Estimating Reflectance Properties and Reilluminating Scenes Using Physically Based Rendering and Deep Neural Networks

Farhan Rahman Wasee

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

October 2020

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Farhan Rahman Wasee**

Entitled: **Estimating Reflectance Properties and Reilluminating Scenes Using Physically Based Rendering and Deep Neural Networks**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Jinqiu Yang

_____ Examiner
Dr. Ching Suen

_____ Examiner
Dr. Jinqiu Yang

_____ Supervisor
Dr. Charalambos Poullis

Approved by _____
Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

October 2020

_____
Dr. Mourad Debbabi, Interim Dean
Faculty of Engineering and Computer Science

Abstract

Estimating Reflectance Properties and Reilluminating Scenes Using Physically Based Rendering and Deep Neural Networks

Farhan Rahman Wasee

Estimating material properties and modeling the appearance of an object under varying illumination conditions is a complex process. In this thesis, we address the problem by proposing a novel framework to re-illuminate scenes by recovering the reflectance properties. Uniquely, following a divide-and-conquer approach, we recast the problem into its two constituent sub-problems.

In the first sub-problem, we have developed a synthetic dataset of spheres with realistic materials. The dataset has a wide range of material properties, rendered from varying viewpoints and under fixed directional light. Images from the dataset are further processed and used as reflectance maps used during the training process of the network.

In the second sub-problem, reflectance maps are created for scenes by reorganizing the outgoing radiances recorded in the multi-view images. The network trained on the synthetic dataset, is used to infer the material properties of the reflectance maps, acquired for the test scenes. These predictions are reused to relight the scenes from novel viewpoints and different lighting conditions using path tracing.

A number of experiments are conducted and performances are reported using different metrics to justify our design decisions and the choice of our network. We also show that, using multi-view images, the camera properties and the geometry of a scene, our technique can successfully predict the reflectance properties using our trained network within seconds. In the end, we also present the visual results of re-illumination on several scenes under different lighting conditions.

Acknowledgments

At the very beginning, I would like to take a moment to express my gratitude towards everyone who directly and indirectly played a role to help me complete this thesis.

Firstly, I am very grateful to my supervisor, Dr. Charalambos Poullis for believing in me and for making me a part of his wonderful lab. I have enjoyed every moment of working side by side with him. His in-depth knowledge, patience and guidance during the most difficult phases of the project helped me navigate my way throughout the last couple of years. It would not have been possible if it was not for him. I would also like to thank the members of my lab and my friends Bodhiswatta [Bodhi] Chatterjee and Pinjing Xu [Alex] for their countless help and advices throughout the time I was a part of the ICT Lab. I would like to thank the respectable committee members, the other professors and staff of Concordia University for the help and support I have received from them.

The research done as part of this thesis is based upon work supported by the Natural Sciences and Engineering Research Council of Canada Grants DG-N01670 (Discovery Grant) and DND-N01885 (Collaborative Research and Development with the Department of National Defence Grant). I want to thank Jonathan Fournier from Valcartier DRDC, and Hermann Brassard, Sylvain Pronovost, Bhakti Patel, and Scott McAvoy from Presagis Inc Canada, for their invaluable discussions and assistance.

Last but not the least, I would like to thank my family for supporting me throughout my life. It would not have been possible at all if it had not been for you and your love. You are the source of my inspiration, joy and all the motivation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, due to the rapid development of cameras, particularly with the widespread availability of handheld cameras and imaging devices, there has been a big rise in the number of images that are taken everyday. Any person owning a cellphone can take a photo anywhere and anytime. The availability of image capturing devices and the ease of use has also made the task of collecting images for large scale datasets easier. Retrieval of images for a place or landmark from different viewpoints and under varying lighting conditions is not challenging anymore. While capturing an image is as easy as clicking a button, doing the inverse of this process is very hard and computationally intense. For any image, performing the inverse process involves recovering information related to its construction, such as the camera properties, the geometry of the scene or finding the reflectance properties of the objects that appeared in the images, etc. Having access to one or more of these information allows one to perform a multitude of tasks ranging from 2D to 3D reconstruction, re-rendering the images, or to make modifications to one or more of the parameters to create alternate versions of the images.

The creation of photorealistic images is a well-studied yet non-trivial process that involves solving the rendering equation [43, 44] at each surface point of the scene visible in a camera's field of view (fov). To generate such photorealistic images, information about the scene geometry, materials, lights, camera, etc have to be known. Once these parameters are available, then it is a matter of choice of the rendering technique that can produce the final render. Indeed being a well-studied process, for the past few decades numerous techniques have been proposed primarily focusing on

the main challenges of (i) physics-based reflection models [21, 34] and (ii) efficient and effective processing of the input information, [19, 20, 35, 39, 40, 54] - to name a few.

Going back from the images to the various parameters involving the image creation process i.e camera poses, scene properties, etc, is a complicated process known as Inverse Global Illumination (IGI). The goal of the process is the recovery of the reflectance properties of the objects in the scene given (i) a set of images capturing the scene, information about the geometry of the scene, and (ii) the illumination conditions at the time the images were captured. Recovering reflectance properties has immense value in various applications. It allows us to render view independent images, rendering under novel lighting conditions or for creating timelapse effects of a scene for visualization. Over the past few years, IGI has been performed on small-scale scenes under a lab setup where the scene can be captured using a laser scanner. In these studies, the lighting environment is captured by using light probes or by placing chromium spheres [25, 26].

Limited work has been done over the years on the field of inverse rendering or inverse global illumination in uncontrolled setup. In recent years, deep convolutional neural network and generative adversarial networks have been used to address this problem. Several techniques have been proposed e.g. [17, 49, 81] in which given images of the scene and in some cases additional information extracted from those images e.g. depth map, semantic maps [67], the network then produces renders of the scene from novel viewpoints. In these techniques, most of the test scenes are either indoor and small scale or architectural landmark sites that have similar structures and texture. Also, despite producing impressive results, the users do not have limited control over the output of the network i.e. the reflectance properties are embedded in the network and cannot be deciphered or extracted.

In this thesis, we present a technique for reilluminating scenes under novel lighting conditions and from novel viewpoints by estimating the reflectance properties of the objects within the images. Uniquely, we recast the problem of recovering the bidirectional reflectance distribution function (BRDF) of four parameters $f_r(\omega_i, \omega_o)$ into its two constituent components. In the first component, a residual neural network is used to reduce the parameter space of the BRDF from the 4-parameters relating to the incoming light direction ω_i and outgoing view direction ωo, to only the 2-parameters relating to ωo. The network learns a function for mapping a sparse reflectance map that encaptures

the outgoing radiances across all the directions, to a 7-vector representing the reflectance properties of the surface. This is achieved by training the network with a vast number of reflectance maps created using renders of the unit sphere with different material properties being lit from a fixed light source and varying viewpoints.

In the second component, all samples of the reflected radiance captured in the images - from arbitrary view directions - are aggregated per triangular face. This results in a reflectance map per triangular face, which is a function of the remaining 2-parameters of the BRDF specifying the outgoing view direction ω_o. We assume that each surface in the scene has a uniform and isotropic material. Under this assumption, this reflectance map is then used as the input to the network. The output of the network is a 7-vector representing the reflectance properties for each surface. This process is repeated for the rest of the surfaces contained in the scene.

In summary, we propose a *two-stage processing pipeline* for inferring the reflectance properties per triangular face, given several images capturing the scene, and finally relighting the scenes under different lighting conditions and viewpoints.

The first stage focuses on the training of the deep neural network for inferring reflectance properties. There are two parts associated with it:

- **Generation of training data:** A synthetic dataset is created with renders from varying viewpoints, under a fixed lighting direction and with large number of varying reflectance properties.

- **Network training** A wide residual neural network is trained on the synthetic dataset using regression.

The second stage focuses on the composition and aggregation of the outgoing radiances from the images and inference of the reflectance properties. It can be broken down into two parts:

- **Composition of reflectance maps for scenes:** A per-triangular face reflectance map is composited by aggregating the corresponding outgoing radiance values captured in the images.

- **Inference from the reflectance Maps:** Inferring reflectance properties for each of the reflectance maps and re-rendering with the predicted properties using path tracing.

1.1 Thesis Organization

The remaining of this thesis is organized as follows: Chapter 2 highlights a few classical and recent methodologies of similar research which includes procedural and classical computer vision based algorithms. This chapter also presents the recent related methodologies involving deep learning and modern computer vision approaches. In the following chapter, Chapter 3, we discuss some theoretical and practical background including rendering techniques, various reflection models, and also discuss the basis of our problem formulation. Chapter 4 describes our complete system pipeline for data generation, the estimation of reflectance properties from reflectance maps. Additionally, this chapter presents some of the rendering results obtained using our approach. Lastly, in Chapter 5, the conclusion and scope of future work are presented.

Chapter 2

Literature Review

In recent years, due to the development of multiple open-source frameworks [5,41,76] and massive computational improvements (mostly GPU based parallel computation) [51, 60], there has been a rise in deep learning based research. This has resulted in the publication of a lot of new network architectures that can have numerous applications. Nowadays, deep learning is being applied in multidisciplinary fields ranging from image processing, natural language processing, video analysis and object tracking to topics like analyzing medical images and artificial/synthetic content creation i.e using deep learning to write texts, creating images of objects or scenes that do not exist, etc [15, 38,46,47,80,107]. In the following few sections, we go over some of the recent and past methods that attempt to solve similar problems such as ours. More specifically, we will first go over some of the most famous deep neural network architecture that has made notable contributions. Then we discuss some appearance modeling techniques which attempt to recover reflectance properties and information about the materials from images. We separately discuss the more conventional procedural based techniques and the deep learning based techniques in two different subsections.

2.1 Deep learning and Convolutional Neural Networks

Many major contributions have led to the recent success of deep learning across different domains. LeNet [56] and AlexNet [53] introduced modern ConvNet architectures and began this uprising. Several factors contributed to this widespread success. Amongst which one is the improvement of

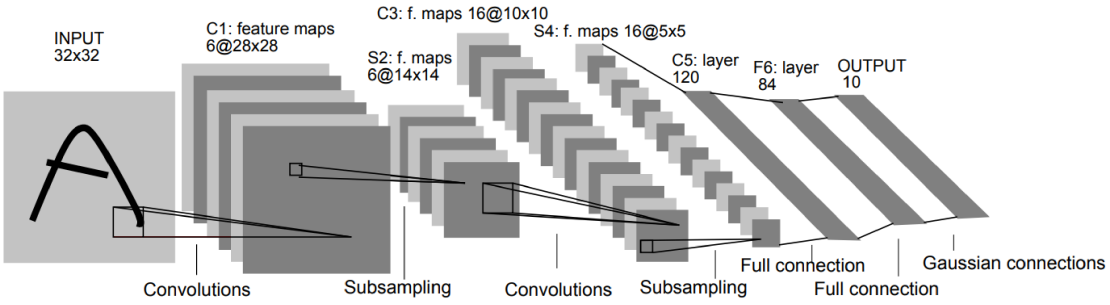

Figure 2.1: The architecture diagram of the LeNet model [56] that was designed to recognize handwritten characters from images.

modern GPU architectures that allow parallel training with backpropagation to be run directly on the GPU system. This has significantly cut down the time required to develop a system, specifically, to train the large models. Other than that, the availability of large datasets along with annotations has increased. This has allowed the scope of more research and allowed researchers to apply deep learning strategies into multidisciplinary fields. Lastly, modern deep learning libraries have allowed implementing a deep neural network within a few lines of codes.

As highlighted in the previous section, our work involves training and tuning a deep convolutional neural network on a realistic synthetic dataset. The choice of the network is justified in a later section with performance comparisons and ablation studies. Some of these networks are described briefly in the following section.

2.1.1 LeNet

LeNet [56] is one of the earliest deep network architectures and it is considered to be the first ConvNet. Figure 2.1 shows the architecture of LeNet. This network achieved state of the art performance in detecting handwritten digits. During that time GPUs were not available to speed up the training, the CPUs were slow as well. Hence, instead of using all the pixels from each image and using a fully connected neural network, LeCun et al. [56] used the assumption that pixels within the same neighborhood share similar characteristics and have relationships with one another. Hence convolutions were used to extract these relationships and the convolution weights were learnable parameters that the network would learn during the training iterations.

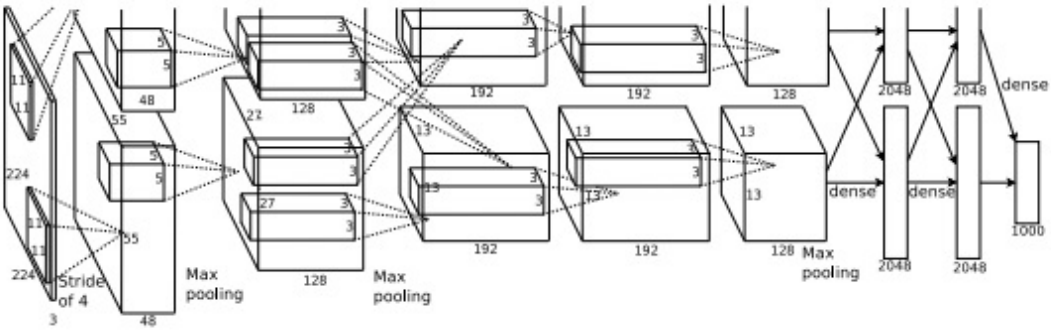

Figure 2.2: The architecture of AlexNet [53] which created a revolution in image classification and introduced several new ideas including the use of data augmentation and GPUs for training.

2.1.2 AlexNet

Although LeNet was the earliest deep network that introduced ConvNets to recognize handwritten digits, it was a shallow network that did not scale well on images from other categories. With the introduction of AlexNet [53], the scenario changed. Krizhevesky et al. [53] enhanced the performance by making the network deeper by adding more layers, by introducing some parameter optimization techniques, and most importantly by introducing GPU training. The architecture of AlexNet can be seen in Figure 2.2. AlexNet also introduced the idea of using data augmentation techniques on images to reduce overfitting and artificially increasing the dataset size. Two parallel GPUs were used to train the AlexNet model that is also divided into two similar but parallel modules. The introduction of AlexNet changed the way of research in computer vision completely.

2.1.3 ResNet

He et al. [32] proposed the deep residual network architecture named ResNet [32] in 2016 which comes with a few variants. It achieved the state of the art results in ImageNet [85] challenge significantly. Figure 2.3 shows a couple of examples of the residual blocks. ResNet introduced multipath training by creating a skip connection with the earlier layers and the latter layers. The success of of ResNet revolutionized the idea of residual learning for ConvNets, something similar like this that was also noticed in U-Net [83] for semantic segmentation. ResNets have smaller blocks that are called residual blocks. For the success of this network, a few other variants have also been proposed

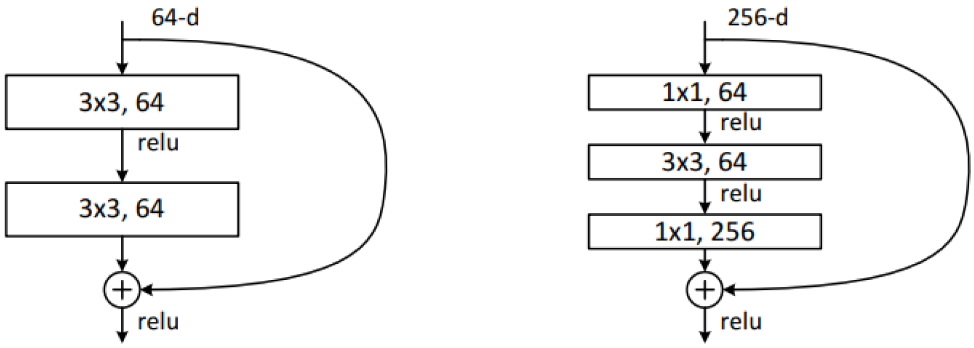

Figure 2.3: Two different types of residual blocks are shown here. The left one corresponds to a residual block for the ResNet-34 [32] model. The right one, on the other hand, shows a bottleneck block for the ResNet-50/101/152 [32].

such as WideResNet [105], Inception-ResNet [91].

2.1.4 ResNeXt

ResNeXt [102] was also proposed by the authors of the original ResNet paper by improving the performance from the previous models. They achieved this using the concept of cardinality. Cardinality is multiple parallel pathways through which the convolution operation and pooling are performed through the ResNet block. These multipath residual blocks can be seen in Figure 2.4. These pathways are similar to the ones used by the older Inception modules [91]. However, the usage of parallel pathways did not make the model computationally heavier because the additional parts are parallelly stacked and are not deep sequential layers. Also, the network was made wider within the blocks that had shareable parameters amongst themselves.

2.1.5 DenseNet

DenseNet [37] was proposed in CVPR 2017 and it got the best paper award by getting state of the art results in CIFAR10 [52], CIFAR100 [52], SVHN [71] and ImageNet [85]. DenseNet is composed of smaller dense blocks. In a single DenseBlock, each layer is connected with all its previous layers. Therefore, a latter layer receives the feature maps from its previous layers which are stacked and concatenated to form the current layer's feature map. This improved overall performance by a wide

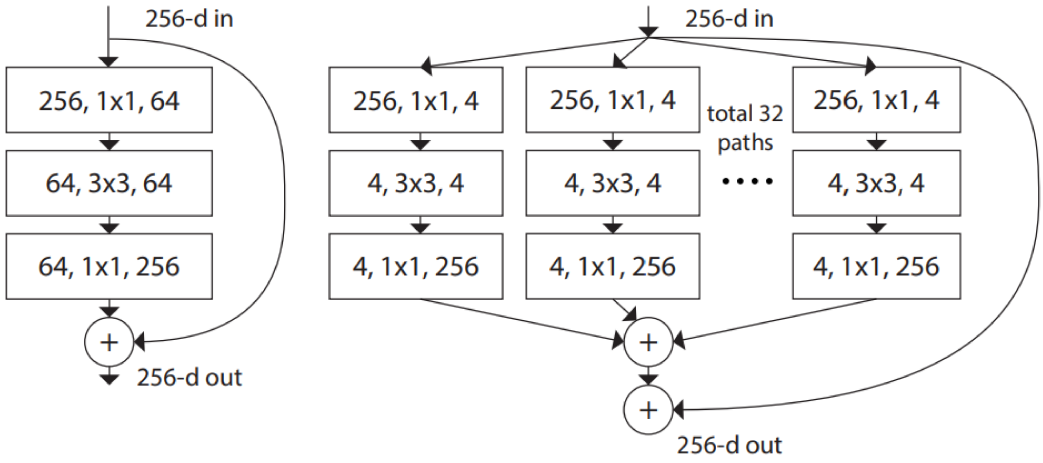

Figure 2.4: Figure illustrating the fundamental concept of parallel paths of ResNext [102]. On the left, we can see the basic residual block of a ResNet. On the right, we can see parallel blocks of this same type, a total of 32 paths converging into one. The cardinality is 32 in this case.

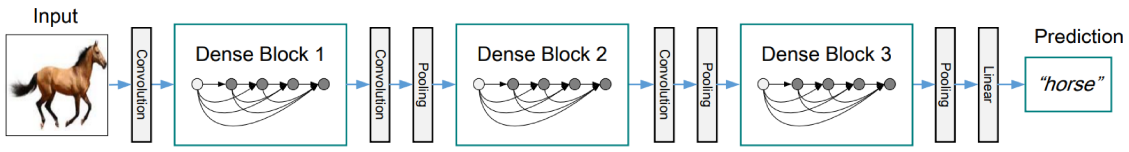

Figure 2.5: Figure illustrating the flow of data through a DenseNet [37]. There are 3 dense blocks here. The layers in between two dense blocks are referred to as transition layers and they contribute towards reducing the size of the feature maps with the help of convolution operations and pooling. Each dense block has 5 layers, with a growth rate, k = 4. Each of these layers also takes in the feature maps from the previous layers as input.

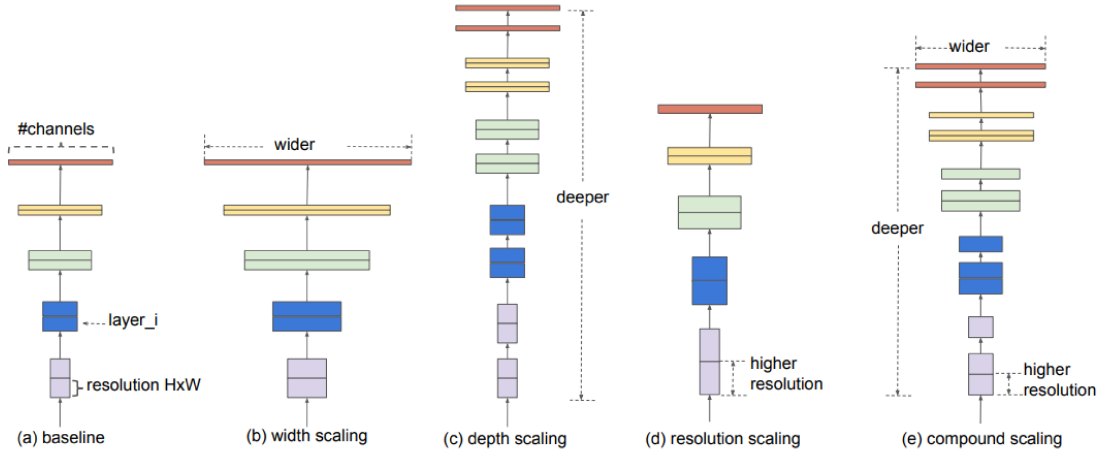

Figure 2.6: This figure shows the motivation behind the architectural design of EfficientNet [92] in contrast to other conventional ConvNets. Here, (a), (b), (c) and (d) show how each dimension i.e. width, number of channels, depth, etc are adjusted to achieve optimal performance. In comparison, (e) shows how EfficientNet uses all of these factors together and uniformly scales them with a fixed ratio to find the optimal results more efficiently.

margin by assuring strong gradient flow across earlier layers. Figure 2.5 shows the pipeline of a DenseNet.

2.1.6 EfficientNet

EfficientNet [92] has managed to obtain state-of-the-art results by getting 84.3% top-1 accuracy in ImageNet [85] in 2019, despite it being 8.4x smaller and 6.1x faster on inference than the previous state-of-the-art. Uniquely, in this paper, the objective of the authors was not just to achieve the best results, but they also focused on reducing the model complexity. By the means of neural architecture search, they perform model scaling for finding the perfect balance of the network depth, width and resolution. Figure 2.6 illustrates their idea. The authors proposed a novel model scaling method that uniformly scales these dimensions using a compound coefficient which results in better performance. They proposed a family of models named EfficientNets. The EfficientNet model family is consisting of 8 models from B0 to B7. Each of these subsequent models is numbered from 0-7, where the numbers indicate more parameters and higher accuracy as they go up.

2.2 Appearance Modeling and Inverse Rendering

Inverse rendering, appearance capture and modeling have been a challenging topic for well over four decades. Humans are very good at understanding materials or the type of object just by using their vision. However, accurately estimating reflectance properties from images remains a challenging problem to this day. There has been a large number of studies and a wide variety of techniques that address this topic. Some of them try to estimate material properties, whereas a few of them try to label objects into a predefined set of material labels. Below we will go over some of the works in this domain and categorize them into procedural-based and the other ones based on deep learning.

2.2.1 Procedural-based

Traditional systems need human assistance, a controlled environment for setting up experiments that capture the appearance of a scene. To capture the illumination or environment, various instruments have to be used. For instance, a laser scanner that can capture the scene geometry, light probes that are used to capture the environment, the light information, etc. [65] presents one such attempt to estimate the effects of light in an image with the help of the scene geometry and camera parameters. The system tries to solve a least square system to find the light distribution which is then used to compute the changes required to go to the target lighting conditions. Another significant work in reflectance estimation was done by Yu et al. [104]. It attempts to recover diffuse and specular reflectance, and model it using a low parameter reflectance model. This requires several calibrated images taken under predefined lighting conditions alongside the geometric model of that scene. Then they try to optimize for the reflectance parameters iteratively. Romeiro et al. [82] were also one of the first to show how reflectance properties can be estimated under known shape and illumination.

[25, 26] proposed a technique for estimating reflectance properties of a landmark under natural illumination. The technique involves scanning the scene geometry using a laser scanner, taking multi-view photographs from different locations, and also capturing the illumination condition for each of the images. In addition to all of these, a subset of BRDF measurements is also taken from different surfaces amongst the scene. Afterwards, an iterative global illumination procedure is performed that estimates the reflectance properties and re-renders the scene. The images are then

compared with the original ones and the parameters are tuned to minimize the errors. Lombardi and Nishino [62] reformulated the problem of acquiring reflectance from images using a probabilistic approach. Despite adding valuable contribution to the field, the downside with many of these approaches is they rely heavily on the setup of the scene, and will need a different approach if one or more information is not available about the scene. Laser scanning a scene is not easy, and it is not always practical to design a method for which the geometry, images, BRDFs, etc will be readily available to estimate the materials. Also, a few of these experiments work with simplistic objects i.e. images of spheres against a solid background or an environment map. Hence, they do not scale up with complex geometric shapes.

Barron and Malik [9, 10] proposed two separate techniques for recovering shape, diffuse albedo and illumination from an image. The first of these two techniques focused on achromatic images while the latter worked on colored images. However, the problem with this approach was the surface materials were modeled using the Lambertian reflection model. Glossiness or the concept of roughness is not defined in the Lambertian model. Which makes the technique non physics based and thus it is limited in its ability when it comes to real-world materials. Brelstaff and Blake [14] proposed a method that attempted to address this issue of identifying specularity in images. Their method works well but it can only identify specular shiny regions from images. Abe et al. [6] use several feature detectors to extract features from images and use an SVM to rank the images based on a few predefined attributes, which are glossiness, roughness, transparency and coldness. Goldman et al. [29] estimated per-pixel mixture weights and a combination of parametric materials. However, this approach requires a few HDR input images which were captured under known lighting conditions, which is again difficult to obtain.

Over the years, several different techniques have been proposed to obtain material properties from images. However, using traditional techniques most often work by relying on laboratory setup. Also, some of these techniques put constraints on additional factors like illumination, shape, etc. Shapes of objects can be of countless types, illumination can have a lot of variations. Besides, materials can have subtle variations producing changes in appearance - addressing all of these factors at once is difficult using traditional techniques and rule-based approaches. Therefore, estimating realistic materials properties under varying illumination, i.e estimating diffuse, specular, roughness

components can be challenging with these methodologies.

To apply a machine learning method, a person has to decide on the features that are to be extracted from the images, videos or other forms of data. Feature descriptors such as SIFT [63], SURF [11], ORB [84], BRISK [57], etc. are commonly used with images. Using features extracted using one of these techniques, a model would be trained. This model would generally be a Support Vector Machine (SVM), Bayes Classifier, Decision Tree, etc. The downside with this approach is choosing the feature descriptor and the model. Since the descriptor would be extracting the feature information from the data, and that information would later be used to train the model, it is crucial that the feature descriptor extracts maximum and most meaningful patterns from the data. Most of the time this is a very difficult decision to make beforehand and it is very challenging to pick one model/descriptor with certainty that would produce optimal performance. In recent years, deep learning approaches have shown significant improvement in this domain. Some of which are discussed in the next section.

2.2.2 Deep Learning based

Since the earlier days of classical Computer Vision and Computer Graphics algorithms, things have come a long way with the help of modern deep learning algorithms. Many tasks that were previously challenging and erroneous can now be solved using deep learning based solutions with near human-level accuracy. In the domain of computer vision and computer graphics, deep learning has achieved more widespread success than machine learning. The revolution of deep learning has opened a new paradigm of ways to tacking the inverse problem of relighting and material estimation, especially with the development of various deep learning frameworks [5, 41, 76]. Lately, differentiable rendering has also been used to solve similar subtasks as seen in [8,17,59,61,73,103].

Dror er al. [27] estimates reflectance properties from just 6 predefined types specified by the Ward model [97] and trains a classifier using synthetically created images of spheres. Rematas et al. [81] addressed this inverse problem by creating reflectance maps from specular materials in natural lighting conditions. Their technique assumes that the object is made up of a single material that is consistent throughout its geometry. They proposed two modules in their system, one of which directly estimates the reflectance maps from the image while the other first predicts per-pixel surface

normals that are used to compute a sparse reflectance map. It is then used in interpolation to create a denser map.

Meka et al. [66] used a system with 5 smaller deep networks to estimate materials for performing material transfer. An image is taken as input which is first masked from its background and then further processed to estimate diffuse and specular components of the material. The technique works well on objects having simple shapes and it estimates a single material for each image. However, this approach has a few limitations like not supporting global illumination and not modeling specularity.

Kim et al. [50] built a lightweight system to estimate surface reflectance from images in realtime. They estimate the surface reflectance by using a simple BRDF representation and by estimating surface albedo and gloss instead of estimating the full 4D BRDF function. They use two networks to estimate the BRDF which enables rendering under different illumination and view independent conditions.

Philip et al. [78] proposed an end to end approach for relighting outdoor scenes, mostly of landmarks. Their system takes a set of multiview images, computes a proxy 3D geometry. A part of their system focus on the shadow removal and refining process using another sub-network. Their system has also been trained on photorealistic synthetic data generated using physics-based raytracing. A similar approach has been taken by Meshry et al. [67] where they attempt to do a total scene capture. The system utilizes publicly available images of landmarks as inputs and uses off the shelf reconstruction algorithms to retrieve the geometry as part of the process.

NeRF [68] which is proposed by Mildenhall et al., proposed a differentiable rendering approach. Their system performs volumetric rendering to compute the final color value for each pixel. The system also takes the direction of a ray into account when computing the final color for a pixel. Therefore, their system can model non-Lambertian materials and complex scenes that exhibit view dependent phenomena such as shininess or specularity.

Chen et al. [18] use one of the recent approaches of using a neural renderer to tackle a similar subproblem. In their work, they designed a neural renderer to synthesize novel viewpoints and perform reillumination by defining the appearance as a combination of environment lighting, object intrinsic attributes, and the light transport function (LTF). These are learnable through training. The scenes are mainly of small-scale objects and it does not work particularly well in the presence of

highly specular objects.

The authors in [69] published a dataset of indoor scenes composed of a fixed number of predefined material categories in multi-illumination conditions. Along with the dataset they tackled the problem of scene illumination prediction by predicting the light probe for the given input image. Also, they formulated image relighting as an image to image translation problem where they train a network to learn the direct mapping (without factoring out the material properties) between two images taken from the same viewpoint under two lighting conditions. The system does not estimate reflectance properties throughout the process and is limited to close shot images that are taken indoors.

There has also been a lot of research done on estimating reflectance for human faces and portrait relighting in particular. A good example of these can be seen in [70, 90, 106]. Reilluminating indoor or outdoor scenes with a wide variety of materials and their properties do not fit well with these techniques. Also, it is impractical and impossible to generate a large dataset of indoor-outdoor scenes comprising objects of different shapes and materials in a studio-like environment as it can be done with human faces.

There are other methods [28, 36, 89] that capture the illumination condition or the environment that the image was shot in, to perform manipulations such as inserting a new object into the scene, etc. However, these methods do not give a way to reintroduce new lighting conditions and re-rendering. [48] shows another method to insert objects into scenes realistically using a single image. However, it requires annotations to be done by the user on the image.

2.3 Conclusion

Despite all the recent advancements and contributions in the field of inverse rendering and material estimation the procedural methods generally work well under constrained environment, controlled lighting and/or with objects with simpler geometry. On the other hand, the recent deep learning techniques produce results that provide limited or no control from the end user. Also, most often, the internal representation of what is going on under the hood in these relighting models is very hard or nearly impossible to interpret. Some of these techniques take in images, and re-render

using different lighting condition. Most often, the user has no knowledge on how the network is internally registering the materials. Whereas, in our technique, the materials are well defined, and the viewpoints can be controlled. Each of the objects in the scene has specific reflectance properties assigned to the geometry. Besides, the material reflectance model used in this work is physically based and can be used to model any realistic material.

Chapter 3

Background

In recent times, there is a large number of publicly available renderers that allow photorealistic rendering using a wide variety of algorithms to produce images with a realistic appearance. A few of these are Mistuba Renderer [74], Physically Based Rendering (PBRT) [77], Arnold [1], Cycles [2] and Maxwell [4].

3.1 Rendering and Different Rendering Techniques

Rendering involves the process of synthesizing images using 3D scenes, illumination and camera positions. Typically, a renderer parses a scene description file and produces a single or a set of images as the output. Generally, the scene description file contains information about the scene geometry, material properties, texture, and additional renderer settings. It may also contain information about the light source(s) along with the intrinsic and extrinsic properties of the camera. The particular 3D scene would be rendered from the viewpoint of this camera with the illumination coming from the light sources. Rendering is the final phase of the graphics pipeline. Rendering can take place in two different parts of the system hardware. Based on that they can be categorized as CPU rendering and GPU rendering. Most renderers work in the CPU and are relatively slow. Whereas, GPU rendering is the more recent technology and it takes advantage of the GPUs for rendering, producing high-quality results in a shorter time. Many techniques and algorithms exist that can be used for rendering and all of them try to solve the rendering equation [45]. In the following sections,

we briefly go over some of the major and mostly used rendering techniques.

3.1.1 Rasterization

Rasterization is one of the oldest and simplest forms of image rendering. It produces very fast renders, however, the resultant images lack realistic shading and appearance. Rasterization essentially breaks down the rendering process into two parts, **i. Visibility checking** and **ii. Shading computation**. Generally, the 3D scene geometry is broken into very small triangles or polygons. The vertices of the triangles are then projected onto the screen. Afterwards, the pixel lying within the projected vertices are filled with color. The final color of a pixel can be calculated using various shading techniques.

Rasterization is an object-centric approach where the algorithm starts its process from the object level and finishes up in the screen space, forming the image. Most often it is a memory-intensive algorithm. It operates by having a few memory buffers during rendering. For example, the z-buffer has to be maintained which is used during depth testing. During rasterization, 3D points are mapped into the 2D image plane using the camera's intrinsic and extrinsic properties. Specifically, the projection of a world space point (X, Y, Z) can be computed using Equation 1. Here, using a homogeneous coordinate system, (u, v, w) is in image/screen space. The first matrix on the right-hand side represents the camera intrinsics. (fx, fy) correspond to the focal lengths of the camera. In a true pinhole camera, both f_x and f_y should have the same value. However, due to flaws in the camera sensors, they can vary a little. Also, the lens distortion and errors in camera calibration can make them unequal. (u_0, v_0) is the coordinate of the principal point of the camera. s represents the skew of the camera lens, which is non-zero if the image axes are not perpendicular to each other. The second matrix on the right-hand side corresponds to the camera extrinsic parameters, the $3x3$ on the left express the rotations, whereas the rightmost column containing (t_x, t_y, t_z) represents the translations. $(X, Y, Z, 1)$ are world space coordinates. Once the 3D point has been converted from the world space coordinate system to the camera space, it can be used to compute the z-distance of the point, which in turn can be used for computing the z-buffer. For shading, a number of common techniques can be applied like Phong [79] or Gouraud [30] shading.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

3.1.2 Ray casting

Ray casting is also an older technique and was introduced in 1982 by Arthur Appel [7]. Unlike rasterization, ray casting operates in image order and goes from image space towards the 3D object space. To start, a ray is generated from the camera through a pixel. Afterwards, the shading is calculated for the hitpoint of that pixel. Ray casting is fast because only one ray is shot for each pixel and there are no additional recursive rays from the hitpoints. The downside of this is images rendered using ray casting has hard shadows and lacks global illumination effects like reflection, refraction, etc. The advantage of using ray casting over older scanline algorithms is, raycasting can easily deal with non-simple shapes like spheres, cylinders and cones. However, due to the lack of realism, it is not that widely used any longer.

3.1.3 Raytracing

Even though raycasting was widely used it could not address a few properties that contribute towards making an image appear realistic. The lack of common appearance attributes like reflection and refraction did not produce satisfactory results. This changed when Turner Whitted introduced his technique [99]. Figure 3.1 illustrates the process. In its simplest form, the process of raytracing begins from the camera. A ray goes through each pixel in the image plane which is being shaded. This ray travels towards the scene until it is intersected with an object. From there it can generate up to three new types of rays, they are: reflection, refraction, and shadow rays. If the object has reflective or refractive properties, secondary rays are determined by the physical laws of reflection or refraction. For finding out if a point is illuminated or shadowed, the technique finds out if there is a direct path from the point towards the light source. If a ray from that point towards the light

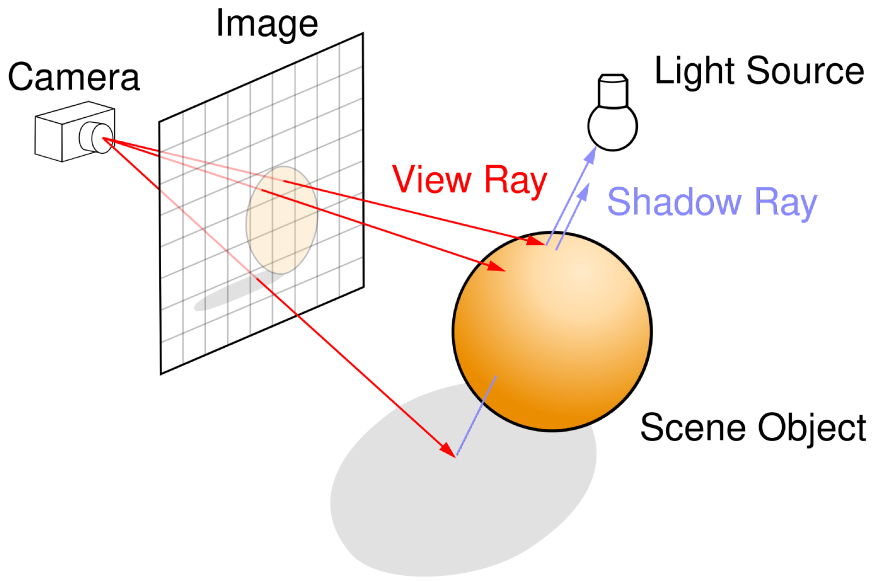

Figure 3.1: The raytracing process involves tracing the rays from the camera through the image plane towards the scene geometry. This diagram from [16] shows the rays are traced for some pixels on the image plane that form the image of the sphere. The view rays get intersected with the sphere, from where the shadow ray is spawned towards the light source. The bottommost view ray hit the surface from where the shadow ray cannot reach the light, hence it is creating the shadow.

gets intersected by another object, then the original point lies in the shadow. Otherwise, the shading for that point is computed using the surface normals, material properties, the lighting direction and intensity. Raytracing is relatively slower than other scanline based algorithm, however, it produces more realistic renders.

3.1.4 Path tracing

Path tracing is a form of ray tracing where instead of sending one ray per pixel, multiple rays are shot through each pixel. Once a ray intersects an object, from that point the ray is bounced. Instead of targeting this ray towards the light, the ray is shot towards a random direction. This secondary ray then repeats the process until a certain level of depth is reached. Then the color is computed by aggregating or by averaging. Path tracing produces visually stunning and photorealistic renders. However, if the number of samples is not large enough then many of the pixels might fail to form a path towards the light, thereby producing a grainy image. Figure 3.2 shows these effects. Also, path tracing can render complex cases like soft shadows, caustics, global illumination, etc.

Figure 3.2: 3 different renders generated using path tracing. The three images are using rendered using sampling sizes 4, 32 and 256 respectively. The noisiness gets stronger as the number of samples is lowered. As the number of samples grows bigger, more rays end up reaching the light source, thereby, producing a cleaner and smoother image.

3.2 Reflection Models

3.2.1 Bidirectional Reflectance Distribution Function

As our objective is focused on modeling the surface appearance, we need to specify how a surface reflects light. The bidirectional reflectance distribution function (BRDF) is a function of four real variables that defines how light is reflected from an opaque surface. It is widely used in computer graphics and computer vision algorithms. Given an incoming light ray at a point on a surface, the BRDF is used to calculate how much of that light will be reflected in a particular outgoing direction. The function takes an incoming light direction and outgoing direction with respect to the surface normal. It returns the ratio of the reflected radiance exiting towards the specific outgoing direction, and the incident irradiance incoming from the direction of the light source.

An important property of BRDF is that it is illumination independent, which means BRDF values calculated or measured for a specific material under a lighting conditions are true for any lighting conditions. This is because BRDFs describe the relationship between the incoming and outgoing radiances for a surface point. As BRDF is the ratio of both the outgoing and incoming radiances, the values are independent of the strength and geometry of the light source. The reflectance at a surface point P is modeled by the bidirectional reflectance distribution function (BRDF) and is given by,

$$f_r(P, \omega_i, \omega o) = \frac{L(P, \omega_o)}{L(P, -\omega_i) cos(\phi) m(\Omega)} \tag{2}$$

where Ω is the solid angle subtended by the light source at the surface point P, $m(\Omega)$ indicates its measure, ω_i and ω_o are the incoming (from the light source) and outgoing (to viewpoint) directions respectively, and ϕ is the colatitude of the light source. The parameters ω_i and ω_o for the incoming and outgoing directions are further parameterized by the azimuth angle ϕ and zenith angle θ making the BRDF a function of four parameters $f_r(\phi_i, \theta_i, \phi_o, \theta_o)$.

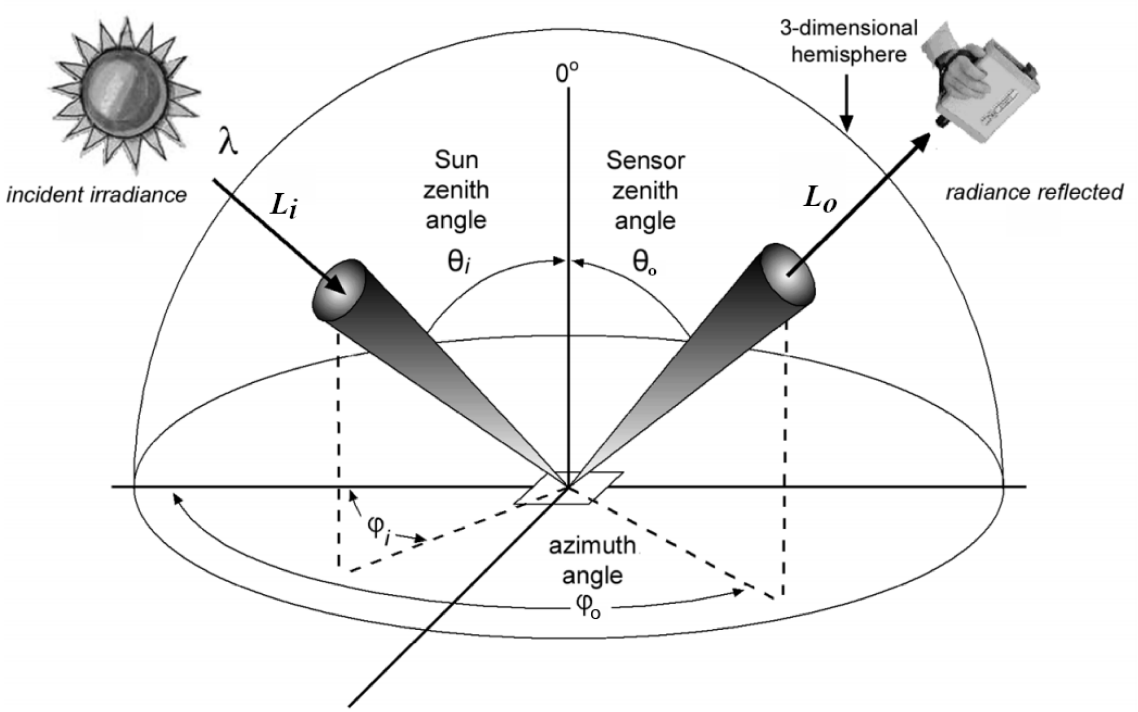

Figure 3.3: The figure [86] illustrates the measurements involved in computing the BRDF. A distant light source emits radiance L_i. The incoming angles are expressed using the zenith and azimuth angles denoted by (θ_i, ϕ_i). The sensor captures the reflected radiance L_o which creates an angle represented by (θ_o, ϕ_o). (Variable symbols are edited from the original figure to match our equations.)

3.2.2 BRDF Models and Acquisition

BRDF Models: There are several methods for obtaining BRDFs for a particular surface. Most commonly, BRDFs are measured with instruments such as the gonioreflectometer. There are a few methods to obtain BRDFs analytically using an empirical model as can be seen from one of the earliest works by Phong [79] which uses a diffuse component and a cosine weighted specular lobe. Later on, Blinn [12] improved this model in an attempt to make it more realistic by adding a specular

component. Ward proposed a model [97] that also supports effects like surface anisotropy. The BRDF model proposed by He et al. [33, 34] considers Fresnel reflections, surface micro-geometry, subsurface scattering, and also self-shadowing. It can model specular reflections and it also supports modeling rough surfaces. Other than these, [55, 58, 98] have also introduced new ways of modeling the BRDF function amongst which [21, 42, 75] fall under the physical BRDF models.

Acquisition: The acquisition of the BRDF (Equation 2) at a surface point P involves measuring the amount of the incoming radiance that is reflected at P for all combinations of ($\omega_i \in \mathbf{S}^2_-, \omega_o \in \mathbf{S}^2_+$. We use $\mathbf{S}^2$ to denote the unit sphere in 3-space i.e. the set of all possible directions in which light can flow. $\mathbf{S}^2_+$ consists of all vectors pointing away from the surface, while $\mathbf{S}^2_-$ consists of vectors pointing into the surface. Figure 3.3 describes part of a general BRDF measuring process.

Acquiring BRDF measurements under lab conditions is a straight-forward procedure which usually involves the use of a gonioreflectometer located in a dark room. Having no additional light sources avoids interference with the measurements. A spotlight illuminating a sample placed at the center of the sphere is mounted on a spherical arm. Its position is varied and a sensor captures the amount of light bouncing off the sample.

Although a lot of research has been done for acquiring BRDFs [23, 31, 64, 72] for models placed indoors under controlled environment and lighting setup, it is not well suited for outdoor environment and lighting conditions. Also, for both small and large scale stationary objects which are located outdoors, acquiring BRDF measurements is an extremely difficult and nontrivial task primarily because of the complex illumination and the fact that the environment cannot be controlled. During the day the sun serves as the only light source of the scene whereas at night a number of light sources (e.g. light posts, moon, flashlights, etc) may contribute to the illumination of the scene. In either case, moving the object around and having control over the illumination are almost impossible. In this case, unless there are strong assumptions made about the scene, acquiring the BRDF measurements for all surface points of the scene becomes intractable.

Perhaps the most popular work on the acquisition of BRDFs of an ancient structure located outdoors under complex illumination condition was the work of Debevec et al in [25, 26]. In their work, the authors had ground-access to the structure during the night which allowed them to capture BRDF measurements using a custom-made gonioreflectometer-like device from four representative

areas of the structure. Based on the assumption that the entire structure could be represented by those 4 BRDFs, they could relight the structure under novel lighting and produce very realistic renders.

In our case there are additional restrictions which exacerbate the difficulties of acquiring BRDFs: (i) Unlike [25, 26] we are not putting constraints on the number of different materials/BRDFs that can be present in our test scenes. Each scene can have a single or multiple arbitrary materials, (ii) Multi-view images have to be available which are captured from a certain distance, orbiting the scene, (iii) As we plan on to train a deep neural network to internally estimate the reflectance properties, we need a lot of training data from different viewpoints and which are composed of different reflectance properties.

Representation. Assuming all the problems relating to measuring BRDFs are overcome, there is still the issue of storage and representation. Using dense/full BRDF measurements can provide the most accurate results. The BRDF for a particular pair (ω_i, ω_o) can be retrieved or interpolated from nearby samples. This however increases computational complexity and imposes a significant cost for performing effective sampling during rendering. In this work, we use a combination of phenomenological and physically-based scattering models.

A diffuse surface scatters incident illumination equally in all directions. These types of surfaces are modeled as a Lambertian reflection model. This is a phenomenological model that assumes that the BRDF is constant and is given by $f_r^L(P, \omega_i, \omega_o) = \frac{\rho}{\pi}$ where ρ is the albedo i.e. the fraction of the arriving light energy that is scattered. The reflected radiance L_o varies linearly with the incident radiance L_i and is given by $L_o = f_r^L(P, \omega_i, \omega_o)L_i = \frac{\rho L_i}{\pi}$.

The Lambertian reflection model does not account for cast shadows or specularities. Specular scattering is modeled using the microfaceted Torrance-Sparrow BRDF [93] given by,

$$f_r^{TS}(P, \omega_o, \omega i) = \frac{D(\omega_h)G(\omega_o, \omega i)F_r(\omega_o)}{4cos(\theta_o)cos(\theta_i)} \tag{3}$$

where $D(.)$ is the Trowbridge and Reitz microfacet distribution function [94], $G(.)$ is the geometric attenuation term, $F_r(.)$ is the Fresnel function, and θ_o and θ_i are the angles between the normal at surface point P and the outgoing direction ω_o and incoming direction ω_i respectively.

The relation between the reflected radiance L_o and the incoming radiance L_i is then given by $L_o = f_r^{TS}(P, \omega_i, \omega_o)L_i = L_i \frac{D(\omega_h)G(\omega_o,\omega i)F_r(\omega_o)}{4cos(\theta_o)cos(\theta_i)}$.

To summarize, in our work the Lambertian reflectance model has been used to model the diffuse component, microfaceted Torrance-Sparrow BRDF [93] was used to model the specularities, whereas, the Trowbridge and Reitz [94] microfacet distribution function handled the surface roughness. Combining all of these would allow us to model the external appearance of a material.

Chapter 4

Estimating Reflectance Properties and Relighting

In this chapter, we discuss our approach towards predicting the reflectance properties from multi-view images of a scene. We present a synthetic multi-view dataset of spheres with varying materials, which is used to create the reflectance maps to train our network. The training process of the network is also discussed in detail. We justify our design choices for the system pipeline throughout this chapter. Performance comparisons using an ablation study is also discussed in the latter part. We conclude this chapter by demonstrating some of the results obtained using our technique.

4.1 Overview

In our work, the acquisition of the multi-view images involves the camera orbiting around the scene from different altitudes. Usually, during the capture process, the sun serves as the single directional light source illuminating the scene and is considered to remain fixed throughout the capturing process. To incorporate the light direction and properties with our system, additional information along with the images describing the illumination are required. Typically, measuring the sun's direction and intensity requires additional captures using specialized equipment [24]. Although this may be appropriate when ground access to the scene is possible, however, it is not always practical. Also, for any multi-view images, it might not be possible to capture the lighting details once the images

are taken. For this reason, we avoid the explicit measurement of light direction ω_i by training a neural network to predict the reflectance parameters given samples of the reflected radiance of an object under arbitrary directional illumination conditions. Formally, the neural network $\Phi(.)$ learns the mapping function $\Phi : I_\mu \longrightarrow \mu$ where μ is a 7-vector representing the reflectance properties. The reflectance properties μ are defined in terms of the diffuse k_d^3 and specular k_s^3 components, and the surface roughness r. The fact that the incident radiance is arriving from a single fixed direction ω_i facilitates the representation of the reflectance properties of a surface point P as a unit sphere $\mathbf{S}^2$ where the reflected radiance value is stored at each point on the sphere's surface corresponding to each outward direction $\omega_o \in \mathbf{S}_+^2$. Thus, the input to the network I_μ is a reflectance map created using the outgoing radiance for each outgoing direction $\omega_o \in \mathbf{S}_-^2$ in a spherical coordinate system.

For the first part of the system, a wide residual network has been trained on multi-view images of varying material properties. As the network will estimate the material properties from images, the network has to learn the mapping between material properties and appearance. Therefore, there should be sufficient training data with a variety of material properties. Also, to learn the relationship between the reflectance properties and appearance changes based on viewpoint, the dataset should contain images from varying viewpoints for a single material. Therefore, we generated this synthetic dataset keeping the following things in mind,

- Coverage of a wide range of material properties
- Variation in viewpoints
- Single directional light source imitating the effects of the sun

4.1.1 Materials

The materials have been defined using a set of 3 different parameters in our system. Table 4.1 lists these properties and their purposes. The three parameters correspond to the diffuse, specular, and roughness components of a material. The diffuse and specular components are made up of a triplet of values. Therefore, the diffuse and specular components each use a 3-vector $[K_r, K_g, K_b]$ where K_r, K_g and K_b control the intensities of the diffuse and specular components over the red, green and blue channels. Whereas, roughness uses a single parameter and can be denoted by r. Therefore, any

material in our system can be defined using a length 7 vector comprising of $[k_d^r, k_d^g, k_d^b, k_s^r, k_s^g, k_s^b, r]$. Each of the values in the vector is a floating point number that ranges between 0 to 1.

Material Property	Description
Kd	Diffuse property
Ks	Specular component
Roughness	Surface roughness

Table 4.1: Properties that are used to define a material.

Effects on appearance: Each of the 3 parameters controls a different aspect over the appearance of an object. The diffuse component (Kd) generally sets the base color of any object which is more commonly referred to as the diffuse color/albedo. The specular component (Ks) is responsible for the color of the reflections visible on shiny surfaces. Roughness controls the shininess and works along with the specular parameter to give off reflections. The renders in Figure 4.1 illustrate the variations the parameters create in the appearance of a sphere. Each row exhibits the changing effects of one specific material property. The three images on the first row were produced with varying the K_d component such that it creates a red, a green and a blue sphere. For these three images the K_d component was assigned to be [0.5 0.0 0.0], [0.0 0.5 0.0] and [0.0 0.0 0.5] respectively. The K_s and roughness (r) properties were kept fixed and the illumination and background are also unchanged throughout the process. The second row shows the effect of changing the specular component K_s. Here, the first and third images have the same K_d. The K_s component controls the color of the specular shine and the highlights. The green tint in the reflection and the specular hotspot in the first two images were produced using a K_s value of [0 1 0]. For the third image, K_s was changed to show a red/pink glow to the sphere. Lastly, the third row shows the effect of changing the roughness (r). It controls the shininess and strength of reflections.

4.1.2 Lights

Changing the position, direction, color or the type of light can produce very different results on the same material. PBRT provides a number of common lighting methods alongside environment maps. For the generation of the training data, our system uses a single directional light source with a constant white color. We assume the light does not move throughout the process. The directional

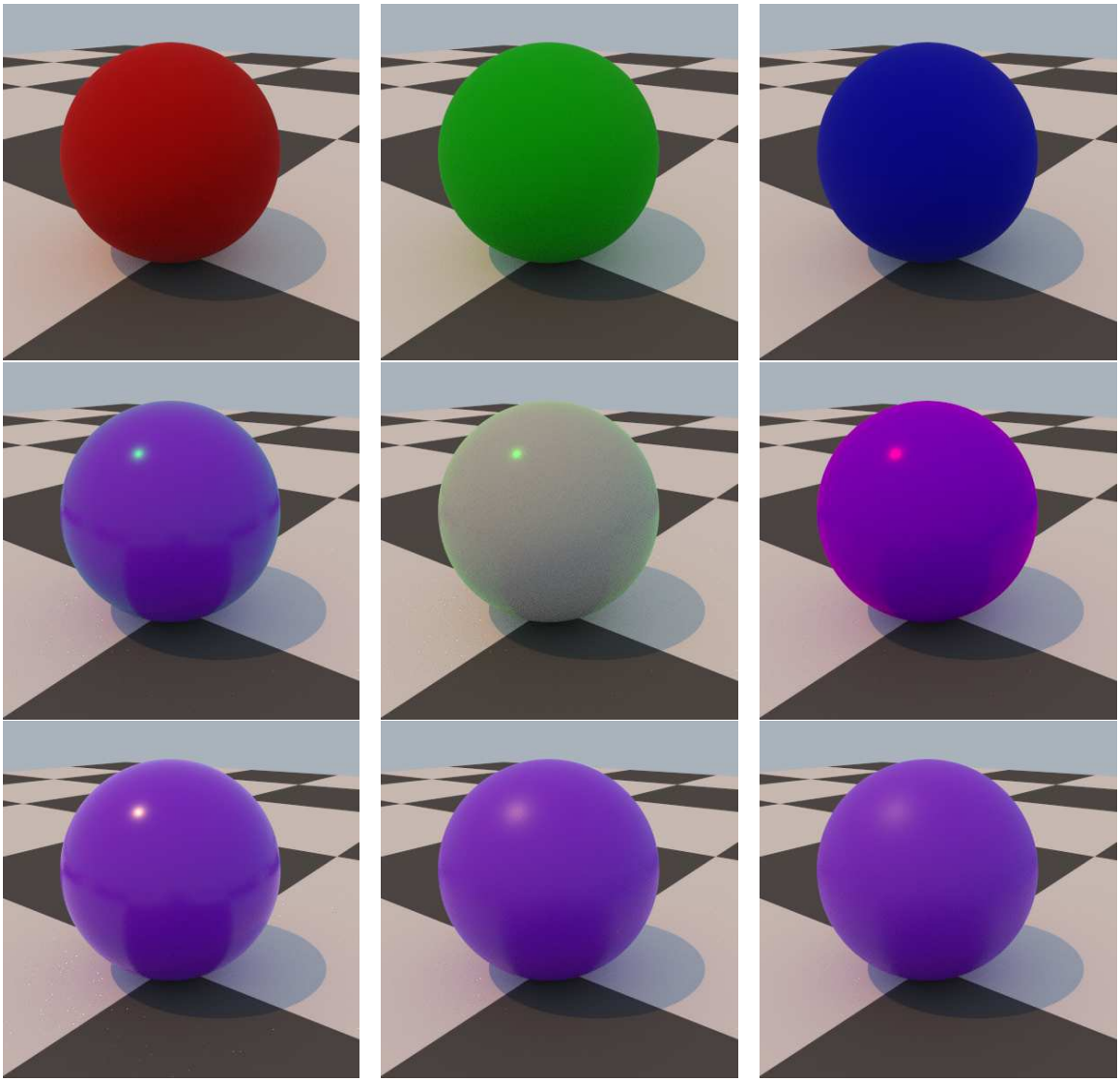

Figure 4.1: Each row highlights each distinct property used to define a material. The topmost row, from left to right illustrates the effect of Kd. Changing the value of Kd alters the diffuse color of the material, which is tweaked in this image to change the color to red, then green and finally to blue. The second row demonstrates the effect of Ks, the specular component, on the overall color of the sphere. The Ks controls the reflectivity component, and also the color of the specular hotspot. Giving it a green-ish color produces a green highlight and also adds a green glow to the brighter reflections. The first two images show the results of having a Ks value on the sphere that creates green highlights on two different diffuse colors. The third image shows the effect of Ks, this time producing red with a purple diffuse color. Finally, the last row (the bottommost) shows the effects of changing the roughness parameters. As we go from left to right, a stronger roughness value has been used to produce less prominent highlights and reflections. This is the reasons why the top row does not contain any reflections, as they all had a very high roughness value.

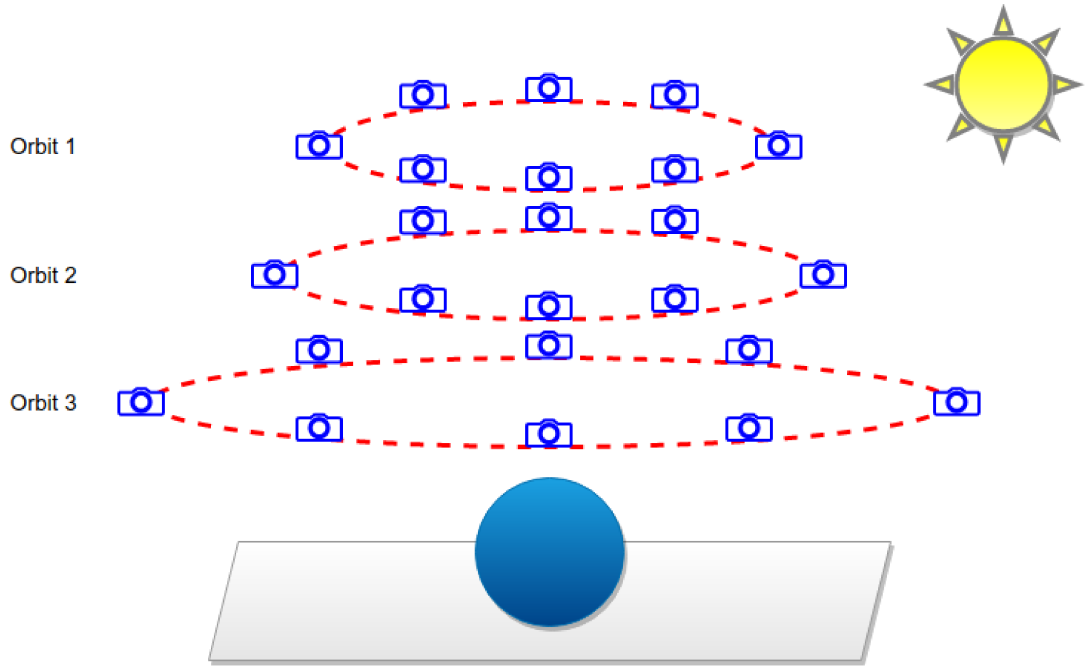

Figure 4.2: The camera viewpoint is varied during the dataset generation. On each orbit, it takes 8 images looking at the origin where the sphere is placed. It performs a total of 3 orbits from different altitudes, thereby, capturing images from 24 different viewpoints for a single material.

light source will simulate the effects of the sun by staying stationary in one location and giving off a white color throughout the camera orbits.

4.1.3 Camera viewpoints

Identifying materials from appearance is a non-trivial task for humans. For people, it is simple to distinguish a shiny aluminium jar from a rough sandpaper, by the variations in appearance from different viewpoints. One of the ways to make the neural network learn these variations is by providing a large number of images from different viewpoints having subtle changes in the appearances i.e. different positions of the specular highlights, changes in reflections, different shadowed regions etc. During training, the network should try to establish the relationship with the material properties and the changes of patterns in appearance within the images that have the same materials. For this reason, the dataset has multiple images for each material.

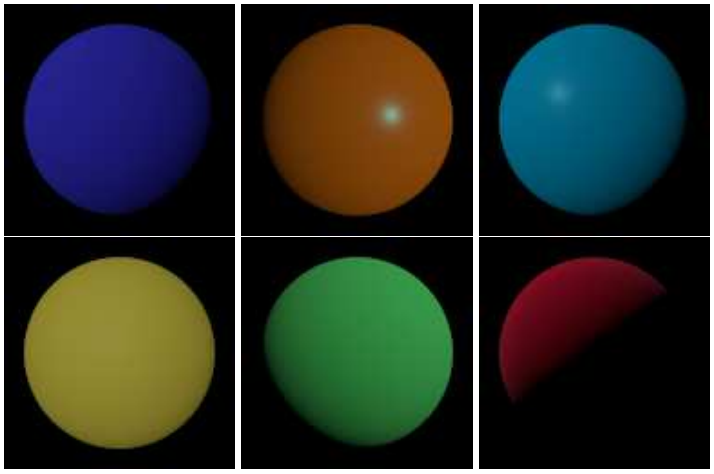

Figure 4.3: A set of images from our generated synthetic dataset comprising of spheres rendered using different materials.

4.2 Dataset and Reflectance Maps Generation

4.2.1 Generating realistic spheres dataset

We have used PBRT for generating our training data. PBRT is a physically-based renderer, that has path tracing support and is open-sourced. PBRT stands on a few base classes i.e. lights, shapes, materials, etc. This makes extending the system with additional features easy. We extended PBRT to generate a number of information for the image that is being rendered. The source code was modified and additional scripts were used to meet the needs of the data generation. To ensure variability in the training dataset the following conditions were varied: **(i)** reflectance properties (diffuse, specular, roughness) and **(ii)** viewpoints. The light direction and color were fixed throughout the data generation process. The scene, therefore, consisted of a unit sphere with uniform, isotropic, and opaque reflectance properties, a perspective camera, and a single directional light source. The camera orientation is defined in spherical coordinates (θ_c, ϕ_c) with $\theta \in \{\frac{\pi}{9}, \frac{5\pi}{18}, \frac{4\pi}{9}\}$ and $\phi \in [0, 2\pi]$ at intervals of $\frac{\pi}{4}$. Figure 4.2 shows the camera positions that were used for orbiting around the sphere at different altitudes. The reflectance properties were defined in terms of the diffuse, specular and roughness components given by $[k_d^r, k_d^g, k_d^b, k_s^r, k_s^g, k_s^b, r]$ where k_d^r, k_d^g, k_d^b are the diffuse component, k_s^r, k_s^g, k_s^b are the specular component, and r is the surface roughness. These parameters are set to different values in their respective range creating a total of 24,000 unique materials. The

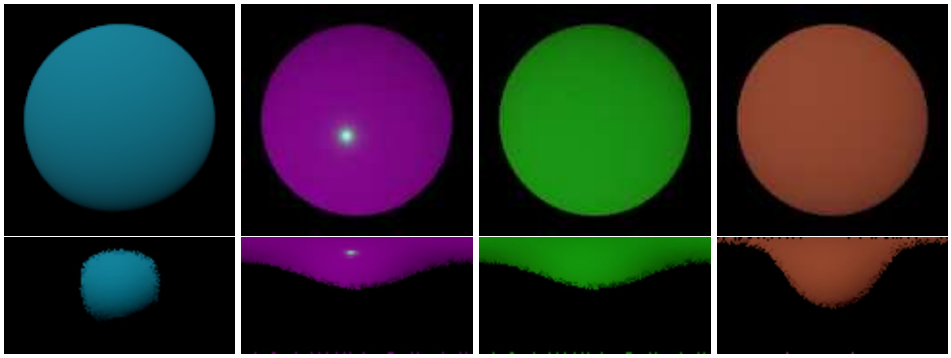

Figure 4.4: The top row shows the images of a few spheres for which the reflectance maps were generated. The associated reflectance map for each sphere is in the bottom row.

parameters are set according to the following two configurations: 20,480 of the materials are uniformly sampled and 3,520 materials are formed by randomly sampling. Each of the materials was rendered from 24 different view points which resulted in a total number of 576,000 images. Figure 4.3 shows a few of the images from our dataset of spheres.

4.2.2 Generating reflectance maps from spheres

For a surface point, a reflectance map shows how the material reflects light in different directions within the sphere centered on the point. We used the spheres on our dataset to create the reflectance maps. Each of the spheres on our dataset had a uniform isotropic material. Therefore, all the points on its surface share the same material properties. As the light is fixed at a fair distance and the radius of the sphere is negligible compared to the distance of the light from the sphere, we can assume, two different points on the surface of the sphere do not create a large difference in angle with the light. For each point on the sphere, we calculate its spherical coordinates in terms of (r, θ, ϕ) where the r is the distance of the point from the origin of the sphere whereas θ and ϕ represent the azimuth and altitude for that point. Using this, every point of the sphere adds a contribution to the reflectance map. A few samples of this process can be seen in Figure 4.4. Instead of generating the reflectance map of shape $180 \times 360 \times 3$ that would correspond to all possible values of (θ, ϕ), we use a smaller sized map of $60 \times 120 \times 3$. Experiments have shown that accuracy remains unaffected while training time is reduced.

4.3 Network Training

4.3.1 Network Architecture

A Wide ResNet-50 [105] architecture was used as the deep neural network model for our system. The model was chosen after evaluating a few different architectures on a number of images in the test set. The results from the experiments are listed in Table 4.2 and discussed in detail in Section 4.4.1. For training, the shape of the input image is $60 \times 120 \times 3$ and the output is a 7×1 vector. A sigmoid activation function was used after the fully connected layer as the 7 dimensional output vector comprises of floating point values ranging from $[0 - 1]$. A number of data augmentation techniques were used to introduce sparsity in the training data and add robustness to the model towards small variations during inference. A $80 - 20$ split was made for training and validation. An additional 30,000 images were generated and retained for subsequent testing. A step learning rate was used starting from 0.0003 and gradually reduced by 10% after every 20 epochs.

4.3.2 Loss function

The network uses a weighted sum-of-squared-errors as the loss function. Since the parameters (k_d, k_s) are size 3-vectors, we scaled the weight for (r) by a factor of 3. This was done to make sure that an error in the roughness parameter is given equal weight as for k_d and k_s. Since values were sampled uniformly to create the materials for training, there are samples that have high roughness but varying k_s. These different *rough materials* that essentially have different k_s values, produce almost identical images since the value of k_s has little or no impact on the final render. To address this, another weight was used to supervise the network to stop penalizing for the k_s parameter whenever a high value of r was seen. With the addition of these two weights, the loss function was designed as Equation 4 which the network tries to optimize on.

$$L(\omega, \hat{\omega}) = sse(k_d, \hat{k_d}) + \theta_1 \times sse(k_s, \hat{k_s}) + \theta_2 \times sse(r, \hat{r}) \tag{4}$$

$$where, \theta_1 = (1 - \sqrt[3]{r}), \theta_2 = 3$$

In Equation 4, L is the loss that takes in the ground truth material properties ω and the prediction $\hat{\omega}$. The "hat" symbol indicates the predicted properties. The loss incurred by the prediction for k_d is simply calculated using the sum of squared errors (SSE). Whereas, for k_s and r, two scaling factors θ_1 and θ_2 are applied. θ_1 controls the weight updates for k_s depending on the value of r. θ_2, on the other hand, makes the error of $(r, \hat{r})$ equally important as the other parameters.

4.3.3 Data augmentations

Data augmentation was introduced during the training process to increase variabilities and to add sparsity to the reflectance maps. Salt-pepper noise was introduced to arbitrary pixels. Due to the nature of the data acquisition process for multi-view imagery, it is common to have incomplete data for a surface point. This is due to occlusion or invisibility from the cameras. The number of images taken around the scene is the maximum number of samples that a point can have on its reflectance map, making the map very sparse. We tried to address this with the addition of salt and pepper noise and by randomly activating just a few samples within the reflectance map while making the rest of it black. Additionally, random vertical and horizontal flips were applied to the images to increase variability and for reducing overfitting.

4.3.4 Training and Inference

Training: The Wide-Resnet50 architecture was trained for 90 epochs. Using a step learning rate with a step size of 20, produced lower validation loss and a smooth training loss. The network was trained in the Compute Canada cluster with a single Nvidia V100 GPU. The network and input were as explained in Section 4.3.1. Instead of using dropout as a regularization technique, our system relies on the data augmentations and weight decay to avoid overfitting on the training set. We used a batch size of 256. Besides performing on the fly augmentation during training, we normalize the data during training, validation and testing to have a mean of 0 and a standard deviation of 1.

Inference: For testing, the system requires multi-view images, the camera properties and the geometry of that scene. If the geometry is not available, it can be generated using COLMAP [87, 88] or VisualSFM [100, 101] before inputting to our system. At the first stage of our system, a buffer is created for all the images using the camera poses and the 3D model. Lets denote the n number

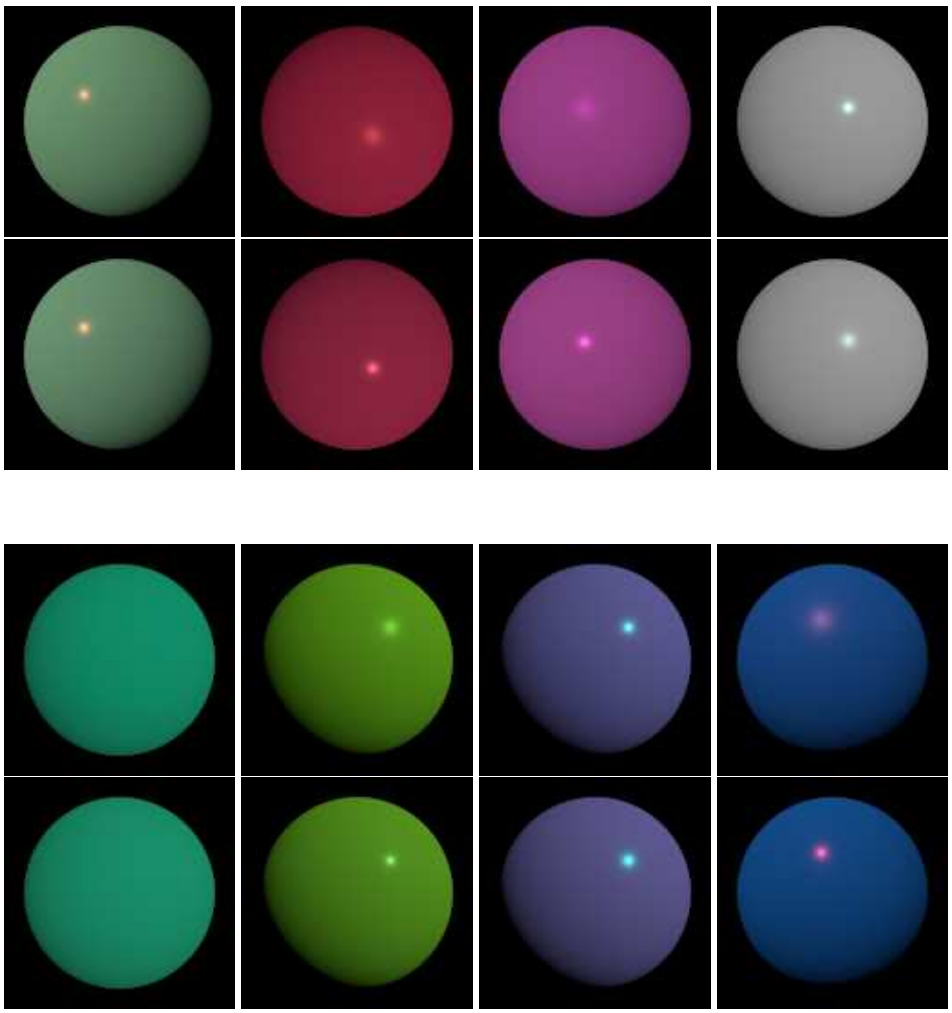

Figure 4.5: Some of the results of predicting material properties and re-rendering using the predicted values on images of spheres. In each of the two subfigures, the top row represent the prediction. and the row next to it represents the ground truths.

of multi-view images as $I = \{I_1, I_2, I_3, ..., I_n\}$ and the camera poses as $C = \{C_1, C_2, C_3, ..., C_n\}$, then the system would generate buffers $B = \{B_1, B_2, B_3, ..., B_n\}$ where $height(I_i) = height(B_i)$, $width(I_i) = width(B_i)$ and $i = 1, 2, ..., n$.

The buffers record a few pieces of information about the closest intersection for each of the pixels. These information include, **(i.)** ID of the closest intersecting triangle from the mesh, denoted by (t_{id}) **(ii.)** World coordinate of the intersecting point (p_x, p_y, p_z) **(iii.)** Normal of the intersecting point (n_x, n_y, n_z) **(iv.)** Projected point in pixel/image space (u, v). Therefore,

$$(t_{id}), (p_x, p_y, p_z), (n_x, n_y, n_z), (u, v) \in B_1, B_2, B_3, ..., B_n$$

After calculating the buffers, the system proceeds by reading in a set of (I_i, B_i, C_i). Then for each (u, v), it collects the RGB value from I_i. Then it has to find the index to place this RGB value in the reflectance map associated with t_{id}. The system calculates the index by computing the angle between the normal (n_x, n_y, n_z) and using the viewpoint direction from C_i. The resultant angle (θ, ϕ) in spherical coordinates is used to place the extracted RGB value into the reflectance map associated with t_{id}. This step is repeated for all the pixels in I_i. After all sets of (I_i, B_i, C_i) are processed, all the triangular surfaces in the scene will have their reflectance maps computed.

A single triangle t_{id} can be very small compared to the size of the entire model as it can be seen from Figure 4.6. Therefore, the visible part of any triangle t_{id} in an image I_i might not have a lot of (u, v) recorded against itself in the buffer. This results in making the reflectance maps very sparse. The number of images that are available for a test scene will also impact the number of samples. The reflectance maps for each of the triangles go into the network for prediction. The output of the network is then assigned to each of the triangles individually. A slight error in the estimation can cause the re-rendered image to appear triangulated.

4.4 Evaluation and Results

We conducted a series of experiments to evaluate our two-stage approach. For comparison, several network architectures were evaluated on the test set. Based on this performance we used the best performing network in our system pipeline. For the second stage, a number of 3D models and their

associated multi-view images were used to relight them. The next subsections contain more details.

4.4.1 Comparison of networks

For the ablation study, separate 30,000 images were held out separately from the training/validation images. The test set was designed to exhibit a lot of variation over each of the material property parameters. We conducted our experiments on a total of 8 different network architectures. We experimented on the different variants of the ResNet architecture [32]. To experiment on deeper and wider networks we have used multiple variants of ResNext [102] and WideResNets [105] in our tests. An older and simpler model like AlexNet [53] was chosen to show how a model with fewer parameters and less depth fits our data. The most recent architecture and state-of-the-art network that we have used is the EfficientNet [92].

The reflectance maps in the test set had variations in viewpoints and materials. The predictions from the network were used to re-render the spheres under original lighting and from the same viewpoint. To analyze the performance of each of the networks the metrics had to do image to image comparisons. We report our results on 3 different metrics. **(i.) Peak Signal to Noise Ratio (PSNR):** PSNR is regularly used to evaluate the quality of digital signal transmission and is also widely used in researches involving image to image comparison. It is a variation of the mean squared error metrics and relies on pixel-by-pixel comparison. A *higher* value of PSNR translates to *lower* differences between the two images. **(ii.) Normalized Root Mean Squared Error (NRMSE):** Normalized Root Mean Squared Error simply compares each of the pixels of the original with the predicted image. In our case, a *lower* value indicates a *better* prediction of the reflectance properties. **(iii.) Structural Dissimilarity Index (DSSIM) [96]:** SSIM is a metric that derives image distortion using the changes in structural information, while also taking into account simpler changes like luminance, contrast, etc. Unlike, RMSE or PSNR which only calculate the errors amongst pixels, SSIM utilizes structural information as well. It assumes that the pixels have strong inter-dependencies when they are spatially close within a neighborhood. These assumptions utilize important information from the images about the structure of objects within the images. While SSIM gives the result in terms of similarity, Structural dissimilarity (DSSIM) calculates the differences in the images and can be derived from SSIM using Equation 5. A *lower* value of DSSIM

Architecture	PSNR	NRMSE	DSSIM
AlexNet [53]	41.6841	0.0564	0.00676
ResNet-18 [32]	45.4900	0.0354	0.00391
ResNet-34 [32]	46.3642	0.0349	0.00401
ResNet-50 [32]	47.2551	0.0325	0.00398
Wide Resnet-50 [105]	**48.0526**	**0.0287**	**0.00326**
ResNext-50 [102]	47.4328	0.0309	0.00375
ResNext-101 [102]	47.5832	0.0307	0.00357
EfficientNet B0 [92]	47.7031	0.0306	0.00358

Table 4.2: Results on the test set across different model architectures using PSNR, NRMSE and DSSIM as metrics. The test set had 30,000 samples.

indicates *lower* errors between the two images.

$$DSSIM(\mu_1, \mu_2) = \frac{1 - SSIM(\mu_1, \mu_2)}{2} \tag{5}$$

Here, DSSIM is the Structural Dissimilarity Index between the two images (μ_1, μ_2), calculated using the SSIM of (μ_1, μ_2). A *lower* value indicates a *lower* difference between the original and predicted image.

To summarize, from the experiments on the test set, **WideResnet-50** outperformed all the other networks. From Table 4.2 it can be seen that this network has the best score over all the three metrics in comparison with the other networks. Therefore, we use this model with our system and for relighting the scenes.

4.4.2 Analysis of image relighting

For testing the network's performance on complex scenes, we needed to have access to a multi-view image dataset and the geometry. We collected a number of free scenes available online and created multi-view images of the model using PBRT. The number of images varies between 200-300 for each of the scenes. For generating the multi-view images, a directional light was used with the camera orbiting around the object from a constant distance. The reflectance maps for all the surface of the scenes were recorded using the approach described in Section 4.3.4. One key point is that there is **no information sharing** amongst the surfaces of the meshes. As our approach is estimating

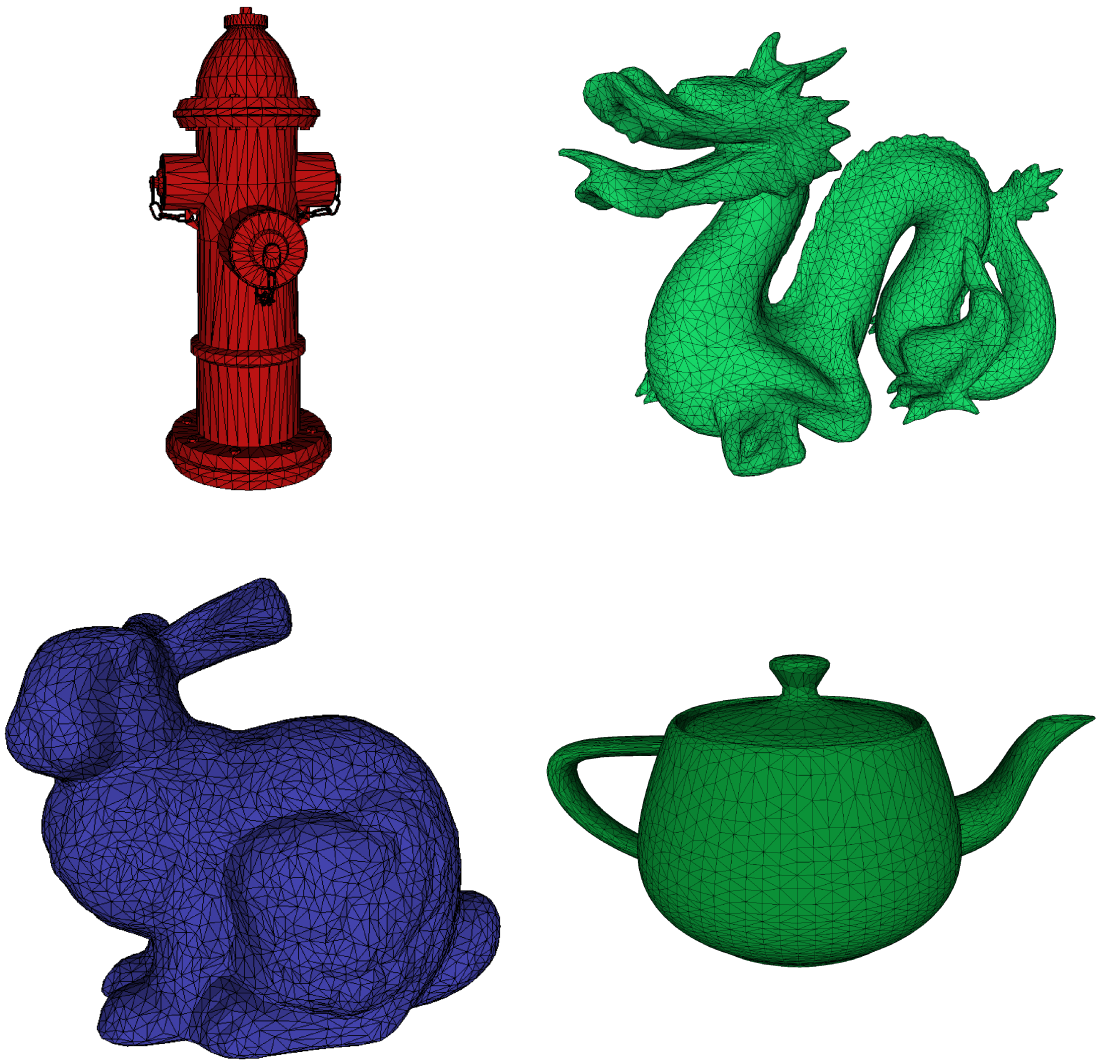

Figure 4.6: This figure illustrates the density of the triangular surfaces in the scene geometry. A more detailed mesh should have a larger number of triangles, making the renders look smoother, but the task of prediction more challenging.

Model	Triangles	Inference time (seconds)	Time/triangle (ms)
Fire Hydrant [3]	6148	6.3449	1.0320
Dragon [22]	13,377	13.5027	1.0094
Stanford Bunny [95]	8038	8.2926	1.0317
Utah Teapot [13]	7809	8.1644	1.0455

Table 4.3: Timing comparisons for inference on the test scenes that are shown in Figures 4.7, 4.8, 4.9 and 4.11

materials **for each triangle** in the geometry, the predictions have to be consistent and accurate to generate a smooth render. Usually, the larger the number of triangles, the smoother and more accurate the geometry is. However, it makes the task of relighting more challenging. Since the sizes of the triangles are small, they do not take up many pixels in the image. Therefore it reduces the number of samples recorded against a triangle. During inference, the task becomes challenging due to the reflectance maps being very sparse. Neighboring triangles that share the material properties can look distorted and triangulated in the rendered image if the predictions vary even slightly. A few examples of the triangulated meshes can be seen from Figure 4.6.

Figure 4.7, 4.8, 4.9 and 4.11 show a few results obtained using our technique. For each of the models, we display results under : **(i.)** Simple directional light **(ii.)** Realistic indoor and outdoor illumination setting. From the results, it can be seen the network is picking up the diffuse colors and also picking up the specular shines from the images. The images do not have any triangular artifacts which can happen due to incorrect estimation by the network. Figure 4.7 shows an example scene. Here the prediction of the diffuse color is consistent throughout the changes in illumination. The network picked up the specular highlights from the original images. The predicted images do not contain any holes or distortions. Similarly in Figure 4.8 and Figure 4.9 produce accurate results without any visible artifacts. Although in Figure 4.9 there are a few spots on the top of the model, but that is the result of the part of the model being on the opposite side of directional light and thereby producing darker regions. Finally, Figure 4.11 also produced results close to the original. Despite minor darkness in few parts at the bottom of the teapot occurring due to the shadow, the network picked up the rest of the details of the model.

For predicting the material properties from reflectance maps, our trained Wide ResNet-50 model

was used. The batch size was kept to 256 for all of the test scenes. For inference, we used a machine with a Core i7 processor and a single NVidia 2080ti GPU. Figure 4.7 had a total of 6148 triangles. The inference time taken was 6.3449 seconds. That translates to 1.0320 *milliseconds per triangle*. Considering the density of triangular mesh and the number of reflectance maps generated, our technique estimates the material properties within a few seconds. The timings for the rest of the models are listed in Table 4.3

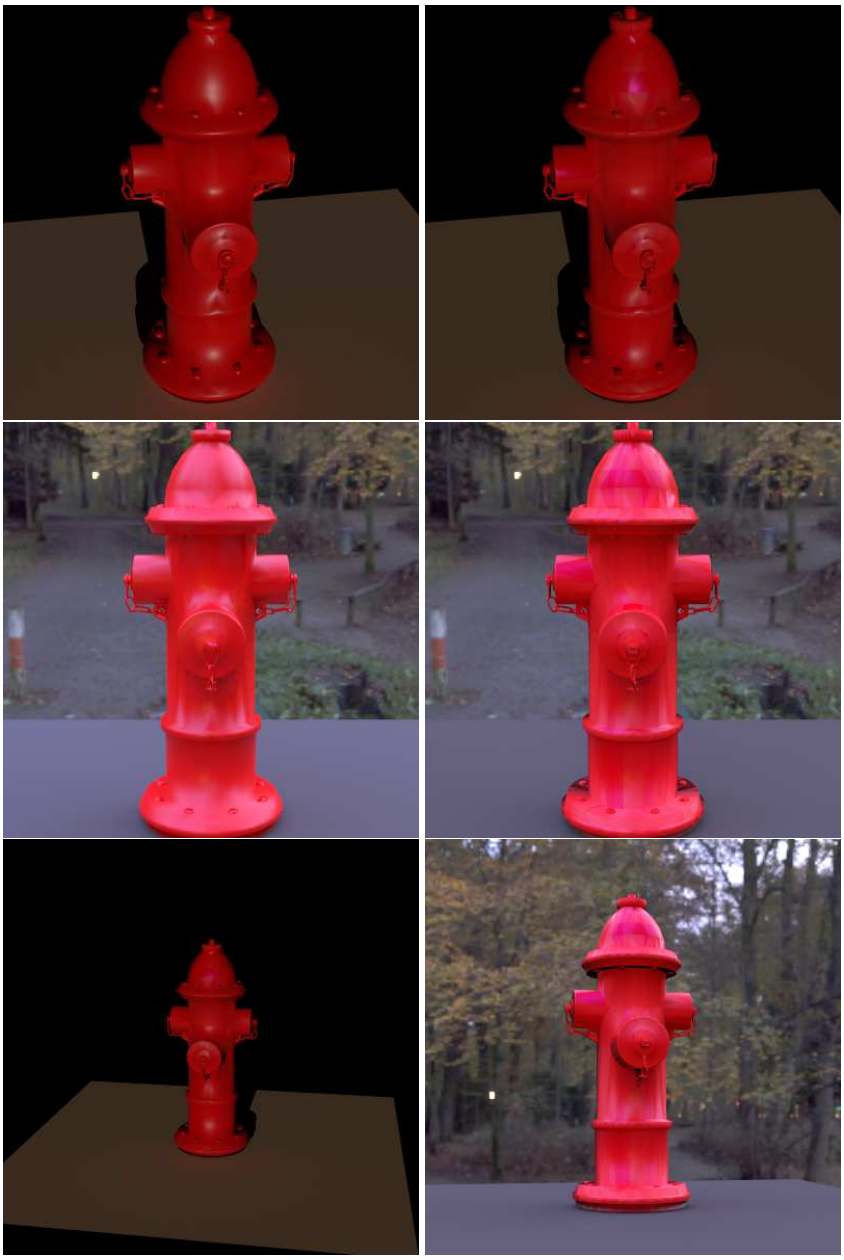

Figure 4.7: This figure shows a re-illuminated scene consisting of a fire hydrant [3] under two different lighting conditions. **Top row:** corresponds to one of the images of the multi-view image set for this model. There is a single directional light source. The multi-view images were used to generate the reflectance maps. The estimated reflectance properties are used and the scene was re-rendered as it is shown on the right (2nd column). **Middle row:** For this row, the image is under a natural daytime lighting condition, created using an environment map (HDR lighting). For the *top two rows*, the first column corresponds to the ground truth, and the second column is the results of our technique. **Bottom row:** shows the ability of our technique to render from novel viewpoints. Both of the camera positions used in this row were not present in the original multi-view images. The model has a total of **6,148** triangles with an inference time of **6.3449** seconds.

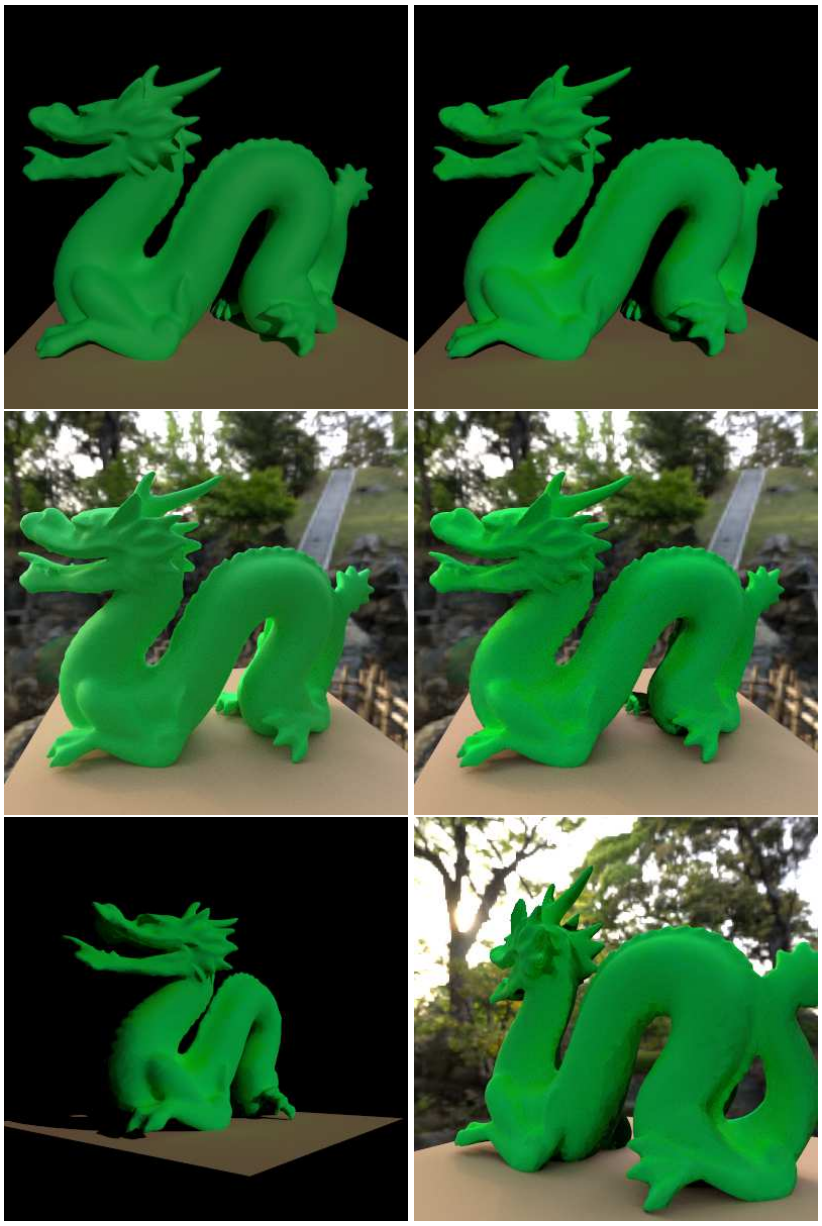

Figure 4.8: This figure uses the renowned dragon model, first appearing in [22]. Similar to the previous figure, the top two rows show relighting under two different light setups. **Top row:** The first image is one of the multi-view images under a single directional light. The image on the right is the results of relighting. **Middle row:** This is a comparison under a daytime lighting illuminated by the sun. The left and right images show the ground truth and prediction respectively. Finally, the **bottom row** shows two different renders from completely novel viewpoints. The model comprises **13,377** triangles, for each of them the reflectance properties were estimated exclusively. The inference time was **13.5027** seconds.

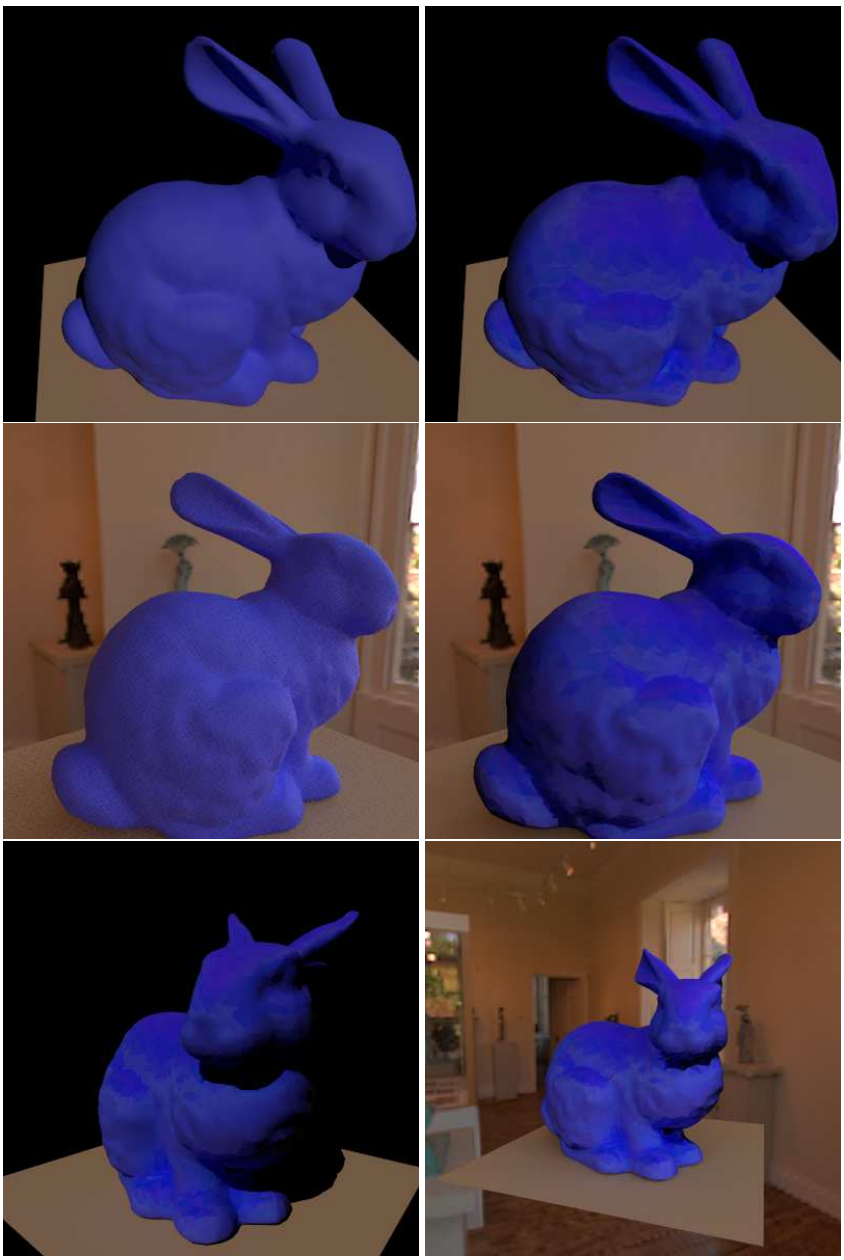

Figure 4.9: The model in this figure is the widely used Stanford Bunny [95]. The **top row** is under a directional light setup. The ground truth image is taken from the original images for this model. The **middle row** shows the model under indoor lighting conditions. For the first two rows, the images on the left are the ground truth and the predictions are on the right. The **bottom row** shows the model rendered from two novel viewpoints under a directional light and an indoor illumination respectively. This model has a total of **8,038** triangles for inference and the time taken was **8.2926** seconds.

Figure 4.10: The Utah teapot model [13] was tested under two different light setups. The first two rows show the comparisons of the ground truth and the prediction respectively in first and second columns. The **top row** shows results under a directional light setup and the **middle row** is in an indoor environment setup. In both of the illumination conditions the colors were estimated accurately, and the small reflections on the upper part of the teapot is present in re-rendered images. The **bottom row** shows two additional images rendered from new camera positions not present in the original images. This model has a total of **7,809** triangles and the inference time was **8.1644** seconds.

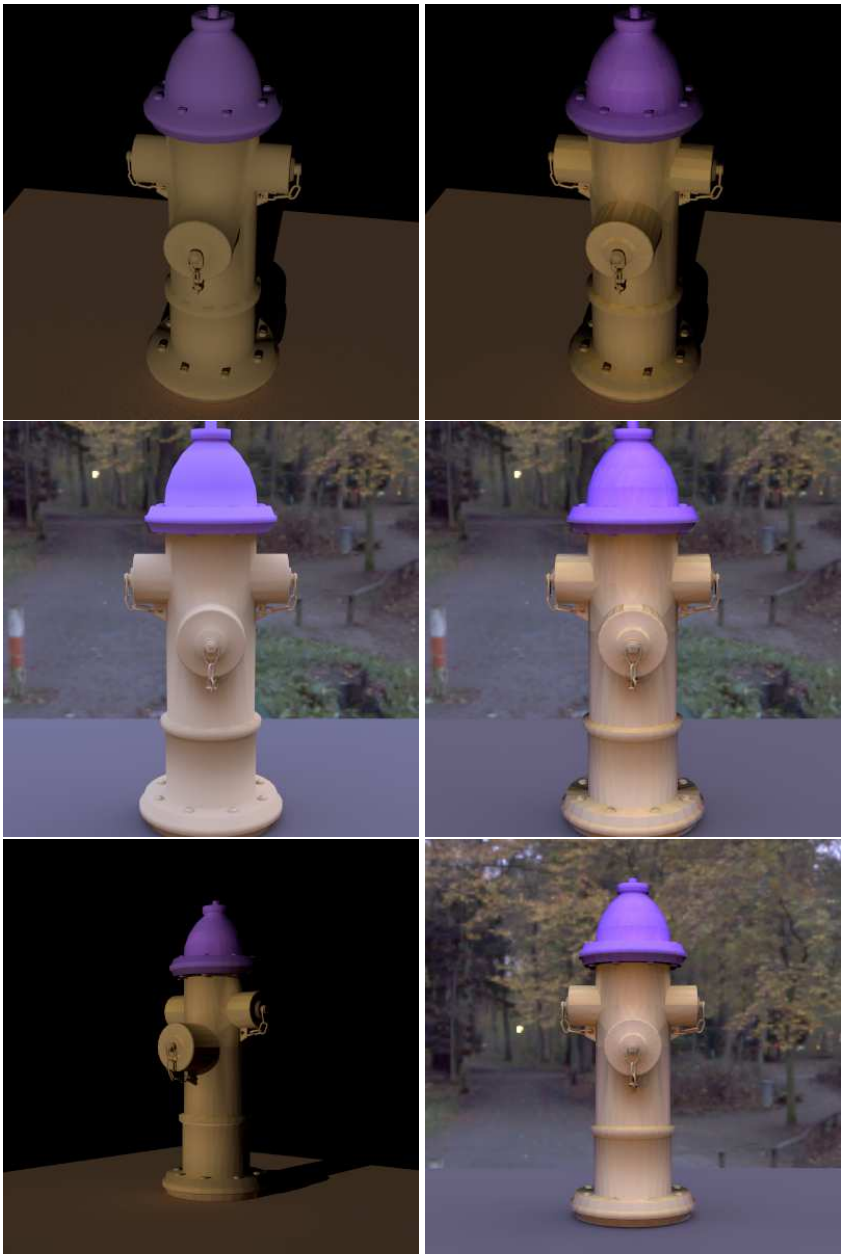

Figure 4.11: Finally the fire hydrant [3] model was relit again by using different materials on its surface. The top part use a rough plastic like material with a purple color. Whereas, the bottom part of the model has a wood like material. In this case as well the predictions look very close to the ground truth with a little triangulation on the parts for which there was a lower number of samples captured. The **top row** show the model under a directional light source. The **middle row** shows the same model relit under an outdoor light setup. And finally the **bottom row** show the model rendered from two different viewpoints not originally present in the multiview images.

Chapter 5

Conclusion and Future work

In this work, we presented a novel technique to **(i.)** Estimate reflectance properties and **(ii.)** To reilluminate scenes under novel lighting conditions. We justify our design choices by providing comparisons with different network architectures and by showing our results on a number of scenes. We show that our model can very quickly estimate reflectance properties quickly and can re-render realistically using physically based rendering.

5.1 Concluding Remarks

In our proposed technique, we have reformulated the problem of image relighting as a two-stage process following a divide and conquer approach. As part of the first stage, a synthetic dataset of images of spheres having different materials and rendered from different viewpoints and under different illumination. Using this we compute the reflectance map and train our Convolutional Neural Network. We discuss the performance of our network and provide comparisons with different architecture using several image ti image comparison metrics. Our test set consisted of an additional 30,000 reflectance maps that were held out separately. We showed that, for a scene, our technique can estimate the reflectance properties within seconds. After identifying the reflectance properties we have used path tracing to realistically render the original scene by changing the illumination condition and by changing the camera viewpoints. Finally, we have presented relighting results several scenes under varying lighting conditions.

5.2 Future Work

Currently, the system is mostly limited to indoor scene under simpler lighting conditions. Hence, as a future work, we plan to extend this to support large-scale outdoor scenes mainly consisting of buildings, roads, landmarks, etc. Our dataset and technique was developed by already keeping these factors in mind. Therefore, we have used a single directional light source that acts as the sun. We have used multi-view images from multiple altitudes as it is usually done for Wide Area Motion Imagery (WAMI). The possible difficulties with outdoor images can be the number of images not being large enough which produces sparse samples in the relfectance maps, complexity arising due to occlusions, cameras not capturing the entire scene in full orbits etc. Once it is extended to support urban outdoor images, another extension can be to conduct a quantitative comparison over outdoor datasets consisting of LIDAR data and multi-view images.

Bibliography

[1] Arnold renderer. https://www.arnoldrenderer.com/. Accessed: 2020-09-23.

[2] Cycles renderer. https://www.cycles-renderer.org/. Accessed: 2020-09-23.

[3] Fire hydrant free low-poly 3d model. https://www.cgtrader.com/free-3d-models/industrial/machine/fire-hydrant-7ba25670-3f38-4a77-a0c9-56ce888c9df2/. Accessed: 2020-09-27.

[4] Maxwell renderer. https://maxwellrender.com/. Accessed: 2020-09-23.

[5] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[6] Takashi Abe, Takayuki Okatani, and Koichiro Deguchi. Recognizing surface qualities from natural images based on learning to rank. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3712–3715. IEEE, 2012.

[7] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 37–45, 1968.

[8] Dejan Azinovic, Tzu-Mao Li, Anton Kaplanyan, and Matthias Niessner. Inverse path tracing for joint material and lighting estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[9] Jonathan T Barron and Jitendra Malik. Color constancy, intrinsic images, and shape estimation. In *European Conference on Computer Vision*, pages 57–70. Springer, 2012.

[10] Jonathan T Barron and Jitendra Malik. Shape, albedo, and illumination from a single image of an unknown object. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 334–341. IEEE, 2012.

[11] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[12] James F Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 192–198, 1977.

[13] James F Blinn and Martin E Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.

[14] Gavin Brelstaff and Andrew Blake. Computing lightness. *Pattern Recognition Letters*, 5(2):129–138, 1987.

[15] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[16] Henrik / CC BY-SA. Ray tracing (graphics), 2020. [Online; accessed 21-September-2020].

[17] 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. In *Advances in Neural Information Processing Systems*, pages 9609–9619, 2019.

[18] Zhang Chen, Anpei Chen, Guli Zhang, Chengyuan Wang, Yu Ji, Kiriakos N Kutulakos, and Jingyi Yu. A neural rendering framework for free-viewpoint relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5599–5610, 2020.

[19] Michael F Cohen, Shenchang Eric Chen, John R Wallace, and Donald P Greenberg. A progressive refinement approach to fast radiosity image generation. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 75–84, 1988.

[20] Robert L Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics (TOG)*, 5(1):51–72, 1986.

[21] Robert L Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (ToG)*, 1(1):7–24, 1982.

[22] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[23] Kristin J Dana. Brdf/btf measurement device. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 460–466. IEEE, 2001.

[24] Paul Debevec, Paul Graham, Jay Busch, and Mark Bolas. A single-shot light probe. In *ACM SIGGRAPH 2012 Talks*, pages 1–1. 2012.

[25] Paul Debevec, Chris Tchou, Andrew Gardner, Tim Hawkins, Charis Poullis, Jessi Stumpfel, Andrew Jones, Nathaniel Yun, Per Einarsson, Therese Lundgren, et al. Estimating surface reflectance properties of a complex scene under captured natural illumination. *Conditionally Accepted to ACM Transactions on Graphics*, 19:2, 2004.

[26] Paul E Debevec, C Tchou, A Gardner, T Hawkins, C Poullis, J Stumpfel, A Jones, N Yun, P Einarsson, T Lundgren, et al. Digitizing the parthenon: Estimating surface reflectance properties of a complex scene under captured natural illumination. In *VMV*, volume 2004, page 99, 2004.

[27] Ron O Dror, Edward H Adelson, and Alan S Willsky. Estimating surface reflectance properties from images under unknown illumination. In *Human Vision and Electronic Imaging VI*, volume 4299, pages 231–242. International Society for Optics and Photonics, 2001.

[28] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090*, 2017.

[29] Dan B Goldman, Brian Curless, Aaron Hertzmann, and Steven M Seitz. Shape and spatially-varying brdfs from photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1060–1071, 2009.

[30] Henri Gouraud. Continuous shading of curved surfaces. *IEEE transactions on computers*, 100(6):623–629, 1971.

[31] Jefferson Y Han and Ken Perlin. Measuring bidirectional texture reflectance with a kaleidoscope. In *ACM SIGGRAPH 2003 Papers*, pages 741–748. 2003.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] Xiao D He, Patrick O Heynen, Richard L Phillips, Kenneth E Torrance, David H Salesin, and Donald P Greenberg. A fast and accurate light reflection model. *ACM SIGGRAPH Computer Graphics*, 26(2):253–254, 1992.

[34] Xiao D He, Kenneth E Torrance, Francois X Sillion, and Donald P Greenberg. A comprehensive physical model for light reflection. *ACM SIGGRAPH computer graphics*, 25(4):175–186, 1991.

[35] Paul S Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 145–154, 1990.

[36] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-Francois Lalonde. Deep outdoor illumination estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[37] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[38] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[39] Henrik Wann Jensen. Global illumination using photon maps. In *Eurographics workshop on Rendering techniques*, pages 21–30. Springer, 1996.

[40] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. AK Peters/CRC Press, 2001.

[41] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014.

[42] James T Kajiya. Anisotropic reflection models. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 15–21, 1985.

[43] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, August 1986.

[44] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, page 143–150, New York, NY, USA, 1986. Association for Computing Machinery.

[45] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[46] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[47] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[48] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. *ACM Transactions on Graphics (TOG)*, 30(6):1–12, 2011.

[49] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[50] Kihwan Kim, Jinwei Gu, Stephen Tyree, Pavlo Molchanov, Matthias Nießner, and Jan Kautz. A lightweight approach for on-the-fly reflectance estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 20–28, 2017.

[51] David Kirk et al. Nvidia cuda software and gpu parallel computing architecture. In *ISMM*, volume 7, pages 103–104, 2007.

[52] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[54] Eric P Lafortune and Yves D Willems. Bidirectional path tracing. In *Proceedings of the 3rd International Conference on Computational Graphics and Visualization Techniques (Compugraphics)*, pages 145–153, 1993.

[55] Eric PF Lafortune, Sing-Choong Foo, Kenneth E Torrance, and Donald P Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 117–126, 1997.

[56] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[57] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. Ieee, 2011.

[58] Robert R Lewis. Making shaders more physically plausible. In *Computer Graphics Forum*, volume 13, pages 109–120. Wiley Online Library, 1994.

[59] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[60] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. Nvidia tesla: A unified graphics and computing architecture. *IEEE micro*, 28(2):39–55, 2008.

[61] Guilin Liu, Duygu Ceylan, Ersin Yumer, Jimei Yang, and Jyh-Ming Lien. Material editing using a physically based rendering network. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[62] Stephen Lombardi and Ko Nishino. Reflectance and natural illumination from a single image. In *European Conference on Computer Vision*, pages 582–595. Springer, 2012.

[63] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[64] Rong Lu, Jan J Koenderink, and Astrid ML Kappers. Optical properties (bidirectional reflectance distribution function) of shot fabric. *Applied Optics*, 39(31):5785–5795, 2000.

[65] Stephen R Marschner and Donald P Greenberg. Inverse lighting for photography. In *Color and Imaging Conference*, volume 1997, pages 262–265. Society for Imaging Science and Technology, 1997.

[66] Abhimitra Meka, Maxim Maximov, Michael Zollhoefer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. Lime: Live intrinsic material estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6315–6324, 2018.

[67] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6878–6887, 2019.

[68] 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. *arXiv preprint arXiv:2003.08934*, 2020.

[69] Lukas Murmann, Michael Gharbi, Miika Aittala, and Fredo Durand. A dataset of multi-illumination images in the wild. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[70] Thomas Nestmeyer, Jean-Francois Lalonde, Iain Matthews, and Andreas Lehrmann. Learning physics-guided face relighting under directional light. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[71] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[72] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of brdf models. *Rendering Techniques*, 2005(16th):2, 2005.

[73] Thu H Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. In *Advances in Neural Information Processing Systems*, pages 7891–7901, 2018.

[74] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6):1–17, 2019.

[75] Michael Oren and Shree K Nayar. Generalization of lambert's reflectance model. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 239–246, 1994.

[76] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[77] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[78] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[79] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[80] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[81] Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Efstratios Gavves, and Tinne Tuytelaars. Deep reflectance maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4508–4516, 2016.

[82] Fabiano Romeiro, Yuriy Vasilyev, and Todd Zickler. Passive reflectometry. In *European Conference on Computer Vision*, pages 859–872. Springer, 2008.

[83] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[84] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[85] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[86] Steven R Schill, John R Jensen, George T Raber, and Dwayne E Porter. Temporal modeling of bidirectional reflection distribution function (brdf) in coastal vegetation. *GIScience & Remote Sensing*, 41(2):116–135, 2004.

[87] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[88] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[89] Shuran Song and Thomas Funkhouser. Neural illumination: Lighting prediction for indoor environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6918–6926, 2019.

[90] Tiancheng Sun, Jonathan T Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul E Debevec, and Ravi Ramamoorthi. Single image portrait relighting. *ACM Trans. Graph.*, 38(4):79–1, 2019.

[91] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

[92] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[93] Kenneth E Torrance and Ephraim M Sparrow. Theory for off-specular reflection from roughened surfaces. *Josa*, 57(9):1105–1114, 1967.

[94] TS Trowbridge and Karl P Reitz. Average irregularity representation of a rough surface for ray reflection. *JOSA*, 65(5):531–536, 1975.

[95] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318, 1994.

[96] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[97] Gregory J Ward. Measuring and modeling anisotropic reflection. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 265–272, 1992.

[98] Harold B Westlund and Gary W Meyer. Applying appearance standards to light reflection models. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 501–51, 2001.

[99] Turner Whitted. An improved illumination model for shaded display. In *Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, page 14, 1979.

[100] Changchang Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 127–134. IEEE, 2013.

[101] Changchang Wu et al. Visualsfm: A visual structure from motion system. 2011.

[102] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[103] Ye Yu and William A. P. Smith. Inverserendernet: Learning single image inverse rendering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[104] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 215–224, 1999.

[105] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[106] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David W. Jacobs. Deep single-image portrait relighting. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[107] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.