Towards Shape Reconstruction through Differentiable Rendering

Sai Praveen Bangaru

CMU-CS-18-129

January 24, 2019

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee
Ioannis Gkioulekas, Chair
Srinivasa Narasimhan

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

Copyright © 2018 Sai Praveen Bangaru.

Keywords: photometric stereo, differentiable rendering, differentiable path tracing, shape optimization, appearance acquisition

Abstract

Radiometric methods for the extraction of shape from images, such as photometric stereo, make simplifying assumptions about the light transport effects underlying those images. Among the most common assumptions are absence of interreflections and Lambertian reflectance. These make radiometric shape reconstruction techniques unsuitable for many classes of common objects, including objects with glossy surfaces or concave shapes. Our goal is to construct an inverse rendering framework that can be used to reconstruct shape and reflectance properties without these assumptions. Towards this goal, we develop a versatile, shape-differentiable, Monte Carlo renderer, which can efficiently estimate the differentials of image intensity values with respct to BSDF and local shape parameters. We combine this differentiable renderer with stochastic optimization and surface reconstruction algorithms, to develop a pipeline that estimates a 3D mesh that best explains captured image measurements. We evaluate this pipeline in experiments using both simulated and captured image datasets, and show that it can accurately reconstruct complex reflectance and shape even in the presence of strong global illumination. Finally, we discuss future extensions towards enabling the application of our inverse rendering framework to measurements from a large variety of 3D sensing systems.

Acknowledgments

I would like to whole-heartedly thank my advisor Ioannis Gkioulekas for his amazing mentorship over the span of a year. He has provided outstanding technical advice and a robust direction for my thesis. The results we have obtained would not have been possible without his consistent and close involvement, guiding me through the times I was stuck at some point, by holding meetings twice or even thrice a week to brainstorm and solve the problem. In addition to this, he also generously shared his experience on navigating a life in academia and pursuing a potential career in research.

I would also like to thank Anat Levin for her highly experienced insights and invaluable advice that ended up forming some of the key parts of my thesis. Furthermore, I thank Srinivasa Narasimhan, who agreed to be on my defense committee and provided valuable feedback on my thesis document, as well as improving presentation skills. I would also like to express support for his amazing graduate course on physics-based vision that served as the foundation for my interest in research.

Finally, I am thankful to my academic advisor David A. Eckhardt for his guidance through my Masters degree as well as his whole-hearted support for my decision to continue research as a career. I also thank Tracy Farbacher for her patience and kindness while scheduling my defense.

Contents

1 Introduction 1

1.1 Overview 1

1.2 Background 2

1.2.1 Monte Carlo path tracing 2

1.2.2 Photometric Stereo 3

1.2.3 Surface integration 4

1.2.4 Adaptive gradient descent 4

1.3 Related Work 5

1.3.1 Photometric Stereo 5

1.3.2 Differentiable Light Transport 6

1.4 Contributions 7

2 Differentiable Rendering 9

2.1 The path integral 9

2.2 Estimating differentials 11

2.3 Evaluating the integral 14

2.4 Implementation considerations 15

2.4.1 Spherical projection of gradients 15

2.4.2 Mesh discretization 16

3 Parametrizing the BSDF 17

3.1 Weighted average of BSDFs 17

3.2 Basis functions 18
3.3 Dictionary reduction 19
    3.3.1 Initial candidates P_0 19
    3.3.2 Pruning the candidates 20
3.4 The GGX BSDF Model 21

4 Shape Optimization **23**
4.1 Loss function 23
4.2 Set-Adaptive gradient descent 24
4.3 Exponentiated gradient descent 24
4.4 Accounting for shadows 25
4.5 Computing the depth 26
4.6 Multiresolution optimization 26
4.7 The Complete algorithm 27

5 Results **29**
5.1 Shape Reconstruction 29

6 Conclusion **37**
6.1 Future Directions 37
    6.1.1 Adaptive sampling 37
    6.1.2 Depth optimization 38
    6.1.3 Importance sampling the gradient 38
    6.1.4 Single-image BSDF acquisition 38

Bibliography **41**

List of Figures

2.1 An illustration of the components that contribute to the radiance along a specific path. Image credit: Eric Veach's PhD thesis 11

3.1 An example of the initial pool of candidate BSDFs generated by varying α and η. In practice we generate several thousand candidates in order to obtain a more efficient dictionary 20

4.1 Block diagram describing all the components of the shape optimization framework 28

5.1 A highly glossy convex mask reconstructed from a set of synthetically rendered images of a mask. The images on the left are the initialization, the images in the middle are the reconstructions obtained by refining the initialization using our method. The images on the right represent the groud truth. In this case, we use multi-resolution rendering to avoid local minima, so the initial photometric mesh is processed at half the resolution, resulting in a blurry top-left image 31

5.2 A sphere rendered under the BSDF reconstructed from the convex mask dataset (right) compared with one rendered using the original BSDF (left), which in this case is the 'gray-plastic' BSDF from the MERL database 32

5.3 A comparison of the BSDF at various input angles θ_i deduced using *(a)* our algorithm (solid lines) with *(b)* the ground truth (dashed lines) and *(c)* the direct linear fit produced using the method in Section 3 (Tabular fit). Method *(c)* represents the best fit that can be produced by our reduced dicitonary of BSDFs (dotted lines). Note that this visualization has BSDF values that are adjusted so that all lines are clearly visible, but the relative error is preserved for comparison. 32

5.4 A bear reconstructed from the DiLiGent dataset. The images on the left are the photometric reconstruction, the images in the middle are the reconstruction obtained by refining the photometric mesh using our method. The images on the right represent the groud truth 33

5.5 An inverted (concave) mask reconstructed from a set of synthetically rendered images of a mask. The images on the left are the photometric reconstruction, the images in the middle are the reconstruction obtained by refining the photometric mesh using our method. The images on the right represent the groud truth 34

5.6 A sphere rendered under the BSDF reconstructed from the concave mask dataset (right) compared with one rendered using the original BSDF (left), which in this case is the 'dark-blue-paint' BSDF from the MERL database 35

5.7 A comparison of the BSDF function at various input angles θ_i deduced using *(a)* our algorithm (solid lines) with *(b)* the ground truth (dashed lines) and *(c)* the direct linear fit produced using the method in Section 3 (Tabular fit). Method *(c)* represents the best fit that can be produced by our reduced dictionary of BSDFs (dotted lines). Note that this visualization has BSDF values that are adjusted so that all lines are clearly visible, but the relative error is preserved for comparison 35

5.8 A concave bowl reconstructed from a set of synthetically rendered images of a bowl under the 'dark-blue-paint' BSDF. The images on the left are the photometric reconstruction, the images in the middle are the reconstruction obtained by refining the photometric mesh using our method. The images on the right represent the ground truth. The two images in the bottom row show the cross-section of the bowl before and after refinement. The blue lines are the ground truth, while the orange lines are the estimates produced by our algorithm. 36

Chapter 1

Introduction

1.1 Overview

Photometric stereo, the de-facto term used for extracting shape from multiple directionally lit images, is almost as old as computer vision itself. At its core, it is fundamentally the solution to a linear system that relies on the fact that exitant radiance from a lambertian surface patch is proportional to the dot product of incident light direction and the path normal. When it came out, this algorithm was extremely powerful. For most objects, it could estimate a per-pixel normal, leading to very accurate shape reconstructions.
It does, however, make several assumptions:

1. Distant lights (Directional lighting)
2. Convex shape (No multiple bounces/interreflections)
3. Lambertian BSDF ($\mathbf{n}.\mathbf{l}$ shading)

These assumptions rarely hold true for common real world objects. For instance, the lambertian BSDF assumption breaks down for even the apparently diffuse objects, because of a grazing-angle *Fresnel* component. This is the phenomenon where seemingly rough surfaces start to behave like mirrors when viewed at an angle close to parallel with the surface.

The assumption of convexity also does not hold true for most objects, though its effects are harder to pin down accurately. Interreflections occur whenever two patches of an object face each other ($\mathbf{n_1}.\mathbf{n_2} > 0$. This means they are most pronounced near sharp inward edges. There also extremely concave surfaces like bowls and mugs that are rarely, if ever, used in photometric stereo papers because of extreme inter-reflections.

The final assumption, directional lighting, is something that our algorithm does not explicitly tackle, but can be directly extended to handle any form of lighting. Most photometric stereo methods are limited by assumption that light hits every patch at the same angle. This simplifies the optimization problem significantly since the solution is independent of the relative distance between the light and the mesh.

In practice, these assumptions can be ignored in some cases, since grazing angle reflections don't account for much of the image, and based on the shape, interreflections may only contribute near sharp edges. In this thesis, we look at objects where we cannot ignore these assumptions. This includes either specular surfaces or highly concave objects, or both.

1.2 Background

The following section contains some background information that are relevant to the components of the framework presented in this thesis.

1.2.1 Monte Carlo path tracing

At the heart of any analysis-by-synthesis problem is the *synthesis*. Since our framework aims to account for as many light transport phenomena as possible, it makes sense to use a fully-fledged physically-based rendering method, like path tracing.

Path tracing is an umbrella term for a family of Monte Carlo estimators that are used to sample paths of light through a scene. However, they all estimate the same fundamental recursive equation known as the *rendering equation*, one of the corner-stones of computer graphics.

$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_{\mathcal{H}^2} f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t) L_i(\mathbf{x}, \omega_i, \lambda, t)(\omega_i.\mathbf{n})\mathrm{d}\omega_i \tag{1.1}$$

where

1. $L_o(\mathbf{x}, \omega_o, \lambda, t)$ is the outgoing radiance at point $\mathbf{x}$ in the direction ω_o
2. λ is the wavelength (our algorithm uses luminance so this parameter is not used)
3. $L_e(\mathbf{x}, \omega_o, \lambda, t)$ is the radiance emitted at point $\mathbf{x}$ in the direction ω_o
4. $f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t)$ is the reflectance function (BSDF) at point $\mathbf{x}$ for light arriving in direction ω_i and exiting in the direction ω_o.
5. $L_i(\mathbf{x}, \omega_i, \lambda, t)$ is the incoming radiance at point $\mathbf{x}$ in the direction ω_i
6. $\mathbf{n}$ is the surface normal at point $\mathbf{x}$
7. $\mathcal{H}^2$ represents the hemisphere of valid directions at point $\mathbf{x}$

For the purpose of this thesis, we use the popular algorithm BDPT [1], which represents a sweet spot in the trade-off between complexity and sample efficiency.

However, we note that for our framework, we only require that the intensity be expressed as an integral over a set of sampled paths and their probabilities. This means we do not necessarily require BDPT. Any similar algorithm, like MLT or PSSMLT, that generates paths with their probabilities and contribution, can be substituted in its place.

1.2.2 Photometric Stereo

Classical photometric stereo ([2]) is a method that extracts surface normals $\mathbf{N}$ from a set of images of an object taken under different (known) illumination directions, but with the same view direction (the sensor and the scene are fixed). This method assumes the captured surface is lambertian (view-independent) and convex (little to no interreflections). This considerably simplifies the image formation model. The intensity at any point is simply $\max(0, \mathbf{n}^T\mathbf{l})$, which, for non-zero intensity values, is a linear model. Since we know the illumination directions $\mathbf{L}$ and the captured image intensities $\mathbf{I}$, the target normals can be computed by minimizing the linear least squares problem.

$$\mathbf{N}^* = \arg\min_{\mathbf{N}} (\mathbf{I} - \mathbf{N}^T\mathbf{L})^2 \tag{1.2}$$

1.2.3 Surface integration

Surface integration is an key part of our framework since we require a full surface to account for inter-reflections. This requires the conversion of the normals $\mathbf{N}$ at each step into consistent depths $\mathbf{Z}$. Given that our algorithm produces noisy normal estimates (stochastic gradient descent), a weighted integration algorithm like Poisson surface reconstruction [3] is a good fit for our framework.

Weighted Poisson surface reconstruction can be summarized as the linear least squares solution to a set of constraints that equates the X and Y differentials to the difference between the center pixel with the vertical and horizontal neighbors respectively. We also constrain the total depth to be 0 so that the linear system is not under-constrained.

$$\mathbf{Z} = \arg\min_{\mathbf{Z}} \sum_{i \leq W, j \leq H} \left((z_{i,j} - z_{i,j+2}) - \left(\frac{\partial z}{\partial y}\right)_{i,j+1} \right)^2 + \sum_{i \leq W, j \leq H} \left((z_{i+2,j} - z_{i,j}) - \left(\frac{\partial z}{\partial x}\right)_{i+1,j} \right)^2 \tag{1.3}$$

subject to the constraint

$$\sum_{i,j} z_{i,j} = 0$$

where

1. The differentials $\frac{\partial z}{\partial y}$ and $\frac{\partial z}{\partial x}$ are simply determined by $\frac{n_z}{n_y}$ and $\frac{n_z}{n_x}$, where n is the normal at that position.
2. $\mathbf{Z}$ is the depth matrix and $z_{i,j}$ is the depth at pixel i, j

1.2.4 Adaptive gradient descent

Gradient descent is a fundamental part of the non-linear optimization problem that is responsible for inferring the normals as well as BSDF parameters.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right)_{\boldsymbol{\theta}=\boldsymbol{\theta}_t} \tag{1.4}$$

The need for adaptive gradient descent algorithms stems from the fact that some parameters of a model can have a more drastic effect on the error than others, while also operating on the same scale (and range of values). This discrepancy introduces a catch-22 situation where an α (learning rate) that is too small will take forever to converge and an α that is too large will cause some of the parameters to diverge quickly (or behave unpredictably). This necessitates adaptive gradients, which take the form of many popular algorithms like AdaGrad, AdaDelta and Adam ([4] and [5]) has a comprehensive evaluation of the various SGD algorithms that we considered. Our algorithm uses a slightly modified version of Adam since it has provably good characteristics.

1.3 Related Work

1.3.1 Photometric Stereo

This thesis is part of the photometric stereo family of algorithms that are characterized by input images that have a fixed viewpoint and are obtained under different sets of lighting conditions. Most photometric stereo algorithms work by computing the normal map rather than the depth itself, including the algorithm presented in this thesis. The various classes of photometric stereo can be classified based on their assumptions.

1. *Convex and lambertian surfaces*: Methods such as classical photometric stereo [2] and near-light photometric stereo [6] assume that the surface is lambertian ($\mathbf{n}.\mathbf{l}$ lighting), as well as convex (lack of interreflections). Our method makes neither of these fundamental assumptions.
2. *Non-convex and lambertian surfaces*: Shape from interreflections [7] is specifically designed to work with surfaces that are non-convex using inverse global illumination to account for interreflections. However, these methods use linear models for global illumination like radiosity, which all assume lambertian surface properties. Our method is not only designed to overcome this limitation, it also estimates the surface BSDF properties in the process.

3. *Convex and non-lambertian surfaces*: Goldman et al.([8]) was one of the first methods to estimate BSDF and normals in tandem, removing the need for reference objects. Alldrin et al. ([9]) proposes a non-parametric model that builds a dictionary of BSDFs and estimates a spatially varying set of weights. Since our method is focused on extracting shape rather than BSDF, our method models the surface reflectance as the linear combination of a predefined dictionary of BSDFs, rather than estimating every input/output direction pair. Our method also accounts for inter-reflections (non-convexity) in the presence of unknown surface reflectance, making our error function highly non-linear and requiring a more involved approach to estimate the surface.

Given the emphasis placed on accounting for inter-reflections, this thesis is also related closely to methods that eliminate global light transport from the measurements, like direct-global separation [10], which eliminates interreflections by projecting checkerboard light patterns. Other examples of this class of algorithms include epipolar scanning [11] and continuous wave time-of-flight cameras.

1.3.2 Differentiable Light Transport

On the differentiable rendering side of things, there are a variety of frameworks built for specific applications. Differentiable rasterizers (like OpenDR [12] and Neural 3D Mesh Renderer [13]) are popular new tools in inferring transformation matrices of scene objects. However, rasterizers make no attempt to simulate global illumination (or even realistic reflectance), and would be unsuitable for shape optimization. The same goes for single bounce differentiable renderers ([14] and [15]), since they do not account for interreflections. One notable framework is the inverse scattering algorithm described in Gkioulekas et al. ([16] and [17]), which uses the same differential formulation of the path integral for computing multi-bounce differentials. That formulation uses transient instead of photometric imaging, and optimizes for heterogeneous volumetric properties rather than surface properties. The framework introduced by Li et al. [18] is the first fully-differentiable path tracer that can theoretically optimize for shape in the presence of interreflections. However, their work considers a very limited number of scene parameters (various transformation matrices and 3 BSDF parameters), and does not handle complex light paths, like caustics (no path space sampling). In contrast, while our algorithm does not handle derivatives w.r.t occlusions, it does handle differentiating per-pixel normals (about a million shape parameters) to extract precise shape, which makes it a better fit for mesh refinement in a controlled setting like photometric stereo. In addition, our algorithm is based on the path space formulation of the derivatives, and uses BDPT, which allows for the sampling of complex paths.

1.4 Contributions

This thesis contributes to the field of physics-based computer vision by

1. Formulating and implementing a theoretically sound Monte Carlo estimator for the derivatives of a physically-based renderer with respect to shape. This is a general-purpose renderer that can be extended to apply to most active shape estimation algorithms.
2. Applying the differentiable renderer to solve the difficult problem of simultaneously estimating both the shape and reflectance of objects that exhibit non-trivial global light transport (inter-reflections) as well as non-lambertian (view-dependent) surface reflectance, in the context of photometric stereo.

Chapter 2

Differentiable Rendering

This chapter first defines the popularly used path integral formulation of light transport. From there, we derive the equations that represent the derivative of this path integral w.r.t shape and BSDF parameters. Further, we briefly describe an augmented BDPT algorithm to efficiently estimate this integral, and explain how to deal with some practical issues that arise during implementation.

2.1 The path integral

The path integral is a common and versatile way of expressing the general rendering equation [19] shown in Chapter 1. Section 8.2 of [20] contains the complete derivation of the path integral, but we reiterate some of the important parts since the path integral provides the foundation for the derivative estimator.

We define a path space $\mathcal{P}_k$ as the set of all paths of length k, which are of the form

$$\bar{x} = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3 \ldots \mathbf{x}_k \tag{2.1}$$

Thus the complete path space $\mathcal{P}$ can be written as

$$\mathcal{P} = \bigcup_{1}^{\infty} \mathcal{P}_k \tag{2.2}$$

For the path integral formulation, we need to convert the solid angle measure $\boldsymbol{\omega}$ that is used in the standard rendering equation to the area-product measure μ_k defined as

$$\mu_k = \prod_{i=0}^{k} A \tag{2.3}$$

where the measure A denotes the area of all valid surfaces that contain points $\mathbf{x}_i$ that contribute to the integrand. The measure A is repeated for each point $\mathbf{x}_i$ since our integral contains one surface integral for each point on the path.

Our differential measure $\mathbf{d}\mu(\bar{x})$ can be expressed as the product of differential area measures

$$\mathbf{d}\mu_k(\bar{x}) = \prod_{i=0}^{k} \mathbf{d}A(x_i) \tag{2.4}$$

To derive the path integral, we start with the three-point integral that is very similar to the rendering equation, but is defined on the area measure instead of the solid angle measure. The radiance from point $\mathbf{x}_1$ to $\mathbf{x}_2$ is expressed as

$$L_o(\mathbf{x}_1, \mathbf{x}_2) = L_e(\mathbf{x}_1, \mathbf{x}_2) + \int_{\mathcal{M}} f_r(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2) L_i(\mathbf{x}_0, \mathbf{x}_1) G(\mathbf{x}_0, \mathbf{x}_1) dA(\mathbf{x}) \tag{2.5}$$

where

1. $L_e(\mathbf{a}, \mathbf{b})$ is the emitted radiance in the direction $\mathbf{a} \rightarrow \mathbf{b})$
2. $f_r(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is the reflectance for the incident direction $\mathbf{a} \rightarrow \mathbf{b}$ and exitant direction $\mathbf{c} \rightarrow \mathbf{b}$
3. $L_i(\mathbf{a}, \mathbf{b})$ represents the incident radiance in the direction $\mathbf{a} \rightarrow \mathbf{b}$ (this is the recursive part)
4. $G(\mathbf{a}, \mathbf{b})$ is the geometric function $\frac{(\hat{\boldsymbol{l}}.\hat{\boldsymbol{n}})(\mathbf{v}.\hat{\boldsymbol{n}})}{||x_i - x_{i+1}||^2}$, which accounts for the cosine angles between the two surfaces as well as the distance between points $\mathbf{a}$ and $\mathbf{b}$.

Expanding this function to account for paths of size k, we can express the radiance along size-k paths L_k as

$$\begin{aligned} L_k = L_o(\mathbf{x}_{k-1}, \mathbf{x}_k) = \int_{\mathcal{M}^k} & L_e(\mathbf{x}_0, \mathbf{x}_1)(G(\mathbf{x}_0, \mathbf{x}_1) f_r(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2))(G(\mathbf{x}_1, \mathbf{x}_2) f_r(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)) \\ & (G(\mathbf{x}_1, \mathbf{x}_2) f_r(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)) \ldots (G(\mathbf{x}_{k-2}, \mathbf{x}_{k-1}) \\ & f_r(\mathbf{x}_{k-2}, \mathbf{x}_{k-1}, \mathbf{x}_k)) dA(\mathbf{x}_0) dA(\mathbf{x}_1) dA(\mathbf{x}_2) \ldots dA(\mathbf{x}_k) \end{aligned} \tag{2.6}$$

which can be simplified as

$$L_k = \int_{\mathcal{M}^k} L_e(\mathbf{x}_0, \mathbf{x}_1) \Big(\prod_i G(\mathbf{x}_i, \mathbf{x}_{i+1}) f_r(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}) \Big) d\mu(\bar{x}) \tag{2.7}$$

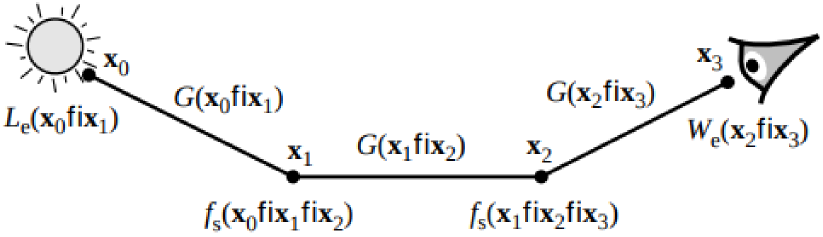

Figure 2.1: An illustration of the components that contribute to the radiance along a specific path. Image credit: Eric Veach's PhD thesis

which is the path integral equation for paths of length k

The general path equation can be expressed as the sum of the contribution for each path size k

$$L = \sum_k \int_{\mathcal{M}^k} L_e(\mathbf{x}_0, \mathbf{x}_1) \Big(\prod_i G(\mathbf{x}_i, \mathbf{x}_{i+1}) f_r(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}) \Big) d\mu(\bar{x}) \tag{2.8}$$

More concisely, if we consider the set $\mathcal{P}$ to be the space of paths of all sizes, then

$$L = \int_{\mathcal{P}} f_p(\bar{x}) d\mu(\bar{x}) \tag{2.9}$$

where the integrand is the path contribution function $f_p(\bar{x})$

$$f_p(\bar{x}) = L_e(\mathbf{x}_0, \mathbf{x}_1) \Big(\prod_i G(\mathbf{x}_i, \mathbf{x}_{i+1}) f_r(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}) \Big) \tag{2.10}$$

2.2 Estimating differentials

As shown in Section 2.1, the path integral formulation of the rendering equation can be written as the sum of products of BSDF and the geometric term at each point on the path.

$$I = \int_{\mathcal{P}} S(x_0, x_1) \prod_i G(x_i, x_{i+1}).f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta}) \mathrm{d}\mu(\bar{x}) \tag{2.11}$$

To simplify our explanation, the geometric term $G(x_i, x_{i+1})$ can be replaced with two terms $\frac{(\hat{\boldsymbol{l}}.\hat{\boldsymbol{n}}_i)(\mathbf{v}.\hat{\boldsymbol{n}}_{i+1})}{||x_i - x_{i+1}||^2} V(x_i, x_{i+1})$ where $V(x_i, x_{i+1})$ is the visibility term that takes the value 1 if the ray from x_i to x_{i+1} has no occlusion, and 0 otherwise.

$$V(x_i, x_{i+1}) = \begin{cases} 0 : \text{ray from } x_i \text{ to } x_{i+1} \text{ is occluded} \\ 1 : \text{otherwise} \end{cases}$$

$$I = \int_{\mathcal{P}} S(x_0, x_1) \prod_i \Big(\frac{(\hat{\boldsymbol{l}}_i.\hat{\boldsymbol{n}}_{i+1})(\hat{\boldsymbol{v}}_i.\hat{\boldsymbol{n}}_i)}{||x_i - x_{i+1}||^2} V(x_i, x_{i+1}).f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta}) \Big) \mathrm{d}\mu(\bar{x}) \tag{2.12}$$

We can express the Monte-Carlo estimator for the above integral as a summation over N importance sampled paths p from some arbitrary distribution $\Pi(\mathcal{P})$ over the complete path space $\mathcal{P}$

$$\hat{I} = \frac{1}{N} \sum_{\bar{x} \sim \Pi(\mathcal{P})}^{N} S(x_0, x_1) \frac{1}{\Pi(\bar{x})} \prod_i \Big(\frac{(\hat{\boldsymbol{l}}_i.\hat{\boldsymbol{n}}_i)(\hat{\boldsymbol{v}}_i.\hat{\boldsymbol{n}}_{i+1})}{||x_i - x_{i+1}||^2} V(x_i, x_{i+1}).f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta}) \Big) \tag{2.13}$$

$S(x_0^{(p)}, x_1^{(p)})$ represents the illumination in the direction from $x_0^{(p)}$ and $x_1^{(p)}$ where p denotes a path from a set of randomly sampled paths $\mathcal{P}$ and $\Pi(p)$ represents the probability of sampling path p.

For clarity we define the variables $\hat{\boldsymbol{l}}_i$ and $\mathbf{v}_i$

$$\hat{\boldsymbol{l}}_i = \mathcal{R}(\hat{\boldsymbol{n}}_i)(\widehat{x_i - x_{i-1}}) \tag{2.14}$$

$$\hat{\boldsymbol{v}}_i = \mathcal{R}(\hat{\boldsymbol{n}}_i)(\widehat{x_i - x_{i+1}}) \tag{2.15}$$

where $\mathcal{R}(\hat{\boldsymbol{n}}_i)$ is the rotation matrix that converts world coordinates to local coordinates (where the normal is parallel to the z-axis). This rotation is dependent on the local normal $\hat{\boldsymbol{n}}_i$

The goal of this framework is to estimate the differentials $\frac{\partial I}{\partial \theta}$, $\frac{\partial I}{\partial \hat{\boldsymbol{n}}}$. To achieve this, we differentiate the path integral,

$$\frac{\partial I}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \int_{\mathcal{P}} S(x_0, x_1) \prod_i \Big(\frac{(\hat{\boldsymbol{l}}_i.\hat{\boldsymbol{n}}_i)(\hat{\boldsymbol{v}}.\hat{\boldsymbol{n}}_{i+1})}{||x_i - x_{i+1}||^2} V(x_i, x_{i+1}).f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta}) \Big) \mathrm{d}\mu(\bar{x}) \tag{2.16}$$

An important property of this integral is that it is *converging* over its domain. This is because this integral is equivalent to using the global illumination operators ($\mathbf{K}$, $\mathbf{G}$ and $\mathbf{M^{-1}}$) defined in James Arvo's thesis [21]. The same thesis contains a proof that the linear operator formulation of the integral converges for all values in its domain. In measure space theory, the derivative of a well-behaved (converging) integral over some measure ($\mu(.)$) can be written as the integral of the derivative (a generalization of Leibnitz integral rule) [22]. In the case of the path integral, the measure space of the integral is the space of all paths $\mathcal{P}$, which is independent of both $\boldsymbol{\theta}$ or $\hat{\boldsymbol{n}}$.

$$\frac{\partial I}{\partial \boldsymbol{\theta}} = \int_{\mathcal{P}} \frac{\partial}{\partial \boldsymbol{\theta}} S(x_0, x_1) \prod_i \Big(\frac{(\hat{\boldsymbol{l}}_i.\hat{\boldsymbol{n}}_i)(\hat{\boldsymbol{v}}_i.\hat{\boldsymbol{n}}_{i+1})}{||x_i - x_{i+1}||^2} V(x_i, x_{i+1}).f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta}) \Big) \mathrm{d}\mu(\bar{x}) \tag{2.17}$$

In a method similar to equation 2.13, we formulate a Monte-Carlo estimator using the sum of samples drawn from an arbitrary distribution $\Pi(.)$.

$$\frac{\partial I}{\partial \boldsymbol{\theta}} = \frac{1}{N} \sum_{\bar{x} \sim \Pi(.)}^{N} \frac{1}{\Pi(\bar{x})} \frac{\partial}{\partial \boldsymbol{\theta}} \Big(S(x_0, x_1) \prod_i \frac{(\hat{\boldsymbol{l}}_i.\hat{\boldsymbol{n}}_i)(\hat{\mathbf{v}}_i.\hat{\boldsymbol{n}}_{i+1})}{||x_i - x_{i+1}||^2} V(x_i, x_{i+1}).f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta}) \Big) \tag{2.18}$$

Since both source lighting $S(.)$ and visibility $V(.)$ are independent of both $\boldsymbol{\theta}$ and $\hat{\boldsymbol{n}}$, we can use the product rule to write the following equation

$$\frac{\hat{\partial} I}{\partial \boldsymbol{\theta}} = \frac{1}{N} \sum_{\bar{x} \sim \Pi(.)}^{N} S(x_0, x_1) \frac{1}{\Pi(\bar{x})} \Big(\sum_i \frac{f'_{\boldsymbol{\theta}}(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})}{f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})} \Big) \prod_i V(x_i, x_{i+1}) \frac{(\hat{\boldsymbol{l}}_i.\hat{\boldsymbol{n}}_i)(\hat{\boldsymbol{v}}_i.\hat{\boldsymbol{n}}_{i+1})}{||x_i - x_{i+1}||^2} f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta}) \tag{2.19}$$

where

$$f'_{\boldsymbol{\theta}}(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta}) = \frac{\partial f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

Most of the terms can be replaced by the path contribution function $f_p(\bar{x})$ (See equation 2.10). This gives us a simplified form.

$$\frac{\hat{\partial} I}{\partial \boldsymbol{\theta}} = \frac{1}{N} \sum_{\bar{x} \sim \Pi(.)}^{N} f_p(\bar{x}) \Big(\sum_i \frac{f'_{\boldsymbol{\theta}}(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})}{f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})} \Big) \tag{2.20}$$

The same process can be repeated to produce the analytic differential for a normal $\mathbf{n}_j$. Note that the surface contains many independent normals and $\frac{\partial \hat{\boldsymbol{n}}_i}{\partial \hat{\boldsymbol{n}}_j} = \delta_{ij}$, so differentiating with respect to a normal n_j that is never touched by a surface defined by n_i will cause the derivative to be 0.

$$\frac{\hat{\partial} I}{\partial \hat{\boldsymbol{n}}_j} = \frac{1}{N} \sum_{\bar{x} \sim \Pi(.)}^{N} f_p(\bar{x}) \Big(\sum_i \frac{f'_{\mathbf{n}_j}(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})}{(\hat{\boldsymbol{l}}_i.\hat{\boldsymbol{n}}_i)(\hat{\boldsymbol{v}}_i.\hat{\boldsymbol{n}}_{i+1}) f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})} \Big) \tag{2.21}$$

where

$$f'_{\hat{\boldsymbol{n}}_j}(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \hat{\boldsymbol{n}}_j) = \frac{\partial (\hat{\boldsymbol{l}}_i.\hat{\boldsymbol{n}}_i)(\hat{\boldsymbol{v}}_i.\hat{\boldsymbol{n}}_{i+1}) f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \hat{\boldsymbol{n}}_{i+1})}{\partial \hat{\boldsymbol{n}}_j} \tag{2.22}$$

For brevity, we redefine the additional term $\left(\sum_i \frac{f'_{\boldsymbol{\theta}}(\hat{\boldsymbol{l}}_i,\hat{\boldsymbol{v}}_i;\boldsymbol{\theta})}{f_r(\hat{\boldsymbol{l}}_i,\hat{\boldsymbol{v}}_i;\boldsymbol{\theta})}\right)$ as the *score throughput* $\mathcal{D}(\bar{x})$ Thus, our final equations are

$$\mathcal{D}_{\boldsymbol{\theta}}(\bar{x}) = \Big(\sum_i \frac{f'_{\boldsymbol{\theta}}(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})}{f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})}\Big) \tag{2.23}$$

$$\frac{\partial \hat{I}}{\partial \boldsymbol{\theta}} = \frac{1}{N} \sum_{\bar{x} \sim \Pi(.)}^{N} f_p(\bar{x}) \mathcal{D}_{\boldsymbol{\theta}}(\bar{x}) \tag{2.24}$$

The same score throughput can be defined for a normal $\hat{\boldsymbol{n}}_j$ as

$$\mathcal{D}_{\hat{\boldsymbol{n}}_j}(\bar{x}) = \sum_i \Big(\frac{f'_{\hat{\boldsymbol{n}}_j}(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})}{(\hat{\boldsymbol{l}}_i.\hat{\boldsymbol{n}}_i)(\hat{\boldsymbol{v}}_i.\hat{\boldsymbol{n}}_i) f_r(\hat{\boldsymbol{l}}_i, \hat{\boldsymbol{v}}_i; \boldsymbol{\theta})}\Big) \tag{2.25}$$

Note that our estimator works with any arbitrary path space probability distribution $\Pi(.)$. Any path tracing algorithm can be applied as long as the score throughput $\mathcal{D}(\bar{x})$ can be computed efficiently along with the intensity estimate $\hat{I}$

2.3 Evaluating the integral

Our algorithm uses Bi-directional path tracing (BDPT) [1] to generate paths $\bar{x}$, their importances $\Pi(\bar{x})$, the radiance-along-path $L(\bar{x})$, as well as the score throughput functions $\mathcal{D}_{\boldsymbol{\theta}}(\bar{x})$ and $\mathcal{D}_{\hat{\boldsymbol{n}}_j}(\bar{x})$ in a single pass.

We summarize the BDPT algorithm's contribution computation step as well our modifications to compute the differentials. The complete algorithm is contained in Lafortune's paper [1], and the algorithm is used as is, except for the contribution step.

BDPT works by first sampling two paths, one from a light source (known as light subpath $\mathbf{y}_0\mathbf{y}_1 \ldots \mathbf{y}_{n_L-1}$) and one from the camera (known as eye subpath $\mathbf{z}_{n_E-1} \ldots \mathbf{z}_0$).

The algorithm then generates paths by combining the first s vertices of the light subpath and the last t vertices of the eye subpath to create a path $\hat{x}_{s,t}$. Once we have a set of paths, the next steps are to compute the path probability $\Pi(\hat{x}_{s,t})$ and the contribution $C_{s,t}$. We will not discuss the computation of path probability as it is exactly the same.

The contribution to $C_{s,t}$ is computed as the product of a sampling weight for s, t paths and the unweighted contribution $C_{s,t} = w_{s,t}C^*_{s,t}$ The unweighted contribution is computed as the product of three components

$$C^*_{s,t} = \alpha^L_s.c_{s,t}.\alpha^E_t$$

where α^L_s depends only on the light path vertices, $c_{s,t}$ depends only on the connecting path and α^E_t depends only on the eye path

At this point, we can compute the score throughput $\mathcal{D}_{\boldsymbol{\theta}}(\bar{x}_{s,t})$ for the path $\bar{x}_{s,t}$ w.r.t $\boldsymbol{\theta}$, in a manner similar to the section above

$$\begin{aligned}\mathcal{D}_{\boldsymbol{\theta}}(\bar{x}_{s,t}) = \sum_{i=0}^{s}\Big(\frac{f'_{\boldsymbol{\theta}}(\mathbf{y}_{i-3},\mathbf{y}_{i-2},\mathbf{y}_{i-1};\boldsymbol{\theta})}{f(\mathbf{y}_{i-3},\mathbf{y}_{i-2},\mathbf{y}_{i-1};\boldsymbol{\theta})}\Big) + \sum_{j=0}^{t}\Big(\frac{f'_{\boldsymbol{\theta}}(\mathbf{z}_{i-1},\mathbf{z}_{i-2},\mathbf{z}_{i-3};\boldsymbol{\theta})}{f(\mathbf{z}_{i-1},\mathbf{z}_{i-2},\mathbf{z}_{i-3};\boldsymbol{\theta})}\Big) \\ + \frac{f'_{\boldsymbol{\theta}}(\mathbf{y}_{s-2},\mathbf{y}_{s-1},\mathbf{z}_{t-1};\boldsymbol{\theta})}{f(\mathbf{y}_{s-2},\mathbf{y}_{s-1},\mathbf{z}_{t-1};\boldsymbol{\theta})} + \frac{f'_{\boldsymbol{\theta}}(\mathbf{y}_{s-1},\mathbf{z}_{t-1},\mathbf{z}_{t-2};\boldsymbol{\theta})}{f(\mathbf{y}_{s-1},\mathbf{z}_{t-1},\mathbf{z}_{t-2};\boldsymbol{\theta})}\end{aligned} \tag{2.26}$$

$$\frac{\partial C^*_{s,t}}{\partial \boldsymbol{\theta}} = \alpha^L_s.c_{s,t}.\alpha^E_t.\mathcal{D}_{\boldsymbol{\theta}}(\bar{x}_{s,t}) \tag{2.27}$$

Our version of BDPT uses this modified contribution function and forms the final image by summing the quantity $\frac{\partial C^*_{s,t}}{\partial \boldsymbol{\theta}}$ over various s-t paths $\bar{x}_{s,t}$. The same process is used to obtain $\mathcal{D}_{\hat{\boldsymbol{n}}_j}(\bar{x})$, only with $f_{\hat{\boldsymbol{n}}}(.)$ (See equation 2.22)

2.4 Implementation considerations

In this section, we discuss some of the practical considerations of implementing the estimator defined above.

2.4.1 Spherical projection of gradients

The normal $\hat{\boldsymbol{n}}$ has three components n_x, n_y, n_z. So the differential $\frac{\partial \hat{I}}{\partial \hat{\boldsymbol{n}}}$ is actually three components $\frac{\partial \hat{I}}{\partial n_x}, \frac{\partial \hat{I}}{\partial n_y}, \frac{\partial \hat{I}}{\partial n_z}$. These components are not independent of each other, because of the constraint $n_x^2 + n_y^2 + n_z^2 = 1$. One way to handle this situation is projected gradient descent which has two steps

1. At iteration t, $\hat{\boldsymbol{n}}_{t+1} = \tilde{\mathbf{n}}_t + \alpha\frac{\partial \hat{I}}{\partial \hat{\boldsymbol{n}}}$

2. Project the new normal back onto the unit sphere by normalizing it $\tilde{\mathbf{n}}_{t+1} = \frac{\hat{\boldsymbol{n}}_{t+1}}{|\hat{\boldsymbol{n}}_{t+1}|}$

While this process is technically correct, it can be very slow to converge because the progress made in step 1 can be partially reset in step 2. A more efficient way would be to pre-project the gradients on to the unit sphere using the vector triple product operator.

$$\left(\frac{\partial \hat{I}}{\partial \hat{\boldsymbol{n}}}\right)_{tangential} = \tilde{\mathbf{n}}_{\mathbf{t}} \times (\tilde{\mathbf{n}}_{\mathbf{t}} \times \frac{\partial \hat{I}}{\partial \hat{\boldsymbol{n}}}) \tag{2.28}$$

The resulting projected gradient is tangential to the unit sphere.

Note that step 2 (normalization) is still necessary to maintain correctness, but the progress is much faster and well defined.

2.4.2 Mesh discretization

The theory laid out so far assumes that every point $\mathbf{x}_i$ on the path has its own unique normal. In practice, we cannot define a new normal for every possible point on a surface, and we rely on discretizing the surface with a *triangular manifold mesh*. However, while the surface (and normals) are present at discrete points, the points that are sampled are not discrete, which means that the normal used by a point has to be interpolated from its nearest discrete normals. Our algorithm uses linear interpolation that uses barycentric coordinates α, β and γ of the point $\mathbf{x}_i$ w.r.t to the triangle vertices a_i, b_i, c_i and corresponding normals n_{a_i}, n_{b_i} and n_{c_i}.

$$\mathbf{n}_i = \alpha \mathbf{n}_{a_i} + \beta \mathbf{n}_{b_i} + \gamma \mathbf{n}_{c_i}$$

Once we compute differentials w.r.t interpolated normal $\mathbf{n}_i$, we need to propagate the gradients back to their discrete counterparts by multiplying the corresponding Jacobian

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}_{a_i}} = \frac{\partial \mathcal{L}}{\partial \mathbf{n}_i} \alpha$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}_{b_i}} = \frac{\partial \mathcal{L}}{\partial \mathbf{n}_i} \beta$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{n}_{c_i}} = \frac{\partial \mathcal{L}}{\partial \mathbf{n}_i} \gamma$$

Chapter 3

Parametrizing the BSDF

In Chapter 2, we detailed a method for obtaining differentials w.r.t arbitrary BSDF parameters. However, the BSDF is a 4D function that obviously has too many free variables if we were to attempt to estimate every combination of $\theta_i, \phi_i, \theta_o$ and ϕ_o. In this chapter, we present the assumptions our algorithm makes about the BSDF, in order to create a system with fewer parameters, which is also reasonably good at estimating real-world BRDFs. Further, we show the procedure used to create this efficient dictionary of basis BSDFs.

3.1 Weighted average of BSDFs

As is fairly common, we restrict the algorithm to use isotropic BRDFs, since most real world materials are isotropic, with the notable exception of brushed metals. This immediately reduces the BSDF to a 3D function. However, given that our optimization problem is already the complex interplay between thousands of normals, the depth *and* the BSDF, our model tries to further cut down on the number of paramters by representing the BRDF as a linear combination of B basis functions.

$$f_s(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}; \boldsymbol{\theta}) = \sum_{\theta_i \in \boldsymbol{\theta}} \theta_i f_i(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$$

In order to satisfy fundamental BSDF properties, each $f_i(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ is also a valid BSDF (satisfies *reciprocity*, *energy conservation* and *non-negativity*). In addition to this, the following conditions are required for the compound BSDF to be valid.

$$\forall \theta \in \boldsymbol{\theta}, \theta \geq 0$$

$$\sum_{\theta \in \boldsymbol{\theta}} \theta \leq 1$$

For the purposes of this thesis, one of our basis functions is a completely dark BSDF f_b, which satisfies,

$$f_b(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = 0 \; \forall \boldsymbol{\omega_i}, \boldsymbol{\omega_o}$$

This allows us to modify the last constraint into,

$$\sum_{\theta \in \boldsymbol{\theta}} \theta = 1$$

While this appears to be a fairly trivial modification, this form makes it possible to use *adaptive exponentiated gradient descent*, a multiplicative form of the gradient descent operator that makes optimization much more feasible when dealing with constrained parameters.

3.2 Basis functions

The BSDF parameterization described in the previous section did not elaborate on the actual functions used $f_i(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$, just that they must be valid BSDFs. In this section, we propose a set of basis functions. The bases are all instances of the GGX family of microfacet BRDFs. Microfacet theory is a popular BSDF model that is based on the assumption that every patch contains a set of miniature 'facets' (microfacets), each of which is oriented at a random direction that is sampled from a normal distribution function. A GGX BSDF can be written in the following form:

$$f(\mathbf{l}, \mathbf{v}) = \frac{D(\mathbf{h})F(\mathbf{v}, \mathbf{h})G(\mathbf{l}, \mathbf{v}, \mathbf{h})}{4(\mathbf{n}.\mathbf{l})(\mathbf{n}.\mathbf{v})}$$

The GGX BSDF has two major parameters that determine its reflectance properties, the roughness α and the Fresnel index of refraction η. Figure 3.1 shows the variation of reflectance with α and η. A pool of such BSDFs is created by sampling αs and ηs in discrete steps $\alpha \in [0.01, 0.46], \eta \in [1.05, 1.95]$. We then select a fixed set of these BSDFs by finding the smallest set that can explain a real-world BSDF dataset, MERL. The elements with non-zero coefficients form the dictionary that will be used as the basis for our optimization problem.

1. $D(\mathbf{h})$ is the normal distribution function evaluated at the half vector $\mathbf{h} = \frac{\mathbf{l}+\mathbf{v}}{|\mathbf{l}+\mathbf{v}|}$.
2. $F(\mathbf{v}, \mathbf{h})$ is the Fresnel function that depends only on the exitant direction $\mathbf{v}$
3. $G(\mathbf{l}, \mathbf{v}, \mathbf{h})$ is the geometric visibility function that is dependent on both exitant and incoming directions ($\mathbf{v}$ and $\mathbf{l}$)

Note that all $\mathbf{l}$, $\mathbf{v}$, $\mathbf{h}$ are vectors represented in the space of the patch normal $\mathbf{n}$ (In this space, $\mathbf{n}$ is exactly equal to the z-axis). Thus they are dependent on the normal and not free vectors. For clarity, we denote the free vectors as $\tilde{\mathbf{l}}$, $\tilde{\mathbf{v}}$, $\tilde{\mathbf{h}}$ and the relationship between the two forms is as follows:

$$\mathbf{l} = \mathcal{R}(\mathbf{n})\tilde{\mathbf{l}}, \mathbf{h} = \mathcal{R}(\mathbf{n})\tilde{\mathbf{h}}, \mathbf{v} = \mathcal{R}(\mathbf{n})\tilde{\mathbf{v}}$$

where $\mathcal{R}(\mathbf{n})$ is a rotation matrix to convert from global to local space.

Note that the BSDF must be differentiable w.r.t normals $\mathbf{n}$ since we need $\frac{\partial f_s(\mathbf{l},\mathbf{v},\mathbf{h})}{\partial \mathbf{n}}$ in order to evaluate the differentials outlined in Chapter 2.

3.3 Dictionary reduction

In summary, the dictionary reduction procedure first instantiates a large number of candidate BSDFs from the GGX family (by randomly sampling different α and η values). To select the best candidates, we solve a quadratic programming problem to derive a linear combination of the candidate BSDFs that best fits a real world BSDF (in this case, the MERL database [23]). The lowest weights are removed and the process is repeated till we have a reasonably small number of BSDFs in the dictionary. For the results in this thesis, we use 40 dictionary elements.

3.3.1 Initial candidates P_0

Since our dictionary is selected from the GGX family of BSDFs, the next step is to sample a reasonably small dictionary of BSDFs that can best express real-world BSDFs. In order to select

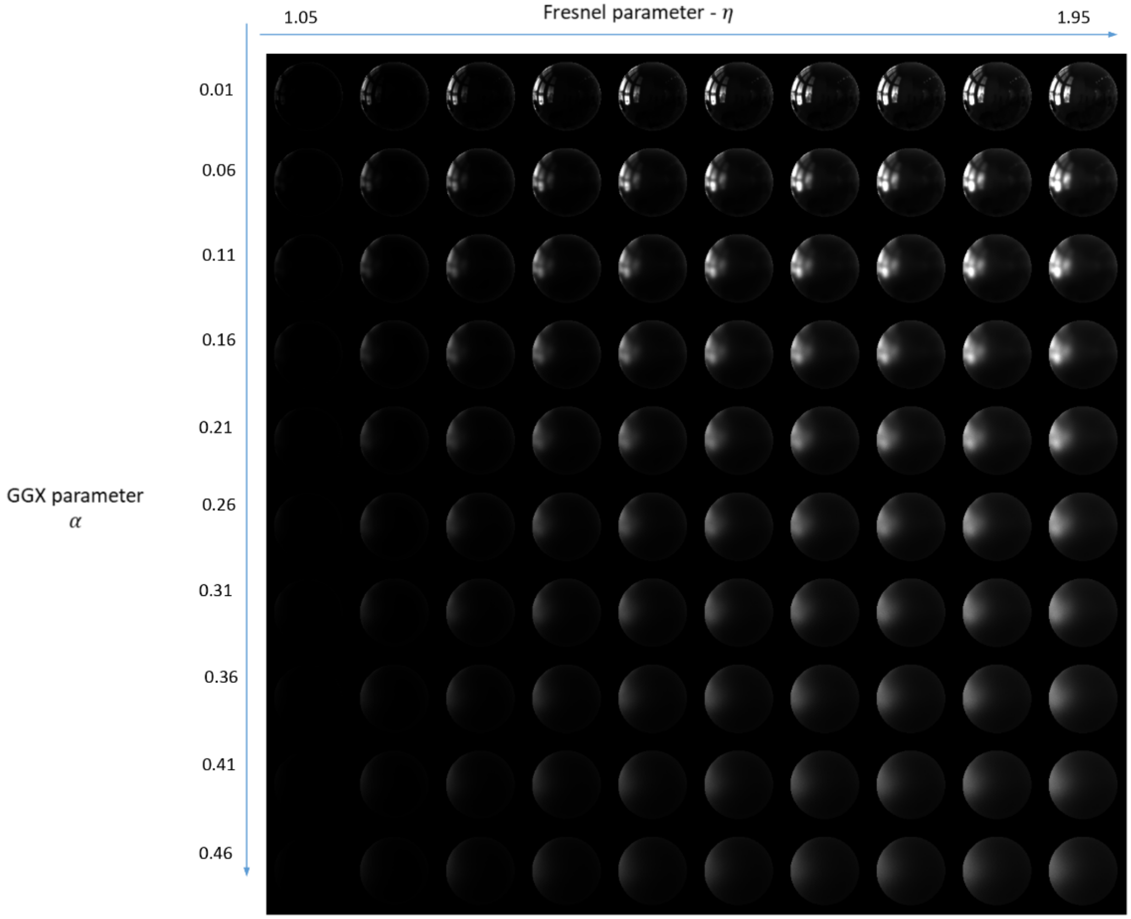

Figure 3.1: An example of the initial pool of candidate BSDFs generated by varying α and η. In practice we generate several thousand candidates in order to obtain a more efficient dictionary

an optimal set of BSDFs, we try to first start with a large pool P of possible candidates. We do this by uniformly sampling the microfacet parameters: roughness α and the refractive index η.

3.3.2 Pruning the candidates

To prune the large candidate pool, we solve a minimization problem to find a linear combination $\mathbf{a}$ of the BSDFs in the current set P_k (at iteration k) that best fits the MERL [23] database of measured real-world BSDFs. To formulate a cost function for this minimization problem, a 'target' is required, which in this case is the MERL [23] database, which provides several non-parametric real-world BSDFs represented using the reflectance at a collection of discrete

$(\theta_o, \phi_o, \theta_i)$ samples.

$$\mathbf{a}_{total} = \sum_i \arg\min_{\mathbf{a}} (\mathbf{Y_i} - \mathbf{D}^T\mathbf{a})^T \mathbf{W} (\mathbf{Y_i} - \mathbf{D}^T\mathbf{a}) \tag{3.1}$$

where

1. Y_i represents the elements of the i^{th} BSDF in the MERL database.
2. D is the matrix of elements formed by the current dictionary P_k
3. W is the diagonal weight matrix formed by the product of two weight matrices W_j and W_t, where W_j is the Jacobian that results from non-uniform sampling on a sphere [24], and $W_t = \text{diag}(\frac{1}{Y+\epsilon})$, which is used to minimize relative error rather than absolute error for better results.

$\mathbf{a}_{total}$ represents the linear fit coefficient of each candidate BSDF averaged over all MERL target BDSFs. This value represents the usefulness of the candidate BSDF, and the ones with the lowest BSDFs are removed from consideration to form a new smaller active set P_{k+1} from P_k

The process described above is repeated until the active set is small enough for efficient rendering.

3.4 The GGX BSDF Model

In this section we present the GGX BSDF specification, as well as its analytical derivatives. The differentiability of the BSDF with respect to the normal is an important precondition for the feasibility of our algorithm.

As described in Section 3.2, microfacet BSDFs are usually made up of 3 functions, which we describe for the GGX family here:

1. The GGX (Trowbridge-Reitz) [25] normal distribution function $D(\mathbf{h})$

$$D(\mathbf{h}; \mathbf{n}) = \frac{\alpha^2}{\pi((\mathbf{n}.\mathbf{h})^2(\alpha^2 - 1) + 1)^2}$$

2. The Cook-Torrance Fresnel function [26] $F(\mathbf{v}, \mathbf{h})$

$$\eta = \frac{1 + \sqrt{F_0}}{1 - \sqrt{F_0}}$$

$$c = \mathbf{v}.\mathbf{h}$$

$$g = \sqrt{\eta^2 + c^2 - 1}$$

$$F(\mathbf{v}, \mathbf{h}; \mathbf{n}) = \frac{1}{2}\Big(\frac{g-c}{g+c}\Big)^2\left(1 + \Big(\frac{(g+c)c-1}{(g-c)c+1}\Big)^2\right)$$

3. The GGX [25] geometric occlusion function $G(\mathbf{l}, \mathbf{v}, \mathbf{h}; \mathbf{n})$

$$G(\mathbf{l}, \mathbf{v}; \mathbf{n}) = G_1(\mathbf{l}; \mathbf{n}).G_1(\mathbf{v}; \mathbf{n})$$

$$G_1(\mathbf{v}; \mathbf{n}) = \frac{2(\mathbf{n}.\mathbf{v})}{(\mathbf{n}.\mathbf{v}) + \sqrt{(\alpha^2 + (1-\alpha^2)(\mathbf{n}.\mathbf{v})^2)}}$$

The GGX microfacet function is differentiable in the domain of valid $\mathbf{l}$ and $\mathbf{v}$ directions. We differentiate the function by hand using the product rule and hardcode the resulting functions into the implementation. While the same procedure can be used with other microfacet models, the recommended method is to use an *auto-differentiator* package to avoid dealing with huge analytical differentials by hand.

Chapter 4

Shape Optimization

This chapter elaborates on the non-linear optimizer that uses gradients obtained from the differentiable renderer to iteratively update normals, BSDF and the shape of the object till we minimize the error between the target and estimated images. We show the loss function formulation followed by the details of the iterative process and the special gradient descent operators required for the algorithm to converge.

4.1 Loss function

The notation for the following sections assume that we have K images, each of size $W \times H$, and B basis BSDFs in our dictionary.

Our loss function is determined by the following equation

$$\mathcal{L} = \sum_{\mathbf{l} \in \mathbf{L}} \sum_{i \leq W, j \leq H} (I_{i,j}^{(l)} - \hat{I_{i,j}^{(l)}}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l}))^2 \mathbf{W^{(l)}}_{i,j}$$

where

1. $\mathbf{N}$ is a $W \times H \times 3$ tensor that represents the normal for each pixel
2. $\mathbf{Z}$ is a $W \times H$ matrix represents the depth at each pixel. $\mathbf{N}$ and $\mathbf{Z}$ together determine the mesh uniquely.
3. $\boldsymbol{\theta}$ is the BSDF parameter vector with B elements, which are the coefficients of the B basis BSDFs.

4. $\mathbf{l}$ is the light direction vector (depending on the estimator $\hat{I}$, this can also represent a set of parameters for various lighting models)
5. $\mathbf{W}$ is the $W \times H \times K$ per-pixel per-image weight tensor. The algorithm uses weighted loss to deal with significant variation in intensities as well as shadows.

4.2 Set-Adaptive gradient descent

To optimize for normals, we need to account for the constraint on the normals of the normalmap N:

$$n_x^2 + n_y^2 + n_z^2 = 1$$

Differentiating our loss function w.r.t $\mathbf{N}$, we have

$$\frac{\partial \mathcal{L}}{\partial \mathbf{N}} = \sum_{\mathbf{l} \in \mathbf{L}} \sum_{i \leq W, j \leq H} 2(\hat{I}^{(l)}_{i,j}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l}) - I^l_{i,j}) \mathbf{W}^l_{i,j} \Big(\frac{\partial \hat{I}^l_{i,j}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l})}{\partial \mathbf{N}} \Big)$$

Once the gradients are computed, the next step is to use adaptive gradient descent (discussed in Chapter 1) to move the normals in the direction of fastest descent. Adaptive algorithms like Adam compute a per-parameter learning rate that allows much faster convergence in case of disparate gradients, which is the case with normal optimization. One problem with vanilla Adam is the assumption that parameters are independent of each other, which is true for something like a neural network, but does not apply to the individual components of a normalmap. This is because of the constraint $n_x^2 + n_y^2 + n_z^2 = 1$. We find that ignoring the effect of this constraint quickly causes the optimization to fail and diverge because of biased gradients. To avoid this, one simple method is to ensure that Adam uses a single effective learning rate for the three components of a normal (the effective rate is still different for different normals). This makes sure that the *direction* of the gradient is unaltered.

4.3 Exponentiated gradient descent

To optimize for BSDF parameters $\boldsymbol{\theta}$, we need to satisfy the constraints:

$$\sum_{\theta \in \boldsymbol{\theta}} \theta = 1$$

$$\forall \theta \in \boldsymbol{\theta}, \theta \geq 0$$

These constraints naturally lend themselves to a special gradient descent method that uses multiplicative updates instead of additive updates, known as *exponentiated gradient descent* [27].

Given a set of parameters and their gradients, for the optimization of θ_i, the update rules are:

$$\theta_i^{t+1} = \frac{\theta_i^t e^{-\eta \frac{\partial \mathcal{L}}{\partial \theta_i}}}{\sum_j \theta_i^t e^{-\eta \frac{\partial \mathcal{L}}{\partial \theta_j}}}$$

where our our differentials are computed as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \sum_{\mathbf{l} \in \mathbf{L}} \sum_{i \leq W, j \leq H} 2(\hat{I}_{i,j}^{(l)}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l}) - I_{i,j}^{l}) \mathbf{W}_{i,j}^{l} \left(\frac{\partial \hat{I}_{i,j}^{l}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l})}{\partial \boldsymbol{\theta}} \right)$$

This optimization strategy maintains both sum-to-1 and the non-negativity constraints. But more importantly, it solves a lot of the problems with standard projected gradient descent, which in this case was extremely impractical due to the nature of the BSDF updates. The repeated projection of the updated vector back to the constrained vector space can lead to a situation where there is either no progress at all or very slow progress. Exponentiated gradient avoids this problem by using multiplicative updates that ensure progress no matter what the initialization is (except for initializing any parameter to 0, which will cause the parameter to be discarded)

4.4 Accounting for shadows

A popular problem when working with photometric stereo is shadows. Most methods do not account for occlusion, and it is very difficult to reliably tell which pixels are occluded and by what, without first constructing a mesh, but then if the mesh was available, there would be no need for a stereo algorithm. This is another chicken-and-egg problem. To deal with this, our algorithm employs a familiar strategy of starting with an initialization and iteratively improving it. However, one problem is that the error from shadowed regions is rather severe, and may cause the mesh to diverge early in the iterative optimization. We employ the use of weighted loss functions to tackle this. The weight function $\mathbf{W}$ is defined as follows:

$$W_{i,j,k} = \begin{cases} 0: \text{if } I_{i,j}^{(k)} \text{ is lower than } 80\% \text{ of } I_{i,j} \text{ of all } K \text{ images} \\ 1: \text{otherwise} \end{cases}$$

This process reduces the contribution from the shadowed/low-signal regions of the image to the loss function, allowing the optimization to focus on the brightly lit areas. This selective optimization technique is also commonly used in other photometric stereo algorithms and, in most cases, provides robustness to occlusion effects.

4.5 Computing the depth

The final stage of the optimization loop is retrieving the depths $\mathbf{Z}$, which uniquely define the surface. Chapter 1 introduced the weighted Poisson surface reconstruction algorithm that uses a linear system to optimize for depth $\mathbf{Z}$ from a normal map $\mathbf{N}$. In this section, we describe the weight vector $\mathbf{W}$ that the framework uses. The reason for using weighted Poisson reconstruction, instead of something like Framkot-Chellapa [28] integration, is that not all of the normals in the normalmap are estimated with the same confidence. Since that the gradients $\frac{\partial z}{\partial y}$ and $\frac{\partial z}{\partial x}$ are obtained by dividing the x and y components by the z component, this introduces higher error for gradients where the z component is smaller (normals that are close to perpendicular to the viewing direction). To combat this error we define our weights W to be proportional to the z component of the normal n. This reduces the effect of grazing angle normals, making the solution more robust.

$$W_{i,j} = n_z^{(i,j)}$$

4.6 Multiresolution optimization

One technique that we find allows for faster and more robust convergence is the use of multi-resolution optimization. All operations in a single iteration occur at a fixed resolution $W \times H$, but the resolution can be varied in between two iterations. We find that this retrieves an approximate mesh first and then proceeds to fill in the details. It also prevents the optimization from slowing down or getting stuck at local minima or saddle points. Our algorithm operates on a fixed schedule. The first $K_0 = 10$ iterations occur at $\frac{W}{4} \times \frac{H}{4}$ followed by $K_1 = 10$ iterations at $\frac{W}{2} \times \frac{H}{2}$. The remaining iterations are processed at full $W \times H$ resolution.

4.7 The Complete algorithm

In this section we describe the flow of the algorithm in practice and how the various components fit together.

```
**Result:** Depth $\mathbf{Z}$
$\mathbf{Z}_0, \mathbf{N}_0 = \text{PhotometricStereo}(\mathbf{I}, \mathbf{L})$
while $|\mathbf{I} - \hat{\mathbf{I}}|^2 \geq \epsilon$ do
    $\tilde{\mathbf{N}}_0 = \mathbf{N}_i$
    while $j \leq T_n$ do
        /* Multiple updates to $\mathbf{N}$ */
        $\frac{\partial \hat{\mathcal{L}}}{\partial \mathbf{N}} = \sum_{\mathbf{l} \in \mathbf{L}} \sum_{i \leq W, j \leq H} 2(\hat{I}^{(l)}_{i,j}(\tilde{\mathbf{N}}_j, \mathbf{Z}_i, \boldsymbol{\theta}; \mathbf{l}) - I^l_{i,j}) \mathbf{W}^l_{i,j} \left( \frac{\partial \hat{I}^l_{i,j}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l})}{\partial \mathbf{N}} \right)$
        $\tilde{\mathbf{N}}_{j+1}$ = Adam($\tilde{\mathbf{N}}_j$, $\frac{\partial \hat{I}}{\partial \mathbf{N}}$)
    end
    $\mathbf{N}_{i+1} = \tilde{\mathbf{N}}_{T_n}$
    $\tilde{\boldsymbol{\theta}}_0 = \boldsymbol{\theta}_i$
    while $j \leq T_\theta$ do
        /* Multiple updates to $\boldsymbol{\theta}$ */
        $\frac{\partial \hat{\mathcal{L}}}{\partial \boldsymbol{\theta}} = \sum_{\mathbf{l} \in \mathbf{L}} \sum_{i \leq W, j \leq H} 2(\hat{I}^{(l)}_{i,j}(\tilde{\mathbf{N}}_j, \mathbf{Z}_i, \boldsymbol{\theta}; \mathbf{l}) - I^l_{i,j}) \mathbf{W}^l_{i,j} \left( \frac{\partial \hat{I}^l_{i,j}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l})}{\partial \boldsymbol{\theta}} \right)$
        $\tilde{\boldsymbol{\theta}}_{j+1}$ = ExpGrad($\tilde{\boldsymbol{\theta}}_j$, $\frac{\partial \hat{I}}{\partial \boldsymbol{\theta}}$)
    end
    $\boldsymbol{\theta}_{i+1} = \tilde{\boldsymbol{\theta}}_{T_\theta}$
    $\mathbf{Z}_{i+1} = \text{WeightedPoissonReconstruction}(\mathbf{N}_{i+1})$
end
```

Algorithm 1: Shape Optimization through Differentiable Rendering

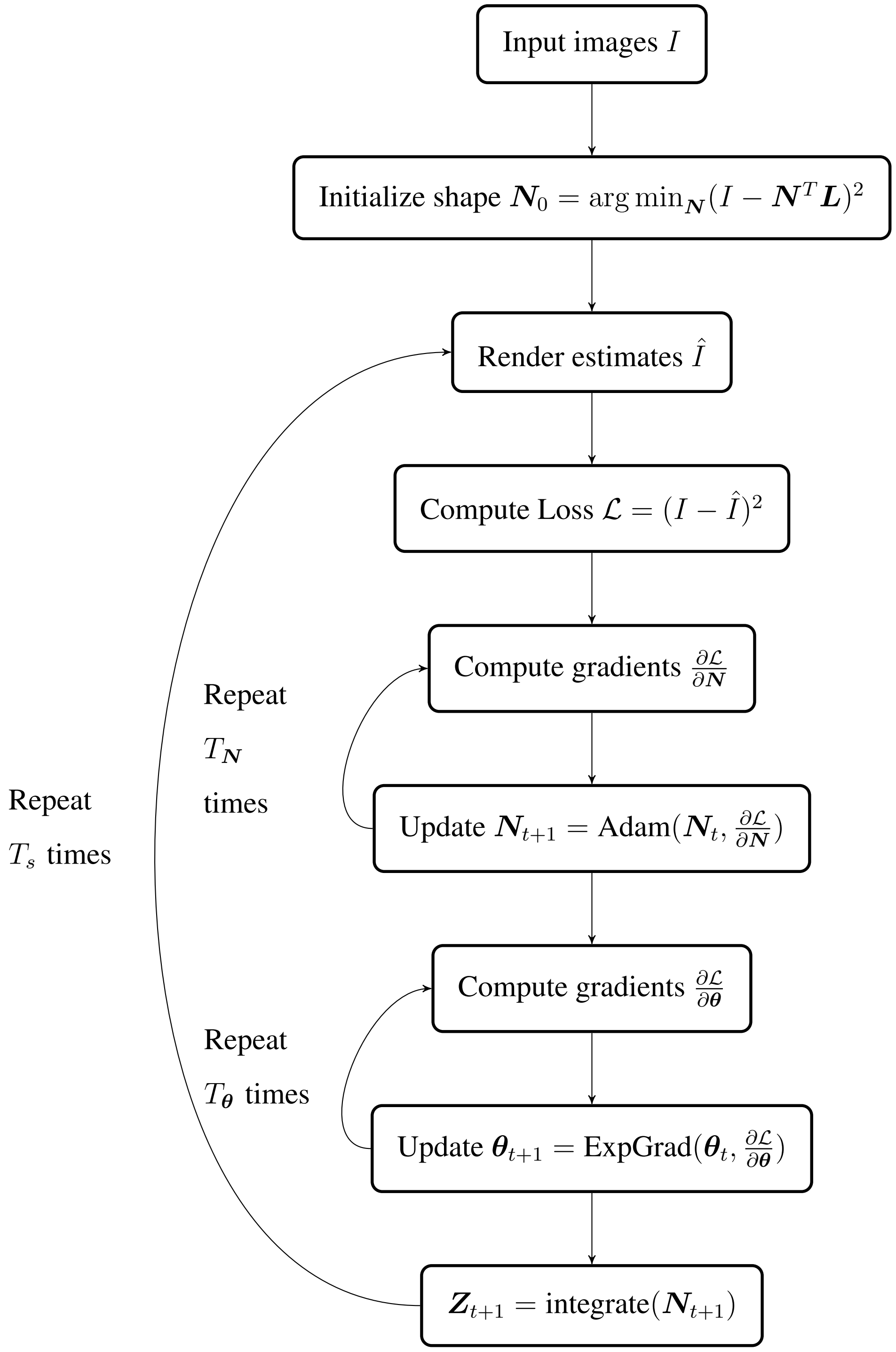
Input images I
Initialize shape $\boldsymbol{N}_0 = \arg\min_{\boldsymbol{N}} (I - \boldsymbol{N}^T \boldsymbol{L})^2$
Render estimates $\hat{I}$
Compute Loss $\mathcal{L} = (I - \hat{I})^2$
Compute gradients $\frac{\partial \mathcal{L}}{\partial \boldsymbol{N}}$
Update $\boldsymbol{N}_{t+1} = \text{Adam}(\boldsymbol{N}_t, \frac{\partial \mathcal{L}}{\partial \boldsymbol{N}})$
Compute gradients $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$
Update $\boldsymbol{\theta}_{t+1} = \text{ExpGrad}(\boldsymbol{\theta}_t, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}})$
$\boldsymbol{Z}_{t+1} = \text{integrate}(\boldsymbol{N}_{t+1})$
Repeat T_s times
Repeat $T_{\boldsymbol{N}}$ times
Repeat $T_{\boldsymbol{\theta}}$ times

Figure 4.1: Block diagram describing all the components of the shape optimization framework

Chapter 5

Results

We present optimization results on multiple shapes and BSDFs. Our results at this point are largely based on images rendered by a physically-based renderer *mitsuba*. This is because of a scarcity of public datasets that deal with concave objects.

But, considering the scope of applications for this specific optimization algorithm, a physically-based renderer is sufficient to capture the relevant lighting effects. Our algorithm does not currently deal with caustics (light paths that are the result of one specular reflection followed by one diffuse reflection), so there are no major discrepancies between synthetic and real-world datasets. To demonstrate this, however, we also include one instance from the publicly available photometric stereo dataset 'DiLiGent' [29].

We do, however, use captured BSDFs in our synthetic datasets, since analytical BSDF models still do not satisfactorily explain real-world BSDFs.

5.1 Shape Reconstruction

We present results on 4 different datasets:

1. **(Synthetic) Convex mask**. A convex mask with the 'gray-plastic' MERL BRDF is used to demonstrate the ability to optimize for BSDF parameters as well as shape in the presence of highly specular surface and mild shadowing. Our algorithm also handles the inter-reflections in the concavities near the nose and lips.

2. **(Captured) Toy Bear**. Taken from the DiliGenT benchmark dataset, this dataset demonstrates our algorithms ability to work on real world photometric stereo data.
3. **(Synthetic) Bowl**. The **bowl** dataset rendered with ’dark-blue-plastic’ BSDF is the primary tool to demonstrate our dataset’s ability to account for interreflections in the presence of non-lambertian BSDF.
4. **(Synthetic) Concave mask**. The inverted mask dataset rendered with the ’dark-blue-plastic’ BSDF combines all the features of our algorithm, including complex shape, arbitrary reflectance and interreflectons.

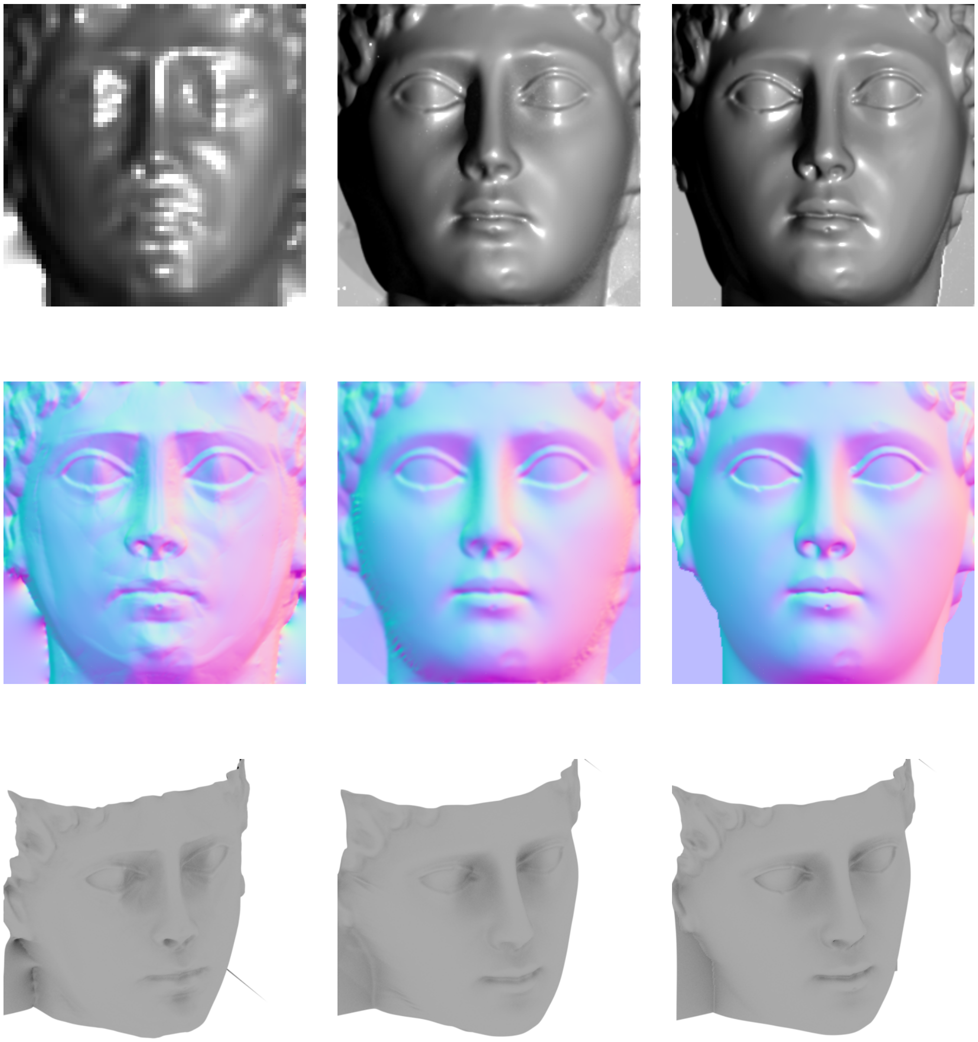

Figure 5.1: A highly glossy convex mask reconstructed from a set of synthetically rendered images of a mask. The images on the left are the initialization, the images in the middle are the reconstructions obtained by refining the initialization using our method. The images on the right represent the groud truth. In this case, we use multi-resolution rendering to avoid local minima, so the initial photometric mesh is processed at half the resolution, resulting in a blurry top-left image

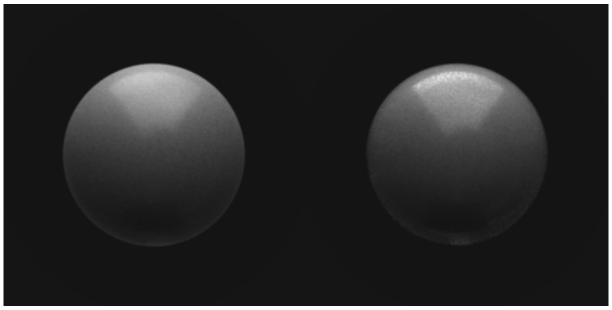

Figure 5.2: A sphere rendered under the BSDF reconstructed from the convex mask dataset (right) compared with one rendered using the original BSDF (left), which in this case is the 'gray-plastic' BSDF from the MERL database

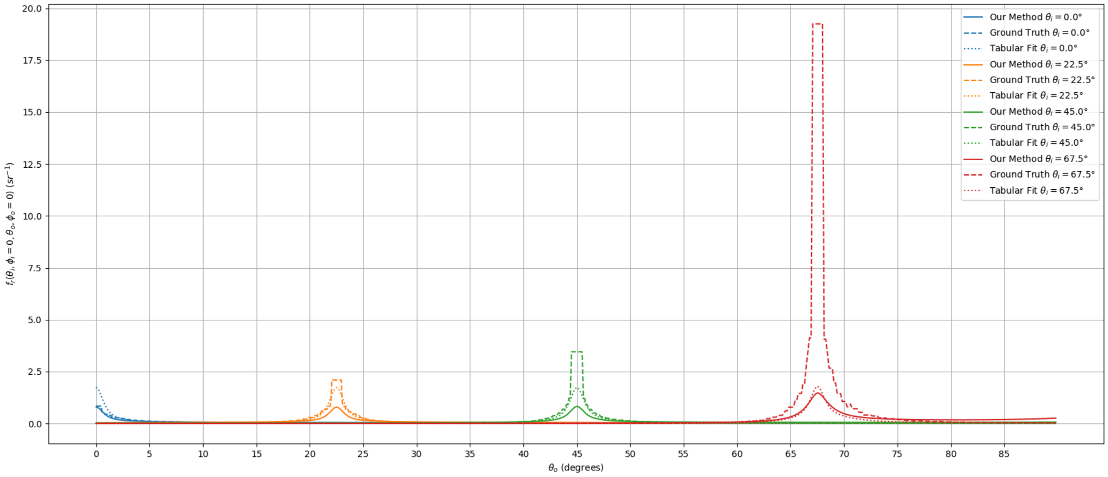

Figure 5.3: A comparison of the BSDF at various input angles θ_i deduced using *(a)* our algorithm (solid lines) with *(b)* the ground truth (dashed lines) and *(c)* the direct linear fit produced using the method in Section 3 (Tabular fit). Method *(c)* represents the best fit that can be produced by our reduced dicitonary of BSDFs (dotted lines). Note that this visualization has BSDF values that are adjusted so that all lines are clearly visible, but the relative error is preserved for comparison.

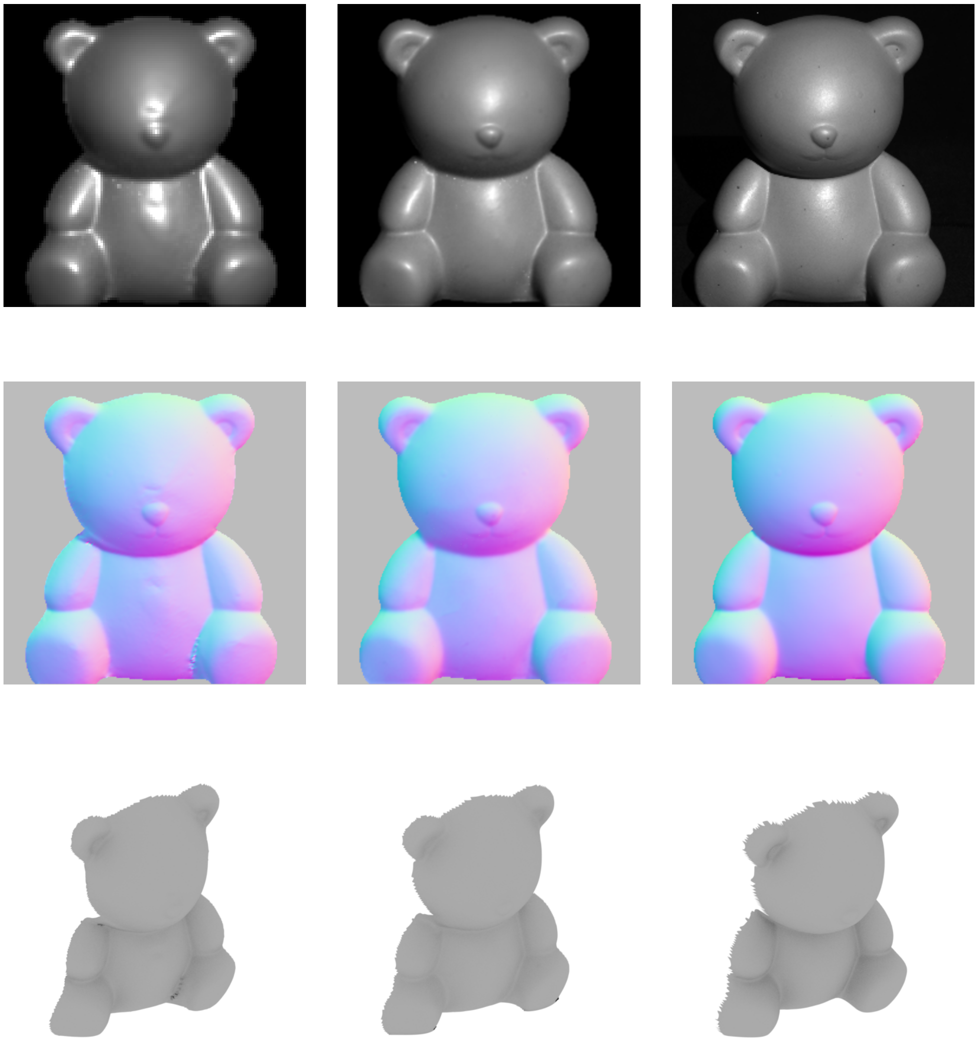

Figure 5.4: A bear reconstructed from the DiLiGent dataset. The images on the left are the photometric reconstruction, the images in the middle are the reconstruction obtained by refining the photometric mesh using our method. The images on the right represent the groud truth

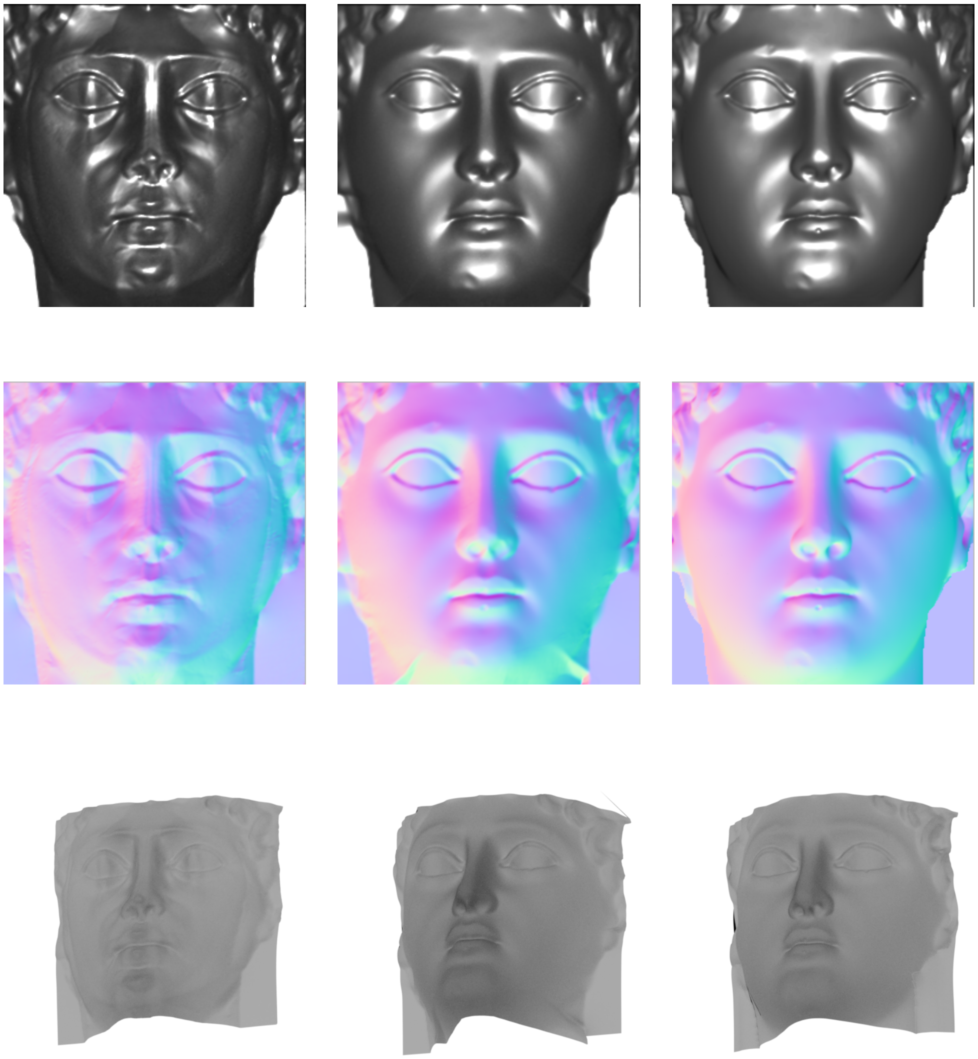

Figure 5.5: An inverted (concave) mask reconstructed from a set of synthetically rendered images of a mask. The images on the left are the photometric reconstruction, the images in the middle are the reconstruction obtained by refining the photometric mesh using our method. The images on the right represent the groud truth

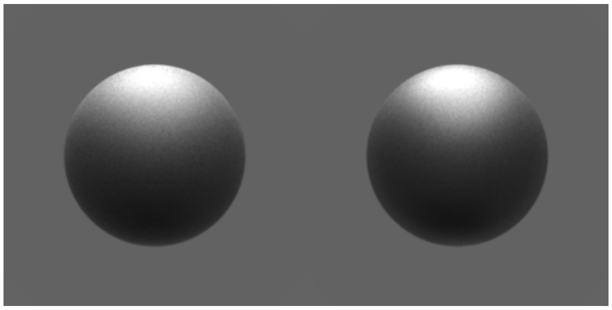

Figure 5.6: A sphere rendered under the BSDF reconstructed from the concave mask dataset (right) compared with one rendered using the original BSDF (left), which in this case is the 'dark-blue-paint' BSDF from the MERL database

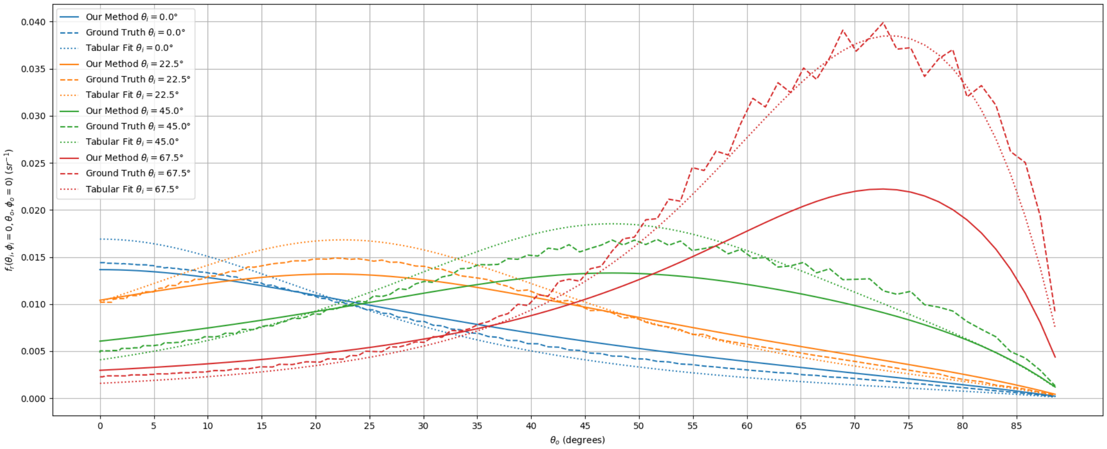

Figure 5.7: A comparison of the BSDF function at various input angles θ_i deduced using *(a)* our algorithm (solid lines) with *(b)* the ground truth (dashed lines) and *(c)* the direct linear fit produced using the method in Section 3 (Tabular fit). Method *(c)* represents the best fit that can be produced by our reduced dictionary of BSDFs (dotted lines). Note that this visualization has BSDF values that are adjusted so that all lines are clearly visible, but the relative error is preserved for comparison

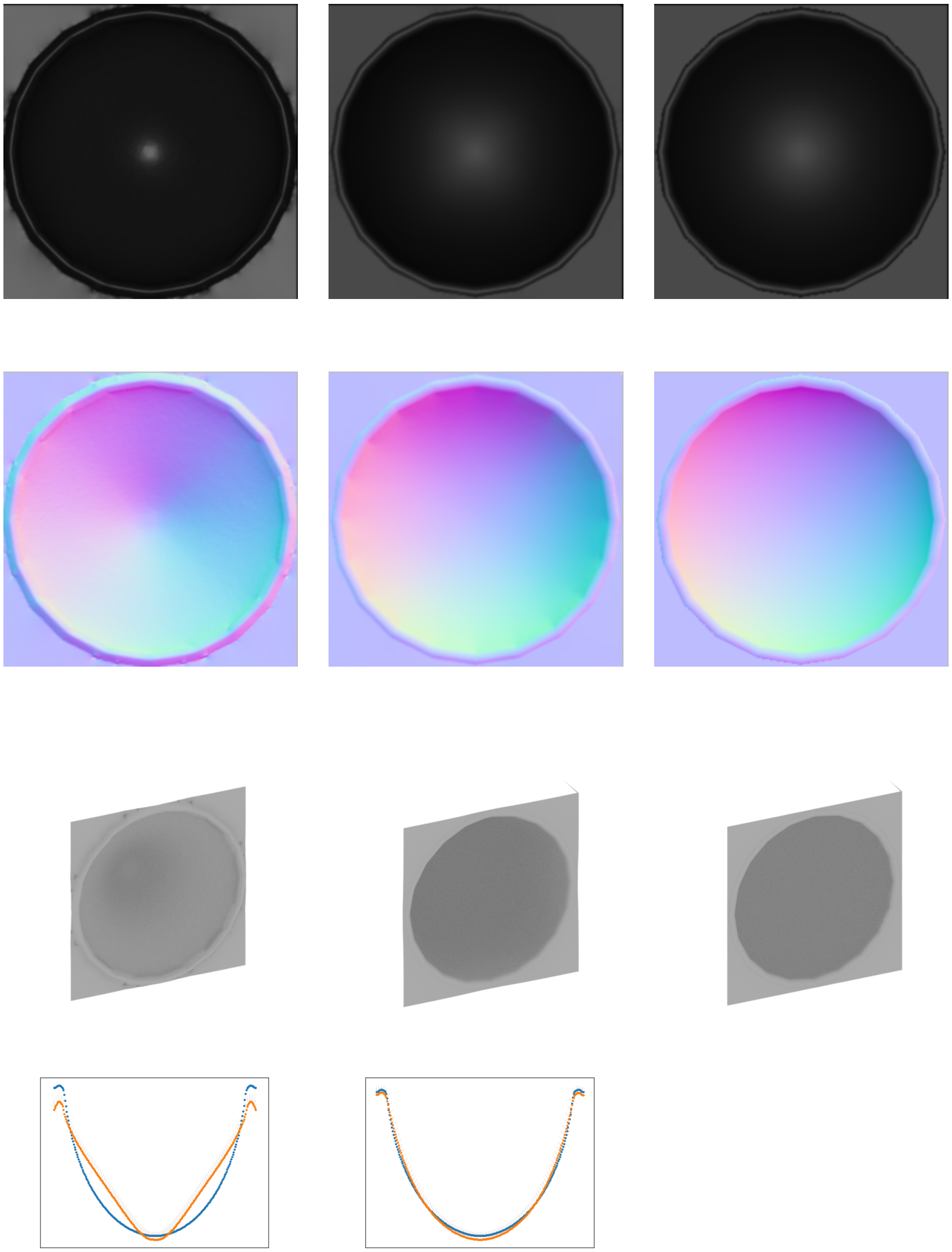

Figure 5.8: A concave bowl reconstructed from a set of synthetically rendered images of a bowl under the 'dark-blue-paint' BSDF. The images on the left are the photometric reconstruction, the images in the middle are the reconstruction obtained by refining the photometric mesh using our method. The images on the right represent the ground truth. The two images in the bottom row show the cross-section of the bowl before and after refinement. The blue lines are the ground truth, while the orange lines are the estimates produced by our algorithm.

Chapter 6

Conclusion

This thesis presents a novel approach to the problem of shape optimization in the presence of complex light transport effects. We have described a complex yet versatile framework for differentiable rendering that has long-reaching applications in inverse rendering. We have further applied this framework to the problem of inferring depth in the presence of interreflections as well as non-lambertian surfaces. The results acheived on both the synthetically rendered datasets, as well as a publically-available benchmark dataset, show significant improvements over traditional methods of shape retrieval.

6.1 Future Directions

6.1.1 Adaptive sampling

One interesting technique that can be used to great effect is adaptive sampling of light paths. Instead of using a fixed number of paths per pixel, it makes sense to prioritize the pixels with higher error, since they have a greater contribution to the total error. This is a form on importance sampling applied to differentiable rendering, and will serve to significantly speed up the optimization process, allowing for greater fidelity within the same time bounds.

6.1.2 Depth optimization

The algorithm presented above iterated between these three steps.

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}) = \sum_{\mathbf{l}\in\mathbf{L}} (I - \hat{I}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l}))^2 \mathbf{W}_i$$

$$\min_{\mathbf{N}} \mathcal{L}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}) = \sum_{\mathbf{l}\in\mathbf{L}} (I - \hat{I}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l}))^2 \mathbf{W}_i$$

$$D = \text{integrate}(\boldsymbol{N})$$

This last step is because there is currently no path tracer algorithm that produces unbiased differentials w.r.t depth. In the immediate future, we are looking to differentiate w.r.t the depth of each pixel, replacing the final step with a third minimization problem. This will ideally improve the versatility of the algorithm.

$$\min_{\mathbf{Z}} \mathcal{L}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}) = \sum_{\mathbf{l}\in\mathbf{L}} (I - \hat{I}(\mathbf{N}, \mathbf{Z}, \boldsymbol{\theta}; \mathbf{l}))^2 \mathbf{W}_i \qquad (6.1)$$

6.1.3 Importance sampling the gradient

One common problem we encountered when computing the gradient is that the BDPT algorithm is designed to importance sample the intensity rather than the gradient, which is an entirely different function and whose magnitude is usually not correlated with the that of the image intensity distribution. This leads to high variance in the estimates and slows down convergence. There is a lot of potential to theoretically derive distributions that allow the importance sampling of the gradient of BSDFs instead of the BSDF itself, in order to significantly reduce variance.

6.1.4 Single-image BSDF acquisition

Consider a simple shape with very few normals, like an inverted prism, which has only three flat faces. Assuming directional lighting and orthographic sensors, the resultant image can only have 3 discrete measurements. This is also why prisms are not suitable for extracing BSDFs. Spheres, which have a unique normal at every position, are the preferred shape for this task. However, given that BSDFs are complex 4D functions, a single-shot of a sphere is not enough to capture the complete function (only a slice). This is where our framework can help. The idea is that concave objects have more information about the BSDF when compared with their equivalent convex surfaces, because the paths that contribute to the interreflections sample a more diverse set of inputs $(\theta_i, \theta_o, \phi_i, \phi_o)$ to the BSDF function. The detailed study by Tsai et al. ([30]) elaborates on this argument and shows that a lot more of the BSDF is sampled by secondary light paths. However, while that paper demonstrated this result for transient images, our framework can be potentially be used to recover more detailed BSDF from a single image of a concave shape, like a bowl or an inverted prism.

Bibliography

[1] Eric P Lafortune and Yves D Willems. Bi-directional path tracing. 1993. 1.2.1, 2.3

[2] Robert J Woodham. Photometric method for determining surface orientation from multiple images. *Optical engineering*, 19(1):191139, 1980. 1.2.2, 1

[3] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. *Indicator*, 1(1):0. 1.2.3

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1.2.4

[5] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 1.2.4

[6] Chao Liu, Srinivasa G Narasimhan, and Artur W Dubrawski. Near-light photometric stereo using circularly placed point light sources. In *Computational Photography (ICCP), 2018 IEEE International Conference on*, pages 1–10. IEEE, 2018. 1

[7] Shree K Nayar, Katsushi Ikeuchi, and Takeo Kanade. Shape from interreflections. *International Journal of Computer Vision*, 6(3):173–195, 1991. 2

[8] 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, 2010. 3

[9] Neil Alldrin, Todd Zickler, and David Kriegman. Photometric stereo with non-parametric and spatially-varying reflectance. 2008. 3

[10] Shree K Nayar, Gurunandan Krishnan, Michael D Grossberg, and Ramesh Raskar. Fast

separation of direct and global components of a scene using high frequency illumination. *ACM Transactions on Graphics (TOG)*, 25(3):935–944, 2006. 1.3.1

[11] Supreeth Achar, Joseph R Bartels, William L Whittaker, Kiriakos N Kutulakos, and Srinivasa G Narasimhan. Epipolar time-of-flight imaging. *ACM Transactions on Graphics (ToG)*, 36(4):37, 2017. 1.3.1

[12] Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014. 1.3.2

[13] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. 1.3.2

[14] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (TOG)*, 37(4):128, 2018. 1.3.2

[15] Guilin Liu, Duygu Ceylan, Ersin Yumer, Jimei Yang, and Jyh-Ming Lien. Material editing using a physically based rendering network. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2280–2288. IEEE, 2017. 1.3.2

[16] Ioannis Gkioulekas, Anat Levin, and Todd Zickler. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *European Conference on Computer Vision*, pages 685–701. Springer, 2016. 1.3.2

[17] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse volume rendering with material dictionaries. *ACM Transactions on Graphics (TOG)*, 32(6):162, 2013. 1.3.2

[18] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. In *SIGGRAPH Asia 2018 Technical Papers*, SIGGRAPH Asia ’18, pages 222:1–222:11, New York, NY, USA, 2018. ACM. 1.3.2

[19] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’86, pages 143–150, New York, NY, USA, 1986. ACM. 2.1

[20] Eric Veach. *Robust monte carlo methods for light transport simulation*. Number 1610.

Stanford University PhD thesis, 1997. 2.1

[21] James Arvo. *Analytic methods for simulated light transport*. PhD thesis. 2.2

[22] Harley Flanders. Differentiation under the integral sign. *The American Mathematical Monthly*, 80(6):615–627, 1973. 2.2

[23] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. Efficient isotropic brdf measurement. 2003. 3.3, 3.3.2

[24] Jonathan Dupuy. *Photorealistic Surface Rendering with Microfacet Theory*. Theses, Université Claude Bernard - Lyon I, November 2015. 3

[25] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 195–206. Eurographics Association, 2007. 1, 3

[26] Robert L Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG)*, 1(1):7–24, 1982. 2

[27] Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. 4.3

[28] RT Framkot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):439–451, 1988. 4.5

[29] Boxin Shi, Zhe Wu, Zhipeng Mo, Dinglong Duan, Sai-Kit Yeung, and Ping Tan. A benchmark dataset and evaluation for non-lambertian and uncalibrated photometric stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3707–3716, 2016. 5

[30] Chia-Yin Tsai, Ashok Veeraraghavan, and Aswin C Sankaranarayanan. Shape and reflectance from two-bounce light transients. In *Computational Photography (ICCP), 2016 IEEE International Conference on*, pages 1–10. IEEE, 2016. 6.1.4