



PER VERTEX BRDF ACQUISITION

Daniel Pinto Coutinho

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Ricardo Guerra Marroquim

Rio de Janeiro
Fevereiro de 2015

PER VERTEX BRDF ACQUISITION

Daniel Pinto Coutinho

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.


Examinada por:



Prof. Ricardo Guerra Marroquim, D.Sc.



Prof. Claudio Esperança, Ph.D.



Prof. Asla Medeiros e Sá, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2015

Coutinho, Daniel Pinto

Per Vertex BRDF Acquisition/Daniel Pinto Coutinho.
– Rio de Janeiro: UFRJ/COPPE, 2015.

XVI, 72 p.: il.; 29, 7cm.

Orientador: Ricardo Guerra Marroquim

Dissertação (mestrado) – UFRJ/COPPE/Programa de
Engenharia de Sistemas e Computação, 2015.

Referências Bibliográficas: p. 68 – 72.

1. BRDF. 2. Interactive. 3. Computer Graphics.
I. Marroquim, Ricardo Guerra. II. Universidade Federal
do Rio de Janeiro, COPPE, Programa de Engenharia de
Sistemas e Computação. III. Título.

To my family

Acknowledgement

First of all, I would like to thank my family. Their immense support and comprehension of my departure to pursue my master's degree were of a great value to me. They were helping in every decision and I am certain that they will continue doing it throughout the rest of my journey.

I want to thank everybody of the Visual Computing Lab of CNR-ISTI in Pisa. They were generously helpful during my stay there and great companions to eat pizza. I would like to specially thank Matteo Dellepiane for all the contribution throughout this work, if it were not for him, probably this dissertation would never have been born. He was a real mentor during my time in Italy. I would also like to thank Roberto Scopigno for the sincere welcome in their group.

All my professors here in LCG have been really supportive, and what I must say, not the usual terror of graduate school advisors. I would like to thank prof. Antonio Oliveira for his help in the design of the optimization algorithm and weekly discussions about the greatest football team in the world, Vasco da Gama. I would also want to thank prof. Claudio Esperança, who is one of the best teachers I ever had the pleasure of knowing. Last, but not least, prof. Ricardo Marroquim has been more than my advisor during these two years, he has been a real friend, giving a lot of support in my academic and personal life. I owe him a lot and will be forever grateful for this experience.

Finally, I want to thank my friends from LCG who have been following my progress and actively participating in this work, whether with suggestions during the weekly meetings or by having fun during the weekends. In order to prevent myself from not citing anyone and causing any resentment, I just want to thank everybody equally.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

AQUISIÇÃO DE BRDF POR VÉRTICE

Daniel Pinto Coutinho

Fevereiro/2015

Orientador: Ricardo Guerra Marroquim

Programa: Engenharia de Sistemas e Computação

Esta dissertação apresenta um sistema de aquisição para aproximar um modelo simples de BRDF por vértices de um objeto do mundo real. A partir de um modelo digital obtido digitalizando o objeto físico e fotos capturadas com diferentes posições de luz, nosso sistema reproduz fielmente o comportamento dos materiais que compõem o objeto. A principal vantagem do nosso trabalho é a capacidade de gerar resultados *online* e fornecer feedback imediato para o usuário. Nós apresentamos um sistema rápido e que não necessita de aparatos complexos ou um ambiente de aquisição muito controlado.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

PER VERTEX BRDF ACQUISITION

Daniel Pinto Coutinho

February/2015

Advisor: Ricardo Guerra Marroquim

Department: Systems Engineering and Computer Science

This dissertation presents an acquisition system that approximates a simple BRDF model per vertex of a real world object. Given a digital mesh from the physical object and photos taken with different light positions, our system faithfully represents the material that composes the object. The main advantage of our work is the ability to generate online results and provide immediate feedback to the user. We present an efficient system that doesn't require complex devices or an over controlled acquisition environment.

Contents

List of Figures	xi
List of Tables	xvi
1 Introduction	1
1.1 Motivation	2
1.2 Thesis Contribution	2
1.3 Thesis Outline	3
2 Background	4
2.1 Reflectance	4
2.1.1 Radiometry	4
2.1.2 BRDF	5
2.1.3 BRDF Models	6
2.1.3.1 Lambertian	6
2.1.3.2 Phong	7
2.1.3.3 Blinn-Phong	7
2.1.3.4 Oren-Nayar	8
2.1.3.5 Ward	8
2.1.3.6 Lafortune	9
2.2 Color Spaces	9
2.2.1 RGB	9
2.2.2 sRGB	9
2.2.3 CIE XYZ	10
2.2.4 CIELAB	10
2.3 Camera models	11
2.3.1 Projective camera	12
2.3.2 OpenGL camera	12
2.3.3 Conversion between different models	13

3	Related Works	14
3.1	Color mapping	14
3.2	Controlled Lighting	15
3.3	General Lighting	18
3.4	Devices	20
3.5	Example-based	21
3.6	Summary	22
4	Algorithm	24
4.1	Sphere detection	26
4.2	Color Chart detection	28
4.3	Mesh-Photo Alignment	29
4.4	Diffuse color	30
4.4.1	Generating New Light Directions	32
4.5	Specular coefficient	37
4.6	Optimization cycle	39
4.6.1	Diffuse and Specular Coefficients	41
4.6.2	Shininess	41
4.6.3	Color	42
4.7	Changing Viewpoints	43
4.7.1	Quality Metric	43
5	Experimentation and analysis	45
5.1	Results	45
5.1.1	Nana	45
5.1.2	Buddha	50
5.1.3	Gertrud	51
5.1.4	Goats	52
5.2	Discussions	54
5.2.1	Local vs Global Optimization	56
5.2.2	SRGB vs RGB	57
5.2.3	Parameters	59
5.2.4	BRDF Model	63
5.2.5	Size	63
5.2.6	Number of photos	64
5.2.7	Time and Feedback	64
6	Conclusion	66
6.1	Future Works	66

List of Figures

2.1	Object rendered with different reflectance models	8
4.1	Mesh-Image alignment for Nana	24
4.2	There are basically four steps in our system: calibration, basic diffuse color estimation, specular coefficient estimation and optimization cycle.	25
4.3	Our typical acquisition environment consisting of the target object, a reflecting sphere to extract light direction, and a color chart for calibration purposes.	26
4.4	Demonstration of a sphere used to capture the light direction	27
4.5	Calculating the light direction from the reflective sphere.	28
4.6	Mesh-Image alignment for the Palm tree model	30
4.7	Cyan represents an area where the vertices are in highlight, orange delimits self-shadowing areas, and purple delimits the area of low diffuse value. All other vertices are considered for this camera-light pair.	32
4.8	We want to calculate a new light direction that generates a specular value lower than T_s . The idea is to set the desired reflection vector \vec{R}' at a position in which $(\vec{R}' \cdot \vec{E}) = T_s$. To find the position \vec{R}' we need to express the difference $T_s - (\vec{R} \cdot \vec{E})$ in terms of angular distance. The angle a is $\cos^{-1}(\vec{R} \cdot \vec{E})$ and b is $\cos^{-1}(T_s)$. We find the difference between them and just rotate R ($b - a$) degrees with respect to $\vec{E} \times \vec{R}$. The best light is then just the reflection of \vec{R}' with respect to the vertex's normal.	33
4.9	Spatial Bins example	35
4.10	The datasets used in our work, top row, left to right: Nana, Nana Back. Bottom row, left to right: Buddha, Gertrud, Goats	36

4.11	Hemisphere coverage during time. Note that after a few iterations, the number of bins is greatly reduced. The Goats case is rather odd, because although it seems a flat surface, it is composed of several bumps whose normals point to directions that are complicated to be captured by the current viewpoint. That it is way it never converges and more viewpoints are needed to fully cover the vertices.	36
4.12	We want to calculate a new light direction that generates a specular value greater than T_s . The idea is to set the desired reflection vector \vec{L}' at a position in which $(\vec{R}' \cdot \vec{E}) = T_s$. To find the position \vec{L}' we need to express the value of T_s in terms of angular distance. The angle a is $\cos^{-1}(T_s)$. Then, we just rotate L a degrees with respect to $\vec{L} \times \vec{N}$. We actually subtract a small amount from a , because the new specular value should be higher than T_s so it can be considered a specular light direction.	39
4.13	The result of the optimization process on the diffuse coefficient.	41
4.14	Result of the specular coefficient's optimization. We can observe how it increases the brightness of the shiniest spots.	41
4.15	Result of the shininess'optimization.	42
5.1	A comparison between the final rendering in 5.1a and a photo with corresponding light direction 5.1b. Our result faithfully captures the object's diffuse color and the highlight areas.	46
5.2	More results for Nana. Left column shows the rendering results and the right column photos with associated light directions. Again, we can notice a good diffuse color extraction and highlight shape reproduction. Our algorithm successfully approximates the highlight area, but was not able to reproduce the same specular intensity, this is more perceptible on the shoes.	46
5.3	Visualization of diffuse and specular coefficients. Observe the dark dots present on Nana's head in the diffuse coefficients photo and how these match the white dots in the specular coefficients photo. The error of specular photos are guiding the optimization process to set the specular coefficient too high while setting the diffuse coefficient too low.	47
5.4	Comparison for the Nana Back between the rendering, left, and the photo, right.	47

5.5	Results for Nana. Left column shows the rendering results and the right column photos with associated light directions. Again the diffuse color approximation results are of high quality as well as the highlight shape. Unlike the first scenario where some highlight were not as intense as in the photo, for the Nana Back in some cases the highlight are even brighter than the photo, as the yellow base in Figure 5.5a.	48
5.6	Nana rendered from a novel viewpoint. Black areas especially in the neck or the hand are vertices that were not seen from the fixed viewpoint, a different viewpoint is required to approximate their BRDFs.	49
5.7	Nana Back rendered from a novel viewpoint.	49
5.8	A comparison between the final rendering in 5.8a and a photo with corresponding light direction 5.8b. While we can adequately approximate the robe’s BRDF, and the highlight shape to some extent, the problem experienced with the diffuse holes prevents a better reflectance approximation.	50
5.9	Comparing the three images it is noticeable that the vertices with only one specular photo are influencing heavily the optimization towards the specular value. This actually seems odd, as one would expect that having less specular photos would guide the optimization towards the diffuse coefficients. However, the very shiny nature of the buddha and the dark diffuse color presents a rather peculiar situation. Very low diffuse values do not yield an error as large as the specular one for the Buddha. We performed the same comparison with the Nana and this behavior was not reproduced.	51
5.10	Results for the Gertrud dataset. We can see how the geometry of its face is much more noisy than the actual statue, which harms the performance of our algorithm. Although, it is still able to extract a good diffuse color, specially of its robe. Since neither the underlying object nor the painting is very specular, there are a few highlights which are not too as bright as expected.	51
5.11	Results for Gertrud. Left column shows the rendering results and the right column photos with associated light directions	52
5.12	Blending between Mesh and Image. Notice the misalignment.	53
5.13	A color mapping comparison. We compare the result of the Photo-scan color mapping software with our basic diffuse color extraction. Observing both results and the photos shown before, our results are much more accurate.	53
5.14	Results for Goats. Left column shows the rendering results and the right column photos with associated light directions	54

5.15	Difference between the photo and the rendering color. Left column represents the difference $P - C$ and right column $C - P$. We can observe that more illuminated areas are brighter in the photo than in our rendering results, whereas less illuminated areas are darker in photos.	55
5.16	Comparison between local and global optimization. Local optimizations fail to capture all highlight areas and its corresponding intensity. Notice how only some of the highlights present in the global optimization result are present in 5.16a.	56
5.17	Models rendered from a novel viewpoint. The local optimization result seems like the object is made from only one type of material, Nana's body is as specular as its head, which is not the behavior seen in the global optimization result nor in the photos shown earlier. . . .	56
5.18	Again we can notice that the specular behavior is not similar to the actual model. Although there are no diffuse holes in the local optimization result, the BRDF obtained also does not correspond to the expected material.	57
5.19	Coefficients obtained during the local optimization in the top row. Compare these with the global optimization results in the bottom row. Local optimization coefficients are too high.	58
5.20	A comparison of the results obtained from the different color spaces. It is clearly noticeable that results from the RGB space are generally darker than the SRGB. This discrepancy increases as the diffuse value decreases, notice how its neck progressively darkens. The highlights are also not as bright as in the SRGB, the bright spot near the top of the head looks more reddish than in the right.	58
5.21	The same divergences for the Nana can be seen in the Buddha. The RGB color space produces a darker result, look at Buddha's left arm and the robe around its right knee. The highlights in SRGB are also brighter, note its right arm.	59
5.22	Rendering of Gertrud with different values of α . Although subtle, there's a noticeable difference in the shapes and intensities of the highlights in each. As expected with a increase of α , the highlights tend to be localized. Observing the highlights in the red areas, we can see that it progressively gets smaller.	60
5.23	Photograph of Gertrud. We can notice that the highlights are more compatible with the results where α is small.	60

5.24	The different values of shininess in the Nana Back dataset produce more varied results than Gertrud. When α is small, as in the top row, the outcomes are moderately similar. In the case when $\alpha = 50$, we can notice more holes in Nana’s head than in the previous results and some highlights, as in its body, can no longer be seen. Some major changes can also be seen when $\alpha = 100$, as some bright dots in its head, and a few white areas near the top of Nana’s head, there is also no more highlights in its body.	61
5.25	From left to right: $n = [100, 1000]$. We could not detect major differences in the results with the change of n . In fact, it may have never even reach the maximum number of 1000, since it probably converges before.	62
5.26	As expected, the results for the diffuse coefficient remains the same with the change of the number of iterations.	62
5.27	Like the results for the diffuse coefficient, the specular coefficients remain the same with the change in the number of iterations.	63

List of Tables

4.1	We repeated the process 10 times for each dataset and took its average. Iterations are the number of lights, and consequently photos, needed to cover all vertices. For the Goats, 38 is the size of the complete dataset, since it never converged.	37
-----	--	----

Chapter 1

Introduction

”Stay a while and listen.”

— ”Deckard Cain”

Photorealistic rendering has been a goal in Computer Graphics almost since its beginning. A lot of focus has been placed on the development of global illumination algorithms[1], which try to mimic the physical behavior of light and its interaction with the environment. However, even the more advanced algorithms rely on the description of the object’s material properties. Rendering a ceramic vase and a bronze one should result in totally different results even if their geometry is identical, but this is only possible if there is some model capable of mimicking the material behavior. BRDFs are such models but for their parameters to be settled, some measurement of the specific material in hands must be carried on.

Bidirectional Reflectance Distribution Function (BRDF) is a mathematical function that describes the reflectance behavior of an object. Given a light direction, the BRDF returns the amount of light reflected towards a viewpoint. One can only imagine the complexity of translating a real world object material in terms of an analytical function. This has been a topic of research in the last decades and there has been a myriad of works published with different approaches [2]. Some of the techniques involve the usage of complex devices such as a gonireflectometer or special domes, while others rely only on information from photos and videos of the object. Our work consists in approximating a simple BRDF model using photos with an associated light direction.

Besides the desire of rendering photorealistic scenes, our work supplies the demand for other areas of research that focus on digital techniques applied to Cultural Heritage.

1.1 Motivation

The Cultural Heritage field focus on preservation, restoration and dissemination of historical artifacts. With the advancement of three-dimensional scanning and great improvements of DSLR cameras, the digitization of such objects with a high degree of quality has been possible. Although we can faithfully create a geometric copy of real world objects and extract a great amount of information about its color, there's still room for improvements regarding the extraction of its reflectance properties.

Even though there are works that almost generally solve the BRDF acquisition problem, they usually have some limitations such as: *object size, need to move the object, special acquisition environments*. Our work presents no restrictions about object's dimensions and its position. Furthermore, some acquisition environment conditions are desirable, but not mandatory.

A common pipeline for digitizing a historical artifact is scanning its geometry and then performing a series of photos acquisition. These tasks are usually done in situ, for example, a museum or archaeological site. After the acquisition step, offline calculations are performed for mesh refinement and/or color mapping of the photos on the model. It is not uncommon during the offline step to notice that there is not enough data to fully represent the object, usually a photo missing from a viewpoint. In certain campaigns this may be a major problem, since another trip to the site may not be viable. A conservative acquisition may be a less dramatic solution, but it spends much more time and acquires more data than necessary, sometimes redundant.

The core of our algorithm is performed using shaders which provides an enormous gain in performance, since all calculations are done in parallel. The use of shaders also allow us to treat self-shadowing with little cost, which could be a rather laborious task otherwise.

1.2 Thesis Contribution

The main contribution of our work is a fast algorithm that can approximate a simple BRDF model of a real world object. Our system is also able to provide the user with feedback about all the information required to faithfully represent the object and overall quality.

Along with our algorithm, we developed an acquisition system capable of providing online feedback about the quality result of our model and information about missing parts. Not only it provides this kind of detail, it can also suggest good light directions in order to have a better result and use less data. This is an important contribution of our work, since we are not aware of other tools used in the field that can provide

such online response.

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

- *Chapter 2*
In this chapter we discuss the basic concepts required for further understanding our work, covering topics ranging from reflectance and radiometry to color spaces and camera models.
- *Chapter 3*
We analyze the most relevant related works in acquiring BRDFs. We also discuss some works that focus only on color mapping, since this is also an inherent part of our algorithm.
- *Chapter 4*
In this chapter we detail how our algorithm works and each extraction part: basic diffuse color, specular estimation and optimization procedure.
- *Chapter 5*
We present our main results and discussions in this chapter.
- *Chapter 6*
Conclusions and directions for future works are discussed here.

Chapter 2

Background

In this chapter we set some of the theoretical background required to understand our work. Section 2.1 describes some definitions about radiometry, BRDF and illustrate some models. Color spaces are discussed in Section 2.2. In the last Section 2.3 we talk about some different camera models used in this work and how to convert between them.

2.1 Reflectance

We are concerned with the acquisition of an object's BRDF, which represents its reflectance, but what is really reflectance? Intuitively, we can describe it as the behavior of a surface with respect to light reflection. This is essentially our task, analyze how the object reflects light from a certain direction. A more formal definition is given by NICODEMUS *et al.* [3]: Reflectance is the fraction of incident light flux that is reflected. In order to understand reflectance, it is necessary to lay out the basic concepts of Radiometry.

2.1.1 Radiometry

Radiometry is the study of measurement of electromagnetic radiation. In this section we explore the basic radiometric concepts in order to comprehend shading models. The electromagnetic radiation is composed of photons possessing an associated wavelength. The wavelengths ranging from approximately 380 to 780 nanometers compose the visible spectrum. This spectrum is visible light to our eyes and perceived as colors.

The energy carried by photons is called *radiant energy* Q , which is measured in *joules* (J). Radiant Energy is most commonly known as the energy of electromagnetic waves, but since these are stream of photons, we can view radiant energy as the energy carried by photons. It's also worth noting that light and electromagnetic

waves are synonyms, therefore radiant energy can also be described as the energy of the light. The number of joules emitted per second is called *radiant flux* Φ , measured in J/s or *Watts*(W). For example, a light of $60W$ emits 60 joules per second. If we want to measure the amount of radiant flux that hits a surface we are measuring the *irradiance* E :

$$E = \frac{d\Phi}{dA}$$

Technically, irradiance is the amount of energy flowing into a surface and exitance, or radiant exitance, is the energy flowing out of a surface. The term radiant flux density can be used to refer to both. For directional light sources, where the light is much farther than the objects in the scene, irradiance is approximately constant over the scene. Light sources are assumed to be directional in our acquisition system.

Considering a unit radius sphere, a patch of the surface area is called *solid angle*, measured in *steradians*(sr). The solid angle represents a set of angle directions in 3D that cover the patch area. If we consider our light source emitting energy over a sphere, the flux of energy that will flow through a given solid angle ω , is the *radiant intensity*(I):

$$I = \frac{d\Phi}{d\omega}$$

If we want to measure the amount of radiant intensity that flows through a surface area, we are talking about *radiance*(L). Radiance is the amount of radiant flux per solid angle per surface area:

$$L = \frac{d\Phi}{d\omega dA \cos\theta}$$

The purpose when evaluating a shading model is to compute the radiance along a given ray.

2.1.2 BRDF

When dealing with photorealistic rendering of objects we need a way to quantify reflectance. For this purpose we use a *Bidirectional Reflectance Distribution Function* (BRDF). A BRDF encapsulates how the object reflects light from all possible incoming positions to every reflecting direction. Picking incident light direction ω_i and a reflecting direction ω_o , the BRDF of a material for the incoming direction ω_i reflected towards ω_o is:

$$f(\omega_i, \omega_o) = \frac{dL(\omega_o)}{dE(\omega_i)}$$

where

$$dE(\omega_i) = L_i(\omega_i) \cos\theta_i d\omega_i$$

It is important to notice that a BRDF describes a general reflectance distribution defined over the entire sphere of directions around the object. When evaluating it, we are providing a current incoming light and a desirable exitant ray, and we are generally asking: "How much of this light is reflected towards that direction?"

Though we have been saying that our goal is to acquire an object's BRDF, there is an important observation to make. A BRDF describes a material and not the object *per se*. It means that a golden paint possesses the same BRDF whether it is on stone or glass, even though the reflected light will usually be influenced by the paint's BRDF as well as the underlying material. Acquiring the BRDF of an object implicates in describing of which materials it is made.

There are some important properties of physically-based BRDFs:

Reciprocity The BRDF remains the same if the incident and exitance directions are exchanged.

Energy of conservation This principle establishes that all exitance radiance is less than or equal to the incident radiance.

Attributing a BRDF to an object implies that its object is composed of only one type of material. Unfortunately, CH artifacts do not behave so nicely in the majority of cases, especially if painted, when the reflectance of an object changes depending on the surface point. In this case we will need a spatially-varying BRDF (SVBRDF). Now, we are dealing with a six-dimensional parameter space:

$$f(\mathbf{x}, \omega_i, \omega_o) = \frac{dL(\mathbf{x}, \omega_o)}{dE(\mathbf{x}, \omega_i)}$$

As can be expected, to capture a real world object's BRDF is not an easy task. We would have to sample light from all possible directions and respectively position a camera in all possible reflecting positions, and measure for every surface point, which is simply unfeasible. What we perform in our work is the acquisition of a small to moderate number of samples and fitting of the data in an analytical BRDF model. Some of these models will be explained next.

2.1.3 BRDF Models

Here we describe known BRDF models in the Computer Graphics community. The models discussed are used in this dissertation and in many of the related works.

2.1.3.1 Lambertian

Simplest model of all, establishes that when light strikes a surface, it reflects uniformly in all directions. That way, from every possible viewpoint you look at a

lambertian surface, its perceived color is the same. However, this is only a theoretical model since real world objects do not exhibit purely lambertian behavior. Figure 2.1a shows an object with purely lambertian behavior, but only reflecting half of the incident irradiance.

2.1.3.2 Phong

Considered a breakthrough during Computer Graphics's early life, Phong model [4] provides a way to calculate a surface's reflectance considering three properties: diffuse coefficient k_d , specular coefficient k_s and shininess α . The idea is that the closer the surface's normal is to ω_i , the greater its radiance will be. Also, highlights can be seen when the reflected light is closer to ω_o . The resultant radiance of a point x is:

$$f(x, \omega_i, \omega_o) = k_d(\vec{N} \cdot \vec{\omega}_i)C_{rgb} + k_s(\vec{R} \cdot \vec{\omega}_o)^\alpha S_{rgb}$$

Where, \vec{N} is the surface's normal at x , \vec{R} is the reflection vector of $-\vec{\omega}_i$ with respect to \vec{N} . The color of the material is C_{rgb} and the light is S_{rgb} . The vector \vec{R} can be easily calculated by:

$$R = \omega_i - 2(\omega_i \cdot \vec{N})\vec{N}$$

We show an example of an object rendered with the Phong's model in Figure 2.1b. Its parameters are: $k_d = 0.5$, $k_s = 0.5$ and $\alpha = 10$.

2.1.3.3 Blinn-Phong

The Blinn model [5], often called Blinn-Phong, is a modification of the Phong model by introducing the halfway vector H . In a perfect mirror, the normal of the surface N would be pointed halfway between the incident light and the reflected ray. So, let H be:

$$H = \frac{\vec{\omega}_i + \vec{\omega}_o}{\|\vec{\omega}_i + \vec{\omega}_o\|}$$

Then, the radiance of a point x using the Blinn model is:

$$f(x, \omega_i, \omega_o) = k_d(\vec{N} \cdot \vec{\omega}_i)C_{rgb} + k_s(\vec{N} \cdot \vec{H})^\alpha S_{rgb}$$

We show an example of an object rendered with the Blinn-Phong's model in Figure 2.1c. Its parameters are the same as the Phong's model.

While both models are very popular and historically important, e.g. the Blinn model is the default implementation up to OpenGL 3, they are not physically plausible and do not maintain the BRDF's properties we established before. The next models we analyze are more physically accurate, and by a certain degree, more complex.



(a) A lambertian object reflecting half of irradiance. (b) An object rendered with the Phong's model. (c) An object rendered using the Blinn-Phong.

Figure 2.1: Object rendered with different reflectance models

2.1.3.4 Oren-Nayar

Oren and Nayar [6] observed that real world objects are not perfect lambertian surfaces, but instead are composed of small planar facets, which are lambertian. Oren-Nayar's model belongs to a class called *microfacet models*. The simplified resultant reflectance of a point is:

$$f(x, \omega_i, \omega_o) = \frac{\rho}{\pi} E_o \cos \theta_i (A + B \text{Max}[0, \cos(\phi_r - \phi_i)] \sin \alpha \tan \beta)$$

Where (θ_i, ϕ_i) are the angles of incidence and the exitance angles are (θ_r, ϕ_r) , ρ is the surface's albedo, and:

$$\begin{aligned} A &= 1.0 - 0.5 \frac{\sigma^2}{\sigma^2 + 0.33} \\ B &= 0.45 \frac{\sigma^2}{\sigma^2 + 0.09} \\ \alpha &= \max[\omega_i, \omega_o] \\ \beta &= \min[\omega_i, \omega_o] \end{aligned}$$

Notice that for the case when $A = 1$ and $B = 0$ we return to the lambertian model, so Oren-Nayar's model can be viewed as its generalization.

2.1.3.5 Ward

Intended to develop a model that both fits measured data and is relatively simple in order to maintain performance. Its full equation is able to generate anisotropic models:

$$f(x, \omega_i, \omega_o) = \frac{\rho_d}{\pi} + \rho_s \frac{1}{\sqrt{\cos(\omega_i) \cos(\omega_o)}} \frac{\exp(-\tan^2 \delta (\frac{\cos^2 \phi}{\alpha_x^2} + \frac{\sin^2 \phi}{\alpha_y^2}))}{4\pi \alpha_x \alpha_y}$$

Where ρ_d and ρ_s are respectively the diffuse and reflectance coefficient, α_x and α_y are the standard deviation of the surface slope in the \vec{x} and \vec{y} directions, δ is the angle between the halfway vector and the surface's normal, and ϕ is the azimuth angle of the half vector projected onto the surface plane.

2.1.3.6 Lafortune

Observing the flaws in Phong’s model, such as its rapid decrease in reflectance towards grazing angles and its inability to represent anisotropic surfaces, Lafortune *et. al* developed a new generalized Phong model LAFORTUNE *et al.* [7]. As seen, Phong employs the classic cosine lobe:

$$f_s(x, \omega_i, \omega_o) = k_s(\vec{R} \cdot \vec{\omega}_o)^\alpha S_{rgb}$$

The parameters that define shape and size of the lobe are k_s and α . Expanding the above dot product and adding the parameter vector $C = \{C_x, C_y, C_z\}$, we have:

$$f_s(x, \omega_i, \omega_o) = k_s(C_x R_x \omega_{ox} + C_y R_y \omega_{oy} + C_z R_z \omega_{oz})^\alpha S_{rgb}$$

Notice that C controls the lobe’s properties. To obtain an anisotropic reflection, we need only that $C_x \neq C_y$, otherwise, we obtain an isotropic reflection.

Despite its limitations, we chose the Phong model due to its simplicity, specially desirable during the optimization steps in our algorithm.

2.2 Color Spaces

Human eyes possess cones that are able to respond to light of different wavelengths, that’s how we perceive color. Since we have three types of cones, only three variables are necessary to represent all visible colors. Depending on how we define these variables and what they represent, there is a large range of possible spaces to create. In this section we study four of them which are used in our work.

2.2.1 RGB

One of the most common color spaces used, and default in OpenGL and GLSL shading language, is the RGB color space. It is an additive system, where we can obtain a color by simply adding two others. The resultant perceived color is a linear combination of the basic colors: *Red*, *Green*, *Blue*.

Usually the system domain may vary depending on the application. It is common to see photographs with integer values of pixel colors ranging in the interval $[0, 255]$. In this case, the vector $(0, 0, 0)$ represents the black color and white is $(255, 255, 255)$. In GLSL language and in our shading model, the values are in the interval $[0, 1]$, where black is represented the same way, but white is the vector $(1, 1, 1)$.

2.2.2 sRGB

The RGB space is device dependent, which means that the same RGB value in a monitor may not be the same perceived color from a different monitor. This happens

because monitors non-linearly convert voltage into light intensities NGUYEN [8]. The conversion is a gamma exponentiation, where gamma usually varies from 2.2 to 2.4. To prevent this effect we perform a gamma correction to our output color. Let C be the resulting perceived color from the shading model above, we would actually send to the monitor a gamma-corrected output L' :

$$L' = L^{1/\gamma}$$

In order to not have the burden to perform a color calibration regarding each device, HP and Microsoft created a standard RGB space called *sRGB*.

It is also common that most photographs taken with digital cameras are represented in sRGB, and therefore are in a non linear space. Additions in non linear spaces do not behave as expected leading, which means that we should first bring them to a linear space by exponentiating to a power of γ .

Fortunately, for both rendering color and reading photographs, OpenGL provides convenient solutions with the support of sRGB textures and framebuffers. The use of the sRGB color space greatly improved our results.

2.2.3 CIE XYZ

The RGB color space was created by a set of conducted experiments in which people would match up a given color by adjusting the weights of the three basic colors. In order for the results to match, some of the weights needed to be negative. The CIE proposed then a set of three different lights sources that were not monochromatic called X , Y and Z . They allowed for color-matching with only positive weights. The conversion between the XYZ model and the RGB depends on the white reference point used, we show the matrix of conversion below for the D65 reference white [9]

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.55648 & -0.204043 & 1.057311 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

And the inverse conversion:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

2.2.4 CIELAB

The CIELAB is a perceptually organized color space, which means that two colors that are perceptually similar are close in distance, while in the CIE XYZ model this would not necessarily happen. This color space is suitable when we want to perform

operations based on how similar two color are. In our work, we use this color system when comparing a pixel from a photo and the rendered color.

The coordinates of the system are: L , which represents lightness, and a and b , which are color dimensions. The conversion from the XYZ model is the following:

$$\begin{aligned} L &= 116f\left(\frac{Y}{Y_w}\right) - 16 \\ a &= 500\left(f\left(\frac{X}{X_w}\right) - f\left(\frac{Y}{Y_w}\right)\right) \\ b &= 200\left(f\left(\frac{Y}{Y_w}\right) - f\left(\frac{Z}{Z_w}\right)\right) \end{aligned}$$

where the color (X_w, Y_w, Z_w) is the reference white color:

$$f(x) = \begin{cases} x^{1/3} & \text{if } x > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3} \left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{otherwise} \end{cases}$$

The inverse conversion is:

$$\begin{aligned} Y &= Y_w f^{-1}\left(\frac{1}{116}(L + 116)\right) \\ X &= X_w f^{-1}\left(\frac{1}{116}(L + 116) + \frac{1}{500}a\right) \\ Z &= Z_w f^{-1}\left(\frac{1}{116}(L + 116) - \frac{1}{200}b\right) \end{aligned}$$

where:

$$f^{-1}(x) = \begin{cases} x^3 & \text{if } x > \frac{6}{29} \\ 3\left(\frac{6}{29}\right)\left(t - \frac{4}{29}\right) & \text{otherwise} \end{cases}$$

Since there is no direct conversion from RGB model to CIELAB, we first perform a conversion to XYZ space and then from CIELAB, and similarly for the inverse.

2.3 Camera models

In our work we deal constantly with two different camera structures. One is the physical camera of the photographs. This real world camera possess properties such as lens aperture, focus and distortion. Besides these intrinsic characteristics, we are also interested in external settings, such as its position and orientation with respect to the object.

The other camera we deal with is the virtual camera in our rendering system. This is an OpenGL camera model and its properties are usually: field of view, near and far clipping planes. Our virtual camera is also affected by the view matrix, which describes its position in world space when rendering the scene.

When dealing with such two distinct models, we need to consistently handle our transition between both so we can assure the coherency and effectiveness of the technique. The subsequent subsections will discuss each model.

2.3.1 Projective camera

The projective camera we study in this section is the finite projective camera that maps points from the projective space \mathbb{P}^3 to \mathbb{P}^2 . Points in \mathbb{P} are represented using homogeneous coordinates, *i.e.* if $\mathbf{X} \in \mathbb{P}^3$, then $\mathbf{X} = (X, Y, Z, W)^T$. The projection maps points in 3D space onto the called *image plane*. We say that the projection transforms points $\mathbf{X} \in \mathbb{P}^3$ to $\mathbf{x} \in \mathbb{P}^2$:

$$\mathbf{x} = P\mathbf{X}$$

We consider the following operations in the camera coordinate system, which may be aligned with the world coordinate system. Let the camera center C be the origin of its system, and the image plane $Z = f$, where f is the camera's focal lens. The ray with origin at C that passes orthogonally through the image plane is the principal axis. The point of interception is called principal point p . Considering cameras with CCD sensors, it may occur that the pixels are not squared. In this case we have scale factors m_x and m_y , which represent the number of pixels per unit distance in each direction. The coordinates of C are described as its distance from the world system's origin and we use a 3×3 rotation matrix to represent the camera's orientation. Finally, the last parameter we have to take into account is the skew factor s , which is zero for most cameras. Let K be the following matrix:

$$K = \begin{pmatrix} m_x f & s & x_0 \\ & m_y f & y_0 \\ & & 1 \end{pmatrix}$$

We call K the *camera calibration matrix* which contains the camera's internal parameters. Camera's external parameters are the rotation matrix R and its center C . Often, we may prefer to not make its center explicit and use the called *translation vector* t instead, where $t = -RC$. We can represent the camera projection matrix as a product of its internal and external parameters:

$$P = K[R|t]$$

In our system, during the calibration set-up we calculate the camera's internal and external parameters. For more details about the projective camera and other camera models, please refer to HARTLEY and ZISSERMAN [10].

2.3.2 OpenGL camera

Within OpenGL's conceptual rendering model, there is a series of transformations by which the scene goes through until its result is shown in an output device. We will discuss three of these transformations in this section: Model, View and Projective.

Model transformation is the transformation which will be applied to our scene that changes its location in relation to space, these consist of translations, rotations and scaling. The view transformation positions our camera in the desired point to render our scene, much like positioning a real world camera to take a photograph. Model and View transformations are usually combined into a single one, called ModelView. The view transformation is the same as the camera's external parameters of the general projective camera.

In OpenGL two types of camera are used: Orthographic and Perspective. For our purposes, we will only talk about the latter. Defining a perspective camera in OpenGL requires to set a viewing frustum defining the region in space which will be rendered. We define this frustum by specifying a vertical field of view θ , an aspect ratio α , and, for practical issues, a near(n) and a far(f) plane. OpenGL camera will project points that are inside the viewing frustum and between the near and far planes to a cube of size two and centered in the origin. Let $c = \cot(\theta/2)$, the projection matrix is the following:

$$P = \begin{pmatrix} c/\alpha & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

It can be noted that the OpenGL camera model is also a projection camera, however a little more restrict. In order to further understand the OpenGL camera model and its transformations, please refer to SHREINER *et al.* [11].

2.3.3 Conversion between different models

From the general projective camera model to the OpenGL camera we use the conversion used by LI [12]. Since their external orientation is the same, we only need to convert the projective camera model to OpenGL's perspective projection. This is done by obtaining a field of view θ and aspect ratio α from the camera's calibration matrix. The aspect ratio can be simply obtained by the image's size and its pixel scale factors m_x and m_y :

$$\alpha = \frac{\text{width } m_x}{\text{height } m_y}$$

The field of view requires to know the camera's focal length, but can also be easily calculated:

$$\theta = 2\text{atan}\left(\frac{\text{height}}{2} / \frac{f}{m_y}\right)$$

The inverse conversion isn't performed in our system, so we leave it as a further reading of the work by Li.

Chapter 3

Related Works

There are some approaches used in estimating BRDFs of real objects: controlled lighting solutions which use various images with fixed viewpoint but varying light directions are described in Section 3.2; techniques that try to perform the acquisition in a general lighting environment are discussed in Section 3.3; and some of them design specific devices for this task (Section 3.4). We start our literature review discussing works performing Color Mapping, which can be considered a BRDF approximation for purely lambertian objects. Since our work by default also performs color mapping, we will talk about some recent works of the area.

3.1 Color mapping

Considering the advances of modern 3D scanners and digital cameras, obtaining mesh datasets consisting of millions of triangles and photo datasets easily surpassing 50 millions pixels is not uncommon. Taking into account this technological progress, the work from CALLIERI *et al.* [13] propose to perform an efficient color mapping while coherently exploring this space of large data. The core of their application is a weighting mask that attributes quality values to pixels based on a set of criteria.

These criteria are encoded in the following masks: Angle, Depth and Border. The first works similarly to the lambertian illumination, the closer the surface's normal is to the viewing direction, the higher is the pixel weight. The depth mask takes into account the distance from the surface and the camera, the rationale behind is to approximate the ratio pixel/surface. The last basic criterion measures how far is a pixel from borders, whether they are image borders or discontinuities in the depth map. Their system also allows for case specific masks, such as stencil or focus masks.

After the masks are computed, applying them to the mesh is straightforward. Given a point in the object's surface, it is easy to know in which images this point is visi-

ble. The final color of the point will be a simple weighted average of the image pixel multiplied by the masks for each image.

Using a handheld recorded video from a consumer depth camera ZHOU and KOLTUN [14] present an optimization approach for mapping color images onto a mesh. Their proposal is based on the current increased availability of consumer depth cameras in the market. On one hand, non-professionals can now create highly accurate geometric models, on the other, color images produced by RGB-D cameras produce optical distortions not present in regular cameras.

In their optimization algorithm they are iteratively trying to optimize the color $C(p)$ of the point p and an extrinsic matrix T_i that maps all vertices p_i for each image I_i . Due to imprecise geometry, inaccuracy of camera localization and optical distortions, there is a non-rigid correction for each image represented as a deformation function F_i . Their objective function is:

$$E_c(C, T, F) = \sum_i \sum_{p \in P_i} (C(p) - \Gamma_i(F_i(u(g(p, T_i))))))^2$$

Where g is just the rigid transformation of the matrix T_i applied to p , $u(g_x, g_y, g_z, g_w)$ is the projection onto the image plane of I_i and $\Gamma_i(u_x, u_y)$ is the color evaluation for coordinates (u_x, u_y) . Since their optimization is realized with greyscale images, the value returned by $\Gamma_i(u_x, u_y)$ is the bilinear interpolation of the greyscale intensity of the pixel (x, y) .

The total number of variables is considerably large: $m + 720n$, with m equals to the number of vertices and n is the number of frames, hence they employ an alternating optimization scheme where sometimes C is optimized while keeping T and F fixed and vice-versa. The optimization of C leads to a linear least-squares problem with a closed form solution. And the optimization of T and F leads to solving n linear systems with 720 variables each.

This work presents fine results, although some effects such as specular highlights or moving shadows can affect the final result. However it does not solve the full reflectance acquisition problem for non lambertian objects.

3.2 Controlled Lighting

LENSCH *et al.* [15] present a similar approach to our method, a fitting process using only a professional digital camera, a reflecting sphere and a dark room. Unlike our approach where for each viewpoint we only take a set of photos with different light directions, they perform the following acquisitions for every viewpoint: two images to detect light source position, one image to register the 3D model with the images and a high dynamic range image with varying exposure time. They use the Lafortune BRDF model in their approach.

For their fitting process, a data structure called *lumitexel* (L) is created for every visible surface point, and for each L there is a list of radiance values for each image where it is visible. The lumitexels are created by projecting the model’s triangles in each photo, picking the image where its projected area is the largest, called I_{best} , then for each pixel p_i of I_{best} the position \vec{x}_i of the model is calculated. After assembling all lumitexels, a process called split-recluster-fit begins.

An initial cluster is created by selecting a given number of lumitexels and a singular BRDF fit is made using a Levenberg-Marquadt optimization process. If the model is composed of more than one type of material, as typically is the case, a single BRDF model f won’t suffice. A split step divides f into f_1 and f_2 and the lumitexels are reclassified accordingly. For both newly generated clusters, a fitting process is repeated. This process repeats until a given number of materials are created. When more than two clusters have been created by successive splits, all initial lumitexels are distributed throughout the clusters.

As can be noted, different points in the model may belong to the same cluster and possess the same BRDF f_i , which does not recreate a true spatially varying BRDF. Thus, for each cluster a set of basis BRDFs are created and every lumitexel in that cluster stores weights for a linear combination of them. To calculate the set of basis BRDFs, a principal function analysis is used.

Due to the clusterization it may happen that small areas may not be represented so well since the number of lumitexels is so small compared to the cluster they belong, causing a misrepresentation of the diffuse color. Usually some of the differences in the results are spotted in highlights, they are not strong enough or not in the correct shape. In our results since every vertex has its own BRDF components we can approximate them independently and sample even small areas, and highlight shapes are usually well represented.

LENSCH *et al.* [16] extended the previous work by changing the calibration of the light source position and estimating normal maps in order to refine the mesh’s geometric details. They recalculate a better normal of a surface point by minimizing the error of the pixel value and the rendered reflectance. Results have demonstrated an average decrease of approximately 60% of the root mean squared error with the use of normal maps. However, this introduces some artifacts on some parts of the mesh, whether they are regions where not enough samples were acquired or regions with high reflectance interreflections. Although this method produces better results, it still maintains the same problems of the previous technique.

Polynomial Texture Maps (PTM) MALZBENDER *et al.* [17] are already a established form of representing real world objects BRDFs. Unlike a traditional texture mapping where a texel stores the color information, coefficients of a biquadratic polynomial are also stored:

$$L(u, v; l_u, l_v) = a_0(u, v)l_u^2 + a_1(u, v)l_v^2 + a_2(u, v)l_u l_v + a_3(u, v)l_u + a_4(u, v)l_v + a_5(u, v)$$

Where L is the resultant luminance at coordinates (u, v) and (l_u, l_v) are the projections of the light vector onto that coordinates. Given N images, they build a $n \times 6$ linear system and solve for the variables a_i using singular value decomposition. Their main observation is that for different light positions, the pixel’s chromaticity remains fairly constant while the luminance may largely vary. Defining the chromaticity of the pixel as $(R_n(u, v), G_n(u, v), B_n(u, v))$, the final color of the pixel (u, v) is:

$$\begin{aligned} R(u, v) &= L(u, v)R_n(u, v) \\ G(u, v) &= L(u, v)G_n(u, v) \\ B(u, v) &= L(u, v)B_n(u, v) \end{aligned}$$

They also show that it is possible to apply filter behaviors on the PTM and some lighting models such as anisotropic surfaces and fresnel effects can be modeled. Although originally designed for texture mapping, there are also some 2D applications for which they are suitable, for example contrast enhancement and depth of focus. Their paper also presents the original approach to the technique called *Reflectance Transformation Imaging* (RTI).

RTI is a computational photographic method to acquire a 2D image which encodes 3D reflectance information of real world objects. RTI images are a form of texture mapping which allow the change of light direction in an interactive way. By capturing a series of photos with controlled lighting conditions, it is possible to extract the normal information of the surface. Besides from an interactive relighting, they permit the enhancement of surface details through a series of filters, such as: diffuse gain, specular enhancement, unsharp masking and multi-light detail enhancement. This improvement of the surface details is of great interest during the study of ancient writings and inscriptions. For this matter, RTI is extensively used in the Cultural Heritage field.

The work of GOLDMAN *et al.* [18] tries to recover at the same time the shape and the spatially-varying BRDFs of objects. Following the observation that many real world objects can be decomposed into a small number of materials, their model is a resultant composition of other smaller substances, and as an input, the number of fundamental materials is provided by the user. The shading model is the isotropic Ward. As a limitation, their work does not model effects such as cast shadows, inter-reflections, transparency and translucency.

Their algorithm works in five steps. The first is calibrating the light intensity and recovering its direction. In the second, they initialize the normal maps using a lambertian photometric stereo, already giving an initial estimate of the diffuse albedo. We perform similar initialization in our work. The next steps are the proper optimization procedure that are repeated until convergence: Optimize BRDF parame-

ters while fixing the normals; computing surface normals and material weight maps; and finally, reconstructing the 3D surface from the newly calculated normals.

Since both the shape extraction and material estimation are done alternatively during the optimization procedure, their overall fidelity is lower than in works where each task is done separately. Also the procedure is subject to overfitting, which usually happens when in most of the images a given pixel is in highlight.

Following the same approach as above the work of ALLDRIN *et al.* [19] recovers both shape and BRDF simultaneously, but instead of using a parametric reflectance model, it uses a bi-variate approximation of measured isotropic BRDFs. It basically does steps similar to the previous work, but since there is no parametric BRDF model, they argue that their work can represent a broader number of materials. However they do not deal with self-shadowing which limits their objects to convex shapes.

3.3 General Lighting

In PALMA *et al.* [20] a statistical method for the estimation of Spatially Varying BRDFs is provided. Unlike our method, their approach is based in video sequences with fixed but general lighting conditions. Similarly to our work, the lighting model used is Phong.

The initial step is to align the video frames to the mesh in order to have a set of color samples projected by each frame, this step is done using PALMA *et al.* [21]. Then, for each texel, their luminance values are separated according to a threshold, such that specular ones are used to estimate environment map and diffuse samples are used in the computation of the diffuse color. Using the object as a probe, without the need of reflecting spheres, they can produce a probability map depicting the light in the environment.

During the diffuse color estimation, the environment map is used to calculate the probability $s_{x,y}$ that the sample in a given frame has a specular behavior. To estimate the diffuse color, a threshold $d_{x,y}$ is calculated using the specular weight $s_{x,y}$ and discarding all samples whose luminance value is higher than $d_{x,y}$. The color is simply a weighted average of the color samples.

The final step of the algorithm consists in estimating the specular behavior of the object. Firstly, it needs to estimate the position and shape of the environment main light sources. Using a median cut algorithm proposed by DEBEVEC [22] it is possible to approximate the environment map with a set of 4096 directional lights, afterwards all clusters are created by generating connected components of the lights in the map. From the clusters it is possible to extract its centroid and local distribution. Then for each texel they detect a temporal luminance peak and extract two

samples c_{min} and c_{max} which correspond to the minimum and maximum differences from the diffuse color respectively. In a general lighting environment, each luminance peak is affected by the many light sources, if more than one exists, however there is a probable main light source that influences the most each peak. The main source light \vec{l} that should create each peak is calculated by selecting the light source whose Mahalanobis distance to the reflected view direction \vec{r}_{max} is the lowest. The specular parameters k_s and η are simply calculated by a resulting system of equations when computing the difference of c_{max} and c_{min} with the pixels luminance.

A user assisted clustering process is performed next, since in the video some object areas may have not been appropriately specularly sampled. An initial automatic step is performed by spreading seeds in areas where the diffuse color is more uniform and then applying a growing region for each seed. A new point is added to the cluster if the euclidean distance of its color and the mean cluster color in the CIELAB space is smaller than a threshold defined by the user. The average of the specular parameters is calculated for each cluster.

Some limitations are presented in this work due to the input data and the Phong model, for example, it presents trouble in recovering highlights, either not fully capturing some highlight areas or resulting in a different appearance than the photos. It may also present blur effects on small specularities. In addition, the clustering step may sometimes require too much manual intervention.

The work presented by DONG *et al.* [23] also tackles the unknown lighting conditions using a video, however instead of a general path as done by PALMA *et al.* [20], the object is rotated around its axis. As BRDF model they chose the isotropic microfacet model ASHIKMIN *et al.* [24]. As an assumption of the object, they ignore self-occlusions which limits their objects to convex shapes, whereas in our work self-occlusion is dealt with.

They need to find for every surface point x diffuse $k_d(x)$ and specular albedos $k_s(x)$, but instead of a specularity parameter, they estimate a Normal Distribution Function $D(\omega_h; x)$. Their objective function minimizes the squared error of the observation in the video frame from the outgoing radiance of the surface point x for all time steps t :

$$\operatorname{argmin}_{k_d, k_s, D_x} \sum_t \sum_x \|I(\omega_o; x, t) - L(\omega_o; x, t)\|^2$$

In order to lower the computational cost of this minimization, they alternate between estimating the normal distribution $D(\omega_h; x)$, the diffuse and specular albedo, and the incident lighting. For the lighting estimation they assume that natural incident lighting is sparse in the gradient domain and use the temporal gradient of the trace, which is the temporal variation in the appearance of a surface point. Since they employ various non-linear steps during the estimations above, there is no guaranteed

convergence rate or global minimum.

3.4 Devices

HOLROYD *et al.* [25] designed a complex coaxial optical scanner capable of synchronously acquiring shape and spatially varying reflectance. Their device consists of a pair of assemblies each containing a coaxial camera and a light source. The reflectance information is acquired by the modularization of light by a high-frequency sinusoid pattern. This sinusoidal illumination allows separating direct reflections from subsurface scattering, diffuse interreflections and other indirect effects.

Their acquisition step consists of leaving one arm in a fixed position as the other moves around the object. For each arm position, four stacks of images are taken, each image consisting of M frames, typically $M = 10$. Subsequently, there are a number of steps to obtain the geometric model and the reflectance. First, the amplitude and phase maps of the sinusoidal illumination are calculated, the amplitude map is proportional to the amount of light reflected from the surface. Next, pose refinement and depth maps calculations occur. For the depth map calculation they use different phase maps of each view and perform a triangulation to extract the surface depth. The direct components of the sinusoidal illuminations seem suitable to use as BRDF samples, but first a correction factor should be applied so they can be used, this is the final step of their pipeline.

For the BRDF acquisition, they segment the mesh into clusters based on the diffuse albedo using the k-means algorithm. Like the idea of LENSCH *et al.* [15], each cluster has a set of basis BRDFs and each vertex of the cluster has weights to linearly combine them. The model used is Cook-Torrance. The results presented are correct within 5.5% of the reference data, but it still poses some problems. The first and most clear is the restriction on the object's size and position, the device does not allow for large objects. The second is the complexity of such device, it is not suitable for practical acquisitions and requires a controlled environment. They also have a limitation of acquiring a BRDF under a global scale that cannot be easily solved without information of light power. We overcome this problem using a color chart.

WILLEMS *et al.* [26] created a minidome with the goal of digitizing cuneiform tablets. Their dome with a radius of 50cm consists of a single camera placed on top and 256 white power LEDs positioned on the knots and edge centers. The entire digitization process is programmable and fully automatic. The geometry reconstruction is performed using Photometric Stereo to recover the object's normal map and the depth map can be calculated by a normal map integration.

The initial purpose of the minidome was the digitization of historical artifacts and

subsequent texturing of the object’s color. However, it can be also used to perform an RTI and extract the object’s reflectance information. Moreover, it can be used to create a dataset for other BRDF algorithms, some of the datasets used in this dissertation were captured in a minidome.

One special work with the usage of a dome is from SCHWARTZ *et al.* [27], currently producing state-of-art results. Their dome consists of 151 DSLR cameras taking HDR sequences with resolution of 12 Megapixels, one LED-Projector mounted on a tripod placed at five to eight different positions, with a resolution of 840x480 and projecting 38 different patterns. Instead of calculating BRDF functions and assigning them to clusters or model’s points, they create a *Bidirectional Texture Function*(BTF) DANA *et al.* [28].

BTF is a six-dimensional texture image parameterized by illumination and viewing angles, and the surface point. BTFs are similar to SVBRDFs, but are more suitable to objects that have significant mesostructure. The BTF also capture effects as self-shadowing, occlusion, inter-reflections, subsurface scattering and color bleeding. In their dome, not only they are able to extract the reflectance behavior of the object, they also reconstruct the mesh in the same process. This is their first step in their digitization pipeline, in which they perform a structured light approach for geometry acquisition using the projector described above. Next, they perform a mesh parametrization and generation of a texture atlas. Since the photos taken are in the global hemispheric camera setup Ω_g and the texture works in a local hemisphere Ω_l , there is a need to resample the data. Finally the BTF dataset has to be compressed to enable efficient storage and rendering.

The whole pipeline can be fully automatized and requires minimum user intervention, although the authors don’t specify if the projector’s movement and positioning is automatically done. With around 200.000 photos taken, the whole process takes about 36 hours. Although, the initial size of the uncompressed datasets can be around 500 *GB*, it can be reduced to approximately 1.5 *GB*. Overall, it is not a computationally cheap system. One considerable limitation of this work is the size and position of objects that can be used, since they need to be placed inside the dome, whereas our approach has no limitation with regards to the object’s dimensions. Furthermore, in our system a model with 1 million vertices only needs an extra storage of 24 *MB*, which poses no problem to current graphics cards even in a scene with multiple objects.

3.5 Example-based

The work of REN *et al.* [29] discusses how to acquire a SVBRDF using common and inexpensive objects such as: a smartphone camera, a handheld linear light source

and a BRDF chart. This last item is similar to the color chart used by our system but instead of colors, materials with known BRDFs are printed. With the purpose of an easy and fast acquisition system, they show reasonable results with videos of only thirty seconds. For the acquisition, they place the camera at an angle of roughly 40 degrees 50cm away from the material, the light is positioned one meter above the surface and kept parallel to a flat surface. The light is translated horizontally in only one axis direction.

Using the BRDF chart’s materials as representatives, the BRDF of a given point is the sum of a normalized lambertian BRDF and a linear combination of specular representatives. As a contribution, their algorithm can also select from a BRDF database a basis of materials to compose the chart. An initial alignment is performed so that the center of the frame sequence is the highlight peak of the material reference. They then perform an optimization to minimize the error between the pixel and rendered color. For cases of bumpy surfaces, an extra light pass is required, but with respect to the other axis direction. This allows for a recovery of the surface’s normal variation.

The technique described presents convincing results within a small acquisition time. However, it has severe limitations. It only recovers the reflectance of flat surfaces, and even those with small height elevations can cause trouble. The instruments only allow for small objects whose data acquisition needs to be done at a short distance. Design and fabrication of a BRDF chart is a nice feature but impractical. If the choice of representative materials is not good enough, the material’s reflectance will not be appropriately captured, because only a BRDF that is a linear combination of the chart’s BRDFs can be extracted.

3.6 Summary

We discuss several works that try to solve the problem of reflectance acquisition, using a range of diverse approaches. Some capture photos with light in a more controlled way, others attempt to acquire videos in a general lighting environment, and there are those that create special devices only for this task. Although many present convincing results, all of them still imposes some limitations. Either they cannot appropriately capture highlight areas very well, create clusters of BRDFs not capturing a true spatially-varying BRDF. Many that present excellent results for high specular surfaces and even anisotropic objects can only capture flat or convex shapes. There are even those that can faithfully capture an object’s BRDF with no regards to its geometry, however, they limit the object’s size and position. In conclusion, to the best of our knowledge, there is no current work that solves the

problem of acquiring the SVBRDF of a general object with no restrictions about its size and position. More importantly, none of them are able to present immediate feedback to the user about the acquisition quality. Therefore, we propose a system that tries to fill this gap.

Chapter 4

Algorithm

The goal of our algorithm is to approximate the BRDF of an object given its mesh and photos from a fixed viewpoint but with different light positions. In order for our method to work we expect the mesh to be aligned with the real world object in the photo (see Figure 4.1). We project the mesh’s vertices onto each image and read the pixel color information. All calculations are done independently per vertex in the fragment shader where we can determine the vertex’s screen position and retrieve the corresponding photo’s pixel value. We use this knowledge to fit a Phong model for every vertex:

$$f(L, E) = k_d(\vec{N} \cdot \vec{L})C_{rgb} + k_s(\vec{R} \cdot \vec{E})^\alpha S_{rgb}$$

Notice that we changed ω_i and ω_o from the BRDF equation to L and E respectively to simplify the notation further on. The unknown parameters to be computed for each vertex are: the diffuse coefficient k_d , the specular coefficient k_s , the shininess α and the diffuse color C_{rgb} . An overview of the whole pipeline is shown in figure 4.2.

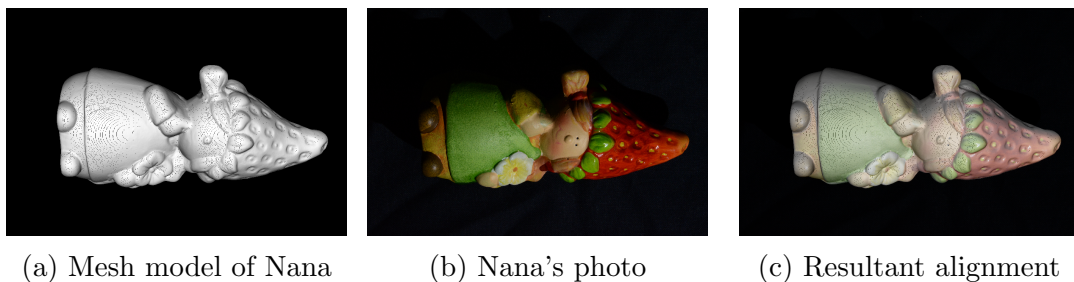


Figure 4.1: Mesh-Image alignment for Nana

Our input data is initially the model’s mesh and a set of photos with respective light directions. We may acquire all the photos interactively in our acquisition procedure or work with a prepared set. The acquisition environment consists of: object whose BRDF will be extracted, a reflecting sphere to acquire light direction and a

color chart so we can correct the color information in our photos. Figure 4.3 gives a clear example of our acquisition environment.

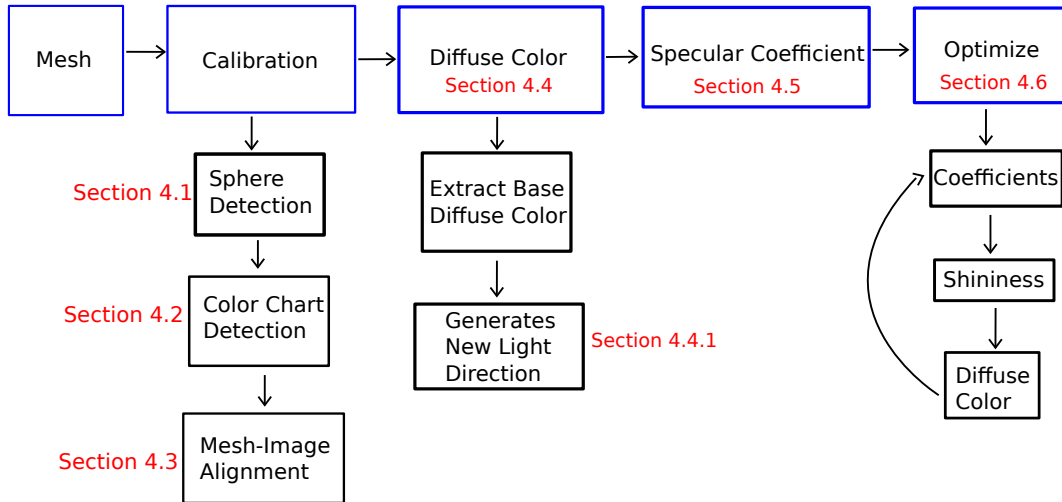


Figure 4.2: There are basically four steps in our system: calibration, basic diffuse color estimation, specular coefficient estimation and optimization cycle.

If we acquire the photos interactively, we start our algorithm with the calibration procedure. This includes locating the sphere (Section 4.1), retrieving the color chart’s values (Section 4.2), and computing the mesh-image alignment (Section 4.3). We assume that for a given viewpoint only the light direction changes. When changing the camera’s position a recalibration is required. After the calibration, we start the BRDF approximation *per se*. We compute an initial diffuse color for each vertex (Section 4.4). Following, there’s the computation of an approximate specular coefficient (Section 4.5). Finally, we perform an optimization on the BRDF’s parameters (Section 4.6).



Figure 4.3: Our typical acquisition environment consisting of the target object, a reflecting sphere to extract light direction, and a color chart for calibration purposes.

4.1 Sphere detection

As an initial step we try to automatically detect the reflecting sphere, using the Hough transform. However, we are prone to false positives and false negatives, depending on the parameters of the algorithm. To provide some control, the user can choose a given sphere among the detected ones or can manually position and adjust the radius. Notice that we only detect the circle of the sphere's projection, but it is enough for our purposes. Figure 4.4 shows an example of a billiard ball used as a reflecting sphere. Most shiny spheres are suitable to be used as reflecting spheres.



Figure 4.4: Demonstration of a sphere used to capture the light direction

The reflecting sphere is important to detect the light direction during the acquisition procedure. We use the highlight projected in the sphere and the algorithm by MUDGE *et al.* [30] to calculate the light direction. In order to simplify the solution we assume that the distances between both camera and light are large with respect to the object. We detect the highlight in the sphere, by simply picking the brightest pixel, this may be adjusted in the future to better deal with cases where the highlight becomes saturated, leading to many pixels having the highest possible brightness. Given the sphere with center C and radius r , and the highlight pixel H in screen coordinates we calculate the sphere's normal at the highlight position:

$$\begin{aligned} N_x &= \frac{H_x - C_x}{r} \\ N_y &= \frac{H_y - C_y}{r} \\ N_z &= 1 - \sqrt{S_x^2 + S_y^2} \end{aligned}$$

Since we assume the camera is distant enough, we can approximate our view direction \vec{V} as the z axis in the world space. To calculate the light direction \vec{L} , we need only to reflect the view vector with respect to the normal calculated above:

$$\vec{L} = \vec{V} - 2(\vec{V} \cdot \vec{N})\vec{N}$$

Figure 4.5 illustrates the process described above.

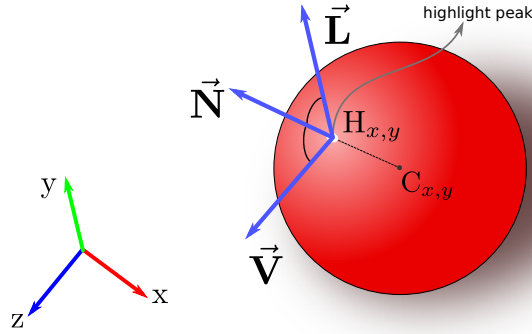


Figure 4.5: Calculating the light direction from the reflective sphere.

4.2 Color Chart detection

The Color Checker is a chart consisting of color samples designed to obtain a measure of how your photos deviate from the true colors. These charts can be varied in size and colors: some are grey-scale only or just a white reference color; there are charts with 140 colors or smaller ones with 24. The model used in our system is the one in the Figure 4.3 with 24 colors, they are displayed in a grid of 4×6 squares. Along with the chart, a list of the colors values are provided. What our system does is correct each photo so that the colors from the detected charts match the true colors.

The color chart detection consists merely in detecting the squares with respective colors in the image. We offer the option of automatic or manual detections. Automatically, it chooses one color, by default orange, and tries to find the area in the image that best matches pure orange. After that, it assumes all the squares are roughly the same size and by the known location of the orange color in the chart, finds the first square of the grid. Then, it is a simple matter of jumping to the other squares. Manually, the user needs to click on the square of the first color, then the algorithm expands an area that best matches the color brown of the chart and repeats the steps of finding the other squares as the automatic version.

For each color square C_{ij} , where the subscript ij indicates its position in the grid, we extract its average color \overline{C}_{ij} . Then, we need to apply a transformation for every color \overline{C}_{ij} so that it matches the true chart color C_{ij}^* . This can be achieved with a polynomial regression for each color channel.

4.3 Mesh-Photo Alignment

The effectiveness of our algorithm relies heavily on the alignment between the mesh and the photo. A misalignment may cause vertices to be attributed a BRDF slightly different, a totally wrong shading or even not be represented at all. Therefore, in order to achieve optimal results we need the best alignment possible, which can be considered as a drawback of our algorithm in comparison to others. However, this is a task extensively done by CH professionals, and is not usually seen as a problem. We perform the Mutual Correspondences algorithm developed by SOTTILE *et al.* [31]. The goal is to find the set of parameters $C = (\theta, \phi, \psi, t_x, t_y, t_z, f)$, where (t_x, t_y, t_z) represent the position of the camera, (θ, ϕ, ψ) its orientation and f its focal length. The algorithm tries to optimize these variables by using two measures: Mutual Information and Correspondence-based method.

Mutual Information is a measure of generic dependency between two random variables. The Mutual Information MI of images I_A and I_B can be calculated as:

$$MI(I_A, I_B) = - \sum_{(a,b)} p(a,b) \log \frac{p(a,b)}{p(a)p(b)}$$

Where $p(a)$ is the probability that a pixel of I_A gets value a , and $p(b)$ similarly for I_B . $p(a,b)$ is the joint probability of event (a,b) . The joint distribution can be estimated by evaluating the joint histogram of the two images and dividing the number of occurrences by the number of pixels. They also use information of normals, reflection and silhouette maps, and ambient occlusion. Then, the algorithm tries to find camera parameters that maximize the Mutual Information between the photo and the rendered mesh.

Image registration correspondence-based methods rely on manually or automatically setting correspondences on both the image and the model. The goal is to find the parameters that minimize the error between the projected 3D correspondences to the 2D image ones.

$$E(Corr, C) = \frac{\sum_{\forall corr_i \in Corr} \sqrt{(xp_i - x_i)^2 + (yp_i - y_i)^2}}{N}$$

Where (xp, yp) is the projected 3D correspondence, (x, y) is the image 2D correspondence, and N is the number of correspondences.

Statistic Image registration algorithms such as Mutual Information are precise and fast, but generally depend on the initial solution and rely on an accurate geometry for best results. Correspondence-based alignment algorithms are robust and independent from the initial position, but are slow and may require many accurate correspondences. In order to obtain an algorithm that is a robust, fast and invariant to the initial position, the Mutual Correspondence algorithm performs a mix of the two previous approaches.

In order to speed up the alignment process we can perform an initial arrangement of the mesh alignment and set some manual correspondences, three is the minimum amount, but four leads to more robust results. The algorithm then tries to maximize mutual info and minimize correspondence errors at the same time. Instead of maximizing the mutual info, it actually minimizes its negative value, so it performs only a minimization task. The balance between both goals is controlled by a weight given by the user. We empirically chose the weight of 0.9 for mutual info and 0.1 for correspondence errors.

Figure 4.6 illustrates the behavior of the algorithm, 4.6a the setup before the algorithm, where we roughly position the mesh at the photo's position and manually set pairs of correspondences on the image and the mesh. The circle dots are the image correspondences and the upside down triangles are the mesh correspondences. Notice in 4.6b how after the Mutual Correspondence algorithm the correspondences are closer and how the object is overall more aligned to the image.



(a) We manually set the mesh to a close position and set some correspondences. The green circle is the image correspondence, and the upside-down triangle the mesh correspondence. (b) Resultant alignment of the Mutual Correspondence algorithm, note the decreased distance between the correspondences and alignment of the palm leaves.

Figure 4.6: Mesh-Image alignment for the Palm tree model

4.4 Diffuse color

As a first approximation of the diffuse color we use the weighted average of the pixel information. We assume our object is lambertian, but only reflects half of the

received radiance. We also experimented setting it to reflect full energy and later recalculating this value, and the final results are the same. The advantage of setting it only to half is that intermediary results look better while the coefficients are not fully adjusted. It is worth mentioning that for each photo we render our mesh as seen by the light so we can mark which vertices are illuminated. Consequently, self-shadowing is not an issue at a small cost of only one extra render pass.

The color of a vertex v is the weighted average of all corresponding pixel colors C_i , where the weights are simply the product $\vec{N} \cdot \vec{L}_i$, and \vec{L}_i is the light direction of photo i :

$$C_v = \frac{\sum_{i \in U_{dp}} w_i C_i}{\sum_{i \in U_{dp}} w_i}$$

However, the pixel color does not represent the true object's reflectance value, it is affected by the light interaction, thus, we need to take this factor in consideration. Remembering Phong's shading equation:

$$f(L, E) = k_d(\vec{N} \cdot \vec{L})C_{rgb}$$

$$C_{rgb} = \frac{f(L, E)}{k_d(\vec{N} \cdot \vec{L})}$$

In our case, the resulting $f(L, E)$ is a pixel, for further simplification we will write it as P . In order to avoid unnecessary divisions and precision errors, we can simplify our equation as:

$$C_v = \sum_{i \in U_{dv}} w_i \frac{P_i}{k_d w_i}$$

$$C_v = \sum_{i \in U_{dv}} \frac{P_i}{k_d}$$

The set U_{dv} indicates the photos considered in our algorithm. These are photos where the vertex v is seen by the camera and by the light, *i.e.* is not in self-shadow. Highlights can induce a strong error in the basic color, therefore we do not take into account values where $(\vec{R} \cdot \vec{E})$ is too high, empirically we found 0.9 to produce good results, we define it as the specular threshold T_s . Also, when looking at grazing angles the colors tend to become darker and also negatively influence our algorithm, so we do not take into account pixels where the value of $(\vec{N} \cdot \vec{L})$ is too low, in our tests 0.5 proved satisfactory, defined as the diffuse threshold T_d . Photos that fit these criteria compose the set U_{dv} . Figure 4.7 shows the result for a given photo.

Notice that we highlighted areas that compose three of the cases above, obviously a vertex not seen by the camera can not be shown. Cyan areas contain vertices which are in highlight, vertices which are in self-shadow are marked by the orange area, and areas with low diffuse value are represented by purple. Note that these criteria are not totally precise since vertices that could fall in some of the categories



Figure 4.7: Cyan represents an area where the vertices are in highlight, orange delimits self-shadowing areas, and purple delimits the area of low diffuse value. All other vertices are considered for this camera-light pair.

above are still considered, for example, the green pixels inside the purple area. This may happen due to inaccurate geometry, however these minor errors are corrected later during the optimization process.

4.4.1 Generating New Light Directions

An important considered aspect is to use as few data as possible. Thus, one important question is: How can we determine that the number of light directions is enough to cover all vertices? The criteria above show us how to select the best light source positions so we can cover as many points as possible. The algorithm to determine the best light position for a given vertex is shown in Algorithm 1.


```

if ( $\vec{N} \cdot \vec{L} < T_d$ ) then
  | BestLight = Vertex's normal;
end
if ( $\vec{R} \cdot \vec{E} > T_s$ ) then
  | SpecularAngle =  $\cos^{-1}(\vec{R} \cdot \vec{E})$ ;
  | SpecularThresholdAngle =  $\cos^{-1}(T_s)$ ;
  | DifferenceAngle = SpecularThresholdAngle - SpecularAngle;
  | RotationAxis = the axis of rotation between  $\vec{E}$  and  $\vec{R}$ ;
  | /* This is the cross product if both vectors are orthogonal.
  |   */
  | NewReflection = Rotate(Light, DifferenceAngle, RotationAxis);
  | BestLight = Reflect(NewReflection, N);
end

```

Algorithm 1: Determining best light direction for a vertex

If a vertex has a low diffuse value, *i.e.* $(\vec{N} \cdot \vec{L}) < T_d$, then the best light direction is simply its normal direction. If a vertex is in highlight, we calculate the angle of its specular value, *i.e.* $\cos^{-1}(\vec{R} \cdot \vec{E})$. The best light direction for the vertex will be one where it is not in highlight, so we calculate the angle of the highlight threshold T_s . If we rotate the current reflection vector \vec{E} until its specular value is equal to the threshold value, then we have a good light direction. To do this, we need to find the rotation angle, this is the difference between the specular threshold angle and the specular angle calculated above. The rotation axis is just $\vec{E} \times \vec{R}$. The new light is simply the reflection of \vec{R}' with respect to the vertex's normal. Figure 4.8 illustrates this process.

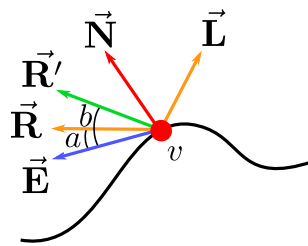


Figure 4.8: We want to calculate a new light direction that generates a specular value lower than T_s . The idea is to set the desired reflection vector \vec{R}' at a position in which $(\vec{R}' \cdot \vec{E}) = T_s$. To find the position \vec{R}' we need to express the difference $T_s - (\vec{R} \cdot \vec{E})$ in terms of angular distance. The angle a is $\cos^{-1}(\vec{R} \cdot \vec{E})$ and b is $\cos^{-1}(T_s)$. We find the difference between them and just rotate R ($b - a$) degrees with respect to $\vec{E} \times \vec{R}$. The best light is then just the reflection of \vec{R}' with respect to the vertex's normal.

This algorithm creates a new light position for every vertex which has not yet been covered by current light directions. As can be expected, for a large number of

uncovered vertices, especially during an early acquisition phase, we need to deal with a large number of new light directions. To select the best among them, *i.e.* the one which satisfies more uncovered vertices, we employ a greedy approach. However, in order to avoid testing all light positions against all vertices, which would give $\theta(n^2)$ tests, we employ a clustering approach. We partition our space into bins, determined by its longitude and latitude, by subdividing each axis in an equal number of parts, in our tests we used 36 divisions. The algorithm to determine best light is as following:

```

for All new light directions do
    Calculate latitude and longitude for current light;
    CurrentIndex = latitude*parts + longitude;
    Center a 3 x 3 grid around CurrentIndex;
    Calculate angular difference of current light with the representative light
    of each grid bin;
    BestIndex = index of bin with smallest angular difference;
    Assign new light to bins[BestIndex];
    Average lights of the bin;
end

```

Algorithm 2: Determining best light direction for all vertices

For all the lights, the algorithm determines their latitude and longitude. This gives us the index of its appropriate bin. However, it is possible that many lights in a bin are concentrated in a localized area and not distributed around its center. Therefore for each new light we calculate its bin's representative light as the average of all lights, which should give an iterative estimate of the bin's distribution. Due to this fact, it may happen that when a light is assigned to a bin, there is a neighboring representative light that is closer than the current bin's representative. Thus, for every new light we check the neighbor bins's representative and select the one to which it is closer.

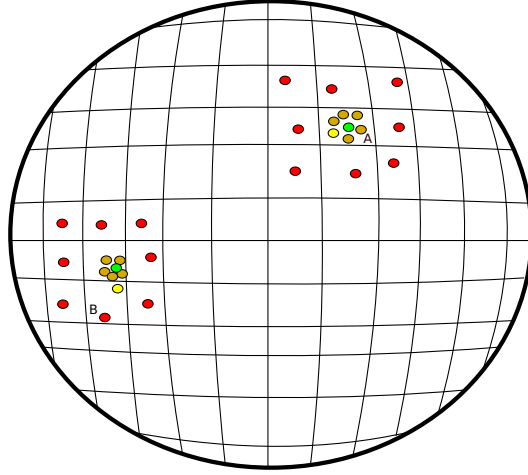


Figure 4.9: Spatial Bins example

Figure 4.9 demonstrates our algorithm and the sphere spatial division. The yellow circle is the current light being analyzed, green is the chosen representative light, dark yellow are the already determined lights that resulted in the green circle, and red circles are the grid bins farther than the chosen one. The case A in the figure illustrates the scenario where the current bin's representative is closer than all its neighbors' representative. On the other hand, case B shows the situation where the light is closer to a neighbor's central light than its current bin. In this case, it is better to assign the new light to the other bin.

Now we test all the representative lights for every vertex and select the one with best coverage. Instead of a quadratic algorithm, now we have $O(parts * n)$ complexity. In our experiments, this was $1296n$, which may be costly for a small n compared to n^2 , but presents much better results for a large n . During the early phases of the acquisition the number of vertices may be in the order of hundreds of thousands. However, the upper bound of $1296n$ is reduced by half, since we only acquire lights over a hemisphere, which gives us a $648n$ limit complexity. We analyzed the hemisphere coverage for our test cases (Figure 4.11). Table 4.1 shows the time results for the tests.

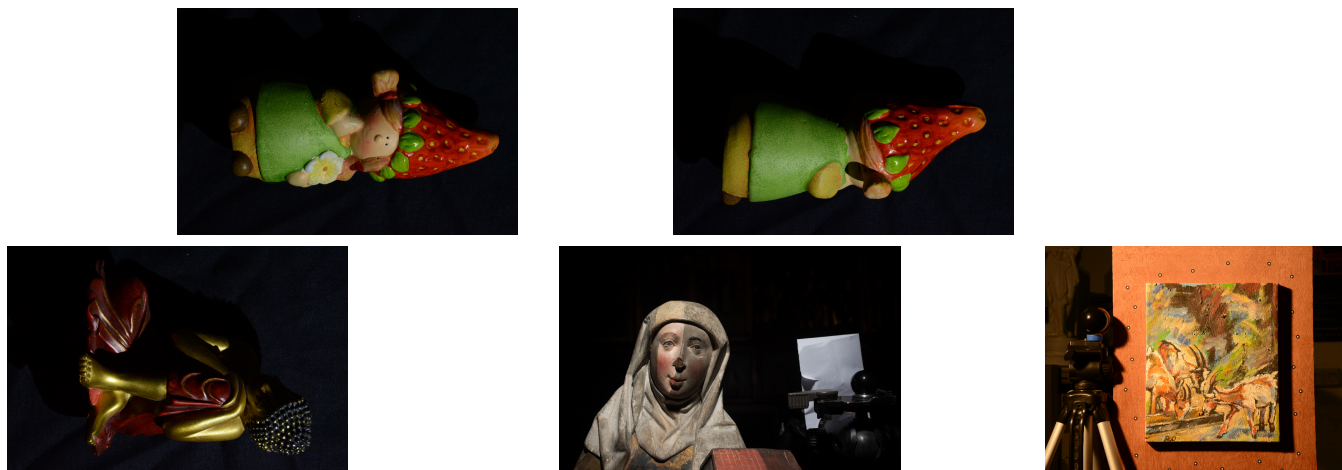


Figure 4.10: The datasets used in our work, top row, left to right: Nana, Nana Back. Bottom row, left to right: Buddha, Gertrud, Goats

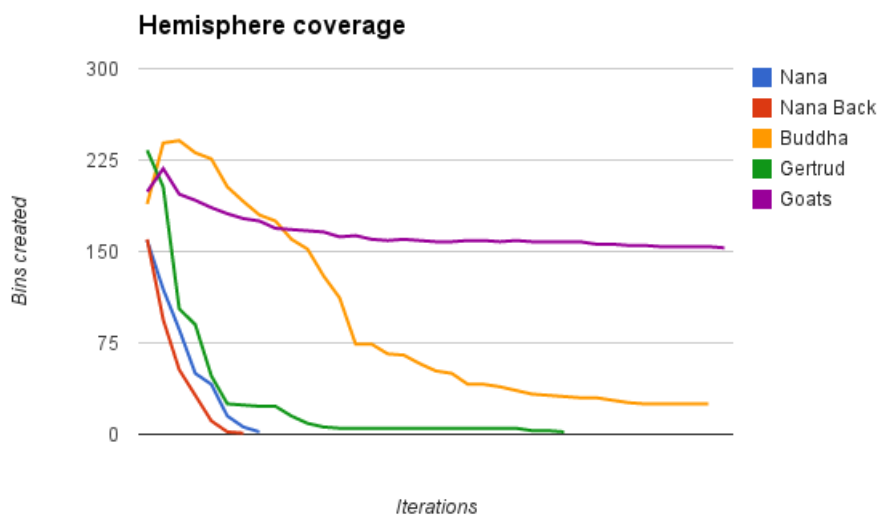


Figure 4.11: Hemisphere coverage during time. Note that after a few iterations, the number of bins is greatly reduced. The Goats case is rather odd, because although it seems a flat surface, it is composed of several bumps whose normals point to directions that are complicated to be captured by the current viewpoint. That it is way it never converges and more viewpoints are needed to fully cover the vertices.

Object	Total time(s)	Standard Deviation	Iterations(s)
Nana	110.053	0.368	8
Nana Back	91.442	1.652	7
Buddha	1478.012	65.263	34.4
Gertrud	1037.9553	33.858	26.1
Goats	2696.563	29.972	38

Table 4.1: We repeated the process 10 times for each dataset and took its average. Iterations are the number of lights, and consequently photos, needed to cover all vertices. For the Goats, 38 is the size of the complete dataset, since it never converged.

We tested with the following datasets: Nana, Nana Back and Buddha each having 115 photos, Gertrud has 37 and Goats 38 photos. The tests consisted in manually choosing the first photo and then automatically calculating the best next light direction and choosing from the dataset the photo with closest light direction until all vertices were covered or there were no more adequate light position available. In this case we would need a new camera position to cover all vertices. The iterations column in Table 4.1 indicates how many lights were needed to achieve full coverage. Notice how smaller is the actual hemisphere coverage from the limit of 648 directions. It is worth noting that due to float-pointing inaccuracies, some variations may occur. For comparison, after the first light the number of uncovered vertices are 114261 for Nana, 108750 for Nana Back, 272121 for Buddha, 1335117 for Goats and 1874735 for Gertrud. The brute-force algorithm yields the same number of light directions for each. Since it is unfeasible to measure time for a $\Theta(n^2)$ with a n as large as these, we did not perform the complete time test as above. Obviously, the first light direction may influence these results, but we can simply claim that they were in general conditions, *i.e.* none of them would result in a worst case scenario. For these tests we have used a predefined set of photos, but the algorithm for deciding the best light direction aligns with our goal of creating an interactive system, since we can decide on site which photos should be taken next, and more importantly, if there are any vertices that have not yet been covered. As explained in Section 1.1, this may be an issue when digitizing Cultural Heritage objects.

4.5 Specular coefficient

After capturing all photos and calculating a basic diffuse color, we go through the whole dataset estimating specular coefficients for all the vertices that have been *specularly covered*. In other words, vertices that have a specular value higher than the threshold T_s in at least one photo. For every photo we subtract the diffuse color

from the pixel and take out the specular value:

$$k_s^i (\vec{R}_i \cdot \vec{E})^\alpha S_{rgb} = P_i - k_d (\vec{N} \cdot \vec{L}) C_{rgb}$$

The superscript i in k_s^i means that this is the specular coefficient k_s calculate from the photo i . Let $(\vec{R}_i \cdot \vec{E})^\alpha S_{rgb}$ be \vec{S}_i , according to Moore-Penrose MOORE, PENROSE [32, 33] its pseudoinverse is its conjugate transposed vector divided by its squared norm:

$$S^+ = \begin{cases} 0^T & \text{if } S = 0 \\ \frac{S^*}{S \cdot S} & \text{otherwise} \end{cases}$$

We can now calculate k_s :

$$k_s^i = (P_i - k_d (\vec{N} \cdot \vec{L}) C_{rgb}) S_i^+$$

The final k_s is simply the average of all k_s^i obtained from each photo:

$$k_s = \frac{1}{|U_{s_v}|} \sum_{i \in U_{s_v}} k_s^i$$

Where U_{s_v} is the set of photos where the vertex v has a high specular value.

As we described initially, this is done for all vertices specularly covered. Nevertheless, this is trickier to acquire than the basic color. Let us assume a perfectly specular surface. To be able to see the light reflected in the surface, it needs to be perfectly positioned so that its reflection from the surface reaches the eye. For all other eye positions, it will not perceive the light. To acquire highlights, there is a small threshold in which the light and the eye must be aligned, for all other light directions and eye positions, another color is seen. Therefore, it is harder to acquire a highlight than the diffuse information.

Fortunately, we can use the same idea for acquiring the next best light direction used for diffuse photos. In fact, we employ the same scheme, we need only to change the algorithm for defining the best light position for a single vertex. Intuitively, we would only need to reflect the eye position with respect to the normal giving us the best specular light. However, we found out that this yields many light directions behind the hemisphere, which can not be captured. So, we try to bring them to the front of the hemisphere using the specular threshold. Algorithm 3 describes the

method illustrated in Figure 4.12.

```

if There is no photo where the vertex's specular value is higher than threshold
then
    IdealLight = reflect( $-\vec{E}$ ,  $\vec{N}$ );
    ThresholdAngle =  $\cos^{-1}(T_s)$ ;
    RotationAxis = axis(IdealLight,  $\vec{N}$ );
    NextLight = rotate(IdealLight, ThresholdAngle, RotationAxis);
end

```

Algorithm 3: Determining the best light direction for specular calculations

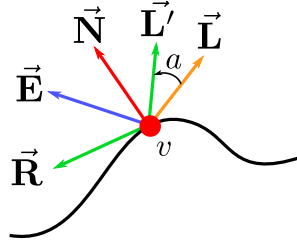


Figure 4.12: We want to calculate a new light direction that generates a specular value greater than T_s . The idea is to set the desired reflection vector \vec{L}' at a position in which $(\vec{R}' \cdot \vec{E}) = T_s$. To find the position \vec{L}' we need to express the value of T_s in terms of angular distance. The angle a is $\cos^{-1}(T_s)$. Then, we just rotate L a degrees with respect to $\vec{L} \times \vec{N}$. We actually subtract a small amount from a , because the new specular value should be higher than T_s so it can be considered a specular light direction.

Using the same spatial bin division, we can efficiently estimate a new best light direction for all vertices which need a good specular coverage.

4.6 Optimization cycle

Both previous steps are performed only once, mainly to obtain good initial estimates, however they are usually not optimal parameters. The next phase of our system is to optimize the parameters, but instead of improving all simultaneously, we perform three optimization steps: first the diffuse and specular coefficients, followed by the shininess parameter, and at last we enhance our base color approximation. As we improve our color estimation, the other parameters need to be reestimated, hence we begin a cycle repeating these three steps until convergence or a maximum number of iterations is achieved.

Our optimization scheme seeks to reduce the error between the pixel and the resulting color of the vertices. We perform a global optimization using all photos of the set U_v of the vertex v at once, where U_v is the complete set of photos where v is not occluded nor in self-shadow. Note that the optimization is global only on a per

vertex basis, we do not take into account neighboring vertices. We use an iterative estimation method where the function f to optimize is:

$$f = (P_i - C_i)^2$$

$$C_i = k_d d_i C_{rgb} + k_s s_i^\alpha S_{rgb}$$

Where we write $d_i = (\vec{N} \cdot \vec{L})$ and $s_i = (\vec{R} \cdot \vec{E})$ to simplify our future notation. Let us revisit the Gauss-Newton method. Consider we want to minimize the function $g(\mathbf{X})$, where \mathbf{X} is the parameter vector. We assume that $g(\mathbf{X})$ has a minimum value. We may expand $g(\mathbf{X})$ about \mathbf{X}_0 in a Taylor series to obtain:

$$g(\mathbf{X}_0 + \Delta) = g + g_{\mathbf{X}}\Delta + \Delta^T g_{\mathbf{X}\mathbf{X}}\Delta/2 + \dots$$

Where subscript \mathbf{X} means differentiation. We wish to minimize this value with respect to Δ , hence we differentiate it and set the derivative to zero, arriving at:

$$g_{\mathbf{X}\mathbf{X}}\Delta = -g_{\mathbf{X}}$$

Where $g_{\mathbf{X}\mathbf{X}}$ is the Hessian of g , *i.e.*, its entries are $\partial g / \partial \mathbf{X}_i \partial \mathbf{X}_j$. Now consider our least-squares minimization problem above, let us call $e = (P_i - C_i)$ and represent f as the function g described above, so we have:

$$g(\mathbf{X}) = e(\mathbf{X})^2$$

$$g_{\mathbf{X}} = 2e_{\mathbf{X}}^T e, \text{ and}$$

$$g_{\mathbf{X}\mathbf{X}} = 2(e_{\mathbf{X}}^T e_{\mathbf{X}} + e_{\mathbf{X}\mathbf{X}}^T e)$$

If we assume that function C is linear, the second term of $g_{\mathbf{X}\mathbf{X}}$ vanishes. Also notice that $e_{\mathbf{X}}$ is the Jacobian matrix J , so we can write $g_{\mathbf{X}} = 2J^T e$ and $g_{\mathbf{X}\mathbf{X}} = 2J^T J$. Using $J^T J$ as the approximation of the Hessian gives us the result:

$$J^T J \Delta = -J^T e$$

Assuming that $J^T J$ is invertible, Δ can be easily calculated. Which means that by approximating our solution with a calculated step of Δ , we can expect to minimize our error. Nevertheless, it may happen that this algorithm converges to a local minimum or does not converge at all, it is strongly dependent on our initial solution.

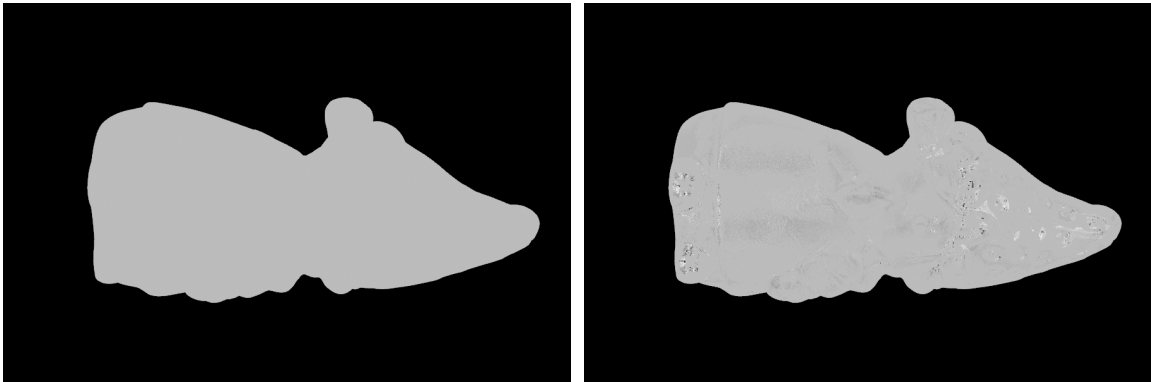
The predominant cost of each iteration of the Gauss-Newton method is calculating the Jacobian and the inverse of $J^T J$, which is a $x \times x$ matrix, where x is the number of parameters of \mathbf{X} . Since we divide our optimization in parts, we are bounded to a maximum of 3 parameters at a time, generating a matrix that is easily invertible, even in GLSL shaders. What remains is to calculate the Jacobian for every set of parameters. We show how to calculate the derivative in each following subsection. We divided the optimization in separate parts mainly because their relationships were not linear. Doing this we can solve the coefficients and color optimizations

using a Non-linear Least-Squares approach. That way, there is more control over the process, since optimizing everything the algorithm could take long to converge or not converge at all.

4.6.1 Diffuse and Specular Coefficients

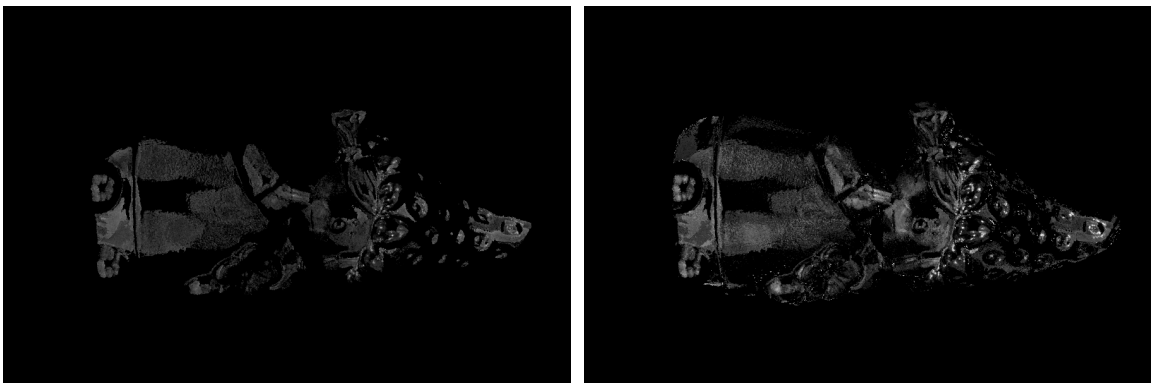
We need the derivatives $\frac{\partial f}{\partial k_d}$ and $\frac{\partial f}{\partial k_s}$ which are easily calculated:

$$\begin{aligned}\frac{\partial f}{\partial k_d} &= -2(P_i - C_i)(d_i C_{rgb}) \\ \frac{\partial f}{\partial k_s} &= -2(P_i - C_i)(s_i^\alpha S_{rgb})\end{aligned}$$



(a) Diffuse coefficient before optimization (b) Diffuse coefficient after optimization

Figure 4.13: The result of the optimization process on the diffuse coefficient.



(a) Specular coefficient before optimization (b) Specular coefficient after optimization

Figure 4.14: Result of the specular coefficient's optimization. We can observe how it increases the brightness of the shiniest spots.

4.6.2 Shininess

Notice that this is the first adjustment we perform on the shininess variable. Up to now we just used an initial empirical value of $\alpha = 10$. Although not optimal, the

chosen value provided satisfactory results as a starting point. The minimum point of f with respect to α is the critical point when $\frac{df}{d\alpha} = 0$. In order to show our final result, we develop this derivative:

$$\begin{aligned}\frac{df}{d\alpha} &= 2(P_i - k_d d_i C_{rgb} - k_s s_i^\alpha S_{rgb})(-k_s S_{rgb} s_i^\alpha \ln(s_i)) \\ \frac{df}{d\alpha} &= 2(P_i - k_d d_i C_{rgb})(-k_s S_{rgb} s_i^\alpha \ln(s_i)) - 2(k_s s_i^\alpha S_{rgb})(-k_s S_{rgb} s_i^\alpha \ln(s_i)) \\ \frac{df}{d\alpha} &= -2(P_i - k_d d_i C_{rgb})(k_s S_{rgb} \ln(s_i) s_i^\alpha) + 2((k_s S_{rgb})^2 (s_i^\alpha)^2 \ln(s_i))\end{aligned}$$

Finding the root of this equation is not a simple task since we have an exponential equation of α . To overcome this we use Newton's method to find the zero of an equation. We need the second derivative $\frac{d^2f}{d\alpha^2}$:

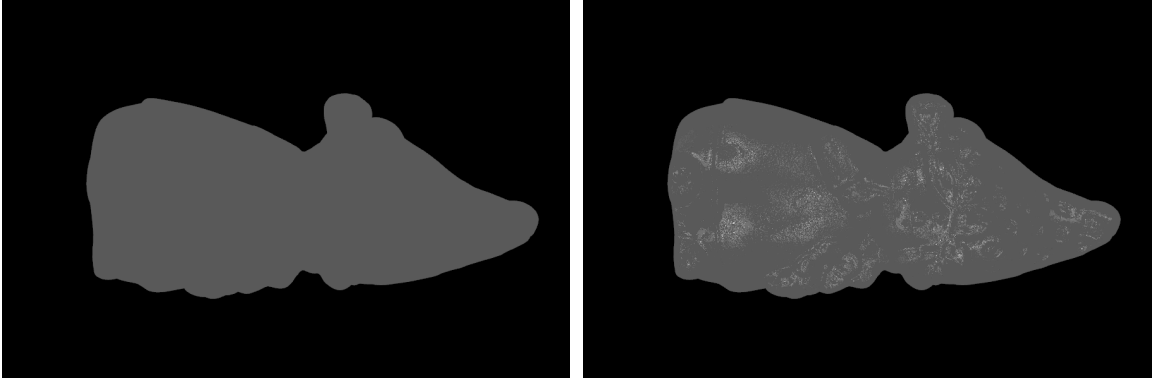
$$\begin{aligned}\frac{d^2f}{d\alpha^2} &= -2(P_i - k_d d_i C_{rgb})(k_s S_{rgb} \ln(s_i) (s_i^\alpha \ln(s_i))) + 2(k_s S_{rgb})^2 2(s_i^\alpha) (s_i^\alpha \ln(s_i)) \ln(s_i) \\ \frac{d^2f}{d\alpha^2} &= -2(P_i - k_d d_i C_{rgb})(k_s S_{rgb} s_i^\alpha \ln^2(s_i)) + 2(k_s S_{rgb})^2 2(s_i^\alpha)^2 \ln^2(s_i) \\ \frac{d^2f}{d\alpha^2} &= -2(P_i - k_d d_i C_{rgb})(k_s S_{rgb} s_i^\alpha \ln^2(s_i)) + 4(k_s S_{rgb} s_i^\alpha \ln(s_i))^2\end{aligned}$$

This step consists, in finding α such that $\frac{df}{d\alpha} = 0$. Since we are only optimizing one parameter this time, the classic Newton's method is suitable. It involves in iteratively finding the zero of a function g in small steps: $x_1 = x_0 + \Delta$. We calculate Δ as the following:

$$\Delta = -\frac{g}{g'}$$

In our case case, $g = \frac{df}{d\alpha}$ and $g' = \frac{d^2f}{d\alpha^2}$. This is a global optimization, therefore our optimization actually finds the root of the function g such that:

$$\begin{aligned}g &= \sum_{i \in U_v} -2(P_i - k_d d_i C_{rgb})(k_s S_{rgb} \ln(s_i) s_i^\alpha) + 2((k_s S_{rgb})^2 (s_i^\alpha)^2 \ln(s_i)) \\ g' &= \sum_{i \in U_v} -2(P_i - k_d d_i C_{rgb})(k_s S_{rgb} s_i^\alpha \ln^2(s_i)) + 4(k_s S_{rgb} s_i^\alpha \ln(s_i))^2\end{aligned}$$



(a) Shininess before

(b) Shininess after optimization

Figure 4.15: Result of the shininess' optimization.

4.6.3 Color

The final optimization step is to improve our basic diffuse color. For this matter, we need the partial derivatives $\frac{\partial f}{\partial r}$, $\frac{\partial f}{\partial g}$ and $\frac{\partial f}{\partial b}$.

$$\begin{aligned}\frac{\partial f}{\partial r} &= -2(Pr_i - Cr_i)(k_d d_i) \\ \frac{\partial f}{\partial g} &= -2(Pg_i - Cg_i)(k_d d_i) \\ \frac{\partial f}{\partial b} &= -2(Pb_i - Cb_i)(k_d d_i)\end{aligned}$$

4.7 Changing Viewpoints

Naturally when digitizing an object, more than one viewpoint will be necessary to cover its entire surface. Unlike techniques as RTI and techniques we have seen in Chapter 3, our work is naturally suitable for multiple viewpoints. In particular this is true due to our vertex-based acquisition. If one vertex appears in photos from different points of view, what matters is the local information of the pixel. The only drawback in changing viewpoints is the required calibration process again.

When there is a change in viewpoint, a vertex may have already appeared before and we need to analyze if its BRDF approximation is already good enough. If we have already made a good estimate for it, there is no need for further calculations. In order to do so, we analyze two quality metrics of every vertex.

4.7.1 Quality Metric

During the whole acquisition, some vertices may already have a good approximation, hence we do not need to further improve its BRDF. It is particularly useful to speed up the procedure when changing camera positions. We employ two metrics to determine a vertex quality: Diffuse Quality and Specular Quality. The first analyzes its diffuse color analyzing only photos outside highlight areas whereas the latter examines a vertex behavior in photos with highlight-only areas. Both qualities are in the range $[0, 2]$, where 2 means a good quality.

The diffuse quality can be viewed as a measure of how we diffusely sampled a vertex and how it approximates the given samples. In essence, we average its diffuse value and its normalized diffuse error, this is the vertex error divided by the norm of the white color, which represents the highest error possible. We calculate the diffuse quality as:

$$DQ = \frac{1}{n_d} \sum_{i \in U d_v} \vec{N} \cdot \vec{L} + \left(1 - \frac{1}{n_d} \sum_{i \in U d_v} \frac{(\vec{N} \cdot \vec{L})(P_i - C_i)^2}{\|P_w\|^2}\right)$$

Where, n_d is $|U d_v|$ and P_w is the white color. Notice that we actually calculate one minus the normalized error, so we are calculating how small the error is. The greater the error, the smaller is the contribution. Similarly, the specular quality is the average of how specularly sampled is the vertex and the normalized error of the specular samples. It is calculated as:

$$SQ = \frac{1}{n_s} \sum_{i \in U_{s_v}} \vec{R} \cdot \vec{E} + \left(1 - \frac{1}{n_s} \sum_{i \in U_{s_v}} \frac{(\vec{R} \cdot \vec{E})(P_i - C_i)^2}{\|P_w\|^2}\right)$$

Where n_s is $|U_{s_v}|$ and we also perform the one minus the normalized error calculation here.

Given these two metrics we can analyze how well sampled a vertex really is, both diffusely and specularly. This is an important metric we can also use when calculating a new light direction. If its diffuse color is not good enough we can ask for a new good diffuse light direction and conversely for the specular behavior. Since in our tests we did not perform a change of viewpoint, we did not specifically use this metric, but it is implemented for future tests.

Chapter 5

Experimentation and analysis

In this chapter we perform a series of tests to analyze the results of our BRDF approximation algorithm. We test it with different materials and examine its behavior. The datasets tested are the ones shown in Chapter 04, using only eight photos of each, except for Goats, which used four photos. The experiments were run on a computer with a i7-3770 CPU 3.40GHz x 8 and a GeForce GTX 660 GPU of 2GB RAM.

Subsequently, we provide a detailed discussion about several key points of our algorithm, these include its robustness with respect to a change of parameters, time and size performances, and several different aspects that changed throughout the development of this work. An important observation is that during the optimization, we attributed weights to each error. They are multiplied by the corresponding $(\vec{N} \cdot \vec{L}_i)$. We perform this to ensure that more important errors are more penalized. For specular areas, we tried to alternate between the former and the following weight $(\vec{R}_i \cdot \vec{E})$, but the results remain indifferent. We also experimented removing the weights or attributing new ones, but the results remained similar.

5.1 Results

5.1.1 Nana

The first object analyzed is the Nana doll, composed of a very specular head and a diffuse body. Besides, groups of similar colors, for example its green belly, tend to present a high color variation, *i.e.*, many close points present different green tonalities, which calls for a good SVBRDF approximation.

The Nana doll is an interesting test subject because it is composed of two very different types of materials and its diffuse color varies a lot on the surface. Results show that our algorithm is able to faithfully capture highlight areas and perform an efficient diffuse color extraction (Figures 5.1 and 5.2). However, there are some

drawbacks mainly regarding the diffuse and specular coefficients (Figure 5.3).



(a) Nana renderization

(b) Photograph of Nana

Figure 5.1: A comparison between the final rendering in 5.1a and a photo with corresponding light direction 5.1b. Our result faithfully captures the object's diffuse color and the highlight areas.

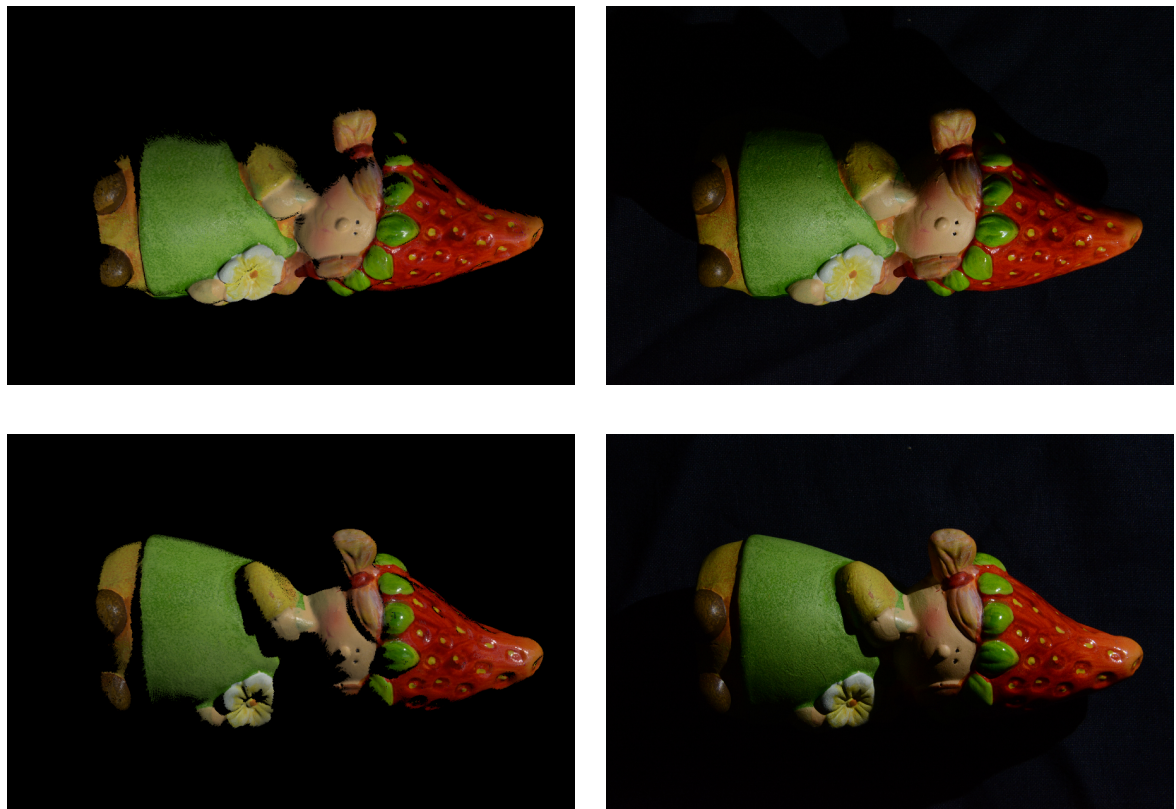
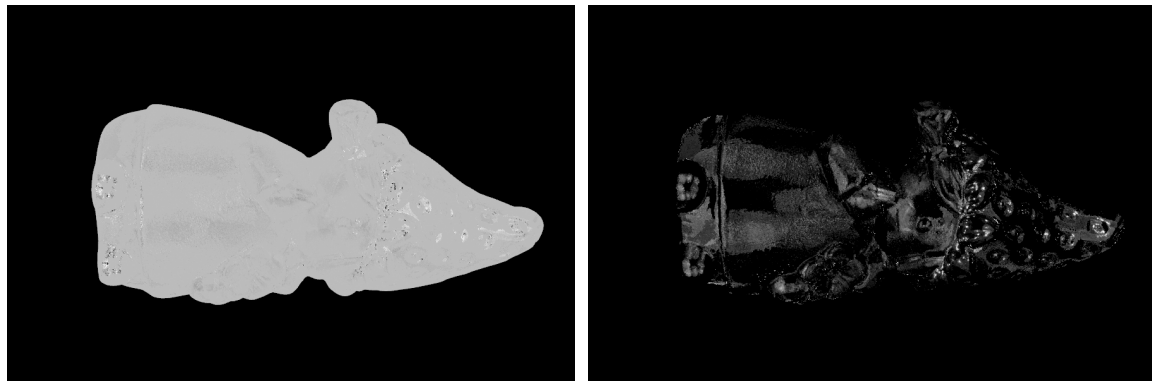


Figure 5.2: More results for Nana. Left column shows the rendering results and the right column photos with associated light directions. Again, we can notice a good diffuse color extraction and highlight shape reproduction. Our algorithm successfully approximates the highlight area, but was not able to reproduce the same specular intensity, this is more perceptible on the shoes.

A recurrent problem in our results is the huge weight that specular photos have

in the optimization, guiding the specular coefficient to become very high and the diffuse coefficient very low. This is visible as the black holes in the Figure 5.19c and the bright spots in 5.19d. Visually perceptible, but not so predominant in the Nana datasets. However, as will be seen, this becomes a major issue for the Buddha dataset.

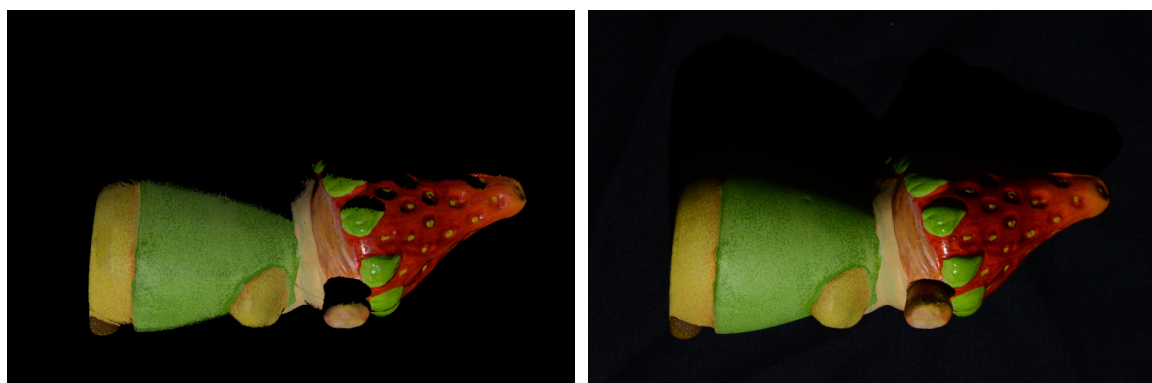


(a) Diffuse coefficient

(b) Specular coefficient

Figure 5.3: Visualization of diffuse and specular coefficients. Observe the dark dots present on Nana’s head in the diffuse coefficients photo and how these match the white dots in the specular coefficients photo. The error of specular photos are guiding the optimization process to set the specular coefficient too high while setting the diffuse coefficient too low.

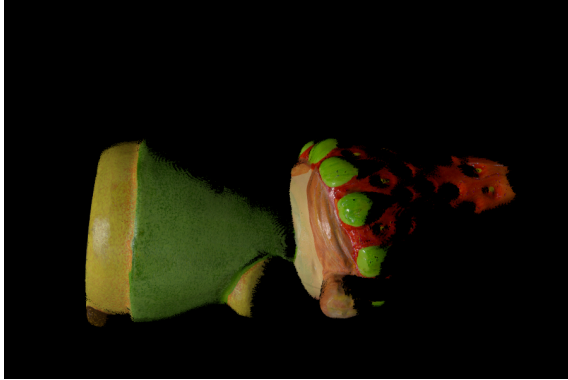
Our second dataset is still the Nana but from a different viewpoint. As expected from the results for the first dataset, we successfully approximated its BRDF. It is important to observe how the diffuse color matches the photo and how the highlight areas are appropriately captured.



(a) Nana Back rendering

(b) Photograph of Nana Back

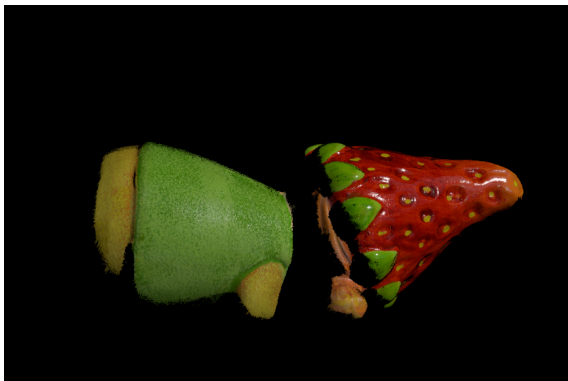
Figure 5.4: Comparison for the Nana Back between the rendering, left, and the photo, right.



(a) Rendering



(b) Photo



(c) Rendering



(d) Photo

Figure 5.5: Results for Nana. Left column shows the rendering results and the right column photos with associated light directions. Again the diffuse color approximation results are of high quality as well as the highlight shape. Unlike the first scenario where some highlight were not as intense as in the photo, for the Nana Back in some cases the highlight are even brighter than the photo, as the yellow base in Figure 5.5a.



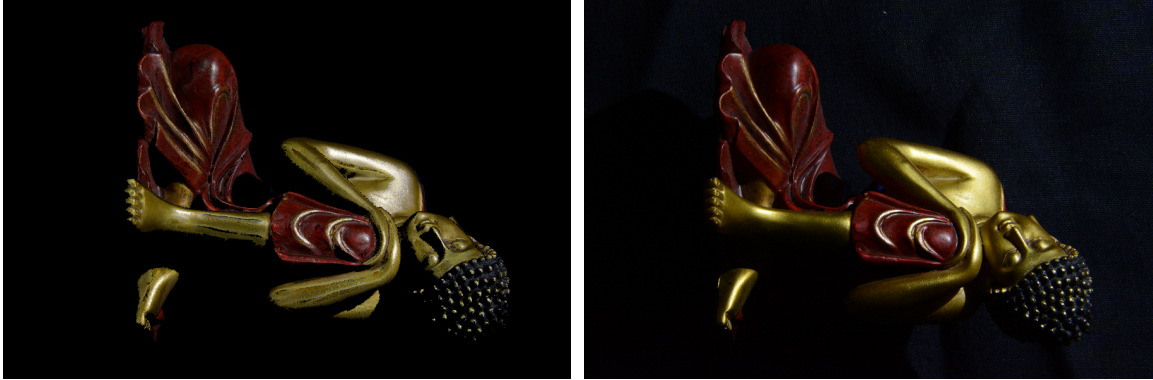
Figure 5.6: Nana rendered from a novel viewpoint. Black areas especially in the neck or the hand are vertices that were not seen from the fixed viewpoint, a different viewpoint is required to approximate their BRDFs.



Figure 5.7: Nana Back rendered from a novel viewpoint.

5.1.2 Buddha

The Buddha is a small statue composed of a highly specular golden part representing the Buddha’s body and a less shiny, but still moderately specular surface composing its robe. The robe is painted in a dark red color. There is also the Buddha’s hair composed of a mostly dark diffuse surface with some golden spots in the middle. Due to the golden material, we already expected to face some problems with the Buddha. We show results in Figure 5.8. Performing some tests, we could conclude that the



(a) Rendering of Buddha, the diffuse holes are much more visible here. (b) Photograph of Buddha. Notice how shiny its surface is.

Figure 5.8: A comparison between the final rendering in 5.8a and a photo with corresponding light direction 5.8b. While we can adequately approximate the robe’s BRDF, and the highlight shape to some extent, the problem experienced with the diffuse holes prevents a better reflectance approximation.

aforementioned issue is caused by the same factor, the specular photos guiding the optimization process. We show Buddha’s diffuse and specular coefficients in Figure 5.9 alongside a result showing how many vertices have exactly one specular photo. The fact that there is only one specular photo for these vertices would make us assume that the specular behavior of these vertices could not be adequately represented. However, what happened was that these sole specular photo could carry so much weight during the optimization that the diffuse component became misrepresented. As unusual as it seems, it is due to the fact that the diffuse areas in the Buddha photo, review Figure 5.8b, are very dark, yielding a smaller error during the optimization as opposed to the shiny areas.

Nevertheless, we are successful in capturing the diffuse color of the less specular part of the Buddha. Note its dark red robe in Figure 5.8. Even the specular behavior of the area is nicely captured.

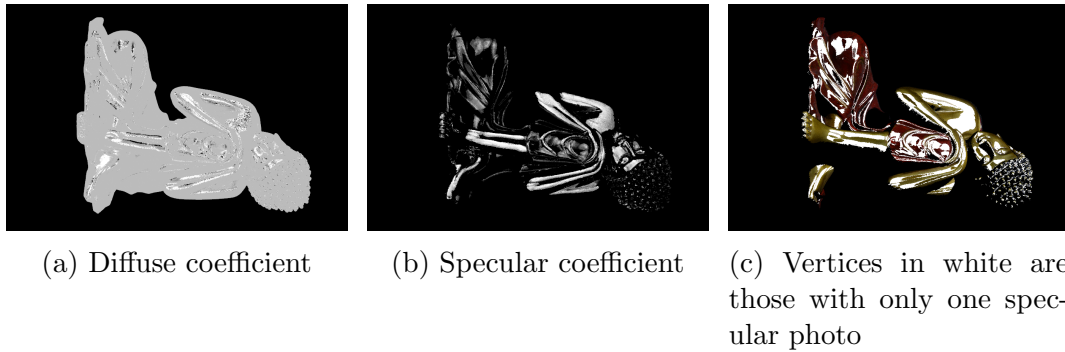


Figure 5.9: Comparing the three images it is noticeable that the vertices with only one specular photo are influencing heavily the optimization towards the specular value. This actually seems odd, as one would expect that having less specular photos would guide the optimization towards the diffuse coefficients. However, the very shiny nature of the buddha and the dark diffuse color presents a rather peculiar situation. Very low diffuse values do not yield an error as large as the specular one for the Buddha. We performed the same comparison with the Nana and this behavior was not reproduced.

5.1.3 Gertrud

Next, we analyze the case of the Gertrud dataset, a wooden statue with a close up of its face. The statue is painted with very different colors, but the object remains predominantly diffuse. There is only one piece of its nose where we can see the wood. See Figure 5.10 for results.

The main drawback in this dataset is the low quality geometry, causing a mesh-image misalignment and affecting the overall BRDF acquisition. This effect can be seen in the extended shadows in Figure 5.11c in comparison with the actual shadows of Figure 5.11b.



Figure 5.10: Results for the Gertrud dataset. We can see how the geometry of its face is much more noisy than the actual statue, which harms the performance of our algorithm. Although, it is still able to extract a good diffuse color, specially of its robe. Since neither the underlying object nor the painting is very specular, there are a few highlights which are not too as bright as expected.

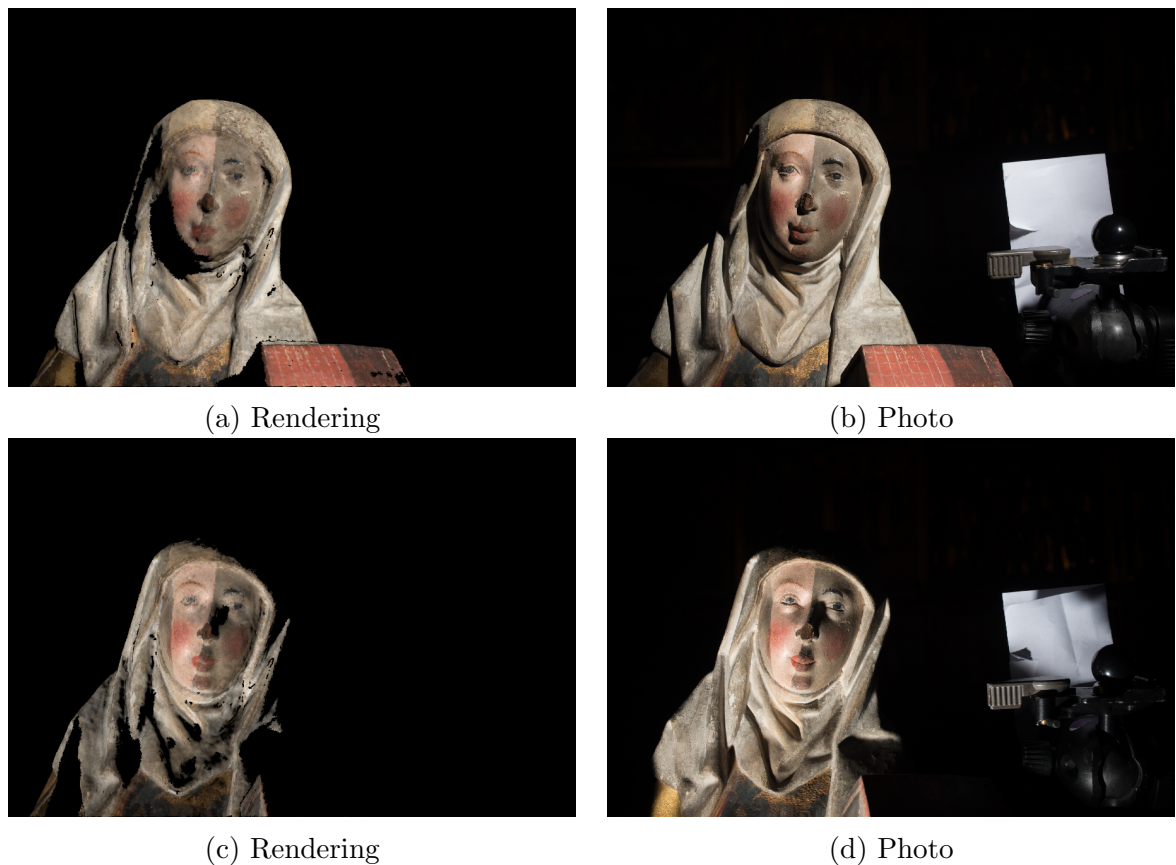


Figure 5.11: Results for Gertrud. Left column shows the rendering results and the right column photos with associated light directions

Notice in the Figure 5.12 that the mesh roughly matches the image which, unfortunately, is a sensible point in our technique. In order to faithfully represent the BRDF of each vertex, an accurate geometry is required.

As initially stated, the Gertrud statue is composed of wood, a complicated material to represent with an analytical BRDF. We were able to approximate some of its highlight areas, but not with the same intensity. Also, due to its complex geometry a higher number of photos would be required to appropriately acquire its reflectance properties. However, we compare our results with a color mapping generated by the Photoscan¹ software (Figure 5.13). Comparing our results with the photos shown earlier, we believe our diffuse color extraction surpasses basic color mapping algorithms.

5.1.4 Goats

Goats is a case that looks simple, but has some intrinsic subtleties. Its geometry, although mainly flat, possess a lot of bumps. Its BRDF is predominantly diffuse, but the diffuse color can become a lot darker depending on the light direction. Results

¹<http://www.agisoft.com/>



Figure 5.12: Blending between Mesh and Image. Notice the misalignment.



(a) The diffuse color generated by the Photocan software

(b) Our result with only the diffuse color shown

Figure 5.13: A color mapping comparison. We compare the result of the Photocan color mapping software with our basic diffuse color extraction. Observing both results and the photos shown before, our results are much more accurate.

are shown in Figure 5.14.



Figure 5.14: Results for Goats. Left column shows the rendering results and the right column photos with associated light directions

It seems that the basic diffuse color is faithfully extracted, but the diffuse behavior with the light in an oblique angle is not appropriately captured. This may be due to the fact that the Phong model behaves poorly when lights are in a grazing angle, specially for flat surfaces [34].

In the next figures we show the difference between the photos and renderings. We perform a subtraction between $P - C$ in Figures 5.15a and 5.15c, where P is the photo and C is the rendering color. Figures 5.15b and 5.15d show the result of $C - P$.

5.2 Discussions

In this section, we make some discussions about decisions made during the development of this work such as: switching from a local optimization to global optimization (Section 5.2.1), and changing from the RGB color space to the SRGB model (Section 5.2.2). We also discuss some main characteristics of the BRDF approximation algorithm, for example, its robustness against its parameters (Section 5.2.3), the choice of the BRDF model (Section 5.2.4), its behavior with respect to size (Section



(a) First photo minus the rendering color



(b) First rendering color minus the photo



(c) Second photo minus the rendering color



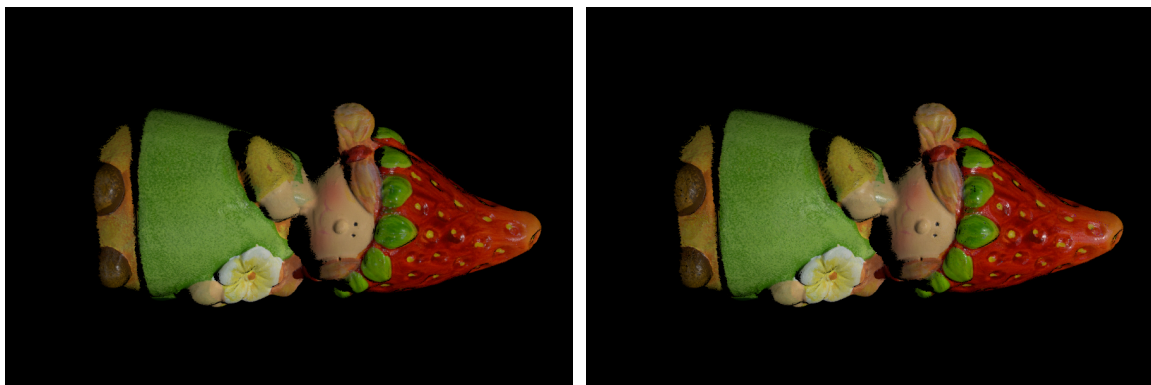
(d) Second rendering color minus the photo

Figure 5.15: Difference between the photo and the rendering color. Left column represents the difference $P - C$ and right column $C - P$. We can observe that more illuminated areas are brighter in the photo than in our rendering results, whereas less illuminated areas are darker in photos.

5.2.5), number of photos (Section 5.2.6) and time (Section 5.2.7).

5.2.1 Local vs Global Optimization

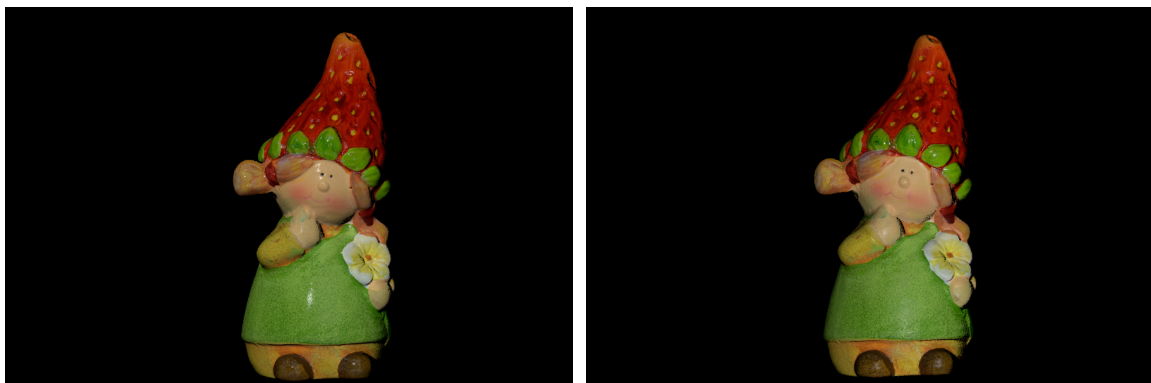
Before we employed the global optimization approach described in Section 4.6, we used to perform a local optimization to improve the BRDF's parameters. Instead of trying to minimize the error of all photos at once, we were constantly minimizing the error with only one photo at a time using the Gradient Descent method. The gradient of our error function consisted in calculating the same current derivatives. We compare the results of the different approaches next.



(a) Result using local optimization

(b) Result of the global optimization

Figure 5.16: Comparison between local and global optimization. Local optimizations fail to capture all highlight areas and its corresponding intensity. Notice how only some of the highlights present in the global optimization result are present in 5.16a.



(a) Result using local optimization in a novel position

(b) Global optimization result in a novel position

Figure 5.17: Models rendered from a novel viewpoint. The local optimization result seems like the object is made from only one type of material, Nana's body is as specular as its head, which is not the behavior seen in the global optimization result nor in the photos shown earlier.

This is an interesting result because it shows two different weaknesses of the local

optimization approach. In Figure 5.16 we can see that the local approach simply can not capture the highlight areas as accurate as the global optimization. Mainly because the results of the optimization are independent, so after optimizing the coefficients for one photo, it tries to optimize for another where there may be no more highlights in the same places. Areas that may be optimal for one photo may be completely altered when optimizing for another, yielding inaccurate results.

While sometimes failing to capture highlights, the local approach also typically gives a more specular behavior to non specular areas. This effect can be noted in Nana's body (Figure 5.17). Locally optimizing the coefficients seems to assign higher values to them, see Figure 5.3 and compare with the coefficients displayed in Figure 5.19. Nana looks like entirely made of one material, which is not the case.



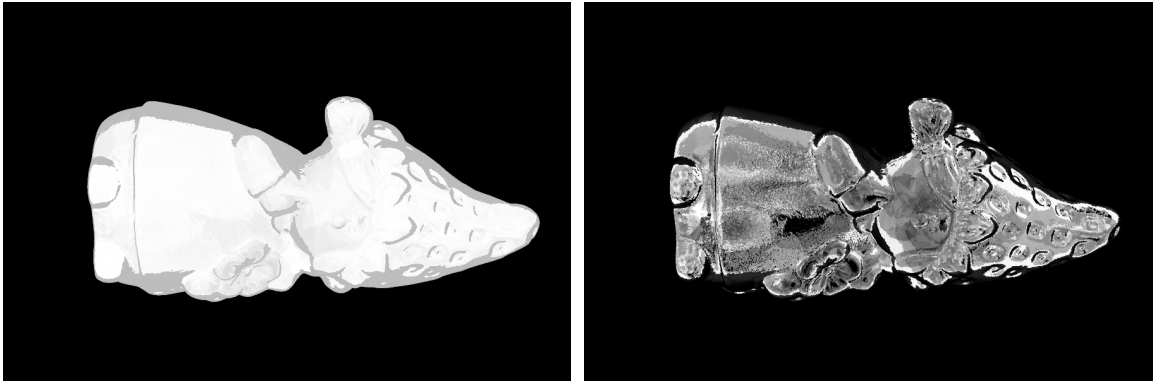
(a) Result using local optimization in a novel position (b) Global optimization result in a novel position

Figure 5.18: Again we can notice that the specular behavior is not similar to the actual model. Although there are no diffuse holes in the local optimization result, the BRDF obtained also does not correspond to the expected material.

5.2.2 SRGB vs RGB

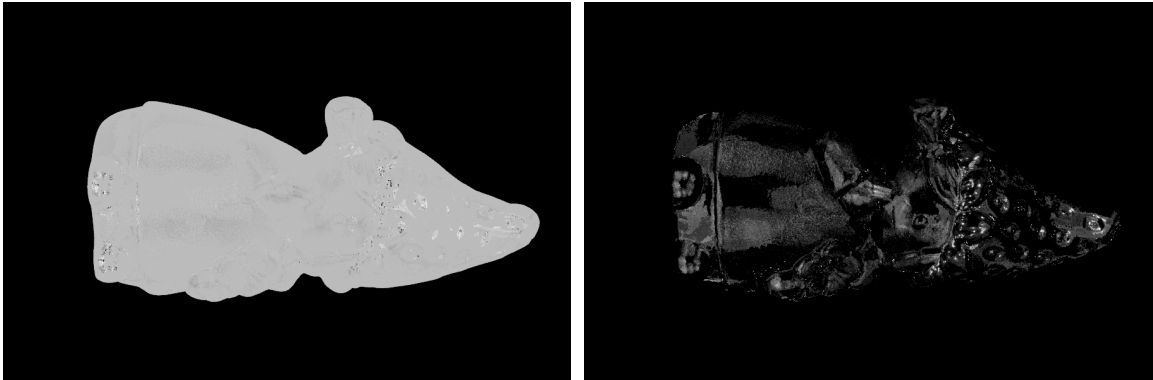
As discussed in Section 2.2, we should always perform operations in a linear space, which is not the case when we are working with the RGB color space. Therefore, the change to the SRGB space. We compare results obtained when using each color model and show why the SRGB is preferable.

Notice how much darker is the RGB result in Figure 5.20. This becomes more evident when the diffuse value decreases. Suppose a lambertian surface, the color is $(\vec{N} \cdot \vec{L})C$. If the dot product is equal to one, then it is the same color, but in a non linear space, the smaller is the dot product, the greater will be the difference.



(a) Diffuse coefficients after the local optimization.

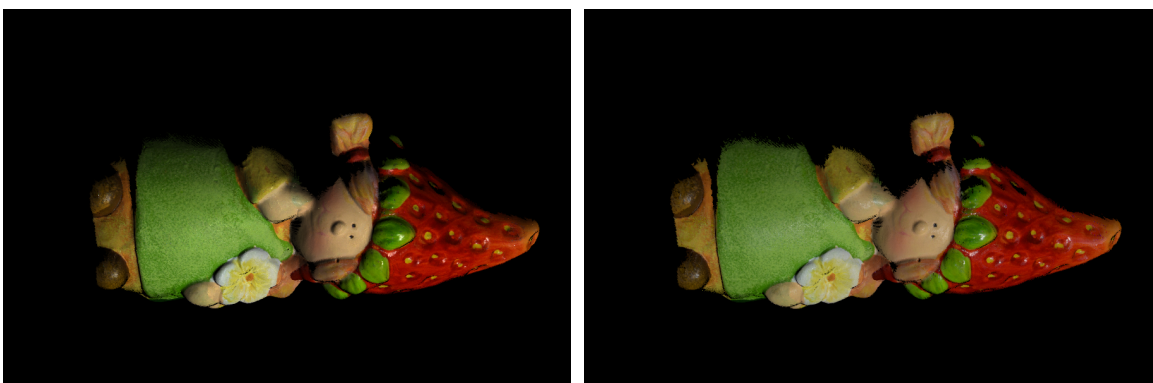
(b) Result of the local optimization on the specular coefficients.



(c) Diffuse coefficient

(d) Specular coefficient

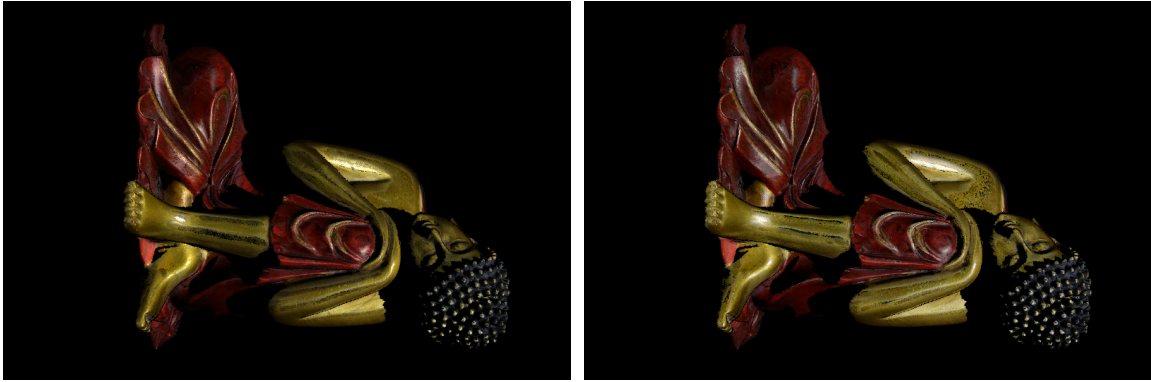
Figure 5.19: Coefficients obtained during the local optimization in the top row. Compare these with the global optimization results in the bottom row. Local optimization coefficients are too high.



(a) Using the RGB space

(b) Results with the SRGB

Figure 5.20: A comparison of the results obtained from the different color spaces. It is clearly noticeable that results from the RGB space are generally darker than the SRGB. This discrepancy increases as the diffuse value decreases, notice how its neck progressively darkens. The highlights are also not as bright as in the SRGB, the bright spot near the top of the head looks more reddish than in the right.



(a) Using the RGB space

(b) Results with the SRGB

Figure 5.21: The same divergences for the Nana can be seen in the Buddha. The RGB color space produces a darker result, look at Buddha’s left arm and the robe around its right knee. The highlights in SRGB are also brighter, note its right arm.

5.2.3 Parameters

A problem encountered with the local optimization approach was the dependence of parameters, the variable steps for the Gradient Descent needed to be determined empirically, which required laborious hours of testing and variable tweaking. With our current workflow we would like to test its parameters dependence, which are basically tests against the initial α value and maximum number of iterations n in the optimization. Our parameters up to now were: $\alpha = 10$ and $n = 400$. For the following analysis we performed tests with $\alpha = [20, 50, 100]$ and $n = [100, 1000]$.

We compare the highlight differences between the results when varying α , but what is more similar to the ground truth? We show a photo of the Gertrud statue in Figure 5.23 for comparison. It is clear that the results with smaller shininess values are more plausible.

Nevertheless, with the Gertrud dataset we did not encounter major differences when varying the initial α . This may be due to the fact that the surface is not too shiny, hence, it did not affect the coefficients computation too much.

In the Nana Back there are some noticeable changes when increasing the initial shininess, notice that when $\alpha = 20$, the results are fairly similar to the original $\alpha = 10$. When $\alpha = 50$ we can already notice more holes in its head than in the original results, again this happens because the optimization sets the diffuse coefficient to a low value. However, its overall behavior is similar to smaller values of α . Some major changes can be seen when $\alpha = 100$, there are even more holes in the model, some shiny artifacts throughout the head and even a major white stripe in its top.

We conclude that there is some influence of the α parameter, however we believe this happens only in extreme cases, since for the majority of objects there would be



(a) Rendering of Gertrud with $\alpha = 10$

(b) Rendering of Gertrud with $\alpha = 20$



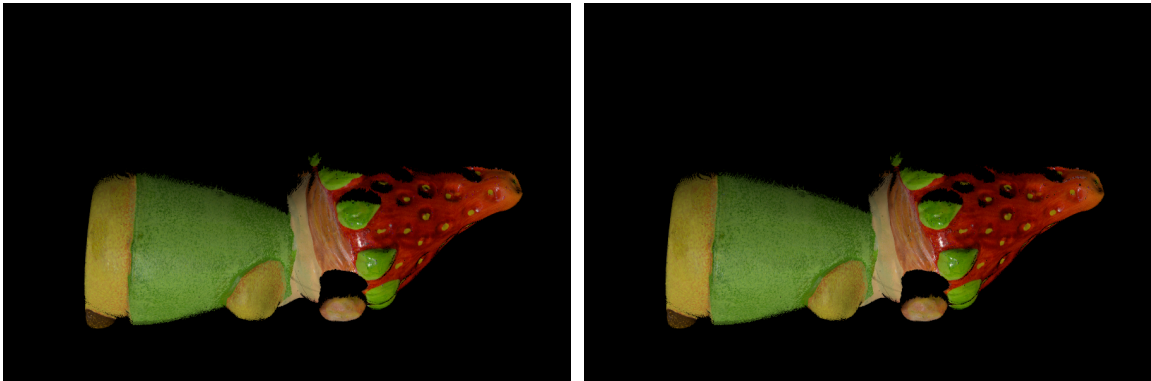
(c) Rendering of Gertrud with $\alpha = 50$

(d) Rendering of Gertrud with $\alpha = 100$

Figure 5.22: Rendering of Gertrud with different values of α . Although subtle, there's a noticeable difference in the shapes and intensities of the highlights in each. As expected with a increase of α , the highlights tend to be localized. Observing the highlights in the red areas, we can see that it progressively gets smaller.

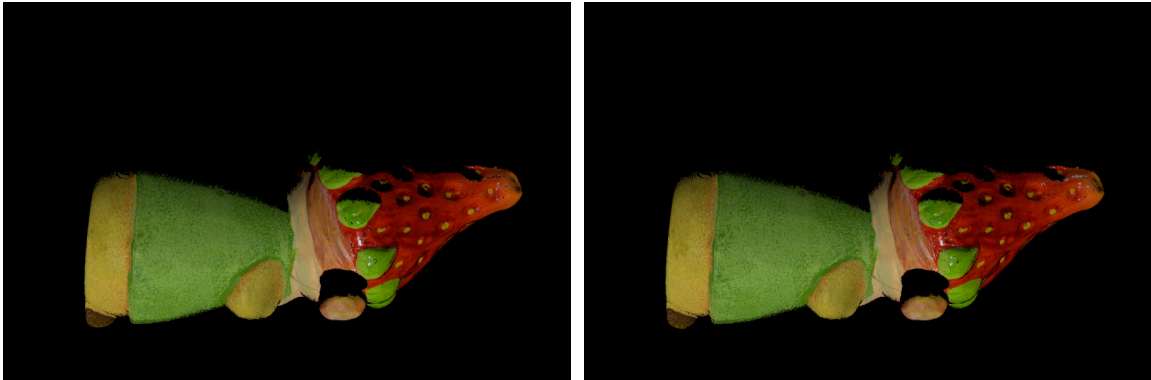


Figure 5.23: Photograph of Gertrud. We can notice that the highlights are more compatible with the results where α is small.



(a) Rendering of Nana Back with $\alpha = 10$

(b) Rendering of Nana Back with $\alpha = 20$



(c) Rendering of Nana Back with $\alpha = 50$

(d) Rendering of Nana Back with $\alpha = 100$

Figure 5.24: The different values of shininess in the Nana Back dataset produce more varied results than Gertrud. When α is small, as in the top row, the outcomes are moderately similar. In the case when $\alpha = 50$, we can notice more holes in Nana's head than in the previous results and some highlights, as in its body, can no longer be seen. Some major changes can also be seen when $\alpha = 100$, as some bright dots in its head, and a few white areas near the top of Nana's head, there is also no more highlights in its body.

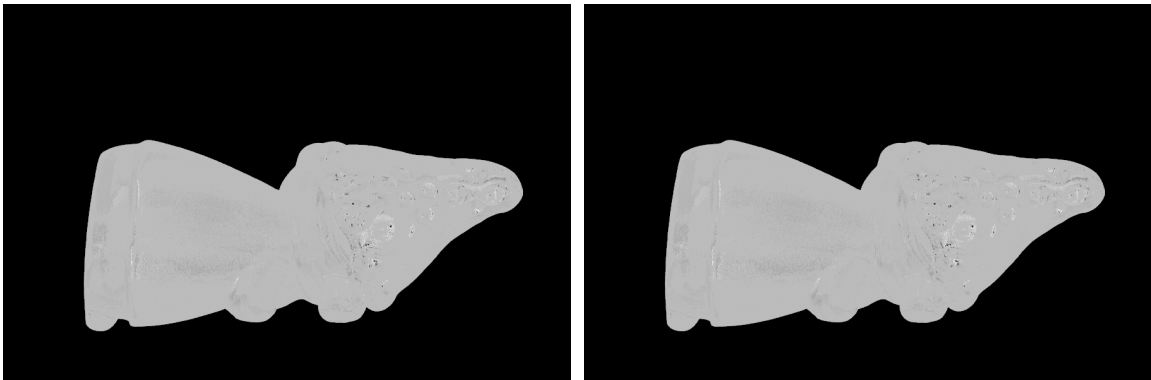
no need to set a value of α as high as 100. We may expect the algorithm to behave robustly with normal changes of shininess.

We also tested the different values for the maximum number of iterations in the optimization process. We conclude that our algorithm behaves very robustly against varying values of this parameters. Changes are slight, if ever existent. See Figure 5.25 for results. We show the result of the coefficients in Figures 5.26 and 5.27.



Figure 5.25: From left to right: $n = [100, 1000]$. We could not detect major differences in the results with the change of n . In fact, it may have never even reach the maximum number of 1000, since it probably converges before.

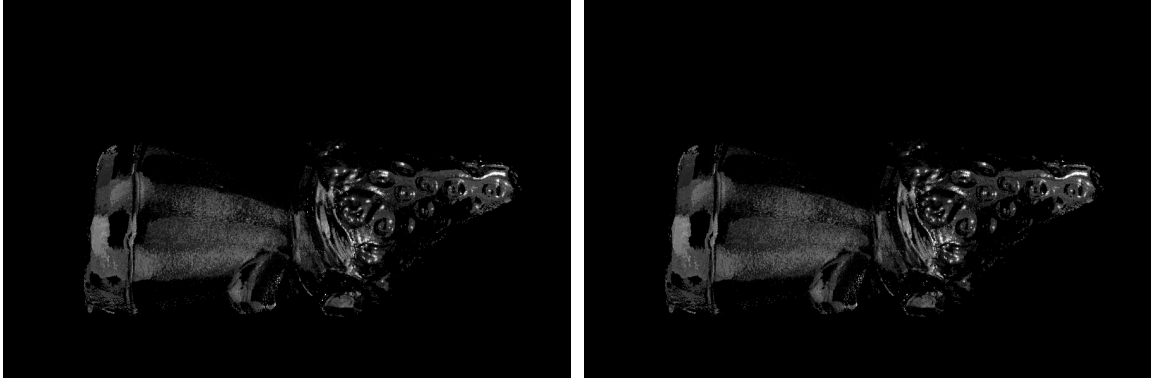
As stated, the results are considerably similar. Therefore, the maximum number of iterations does not seem to pose a restriction on our algorithm at all.



(a) Nana's diffuse coefficient after a maximum of 100 iterations for each optimization process. (b) Nana's diffuse coefficient after a maximum of 1000 iterations for each optimization process.

Figure 5.26: As expected, the results for the diffuse coefficient remains the same with the change of the number of iterations.

Our BRDF approximation algorithm needs only two initial parameter: shininess and maximum number of iterations in the optimization. We verified its behavior against these two variables. The latter shows that our algorithm probably converges



(a) Nana’s specular coefficient after a maximum of 1000 iterations for each optimization process. (b) Nana’s specular coefficient after a maximum of 1000 iterations for each optimization process.

Figure 5.27: Like the results for the diffuse coefficient, the specular coefficients remain the same with the change in the number of iterations.

fast and does not pose an issue. A great change in the shininess can affect the result, but we believe these are only present in drastic changes.

5.2.4 BRDF Model

The results obtained with Phong’s model are desirably accurate and suitable for a fast BRDF acquisition system. The choice for the Phong model resided in its simplicity and consequent easiness to use in an iterative method during the optimization. However it is still a naturally limited model and we believe a further discussion is necessary. For example, a common argument towards BRDF models that use a half-vector lobe instead of the cosine lobe are that the latter presents inconsistent results with grazing angles. We assume this happens in the Goats tests, Phong’s model can not appropriately represent it.

One idea for future work and to improve results is to capture all data and have a good approximation of the BRDF in situ, and later perform a thorough BRDF fitting with possibly a different reflectance model. Nevertheless, Phong’s model allows us to perform a fast and faithful representation.

5.2.5 Size

There are two considerations about size to make: the algorithm and the results storage. During the whole approximation, our algorithm can be very data consuming. In terms of CPU and RAM memory, it is relatively cheap, it is almost mandatory nowadays to have a computer with at least 4GB of RAM, so it hardly imposes a limitation to our system. However, even advanced GPUs hardly surpass 2GB, as the one used in our tests. This limit combined with a naive implementation of texture

arrays in OpenGL posed a restriction of how much data we could fit inside the GPU. This is why we could not perform tests with more than 10 photos at once, since the global optimization needs to access all the photos at the same time. Nevertheless, we believe a more careful implementation can solve this problem.

On the other hand, since our data consists of only 6 floating-point numbers for each vertex of the mesh, and considering the IEEE-754 standard for single precision numbers, a model with 1 million vertices only needs an extra storage of 24 *MB*. Which means storing the results obtained by our system is a direct and rather cheap storage. Many PLY objects already store color information in its encoding, to store the other three coefficients of the Phong model would be a simple task. Unlike the BTF from [27] which even compressed occupies 1.5*GB*, our results are provided at a small cost.

5.2.6 Number of photos

Due to limitations discussed in Section 5.2.5, we could not increase the number of photos too much. The default value for all the datasets was eight photos, but for the Goats where we used four. The Goats’s model is the largest mesh, which already occupies a significant amount on the memory. Because of memory restrictions we could only test with up to 10 photos which did not yield the result variety we wanted.

5.2.7 Time and Feedback

Time was barely discussed in our results but it is worth mentioning since in the beginning we proposed an efficient acquisition system. For our datasets, the total time was: Nana 42.7443s, Nana Back 40.2035s, Buddha 44.9737s and Gertrud 19.1031s. The Goats dataset was processed in a different computer and the results are not compatible. These results include all the steps of the BRDF approximation described in Chapter 4 without the calibration procedure. First, we did not find any work in the literature that could provide results in less than a minute, most performed over hours of processing, the exception being the work of REN *et al.* [29] that takes about 20 minutes. Nevertheless, for an online capture, a waiting time of 20 minutes is a considerable holdback. Second, these are tests performed with an already acquired data. The time of the calibration procedure is rather small, sphere and chart detection are nearly instantaneous and the Mutual Correspondence algorithm for the mesh-image alignment takes no more than ten seconds.

The important contribution of our algorithm is that it allows the creation of the interactive acquisition system. For each new photo taken, our algorithm instantly processes it and generates a new result. This is done only for the initial basic color, for the resulting BRDF there would be the need to wait around 45s according to

the results above. However, this initial feedback allows the user to know if all the vertices were covered, and if not, to take another photo from another viewpoint or ask for a new light direction. Our system also provides the visualization of which vertices have a specular photo allowing for an immediate decision. From the tests described in Section 4.4, initial results of calculating a new light direction can take at least 30s with our current approach. We can make it almost instantaneously if we do not test all the lights in the hemisphere to check for the best one, but just selecting the bin with the highest number of lights. From experiments we realized that this is often not the chosen light, but it is always near the most efficient ones, so it could a worthwhile trade-off.

Chapter 6

Conclusion

”End? No, the journey doesn’t end here.”

— ”Gandalf - The Lord of the Rings: The
Return of the King”

In this work we have presented a system for simple BRDF approximation using photos with light directions in a fixed viewpoint. We presented a fast algorithm that can faithfully extract the basic color of the materials and is also precise at reproducing highlights. There are some drawbacks regarding shiny objects, but we believe minor corrections in our optimization algorithm could solve the problem. This system was designed having in mind CH professionals who need to perform digitization of historical artifacts, and many times need to travel to a certain location for this task. To the best of our knowledge, there is no tool in the research community that can provide the same immediate feedback, allowing the user to make decisions during the acquisition. Aligned with results computed in less than a minute we provide an efficient BRDF acquisition system. Nevertheless, there is always room for improvement and experimentation. Here we summarize some avenues of future research which have come up during the work on this dissertation

6.1 Future Works

Our experiments show convincing and appealing, but yet imperfect results. For some datasets the BRDF approximation is really faithful, while for others there were some drawbacks. Nonetheless, we believe some minor changes could already fix some small problems we experienced during the tests.

Solve the GPU limitation issue There is a need to overcome the maximum number of photos used in our GPU. This could probably be achieved with better

data structures or by subdividing the problem.

Solve the coefficients problem A recurrent artifact in our results was some very low diffuse coefficients. In some datasets, they were not a big issue, but in the case of the Buddha, for example, it greatly hindered the final solution. There is a need to analyze how to weight each photo to guarantee that the optimization is not biased.

Field tests Up to now we have only used pre-acquired datasets. Our next step is to perform field tests with our system in order to evaluate its performance during an online acquisition. Besides, in none of the tests we used the color checker, we pretend to use it during the next tests.

Use neighbor's information during optimization As can be noted specially in the images of the coefficients such as Figure 5.3, there are many sparse areas that do not seem covered in the optimization approach, for example, the specular coefficients of Nana's head. To fully acquire all the surface's points requires too many photos, which compels us to create a smarter approach. One idea is to use the neighbor's information if they already have a high quality BRDF approximation, we may compute this with the quality metric discussed in 4.7.1.

Post computation after acquisition Although Phong's model provides convincing results, there may be a need to use another reflectance model for objects with complicated materials. One idea would be to make sure in situ we capture all the relevant data for a good BRDF approximation, our algorithm can provide that assurance, and perform a more time-consuming optimization approach offline.

Bibliography

- [1] RITSCHER, T., DACHSBACHER, C., GROSCH, T., et al. “The State of the Art in Interactive Global Illumination”, *Comput. Graph. Forum*, v. 31, n. 1, pp. 160–188, fev. 2012. ISSN: 0167-7055. doi: 10.1111/j.1467-8659.2012.02093.x. URL: <<http://dx.doi.org/10.1111/j.1467-8659.2012.02093.x>>.
- [2] WEYRICH, T., LAWRENCE, J., LENSCH, H. P., et al. “Principles of Appearance Acquisition and Representation”, *Foundations and Trends in Computer Graphics and Vision*, v. 4, n. 2, pp. 75–191, out. 2009.
- [3] NICODEMUS, F., RICHMOND, J., HSIA, J., et al. “Geometrical considerations and nomenclature for reflectance”, *Applied Optics*, v. 9, pp. 1474–1475, 1977.
- [4] PHONG, B. T. “Illumination for Computer Generated Pictures”, *Commun. ACM*, v. 18, n. 6, pp. 311–317, jun. 1975. ISSN: 0001-0782. doi: 10.1145/360825.360839. URL: <<http://doi.acm.org/10.1145/360825.360839>>.
- [5] BLINN, J. F. “Models of Light Reflection for Computer Synthesized Pictures”. In: *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '77, pp. 192–198, New York, NY, USA, 1977. ACM. doi: 10.1145/563858.563893. URL: <<http://doi.acm.org/10.1145/563858.563893>>.
- [6] OREN, M., NAYAR, S. K. “Generalization of Lambert’s Reflectance Model”. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pp. 239–246, New York, NY, USA, 1994. ACM. ISBN: 0-89791-667-0. doi: 10.1145/192161.192213. URL: <<http://doi.acm.org/10.1145/192161.192213>>.
- [7] LAFORTUNE, E. P. F., FOO, S.-C., TORRANCE, K. E., et al. “Non-linear Approximation of Reflectance Functions”. In: *Proceedings of the*

- 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pp. 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN: 0-89791-896-7. doi: 10.1145/258734.258801. URL: <<http://dx.doi.org/10.1145/258734.258801>>.
- [8] NGUYEN, H. *Gpu Gems 3*. First ed. Natick, MA, Addison-Wesley Professional, 2007. ISBN: 9780321545428.
- [9] JUDD, D. B., MACADAM, D. L., WYSZECKI, G., et al. “Spectral Distribution of Typical Daylight as a Function of Correlated Color Temperature”, *J. Opt. Soc. Am.*, v. 54, n. 8, pp. 1031–1040, Aug 1964. doi: 10.1364/JOSA.54.001031. URL: <<http://www.opticsinfobase.org/abstract.cfm?URI=josa-54-8-1031>>.
- [10] HARTLEY, R. I., ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. Second ed. Oxford, UK, Cambridge University Press, ISBN: 0521540518, 2004.
- [11] SHREINER, D., SELLERS, G., KESSENICH, J. M., et al. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. 8th ed. Boston, Addison-Wesley Professional, 2013. ISBN: 0321773039, 9780321773036.
- [12] LI, M. “Correspondence Analysis Between The Image Formation Pipelines of Graphics and Vision”. In: *Proceedings of the IX Spanish Symposium on Pattern Recognition and Image Analysis*, pp. 187–192. Publications de la Universitat Jaume I., 2001.
- [13] CALLIERI, M., CIGNONI, P., CORSINI, M., et al. “Technical Section: Masked Photo Blending: Mapping Dense Photographic Data Set on High-resolution Sampled 3D Models”, *Comput. Graph.*, v. 32, n. 4, pp. 464–473, ago. 2008. ISSN: 0097-8493. doi: 10.1016/j.cag.2008.05.004. URL: <<http://dx.doi.org/10.1016/j.cag.2008.05.004>>.
- [14] ZHOU, Q.-Y., KOLTUN, V. “Color Map Optimization for 3D Reconstruction with Consumer Depth Cameras”, *ACM Trans. Graph.*, v. 33, n. 4, pp. 155:1–155:10, jul. 2014. ISSN: 0730-0301. doi: 10.1145/2601097.2601134. URL: <<http://doi.acm.org/10.1145/2601097.2601134>>.
- [15] LENSCH, H., KAUTZ, J., GOESELE, M., et al. “Image-Based Reconstruction of Spatially Varying Materials”. In: Gortler, S., Myszkowski, K. (Eds.), *Rendering Techniques 2001*, Eurographics, Springer Vienna, pp. 103–114,

2001. ISBN: 978-3-211-83709-2. doi: 10.1007/978-3-7091-6242-2_10. URL: <http://dx.doi.org/10.1007/978-3-7091-6242-2_10>.

- [16] LENSCH, H. P. A., KAUTZ, J., GOESELE, M., et al. “Image-based Reconstruction of Spatial Appearance and Geometric Detail”, *ACM Trans. Graph.*, v. 22, n. 2, pp. 234–257, abr. 2003. ISSN: 0730-0301. doi: 10.1145/636886.636891. URL: <<http://doi.acm.org/10.1145/636886.636891>>.
- [17] MALZBENDER, T., GELB, D., WOLTERS, H. “Polynomial Texture Maps”. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01*, pp. 519–528, New York, NY, USA, 2001. ACM. ISBN: 1-58113-374-X. doi: 10.1145/383259.383320. URL: <<http://doi.acm.org/10.1145/383259.383320>>.
- [18] GOLDMAN, D., CURLESS, B., HERTZMANN, A., et al. “Shape and spatially-varying BRDFs from photometric stereo”. In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, v. 1, pp. 341–348 Vol. 1, Oct 2005. doi: 10.1109/ICCV.2005.219.
- [19] ALLDRIN, N. G., ZICKLER, T., KRIEGMAN, D. J. “Photometric stereo with non-parametric and spatially-varying reflectance.” In: *CVPR*. IEEE Computer Society, 2008. URL: <<http://dblp.uni-trier.de/db/conf/cvpr/cvpr2008.html#AlldrinZK08>>.
- [20] PALMA, G., CALLIERI, M., DELLEPIANE, M., et al. “A Statistical Method for SVBRDF Approximation from Video Sequences in General Lighting Conditions”, *Comp. Graph. Forum*, v. 31, n. 4, pp. 1491–1500, jun. 2012. ISSN: 0167-7055. doi: 10.1111/j.1467-8659.2012.03145.x. URL: <<http://dx.doi.org/10.1111/j.1467-8659.2012.03145.x>>.
- [21] PALMA, G., CALLIERI, M., DELLEPIANE, M., et al. “Geometry-aware Video Registration.” In: Koch, R., Kolb, A., Rezk-Salama, C. (Eds.), *VMV*, pp. 107–114. Eurographics Association, 2010. ISBN: 978-3-905673-79-1. URL: <<http://dblp.uni-trier.de/db/conf/vmv/vmv2010.html#PalmaCDCS10>>.
- [22] DEBEVEC, P. “A Median Cut Algorithm for Light Probe Sampling”. In: *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, pp. 33:1–33:3, New York, NY, USA, 2008. ACM. doi: 10.1145/1401132.1401176. URL: <<http://doi.acm.org/10.1145/1401132.1401176>>.

- [23] DONG, Y., CHEN, G., PEERS, P., et al. “Appearance-from-motion: Recovering Spatially Varying Surface Reflectance Under Unknown Lighting”, *ACM Trans. Graph.*, v. 33, n. 6, pp. 193:1–193:12, nov. 2014. ISSN: 0730-0301. doi: 10.1145/2661229.2661283. URL: <<http://doi.acm.org/10.1145/2661229.2661283>>.
- [24] ASHIKMIN, M., PREMOŽE, S., SHIRLEY, P. “A Microfacet-based BRDF Generator”. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '00*, pp. 65–74, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN: 1-58113-208-5. doi: 10.1145/344779.344814. URL: <<http://dx.doi.org/10.1145/344779.344814>>.
- [25] HOLROYD, M., LAWRENCE, J., ZICKLER, T. “A Coaxial Optical Scanner for Synchronous Acquisition of 3D Geometry and Surface Reflectance”. In: *ACM SIGGRAPH 2010 Papers, SIGGRAPH '10*, pp. 99:1–99:12, New York, NY, USA, 2010. ACM. ISBN: 978-1-4503-0210-4. doi: 10.1145/1833349.1778836. URL: <<http://doi.acm.org/10.1145/1833349.1778836>>.
- [26] WILLEMS, G., VERBIEST, F., MOREAU, W., et al. “Easy and cost-effective cuneiform digitizing”. In: *The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST 2005)*, pp. 73–80, 2005.
- [27] SCHWARTZ, C., WEINMANN, M., RUITERS, R., et al. “Integrated High-Quality Acquisition of Geometry and Appearance for Cultural Heritage”. In: *The 12th International Symposium on Virtual Reality, Archeology and Cultural Heritage VAST 2011*, pp. 25–32. Eurographics Association, Eurographics Association, out. 2011. ISBN: 978-3-905674-34-7. doi: 10.2312/VAST/VAST11/025-032. URL: <<http://diglib.eg.org/EG/DL/WS/VAST/VAST11/025-032.pdf>>.
- [28] DANA, K. J., VAN GINNEKEN, B., NAYAR, S. K., et al. “Reflectance and Texture of Real-world Surfaces”, *ACM Trans. Graph.*, v. 18, n. 1, pp. 1–34, jan. 1999. ISSN: 0730-0301. doi: 10.1145/300776.300778. URL: <<http://doi.acm.org/10.1145/300776.300778>>.
- [29] REN, P., WANG, J., SNYDER, J., et al. “Pocket Reflectometry”. In: *ACM SIGGRAPH 2011 Papers, SIGGRAPH '11*, pp. 45:1–45:10, New York, NY, USA, 2011. ACM. ISBN: 978-1-4503-0943-1. doi: 10.1145/1964921.1964940. URL: <<http://doi.acm.org/10.1145/1964921.1964940>>.

- [30] MUDGE, M., MALZBENDER, T., SCHROER, C., et al. “New Reflection Transformation Imaging Methods for Rock Art and Multiple-viewpoint Display”. In: *Proceedings of the 7th International Conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage, VAST’06*, pp. 195–202, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. ISBN: 3-905673-42-8. doi: 10.2312/VAST/VAST06/195-202. URL: <<http://dx.doi.org/10.2312/VAST/VAST06/195-202>>.
- [31] SOTTILE, M., DELLEPIANE, M., CIGNONI, P., et al. “Mutual Correspondences: An Hybrid Method for Image-to-geometry Registration.” In: Puppo, E., Brogni, A., Floriani, L. D. (Eds.), *Eurographics Italian Chapter Conference*, pp. 81–88. Eurographics, 2010. ISBN: 978-3-905673-80-7. URL: <<http://dblp.uni-trier.de/db/conf/egItaly/egItaly2010.html#SottileDCS10>>.
- [32] MOORE, R. “On the Reciprocal of the General Algebraic Matrix”, *Bulletin of the American Mathematical Society*, v. 26, n. 9, pp. 394–396, 06 1920. URL: <<http://projecteuclid.org/euclid.bams/1183425340>>.
- [33] PENROSE, R. “A generalized inverse for matrices”, *Mathematical Proceedings of the Cambridge Philosophical Society*, v. 51, pp. 406–413, 7 1955. ISSN: 1469-8064. doi: 10.1017/S0305004100030401. URL: <http://journals.cambridge.org/article_S0305004100030401>.
- [34] AKENINE-MÖLLER, T., HAINES, E., HOFFMAN, N. *Real-Time Rendering 3rd Edition*. Natick, MA, USA, A. K. Peters, Ltd., 2008. ISBN: 987-1-56881-424-7.