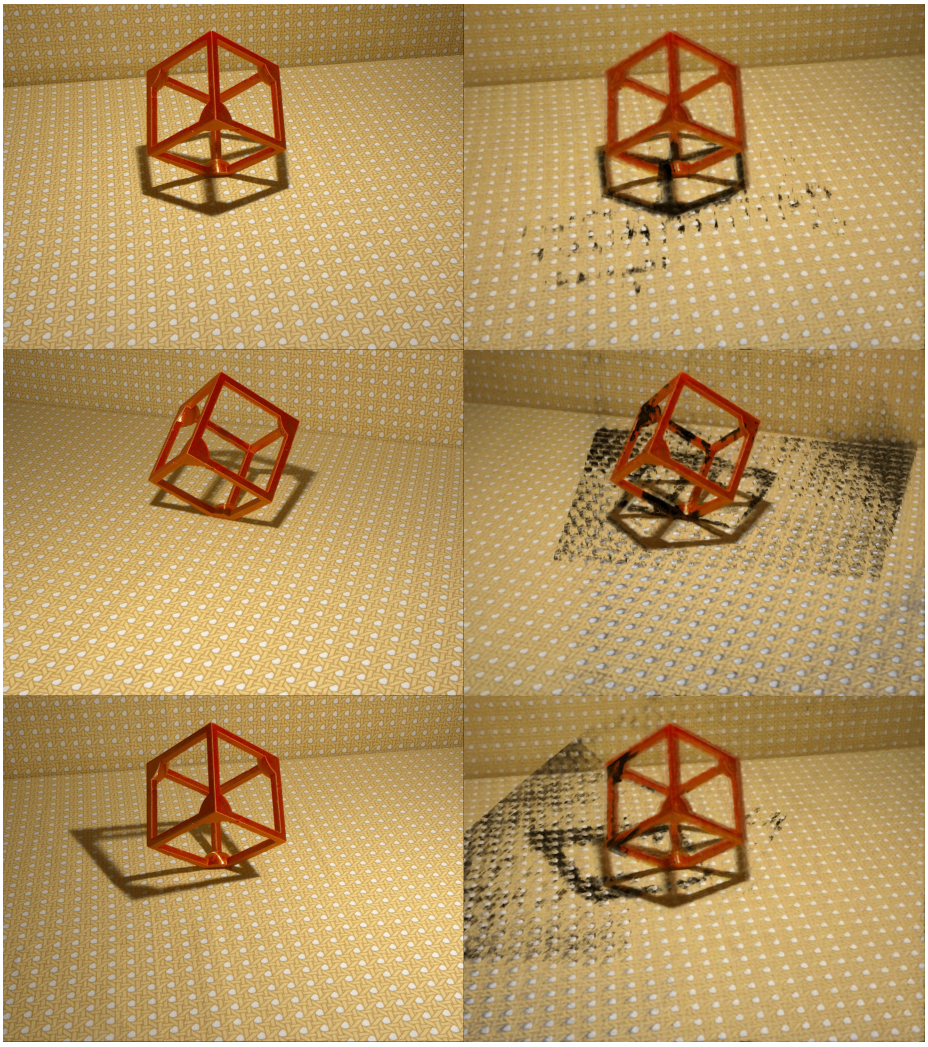

Figure 4.7: Shadow Hints Under Different Conditions

memorize the whole scene. We feed these shadow hints directly into the visibility network to make things more convenient, mainly because there was no need for massive architectural change.

4.3 Shadow Hints

Creating the shadow hints only requires a few extra lines of code in the renderer and mainly comprises 3D vector arithmetic. First, it is essential to visualize the process as a single ray cast during volume rendering. The black arrow is a NeRF ray shooting out of the camera. NeRF queries

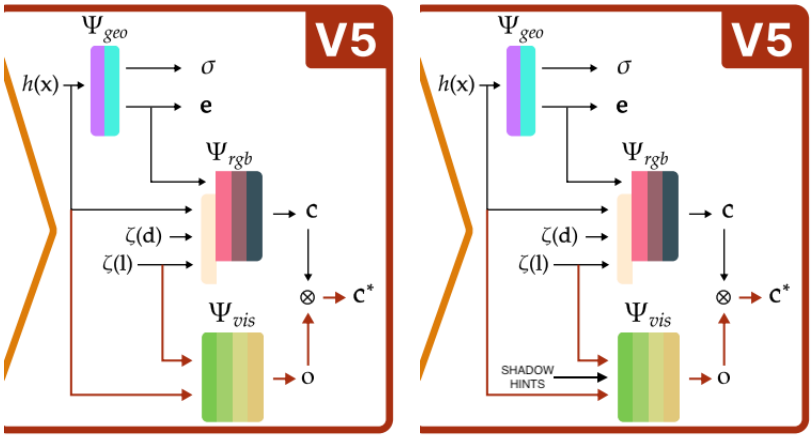

Figure 4.8: Diagram of Shadow Hints Insertion

at distinct points; these sets are known as the $xyzs$ and are represented as black dots. The blue vectors represent the projection from the light source to the black dots before they collide with the surface. The green distance shows the difference between the projected and actual points in 3D space. We do not set a manual threshold but let the Ψ_{vis} learn these green distances coined as shadow hints.

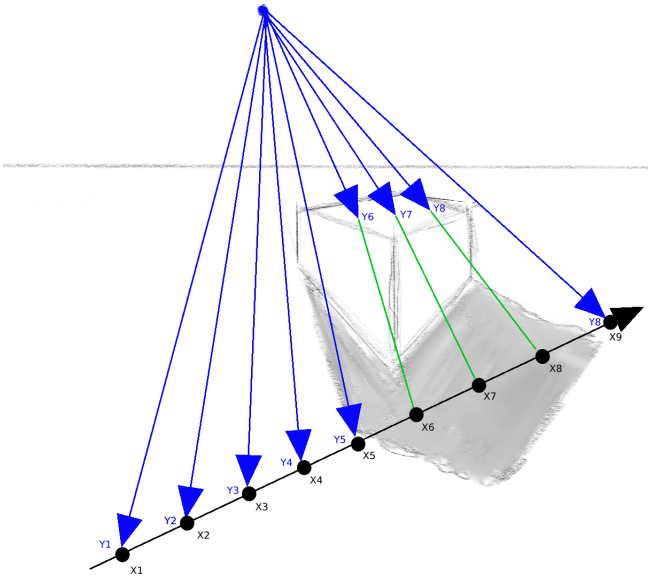

Figure 4.9: Diagram of Shadow Hints Along NeRF Ray

From figure 4.9, the simplified code is relatively easy to follow. The

function distances take in five parameters, the $xyzs$, $w2l$ which the world to the light matrix, $distance\_map$ as shown in figure 4.9, l_rays_d are the directional vectors of each light ray corresponding to each pixel in the $depth\_map$. l_rays_o but as the origin of each light source. The first step is to make the $xyzs$ into homogeneous coordinates. Then, it is possible to find the corresponding pixel in the $depth_map$ to the 3D point. Once the indices of the pixel are known, the depth is indexed. A predicted projection is made using the corresponding ray directions and origins, shown as the blue arrow. After this, a simple norm is calculated to measure the difference shown as the green line representing the shadow hints.

```
def distances(xyzs, w2l, depth_map, l_rays_d, l_rays_o):
    # Concatenate the ones tensor with the points
    #tensor along the third dimension
    ones = torch.ones_like(xyzs[:, :, :1])
    homog_xyzs = torch.cat((xyzs, ones), dim=2)

    #Map 3D points to pixel coordinates
    pixel_cords = w2l @ homog_xyzs
    pixel_ints = pixel_cords/pixel_cords[..., 2]
    pixel_ints = pixel_cords_ints[:, :, :2]

    #Finds depth values of 3D coordinates
    index = pixel_ints[:, :, -2:].to(int64)
    depths = depth_map[..., index[...,1], index[...,0]]

    #Project them onto the surface from a light point
    projection = depths*l_rays_d + l_rays_o

    #calculate distance from projection to xyzs
    dist_points = torch.norm((projection-xyzs), dim=2))

    return dist_points
```

The code above is a simplified version of the actual process; this is done to promote legibility. The actual code requires a lot of matrix reshaping and permuting, so those parts were left out. There are many edge cases to be handled, for example, when the $xyzs$ point might be indexed outside of the discrete distance map.

4.4 Results

The table below is divided into two; the four numeric columns on the left are our experiments with shadow mapping implemented, and the last four columns are copied from the ReNe paper. We can notice that the structural similarity index measure sharply rises with shadow mapping. Meanwhile, PSNR has increased but slightly. PSNR stands for "Peak Signal-to-Noise Ratio." It is a metric used to measure the quality of an image or video compression by quantifying the ratio of the maximum possible signal strength to the noise introduced during compression. Its maximum value is 48, while anything between 25 and 29 for inverse rendering tasks is considered realistic. SSIM stands for "Structural Similarity Index." It is a metric used to assess the similarity between two images, considering pixel-wise differences and incorporating information about structure, luminance, and contrast. For a static NeRF scene, usually, this value is above .8. [11] Given the complex nature of our problem, we could only achieve around .6.

In most images, the cast shadow only represents a small part of the image; if our method were to create perfect shadows, we would still expect a slight increase in these metrics. Calculating the algorithm's effect on only the shadows would require us to have a perfect ground truth mask of where the shadows were cast. Therefore, we found it to be better to compare by human evaluation. To do this, we created one graphic where the top images are the RGB render and shadow predictions from the V5

	S-Hint				V5			
	Easy		Hard		Easy		Hard	
Object	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Apple	**26.52**	**0.66**	**26.35**	**0.66**	26.44	0.62	26.25	0.62
Cheetah	**26.02**	**0.67**	**25.14**	**0.67**	25.66	0.61	24.64	0.60
Cube	**25.02**	**0.59**	**24.64**	**0.59**	24.9	0.54	23.98	0.53
Dinosaurs	25.49	**0.65**	24.88	**0.64**	*25.75*	0.65	*24.98*	0.64
FlipFlop	25.74	**0.66**	25.39	**0.66**	*25.85*	0.61	*25.42*	0.61
Fruits	**26.11**	**0.66**	**25.93**	**0.67**	25.93	0.62	25.72	0.62
Garden	**25.78**	**0.67**	**25.08**	**0.66**	25.74	0.66	25.08	0.66
Helicopters	**25.25**	**0.62**	**24.84**	**0.62**	25.12	0.61	24.73	0.61
Kittens	**25.99**	**0.68**	**25.09**	**0.67**	25.9	0.64	24.96	0.63
Lego	25.87	**0.65**	25.58	**0.65**	*26.07*	0.61	*25.77*	0.61
Lunch	**25.91**	**0.64**	**24.75**	**0.63**	25.84	0.6	24.71	0.59
Plant	**26.64**	**0.67**	**26.07**	**0.67**	26.55	0.67	25.93	0.67
Reflective	**25.95**	**0.66**	**25.44**	**0.66**	25.79	0.61	25.28	0.61
Robotoy	**26.26**	**0.66**	**25.73**	**0.66**	26.24	0.65	25.55	0.65
Savannah	**25.21**	**0.66**	24.30	**0.65**	25.15	0.62	*24.31*	0.61
Shark	**25.81**	**0.62**	**25.59**	**0.62**	25.59	0.57	25.32	0.56
Stegasaurus	**26.03**	**0.65**	**25.82**	**0.66**	25.87	0.63	25.65	0.63
Tapes	25.71	**0.62**	25.26	**0.62**	*25.84*	0.58	*25.41*	0.57
Trucks	**25.81**	**0.69**	25.15	**0.69**	25.8	0.67	*25.16*	0.66
Wood Toys	25.61	**0.64**	25.12	**0.64**	*25.69*	0.61	*25.24*	0.6
Average	**25.84**	**0.65**	**25.31**	**0.65**	25.79	0.62	25.20	0.61

Table 4.1: Empirical Results Comparisons of Shadow Hint and Standard V5

architecture, right below, the same for the shadow mapping architecture, on the right is the ground truth of the RGB ground truth. These help make the results much more evident. All provided images are from the hard test set; this was done to ensure the light condition was never seen in training.

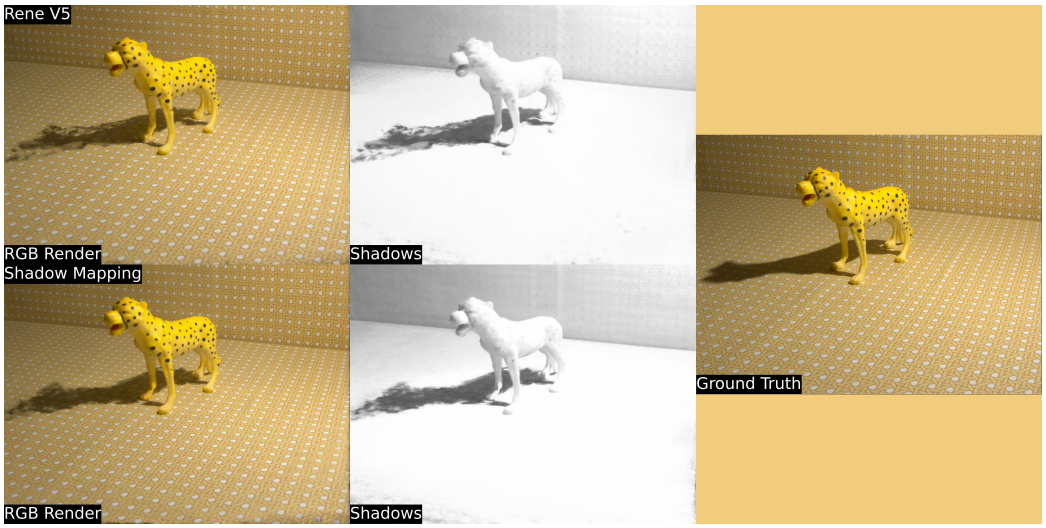

Figure 4.10: Comparison Diagram of Cheetah
The shadow is far from perfect, but the improvements are immense. Around the edges, there are far fewer fragments. It is as if the shadow is now a complete piece and shows more geometric accuracy.

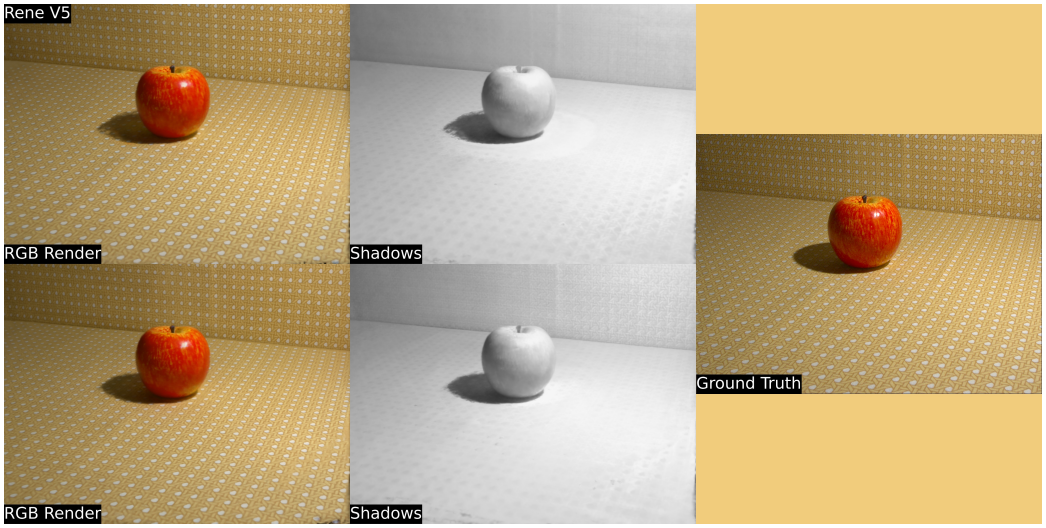

Figure 4.11: Comparison Diagram of Apple
The apple stands out superior to most results, although it faces challenges, particularly around the edges. Even minor deviations from the ground truth become unmistakably apparent in synthetic renders. With shadow mapping, the edges are much more refined.

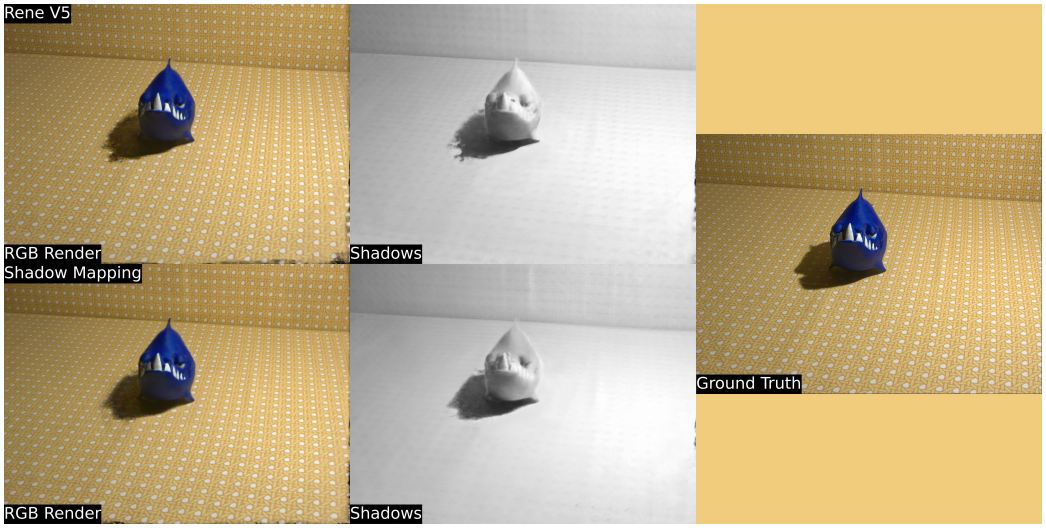

Figure 4.12: Comparison Diagram of Shark
The shark exhibits sharper and more jagged geometry, which V5 handles well overall. Shadow mapping enhances the refinement of shadows, although they may not precisely match the ground truth upon close comparison. Nevertheless, the improvement is significant enough to impress a novice observer convincingly.

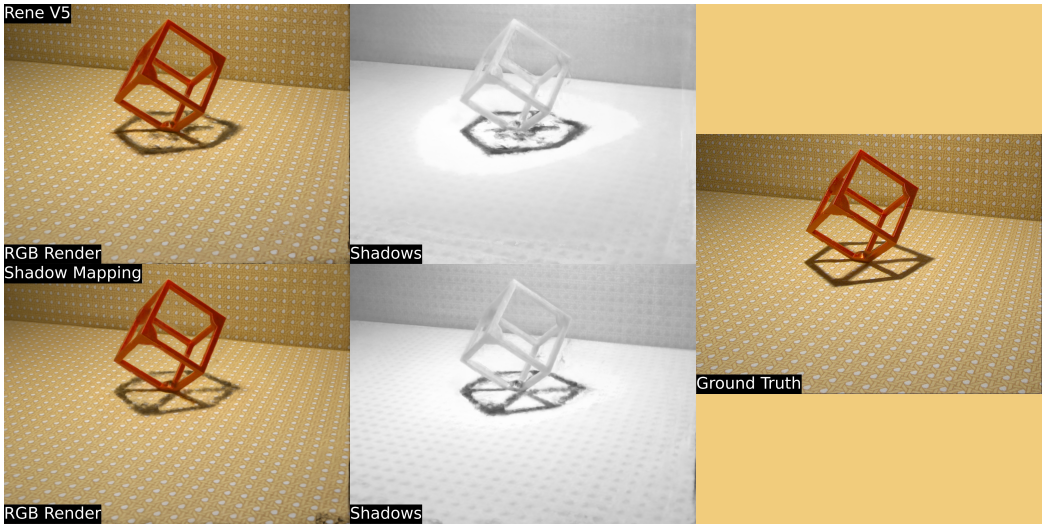

Figure 4.13: Comparison Diagram of Cube
The cube experienced substantial enhancement, especially given its slender and intricate form, which resulted in complex shadows. Due to the intricate geometry, it becomes challenging for V5 to identify the shadow's occlusion precisely. The elevated points in the scene, where the light source varies significantly, pose a rendering challenge for V5. This example vividly illustrates how the RGB network renders accurate shadows without understanding the geometry.

Chapter 5

Conclusion

Real-world relighting is a critical problem. Classical methods have been well developed over the past few years but lack being data-driven. With data-driven techniques, it is easier to derive new rendering methods; however, they require structured data and computational resources. The relighting methods built on top of NeRF require a tremendous amount of computation to get an understanding of global geometry. The brute force methods shown in the literature do not work in the real world, regardless of how convincing their results were. Effective relighting demands careful consideration at every stage of the rendering pipeline, from data capture and training to the final rendering process. The objective of this thesis was not to overhaul an existing pipeline significantly. Instead, as a solo project, the goal was to demonstrate that through a first-principles approach to understanding the essence of shadows, it is possible to generate a more geometrically accurate shadow effectively and at a negligible cost.

The results provided in this thesis prove our hypothesis that relighting to create cast shadows requires an understanding of global geometry. In the literature review, it was shown that to do this by brute was prohibitively expensive. The best method we initially wanted to compare against took two days with four Nvidia RTX 2080Tis; with our one 2080Ti GPU, we opted for a different solution. Using a classical computer graphics technique known as shadow mapping, we showed that injecting information about the geometry directly into architecture is possible. This information came in the form of shadow hints which can be derived in constant time and injected with zero architectural changes. However, this did require considerable changes in the processing pipeline, and the distance maps were not free of cost. It still shows that hybrid methods have great potential when mixed with NeRF.

Since the inception of this thesis, alternative methods for relighting and new datasets have been released. The OpenIllumination [8] dataset

was created with 108K images of 64 objects with pattern lighting. Relightable Neural Radiance Fields (ReNeRFs) showed that with a simple without dense lighting and many of the previous assumptions needed for relighting. "Relighting Neural Radiance Fields with Shadow and Highlight Hints" [19], also trained on a 360 dataset with SDFs as their primary way; this still requires 20 hours on four Nvidia V100 GPUs. Our hybrid approach built on V5 requires much less time. There are still aspects that can be improved on our pipeline, so we chose the easiest to implement that went to prove our hypothesis.

It should be noted that not every inverse rendering problem requires the use of NeRF. Rendering an image from a NeRF involves millions of neural network calls, which makes them notoriously tricky for portability. Meanwhile, classical computer graphics pipelines can be run on modern mobile web browsers, even with ray tracing. For this reason alone, I think the future of inverse rendering will rely heavily on hybrid solutions between classical computer graphics and machine learning. Another method for asset generation and materials capture does precisely this. It is possible for geometry to be captured using classical photogrammetry with meshes, while they use machine learning-based approaches for material capture, which BRDF learns. These results are on par with what a NeRF can learn in a controlled environment, yet the models can be ported to any modern 3D engine. Given the software legacy of traditional computer graphics and the hardware accelerators built around them, it seems likely that explicit representation will remain around but also improve through machine learning-based methods.

This thesis brought satisfaction from achieving a new benchmark but primarily from the valuable lessons gained throughout the journey. It enabled me to delve into the principles governing light and its curious predictability yet enigmatic properties that perplexed the likes of Einstein. Without light, we would all live in darkness. Its projected beauty

derives every sight, color, and form we comprehend as an intricate dance. Light unveils the all-beautiful visual tapestry of existence. In the luminous orchestration of our reality, light is the sole conductor, revealing all that we perceive.

Bibliography

[1] S. Bi, Z. Xu, P. Srinivasan, B. Mildenhall, K. Sunkavalli, M. Hasan, Y. Hold-Geoffroy, D. Kriegman, and R. Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020. URL: http://arxiv.org/abs/2008.03824v2.

[2] M. Boss. *Neural Reflectance Decomposition*. PhD thesis, 2023.

[3] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. P. A. Lensch. Nerd: neural reflectance decomposition from image collections, 2021. arXiv: 2012.03918 [cs.CV].

[4] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. In *International Conference on Computer Graphics and Interactive Techniques*, 1981. URL: https://api.semanticscholar.org/CorpusID:207551006.

[5] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: convergence and generalization in neural networks, 2018. eprint: arXiv:1806.07572.

[6] W. Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, September 2008.

[7] D. Levy, A. Peleg, N. Pearl, D. Rosenbaum, D. Akkaynak, S. Korman, and T. Treibitz. Seathru-nerf: neural radiance fields in scattering media. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 56–65, 2023.

[8] I. Liu, L. Chen, Z. Fu, L. Wu, H. Jin, Z. Li, C. M. R. Wong, Y. Xu, R. Ramamoorthi, Z. Xu, and H. Su. Openillumination: a multi-illumination dataset for inverse rendering evaluation on real objects. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL: https://openreview.net/forum?id=pRnrg2bWr0.

[9] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. DOI: 10.1109/2945.468400.

[10] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[11] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[12] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

[13] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019.

[14] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. Nerv: neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021.

[15] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[16] M. Toschi, R. De Matteo, R. Spezialetti, D. De Gregorio, L. Di Stefano, and S. Salti. Relight my nerf: a dataset for novel view synthesis and relighting of real world objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20762–20772, June 2023.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017. eprint: `arXiv:1706.03762`.

[18] Y. Xu, G. Zoss, P. Chandran, M. Gross, D. Bradley, and P. Gotardo. Renerf: relightable neural radiance fields with nearfield lighting. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22524–22534, Los Alamitos, CA, USA. IEEE Computer Society, October 2023. DOI: `10.1109/ICCV51070.2023.02064`. URL: `https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.02064`.

[19] C. Zeng, G. Chen, Y. Dong, P. Peers, H. Wu, and X. Tong. Relighting neural radiance fields with shadow and highlight hints. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023.

[20] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron. Nerfactor: neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics*, 40(6):1–18, December 2021. ISSN: 1557-7368. DOI: `10.1145/3478513.3480496`. URL: `http://dx.doi.org/10.1145/3478513.3480496`.