A GENERALIZED SURFACE APPEARANCE REPRESENTATION FOR COMPUTER GRAPHICS

by
David Kirk McAllister

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2002

Approved by:

Advisor: Anselmo Lastra

Reader: Gary Bishop

Reader: Steven Molnar

Henry Fuchs

Lars Nyland

ABSTRACT

DAVID KIRK MCALLISTER. A Generalized Surface Appearance Representation for Computer Graphics

(Under the direction of Anselmo Lastra)

For image synthesis in computer graphics, two major approaches for representing a surface's appearance are texture mapping, which provides spatial detail, such as wallpaper, or wood grain; and the 4D *bi-directional reflectance distribution function* (BRDF) which provides angular detail, telling how light reflects off surfaces. I combine these two modes of variation to form the 6D *spatial bi-directional reflectance distribution function* (SBRDF). My compact SBRDF representation simply stores BRDF coefficients at each pixel of a map. I propose SBRDFs as a surface appearance representation for computer graphics and present a complete system for their use.

I acquire SBRDFs of real surfaces using a device that simultaneously measures the BRDF of every point on a material. The system has the novel ability to measure anisotropy (direction of threads, scratches, or grain) uniquely at each surface point. I fit BRDF parameters using an efficient nonlinear optimization approach specific to BRDFs.

SBRDFs can be rendered using graphics hardware. My approach yields significantly more detailed, general surface appearance than existing techniques for a competitive rendering cost. I also propose an SBRDF rendering method for global illumination using prefiltered environment maps. This improves on existing prefiltered environment map techniques by decoupling the BRDF from the environment maps, so a single set of maps may be used to illuminate the unique BRDFs at each surface point.

I demonstrate my results using measured surfaces including gilded wallpaper, plant leaves, upholstery fabrics, wrinkled gift-wrapping paper and glossy book covers.

To Tiffany, who has worked harder and sacrificed more for this than have I.

ACKNOWLEDGMENTS

I appreciate the time, guidance and example of Anselmo Lastra, my advisor. I'm grateful to Steve Molnar for being my mentor throughout graduate school. I'm grateful to the other members of my committee, Henry Fuchs, Gary Bishop, and Lars Nyland for helping and teaching me and creating an environment that allows research to be done successfully and pleasantly.

I am grateful for the effort and collaboration of Ben Cloward, who masterfully modeled the Carolina Inn lobby, patiently worked with my software, and taught me much of how artists use computer graphics. I appreciate the collaboration of Wolfgang Heidrich, who worked hard on this project and helped me get up to speed on shading with graphics hardware. I'm thankful to Steve Westin, for patiently teaching me a great deal about surface appearance and light measurement. I'm grateful for the tremendous help of John Thomas in building the spatial gonioreflectometer. I'm grateful to Nvidia Corp. for equipment and employment, and to the people of Nvidia for encouragement, guidance and support.

I am grateful to my parents, Stephen and Irene McAllister, for teaching, mentoring, and encouraging me. I am thankful to my children, Naomi, Hazel, and Jonathan McAllister for believing in me and praying for me.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

The following symbols are used in this dissertation.

β	Angle of anisotropy in tangent plane rel. to tangent vector $\vec{T}$
E	Irradiance (watts/meter^2)
L	Radiance (watts / meter^2/steradian)
$L(\omega)$	Radiance in direction ω
ω_i	Incident direction vector (unit length, except as noted)
ω_r	Exitant (reflected) direction vector (unit length, except as noted)
$\vec{T}$	Tangent vector (unit length)
$\vec{B}$	Binormal vector (unit length)
$\vec{N}$	Normal vector (unit length)
Ω_i	Incident hemisphere (as an integration domain)
Ω_r	Exitant (reflected) hemisphere (as an integration domain)
$\delta(a-b)$	Dirac delta function. $\delta(a-b)$=1 when a=b and 0 otherwise.
C	3×3 matrix within Lafortune BRDF representation
n	Specular exponent within a BRDF representation
p	The number of parameters defining a specular highlight
ρ_d	Diffuse albedo – a RGB triple
ρ_s	Specular albedo – a RGB triple
$\zeta_r(\theta_r)$	Solid angle of a cone of angle θ_r
ζ_Ω	Solid angle of a hemisphere. $\zeta_\Omega = 2\pi$ sr.
sr	Steradian – SI unit of solid angle

1. INTRODUCTION

One of the many goals of computer graphics is the creation of detailed, expressive images. These images may either be photorealistic – intended to look like the real world, or non-photorealistic – styled by an artist. Image synthesis of either kind benefits from increased detail and expressivity in the components of the scene. Four major components of a typical computer generated scene are

- The light sources illuminating the scene;
- The transmittance effects as light travels through the atmosphere or translucent volumes;
- The shape of objects in the scene, usually represented using geometry; and
- The appearance of the objects' surfaces – the color, texture, or reflectance of the surfaces.

Computer graphics rendering is the process of synthesizing pictures given the data representing these four components of a scene. The data may come from a variety of sources that fall into two broad categories – measured and synthesized. Measured light source data is available from most manufacturers of light bulbs and fixtures. Synthetic lights may be defined by artists by specifying just a few parameters. Measurements of transmittance effects for transparent objects usually come from manufacturer of the material or from published tables. Synthetic transmittance effects can generally be defined with just a few parameters for opacity, index of refraction and so on. Measured geometry typically comes from range scanners such as CyberWare or DeltaSphere. Synthetic geometry usually comes from artists working with modeling software such as 3D Studio Max or Maya. Measured and synthetic surface appearance, the subject of this dissertation, will be covered in much greater detail.

1.1. Surface Appearance

The most basic notion of a surface's appearance is its color. A surface's color comes from the fraction of light of each wavelength that it reflects. So when discussing appearance,

we will speak of *reflectance*, rather than color. The reflectance at a point on a surface usually varies by the direction to the light and the direction from which it is viewed. The light from every incoming direction scatters out in a hemispherical distribution. Although the scattering distribution can be arbitrary, most surfaces scatter light in simple ways. For example, a highlight is caused by light reflecting most strongly toward one particular direction. The function that expresses the reflective scattering distribution for all incident and exitant directions is called the *bi-directional reflectance distribution function*, or BRDF (Nicodemus 1970).

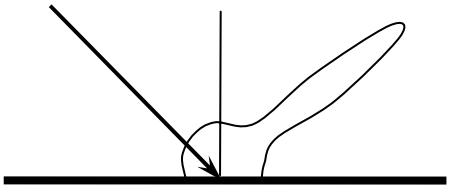


Figure 1.1: *The BRDF is a 4D function, but 2D slices of it can be graphed for a given incoming direction. This polar plot shows that some light scatters in all directions, but the majority scatters forward near the reflection direction.*

For computer graphics, BRDFs are typically represented by a model with a few parameters for capturing the reflectance properties of typical surfaces. Synthetic BRDFs are created by artists choosing parameters for these models. BRDFs are measured using reflectance measurement devices, such as the *gonioreflectometer*, to be described in Chapter 3, that yield large tables of reflectance values. These tables may optionally be fit to an empirical BRDF model, or to a BRDF representation consisting of basis functions.

Just as reflectance usually varies angularly, it usually varies from point to point over the surface. In computer graphics we refer to this spatial variation of color as *texture*. Texture is usually represented as an image mapped over the surface. Synthetic textures are created by artists using paint programs, and measured textures come from photographs.

But there are two problems with this approach. First, a texture map stores a simple color at each point, not the point's BRDF. Second, a camera measures the radiance leaving the surface, rather than the reflectance of the surface, so by itself it cannot measure the whole of a surface's appearance.

1.2. Combining BRDF and Texture

To properly treat surface appearance requires a device that can measure both over the space of the surface like a camera, and over the incident and exitant directions like a gonioreflectometer; and a representation that can store reflectance over space as well as incident and exitant directions.

Surface appearance, although a well-studied area, has not had a fully general representation capable of representing both the spatial and directional reflectance detail that yields the observed appearance of a surface. Compare this to surface shape, which can be represented quite generally using a mesh of triangles. An arbitrarily fine triangle mesh is an effective representation for the surface of any solid object.

In this dissertation I will present a unification of BRDFs and textures, yielding the *spatial bi-directional reflectance distribution function*, which I abbreviate SBRDF. The remainder of the dissertation will be centered on demonstrating the following thesis.

> *A spatially and bi-directionally varying surface reflectance function can be measured for real surfaces, compactly represented as a texture map of low-parameter BRDFs, and rendered at interactive rates.*

I have implemented a complete pipeline for processing SBRDFs. I constructed an SBRDF measurement device and used it to acquire tabulated SBRDF data of several different kinds of real surfaces. I specified a simple, flexible SBRDF representation and implemented two methods to fit BRDF coefficients to approximate the tabulated SBRDF data. Finally, I implemented two novel methods of synthesizing images with SBRDF surfaces that are suitable for real-time image synthesis using graphics hardware. The following sections describe these results in detail.

1.3. Measuring Appearance

Chapter 3 describes a method of measuring the SBRDF of real surfaces. The measurement device is made for measuring approximately planar surfaces of up to 30 × 30 cm that can be physically mounted on the device. The surfaces I use as examples include gilded wallpaper, upholstery fabric, gift wrapping paper, plant leaves, and glossy book

covers. I call the device a *spatial gonioreflectometer*, since it measures the BRDF using discrete samples of incident and exitant directions like a gonioreflectometer, but performs these measurements at all points on the surface, yielding spatial variation or texture.

The method is an extension of the practice of using photographs as texture maps, but addresses the challenges presented above – it measures the reflectance at each point on the surface and over the incident and exitant direction domains, and computes reflectance, rather than simply measuring radiance as does a standard photograph. In particular, the device consists of a digital camera to measure the exitant radiance at each point on the surface sample, a calibrated, motorized light to allow the reflectance to be computed from the known irradiance from the light and the measured exitant radiance, and a pan-tilt-roll motor unit to vary the pose of the surface sample relative to the camera and light.

1.4. Representing & Fitting

The representation of surface appearance that I propose is simply to store the parameters of a BRDF at each pixel of a texture map. This is akin to a standard texture map, but rather than storing a simple color at each pixel my representation stores a BRDF. I use the Lafortune BRDF representation (Lafortune, Foo et al. 1997) As Chapter 4 will discuss, this representation captures in just a few coefficients the major phenomena found in reflectance functions of real surfaces, and can be extended to an arbitrary number of coefficients to increase the generality. The chosen number of coefficients is a tradeoff among generality, storage space and rendering time.

The result of using a low-parameter representation is that a single SBRDF is only a small factor larger than a comparable standard texture map, but is able to represent surfaces much more generally and accurately. Chapter 0 also describes two methods to fit the parameters of the BRDF at each pixel to approximate the reflectance samples generated with the spatial gonioreflectometer described in Chapter 3.

1.5. Rendering

Chapter 5 will discuss interpolation of BRDFs and present a proof showing the validity of interpolating tabulated BRDFs and then discuss the problems associated with interpolating BRDF parameters. Since pixel interpolation is one of the most fundamental

operations of computer graphics it is important to understand the implications of interpolating pixels whose data type is a set of BRDF parameters.

Chapter 6 will discuss two methods of rendering novel images with surface appearance represented as an SBRDF. Conceptually, the SBRDF representation is orthogonal to the kind of renderer being used. I have trivially implemented SBRDF rendering in the Utah Real-Time Ray Tracer (Parker, Shirley et al. 1998; Shirley 2000). Figure 6.1 shows an example. I have also implemented two SBRDF shaders for graphics hardware. The first is for illumination with discrete lights and the second is for global illumination represented in environment maps. The environment map implementation advances the capabilities of preconvolved environment map rendering (Kautz, Vázquez et al. 2000) by separating the convolved environment map from the BRDF with which it is convolved, making it suitable not only for a different BRDF at every pixel, as with SBRDFs, but also for different surfaces, each with a uniform BRDF.

Figure 1.2: *Real-time rendered result using graphics hardware. White upholstery fabric, drapes, lamp, and plant leaves use measured SBRDFs. Floor and cherry wood use hand painted SBRDFs.*

Figure 1.3: *Rendered result using environment map rendering method using a simulator of future graphics hardware.*

Chapter 7 presents conclusions and possibilities for future work.

2. SURFACE APPEARANCE

The study of surface appearance begins with an understanding of the way light is quantified, followed by the way light interacts with surfaces and participating media until it reaches the sensor, such as the eye or camera, where the light will be perceived. For purposes of computer graphics, we most often deal with light as rays, using geometric optics, rather than as waves, using physical optics. Physical optics is, however, useful for modeling such effects as diffraction, interference, and polarization. Within ray optics we can treat each wavelength independently, or consider distributions of wavelengths. It is the combination of energy at different wavelengths that defines the ray's color.

Thinking of light as rays we can consider a function $f(x,\vec{\omega})$ representing the light at each point in space x traveling in each direction $\vec{\omega}$. This function has been called the Global Radiance Function (Shirley 1991), the Light Field (Gershun 1936) (Levoy and Hanrahan 1996), and the Plenoptic Function (Adelson and Bergen 1991). An image is a 2D sample of this 5D function. A pinhole camera image may be created by fixing x at the focal point, or pinhole. The radiance over all directions $\vec{\omega}$ at x forms an image. A camera uses a finite size aperture rather than a point at x, and samples f over the subset of directions $\vec{\omega}$ that intersect the camera film. Likewise, an eye uses a finite aperture centered about x, and samples f over the subset of directions $\vec{\omega}$ that intersect the retina.

Considering light sensing using a camera versus using an eye illustrates the issues of sensor response and visual perception. The Global Radiance Function exists independent of any measurement device or observer, and the characteristics of each sensor or observer affect the measured or perceived function values. For example, camera film has a characteristic curve that specifies to what degree the film chemistry responds to energy of each given wavelength. CCD cameras likewise have a characteristic curve specifying the amount of charge collected by the sensels (CCD pixels) for a unit amount of energy at each wavelength.

An eye's response to light likewise varies over different wavelengths, but this is only the beginning of visual perception. Many factors affect the ultimate response a person has to an image. These perceptual issues are studied within many disciplines, including computer graphics, but appearance measurement and image synthesis work can deal directly with the Global Radiance Function, prior to perception by any human observer[1]. *Photometry* is the study of physical properties and quantities related to light that take into account the response curve of a human eye to light at different wavelengths. The study of the same physical properties and quantities independent of the response of a human eye is called *Radiometry* (Palmer 1999). Since surface appearance measurement and image synthesis can be independent of a human observer, I use radiometric calculations and measurements.

2.1. Radiometry

The following discussion of radiometry begins with the definition of several quantities. Shirley, Hanrahan, and Palmer provide useful and clear references on radiometry (Shirley 1991; Hanrahan 1993; Palmer 1999).

Consider the experiment of measuring the Global Radiance Function using a camera with a finite shutter speed, a finite sized aperture, and a finite area CCD. The camera measures energy. **Energy**, measured in joules (J), is represented by the symbol Q. The energy of a single photon striking the CCD is proportional only to its wavelength, so the total energy measured at the camera is the sum of the energy of each individual photon striking the image plane. By taking the derivative of energy with respect to time (corresponding to shutter speed), solid angle (corresponding to aperture), and area (corresponding to CCD area), we can arrive at the quantity the camera measures within the Global Radiance Function.

Power, measured in watts (W), is represented by the symbol Φ. Power is the derivative of energy with respect to time: $\Phi = dQ/dt$. Since power is independent of time it is the quantity used by non-integrating detectors like spot photometers, and continuous sources

[1] Work such as mine that depends on measurement and display devices typically inverts the response curve of the sensors and displays so that the system can perform internal computations in the space of the Global Radiance Function.

like light bulbs. Energy is the integral of power over time, so it is used for integrating detectors such as a CCD camera.

Considering the power at a differential surface area yields **irradiance**, measured in watts/meter^2. Irradiance, E, is the power per unit area incident upon A from a direction perpendicular to A. Or $E=d\Phi/dA$. The power per unit area exiting A is **radiant exitance** or **radiosity**, and is also measured in W/m^2.

Considering the power over just a differential solid angle yields **radiant intensity**, measured in W/sr. Radiant intensity, I, integrated over solid angle, $\vec{\omega}$, is power. Or $I=d\Phi/d\omega$.

Considering the power at a differential area and at a differential solid angle yields **radiance.** Radiance, measured in W/m^2/sr, is represented by the symbol L. $L=d\Phi/d\vec{\omega}\ dA\cos\theta$. Radiance represents the power per unit area perpendicular to the direction of the ray per unit solid angle. The $\cos\theta$ projects the differential surface to the ray direction.

This discussion shows that by considering a small "differential camera" with a differential exposure time, differential area and differential solid angle, we can see that the quantity of the Global Radiance Field or Light Field is radiance, making radiance the quantity used for light transport for image synthesis.

With radiance defined we can revisit irradiance to more clearly see the relationship between the two:

$$E = \int_{\Omega_i} L_i \cos\theta_i d\omega_i \tag{2.1}$$

The factor $\cos\theta_i d\omega_i$ is often called the projected solid angle and represents the projected area onto the base of a hemisphere of a differential area on the hemisphere surface. θ_i is the polar angle – the angle between the normal and $\vec{\omega}_i$.

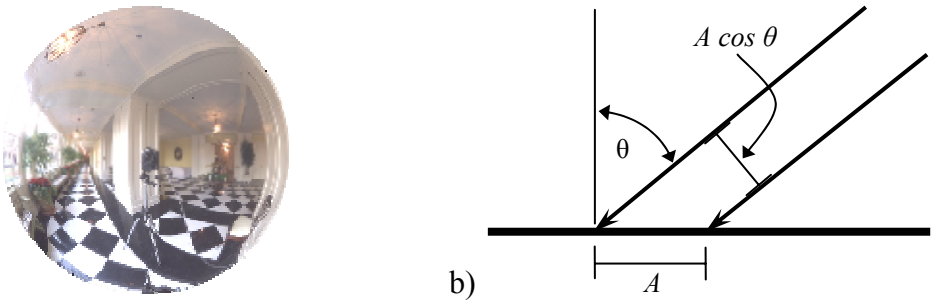


Figure 2.1: *a) An incident radiance hemisphere Ω_i. b) A beam illuminating a surface area A has a cross-sectional area A cos θ for an incident polar angle θ.*

2.2. Spectral Radiance and Tristimulus Color

Radiance and the other five radiometric quantities are scalar values representing the total quantity over all wavelengths. Spectral radiance is the derivative per unit wavelength of radiance. This allows each wavelength to be treated independently, either by discretizing the quantity over wavelength or by treating the spectral radiance as a continuous function parameterized by wavelength. The other radiometric quantities also have analogues parameterized over wavelength. The terms are constructed by prepending the word "spectral" to the quantity's name.

The human eye contains three varieties of photoreceptor cells called *cones* that each have a different response curve to light of differing wavelengths. The response of these three cone types yields the human perception of **color**. Because of the three varieties of photoreceptor cells, three numerical components are necessary and sufficient to describe a color that is ultimately meant for human perception (Poynton 1999).

The Commission Internationale de L'Éclairage (CIE) characterized the response curve (luminous efficiency curve) V(λ) to light at different wavelengths for the hypothetical *standard observer*. This curve and two other curves based on statistics from experiments involving human observers are used as the spectral weightings that result in a color represented with the CIE XYZ tristimulus values. These three primaries can be thought of as axes of a color space. Other color spaces can be constructed using different tristimulus values. Most current CCD cameras and computer displays use tristimulus sensors with high response to red, green, and blue wavelengths. Most computer graphics hardware and software use RGB colors as well. CCDs and computer displays all vary in the precise spectral response of the RGB stimuli, so each has a somewhat different color space.

Colors in one tristimulus color space may be transformed to another tristimulus color space by transforming the color's three-vector by a 3×3 matrix. All tristimulus color spaces have black at the origin, so the transformation is only a shear and rotation. Upon converting to the new color space, any colors with coordinates greater than unity or less than zero are outside the gamut of the device and cannot be accurately reproduced. Only color spaces for which all three dimensions represent spectral weighting functions are called tristimulus color spaces. The hue-saturation-value color space, for example, is not a tristimulus color space.

Luminance, L_v, is the photometric quantity equivalent to the radiometric quantity radiance. Thus luminance can be computed from spectral radiance by integrating the product of the spectral radiance and the response curve of the standard observer:

$$L_v = \int L(\lambda) V(\lambda) d\lambda \tag{2.2}$$

Luminance is the achromatic perceived brightness of an observed spectral power distribution. Luminance is the Y coordinate of the CIE XYZ tristimulus color space. Because of this, the luminance of a color in another tristimulus color space such as RGB can be computed as the projection of the color onto the Y axis represented in the RGB space. For example, for a pixel in the RGB space used by Sony Trinitron phosphors, luminance is

$$L_v = \begin{bmatrix} 0.2582 & 0.6566 & 0.0851 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.3}$$

2.3. Reflectance and BRDF

Surfaces reflect light. Reflection is the process by which light incident upon a surface leaves a surface from the same side (Hanrahan 1993). Consider an incident hemisphere consisting just of light incident from a differential solid angle about a direction ω_i. The amount of light reflected in some other direction ω_r is directly proportional to the irradiance of this hemisphere. This proportion is the *bi-directional reflectance distribution function*, abbreviated BRDF[2]. With the definition of irradiance from Equation (2.1), the BRDF is defined as

[2] (Marschner 1998) is a useful reference regarding the BRDF.

$$f_r(\omega_i \to \omega_r) = \frac{L_r(\omega_r)}{E(\omega_i)} = \frac{L_r(\omega_r)}{L_i(\omega_i)\cos\theta_i d\omega_i} \tag{2.4}$$

From this equation we can determine the units of the BRDF. The units of radiance in the numerator and denominator cancel, the $\cos\theta_i$ is unitless, and the units of the solid angle $d\omega_i$ are steradians, so the BRDF has units of inverse steradians. As such the BRDF can assume values from zero to infinity. This becomes more clear by thinking of the BRDF as the concentration of radiant flux per steradian.

The BRDF is used in the *reflectance equation*, which computes the exitant radiance leaving the surface in a direction ω_r given an incident hemisphere $L_i(\omega_i)$:

$$L_r(\omega_r) = \int_{\Omega_i} f_r(\omega_i \to \omega_r) L_i(\omega_i) \cos\theta_i d\omega_i \tag{2.5}$$

Image synthesis is the process of computing the reflectance equation at all surface points visible in the synthesized image. This computation is the topic of Chapter 6. More generally, image synthesis also includes the computation of $L_i(\omega_i)$, the incident radiance hemisphere at each point, typically due to light reflecting off other surfaces. This global problem is expressed by the *rendering equation* (Kajiya 1986), and the solution of this equation is called *global illumination*.

The BRDF has a number of interesting properties.

1. All real BRDFs follow *Helmholtz reciprocity*, which states that the BRDF is equivalent when reversing the directions:

$$f_r(\omega_i \to \omega_r) = f_r(\omega_r \to \omega_i) \qquad \forall \omega_i, \omega_r \tag{2.6}$$

2. *Conservation of energy* states that the total energy reflected at a point cannot be greater than the total energy incident at that point:

$$\int_{\Omega_r} \int_{\Omega_i} f_r(\omega_i \to \omega_r) L_i(\omega_i) \cos\theta_i d\omega_i \cos\theta_r d\omega_r \le \int_{\Omega_i} L_i(\omega_i) \cos\theta_i d\omega_i \tag{2.7}$$

For an incident hemisphere that only contains light from a single direction with an infinitesimal solid angle we can simplify this to

$$\int_{\Omega_r} f_r(\omega_i \to \omega_r) \cos\theta_r d\omega_r \le 1 \qquad \forall \omega_i \tag{2.8}$$

3. Another attribute BRDFs may possess is *isotropy*. Isotropic BRDFs yield the same reflectance for a given ω_i, ω_r pair when the surface is rotated about its normal:

$$f_r\left(R_\phi\left(\omega_i\right) \rightarrow R_\phi\left(\omega_r\right)\right) = f_r\left(\omega_i \rightarrow \omega_r\right) \qquad \forall \omega_i, \omega_r, \phi \tag{2.9}$$

where $R_\varphi(\omega)$ is a rotation by φ of ω about the surface normal. Many smooth surfaces are isotropic. Surface points with a preferred direction are *anisotropic*. Many fabrics, such as the white fabric used in my examples, are anisotropic. Brushed metal is another common anisotropic surface. Anisotropic surfaces have a distinguished direction such as the direction of the brush stroke or direction of the threads. The angle β will be used to represent this direction of anisotropy relative to the principle directions of the surface. For anisotropic surfaces the ω_i, ω_r pair must be defined in terms of this direction.

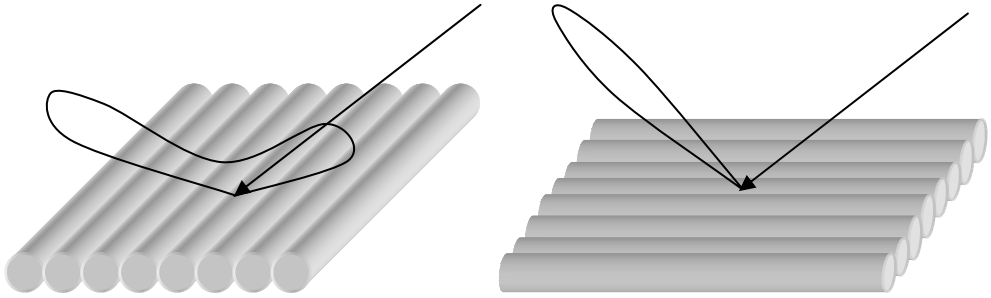


Figure 2.2: *An anisotropic surface consisting of mirror-reflecting cylindrical microgeometry can cause strong retroreflection for incident light perpendicular to the groove direction, but only forward reflection for incident light parallel to the groove direction.*

4. Many isotropic BRDFs exhibit *bilateral symmetry*. The plane containing the incident direction ω_i and the surface normal is called the *incidence plane*. Isotropic surfaces typically but not necessarily scatter energy symmetrically to the left and right of the incidence plane.

2.4. BRDF Models and Surface Properties

Reflection is often thought of in terms of scattering – in what directions do photons incident on a surface from a given direction scatter? Surface scattering behavior is a complex topic modeled in a variety of ways. This discussion will ignore scattering effects related to transmission. Thinking of light–surface interaction in terms of scattering already ignores spectral and polarization effects as well as fluorescence and phosphorescence.

Scattering is usually expressed in terms of microfacet distributions. Surfaces can be thought of as consisting of microscopic reflectors that individually follow the law of reflection, which states that a ray's reflection direction equals the reflection about the normal

of its direction of incidence. The normals of the microfacets have some statistical distribution relative to the normal of the surface. Most BRDFs can be thought of in terms of a few different modes of scattering.

Lambertian diffuse reflection arises when the distribution of microfacets is such that light is scattered equally in all directions. In other words, the BRDF is a constant. For a perfect diffuse reflector, the BRDF is $1/\pi$. *Mirror reflection*, sometimes called specular reflection or pure specular reflection, occurs when all microfacets have a normal parallel to the surface's normal, so all incident light obeys the law of reflection. A mirror has a BRDF inversely proportional to $\cos\theta_i$, so as the incident angle becomes more grazing the BRDF increases while the projected solid angle decreases equivalently, keeping the exitant radiance in the reflected direction a constant proportion of the incident radiance.

Most surfaces are not idealized in either of these two ways, yielding a scattering mode variously called specular, rough specular, directional diffuse, or glossy reflection. In this work I will use the term specular reflectance, but refer to the resulting visual effect as glossy reflection.

The prevailing mathematical model for microfacet-based BRDFs comes from Torrance et al. (Torrance and Sparrow 1967; Blinn 1977; Cook and Torrance 1981; He, Torrance et al. 1991) and appears in its modern form as

$$f_r = \frac{DGF}{4\cos\theta_r \cos\theta_i} \tag{2.10}$$

D is the microfacet distribution and is represented using any standard distribution function such as a Gaussian, or an exponentiated cosine.

G is geometric attenuation term, which accounts for surface microfacets occluding or shadowing other microfacets.

F is the Fresnel term, which is related to a surface's index of refraction and extinction coefficient. Shirley (Shirley 1991) provides a full treatment of the Fresnel equations. The Fresnel term approaches one at grazing angles, causing any surface to approach a perfect mirror at a grazing angle. In fact, as with a perfect mirror, the BRDF approaches infinity at a grazing angle, while the projected solid angle approaches zero. In this way the Fresnel term

modulates between appearance of the surface under more standard viewing angles and its mirror appearance at grazing angles.

The microfacet distribution models do not handle all classes of surfaces. Many surfaces are layered, with different types of scattering behavior occurring in the different layers. Some BRDF models are based on layers, such as Hanrahan and Krueger (Hanrahan and Krueger 1993) and Debevec et al. (Debevec, Hawkins et al. 2000). A more recent approach is to treat subsurface scattering as a light transport problem, as in Jensen et al. (Jensen, Marschner et al. 2001).

The apparent color of a surface at each point is determined by the pigment colors within the surface. A complex example is car paint, which often has multiple colors of paint flakes suspended in one layer, with a clear coating in another layer.

2.4.1. Mesostructure

The fact that BRDF models for computer graphics often use a microfacet distribution suggests that the BRDF is a measurement of the surface appearance *at a particular scale*. BRDFs are measured over a very wide range of scales, from the BRDF of semiconductor layers to the BRDF of forests as seen from satellites. The full appearance of a surface is usually represented in different ways at different scales. These scales can be seen as band pass filters for the geometry of the surface. The BRDF represents all reflection occurring below some sampling threshold. It is geometry below this scale that is represented as a microfacet distribution, without implying an actual microfacet size. For example, trees and leaves are microfacets in many forest BRDF models (Verhoef 1984).

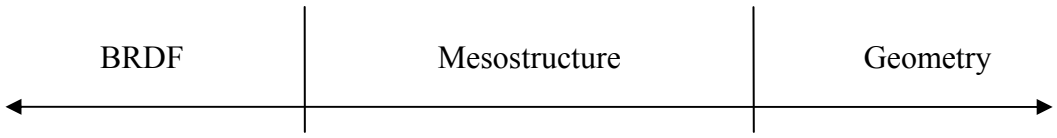


Figure 2.3: *A surface's geometry may be thought of as information in multiple frequency bands.*

At the other end of the spatial frequency spectrum is the surface's shape, which is represented using geometry. Between the surface shape and the BRDF lies the *mesostructure*. The boundary between the surface shape and the mesostructure bands is determined by the application based on the system's ability to render mesostructure effects and the nature of the

surfaces. Smooth surfaces like teapots often have a null mesostructure band. Mesostructure for the forest example could be stored in a terrain height field. For surfaces seen from a distance on the order of one meter, mesostructure could be what is commonly thought of as *texture*. Stucco is one example.

Two common representations of the mesostructure are *bump maps* and *displacement maps*. Bump maps represent the facet orientation at each point on the surface. The BRDF is evaluated at each point with ω_i and ω_r relative to this local facet orientation. Displacement maps are represented as height fields relative to the surface geometry and can be used to render correct surface self-occlusion, self-shadowing, and geometric perturbation, visible especially at the silhouette. Becker and Max (Becker and Max 1993) discuss the concepts involved in blending from one representation to another at run-time. Cohen et al. (Cohen, Olano et al. 1998) provide an algorithm for simplifying model geometry while storing the higher frequencies in a bump map.

2.5. Texture

Just as bump and displacement maps represent the mesostructure geometric frequencies over the surface, the BRDF may also vary over the surface and be stored in a texture map. In computer graphics, *texture* refers not to the geometric mesostructure, but to spatial variation of the diffuse component of the BRDF of a surface, or more generally to spatial variation over a surface of any appearance parameter.

This change in terminology results from the original texture mapping work of Catmull (Catmull 1974) in which Catmull rendered the texture of stone castle walls and stored an image of the walls (essentially spatially-varying diffuse reflectance) as a means of representing the mesostructure or texture of the walls. Thus, texture maps were originally used not to store mesostructure, but to store an image that would give the appearance of detailed mesostructure when rendered.

From then, texture maps, subsuming all maps parameterized over a surface or other domains such as environment maps (Green 1986), have become one of the most fundamental content representations used in computer graphics. Indeed, the global radiance function, or light field, as a whole may be stored in texture maps and rendered directly (Gortler,

Grzeszczuk et al. 1996; Levoy and Hanrahan 1996). Image-based rendering has grown to encompass any image synthesis that includes images as input. Since this includes all forms of texture mapping, I will narrow the term texture mapping to only include maps parameterized over a surface. These maps may either be stored as 2D images, or computed procedurally.

Just as the BRDF parameterizes a surface point's reflectance over the incident and exitant directions, texture maps parameterize reflectance over the spatial dimensions of the surface. This parameterization is defined as part of the model, usually by specifying coordinates of the map at each vertex.

2.5.1. Sampling and Signal Reconstruction

A renderer samples the texture map as part of the process of synthesizing an image of the scene. The sampling of the texture does not generally correspond directly to the pixel centers of the map. This is an instance of the signal sampling and reconstruction problem of digital imaging. Posed in terms of rendering a synthetic image of a scene containing textured surfaces, the problem is divided into four parts:

1. Reconstruct the continuous texture signal from the 2D array of point samples
2. Remap the continuous signal to the synthetic image space
3. Low pass filter the remapped signal
4. Resample the reconstructed (continuous) texture function at the synthesized image pixels.

At present, most renderers except real-time television special effects hardware perform this process in the space of the synthesized image – usually raster order. The renderer inverts the map from texture space to screen (synthesized image) space and samples the location in the texture map corresponding to the screen pixel. This implements steps 2 and 4 above. Heckbert (Heckbert 1986) provides a survey of techniques for the reconstruction and filtering of texture maps. The key function of step 3 is to remove irreproducible high frequency content from the remapped signal prior to resampling it. This is the fundamental operation of anti-aliasing.

Williams (Williams 1983) implements steps 1 and 3 together by creating a MIP-map (a multiresolution image pyramid) with successively low pass filtered and subsampled instances of the texture map. These are created *a priori* using any finite impulse response

filter. Step 4 is then computed by bilinearly sampling two MIP-map levels and linearly interpolating between the two. This implements a filter kernel of approximately the size of the pixel's pre-image in the texture map. The pre-image is the projection onto the texture map of the pixel's point spread function. Figure 2.4 illustrates texture resampling. The pixel's pre-image is elliptical when considering the pixel to be a radially symmetric point spread function.

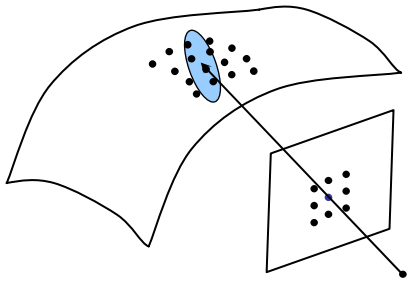


Figure 2.4: *A pixel's pre-image in a texture map is an arbitrary shape that must be approximated when resampling the map.*

At grazing angles the pixel's pre-image can become arbitrarily anisotropic, or long relative to its width. Trilinear MIP-mapping uses an isotropic filter kernel, making the synthetic image excessively blurry at grazing angles. By blending multiple trilinear samples from the MIP-map, an anisotropic ellipse can be approximated, but at arbitrary computational cost (or fixed cost for limited quality).

Summed area tables (Crow 1984) are a different method of prefiltering a texture map for constant time resampling. Each table entry contains the definite integral over the domain of the texture from (0,0) to that table entry. An arbitrary axis-aligned rectangle approximating a pixel's pre-image may then be sampled in constant time by sampling the summed area table at the four corners of the rectangle, similar to computing a definite integral over a 2D domain.

The Elliptical Weighted Average filter (Greene and Heckbert 1986) samples an arbitrarily oriented elliptical pre-image using several samples distributed over the ellipse. This gives very high quality resampling, but for a potentially unbounded computational cost. Wolberg (Wolberg 1990) covers image warping and resampling in general.

2.6. Representing Surface Appearance

Almost all real surfaces have some amount of reflectance variation over the surface. Likewise, most real surfaces have some amount of reflectance variation over the incident and exitant hemispheres. For this reason, texture mapping and BRDFs are equally fundamental to representing a surface's appearance.

I propose the term *spatial bi-directional reflectance distribution function*, SBRDF, to apply to the full reflectance function of a surface, including only the BRDF frequency band, not the mesostructure or geometry. This function is six-dimensional and is parameterized over the surface, like a texture map, and over the incident and exitant direction sets, like a BRDF:

$$f_{sr}\left(u, v, \omega_i, \omega_r\right) \tag{2.11}$$

Many similar terms exist for similar or the same concept as the SBRDF. For example, "shift-variant BRDF" and "space variant BRDF" are the same as the SBRDF. While shift-variant filters and space variant filters, from which the terms arise, are functions with parameters varying over a domain as is the SBRDF, the problem I see with applying these adjectives to BRDFs is they are always filters that operate *in that same domain*, whereas the analogous portion of an SBRDF is a BRDF, which has a bi-directional domain, not a spatial domain.

The "bi-directional texture function" of Dana et al. (Dana, Ginneken et al. 1999) is often considered to be the same as the SBRDF, but the BTF, as originally introduced, represents bi-directionally varying image statistics, without the ability to uniquely address any point on the surface. Section 3.1 gives more detail about the BTF.

The term "spatially varying bi-directional reflectance distribution function," a term used in discussion, but not so far in the literature, means the same as the SBRDF, but does not connote the fact that the spatial and bi-directional detail are both of equal importance and that a single function varies over the 6D domain.

Debevec et al. (Debevec, Hawkins et al. 2000) refer to the "reflectance field" of a human face. This term would be suitable for the general SBRDF. In practice, Debevec used a specialized BRDF function and did not vary the entire BRDF over the surface.

Finally, (Malzbender, Gelb et al. 2001) introduced the "polynomial texture map," a 4D function that represents the 4D slice of the SBRDF that corresponds approximately to holding the exitant direction constant in the normal direction. Sections 3.1 and 4.1 discuss the above previous work in greater detail.

2.6.1. Spatial vs. Angular Detail

In attempting to unify texture maps, which are spatial reflectance distribution functions, and BRDFs, bi-directional reflectance distribution functions, we should consider the differing properties of the two. (Functions with only non-negative values can be called distribution functions.)

BRDFs of real surfaces are nearly always continuous. Consider the highlight made by a point light source reflecting off a smooth surface of uniform BRDF. The highlight nearly always has a smooth falloff from peak to tail. For this reason microfacet distributions can safely be modeled as Gaussian distributions. Counter-examples might include manufactured materials with discontinuous microfacet distributions such as micro-mirror devices. Ashikhmin et al. (Ashikhmin, Premoze et al. 2000) present a microfacet-based BRDF generator that can generate discontinuous BRDFs.

In addition to being C^0 continuous, most BRDFs exhibit higher-order smoothness. This makes BRDFs nicely representable as a sum of smooth basis functions, for example Zernike polynomials (Koenderink, Doorn et al. 1996), spherical harmonics (Westin, Arvo et al. 1992), wavelets (Lalonde and Fournier 1997), or generalized cosine lobes (Lafortune, Foo et al. 1997). The number of basis functions required for a given goodness of fit depends on the frequency content of the BRDF and the flexibility of the basis functions.

One potentially useful way to choose the boundary between the BRDF and mesostructure bands of a surface's geometry is to attempt to place all directional reflectance discontinuities in the mesostructure layer, ensuring the bi-directional continuity of the BRDF at each point. This is desirable because approximating BRDF discontinuities with a finite number of smooth functions always leads to angular smoothing of the discontinuity.

The texture (spatial reflectance) of a surface generally has quite sharp discontinuities. For example, printed or stenciled surfaces contain sharp discontinuities. Natural surfaces

such as animal coats and leaves also have sharp discontinuities. Because of the prevalence of high frequencies and discontinuities in the spatial domain, textures are typically stored discretely as raster images. However, these images are often represented using sums of smooth functions as with the discrete cosine transform or the wavelet transform. When doing so, it is important to maintain high frequencies near discontinuities to avoid artifacts.

2.6.2. Combining Spatial and Angular Detail

Since spatial and angular detail are very different in nature, they are usually combined only in *ad hoc* ways, although each by itself is formalized and well understood.

Combining the spatial detail of texture mapping with the angular reflectance variation of BRDFs is not entirely new. Software renderers often store BRDF parameters in texture maps, usually while holding some parameters constant over the surface or generating them procedurally (Cook 1984; Hanrahan and Haeberli 1990). Likewise, the standard shading models for graphics hardware essentially vary the diffuse component of the Phong BRDF model (Phong 1975) in a texture map, while holding the specular parameters constant (Neider, Davis et al.). Kautz and Seidel (Kautz and Seidel 2000) store all the parameters of simple BRDF models at each pixel. Bi-directional Texture Functions (Dana, Ginneken et al. 1999) store tabulated reflectance values over direction and approximate spatial location (see Section 3.1).

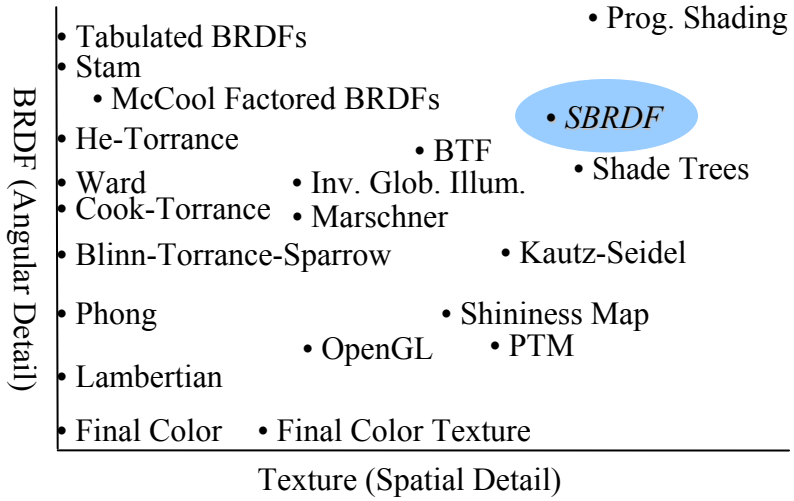


Figure 2.5: *Surface appearance representations are typically strong in either spatial or angular reflectance detail but not generally both. The vertical axis represents angular detail and the horizontal axis represents how much angular detail is allowed to vary spatially.*

Figure 2.5 shows several surface appearance representations, emphasizing representations that focus on reflectance; i.e., ignoring those that focus on mesostructure. Mesostructure is orthogonal to most of these representations. The vertical axis ranks representations by their ability to represent increasingly general BRDFs. The horizontal axis ranks representations by the amount of the BRDF that is allowed to vary spatially. For example, Inverse Global Illumination (Yu, Debevec et al. 1999) represents the diffuse component of the BRDF in a texture map, but holds the parameters of a Ward BRDF constant over each polygon. Likewise, "shininess mapping" simply uses the standard Phong model of graphics hardware, but allows the specular albedo to be stored in a texture map as well as the diffuse albedo.

2.6.3. Representing the SBRDF

Many digital images fall into two broad categories – those representing radiance and those representing reflectance. Just as high dynamic range radiance maps (Debevec and Malik 1997) dramatically increase the usability of images representing radiance by simply using a pixel representation adequate to the properties of radiance I believe that using a BRDF as a pixel representation will similarly increase the usability of texture maps representing reflectance.

By storing the parameters of a BRDF at each pixel, texture maps can represent the spatial bi-directional reflectance distribution function. This simple representation is spatially discrete like a standard texture map, but bi-directionally continuous like a BRDF. The Lafortune representation, described in Section 4.2, requires only a few coefficients to achieve a good approximation of most BRDFs, but can use an arbitrary number of coefficients for increased accuracy. A texture map with Lafortune BRDF parameters at each pixel has only a small storage size overhead relative to a standard RGB texture map. As with the conceptual SBRDF function, I refer to this representation as an SBRDF.

Allowing total variation of the BRDF over the surface, while using a general BRDF representation consisting of basis functions places the proposed representation high on both axes of Figure 2.5. Of representations known to me, only programmable shading allows more angular reflectance detail to vary over the surface. Likewise, only tabulated BRDF

representations and models accounting for spectral effects such as Stam (Stam 1999) offer more angular expressivity than the general basis function representation I employ.

I believe that two benefits will come from storing at each texel a unique BRDF, rather than just a few components of the otherwise-uniform BRDF. First, a general BRDF at each pixel will allow the realism of rendered results to scale with the quality of measured data without the representation limiting the quality. This is because no parameters are constrained in their variation over the surface. The chosen BRDF representation at each pixel may need to improve over time, however. I accommodate this to some extent by using a BRDF representation that scales in quality with the number of coefficients. Second, a unique BRDF at each texel provides a surface representation that can be used as an interchange format between measurement systems, paint and modeling systems, and rendering systems. This should enable the creation and sharing of SBRDF libraries, as is done with 3D models and standard texture maps today.

3. MEASURING SURFACE APPEARANCE

The spatial bi-directional reflectance distribution function can be measured from real surfaces. This chapter describes a device I designed and, with help, built to perform these measurements. The device consists of a pan-tilt-roll motor unit that holds the surface sample, a calibrated light on a motorized rail, and a stationary calibrated CCD camera. The design of the device is original, but not groundbreaking. However, a few simple extensions to existing systems made the device able to capture the full SBRDF of real surfaces, which has not been done before. I have acquired samples of gilded wallpaper, plant leaves, upholstery fabrics, wrinkled gift-wrapping paper, glossy book covers, and other materials.

In this chapter I will describe other appearance measurement devices, focusing on those that capture spatial variation. I will describe the device I built and compute its resolution specifications as a function of the specifications of its components. In particular I will show how to compute the necessary sampling density in the BRDF space to adequately measure features of a given size. I will review the instrumentation practices I followed, the control and use of the device, and demonstrate results for materials I measured.

3.1. Previous Work

A great deal of recent work applies to representing real world materials for computer graphics. Light Fields (Levoy and Hanrahan 1996; Wood, Azuma et al. 2000; Chen, Bouguet et al. 2002) and Lumigraphs (Gortler, Grzeszczuk et al. 1996) represent the Global Radiance Function directly by keeping a dense sampling of the light rays in a scene, reconstructing images simply by selection and interpolation from this database. These approaches store *exitant radiance*, rather than *reflectance*, so they do not handle changes in illumination.

To extend this database approach to a combination of all incident directions and locations and exitant directions and locations, we arrive at an eight-dimensional entity called the reflectance field by Debevec (Debevec, Hawkins et al. 2000). No device has been built to acquire the eight-dimensional reflectance field. Two of the dimensions are only needed to

account for global effects such as indirect illumination and subsurface scattering. Without these, we are left with the six-dimensional SBRDF. Measuring reflectance instead of radiance gives the rendering system the freedom to convolve the reflectance function with arbitrary incident light to synthesize new images of a scene. Chapter 2 describes reflectance and BRDF. Chapter 6 describes image synthesis given the BRDF and incident light.

The traditional device for acquiring BRDFs of real materials is the gonioreflectometer, a specialized device that positions a light source and a sensor relative to the material (Foo 1997). These devices obtain a single sample for each light–sensor position and are therefore relatively slow. Imaging devices such as CCD cameras sacrifice spectral resolution but obtain a large number of samples simultaneously. Two methods are used to cover a wide variety of different angles from a single image of a homogeneous material. One is to use curved materials (Lu, Koenderink et al. 1998; Marschner, Westin et al. 1999), while the other is to use optical systems such as wide-angle lenses or curved mirrors (Ward 1992; Karner, Mayer et al. 1996).

More recent work focuses on spatially varying materials. Often, the diffuse component is captured at a high spatial resolution, while the specular component is either constant across a polygon (Yu, Debevec et al. 1999), or interpolated from per-vertex information (Sato, Wheeler et al. 1997). Debevec et al. (Debevec, Hawkins et al. 2000) describe a method for fitting a specialized reflection model for human skin to photographs of human faces. Both specular and diffuse parameters of the reflection model can vary sharply across the surface, but other parameters like the de-saturation of the diffuse component at grazing angles are constant, and only apply to human skin.

Lensch et al. (Lensch, Kautz et al. 2001) compute a spatially varying BRDF of an object with known geometry. They assume that many points over the surface share the same BRDF and by only handling isotropic BRDFs they require only about 15-30 images per object. Their work focuses on finding a few basis BRDFs and representing each texel as a linear combination of these bases, with small per-texel detail. Thus, their system only applies to surfaces with a few different isotropic BRDFs.

Dana et al. (Dana, Ginneken et al. 1999) propose the Bi-directional Texture Function (BTF). They constructed an acquisition system similar to the spatial gonioreflectometer and

use it for acquiring two kinds of data – directionally varying image histogram statistics and the aggregate BRDF of the surface. Their data are nearly suitable for SBRDF acquisition except that their system uses a smaller sample size (10 cm × 10 cm), lower spatial resolution (640 × 480), and lower angular resolution (205 fixed poses). The poses were chosen to only sample the isotropic BRDF domain. But more importantly, the data are not spatially registered (see Section 3.6). The data are useful for reducing each pose to statistics (histogram or reflectance) and smoothly varying these statistics over the bi-directional domain, but are not quite adequate for fitting different BRDFs at each texel, and the authors do not do so. Except for the registration issue, a BTF could be thought of as a tabulated SBRDF. Liu et al. (Liu, Shum et al. 2001), however, did register some samples from this database using image correlation. They estimated height fields for the surfaces and implemented a directionally aware texture synthesis algorithm for generating surfaces with statistically similar mesostructure.

Polynomial Texture Maps (PTMs) (Malzbender, Gelb et al. 2001) represent a four-dimensional subspace of the SBRDF by holding the exitant direction constant (approximately in the normal direction) at each pixel and varying the incident light direction over the hemisphere. PTMs fit a bi-quadratic polynomial to the exitant radiance in the normal direction parameterized over the incident hemisphere at each pixel. This provides very compact storage and the ability to arbitrarily relight the polygon. PTMs are not typically used to texture arbitrary 3D geometry, but can do so if the exitant radiance in the normal direction is a good estimator of the exitant radiance in all directions. This assumption holds for Lambertian diffuse BRDFs and for the broader class of BRDFs that are not a function of the exitant direction.

The final size of an SBRDF in my proposed representation is approximately equivalent to that of a PTM, but the SBRDF additionally allows the variation in exitant direction required to render surfaces of arbitrary spatially varying BRDF on arbitrary geometry. As with the SBRDF, the PTM is spatially discrete but bi-angularly continuous. The SBRDF and PTM share the characteristic of angularly smoothing surface self-occlusions and self-shadows. However, since the SBRDF uses a pixel representation specifically designed to match the behavior of light scattering, I expect it to better fit the measured data than does a PTM.

Since the BRDF is a 4D function, the measurement device must have at least four degrees of freedom. However, most measurement systems constrain the device to three degrees of freedom and only measure isotropic BRDFs. Of the above approaches, only Ward (Ward 1992) and Karner et al. (Karner, Mayer et al. 1996) measure anisotropic BRDFs, and both require manually repositioning the camera or light over two dimensions of the domain to do so. The PTM system and the Debevec et al. (Debevec, Hawkins et al. 2000) system only acquire a 2D slice of the BRDF by varying the light source over two dimensions while fixing the camera position. The two camera sensor dimensions are used for spatial variation.

3.2. The Spatial Gonioreflectometer

I have extended the concept of the gonioreflectometer (Foo 1997), to spatially varying surfaces. The *spatial gonioreflectometer* I designed, shown in Figure 3.1, acquires image data of real planar materials up to 30 × 30 cm in area. The planar constraint simplifies registration of poses and prevents large-scale self-occlusion and self-illumination.

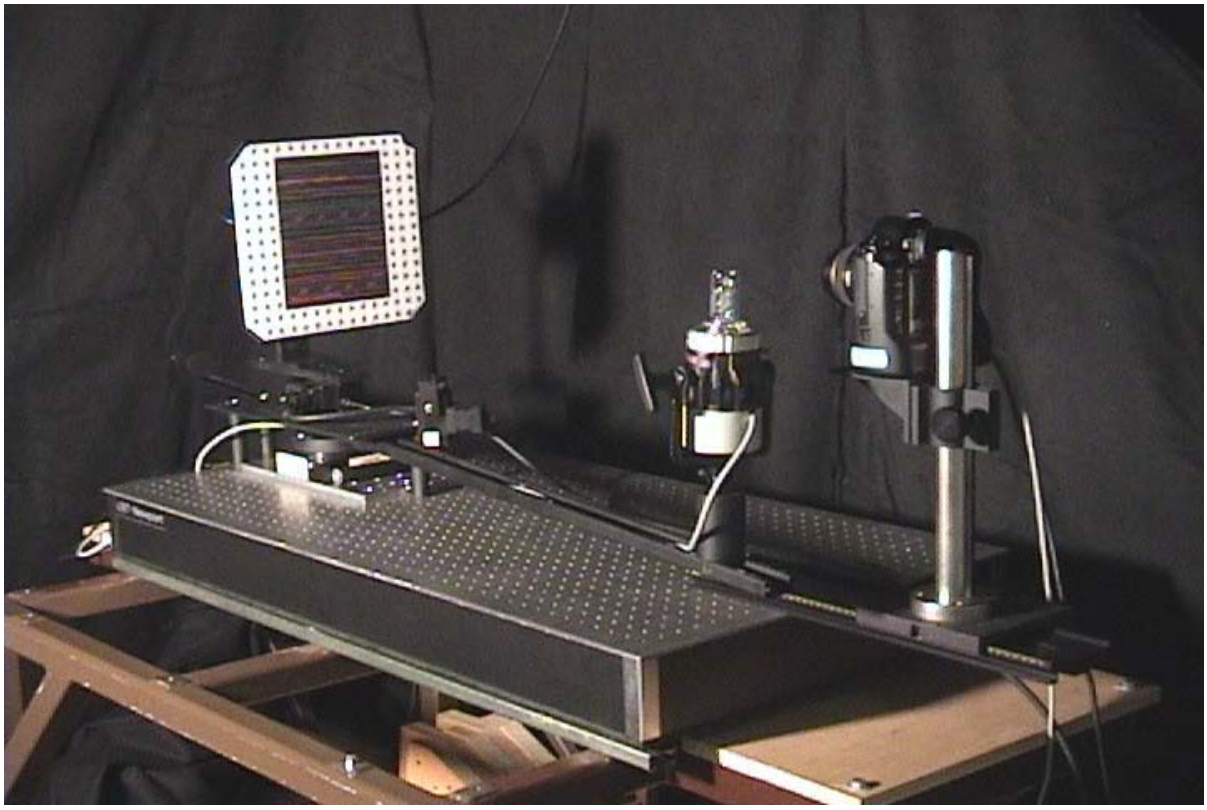


Figure 3.1: *The spatial gonioreflectometer, including motorized light, camera, pan-tilt-roll unit, and sample material with fiducial markers.*

The samples are mounted using spray adhesive to a sheet of foam core with fiducial targets printed at known locations around the perimeter, and then attached to a tilt-roll motor, which is mounted via an adjustable bracket to a panning motor. The light rotation motor is mounted directly under the panning motor. The light is affixed to an adjustable rail, and usually positioned one meter from the center of rotation. All components are mounted on an optical bench for rigidity and stability. The light source travels in the plane of the optical bench on an arc from ten to 175 degrees from the camera. The light and camera obviously cannot be exactly co-aligned. The four motors provide the four degrees of freedom necessary to acquire an anisotropic BRDF.

The workflow for acquiring an SBRDF begins by mounting the surface material sample to the acquisition apparatus. The motors move to each camera-light pose sequentially and pause while a computer-controlled digital camera photographs the material. Once all images have been acquired they are converted to an intermediate representation consisting of registered, rectified, high dynamic range images, with BRDF values being computed from the pixel radiance values. A BRDF representation is then fit to the hundreds or thousands of bi-directional reflectance samples at each pixel of this intermediate representation, yielding the final SBRDF.

The camera images are uploaded to the computer in real time, but require about three seconds each to transmit. On the last image per pose the motors can move while the image uploads. The total time per pose is between five and fifteen seconds. An entire acquisition experiment of 300 to 8000 samples requires from 45 minutes to 36 hours. Two hours is typical, and the entire process is fully automatic. The storage for the camera images for a single material ranged from 600 MB to 30 GB[3].

[3] The camera's file format uses lossless JPEG compression stored within a TIFF file.

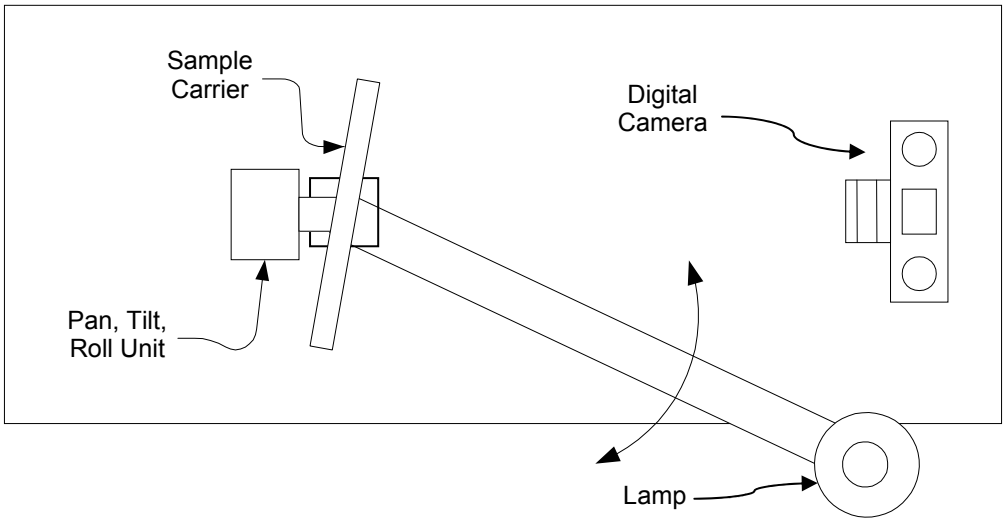


Figure 3.2: *Diagram of acquisition device. The device includes a stationary digital camera, a tilt-roll motor unit with the planar sample attached, a pan motor attached to the tilt-roll unit, and a calibrated light on an adjustable rail that swings 170˚ within the plane of the optical bench.*

3.2.1. Resolution Requirements

The achievable measurement results depend greatly on the characteristics of the motors and camera used and the dimensions of the device, such as the size of the planar sample and the distance from the camera to the sample. My goals for the device were these:

1. acquire all SBRDFs, even anisotropic SBRDFs, completely automatically without having to manually move components,
2. have repeatable enough motor positioning that error in estimated vs. actual light and camera positions would be insignificant,
3. acquire planar samples at least as large as the repeat period of typical upholstery fabrics and wall papers (I chose 30 × 30 cm),
4. have the light and camera close enough to the sample that the direction to each would vary significantly (about 10˚) across the sample, and
5. have enough camera pixels per mm on the surface to resolve individual threads.

The first two goals mainly depend on the choice of motors and the latter three depend mainly on the choice of camera. The following sections discuss each component, including an analysis of the resolution-related design goals.

3.3. Motors

Since the BRDF is a 4D function, the device must have at least four degrees of freedom in order to measure it. My device holds the camera fixed, moves the light, and moves the planar sample with three degrees of freedom. This achieves design goal 1. Goal 2 requires more analysis. Two attributes of stepper motors are their *resolution* and *repeatability*, both measured as angles. The resolution is the total arc that the motor can rotate divided by the number of discretely addressable or detectable positions in that arc. Table 3.1 shows the resolution for the two kinds of motors I used. Note that their resolution differs by two orders of magnitude.

Repeatability is the maximum angular difference between multiple movements of the motor to the same position number. This angle is usually orders of magnitude larger than the resolution due to play in the motor mechanics and forces acting on the motor. Displacement due to angular error is proportional to the radius about the axis of rotation. This provides a good way to measure repeatability. I affixed a laser to the motor and placed a planar sheet of paper at a grazing angle to the laser at a distance of about ten meters, and marked the location of the laser spot on the paper. By moving the motor to arbitrary positions and then moving it either forward or backward to the original position number, and noting the location of the laser spot, I measured the repeatability.

To determine the adequacy of motors based on these specifications one likewise computes the repeatability at the extent of the object being rotated. For the pan-tilt-roll unit this means measuring error at the corner of the sample carrier, a distance of $152\sqrt{2}$ mm from the tilt motor center. The light motor error must be estimated at the end of the one meter long rail. Table 3.1 shows these estimates. I mounted the light on a manually adjustable one-meter rail attached to a Compumotor 6000 Series[4] motor. I used two Directed Perception, Inc. Model PTU-46-70[5] for the pan and tilt-roll motors.

Unlike a robot arm, the error of each motor is not amplified by a long arm followed by another motor since all the motors are close together and the tilt-roll unit's center coincides with the axis of rotation of the pan motor. Likewise, the tilt motor's center

[4] Compumotor Division, Parker Hannifin Corp., Rohnert, CA
[5] Directed Perception, Inc., Burlingame, CA

coincides with the axis of the roll motor. The sample carrier is one inch from the axis of the tilt motor's rotation. This is accounted for in computing the repeatability at the sample extent.

	Directed Perception	Compumotor
Angular Resolution (rad)	2.244E-04	2.792E-06
Angular Resolution (deg)	0.0129	1.600E-04
Angular Repeatability (rad)	0.0137	0.00246
Angular Repeatability (deg)	0.786	0.1411
Resolution at Sample Extent (mm)	0.0476	0.000592
Resolution at 1 meter (mm)	0.2244	0.0028
Repeatability at Sample Extent (mm)	2.97	0.5273
Repeatability at 1 meter (mm)	13.7	2.462

Table 3.1: *Resolution and estimated repeatability of both motors used. The resolution and repeatability are expressed both as angles and as distances for points at the corner of the sample carrier and the end of the light rail.*

Ideally, the motors would have low enough error to provide subpixel repeatability so that when the motors were moved to a specified set of position numbers from multiple starting locations, sample surface points would always map to the same CCD location, within a subpixel tolerance. Section 3.4 will discuss camera resolution, but briefly, the necessary spatial repeatability would be approximately 0.25 mm when the sample plane faces the camera, which is nearly achievable, but the required repeatability increases for grazing angles. Since these motors cannot achieve the necessary level of repeatability, I rely on image registration, covered in Section 3.6, to perform registration adjustments.

3.3.1. Angular Accuracy

Although the motor repeatability is not high enough for subpixel registration accuracy, the motor specifications are more than adequate for goal 2, the computation of directions to the light and camera from the surface. For the light, the key question is how much error in measured radiance will be introduced for a given error in the direction from a surface point to the light? The error will be the worst in the steep regions of the BRDF, such as at highlight edges. Using a standard exponentiated cosine model (Phong 1975),

$$f_n(\theta) = \cos^n \theta \tag{3.1}$$

we can see how much reflectance falloff occurs in these regions. θ is the angle away from the highlight peak. Figure 6.3 shows a graph of $\cos^n \theta$. A fairly sharp highlight with $f_n\left(1^\circ\right) = 0.75$ (i.e., 75% as much energy is reflected 1˚ away from the peak as is directed toward the peak) has a specular exponent n=1888.

What is the angular error of the motor system I use? The error depends on both the light motor and on the pan-tilt-roll unit because both affect the orientation of the surface relative to the light. The pan and tilt motors greatly affect the surface orientation in the plane of the light, camera, and sample center, while the roll motor is perpendicular to this plane. The effects of the two motors are not compound, so the repeatability tolerances can be summed, equaling 1.57˚. The tolerance of the light is added, totaling 1.713˚. $f_{1888}\left(2.713^\circ\right) = 0.120$ – significantly lower reflectance than at θ=1˚.

This illustrates that measurement systems such as mine impose a relationship between the motor tolerances and the minimum surface roughness that can be acquired with a given error tolerance. A BRDF measured by a system with a given angular error tolerance is equivalent to the true BRDF convolved with the probability density function of the error tolerance, measured by a system with zero error. This assumes dense sampling and an appropriate mapping from the BRDF measurements to the continuous BRDF function. This is one obstacle to measuring the BRDF of perfect mirrors.

Many image synthesis systems including computer graphics hardware limit the specular exponent to n=128 or less. $f_{128}\left(1^\circ\right) = 0.981$ and $f_{128}\left(2.713^\circ\right) = 0.866$, which is acceptable, since this light-camera pose will still clearly yield a highlight at this pixel, and in bright regions the viewer of rendered images is less sensitive to brightness changes, even relative changes. Most of the materials I have measured, except the glossy gold magazine cover, have specular exponents significantly less than 128 over the surface.

3.3.2. Sampling Density

The nature of the surface also affects the set of motor positions used to sample it. This sampling density must be high enough that highlights can be reconstructed, and highlights are often narrow, as the above analysis mentions.

The device operator must specify θ_d, the angle between adjacent samples over the incident and exitant hemispheres. The sampling density can be seen as a Nyquist frequency for adequately capturing highlights of a maximum sharpness. I present here a set of equations for computing θ_d from a known or estimated maximum highlight sharpness. The device will then densely sample the surface with samples spaced approximately θ_d apart in angle.

We first choose *p,* the number of samples necessary to fully constrain the fitting of the BRDF equation to the highlight. It is equal to the number of parameters the BRDF representation uses to represent a highlight lobe. For example, the Lafortune representation, described in Section 4.2, represents a highlight using *p=4* parameters. It is usually desirable to choose a larger *p* to overconstrain the parameter fit. Then consider the maximum angle away from the highlight peak for which the highlight is visible within some threshold. For example, find θ_r such that $f(\theta_r)$=0.3. For n=128, θ_r=7.84˚. Let us consider the ratio $g(\theta_r)$ of the highlight's solid angle[6], $\zeta_r(\theta_r)$, to the hemisphere's solid angle, ζ_Ω:

$$g(\theta_r) = \frac{\zeta_r(\theta_r)}{\zeta_\Omega} = \frac{2\pi(1-\cos\theta_r)}{2\pi} = 1-\cos\theta_r \tag{3.2}$$

For the n=128 example material, $g(7.84^\circ)$=0.00934, so slightly less than 1% of the hemispherical area contains the highlight[7]. This is a two-dimensional analysis, but the volume of the highlight in the 3D isotropic BRDF domain and the 4D anisotropic BRDF domain is the same percentage of the domain's total volume as in 2D. One way to see this is to choose light and camera directions sequentially. First choose the incident direction in either a hemispherical subdomain for the anisotropic case or an elevation angle, θ_i, subdomain for the isotropic case. This determines the location of the highlight in the exitant hemisphere. Then when the exitant direction is chosen, the fraction of choices that will sample the highlight is clearly $g(\theta_r)$. Figure 3.3 shows this as an extrusion.

[6] This is a form of the equation for a spherical zone, $A = 2\pi rh$, where r is the radius of the sphere and h is the height along the sphere axis of the latitudinal zone.
[7] This assumes simple Phong highlights and ignores edge effects.

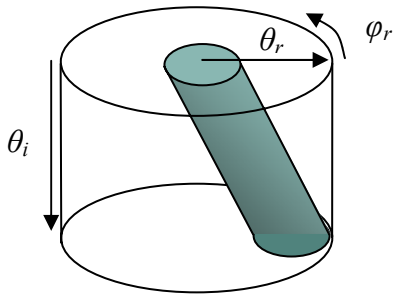


Figure 3.3: *Visualizing the isotropic BRDF space as a cylinder shows that a highlight typically exists for all θ_i, forming an extrusion.*

This brings to light the interesting fact that, while more samples are clearly required to sample the anisotropic BRDF domain to the same density as the isotropic BRDF domain, the total number, m, of uniformly spaced samples required to measure an average of p samples within the highlight *is the same in both domains*.

Given $g(\theta_r)$ and p we can compute

$$m = \frac{p}{g(\theta_r)} \tag{3.3}$$

For the n=128 example material, m=428 samples. In practice this will be an approximation because of the difficulty of uniformly sampling a hemisphere.

To measure an average of p samples within the highlight subtending θ_r, what must the angular sampling density, θ_d, be? The answer depends on the dimensionality. In two dimensions (a hemispherical BRDF subspace with the incident or exitant direction fixed), $\theta_{d,2}$ is the side length in radians of a square patch on the hemisphere with area equal to the highlight area divided by p:

$$\theta_{d,2} = \sqrt{\frac{\zeta_r(\theta_r)}{p}} = \sqrt{\frac{2\pi(1-\cos\theta_r)}{p}} \tag{3.4}$$

For the n=128 example material, $\theta_{d,2}$=6.94˚.

In three dimensions the process is similar. We compute the total volume of the highlight by extruding the 2D highlight area through the domain of the incident direction, the size of which is π/2 rad. The volume per sample is the highlight volume divided by p, and is measured in rad·sr. The side length of this cube yields $\theta_{d,3}$:

$$\theta_{d,3} = \sqrt[3]{\frac{\frac{\pi}{2}\zeta_r(\theta_r)}{p}} \tag{3.5}$$

For the n=128 example material, $\theta_{d,3}$=16.3˚.

And finally, for the 4D anisotropic BRDF, we compute the volume of the highlight by again extruding the 2D highlight area through the incident direction's domain, which is 2π sr. The volume per sample is the highlight volume divided by p. The side length of this hypercube yields $\theta_{d,4}$:

$$\theta_{d,4} = \sqrt[4]{\frac{2\pi\zeta_r(\theta_r)}{p}} \tag{3.6}$$

For the n=128 example material, $\theta_{d,4}$=31.6˚. Note that m=428 in all three domains.

In practice, the device operator specifies θ_d directly, usually between 6 and 10 degrees. This density is chosen empirically based on the maximum gloss of the surface. The operator also specifies whether the sample includes anisotropic BRDFs.

I employ a pose choosing algorithm that models the movement of the four motors over their entire domain in small increments (about 1˚) and at each pose determines whether the ω_i, ω_r direction pair from the center of the sample material to the light and camera, in the local surface reference frame, is within θ_d of an already chosen direction pair. If not, the direction pair is added to the list of poses to sample.

Samples with incident polar angle greater than 75˚ or exitant polar angle greater than 65˚ were rejected. The pose choosing algorithm uses a bi-angular distance metric similar to that of Liu et al. (Liu, Shum et al. 2001). It treats reciprocal direction pairs as equivalent. For isotropic samples it also treats symmetrical direction pairs as equivalent. Table 3.2 shows the actual number[8] of direction pairs chosen and measured when the user requested a given θ_d.

[8] This number is larger than the corresponding m because an error in an early version of my distance metric caused me to sample more densely near the poles.

θ_d	Isotropic	Anisotropic
6˚	1426	
7˚	957	
8˚	683	17499
9˚	493	11216
10˚	385	7663
15˚		1820
20˚		671

Table 3.2: *Number of light-camera poses used to sample the BRDF at the given angular density over the isotropic and anisotropic BRDF domains.*

One advantage of the proposed spatial gonioreflectometer is that every surface point is visible to both the camera and the light in every exposure. Systems that measure a solid object have an average of half the surface points occluded from the camera and half occluded from the light in any given pose, so an average of four times the number of poses is needed. An advantage of a motor-controlled device, besides the obvious reduction in manual effort, is that the motor system can cause the chosen samples to be uniform over the BRDF domain. This greatly increases the likelihood that *m* samples will indeed sample the surface within the specified angular sampling density. With manual positioning, many more samples are needed to increase the likelihood of sampling all features with the given sampling rate.

3.4. Camera

I use a digital camera, rather than a spectroradiometer, as a sensor since I require a spatially varying measurement rather than a single measurement over the whole surface. The ideal camera for use in a device such as this would have high light sensitivity, high bit-depth, lossless image representation, access to raw CCD values, manual control of all image appearance properties and a high transmission bandwidth to the computer, in addition to satisfying the resolution-related goals of Section 3.2.1.

Table 3.3 shows the relevant specifications of five camera-lens configurations that I considered using. The Kodak DCS 520[9] is a through-the-lens still-image camera with a moving mirror and shutter. The Kodak has higher bit-depth and resolution, but requires three

[9] Kodak DCS 520, Kodak Inc. (Also sold as the Canon EOS D2000. I used the Canon.)

seconds to transmit an image to the computer. The Sony DFW-SX900[10] is a full-frame, eight-bit-per-channel video camera with no moving components. The Sony has faster transmission time (one third second), but lower resolution and light sensitivity. Both have manual control, but neither gives access to raw CCD values.

Figure 3.4 shows a diagram of relevant values. The resolution goals correspond to sections (b), (c), and (d) of Table 3.3, which hold a given parameter fixed while varying the other two. Section (b) fixes the size, l, of the sample material and computes the sampling density in one dimension on the surface, p, and the range, r, to the surface as a function of vertical field of view, fov_v, (since the vertical FOV is smaller it constrains the distance to avoid cropping). Equations (3.7) and (3.8) relate these attributes.

$$l = 2r \tan \frac{fov_v}{2} \tag{3.7}$$

$$p = \frac{res_v}{l} \tag{3.8}$$

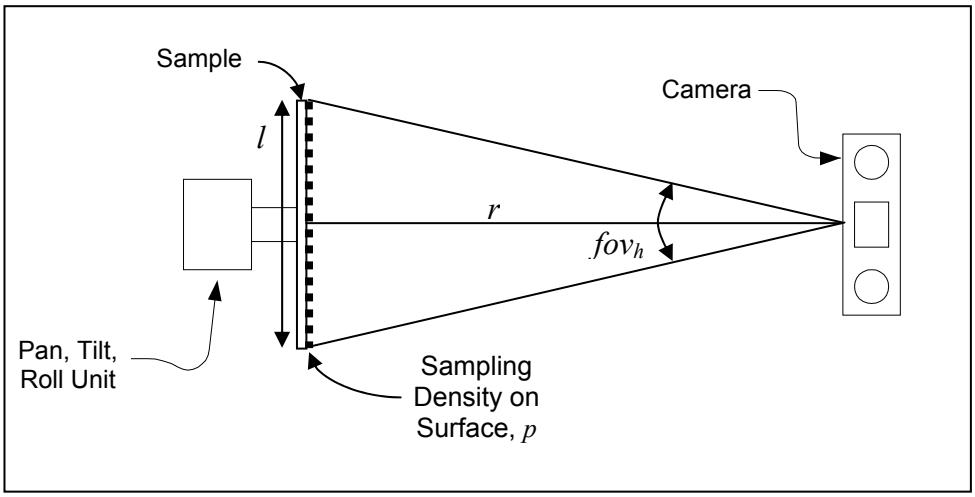


Figure 3.4: *Geometric layout of camera and sample for resolution computation.*

I chose the Kodak camera because of its higher resolution and bit-depth, despite the fact that it requires nearly three seconds to transmit an image to the host, a major bottleneck in the experimental setup. I chose the 50 mm focal length as a compromise between wide field-of-view and good optical properties. I desire a wide field of view so that the camera, mounted on an optical rail, can be close to the sample. As with having a local light, a local

[10] Sony DFW-SX900, Sony inc.

viewer lets different points on the sample that have the same BRDF be sampled from substantially different angles within a single photograph, providing a denser angular sampling of that BRDF. However, short focal lengths (wide field of view lenses) typically require a large amount of glass, increasing geometric and photometric lens distortion. I calibrated the camera and lens geometry using the Zhang method (Zhang 2000) and code of Bouguet (Bouguet 2000).

A final consideration regarding the geometry of the camera is its aperture setting. A small aperture is needed to keep the surface in focus. At a range of one meter, a 30 × 30 cm surface seen from an oblique angle extends considerably beyond the depth of field of wide aperture settings. I used *f*/16 for all experiments, which was the widest aperture to show no blurring at oblique angles. However, the narrower the aperture, the less energy hits the sensor per time, worsening the signal to noise ratio at the CCD. Thus, the narrow aperture calls for a bright light source.

		Kodak DCS 520	Kodak DCS 520	Kodak DCS 520	Sony DFW-SX900	Sony DFW-SX900
(a)	**Lens Focal Length (mm)**	**14**	**20**	**50**	**12**	**16**
	Lens Focal Length (pix)	1082	1666	4165	2375	3167
	CCD Width, res_h (pix)	1728	1728	1728	1280	1280
	CCD Height, res_v (pix)	1152	1152	1152	960	960
	Horiz. Field of View (deg)	77.182	54.823	23.439	30.160	22.849
	Vert. Field of View (deg)	56.028	38.144	15.748	22.849	17.237
	Horiz. FOV, fov_h (rad)	1.347	0.957	0.409	0.526	0.399
	Vert. FOV, fov_v (rad)	0.978	0.666	0.275	0.399	0.301
(b)	**Range to Surface, r (mm)**	281	433	1084	742	989
	Pixels per mm, p	3.84	3.84	3.84	3.2	3.2
	mm per Pixel	0.260	0.260	0.260	0.313	0.313
	Surface side length, l (mm)	*300*	*300*	*300*	*300*	*300*
(c)	***Range to Surface, r (mm)***	*1000*	*1000*	*1000*	*1000*	*1000*
	Pixels per mm, p	1.083	1.666	4.165	2.375	3.167
	mm per Pixel	0.924	0.600	0.240	0.421	0.316
	Surface side length, l (mm)	1064	691	276	404	303
(d)	**Range to Surface, r (mm)**	270	416	1041	593	791
	Pixels per mm, p	*4.000*	*4.000*	*4.000*	*4.000*	*4.000*
	mm per Pixel	0.250	0.250	0.250	0.250	0.250
	Surface side length, l (mm)	288	288	288	240	240

Table 3.3: *Camera resolution attributes. a) Geometric properties of each camera with specified focal length lens. b) Fixing sample size causes sampling density to vary and dictates distance from camera to sample. c) Fixing distance to sample causes maximum sample size and sampling density to vary. d) Fixing sampling density causes maximum sample size and distance from camera to sample to vary.*

3.4.1. Camera Response Curve

The Kodak camera does not allow access to the raw CCD values. The host processing software applies several color balancing and spatial convolution filters to the image stored by the camera. But not even the stored images contain raw CCD values. A response curve is applied by the camera circuitry before storing the file. I characterized this response curve in a lab by placing the light source directly in front of the camera, with the two separated by neutral density filters. By varying the neutral density filters and the exposure time I measured the pixel values returned for known relative radiance values. I fit a cubic curve in log-log space to the measurements. A line in log-log space corresponds to a gamma function. A cubic similarly preserves precision for low exposure values.

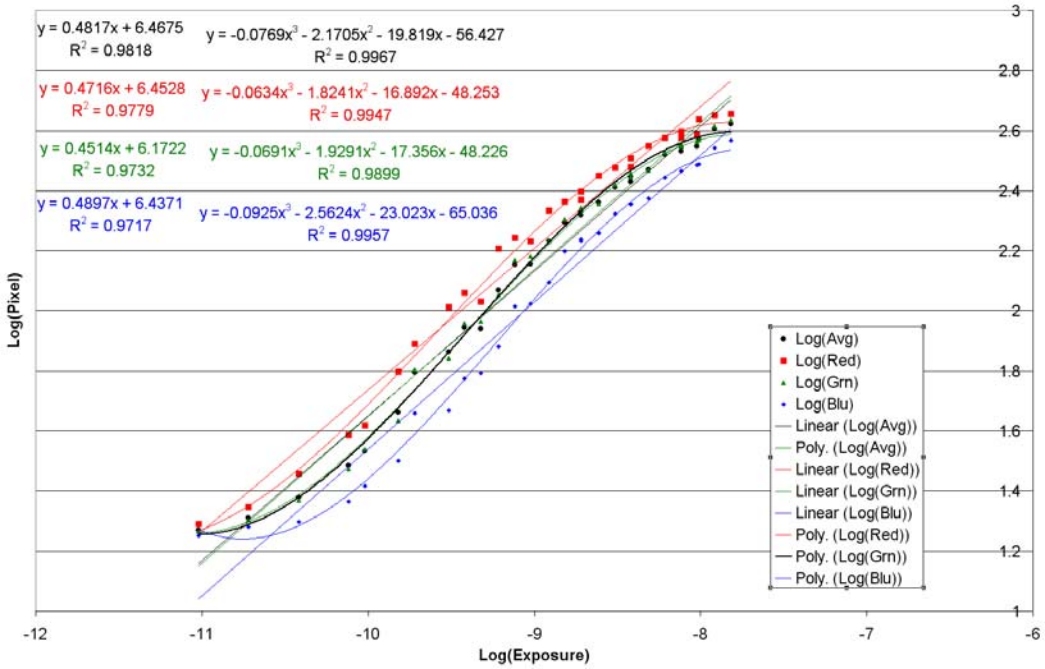


Figure 3.5: *Measured relative radiance values with varying shutter speed and neutral density filters. Shows separate red, green, blue, and luminance curves. A line and a cubic curve are fit to each dataset. The equations of these appear in the upper left.*

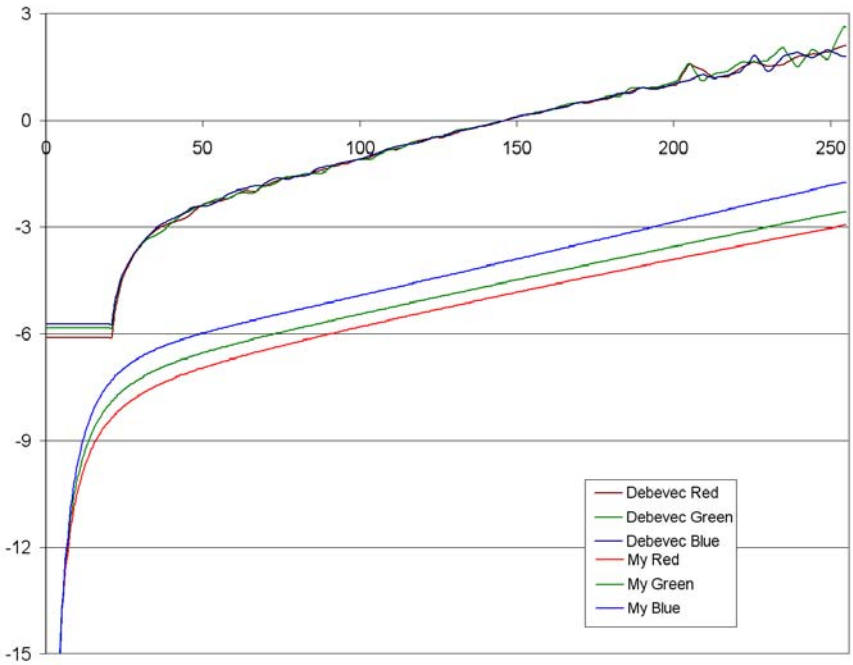


Figure 3.6: *Comparison of response curve measured using cubic curve fit vs. the linear fitting method of Debevec. The vertical axis is relative exposure (relative energy) and the horizontal axis is pixel value. The vertical offset of the two curve bundles is due to an arbitrary scale factor in the Debevec method.*

I compared my direct measurements of the response curve against a curve I computed with the method of Debevec and Malik (Debevec and Malik 1997). The two differ only by the scale factor except at very high and low pixel values, where mine is better behaved because of the low parameter curve fit. I used spectroradiometer[11] measurements to compute the scale factor to absolute radiance in watts/m^2/sr. Although not necessary for my reflectance experiments, the resulting measured data sets could be more useful for future research as absolute radiance. The response curve is used to convert a pixel value and its exposure factor (the exposure time, aperture, and neutral density filter value, if any) to a radiance value.

Most of my samples had regions that were too bright and others that were too dark to accurately measure with a single exposure setting. They required two or three photographs at different shutter speeds at each pose, depending on the dynamic range of the exitant radiance across the sample. Prior to beginning an acquisition session, the device operator chooses an exposure bracket for both the darkest and lightest poses. The angle between the view direction and the reflection direction is then used to predict the shutter speeds necessary to properly expose the surface. This attempts to ensure that every point on the surface will be properly exposed by one of the photographs taken at each pose. A brute force approach might acquire a wide range of shutter speeds at each pose, while a more sophisticated approach might analyze the images in real time and adjust the shutter speed if they are saturated or underexposed.

Another part of camera calibration is dark current correction, which subtracts from each pixel a constant to account for that CCD sensel's[12] tendency to increase its charge in a systematic way without photons hitting it. The constant is determined by acquiring exposures in the absence of illumination. I implemented dark current correction, but since dark current noise is a time integrating effect, it was minimal because of the relatively short exposure times I used (2^{-11} to 2^{-1} seconds).

Flat field response correction is another standard part of camera calibration in which the radial response falloff due to the lens geometry is estimated and reversed. My

[11] PR-705, Photo Research, Inc.
[12] A sensel is a pixel of a sensor such as a CCD.

experiments showed flat field response correction to be unnecessary with the 50 mm lens, which contains very little glass and thus caused no detectable vignetting.

3.4.2. Color Space Conversion

Section 2.2 describes the concept of a tristimulus color space. Every RGB color camera and display has a slightly different RGB color space. A 3 × 3 matrix maps one to another. To map the color space of the measurement camera to the CIE XYZ color space requires knowing this 3 × 3 matrix. This can be computed based on the camera's measured response to radiance of known XYZ. I photographed a ColorChecker[13] chart (see Figure 3.7) in high dynamic range and averaged the radiance across each patch. I also measured each patch using a spectroradiometer to determine its XYZ values under the current illumination. The matrix that transforms the camera RGB colors to CIE XYZ values was computed by least squares. This matrix was multiplied by another matrix to convert XYZ to Sony Trinitron RGB color space.

Figure 3.7: *Macbeth ColorChecker chart photographed at high dynamic range and corrected for camera response.*

[13] GretagMacbeth, Inc., New Windsor, NY

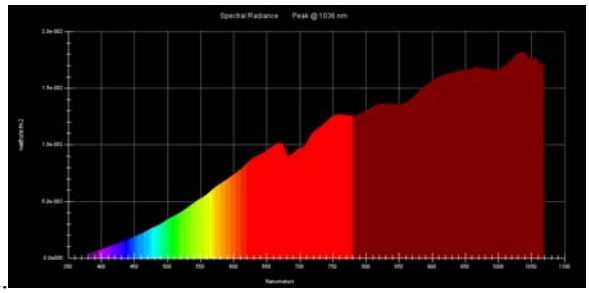
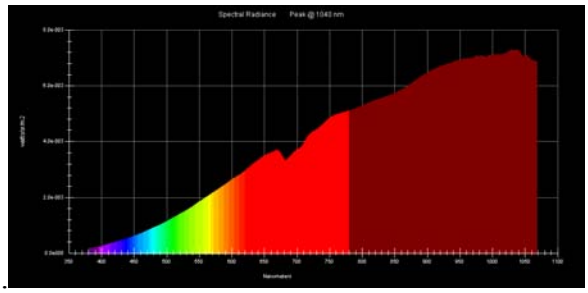
3.5. Light Source

Based on the experience of Gösele et al. (Gösele, Heidrich et al. 2000), I chose a 200W Metal Halide source with an arc length of 5mm[14]. The light comes with a non-frosted ultraviolet filter and is otherwise just a bare bulb, which well-approximates a point source. I wanted to treat the light source as a local, point light rather than a directional light. Within a given pose a local light illuminates each point on the surface from a different angle. If different points on the surface have the same BRDF, this provides more measurements of each different BRDF. This fact was used in a similar way by Karner et al. (Karner, Mayer et al. 1996).

I model the illumination incident on the sample assuming radially uniform radiance exiting the light. My lighting calculations assume there is no additional incident light. I chose a very bright light to improve the signal to noise ratio at the sensor. I also shrouded the light to make a small escape aperture and covered all surfaces with black gaffer's tape, paint, or cloth. With the calibrated camera I photographed the light source from a distance of one meter using several neutral density filters and averaged the radiance over the escape aperture of the light fixture to compute L_s, the absolute radiance from the light source. Although for geometric computations I treat the light as a point source, for irradiance computations I measured the area of the light's escape aperture, A_s.

The light is driven by a flicker-free power supply to eliminate illumination inconsistencies for short exposure times. I let the light warm up for 15 minutes before acquisition, calibrate the bulbs, and use each bulb only within its rated lifetime.

[14] Bug Lite 200 from K5600 Lighting, Inc., Davie, FL

a.  b. 

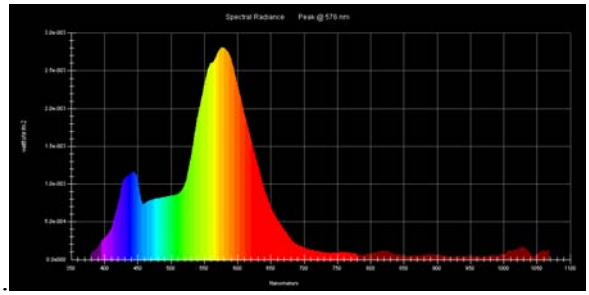
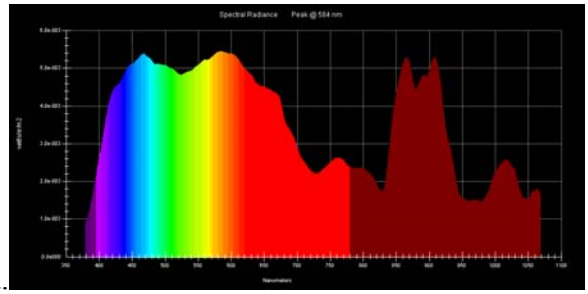
c.  d. 

Figure 3.8: *Sprectral power distributions for white copier paper illuminated by four light sources: a) halogen, b) 60W incandescent, c) fluorescent, and d) 200W Metal Halide. The latter has the fullest spectrum in the visible range and thus the highest color rendering index.*

The spectrum of the light should be full, even though the system's sensor is an RGB color camera. Having a full spectrum ensures that the sample material is strongly illuminated at the wavelengths sampled by the camera. The fullness of a light source's spectrum is represented by its color-rendering index (CRI), which compares the light's spectrum to a black body source of the same color temperature, with 100 being the theoretical maximum. The Bug Lite 200 has a CRI above 90. The spectrum is significantly fuller than other standard light sources, as shown in Figure 3.8.

3.6. Computing the Sampled SBRDF

After acquiring the full set of photographs of the sample, the system converts the set of photographs into an intermediate representation consisting of a set of rectified, registered, high dynamic range images, with bi-directional reflectance values at each pixel (Figure 3.11, Rows 1-3). All intermediate images have the same user-specified resolution as the final SBRDF texture map. Chapter 0 describes the use of the intermediate image set to compute the parameters of a BRDF representation at each pixel from the set of bi-directional reflectance values at the pixel. Table 3.4 lists each material I acquired, its chosen resolution, and the size of the intermediate representation.

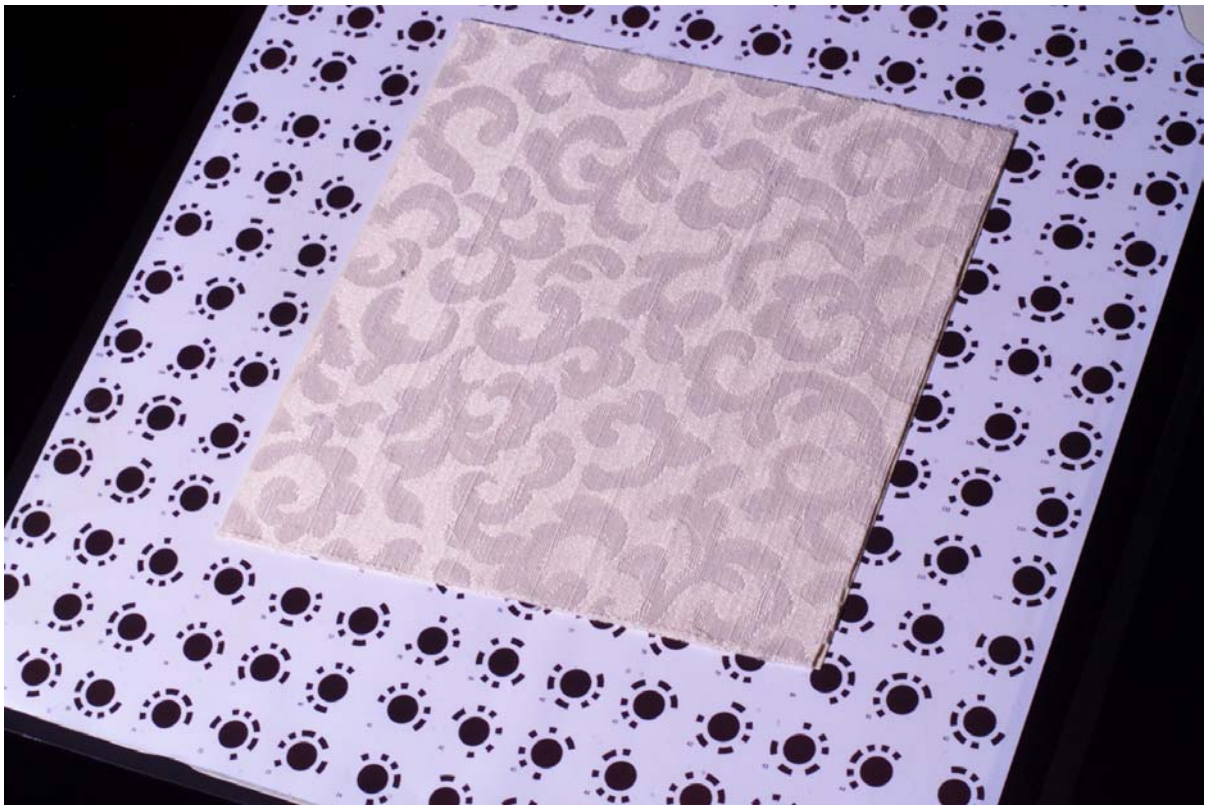


Figure 3.9: *A source image of the White Fabric sample, showing the fiducial markers.*

To compute the intermediate representation of the SBRDF, the system processes each pose sequentially. At each pose it first computes the location of the light and camera in the space of the sample, based on the recorded motor positions. As Section 3.3 explains, the device tolerances are not sufficient to compute a transformation from the camera image to the texture map space on the plane of the sample to subpixel resolution. The system instead computes this planar homography based on fiducial markers made using code and methods from Marschner (Marschner 1998). See Figure 3.9. It computes a least squares solution of the mapping from the camera image locations of the fiducial markers to their known locations printed on the sample plane. The RMS registration error between the printed fiducial locations and transformed locations averaged 0.15 mm in the sample plane over all poses and markers. Typical camera pixel size at the sampling plane was 0.25 mm on a side, so it achieved subpixel registration accuracy on average. Just using motor positioning, on the other hand, would have yielded a registration error up to 2.97 mm.

As an example of the registration quality, I scanned the test pattern shown actual size on the left of Figure 3.10. I scanned densely, using 2200 poses. The diffuse component of the resulting SBRDF, made using the algorithms of Chapter 0, is shown on the right.

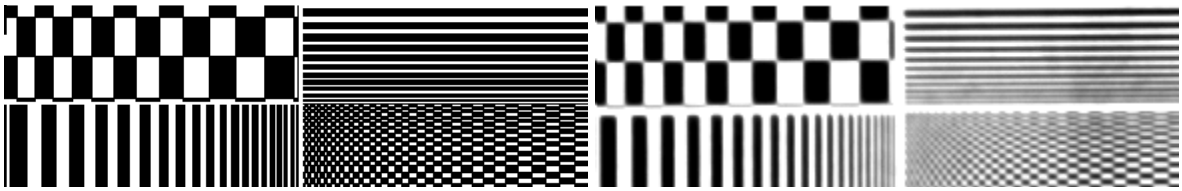


Figure 3.10: *2200 registered photographs of the test pattern on left were averaged to create the SBRDF on right, shown actual size at 400 × 200 pixels. The resolvability of the small checkerboard squares demonstrates the high registration quality. The camera was at a distance of 1 m.*

The existing blur in the result image comes from using finite sampling kernels in the 6D SBRDF space. That is, each CCD sensel samples a finite area on the surface, similar to the ellipse in Figure 2.4. Likewise, the motor positioning error can be seen in the limit as a sampling aperture over the four dimensions of the BRDF.

3.6.1. Computing Reflectance

The system next computes a radiance image from the photographs at a given pose and resamples the radiance image into the texture space using bi-cubic sampling. As each pixel $\vec{X}$ of the texture map is resampled from the source image its value is also converted from radiance to a BRDF sample, by applying Equation (2.4), which I repeat here:

$$f_r\left(\omega_i \to \omega_r\right) = \frac{L_r\left(\omega_r\right)}{E\left(\omega_i\right)} = \frac{L_r\left(\omega_r\right)}{L_i\left(\omega_i\right)\cos\theta_i d\omega_i} \tag{3.9}$$

For this computation, $L_r(\omega_r)$ is simply the radiance value sampled from the source image. $L_i(\omega_i)=L_s$ from Section 3.5. Then,

$$d\omega_i = \frac{A_s}{r\left(\vec{X}\right)^2} \tag{3.10}$$

is the solid angle of the light as seen from point $\vec{X}$. $r(\vec{X})$ is the distance from $\vec{X}$ in world space to the light. θ_i is the angle ω_i makes with the normal, computed separately for each $\vec{X}$.

Computing the θ_i and $d\omega_i$ per pixel instead of per pose each affect the irradiance computation by approximately 10%.

3.7. Acquisition Results

The surfaces I scanned are listed in Table 3.4. The raw camera images are stored with lossless compression, and are approximately 2 MB in size. Each surface had 1, 2, or 3 images per pose depending on the dynamic range of the exitant radiance over the surface. The size of the database in this form is given in Table 3.4, column 5. The intermediate representation with rectified, resampled images is a single file whose size is directly proportional to the chosen resampled size and the number of poses acquired, with a four-byte red-green-blue-exponent (RGBE) pixel representation (Ward 1994). A few poses were rejected for bad registration.

					Line Search			Levenberg-Marquardt (1 lobe)			Levenberg-Marquardt (2 lobe)		
	# Poses	Sampled Width	Sampled Height	Sampled Size (MB)	Fit Time (sec)	Avg. Pixel Error	Max. Pixel Error	Fit Time (sec)	Avg. Pixel Error	Max. Pixel Error	Fit Time (sec)	Avg. Pixel Error	Max. Pixel Error
Blue Fabric	456	256	256	116	3926	0.000	0.002	4556	0.0003	0.0020	30274	0.0001	0.0017
Gift Wrap	314	128	128	18	170	0.034	0.332	2397	0.0075	0.0868	3897	0.0062	0.0869
Gold Fabric	2455	512	512	2400	6870	0.001	0.041	55628	0.0021	0.0783	103611	0.0021	0.0815
Gold Paper*	422	256	256	89	549	1.779	706.1	9888	1.6307	707	12627	1.7049	686.0
Long Leaf	311	1024	128	143	733	0.005	2.030	7902	0.0038	1.3085	28062	0.0102	2.0310
Test Pattern	2200	400	200	700	3111	0.043	0.283	61343	0.0437	0.2829	80123	0.0428	0.2829
Tan Fabric	385	512	512	375	7628	0.0020	0.017	47660	0.0021	0.0411	44061	0.0020	0.0354
Wall Paper	390	256	256	90	2315	0.001	0.007	11061	0.0007	0.0049	15106	0.0007	0.0068
White Fabric	7650	512	512	7500	**	0.0048	0.0104	**	0.0043	0.0113	**	0.0043	0.0105
White Fabric	7650	64	64	119	730	0.005	0.008	33852	0.0051	0.0100	43645	0.0025	0.0049

Table 3.4: *Acquisition and fitting results for nine acquired materials. See Chapter 0 for a description of the data fitting. *: The Gold Paper material includes mirror reflection that my system does not currently handle, causing the large error. **: This 8 GB data file was processed on an SGI mainframe, so the fit times cannot be compared. The two lobe fit took approximately 30 times as long as the line search fit.*

The sampled SBRDF data is difficult to visualize, as it is a 6D function. To explore my measurement results I use an interactive viewer that includes three views of the data: a 2D spatial view of the texture for an interactively chosen incident and exitant direction pair, a hemispherical view of BRDF values in all exitant directions for an interactively chosen incident direction and spatial texel, and a standard polar plot of the BRDF values for the same incident direction and spatial texel.

Figure 3.11 shows these visualizations using the vinyl wall paper with gilded stripes. Rows 1, 2, and 3 show rectified acquired views for various incident and exitant directions. Rows 4, 5, 6, and 7 show bi-directional reflectance for both a gilded texel and a vinyl texel. The data show substantial forward scattering, increasing for more grazing angles, with the peak below the specular direction. Retroreflection also increases at grazing angles.

As Section 3.3.2 explains, the sampled light and camera directions are not on a uniform parameterization of the hemisphere. Also, each texel of the SBRDF has a slightly different direction to the light and camera. Thus, I cannot visualize a large set of exitant directions exactly corresponding to a given incident direction. So for the exitant hemisphere visualizations I again employ the bi-angular distance metric (Liu, Shum et al. 2001) to find the closest acquired pose to the queried incident-exitant pair at each pixel. Conceptually, this is a sampling of Voronoi regions in the 4D BRDF space, sampled on the hemisphere. Since the incident directions of the acquired poses do not exactly match the queried incident direction, the visualizations appear noisier than the data really is in the 4D BRDF space. The same is true of standard incidence plane plots. The nearby off-plane samples typically have different BRDF values than the in-plane samples.

Figure 3.12 through Figure 3.18 are visualizations of acquired sample materials listed in Table 3.4. The top half of each figure shows the rectified reflectance image closest to the chosen incident and exitant direction pair, with θ_i and θ_r being the polar angle of the direction vector above the X axis.

The first column of each figure visualizes individual pixels of each measured SBRDF using exitant hemispheres. Each hemisphere is a visualization of the pixel at the center of the hemisphere. These sample materials were chosen because of their greatly varying appearance, both spatially over the surface and angularly over the incident and exitant hemispheres, as can be seen in these visualizations. Interactive visualizations outside the incidence plane reveal even more variation, as the rendered results of Chapter 6 show.

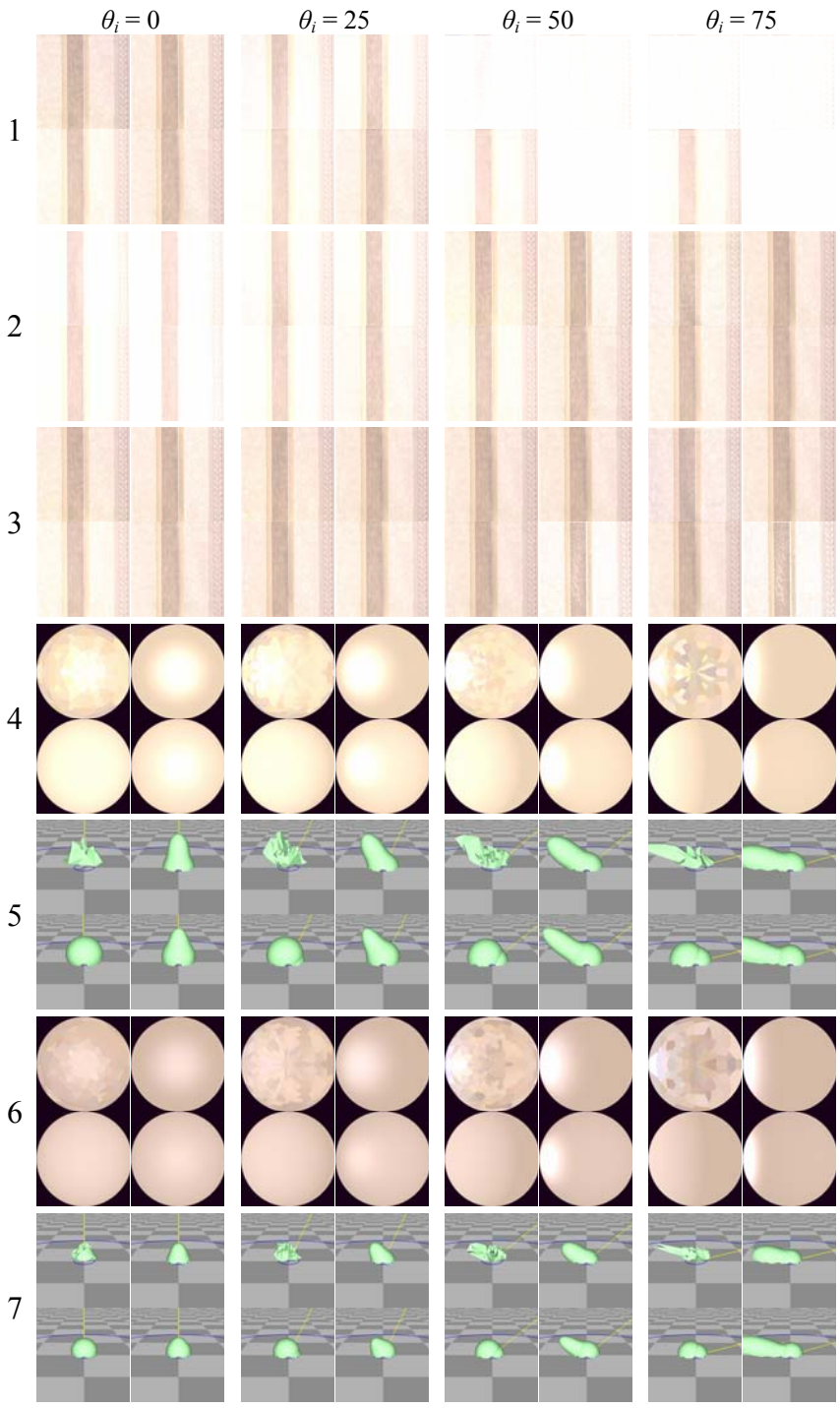


Figure 3.11: *Measured and fit results for vinyl wallpaper with gold leaf. In each cell, the upper-left image shows the measured data, lower left is the line seach fit, upper right is one-lobe L-M, and lower right is two-lobe L-M. Columns differ in incident polar angle above +X axis. Rows 1, 2, and 3 are sampled from -60, 0, and 60 degrees exitance above X axis. Rows 4 and 5 visualize the BRDF of one gilded pixel using reflectance hemispheres and polar plots. Rows 6 and 7 visualize the BRDF of one brown texel. The yellow line in the polar plots is the incident direction.*

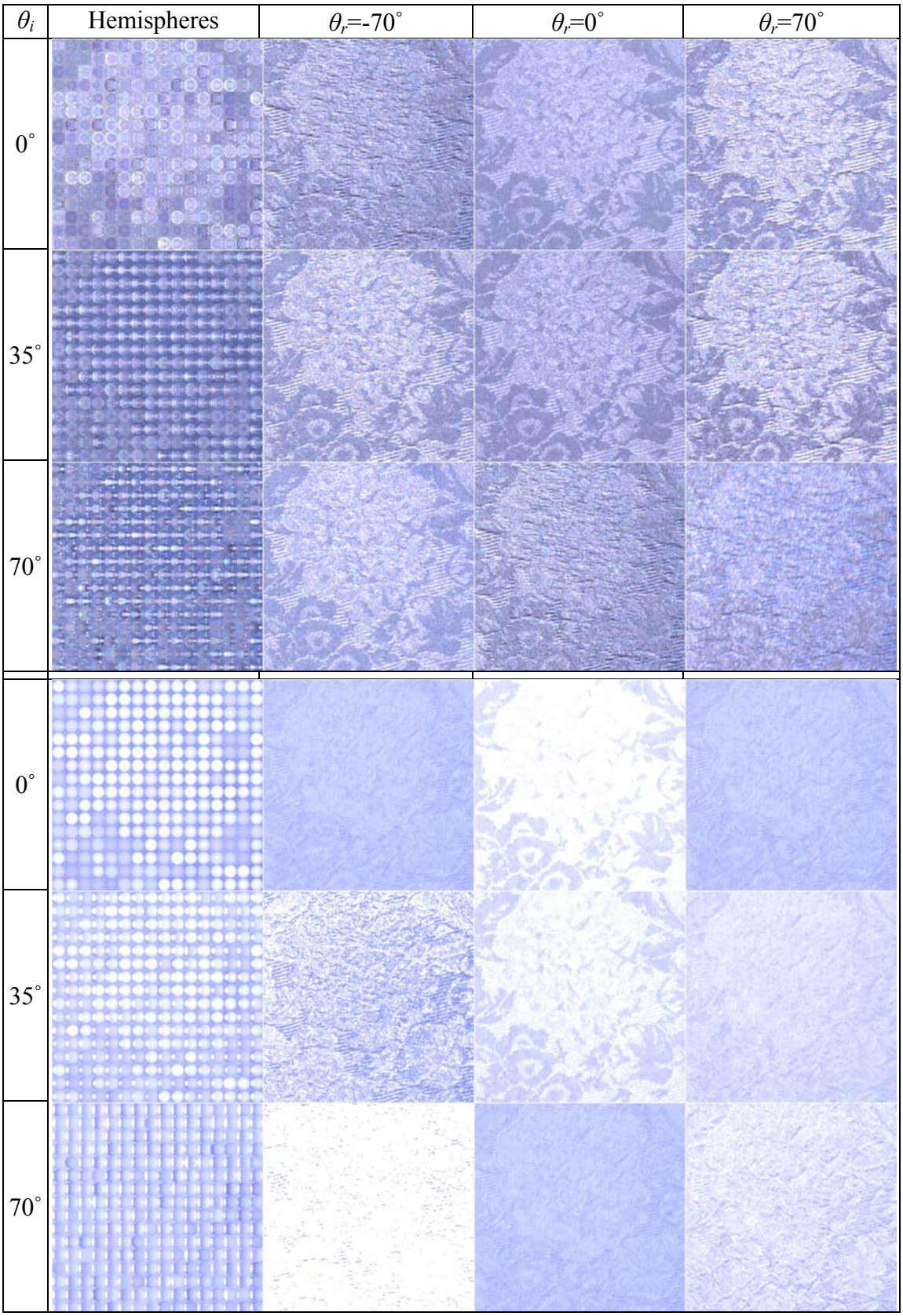


Figure 3.12: *Blue Fabric. Top: Measured data. Bottom: Two-lobe L-M fit.*

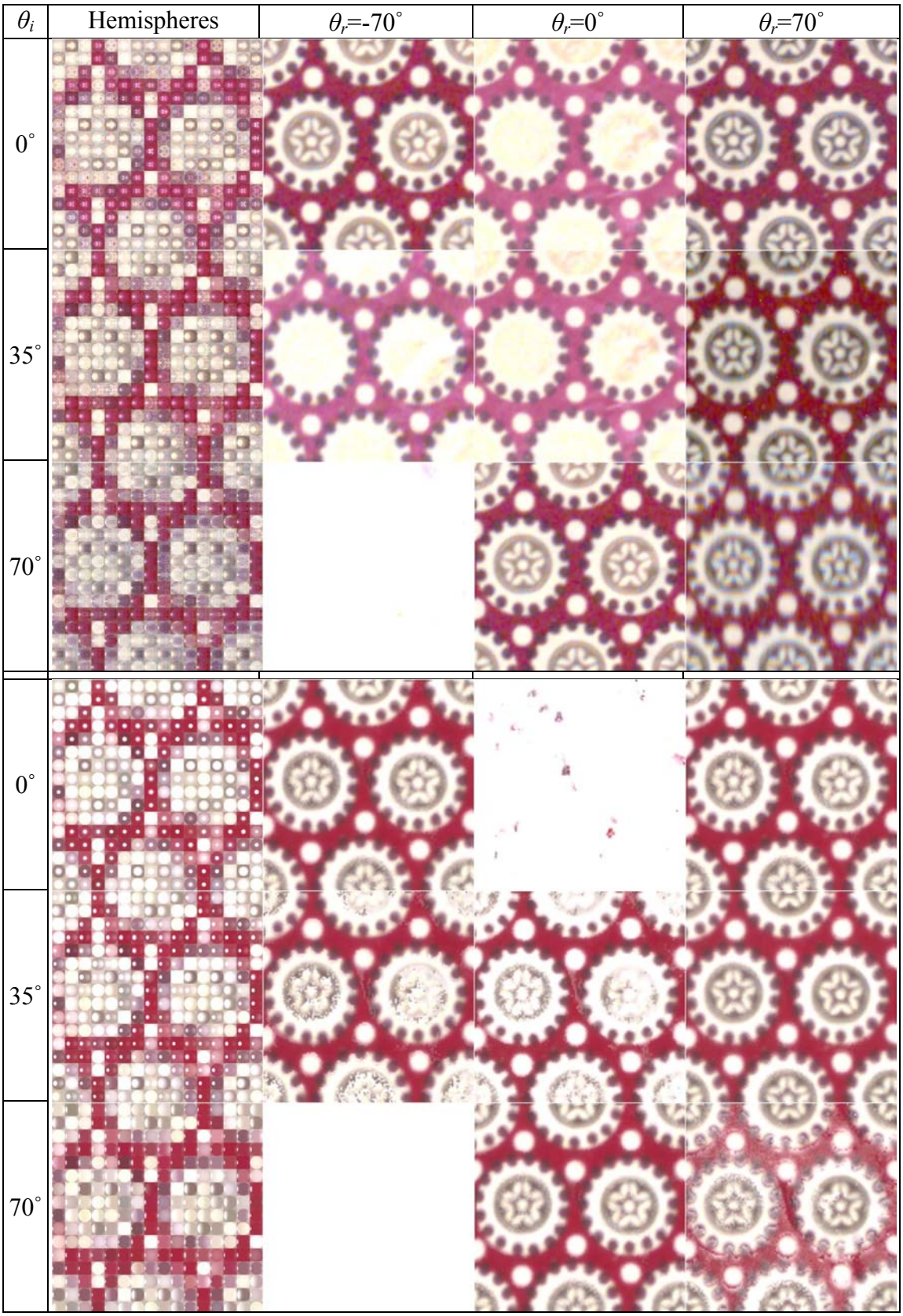


Figure 3.13: *Gift Wrap. Top: Measured data. Bottom: Two-lobe L-M fit.*

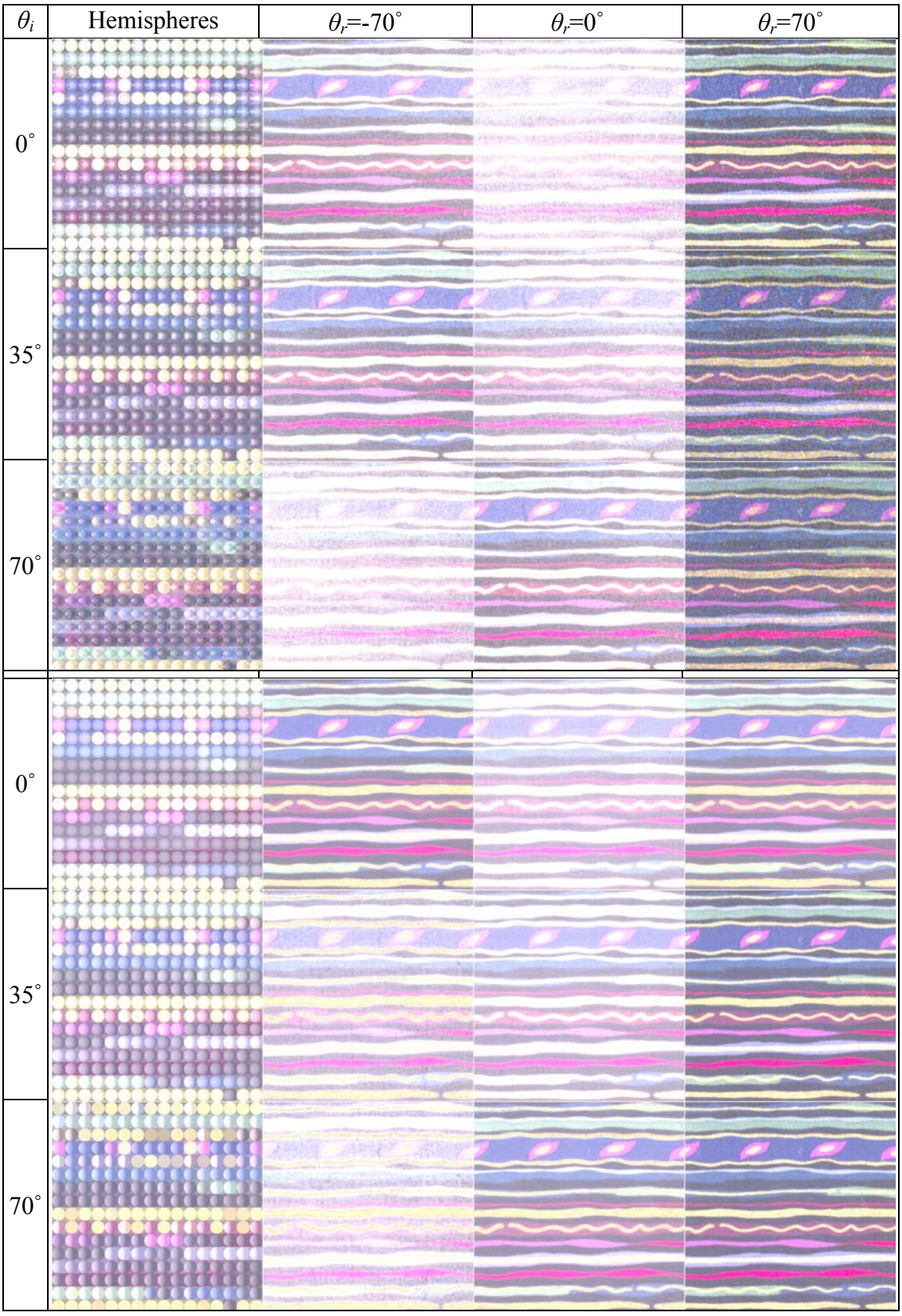


Figure 3.14: *Gold Fabric. Top: Measured data. Bottom: One-lobe line search fit.*

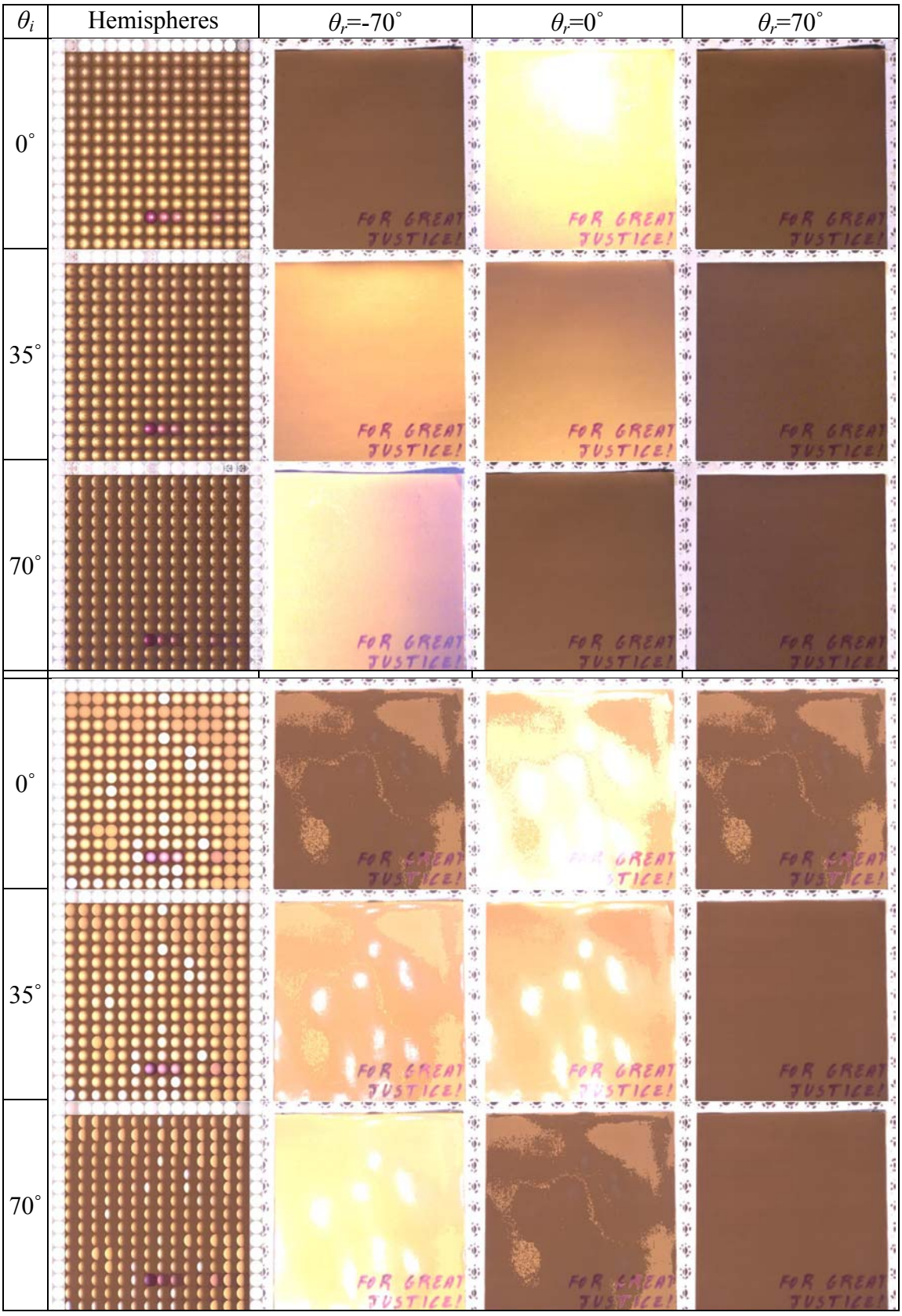


Figure 3.15: *Gold Paper. Poor fitting results are due to extraordinarily bright highlights.*

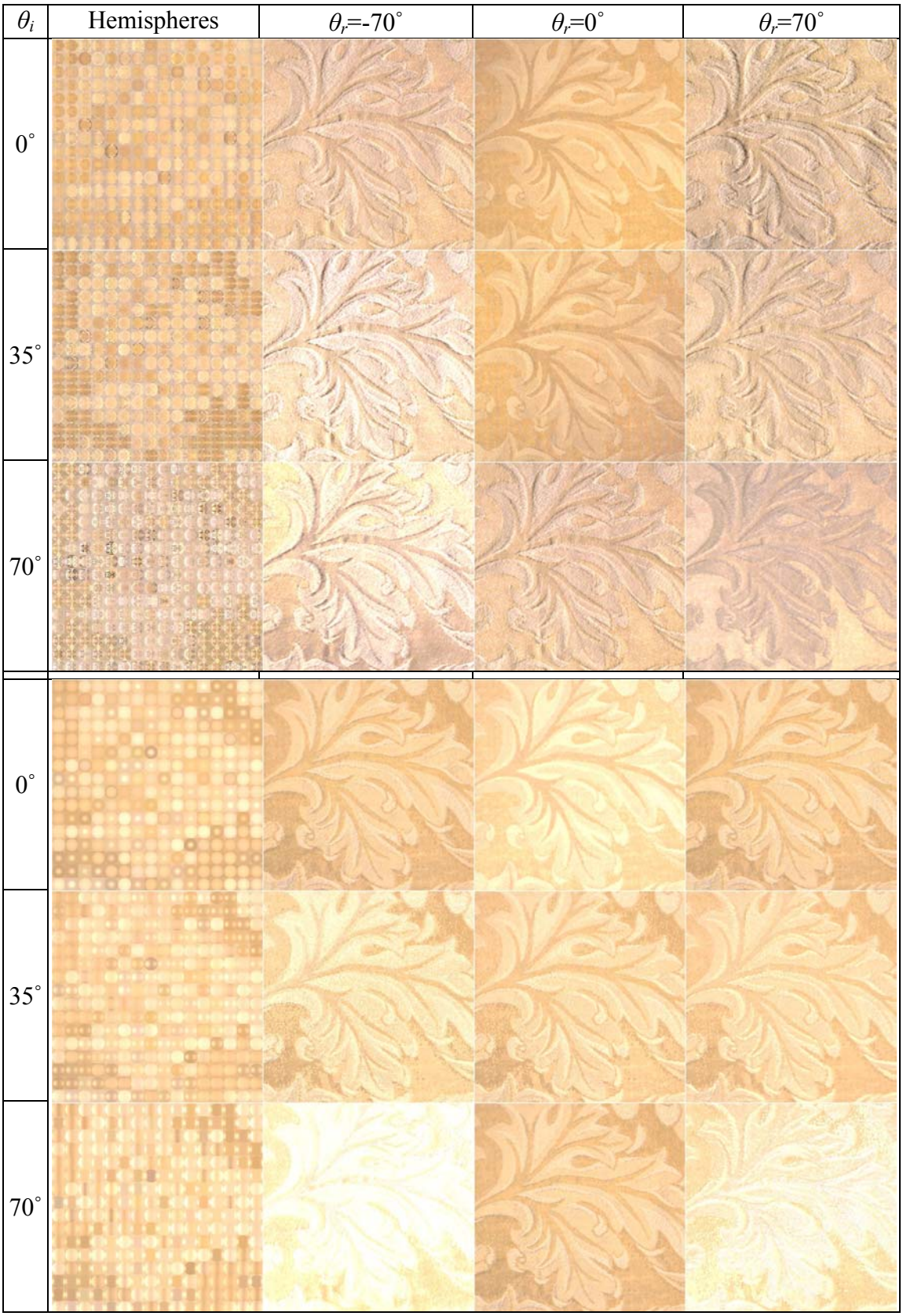


Figure 3.16: *Tan Fabric. Top: Measured data. Bottom: Two-lobe L-M fit.*

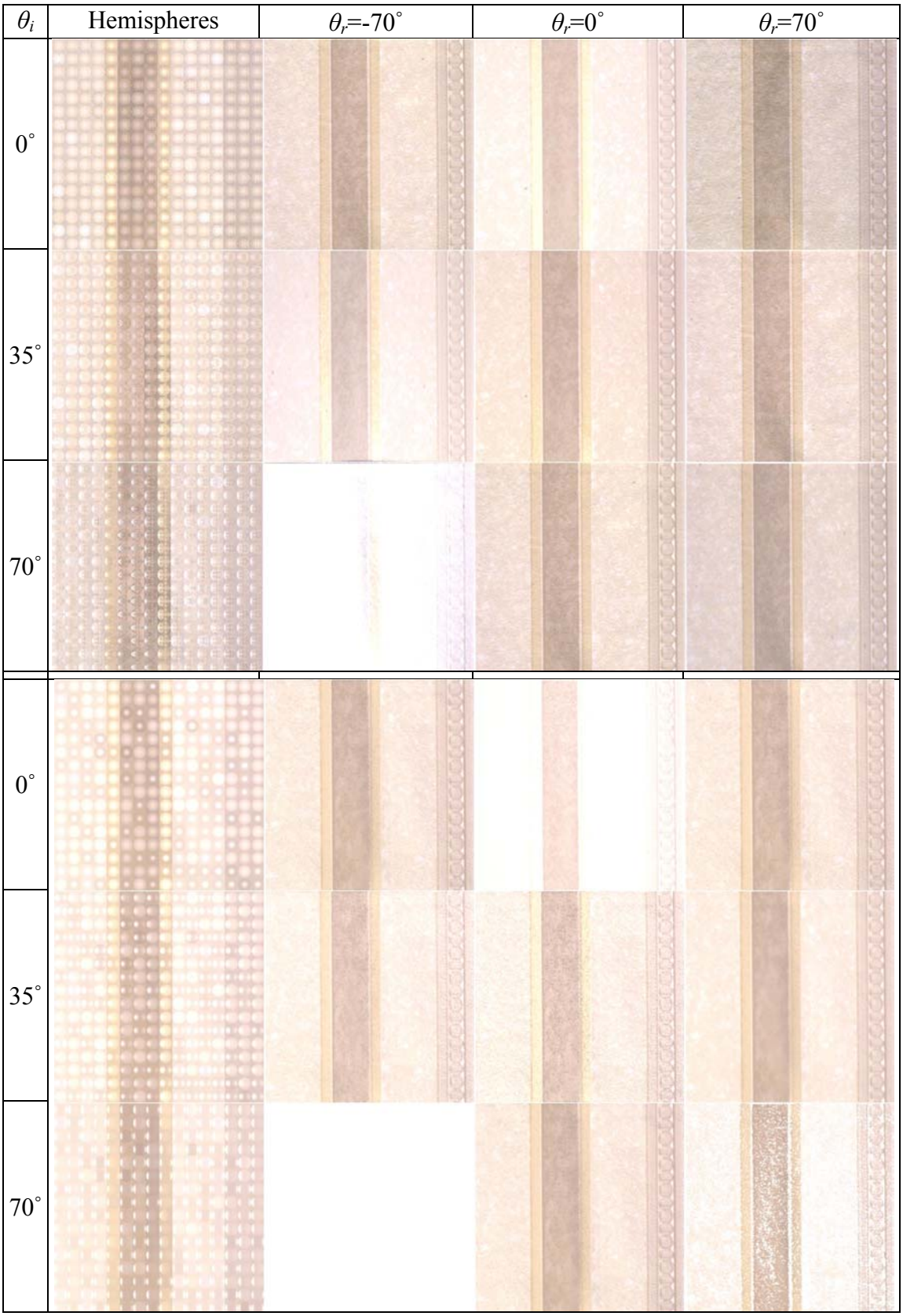


Figure 3.17: *Wall paper. Top: Measured data. Bottom: Two-lobe L-M fit.*

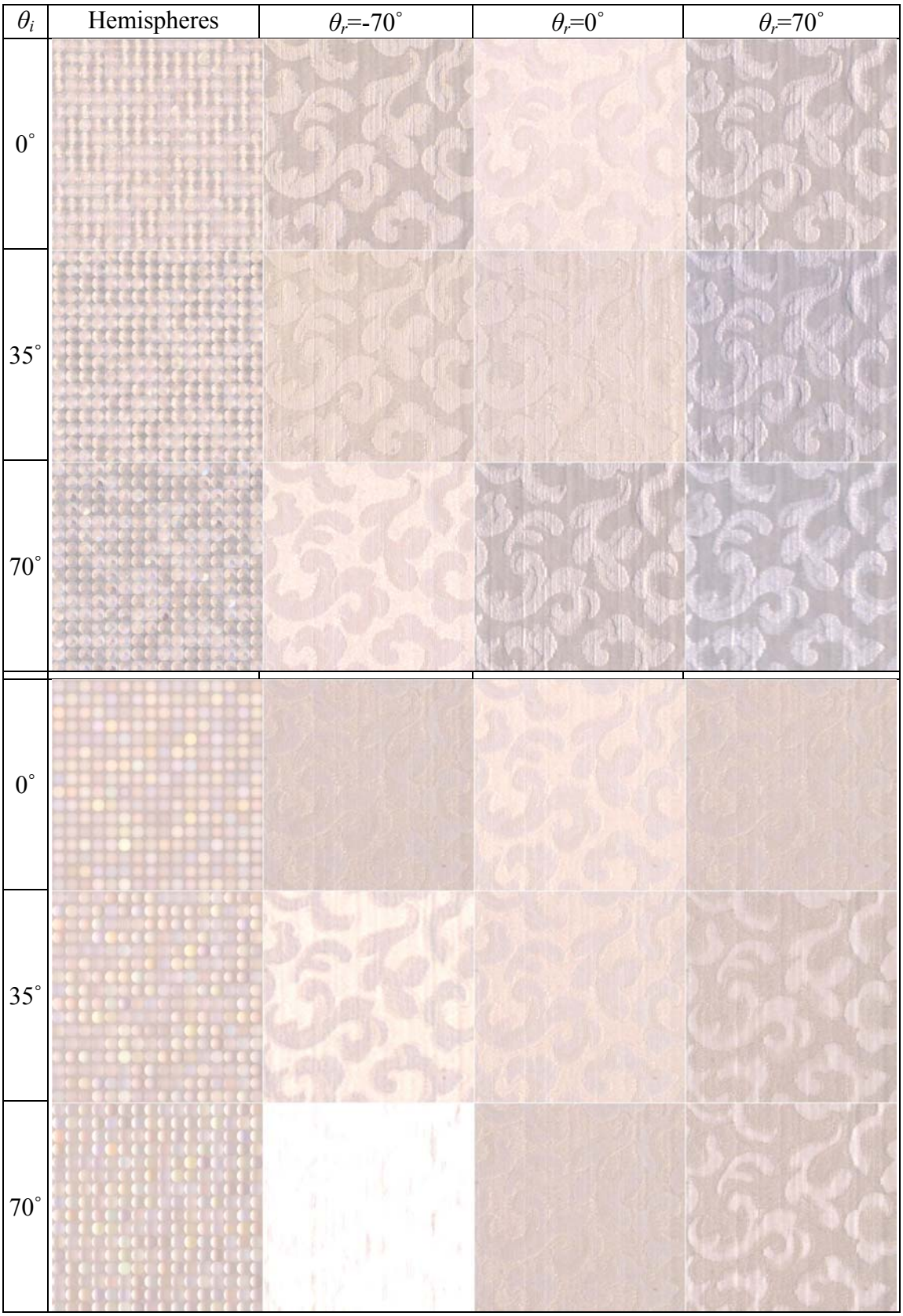


Figure 3.18: *White Fabric. Top: Measured data. Bottom: Two-lobe L-M fit.*

4. REPRESENTING SBRDFS

The measured SBRDFs I have acquired require up to 8 GB in their resampled, tabulated form (see Table 3.4). This illustrates the importance of some sort of data reduction such as statistical compression or projecting the data onto the parameters of a function. Although the data storage capacity of computers constantly increases, making the full, tabulated SBRDF a theoretically viable representation, three important benefits come from reducing the data to a set of function parameters.

1. Most practical uses of SBRDFs, such as rendering using graphics hardware, are far from being able to handle 8 GB surface representations.
2. Fitting to functions provides a method of regularizing the data to constrain it to more physically probable values, i.e. it reduces measurement noise and outliers.
3. Fitting to functions provides a method of interpolating the data. More appropriate functions provide more physically realistic interpolation behavior.

Only the first of these benefits is offered by statistical compression methods. The realization of these three benefits depends on the choice of representation function. To realize benefit number 1 the representation should

- be compact, and
- have a mathematical formulation suitable for efficient evaluation.

For benefits 2 and 3, the representation should

- be flexible enough to represent a wide variety of materials and BRDF behaviors,
- require very few coefficients to match basic scattering behavior of real surfaces, and
- handle a variable number of coefficients for a size and efficiency vs. quality tradeoff.

Additionally, for practical use, the representation should

- allow easy editing of existing materials and modeling of new materials from scratch.

Chapter 2 discussed the nature of the BRDF and texture maps and Section 2.6 discussed considerations involved in combining the two, with one conclusion being that the most suitable form for the SBRDF is a spatially discrete function – a texture map – with a bi-

angularly continuous function – a BRDF – at each pixel. The questions answered in this chapter are what form the BRDF used at each pixel should take and how to compute its coefficients given the tabulated SBRDF.

I will first survey many available BRDF formulations that have been used in computer graphics and then discuss two methods for fitting the BRDF parameters at each texel to yield the final, compact SBRDF. BRDF parameter fitting can be difficult and inefficient for measured data. I address this by developing a more stable, much more efficient fitting method at the expense of accuracy of fit, and compare this to the standard method. Another result I present in this chapter is this system's novel ability to measure and estimate a different direction of anisotropy at each texel for anisotropic materials, which is a key component of the look of many real materials.

4.1. Previous Work

Sections 6.1.1 and 6.9 discuss BRDF factorization and prefiltered environment maps, which can both be seen as a modified way to store tabulated BRDFs. Apart from these tabulated forms, BRDF equations can be divided into two varieties – *BRDF models* and *BRDF representations*. A BRDF model is a reflection function that attempts to model the observed scattering behavior of a class of real surfaces, usually in a physically motivated manner. A survey of several models can be found in Shirley et al. (Shirley, Hu et al. 1997).

Most BRDF models have a fixed number of parameters. Each parameter has some physical meaning, usually relating to some characteristic of the class of surfaces to which the model applies. With most models it is possible to choose parameter values that break the physical plausibility constraints described in Section 2.4.

The Phong model (Phong 1975) is an empirical model that creates a highlight, but breaks all the rules of physically plausibility from Section 2.4. Physically plausible variations of the Phong and other models have been proposed (Lewis 1993).

The microfacet-based Equation (2.10) is a generalization of many models (Torrance and Sparrow 1967; Blinn 1978; Cook and Torrance 1981; He, Torrance et al. 1991). The Torrance-Sparrow model has been used in acquisition systems (Sato, Wheeler et al. 1997;

Yu, Debevec et al. 1999). Debevec et al. (Debevec, Hawkins et al. 2000) used a modified Torrance-Sparrow suited for human skin.

The Ward reflection model (Ward 1992) uses an elliptical Gaussian sharpness function, and can thus create anisotropic highlights with an elliptical cross-section. Banks proposed an empirical anisotropic model that is not physically plausible (Banks 1994).

In contrast to these models, BRDF *representations* approximate the BRDF by a projection onto general basis functions. Examples of basis functions that have been used for BRDFs in computer graphics include spherical harmonics, spherical wavelets, and Zernike polynomials.

Spherical harmonics are spherical analogues of sines and cosines. They are localized in the frequency domain, but are global in the spatial domain. With considerable modification, Westin et al. (Westin, Arvo et al. 1992) remap spherical harmonics from their spherical domain to the hemisphere × hemisphere domain of BRDFs.

Zernike polynomials are used in optics to characterize lenses and have also been applied to BRDF representation (Koenderink, Doorn et al. 1996). As with spherical harmonics, Zernike polynomials are not spatially localized. One advantage of polynomials and spherical harmonics is that they are a linear representation, so the coefficients can be computed in closed form using a linear least squares approach.

Schröder and Sweldens introduced spherical wavelets (Schröder and Sweldens 1995), and proposed using them to represent the exitant hemisphere of a BRDF with a fixed incident direction. Their method was extended to the full BRDF domain (Lalonde and Fournier 1997). Spherical wavelets are spatially localized but are multiresolution, so evaluating the BRDF still requires evaluation of coefficients at a full range of scales.

None of these bases were designed specifically for BRDFs. The individual basis functions do not correspond well to the lobe shape of most BRDFs (Lafortune, Foo et al. 1997). Because of this, many coefficients are usually necessary. The number of basis functions required for a given goodness of fit depends on the frequency content of the BRDF and the flexibility of the basis functions. Using 50 to 200 coefficients is common for these three representations. Using too few coefficients results in poor quality fitting and in ringing.

Another problem with these bases is that the coefficients have no intuitive meaning. This makes it very hard to make a BRDF specification interface for these representations.

Another possibility is to use as a basis function the portion of a BRDF model that defines its lobe shape. McCool et al. (McCool, Ang et al. 2001) suggest a sum of Phong (cosine) or Ward (Gaussian) lobes with differing parameters. However, this is not a very suitable representation for general BRDFs. In particular, the highlight peak of Phong and Ward lobes is always in the reflection direction, so regardless of the number of lobes being summed it is impossible to approximate an off-specular peak, a retroreflective peak, or a non-lobe-shaped BRDF like velvet. A spatially localizable basis is more desirable.

4.2. The Lafortune Representation

Lafortune et al. (Lafortune, Foo et al. 1997) introduced the generalized cosine lobe basis. The generalization is that the lobe peaks can aim in arbitrary directions relative to the incident direction, which makes a sum of lobes much more effective than with Phong (Phong 1975) or Ward (Ward 1992) bases. The basis functions are spatially local, which addresses the problem shared by most basis function representations of either requiring many coefficients or suffering from ringing and poor fitting results. The Lafortune representation consists of a sum of terms:

$$f_r\left(\omega_i \to \omega_r\right) = \rho_d + \sum_j \rho_{s,j} \cdot s_j\left(\omega_i, \omega_r\right) \tag{4.1}$$

where ρ_d is the diffuse reflectance. The terms in the summation are specular lobes. Each lobe j has an albedo, $\rho_{s,j}$, and a lobe shape, s_j. ω_i and ω_r are the incident and exitant directions. The lobe shape is defined as follows, similar to a standard Phong lobe:

$$s\left(\omega_i, \omega_r\right) = \left(\begin{bmatrix} \omega_{r,x} \\ \omega_{r,y} \\ \omega_{r,z} \end{bmatrix}^T \cdot \begin{bmatrix} C_x & & \\ & C_y & \\ & & C_z \end{bmatrix} \cdot \begin{bmatrix} \omega_{i,x} \\ \omega_{i,y} \\ \omega_{i,z} \end{bmatrix} \right)^n \tag{4.2}$$

This can be thought of as a generalized dot product in which the three terms are scaled arbitrarily:

$$s(\omega_i, \omega_r) = \left(C_x \omega_{i,x} \omega_{r,x} + C_y \omega_{i,y} \omega_{r,y} + C_z \omega_{i,z} \omega_{r,z}\right)^n \quad (4.3)$$

The Lafortune representation is evaluated in local surface coordinates. The X and Y axes are the principal directions of anisotropy and Z is the normal. Defining the matrix C as C_x=-1, C_y=-1, C_z=1 causes ω_i to reflect about the normal, yielding a standard Phong lobe. But each lobe is significantly more general than a Phong lobe: the C coefficients of the lobe may also take on other values to shear the specular lobe in ways that represent real surface scattering behavior but still enforce reciprocity and conservation of energy. The lobe's peak will be in the direction $C \cdot \omega_i$. For isotropic BRDFs, $C_x = C_y$. For off-specular reflection, $|C_z| < |C_x|$, pulling the lobe toward the tangent plane. For retroreflection, $C_x > 0$ and $C_y > 0$. When C_x and C_y have opposite sign, a lobe will forward scatter light arriving parallel to the principal direction of anisotropy, but back scatter light arriving perpendicular to it, as arises with parallel cylinder microgeometries (Poulin and Fournier 1990; Westin, Arvo et al. 1992). The Lafortune representation's flexibility to aim and scale each scattering lobe is key in approximating BRDFs using few lobes. This property also enables the glossy environment mapping technique of Section 6.9. The Lafortune representation is well suited to the shape that BRDFs typically have, is compact, and is capable of representing interesting BRDF properties such as the increasing reflectance of Fresnel reflection, off-specular peaks and retro-reflection. It does not handle the increasing lobe sharpness of Fresnel reflection, however. This causes reflection at grazing angles to be brighter, but not sharper, unlike the behavior of real surfaces.

For SBRDFs I represent the ρ albedo values as RGB triples but share the C_x, C_y, C_z, and n values between channels, totaling seven coefficients per specular lobe. Sharing the lobe shape parameters between channels disallows the representation of possibly useful diffraction effects in highlight areas, but is a more compact, more stable to fit representation, and maps more directly to the hardware rendering algorithms of Chapter 6. Note that these seven parameters only contain six degrees of freedom.

I use a selectable number of specular terms depending on the nature of the surface and the desired quality vs. speed tradeoff for data fitting and rendering. With as few as ten coefficients (three diffuse plus seven specular), the SBRDF representation is only a small

factor larger than an RGB texture map. For example, using 16-bit fixed-point numbers and a single lobe, the SBRDF is only three times as large as a comparable RGB image. For anisotropic materials I also store the direction of anisotropy, β, at each pixel. This is shared for all specular lobes since it is an attribute of the surface, not of the lobe. The range of β need only be $-\frac{\pi}{4} \le \beta < \frac{\pi}{4}$ because the signs of C_x and C_y can be reversed to yield all unique directions. β is employed by constructing a C' matrix that includes the rotation about Z by β:

$$C' = R_{-\beta} C R_{\beta} \tag{4.4}$$

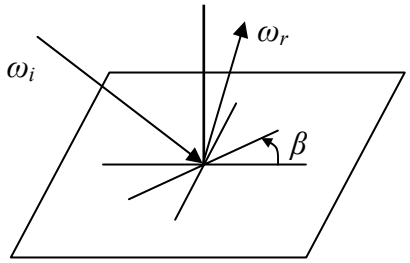


Figure 4.1: *The direction of anisotropy, β, relative to the principal directions of the surface parameterization.*

4.3. Data Fitting

Creating an SBRDF from the tabulated SBRDF samples can be seen as compression, approximation, or data fitting. The system must compute the parameters of the BRDF representation at each pixel, given the bi-directional reflectance values at the pixel. I currently treat each pixel completely independently. Some methods (Debevec, Hawkins et al. 2000; Lensch, Kautz et al. 2001) depend on spatial coherency since they do not have enough angular samples to appropriately fit a BRDF by treating each pixel independently. This has the positive effect of causing pixels that should have the same BRDF to look the same, but can have the negative effect of causing pixels that should look different to look the same.

At each texel the fitting algorithm first estimates ρ_d. Lafortune (Lafortune, Foo et al. 1997) and many image-based algorithms (Pulli, Cohen et al. 1997; Rushmeier, Bernardini et al. 1998) estimate ρ_d as the minimum of the bi-directional reflectance samples:

$$\rho_d = \min_k f_{r,k} \tag{4.5}$$

where $f_{r,k}$ is the k^{th} reflectance sample at the current pixel. This estimator is reasonable in theory because the specular lobes are strictly positive, so the minimum of the BRDF samples is the sample that provides the tightest upper bound on ρ_d. However, real measurement systems have error, some of which is likely to be zero-mean. In the spatial gonioreflectometer, some error also results from imperfect subpixel registration and sampling at oblique angles. Some contribution from neighboring surface points with a possibly lower reflectance value can affect a given BRDF sample. These three kinds of error can lead to BRDF samples smaller than the surface point's true ρ_d. For this reason I estimate ρ_d using a chosen percentile (usually 10th) of each color channel. Figure 4.2 shows the reflectance samples for a selected surface point. Note that the minimum value is clearly not the best estimator of the diffuse reflectance.

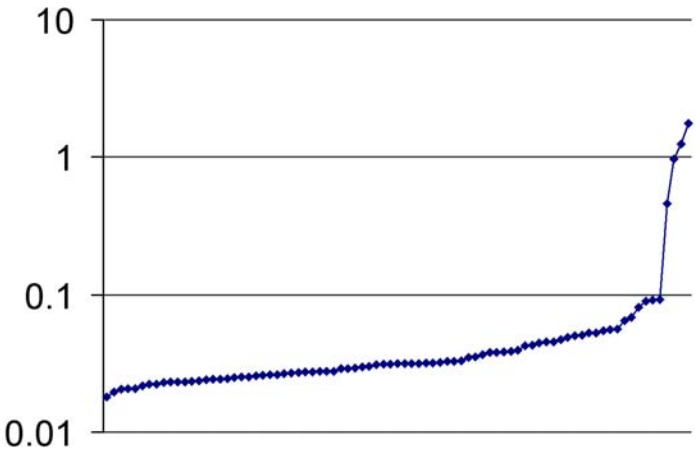


Figure 4.2: *Log scale plot of reflectance values at one pixel. Samples are sorted horizontally from smallest to largest reflectance.*

Marschner (Marschner 1998) computes ρ_d as the mean of all samples, weighted by the product of $\vec{N} \cdot \omega_i$ and $\vec{N} \cdot \omega_r$. The weights correspond to measurement accuracy. This method is useful for Lambertian diffuse surfaces, but will be biased for surfaces with any directional diffuse component.

After subtracting ρ_d from each of the reflectance samples at the current pixel, the system fits specular lobes to the specular residual at each BRDF sample, $f_{s,k}$, and its incident and exitant direction vectors, $\omega_{i,k}$ and $\omega_{r,k}$.

$$f_{s,k} = \sum_j \rho_{s,j} \left(\mathrm{C}_{\mathrm{x,j}} \omega_{\mathrm{i,x,k}} \omega_{\mathrm{r,x,k}} + \mathrm{C}_{\mathrm{y,j}} \omega_{\mathrm{i,y,k}} \omega_{\mathrm{r,y,k}} + \mathrm{C}_{\mathrm{z,j}} \omega_{\mathrm{i,z,k}} \omega_{\mathrm{r,z,k}} \right)^{n_j} \qquad \forall k \tag{4.6}$$

Since my chosen form of the Lafortune representation shares C_x, C_y, C_z and n across all color channels I prepare to estimate these by first computing the luminance-weighted average of each $f_{s,k}$ (see Equation (2.3)). I refer to these as $\overline{f}_{s,k}$ and I fit the lobe parameters to these. The entire nonlinear optimization is thus performed on scalar data:

$$\overline{f}_{s,k} = \sum_j \left(C_{x,j}\omega_{i,x,k}\omega_{r,x,k} + C_{y,j}\omega_{i,y,k}\omega_{r,y,k} + C_{z,j}\omega_{i,z,k}\omega_{r,z,k} \right)^{n_j} \qquad \forall k \tag{4.7}$$

A single $\rho_{s,*}$ will be computed later for the color of all lobes.

Since the remainder of the Lafortune representation consists of a sum of nonlinear equations, the coefficients must be estimated using a nonlinear optimization method. I do this using either a new method that I propose or using the Levenberg-Marquardt (L-M) method (Presse, Flannery et al. 1988), as did Lafortune et al. (Lafortune, Foo et al. 1997) and Lensch et al. (Lensch, Kautz et al. 2001).

For either method, the parameters to be fit nonlinearly are C_x, C_y, C_z and n for each lobe j, and optionally the direction of anisotropy, β. C_y and β only apply to anisotropic materials.

4.3.1. Levenberg-Marquardt Nonlinear Optimizer

L-M requires an initial guess for each parameter. I initialize one lobe for forward scattering in the reflection direction: C_x=-1, C_y=-1, C_z=1, n=10. If more than one lobe is desired I initialize the second lobe for back-scattering: C_x=1, C_y=1, C_z=1, n=5. Additional lobes are initialized to random values. I initialize the angle β to 0.

The L-M optimizer requires the Jacobian matrix of Equation (4.7). I compute the Jacobian numerically by evaluating the model at all data points k with a small epsilon step for each parameter value. For optimizations that include the computation of β, the evaluation of the model within the Jacobian computation must take β into account. I do this by evaluating Equation (4.7) is then evaluated for all k using C' from Equation (4.4) instead of C.

This method is very computationally intensive to run at each pixel, as the timing results in Table 3.4 illustrate. For this reason it is not currently suitable for use in an interactive tool such as a paint program plug-in. I use the L-M method only in offline batch

jobs and for fitting one or a few pixels within an interactive program. I also experienced some stability problems. Some resulting SBRDFs have sharp discontinuities or outlying pixels, especially for the specular exponent parameter. Also, when using three or more lobes I observed significantly differing results when starting with different initial guesses. In general, I found the method quite suitable for one or two lobes, but not for three or more. Section 4.4 will further describe the results of the L-M method. When fitting one or two lobes choosing a random initial guess leads to the same solution in greater than 90% of the cases.

4.3.2. Line Search for Single Lobe

To address the slowness of the L-M method I propose a more efficient fitting algorithm that is constrained to a single specular lobe. It is also more stable since it includes only a single nonlinear parameter. This method is meant for use in an interactive application used by a digital content artist. It needs to be fast and robust; accuracy is secondary.

I perform a Brent line search (Presse, Flannery et al. 1988) over the specular exponent, n. For each value of n I compute the n^{th} root of $\overline{f}_{s,k}$:

$$\sqrt[n]{\overline{f}_{s,k}} = \mathrm{C_x}\omega_{\mathrm{i,x,k}}\omega_{\mathrm{r,x,k}} + \mathrm{C_y}\omega_{\mathrm{i,y,k}}\omega_{\mathrm{r,y,k}} + \mathrm{C_z}\omega_{\mathrm{i,z,k}}\omega_{\mathrm{r,z,k}} \qquad \forall k \tag{4.8}$$

This is a system of linear equations, one per reflectance sample k:

$$\begin{bmatrix} \sqrt[n]{\overline{f}_{s,1}} \\ \sqrt[n]{\overline{f}_{s,2}} \\ \vdots \end{bmatrix} = \begin{bmatrix} \omega_{\mathrm{i,x,1}}\omega_{\mathrm{r,x,1}} & \omega_{\mathrm{i,y,1}}\omega_{\mathrm{r,y,1}} & \omega_{\mathrm{i,z,1}}\omega_{\mathrm{r,z,1}} \\ \omega_{\mathrm{i,x,2}}\omega_{\mathrm{r,x,2}} & \omega_{\mathrm{i,y,2}}\omega_{\mathrm{r,y,2}} & \omega_{\mathrm{i,z,2}}\omega_{\mathrm{r,z,2}} \\ \vdots & \vdots & \vdots \end{bmatrix} \cdot \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \tag{4.9}$$

The residual of the solution of this overdetermined linear system is used as the error functional for the line search. The C_x, C_y, C_z and n values corresponding to the minimum residual are returned as the lobe shape parameters.

This method is from two to twenty times more computationally efficient than Levenberg-Marquardt with a single lobe, as the timing results in Table 3.4 illustrate. Section 4.4 will further describe the results of this line search method.

4.3.3. Computing ρ_s

Finally the algorithm computes the $\rho_{s,*}$ RGB vector. Although the data representation stores a different ρ_s for each lobe, I fit a single ρ_s to the sum of the lobes. This is a result of having fit the lobes to the luminance of the specular residual, $\bar{f}_{s,k}$. All lobes are fit simultaneously to $\bar{f}_{s,k}$ so ρ_s must also be shared. I compute $\rho_{s,*}$ as

$$\rho_{s,*} = \frac{1}{m}\sum_{k=1}^{m}\frac{f_{s,k}}{\bar{f}_{s,k}} \tag{4.10}$$

4.4. SBRDF Fitting Results

The data fitting error is shown in Table 3.1 both for the line search algorithm and the Levenberg-Marquardt one- and two-lobe fits. The error measurements are in sr^{-1}, the units of the BRDF. For comparison, the luminance-weighted diffuse component of the paper grid sample averaged 0.30 sr^{-1}. The error is the RMS difference between measured and fit BRDF values over all sampled directions.

From the reflectance hemispheres in rows 4 and 6 of Figure 3.11, all three fits appear to have essentially the proper behavior. The polar plots in rows 5 and 7 indicate that the line search algorithm tends to match the measured data well in magnitude and direction of the specular peak, but generally has much too broad specular peaks, with nearly all specular exponents being less than 4.0 for all materials. This is due to noise in small reflectance values being magnified out of proportion by taking the n^{th} root thereof in Equation (4.8). I verified this by sampling a Lafortune BRDF over its domain and fitting parameters to the samples. This yielded a substantially correct fit in all cases. I then added normally distributed random noise after sampling the function and again ran the fit. The fit would typically converge for noise standard deviations less than 10^{-4} sr^{-1}, but not for larger errors. Using L-M, on the other hand, the solution would usually converge for noise levels as 10^{-2} sr^{-1}.

The Levenberg-Marquardt fits, on the other hand, have a much lower RMS error than the line search fit, often by a factor of two to four. As seen in the polar plots, the shape mimics that of the measured data well, especially when two lobes are used. The RMS error for two lobes tends to be 10% to 50% less than for one lobe. This indicates that two lobes

provide a better fit. However, for materials with a single distinct lobe, fitting a second lobe is of little additional benefit.

The rendered images of Chapter 6, especially Figure 6.6, show additional results of these data fitting algorithms.

4.4.1. End-to-End Comparison

In order to visually validate the end-to-end measurement and data fitting system that I employed I created and photographed a test scene and then rendered an approximation of the scene using measured input. The results are shown in Figure 4.3. I used a floating-point RGB software renderer with the measured illuminant of Section 3.5. The white fabric SBRDF uses the Levenberg-Marquardt fit with two lobes, without fitting the direction of anisotropy.

The basic BRDF-related effects of the white fabric are well-approximated. The highlight in the silk-like background threads appears nearly the same. The foreground threads are clearly more diffuse than the background threads and have a similar behavior. The thick, horizontal threads are clearly visible in both the real and rendered images, with some self-shadowing apparent in both. Perhaps the largest difference between the two pairs is the self-shadowing and bumpiness that is not fully captured in the SBRDF. Section 7.2.1 describes an extension to the SBRDF representation that could improve this effect.

Figure 4.3: *Left: Photographs of cylinder with white fabric. Right: Software rendered approximation of same scene geometry using measured SBRDF of white fabric. The top and bottom pairs differ only in light position.*

4.5. Anisotropy

Without fitting the direction of anisotropy, β, my system can measure anisotropic surfaces in which the direction of anisotropy at every point is parallel to a texture axis. This is shown in Figure 4.4, which uses the two lobe L-M fit, without fitting β.

The thread directions of this sample aligned well with the texture axes. When fitting β, the average fit $|\beta|$ was less than 10^{-3} degrees at over 90% of the pixels. So to test the fit of β I resampled the raw images again, this time at 23˚ rotation. A two lobe L-M fit including β yielded $|\beta| < 0.1$ degrees at over 90% of the pixels. I believe this disappointing result is mainly due to insufficient registration accuracy. The spatial registration errors on the order of the thickness of a thread or larger essentially broaden the specular lobes making it more difficult to compute their orientation reliably.

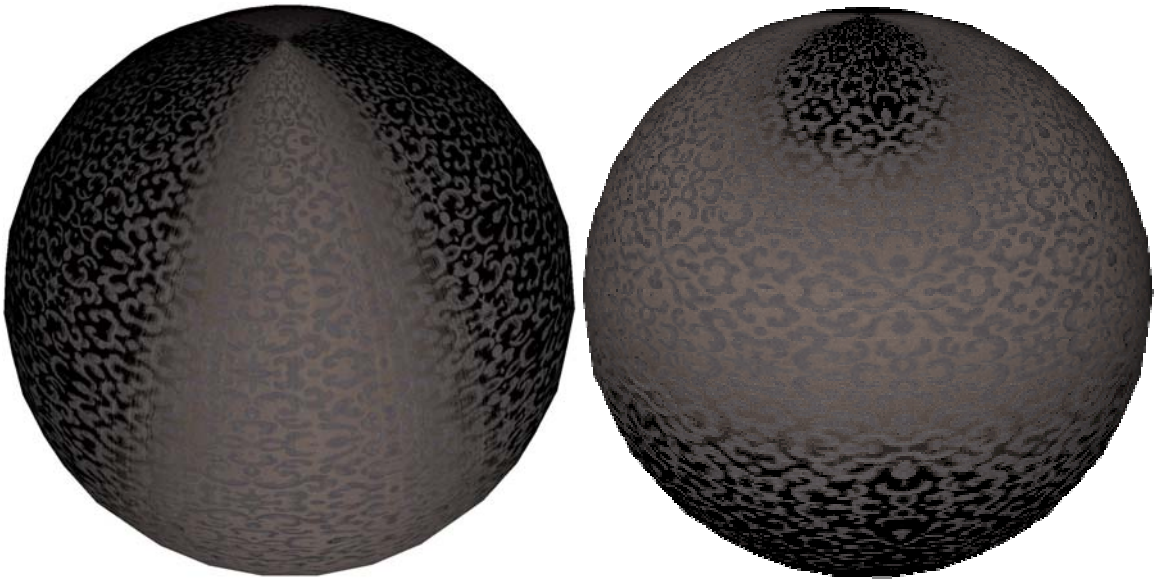


Figure 4.4: *Specular component of anisotropic White Fabric sample. In the left image the silk background threads run horizontally, creating a cat's eye-shaped highlight. In the right image the background threads run vertically, creating a crescent-shaped highlight. The cotton foreground threads are less specular, but still anisotropic, creating a broad, elongated highlight.*

4.6. Manually Synthesizing SBRDFs

Creating SBRDFs from scratch or by editing existing SBRDFs is an obviously useful capability, as it is for digital images in general. In the absence of such a program, I have implemented some simple SBRDF editing operations including fitting an average BRDF to the data at many SBRDF pixels, clustering pixels of like BRDF, interactively choosing BRDF parameters for painting, visualizing SBRDF channels, visualizing the SBRDF under illumination, mask-based compositing of multiple SBRDFs, etc.

Editing individual pixels of SBRDFs can take advantage of existing paint programs. This is done by quantizing the SBRDF (into, e.g., 256 unique BRDFs) and editing the palette index image in the paint program. After painting, the system resynthesizes the SBRDF by comparing the original and edited color index images and for any pixel with a different value, replacing the SBRDF pixel with the indexed BRDF from the palette. Ben Cloward, an artist, used this technique to make the Gift Wrap and Wall Paper SBRDFs properly tileable.

The expected small-scale blur caused by imperfect image registration and oblique sampling can be removed from the diffuse channel by replacing it with a single image of the surface. As with most SBRDF editing operations this has no physical validity, but can be useful nonetheless. I did this with the White Fabric sample in most example images. I chose one of the actual source images that had ω_i and ω_r vectors close to the normal. The new diffuse image should usually be normalized to the same average intensity as the original diffuse channel. This channel replacement technique could also be used with an image of the surface and fiducial markers scanned with a flat bed scanner to dramatically increase the resolution of the diffuse channel.

Figure 4.5: *Left: Diffuse channel of White Fabric sample computed from all input samples. Right: Source image used to replace diffuse channel in order to increase sharpness.*

A final SBRDF synthesis method that I have found to be successful is to synthesize the SBRDF channels from a standard RGB image. For example, Ben Cloward hand-edited a photograph of cherry wood to make a repeating tile pattern. After gamma correction I used this image as the diffuse channels of an SBRDF, set ρ_s to white, and manually chose lobe shape parameters for the SBRDF. I used a normally distributed random variable with mean

110 and variance 1 for the lobe exponent n. I set C_x=C_y=-1.01 and C_z=0.9. This causes reflectance to increase substantially for grazing angles while causing the diffuse reflectance to dominate at more vertical angles. To simulate differing reflection within the grain I thresholded the image and set n = 6 for the darkest pixels in the grain. The table top of Figure 6.11 uses this cherry wood SBRDF.

5. BRDF INTERPOLATION

Since I propose SBRDFs as a generalization of texture mapping it is important that the operations typically performed on texture maps may be supported as well on SBRDFs. Examples of high-level operations include image warping, MIP-map creation, texture mapping, painting, and image touchup. This chapter discusses the mathematical issues of the required operations on BRDFs and describes software methods I have employed for creating and operating on SBRDFs.

5.1. Interpolating BRDFs

The examples of high-level image operations that need to be enabled for SBRDFs all rely on the low-level operation of linear interpolation of pixel values. This is fundamental to the operation of resampling, wherein a pixel value is computed as a linear combination of nearby pixel values.

In image warping, the source image is resampled at subpixel locations to compute the pixel values of the warped image. In MIP-map creation the source image is convolved with progressively larger filter kernels, and the resulting image is resampled at progressively coarser scales. Each pixel becomes the weighted sum of the nearby pixels of the fine-resolution level. In texture mapping, the texture image is resampled at the subpixel coordinates of the result pixel's projection into the texture map. The pixel value is most often computed via trilinear blending of the four closest pixels of the closest two scaled images. A wide variety of digital paint operations have been invented (Cook 1984; Porter and Duff 1984; Smith 2001). Many involve a weighted blend of the existing pixel value with the pixel value of the paint being applied. Image touchup similarly has an enormous suite of operations, many of which involve linear interpolation of pixel values.

These key operations require interpolation of pixel values. This is simple for RGB colors since the components are linear in the resulting radiance. However, pixel values of an

SBRDF are parameters of the BRDFs. Usually the BRDF model is nonlinear, so linearly interpolating these BRDF parameters is invalid.

We generally desire an interpolation solution that gives results equivalent to interpolating the rendered result, i.e. interpolating exitant radiance. Since exitant radiance is linear in bi-directional reflectance the goal is a set of BRDF parameters, *q(t)*, that yield linearly interpolating reflectance values at all points on the domain:

$$f_{rt}\left(\omega_i \to \omega_r; q(t)\right) = t \cdot f_{r0}\left(\omega_i \to \omega_r; q(0)\right) + (1-t) \cdot f_{r1}\left(\omega_i \to \omega_r; q(1)\right) \qquad 0 \le t \le 1 \qquad (5.1)$$

Since in general no *q(t)* can satisfy this equation for all ω_i and ω_r simultaneously *q(t)* must be approximated. I propose different approximations for different situations. For the creation of MIP-map levels from measured data one could create SBRDFs separately for each MIP level from the original source data by simply choosing successively smaller resolution intermediate representations in Section 0.

For hardware rendering, I currently have little control over the interpolation of texels contributing to a given texture lookup. Common choices involve point sampling, linear interpolation, and linear MIP-map interpolation. Future hardware with programmable fragment shading could present the nearest texels to the fragment shader unmodified and allow the shader to interpolate them as part of the fragment program.

The SBRDF shader on graphics hardware allows all three texture sampling methods. Allowing the hardware to interpolate the parameters is often a reasonable approximation, especially considering that the shading cost to blend radiance values instead of BRDF parameters is linear in the number of texels evaluated, or 8x for tri-linear MIP-mapping.

5.2. Sampled BRDF Space

The desired result of BRDF interpolation is a BRDF whose *evaluated results*, rather than *parameters* interpolate the source BRDFs. The most straightforward way to do this is simply to do it – sample $f_{r0}(\omega_i \to \omega_r)$ and $f_{r1}(\omega_i \to \omega_r)$ over their domain and interpolate this vector of reflectance values, then fit $f_{rt}(\omega_i \to \omega_r)$ to the interpolated values at each *t*. For most interpolation needs in software, such as warping SBRDFs and pixel blending while painting,

this will presumably be the most accurate, yet slowest approach. The question may arise whether it is, in fact, always physically valid to interpolate sampled BRDF vectors.

Think of sampled BRDFs as vectors in a space of dimensionality proportional to the number of bi-directional samples in the BRDF representation. The dimensionality of this space could be in the millions for dense angular samplings. For the measured data acquired with the spatial gonioreflectometer, the dimensionality would range from 300 to 8000.

One can reason about the shape of the subspace occupied by all physically valid BRDFs. The shape of this subspace is important because linear combinations of physically valid sampled BRDFs could yield non-physically valid sampled BRDFs. For example, computing $f_{rt}(\omega_i \to \omega_r) = f_{r0}(\omega_i \to \omega_r) + t \cdot \left(f_{r1}(\omega_i \to \omega_r) - f_{r0}(\omega_i \to \omega_r) \right)$ for *t=-1000* is likely to yield negative BRDF values.

To begin to characterize this space I will show that it is physically valid to linearly interpolate between two sampled BRDFs consisting of BRDF measurements for any set of $<\omega_i, \omega_r>$ incident and exitant directions. From this, it is simple to show that all points in the convex hull of the set of physically valid BRDFs are physically valid.

A physically valid BRDF is one that follows the constraints of Section 2.3 – non-negativity, Helmholtz reciprocity, and conservation of energy. Linear interpolation between two physically valid sampled BRDFs yields a physically valid BRDF for positive interpolation weights that sum to one - that is, for interpolation on the line segment between the two BRDF vectors. I will prove this for each constraint on physical validity. The two physically valid sampled BRDFs to be blended are f_{r0} and f_{r1}. And

$$f_{rt}(\omega_i \to \omega_r) = f_{r0}(\omega_i \to \omega_r) + t \cdot \left(f_{r1}(\omega_i \to \omega_r) - f_{r0}(\omega_i \to \omega_r) \right) \qquad 0 \le t \le 1 \qquad (5.2)$$

Non-negativity: Since f_{r0} and f_{r1} are physically valid,

$$\begin{aligned} f_{r0}(\omega_i \to \omega_r) &\ge 0 \qquad \forall \omega_i, \omega_r \\ f_{r1}(\omega_i \to \omega_r) &\ge 0 \qquad \forall \omega_i, \omega_r \end{aligned} \qquad (5.3)$$

By the definition of linear interpolation we know

$$\begin{aligned} \min\left(f_{r0}\left(\omega_i \to \omega_r\right), f_{r1}\left(\omega_i \to \omega_r\right)\right) &\le f_{rt}\left(\omega_i \to \omega_r\right) \qquad \forall \omega_i, \omega_r \\ &\le \max\left(f_{r0}\left(\omega_i \to \omega_r\right), f_{r1}\left(\omega_i \to \omega_r\right)\right) \qquad 0 \le t \le 1 \end{aligned} \tag{5.4}$$

Thus,

$$f_{rt}\left(\omega_i \to \omega_r\right) \ge 0 \qquad \forall \omega_i, \omega_r, t \tag{5.5}$$

Reciprocity: Reciprocal BRDFs have

$$f_r\left(\omega_i \to \omega_r\right) = f_r\left(\omega_r \to \omega_i\right) \qquad \forall \omega_i, \omega_r \tag{5.6}$$

If the sampled BRDF space does not contain the reciprocals, then this physical property is not a constraint and is satisfied by default. If the space does contain the reciprocals, the following must be satisfied:

$$f_{rt}\left(\omega_i \to \omega_r\right) = f_{rt}\left(\omega_r \to \omega_i\right) \qquad \forall \omega_i, \omega_r, t \tag{5.7}$$

Since f_{r0} and f_{r1} are physically valid,

$$\begin{aligned} f_{r0}\left(\omega_i \to \omega_r\right) &= f_{r0}\left(\omega_r \to \omega_i\right) \qquad \forall \omega_i, \omega_r \\ f_{r1}\left(\omega_i \to \omega_r\right) &= f_{r1}\left(\omega_r \to \omega_i\right) \qquad \forall \omega_i, \omega_r \end{aligned} \tag{5.8}$$

Equation (5.2) can be rewritten as

$$f_{rt}\left(\omega_i \to \omega_r\right) = f_{r0}\left(\omega_r \to \omega_i\right) + t \cdot \left(f_{r1}\left(\omega_r \to \omega_i\right) - f_{r0}\left(\omega_r \to \omega_i\right)\right) \qquad 0 \le t \le 1 \tag{5.9}$$

which simplifies to Equation (5.7).

Conservation of Energy: In the sampled BRDF space we express the conservation of energy constraint from Equation (2.8) as a discrete sum:

$$\sum_{\omega_r \in \Omega_r} f_r\left(\omega_i \to \omega_r\right) \cos\theta_r \le 1 \qquad \forall \omega_i \tag{5.10}$$

This must be true at all $0 \le t \le 1$. For each ω_i, we shall consider a subvector $f_r(\omega_i \to *)$ consisting of all bi-directional reflectance values with incident direction ω_i. We then define

$$f_c\left(\omega_i \to *\right) = f_r\left(\omega_i \to *\right) \cos\theta_r \tag{5.11}$$

where θ_r is the polar angle corresponding of the ω_r corresponding to each ω_i. The constraint is then

$$\left\| f_c\left(\omega_i \to *\right)\right\|_1 \le 1 \qquad \forall \omega_i \tag{5.12}$$

Since conservation of energy holds at f_{r0} and f_{r1}, we know

$$\begin{aligned} \left\| f_{c0}\left(\omega_i \to *\right)\right\|_1 &\le 1 \qquad \forall \omega_i \\ \left\| f_{c1}\left(\omega_i \to *\right)\right\|_1 &\le 1 \qquad \forall \omega_i \end{aligned} \tag{5.13}$$

We can linearly interpolate each subvector $f_{ct}(\omega_i \to *)$:

$$f_{ct}\left(\omega_i \to *\right) = \left(1-t\right)\cdot f_{c0}\left(\omega_i \to *\right) + t \cdot f_{c1}\left(\omega_i \to *\right) \qquad 0 \le t \le 1 \tag{5.14}$$

By linearity of vectors we know

$$\left\| f_{ct}\left(\omega_i \to *\right)\right\|_1 = \left(1-t\right)\cdot \left\| f_{c0}\left(\omega_i \to *\right)\right\|_1 + t \cdot \left\| f_{c1}\left(\omega_i \to *\right)\right\|_1 \qquad 0 \le t \le 1 \tag{5.15}$$

Since t and $(1-t)$ form a partition of unity,

$$\left\| f_{ct}\left(\omega_i \to *\right)\right\|_1 \le 1 \qquad 0 \le t \le 1 \qquad \forall \omega_i \tag{5.16}$$

These proofs show that the line segment between f_{r0} and f_{r1} consists entirely of physically valid BRDFs. Since the entire line segment is valid, the triangle formed by this line segment and a third physically valid BRDF is valid. The entire convex hull of the set of physically valid BRDFs can be constructed from such triangles and is physically valid. And by interpolating points on the hull, the entire interior of the convex hull is physically valid. Thus, all linear interpolation operations performed between a set of physically valid sampled BRDFs are physically valid when all weights are non-negative and sum to one.

6. RENDERING SBRDFS

This chapter discusses the rendering of 3D models with SBRDF materials mapped to their surfaces.

6.1. Previous Work

Kautz and Seidel (Kautz and Seidel 2000) propose a widely applicable paradigm for evaluating spatially varying BRDFs in graphics hardware in which the BRDF equation is divided into linear operations such as dot products that can be evaluated by the graphics hardware, and nonlinear operations that are precomputed over their domain and stored in lookup tables as texture maps. Transformed BRDF parameters are stored per texel in texture maps and used for hardware shading. The SBRDF rendering method that I propose follows this paradigm since it directly stores BRDF parameters per texel. Section 6.8 compares the proposed specialization of this paradigm to the models explored by Kautz.

Other fairly simplistic methods of varying portions of the BRDF over the surface for use in graphics hardware include standard diffuse texture mapping, interpolating the specular albedo across a polygon, and an assortment of per-pixel lighting methods surveyed by Taylor (Taylor 2001).

6.1.1. Factorization Methods

Heidrich (Fournier 1995; Heidrich and Seidel 1999) proposed factoring BRDF equations into simpler functions that could be precomputed into texture maps and rendered using compositing or multitexturing arithmetic in a constant number of rendering passes per light. Kautz and McCool (Kautz and McCool 1999) perform factorization numerically using singular value decomposition. McCool et al. (McCool, Ang et al. 2001) solve problems of previous factorization approaches, with an elegant factorization that includes only positive factors. This method also works directly with arbitrary scattered bi-directional reflectance

samples. By performing the factorization in log space, the highlights tend to be smoother, avoiding aliasing (though having smaller peaks).

These factorization methods all apply to surfaces with a uniform BRDF over the surface, whereas the SBRDF method and that of Kautz and Seidel store BRDF parameters over the surface, rather than evaluating the BRDF at regular parametric intervals and storing the results in texture maps. This is the key characteristic that enables total variation of the BRDF over the surface.

However, some spatial variation can be achieved in factorization methods by using diffuse textures, or a weighted blend of several BRDFs. This is done by rendering passes for each BRDF for each light and accumulating the weighted results. In practice, this only works well for a small number of BRDFs because of the rendering cost of the layers and the storage space for the weight textures. This is explained in comparison to the proposed method in Section 6.8.

One advantage of the BRDF factorization methods is that very flexible BRDFs may be represented, since each texel is essentially a coefficient of a BRDF representation. This means that these methods typically do not require a number of rendering passes proportional to the complexity of the BRDF, whereas the number of passes for the Lafortune representation used in SBRDFs depends on the number of lobes. BRDFs with a single lobe that has a depressed center, such as velvet, may require several Lafortune lobes to adequately approximate.

Peercy et al. (Peercy, Airey et al. 1997) describe a method for implementing bump mapping in hardware that supports Phong shading. Their method computes the tangent space at each vertex, transforms the light and halfangle vectors into this space and interpolates them across the polygon. At each pixel they are normalized. The perturbed normal is encoded in a texture map, sampled at each vertex, and used along with the light and halfangle vectors at the vertex to compute the shading. Although I know of no Phong shading hardware that was built following this paper, the paper inspired several bump mapping implementations based on encoding transformations of the normal in a texture map.

An alternative approach for creating visually rich materials would be to use procedural shaders (Hanrahan and Lawson 1990), which has recently become feasible for

graphics hardware (Rhoades, Turk et al. 1992; Olano and Lastra 1998; Peercy, Olano et al. 2000; Proudfoot, Mark et al. 2001). While this approach provides the most flexible method of defining surface appearance, any shader modification beyond simply changing parameter values requires significant programming skills.

6.2. Rendering Concepts

To use SBRDFs in rendering, the SBRDF surface representation is first mapped to the surface of a geometric model. This mapping is defined by the designer of the model to be rendered. It simply consists of specifying texture coordinates for each vertex of the model. Texture coordinates may be computed as a function of the position of the vertex, either by the modeling software, the rendering application, or the geometry processing unit of the graphics hardware. See Section 6.4 for a description of the graphics hardware used as a reference for this work. Most often, texture coordinates are specified by the model designer within the modeling software. The SBRDF may be intended to be tiled across the surface of the model, as with the repeating pattern of an upholstery fabric, or mapped without tiling to specific geometry, as with the label on an aluminum can.

In the case of polygonal models the texture coordinates are only defined at the vertices, but the mapping from points $\vec{X}$ on the object surface to points <*s*, *t*> in the *u,v* texture space is well defined at every $\vec{X}$ by interpolating the texture coordinates at the vertices of the polygon containing $\vec{X}$. In the case of parametric models such as NURBS the texture coordinates are defined as a function of the *u,v* surface parameter space.

The definition of a complete scene to be rendered usually consists of several models. The models may have different surface representations, only some of them represented as SBRDFs. Once the scene is defined it is loaded by the rendering application and rendered from a particular camera pose. Rendering surfaces with SBRDFs is typically done either by an inverse renderer such as a ray tracer, or a forward renderer, such as common computer graphics hardware. In both cases the renderer computes a mapping from scene geometry to image pixels, and computes the color of each pixel based on a specified shading algorithm and its inputs. A conceptual description of the shading algorithm for SBRDFs follows.

6.3. SBRDF Shading

The SBRDF shading method is easiest to understand within the ray tracing framework and will be presented first in this way. The discussion of SBRDF shading in graphics hardware begins in Section 6.5.

In a ray tracer (Whitted 1980) a pixel's color is computed as a function of the radiance (color) of rays passing through the camera center of projection, P_v, from the direction of the pixel. The key operation within ray tracing then becomes the computation of a ray color, L_R, for a ray R. Since this is a query operation – querying the radiance of the ray R – the query ray actually points in the direction opposite the flow of the light whose radiance is being computed. When R intersects an SBRDF surface at a location $\vec{X}$, the SBRDF shading computation is performed at $\vec{X}$, yielding the radiance L_R of the ray R.

The shading computation begins by computing the world space location of the surface intersection point, $\vec{X}$. A local coordinate frame at $\vec{X}$ must then be computed in order to define ω_i, and ω_r. This frame at $\vec{X}$ consists of the surface normal $\vec{N}$, the tangent in the u direction of texture coordinates on the surface $\vec{T}$, and the binormal $\vec{B}$, which is perpendicular to both $\vec{N}$ and $\vec{T}$. $\vec{T}$, $\vec{B}$, and $\vec{N}$ form a right-handed orthonormal basis with origin at $\vec{X}$.

For a parametric surface the computation of $\vec{T}$, $\vec{B}$, and $\vec{N}$ follows naturally from the surface parameters and the differential geometry of the surface. For polygonal models, $\vec{N}$ and $\vec{T}$ at polygon vertices must be specified at model creation time or may be estimated in software using finite differences. Computing $\vec{N}$ from model geometry is well understood. Since computing $\vec{T}$ has only been needed recently as more advanced shading algorithms have been used, I include one method of computing $\vec{T}$ here (Taylor 2001)[15]. As with computing vertex normals, vertex tangents are computed simply as a blend of the face tangents that share the vertex. For a face consisting of vertex positions P_0, P_1, P_2, with texture coordinates $<s_o,t_0>$, $<s_1,t_1>$, $<s_2,t_2>$, the face tangents are computed as follows:

[15] I had to make some mathematical corrections.

$$\begin{aligned}
\vec{V}_1 &= P_1 - P_0 \\
\vec{V}_2 &= P_2 - P_0 \\
dU_1 &= s_1 - s_0 \\
dU_2 &= s_2 - s_0 \\
\vec{B} &= \frac{dU_1\vec{V}_2 - dU_2\vec{V}_1}{\left|dU_1\vec{V}_2 - dU_2\vec{V}_1\right|} \\
\vec{T} &= \vec{B} \times \vec{N}
\end{aligned} \tag{6.1}$$

The exitant direction, ω_r, is then computed as $\omega_r = -R$ since the query ray R points toward the surface but the exitant direction for the radiance computation points out of the surface. Likewise, $L_R = L(\omega_r)$, the exitant radiance in direction ω_r as defined below.

The next step in the computation of L_R is to evaluate the constant parameters of the BRDF equation at $\bar{X}$: ρ_d, $\rho_{s,i}$, $C_{x,i}$,$C_{y,i}$, $C_{z,i}$, n_i. These parameters are obtained from the SBRDF texture map at $<s_x,t_x>$, the texture coordinates corresponding to point $\bar{X}$. Section 5.1 discussed interpolation of BRDF parameters from nearby texture sample locations since $<s_x,t_x>$ will not generally fall exactly on a texture.

Given the BRDF parameters, L_R is then computed using the Reflectance Equation, from Equation (2.5):

$$L(\omega_r) = \int_{\Omega_i} f_r(\omega_i \to \omega_r) L_i(\omega_i)(\vec{N} \cdot \omega_i) d\omega_i \tag{6.2}$$

This integral is evaluated in different ways by different renderers. A standard ray tracer will only evaluate light directions $\omega_{i,l}$, with $L_i(\omega_{i,l})$ being the radiance from light l, taking into account distance attenuation and shadowing. The exitant radiance $L(\omega_r)$ is the sum of the exitant radiance due to incident radiance from each light l:

$$L(\omega_r) = \sum_l f_r(\omega_{i,l} \to \omega_r) L_i(\omega_{i,l})(\vec{N} \cdot \omega_{i,l}) d\omega_i \tag{6.3}$$

A Monte Carlo ray tracer evaluates the integral by probabilistically choosing directions ω_i and evaluating $L_i(\omega_i)$ by recursively tracing rays in these directions. The value of the integral is then the weighted sum of these sampled radiance values. Shirley provides a good discussion of Monte Carlo integration for ray tracing (Shirley 2000).

Figure 6.1: *Rendered SBRDF surfaces using Utah Real-Time Ray Tracer.*

I directly implemented this SBRDF shader in the Utah Real-Time Ray Tracer (Parker, Shirley et al. 1998; Shirley 2000). Figure 6.1 shows an example. This implementation uses point sampling of texture maps to avoid interpolation of nonlinear parameters. Thus, radiance values are computed at each point $\bar{X}$ based on BRDF parameters taken from the nearest texture element. Under texture magnification this yields blocky artifacts at the transition between nearest texels. However, shading at each screen pixel is still smooth within each block. Under texture minification filtering has no blocky artifacts. Each primary ray samples the nearest texel, but all primary rays within each pixel's aperture are convolved with the filtering kernel, reducing the artifacts associated with texture minification when point sampling.

6.4. Description of Current Hardware

Wolfgang Heidrich and I implemented SBRDF shading in graphics hardware[16]. To understand the SBRDF rendering method for graphics hardware the following description of the Nvidia GeForce 4 (NVIDIA 2002) architecture is provided. The architecture is a strictly

[16] As this is my dissertation it is important to specify Wolfgang's rendering contributions vs. my own. I defined the SBRDF representation and the layout of the SBRDF data as textures. Wolfgang implemented a three-pass rendering approach for a single directional light that included some computation on the host. A student of Wolfgang's extended the implementation to a single local light using a vertex program. I reimplemented the shader using two passes, a correct evaluation of the irradiance, no host computation, the remapped exponentiation table, and any number of lobes and lights. The environment map shader is also my own.

forward-feeding data flow from vertex input to frame buffer output. The maximum number of operations that can be performed in rendering a pixel is strictly defined by the chip architecture, with no iteration allowed by the architecture. Figure 6.2 shows a block diagram of relevant portions of this hardware.

Geometry processing is performed by a pair of identical user-programmable geometry processors. These processors execute an assembly language with 17 opcodes. The opcodes are processed in order with no loops, jumps, or conditional execution. Registers in the geometry processor consist of a set of per-vertex inputs, a set of constant inputs, a set of temporary registers and a set of per-vertex outputs. All registers are IEEE 32-bit float four-vectors. A vertex program outputs the post-transform vertex position and other parameters used to setup and rasterize the screen triangle. The attributes that may be rasterized include primary color, secondary color, and four sets of texture coordinates.

The next two modules – the texture unit and the register combiners – comprise the pixel shader of the GeForce 4. The texture formats available for 3D rendering include one-channel 8-bit, two-channel 16-bit, and four-channel 8-bit textures. The texture shader consists of four texture units. Each texture unit takes texture coordinates as input, performs a texture lookup, and filters the looked-up texels to yield a resulting texel value (usually interpreted as a color). The texture lookup result may be passed to a register of the register combiners and to the next texture unit to be interpreted as texture coordinates or a perturbation to texture coordinates. The texture unit can handle arbitrary-sized 2D textures, power-of-two 2D textures, and cube maps. The latter two formats may have associated MIP-map pyramids for precomputed down-filtering.

The register combiners operate on fixed point four-vector data stored in a set of temporary registers. The values stored in the registers are rasterized colors or texture coordinates, texture lookup results, or constants specified by the application. Each combiner stage may be thought of as a single VLIW instruction (Mark and Proudfoot 2001). A combiner has a register mapped to each of its four inputs, A, B, C, and D. Inputs A and B are either component-wise multiplied or their dot product is computed and replicated across the color channels. C and D have one of the same two operations applied to them. The results of

the AB and CD operations are either added or muxed together. The final result may then be scaled and biased. A final combiner stage exists for specialized operations like per-pixel fog.

After the shader has computed the final color of the fragment, the raster operation unit (ROP) performs a series of operations between the fragment and the corresponding frame buffer pixel value. These include the alpha kill, Z test, stencil test, Z and stencil writes, and color blend or pixel write. The color blend is a linear blend between the incoming fragment and the frame buffer pixel value, with blend weights being computed by a selected hard-coded operation. The inputs and outputs to the color blend are all 8-bit four-channel fixed point colors. The video scan-out units can perform gamma correction on these values.

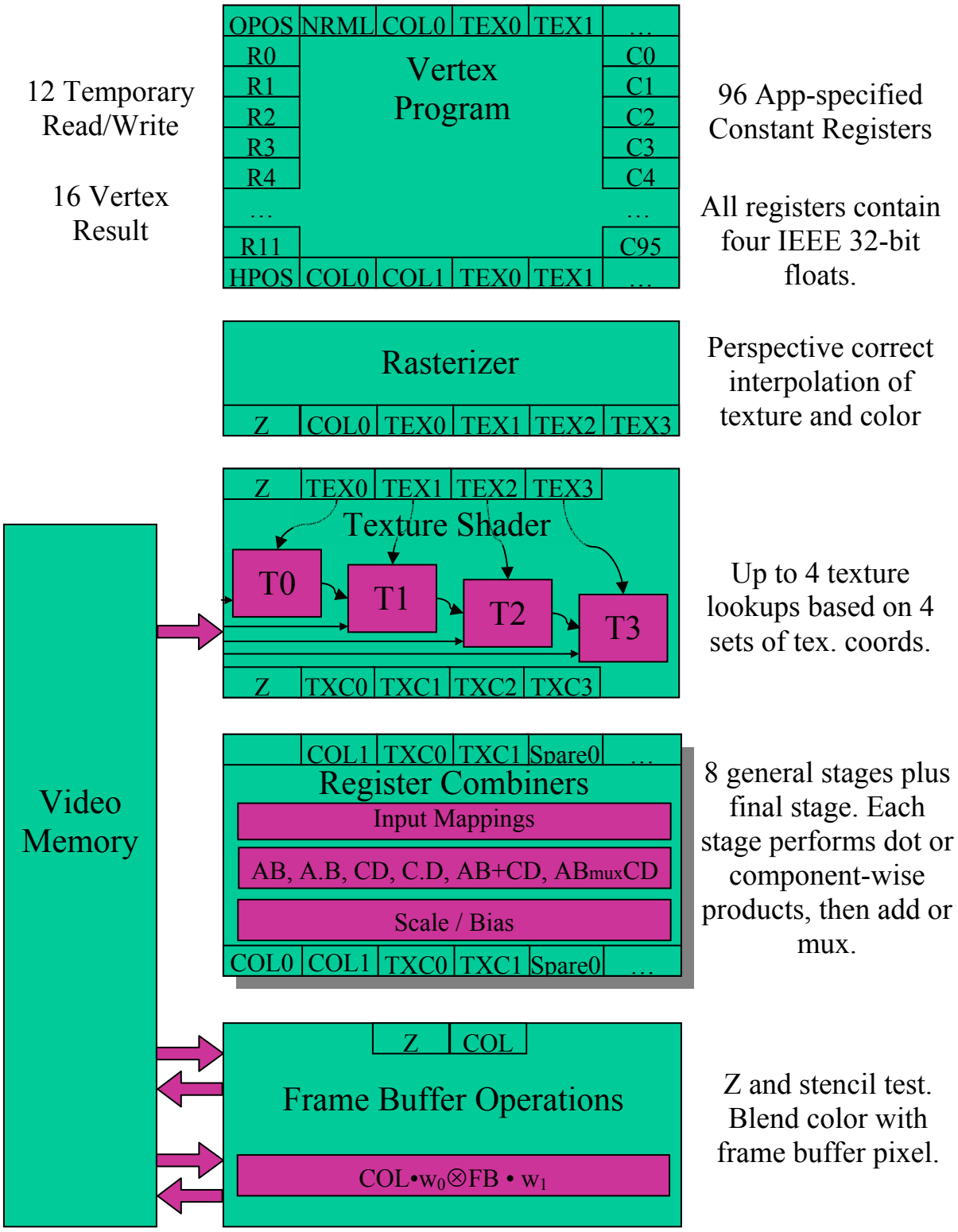


Figure 6.2: *Selected portions of the Nvidia Geforce 4 graphics hardware pipeline.*

6.5. *Representing SBRDFs as Texture Maps*

To prepare for hardware rendering using SBRDF shading, the SBRDF must be converted from its application memory representation to texture maps for use by the graphics hardware. Recall from Section 4.2 that the Lafortune BRDF representation is

$$f_r\left(\omega_i \to \omega_r\right) = \rho_d + \rho_m \cdot \delta\left(\omega_i, \omega_r\right) + \sum_j \rho_{s,j} \cdot s_j\left(\omega_i, \omega_r\right) \tag{6.4}$$

$$s\left(\omega_i, \omega_r\right) = \left(\begin{bmatrix} \omega_{r,x} \\ \omega_{r,y} \\ \omega_{r,z} \end{bmatrix}^T \cdot \begin{bmatrix} C_x & & \\ & C_y & \\ & & C_z \end{bmatrix} \cdot \begin{bmatrix} \omega_{i,x} \\ \omega_{i,y} \\ \omega_{i,z} \end{bmatrix} \right)^n \tag{6.5}$$

As mentioned in Section 0, bi-directional reflectance values range from 0 to infinity (Hanrahan 1993). The typical dynamic range of the Lafortune representation parameters is somewhat more constrained, making it more feasible to map them to 8-bit values. However, the loss of precision and dynamic range is occasionally visible as excessive saturation, highlight discontinuities, and missing reflection at grazing angles.

The application stores a single three-channel texture map containing the ρ_d diffuse albedo values. ρ_d is reasonably well suited to an 8-bit fixed point representation, though clamping occurs for the uncommon case in which ρ_d is greater than one. The alpha channel of this map can be used for transparency. Transparency is not easily implemented in a multipass shading algorithm but the alpha channel can be used to kill transparent pixels. Killing pixels based on alpha from a texture map creates a jagged edge that current full-scene anti-aliasing hardware does not alleviate. Despite this drawback, this technique is used to make leaf-shaped silhouettes on the polygon leaf models in Figure 6.7.

For each lobe *j*, the application defines two additional texture maps. A three-channel map contains ρ_s and a four-channel map contains C_x, C_y, C_z, and n. If mirror reflection is to be enabled, ρ_m is loaded into a three-channel texture map.

Recall from Section 4.2 that since seven parameters are used per lobe but the lobe only contains six degrees of freedom, ρ_s at each pixel may be scaled by an arbitrary factor e. C_x, C_y, and C_z are then scaled by $\sqrt[n]{e}$. e is chosen in such a manner as to preserve numerical

accuracy at that pixel. For example, the application can avoid clamping by scaling ρ_s such that the maximum of its color channels is 1. Alternatively, ρ_s can be scaled up such that the minimum of the ρ_s color channels equals the minimum of the correspondingly scaled C_x, C_y, and C_z. This balances numerical precision between the two quantities.

The remaining parameters C_x, C_y, and C_z present the largest problem for representation in eight-bit fixed point. These are signed quantities, represented for hardware use by biasing by –128. This leaves only seven bits of precision. These values have typical ranges from –1.1 to 1.1 for C_x, and C_y, and from 0 to 1.1 for C_z, with the smallest magnitude values being about 0.1. These values affect the direction to which a BRDF incident or exitant direction is sheared before computing its dot product with the other direction vector. The resulting clamping for large magnitude values and precision loss for small magnitude values reduce the shader's ability to represent very high reflectance at grazing angles. At the cost of another bit of precision, C may be mapped using the hardware's -2..2 mapping to preserve C values greater than 1.

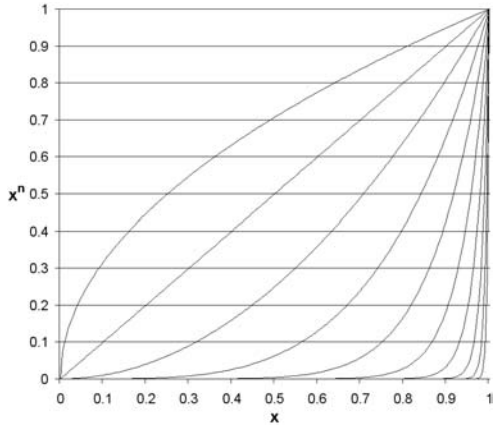


Figure 6.3: *Exponents x^n for various values of n. From top to bottom, n=0.5, 1, 2, 4, 8, 16, 32, 64, 128, and 256. Highlight shape changes very little for large n.*

The exponentiation within $s(\omega_i, \omega_r)$ cannot be performed directly in current graphics hardware, so, following Kautz and Seidel (Kautz and Seidel 2000), the application creates a 256 × 256 lookup table storing $f(x,n) = x^n$, and stores this in a texture. However, the hardware cannot sample this map using sub-texel precision because the indices only contain eight bits. Because of this it is important to avoid sharp discontinuities in this map. In particular, for large n, the highlight falloff is very steep as x diminishes from 1. The application thus remaps x as: $x' = x^2$. Also, specular highlights become smaller as the specular exponent increases.

Figure 6.3 shows that this change is much more rapid for small exponents, with specular exponents ranging from 0 to ∞. A highlight size falloff more linear in the sharpness parameter would be preferable. The lobe sharpness portion of the Lafortune representation could easily be replaced with an arbitrary sharpness function of the generalized dot product, such as a roughness parameter that varied from 0 to 1. However, my measured data uses the standard specular exponent formulation, so the renderer simply remaps n as $n'=255\ (n/255)^2$ to give more precision for smaller exponents.

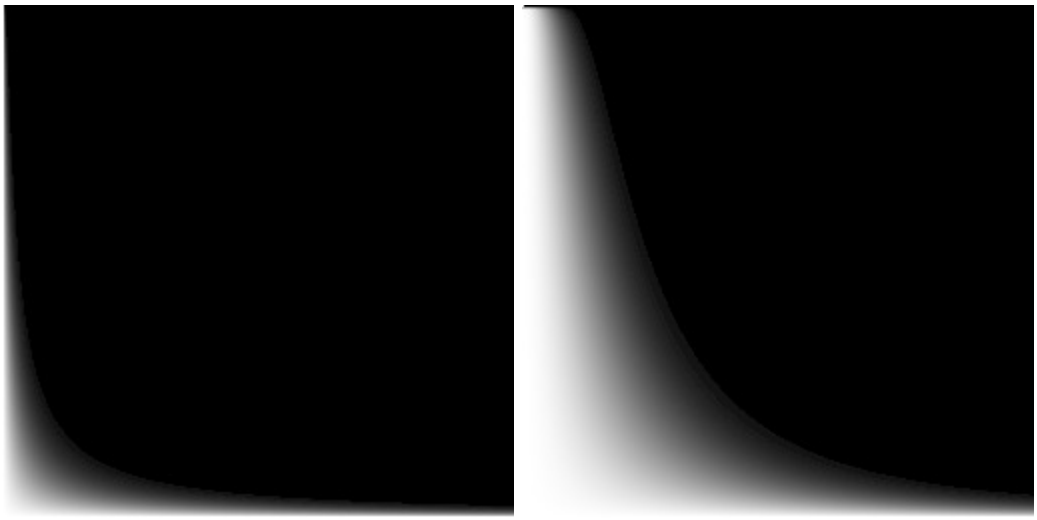


Figure 6.4: *Original and remapped exponent tables. N is on horizontal axis and x is on vertical axis. The goal is to have as smooth a gradient as possible in both dimensions.*

Figure 6.5: *Texture maps used in Figure 6.6(d): diffuse albedo (ρ_d), lobe albedo (ρ_s), lobe shape (C), and lobe exponent (n). Lobe shape maps -1..1 to 0..1. The lobe exponent is stored in the alpha channel of the lobe shape texture.*

The shading parameters are in a linear color space and the shader operates in a linear space. Thus, the graphics card must perform gamma correction on final frame buffer values for display. Most texture maps for game content have been created for use on non-gamma corrected displays, targeting the standard PC monitor gamma of 2.2. Thus, when loading

non-SBRDF texture maps into a rendering application that also uses SBRDF shading, the non-SBRDF textures must be un-gamma corrected so they do not appear washed out in the final display. This is done by applying a gamma value of 0.45.

6.6. Mapping the SBRDF Shader to Graphics Hardware

To implement Equation (6.2) on the graphics hardware described in Section 6.4, the equation must be expressed entirely using discrete arithmetic and divided into portions corresponding to hardware operations. Let us first substitute the Lafortune BRDF representation into Equation (6.2).

$$L(\omega_r) = \int_{\Omega_i} \left(\rho_d + \rho_m \cdot \delta(\omega_i, \omega_r) + \sum_j \rho_{s,j} \cdot s_j(\omega_i, \omega_r) \right) L_i(\omega_i) \left(\vec{N} \cdot \omega_i \right) d\omega_i \tag{6.6}$$

Recall that $s(\omega_i, \omega_r)$ is defined in Equation (6.5). Reordering the equation according to the terms of the BRDF yields

$$L(\omega_r) = \int_{\Omega_i} \rho_m \cdot \delta(\omega_i, \omega_r) L_i(\omega_i) \left(\vec{N} \cdot \omega_i \right) d\omega_i + \int_{\Omega_i} \left(\rho_d + \sum_{j=1}^{k} \rho_{s,j} \cdot s_j(\omega_i, \omega_r) \right) L_i(\omega_i) \left(\vec{N} \cdot \omega_i \right) d\omega_i \tag{6.7}$$

The ρ_m term is usually not used and is not particularly relevant to the present research. It may be enabled by simply rendering the object with the frame buffer color blending operation set to add the radiance of the diffuse and specular terms to the radiance from the mirror term using the standard reflection environment mapping method (Green 1986). For clarity the present discussion will omit this term, yielding

$$L(\omega_r) = \int_{\Omega_i} \left(\rho_d + \sum_{j=1}^{k} \rho_{s,j} \cdot s_j(\omega_i, \omega_r) \right) L_i(\omega_i) \left(\vec{N} \cdot \omega_i \right) d\omega_i \tag{6.8}$$

We next replace the integral over the incident hemisphere by a discrete sum over the hardware lights:

$$L(\omega_r) = \sum_l \left(\rho_d + \sum_{j=1}^{k} \rho_{s,j} \cdot s_j(\omega_i, \omega_r) \right) L_i(\omega_i) \left(\vec{N} \cdot \omega_i \right) \tag{6.9}$$

We then move ρ_d within the sum over the BRDF lobes by using a delta function to cause ρ_d to only be added once per light:

$$L(\omega_r) = \sum_l \sum_{j=1}^{k} \left(\rho_d \cdot \delta(j,1) + \rho_{s,j} \cdot s_j(\omega_i, \omega_r) \right) L_i(\omega_i) \left(\vec{N} \cdot \omega_i \right) \tag{6.10}$$

6.7. Details of Shader Implementation

Current graphics hardware can evaluate Equation (6.10) in two rendering passes per lobe per light. That is, the geometry using SBRDF surfaces must be rasterized multiple times with different portions of Equation (6.10) being evaluated in each pass and stored in video card memory, then subsequently combined, yielding a final frame buffer result. The pair of passes corresponds to the term within the double sum. These exitant radiance terms are summed in the frame buffer to yield the total exitant radiance.

First Pass

The first of each pair of passes is rendered into a p-buffer – a portion of video card memory that can store an off-screen image. P-buffers may be rendered to like a screen window and read from as texture maps. This first pass computes the lobe shape, $s(\omega_i, \omega_r)$, except for the exponentiation. The application sends vertex positions, normals, tangents, and texture coordinates for all objects with SBRDF surfaces. A vertex program computes the binormal, $\vec{B}$, based on the normal and tangent vectors, then transforms the camera and light vectors (for either point or directional lights) by the $\vec{T}$, $\vec{B}$, $\vec{N}$ frame into local surface coordinates. When normalized, this yields ω_r and ω_i. These are stored as texture coordinates in TEX1 and TEX2 and rasterized along with the standard surface texture coordinates. Also, the texture coordinates are passed unmodified to the rasterizer in TEX0. These vectors must be renormalized at each pixel. This is performed in the texture shader using a normalization cube map (Kilgard 1999). The results are stored in TXC1 and TXC2.

For each rasterized fragment, the texture shader also reads the lobe shape parameters indexed by TEX0 – C_x, C_y, C_z and n' and stores the result in TXC0. In the first register combiner stage ω_i (in TXC1) is component-wise multiplied by $<C_x, C_y, C_z>$ (in TXC0) and stored in the Spare0 register. In a second combiner stage the dot product of Spare0 and ω_r (in TXC2) is computed, yielding x, the result of the generalized dot product portion of $s(\omega_i, \omega_r)$. This value is squared in the final combiner to yield x', then converted to eight-bit fixed point

and replicated across the red, green, and blue components of the output; *n'* is written to the alpha output, and the resulting fragment is written to the p-buffer.

Second Pass

The second of the two passes performed per lobe per light computes the rest of one term of Equation (6.10). The vertex program associated with pass 2 transforms the vertex to screen space, passing these coordinates as texture coordinates. Interpolation of these texture coordinates must not use perspective correction so that the interpolated coordinates will precisely index the p-buffer pixel corresponding to the fragment now being rasterized. To disable perspectively correct texture interpolation the vertex program multiplies the texture coordinates by the vertex's *w* coordinate (Segal, Korobkin et al. 1992). The standard perspectively correct interpolation equation is

$$s_p(s,q,w) = \frac{in\left(\frac{s}{w}\right)}{in\left(\frac{q}{w}\right)} \tag{6.11}$$

where *in(x)* is linear interpolation of the value *x* at the current pixel from its specified values at the triangle vertices.

Perspective correction is disabled by multiplying the *s,t,r,* and *q* coordinates of TEX0 by the *w* coordinate of the pixel, yielding the linear interpolations of *s* and *t*:

$$s_n(s,q,w) = in\left(\frac{s}{q}\right) = \frac{in\left(\frac{sw}{w}\right)}{in\left(\frac{qw}{w}\right)} = s_p(sw,qw,w) \tag{6.12}$$

TEX2 and TEX3 both receive the standard surface texture coordinates. The last value computed by the vertex program is the irradiance due to light *l*. The light direction is computed as in pass 1. The dot product with the vertex normal is computed in world space. Negative values of the dot product represent a light behind the polygon and are thus clamped to 0. This value is multiplied by the radiance of the light, optionally including the inverse squared falloff, and the result is stored as a color in COL0.

The COL0, and TEX0 through TEX3 values are then rasterized and fed to the Texture Shader (see Figure 6.2). The p-buffer created in pass 1 is indexed by the screen coordinates to fetch the value written to this pixel in pass 1. This is *x', n',* which are used to perform a dependent texture read $s_j(\omega_i, \omega_r) = map(x', n')$. The result is stored in all three channels of the TXC1 output color sent to the register combiners. The lobe albedo and diffuse albedo at the given point are also sampled and passed to the register combiners in TXC2 and TXC3.

Pass 2 uses two register combiner stages. The first stage component-wise multiplies $\rho_{s,j}$ (in TXC2) by $s_j(\omega_i, \omega_r)$ (in TXC1) and stores the result in Spare0. The second stage maps Spare0 to input A, and the irradiance COL0 to inputs B and D. For the first lobe ($j=1$) the combiner maps ρ_d (in TXC3) to C and otherwise maps 0 to C. The output AB+CD is the exitant radiance for this term of Equation (6.10), and is sent to the frame buffer blending stage, with the alpha component of ρ_d being sent as the transparency of the fragment. The blending stage kills the fragment if its alpha value is transparent, or if the fragment fails the depth test. Surviving fragments, C_s, are accumulated with the corresponding frame buffer pixel C_d, using simple addition: $C_o = C_s * 1 + C_d * 1$.

6.8. Hardware Rendering Results

Figure 6.6 and Figure 6.7 show rendered results for a variety of measured and painted surfaces. These were hardware rendered using an Nvidia Geforce 4 card.

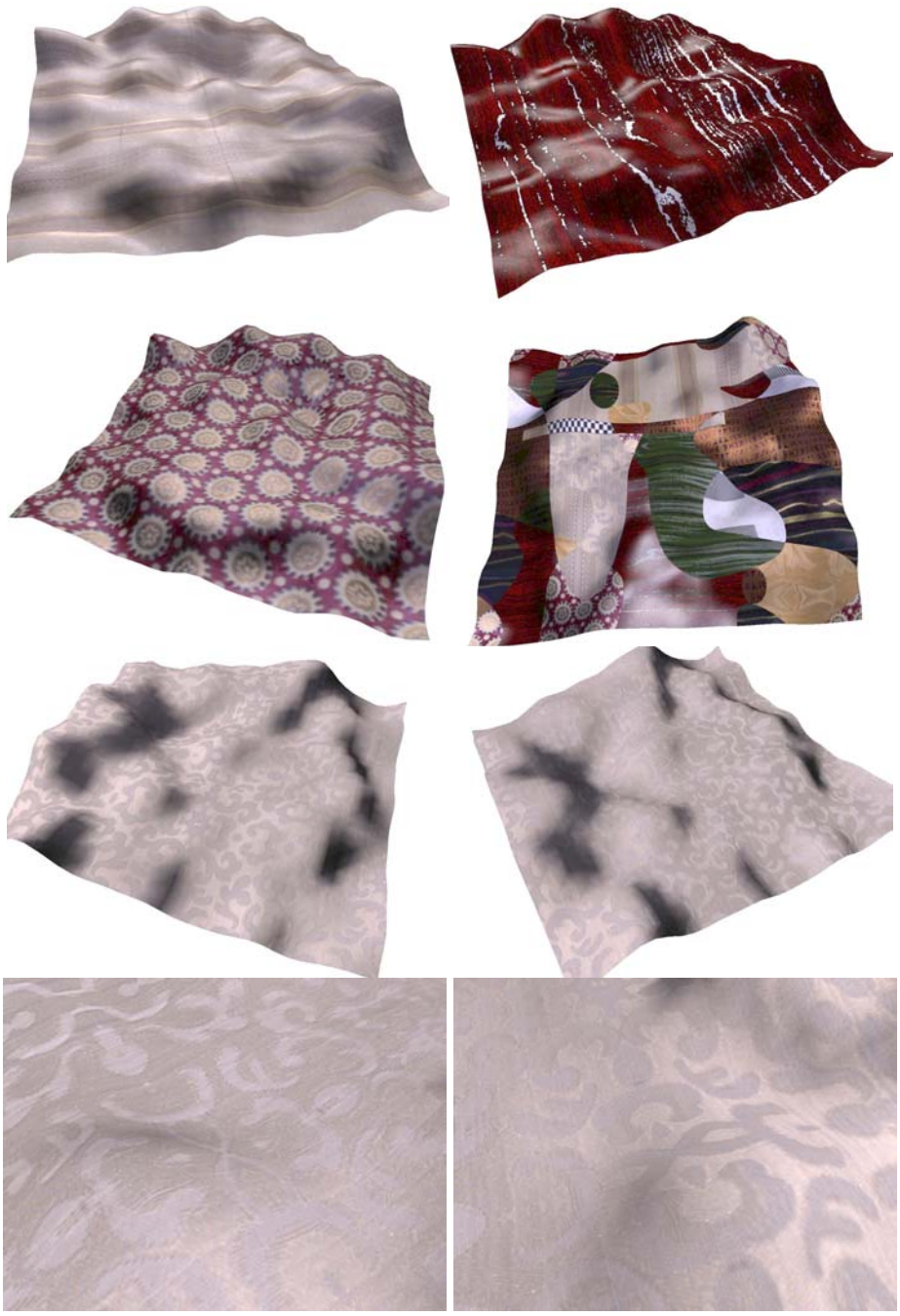


Figure 6.6: *Hardware rendered results using the method of this paper. Row 1: a) measured gilded wall paper, b) hand painted cherry wood. Row 2: c) measured gift wrap, d) SBRDF made by combining ten measured and synthetic SBRDFs. Row 3: Measured upholstery fabric with two lobes per texel. Note the qualitative change in appearance of the foreground and background threads under 90˚ rotation. Row 4: Detail of upholstery fabric, 30˚ rotation.*

Figure 6.7: *Hardware rendered result using SBRDF shader. The couch, tan chair, blue chairs, table cloth, leaves, brass table, and gift wrap have measured SBRDFs. The cherry wood and floor wood are painted SBRDFs. Surfaces have 1 or 2 lobes per texel. Three hardware lights are used. Average frame rate for this scene is 18 fps.*

To demonstrate the applicability of the SBRDF shader to interactive rendering with current graphics hardware I collaborated with Ben Cloward, a video game artist, to create a walkthrough of a complex scene. Ben modeled the lobby of the Carolina Inn shown in the environment map of Figure 6.9. The model consists of 221,925 vertices and 261,549 triangles. Some surfaces have simple texture maps, but most surfaces use SBRDFs accounting for 188,833 vertices and 213,828 triangles. Twelve different SBRDFs are used, consuming 26 MB of texture memory. On a 2500 frame walkthrough path that renders an average of approximately 25% of the geometry per frame, rendering performance is shown in Table 6.1.

	SBRDF			No light, no tex	Light, no tex	No light, tex	Light, tex
# lights	# lobes						
	1	2	3				
1	50.7	29.5	20.8	176.5	156.8	109.7	95.7
2	29.4	16.2	11.1		140.6		93.1
3	20.8	11.1	7.57		122.1		77.0

Table 6.1: *Frame rate for SBRDF shader, for the Figure 6.7 scene. 800×600, 2× AA. Although the model contains SBRDFs with 1 or 2 lobes, for timing tests, all surfaces were forced to the stated number of lobes. SBRDF results are compared against simple one-pass approaches.*

Especially for a single lobe, the rendering performance is very acceptable. One lobe is usually sufficient for interactive applications, with multiple lobes being more suitable for CAD applications where users typically study a surface more closely. Also, a single lobe is more acceptable within the SBRDF regime than for a spatially uniform surface since the high surface frequency can mask detailed highlight shape, as with my fabric samples. The white fabric of the lobby model uses two lobes. All other surfaces use one lobe.

The major source of visual artifacts stems from using the lookup table. The rounding to eight bits as x' and n' are stored in the p-buffer causes adjacent pixels with slightly different x' values to index the same table entry, yielding artifacts at highlight tails.

Another artifact results from the clamping of x' to 1 before being written to the p-buffer in the first pass. The Lafortune representation depends on dot product values $x > 1$ to represent the increasing reflectance at grazing angles of Fresnel reflection. If $n \geq 1$, for $x > 1$ then $x^n > x$, but for $x < 1$ then $x^n < x$. With x being clamped to 1 it is impossible to yield a bi-directional reflectance value x^n greater than 1. Using the exponentiation lookup table the lookup result will instead exactly equal 1. This value is then multiplied by the irradiance, which approaches zero as the light direction approaches the grazing angle, yielding a final exitant radiance that is usually less than 1. DirectX 9 class hardware resolves these artifacts. I have implemented a discrete-lights SBRDF shader for DirectX 9 class hardware. Because of the fragment shader's flexibility, many lobes and many lights can be evaluated by the shader in a single pass. This avoids the quantization of x and n since they are never stored in the frame buffer. Also, the x^n lookup table is avoided entirely, replaced by a floating point *pow* function. This removes the table-related quantization in the highlight and allows dramatically

improved Fresnel reflection for grazing angles since the bi-directional reflectance can now approach infinity instead of being clamped to 1.

With BRDF factorization methods, for a given surface to have different BRDFs at different points, each unique BRDF must be stored in a set of texture maps, and a number of rendering passes must be performed for each BRDF for each light, as mentioned by McCool (McCool, Ang et al. 2001). These results are modulated by per-texel weights for each BRDF in order to composite the per-BRDF renderings into a final result. For a very small number of BRDFs, this representation is compact, since it is essentially a palette with per-texel weights, and the rendering method works quite well.

However, an advantage of storing BRDF parameters per texel is that the number of required rendering passes does not depend on the number of different BRDFs. Table 6.2 compares the number of rendering passes required on the Nvidia Geforce 4 hardware for a varying number of lights, and for varying surface complexity. For SBRDFs, each texel has a unique BRDF, but the number of lobes is fixed for the surface (though not for a whole scene). For McCool's method, the total number of BRDFs on the surface is fixed, and each BRDF is evaluated at each texel. Figure 6.6(d) shows a single SBRDF made by combining many measured and synthetic SBRDFs. This SBRDF consists of over one million unique BRDFs since it is a 1024 × 1024 image. However, many texels are obviously very similar to others. I have not exploited this similarity, but a method like that of Lensch et al. would do so (Lensch, Kautz et al. 2001). By manually analyzing this SBRDF, I estimate that 48 significantly different basis BRDFs appear in the SBRDF. Assuming that factorization methods such as McCool's could be extended to handle the per-texel deviation from the bases and still render in N+1 passes per BRDF for N lights, the final column of Table 6.2 shows the estimated number of passes to render Figure 6.6(d) using McCool's method. The values are comparable for other BRDF factorization methods. This shows that rendering surfaces with BRDF parameters per texel is more efficient than rendering basis BRDFs for surfaces with a great deal of spatial BRDF variation. On more programmable hardware, these numbers could be reduced to a constant cost by storing BRDF indices per texel rather than weights, and evaluating only the indexed BRDFs at each pixel.

	SBRDF			McCool			
# lights	**# lobes**			**# BRDFs**			
	1	**2**	**3**	**1**	**2**	**3**	**48**
1	2	4	6	2	4	6	96
2	4	8	12	3	6	9	144
3	6	12	18	4	8	12	192
4	8	16	24	5	10	15	240

Table 6.2: *Number of passes required for SBRDF shader vs. the factorization method of McCool et al. for varying number of lights and varying surface complexity on an Nvidia Geforce 4.*

The SBRDF rendering method proposed here fits the paradigm of Kautz and Seidel (Kautz and Seidel 2000) in that linear operations are used to compute texture indices. Nonlinear operations are stored in texture maps, and the results are used in additional linear operations, completing the BRDF evaluation. Kautz and Seidel demonstrated their method with a novel anisotropic Blinn-Phong model (Blinn 1978). The potential flexibility of the paradigm was shown using three other models, including that of Banks (Banks 1994). These are demonstrated with software simulations. The SBRDF rendering method shares some precision and clamping artifacts that Kautz and Seidel encountered. Dependent texture reads in the reference hardware of Section 6.4 are not as general as predicted by Kautz and Seidel, so applying their proposed ideas required significant modification, including the use of the p-buffer to store intermediate shader results.

Within the Kautz and Seidel framework, the major distinguishing factor that applies to current graphics hardware is the use of the Lafortune representation, which has several properties not possessed by the models explored by Kautz and Seidel. In particular, the lobe directions of the Lafortune representation can aim in arbitrary directions relative to the incident direction, which makes a sum of lobes much more effective than with Phong or Ward bases (Phong 1975; Ward 1992).

The Lafortune representation is particularly well suited to this paradigm. The proposed rendering method requires two passes per lobe per light on a GeForce 4, but could most likely by done in one pass on an ATI Radeon 8500, which allows some register combiner results to be used as dependent texture coordinates within a single pass. In hardware, the anisotropic Ward model would require at least ten texture reads and four

rendering passes per lobe per light, but likely only three passes on a Radeon 8500. The Banks model could be evaluated in two passes per light.

6.9. Spatially Varying Environment Reflection

Beyond illuminating SBRDFs with discrete point or directional lights, SBRDFs may be illuminated with incident light from all directions – global illumination. Global illumination algorithms such as ray tracing (Whitted 1980) and radiosity (Cohen and Wallace 1993) perform reflectance calculations at surface points based not just on specified luminaires included in the scene, but also irradiance sampled from the geometry of the surrounding environment. Because these algorithms require access to the scene geometry at illumination time, they have not yet been realized directly in graphics hardware, although a "radiositized" model with precomputed Lambertian diffuse illumination can be simply rendered using graphics hardware.

For computer graphics hardware, two simplified illumination implementations are often used. First, incident radiance may be assumed to only come from a discrete set of point or directional lights, with no interreflections between surfaces in the scene. Second, incident radiance may be stored in a map parameterized over all incident directions – an environment map. Both of these representations enable evaluation of the local reflection in constant time, which is what makes them suitable for hardware implementation. Using discrete point lights allows the direction to the light from the surface point being evaluated to be computed correctly at each point. However, mirror reflection is not possible, and glossy reflection only reflects the light sources, which yields specular highlights, but does not reflect the scene geometry. Environment maps, on the other hand, represent only the incident direction of the radiance, so it is as if all illumination comes from an infinite distance. Each point in the scene that is illuminated with the environment map receives the same illumination, invariant of point location.

The original use of environment maps (Blinn 1976) enabled mirror reflections. More recent work (Cabral, Olano et al. 1999; Heidrich and Seidel 1999) enables glossy reflection of environment maps by convolving the environment map *a priori* with portions of the BRDF equation. These methods are thus only suitable for a single BRDF per set of maps. Voorhies and Foran (Voorhies and Foran 1994) use environment maps to store the

precomputed highlights from directional lights convolved with a Phong BRDF. Bastos et al. (Bastos, Hoff et al. 1999) represent a sampling of the scene geometry using images with depth, and preconvolves these with spatially uniform portions of the BRDF for specific reflectors. This method allows reflections of local geometry rather than just light at infinity. Kautz et al. (Kautz and McCool 2000) create a set of prefiltered environment maps for a given BRDF by fitting lobes that are fixed in orientation relative to a selected exitant polar angle. Many such angles are used, creating a 3D environment map. This 3D map is indexed based on the exitant direction and exitant polar angle. When summing multiple lobes, each lobe uses a separate 3D environment map. This method has the important advantage of handling increasing lobe sharpness at grazing angles. Kautz et al. (Kautz, Vázquez et al. 2000) provide faster map convolution, enabling dynamically changing glossy environment maps, and anisotropic environment maps constrained to an approximate Banks model.

These techniques do not address different BRDFs at each point on a surface, except for a textured diffuse term, and potentially modulating the sampled environment map with a specular albedo.

Cabral and Heidrich (Cabral, Olano et al. 1999; Heidrich and Seidel 1999) both discuss the problem that the spherical environment map representation has a built-in view direction dependence, resulting in inadequate sampling in many directions. Heidrich and Seidel (Heidrich and Seidel 1999) address this with a new environment map parameterization defined as a reflection off two facing paraboloids. This parameterization sufficiently improves sampling density that it can be used from any viewpoint. Another view independent environment map parameterization is the cube map, in which a cube is centered about a point and each pixel of the six cube faces encodes the radiance arriving at the center through that pixel. The hardware of Section 6.4 includes dedicated cube map circuitry and also handles spherical and parabolic environment maps, which require no dedicated circuitry.

Previous methods for glossy reflection of environment maps in graphics hardware share the constraint that the environment map must be preconvolved with (a portion of) each different BRDF. Thus, a different set of environment maps is required for each different BRDF. Although burdensome, this is at least possible for scenes consisting of a fairly small number of objects, each with one or a few discrete BRDFs. However, in the spatially varying

BRDF regime, every point on every surface is treated as having a unique BRDF, so the existing methods do not directly apply. I present a method for rendering from preconvolved environment maps that allows each point on the surface to have a different BRDF, including a different glossiness at each point, and for each lobe.

The derivation begins with a modified form of Equation (6.8) that has separate diffuse and specular terms:

$$L(\omega_r) = \rho_d \int_{\Omega_i} L_i(\omega_i)(\vec{N} \cdot \omega_i) d\omega_i + \sum_j \rho_{s,j} \int_{\Omega_i} s_j(\omega_i, \omega_r) L_i(\omega_i)(\vec{N} \cdot \omega_i) d\omega_i \tag{6.13}$$

The incident radiance $L_i(\omega_i)$ is stored in the environment map, indexed by the incident direction. The diffuse term can be easily encoded in an environment map indexed simply by $\vec{N}$:

$$D(\vec{N}) = \int_{\Omega_i} (\vec{N} \cdot \omega_i) L_i(\omega_i) d\omega_i \tag{6.14}$$

$D(\vec{N})$ is independent of the BRDF, so it is precomputed once for all objects that are to reflect the environment map $L_i(\omega_i)$. This was done by Miller (Miller and Hoffman 1984) and by Green (Green 1986). The specular terms also take advantage of precomputed maps. Just as $\vec{N}$ is used to index the preconvolved diffuse map, a function of the view direction will index a preconvolved specular environment map:

$$p_j(\omega_r) = \begin{bmatrix} C_x & & \\ & C_y & \\ & & C_z \end{bmatrix} \cdot \begin{bmatrix} \omega_{r,x} \\ \omega_{r,y} \\ \omega_{r,z} \end{bmatrix} \tag{6.15}$$

$p_j(\omega_r)$ is the peak vector of the lobe-shaped sampling kernel – the incident direction of maximum influence on the exitant radiance toward ω_r due to lobe j. Equation (6.13) becomes

$$L(\omega_r) = \rho_d D(\vec{N}) + \sum_j \rho_{s,j} \int_{\Omega_i} (p(\omega_r) \cdot \omega_i)^{n_j} L_i(\omega_i)(\vec{N} \cdot \omega_i) d\omega_i \tag{6.16}$$

The specular environment map is

$$S\left(\omega_p, n\right) = \int_{\Omega_i} \left(\frac{\omega_p}{\left\| \omega_p \right\|} \cdot \omega_i \right)^n L_i\left(\omega_i\right) d\omega_i \tag{6.17}$$

As discussed below, this map is parameterized both on the incident kernel peak direction ω_p and on the exponent n. The exitant radiance formulation used in hardware rendering becomes

$$L\left(\omega_r\right) \approx \rho_d D\left(\vec{N}\right) + \sum_j \rho_{s,j} S\left(p\left(\omega_r\right), n_j\right) \left\| p\left(\omega_r\right) \right\|^{n_j} \left(\vec{N} \cdot p\left(\omega_r\right)\right) \tag{6.18}$$

The $\left\| p\left(\omega_r\right) \right\|^{n_j}$ factor arises because S is computed with a normalized ω_p, so the incident radiance must still be scaled by the magnitude of the lobe. This equation is only an approximation since the irradiance falloff $\vec{N} \cdot \omega_i$ must be computed inside the integral over ω_i, but this could not then be stored in an environment map since it would be parameterized by both $p(\omega_r)$ and $\vec{N}$. The proposed formulation instead weights all incident directions equally within the integral but weights $S(\omega_p,n)$ by $\vec{N} \cdot p\left(\omega_r\right)$. This problem and resolution were explained by Kautz and McCool (Kautz and McCool 2000). This is a high quality approximation, and the quality improves for increasing values of n. A problem the proposed method shares with many preconvolution methods is that by removing $\vec{N} \cdot \omega_i$ from the integral, some light from below the surface is included in $L(\omega_r)$. This has not presented a practical problem.

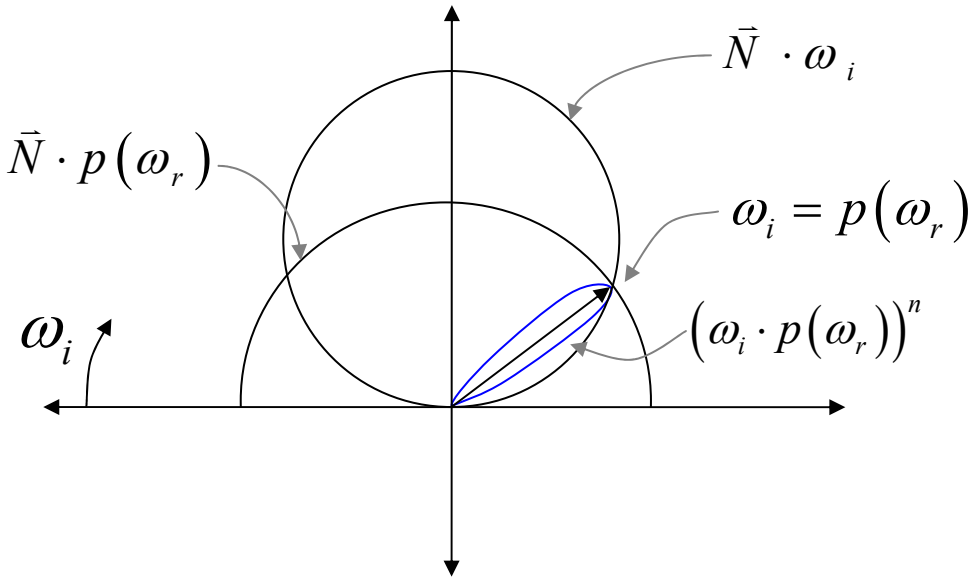


Figure 6.8: *$\vec{N} \cdot \omega_i$, evaluated within the integral, is approximated by $\vec{N} \cdot p(\omega_r)$, evaluated outside the integral. The approximation quality depends on the exponent, n.*

Two key facts enable us to achieve spatially varying BRDFs illuminated by environment maps. First, the Lafortune representation uses a specular exponent, which creates radially symmetric lobes. In general, only radially symmetric lobes may be used to preconvolve an environment map because otherwise the map would only be correct for a single normal direction. Second, the Lafortune representation directly computes a general lobe peak vector. Thus, all BRDF properties that the Lafortune representation expresses using the peak vector – anisotropic peak direction, anisotropic lobe shape (by summing radially symmetric lobes), the increasing reflectance of Fresnel reflection, and forward reflective, retroreflective, and off-specular peaks – are independent of the environment map.

Several representation alternatives arise for the $S(\omega_p,n)$ map. The possibilities are constrained by the graphics hardware's small choice of texture representations. One possibility is to represent *S* in a single cube map, with the most specular value of *n* being stored in the finest MIP level, and successively diffuse *n* being stored in coarser levels. The decreasing resolution of MIP levels corresponds nicely to the decreasing high frequency content of environment maps convolved with wider specular lobes. This property allows the most compact representation of *S.* Another approach is to store a small set of cube maps for discrete values of *n*, and render with each of these, weighting each by basis functions

evaluated at the pixel's n, yielding a result interpolated to the pixel's n from maps for nearby values of n. A third possibility is to use a 3D texture, with the s and t map dimensions mapping to a parabolic map (Heidrich and Seidel 1999) and the r dimension mapping to n. This is similar to the representation used by Kautz and McCool, except their r dimension mapped to exitant polar angle.

I believe glossy environment reflection of SBRDFs is impractical for current graphics hardware. Today's hardware does not allow choosing MIP levels based on a value sampled from a texture map, so there is no way to choose a gloss level based on the pixel's n. This prevents use of the single MIP-map representation. The 3D texture representation could be used, except it would be difficult to compute the indices into this map for each fragment being shaded. If the C_x, C_y, and C_z values were constant over the polygon, the s and t coordinates could easily be computed in a vertex program with a per-fragment n value possibly mapped to r. This would yield spatially varying gloss level, but would not enable spatially varying the lobe direction effects. The representation with several cube maps could be rendered with one pass per lobe per map level by using the bump map vector perturbation circuit to compute $p(\omega_r)$ at each fragment. However, the computation of fragment weights based on the fragment n and the current map's n would be difficult and may require different weight maps for each environment map n. Alternatively, n could be held constant here, yielding a modified form of bump mapped environment mapping.

For future hardware, I believe all three representations will be practical, since dependent texture lookups have been gradually becoming more general. Using a software renderer, I have implemented both cube map approaches. The shader directly follows Equation (6.18), with the note that the view vector is rasterized in local surface coordinates, transformed by C to yield $p(\omega_r)$, and then transformed by the local frame to world coordinates for sampling the environment map. This requires rasterizing $\vec{N}$ and $\vec{T}$.

For the set of discrete cube maps, I used linear interpolation between values of n for specular exponents 1, 4, 16, 64, and 256. The blending artifacts were minimal, but the large number of texture accesses makes this method less suitable for real-time rendering.

The MIP-mapped cube map representation works well. This method is the most space efficient, requiring a single map, and is also the most computationally efficient since the

texture coordinate computation is performed in a dedicated cube mapping circuit, rather than in register combiner passes. Cube map MIP levels generated via environment map convolution with different specular exponents are not equivalent to general MIP level computation. However, it is very visually similar and has not presented a problem.

I have not yet implemented the 3D texture representation with parabolic maps. Although the parabolic mapping would need to be computed in the fragment shader rather than a dedicated circuit, I believe the implementation should be straightforward and provide good results.

6.10. Environment Reflection Results

Figure 6.10 and Figure 6.11 are rendered results employing the SBRDF environment reflection technique, using the high dynamic range environment map of Figure 6.9. Note the synthetic texture of wood. The grain, which has a specular exponent of about 10, yields much lower gloss highlights than the wood foreground, with a specular exponent of about 150. The measured white upholstery fabric is represented with two lobes. At most texels, these have specular exponents of about 1 and 5 for the silky thread and about 1 and 2 for the cotton thread. For most silk texels the high exponent lobe is anisotropic – forward scattering when parallel to the threads, and back scattering when perpendicular to them.

Figure 6.9: *Left: One face of high dynamic range cube map used in figure 3. Right: Prefiltered maps for specular exponents (top to bottom, left to right) 256, 64, 16, 4, 1, 0 (diffuse). All prefiltered maps are 128 × 128 pixels.*

Figure 6.10: *Synthetic SBRDF with varying direction of anisotropy approximating circularly brushed metal. The SBRDF uses a three-lobe metal approximation.*

Figure 6.11: *Software rendering using the SBRDF prefiltered environment map technique. Specular exponents range from 1 to 5 for the couch, 5 to 15 for the grain of the wood, and 75 to 150 for the wood foreground. The wood is increasingly reflective at grazing angles.*

7. CONCLUSION & FUTURE WORK

I have defined the spatial bi-directional reflectance distribution function and implemented a device for measuring this function from real surfaces. I have analyzed the tradeoffs involved in the design of such a device, with one key result being a set of equations that relate sampling density in the BRDF domain to the size of highlights that can be adequately sampled. I have measured a variety of real surfaces, yielding spatially and angularly dense reflectance data that was used by me and can be used by future researchers.

I have applied existing methods to fitting this data to a very compact SBRDF representation and have developed a new method that overcomes the slowness of the prevailing approach, at the expense of accuracy. I have, for the first time, measured the BRDFs of surfaces with differing directions of anisotropy over the surface.

I have proposed a method of approximately linearly interpolating nonlinear BRDFs and have proven that interpolating between vectors of BRDF samples yields a physically plausible result.

With Wolfgang Heidrich I have implemented two novel methods of synthesizing images with SBRDF surfaces in graphics hardware. The first rendering approach allows far more general BRDFs to be rendered at each distinct point on the surface for a given rendering cost than competing approaches. I have also implemented a more flexible method of rendering prefiltered environment maps that allows a different BRDF at each point on the surface to correctly reflect the environment map in constant time using a single set of environment maps for all BRDFs.

These results combine to prove the thesis statement of this dissertation:

> *A spatially and bi-directionally varying surface reflectance function can be measured for real surfaces, compactly represented as a texture map of low-parameter BRDFs, and rendered at interactive rates.*

I believe that spatial bi-directional reflectance distribution functions of the variety I describe are a viable approach for representing surface appearance for computer graphics. I expect that SBRDFs will be used in rendering applications first with hand-painted SBRDFs and SBRDFs from my online library, and then with acquired SBRDFs as the technology for doing so becomes less costly and time consuming.

I now outline possible directions for future work.

7.1. Acquisition Alternatives

A suitable video camera could be used with the spatial gonioreflectometer. With short enough exposure times this could allow the motors to move continuously while the camera records images. High dynamic range could be handled by repeating the entire process at different shutter speeds or continually changing shutter speeds, performing the reconstruction from low to high dynamic range on the sampled SBRDF as a whole. This could significantly reduce acquisition time. I believe this alternative is nearly feasible as the bandwidth from video camera to PC increases and video camera CCD resolution increases. This would enable a closed loop in which the system could make pose and exposure decisions in real time based on the data returned.

I envision a simpler, less costly approach to acquiring SBRDFs that would be more suitable for digital content creators since it would allow surfaces to be acquired *in situ*, rather than in a lab. The new approach involves the following steps. Set a frame or stage with fiducial targets around the surface to be scanned. Control light as necessary with black curtains. Manually place the radiometrically calibrated light at various poses. Estimate the light position using reflecting spheres (Lensch, Kautz et al. 2001) or using triangulation from the tips of two pencils at known locations and the known locations of their shadows. For each light position acquire several camera poses. Estimate the camera pose using the fiducial markers. If the dynamic range of the scene requires more than one exposure per pose a tripod must be used. Otherwise, the camera can be hand held or a video camera could be used as by Gortler et al. (Gortler, Grzeszczuk et al. 1996). Processing of acquired poses is the same as for my spatial gonioreflectometer.

The acquisition system could also be enhanced to handle geometric objects (Sato, Wheeler et al. 1997; Lensch, Kautz et al. 2001).

7.1.1. The BRDFimeter

A small hand-held device could be built to measure the BRDF of a single point on a surface in a few seconds at most. Potentially the device could be light-weight and robust like an ultrasound probe or a hand-held scanner. I call the device a BRDFimeter.

The BRDFimeter has CCDs on the top and four sides of a hemi-cube. The volume surrounded by the CCDs will have optics that direct light from the point at the center of the cube base to the CCD sensels in such a way that each CCD sensel only measures photons coming from the center of the base – the surface point whose BRDF is being measured. The bottom face of the cube is black with a small aperture in the center.

Light will enter the system through optical fibers or other optics entering from between the edges of two CCDs. A discrete number of optical fibers will enter the system and only one will be illuminated at a time. A measurement is taken by illuminating each optical fiber sequentially and then reading the image from all the CCDs. This gives the exitant radiance hemisphere for each incident direction. Taking multiple exposures with different exposure times can allow high dynamic range measurements.

One problem with the hemicube configuration of CCDs is that an entire CCD covers the top of the cube so no light can enter there. To allow optical fibers to enter from directly above the sample a different configuration of CCDs could be chosen. One example is to have a 2x2 array of CCDs on the top, with optical fibers entering through the cracks between them. This will densely parameterize the exitant hemisphere, but not the incident hemisphere. The density will be more than sufficient for isotropic BRDFs, and will suffice only for some anisotropic BRDFs.

The BRDFimeter could be small and light enough that it could be a hand-held device. The user holds it against the surface to be measured and presses a button to acquire the BRDF. The BRDFimeter could also be placed on a scanning platform and used like a flatbed scanner. A planar sample like those I use is placed on the scanner bed and the BRDFimeter

scans across it, measuring the BRDF at each point. Such a device would have been very useful in this research.

7.2. Fitting and Representation

Extending the line search method to multiple lobes is certainly desirable, but the obvious greedy algorithm of subtracting the first lobe and fitting a subsequent lobe to the residual is not sufficient since the residual will not, in general, be lobe-shaped, but have multiple peaks. I believe that a greedy algorithm that began with more diffuse lobes and progressed toward sharper lobes would be more successful, just as estimating and subtracting ρ_d prior to estimating specular reflectance has proven successful for many researchers.

Also, fitting approaches for the Lafortune representation that first estimate the C matrix based on measured reflectance peaks, and then estimate an appropriate sharpness function seem fruitful. Since a BRDF is a continuous 4D function, the peak exists as a ridge in that space. Finding the ridge directly could form the basis of an efficient fitting method.

The Lafortune representation uses an exponentiated cosine as its sharpness function. An arbitrary function of the generalized dot product could be substituted. Useful examples might include an elliptical Gaussian to express azimuthally broader scattering using a single lobe, or a roughness function that varies from 0 for a perfect mirror to 1 for Lambertian diffuse, as does the HTSG model (He, Torrance et al. 1991). This range is more linear in lobe width than the specular exponent, which is a useful property in many instances, including the table lookup of Section 6.6 and interpolation of the sharpness parameter.

Likewise, the sharpness function would better model real scattering behavior if it became more narrow (mirror-like) at grazing angles. The Lafortune lobe becomes larger at grazing angles but does not become sharper. Interpolating between two sharpness parameter values, n_0 and n_1, based on polar angles could be accomplished by modifying Equation (4.2):

$$\begin{aligned} n &= n_1 t + n_0 (1-t) \\ t &= \frac{\bar{N}\cdot\omega_i + \bar{N}\cdot\omega_r}{2} \end{aligned} \tag{6.19}$$

Finally, I was careful to make the measured data as generally useful as possible. This included computing absolute radiance of the light and the surface measurements. This data

could be used for light field rendering. I also was careful to keep the light and camera as close to the surface as possible, within the constraints of Section 3.2.1 so that the data can be used to maximum benefit with fitting techniques that take spatial coherency into account.

7.2.1. Bump Mapping

The C matrix consists of simply the diagonal elements C_x, C_y, C_z and can perturb a vector arbitrarily within the local coordinate frame. However, it is not as general as bump mapping. The wrinkles in the Gift Wrap sample (Figure 6.6) show this. The C_x, C_y, C_z values are automatically fit to shear the lobe peak in the direction of reflection after perturbation of the normal. However, the highlight appears the same even under surface rotation about the normal. Malzbender et al. (Malzbender, Gelb et al. 2001) estimated the normal perturbation using sparser sampling than mine, so I believe tabulated SBRDFs have sufficient information to estimate the local normal.

For rendering SBRDFs with bump maps one could use two three-channel maps to represent the six unique elements of the symmetric matrix C, including the normal perturbation for bump mapping, rotation to the direction of anisotropy, and the reflection and shear by C_x, C_y, C_z. At each fragment, ω_r is simply transformed by this 3×3 matrix.

7.3. Rendering

Although using textures as function lookup tables is the key enabler of many advanced shading techniques today, such as BRDF factorizations, normalization maps, the nonlinear function maps of Kautz et al. (Kautz and Seidel 2000), and my exponent table, this is probably not the most viable approach in the long term. This is because memory bandwidth has been growing much more slowly than on-chip computation power, so it makes sense to reserve memory bandwidth for content that cannot be derived by computation. This calls for efficient implementation of per-fragment shading.

Texture sampling within graphics hardware assumes that texel values are linear in the exitant radiance of the pixel, so linear interpolation of texels is acceptable. However, when storing input parameters to nonlinear functions in texture maps, linear interpolation is inappropriate.

One approach to this problem is to allow the fragment shader the flexibility of receiving all sampled texels for this texture lookup unfiltered, together with their filter weights, and processing them as desired in the fragment shader. For minification filtering, MIP levels may be created with parameters that will yield shader results that approximate shading followed by filtering.

Another item of future rendering work would be to render two lights per pair of passes within the current rendering regime. I believe this may be possible. The first pass would compute *x'* for each light, and store *x'* and *n'* for each in the frame buffer. The second pass would employ dependent texture reads for each light.

7.4. Painting SBRDFs

I see SBRDFs, whether using the Lafortune representation or some other BRDF representation, as a useful method of representing fairly general surface appearance for computer graphics. As such, I believe tools need to be developed to allow artists to operate on SBRDFs. Beyond the simple SBRDF editing operations of Section 4.6, I have in progress an SBRDF painting application that natively handles BRDF pixels.

One challenge in such a program is how to define the current paint brush color because BRDFs have several parameters. I propose three methods: 1) an eye dropper tool to select an existing BRDF from an SBRDF pixel; 2) an interpolator tool to blend between two chosen BRDFs; and 3) a dialogue with widgets to choose the various parameters, including standard color choosing dialogues for defining ρ_d and ρ_s. Interpolation of BRDFs, needed in several ways by the paint program, will be performed by sampling the BRDFs over their domain, interpolating the samples, and fitting a new BRDF to them, as in Section 5.1.

8. BIBLIOGRAPHY

Adelson, E. H. and J. R. Bergen (1991). The Plenoptic Function and the Elements of Early Vision. Computational Models of Visual Processing. M. Landy and J. A. Movshon. Cambridge, MA.

Ashikhmin, M., S. Premoze, et al. (2000). A Microfacet-based BRDF Generator. Proc. of SIGGRAPH '00, New Orleans, LA. 65-74.

Banks, D. C. (1994). Illumination in Diverse Codimensions. Proc. of SIGGRAPH '94, Orlando, FL. 327-334.

Bastos, R., K. Hoff, et al. (1999). Increased Photorealism for Interactive Walkthroughs. Proc. of Symposium on Interactive 3D Graphics. 183-190.

Becker, B. G. and N. L. Max (1993). Smooth Transitions Between Bump Rendering Algorithms. Proc. of SIGGRAPH '93, Anaheim, CA. 183-190.

Blinn, J. F. (1976). "Texture and Reflection in Computer Generated Images." Communications of the ACM **19**(10): 542-546.

Blinn, J. F. (1977). Models of Light Reflection for Computer Synthesized Pictures. Proc. of SIGGRAPH '77. 192-198.

Blinn, J. F. (1978). Models of Light Reflection for Computer Synthesized Pictures. Proc. of SIGGRAPH '77. 286-292.

Bouguet, J. Y. (2000). Camera Calibration Toolbox for Matlab. http://vision.caltech.edu/bouguetj/calib_doc.

Cabral, B., M. Olano, et al. (1999). Reflection Space Image Based Rendering. Proc. of SIGGRAPH '99, Los Angeles, CA. 165-170.

Catmull, E. E. (1974). A Subdivision Algorithm for Computer Display of Curved Surfaces. Department of Computer Science. Salt Lake City, University of Utah.

Chen, W. C., J.-Y. Bouguet, et al. (2002). Light Field Mapping: Efficient Representation and Hardware

Rendering of Surface Light Fields. Proc. of SIGGRAPH '02, San Antonio, TX.

Cohen, J., M. Olano, et al. (1998). Appearance-Preserving Simplification. Proc. of SIGGRAPH '98, Orlando, FL. 115-122.

Cohen, M. F. and J. R. Wallace (1993). Radiosity and Realistic Image Synthesis. Cambridge, MA, Academic Press, Inc.

Cook, R. L. (1984). Shade Trees. Proc. of SIGGRAPH '84. 223-231.

Cook, R. L. and K. E. Torrance (1981). A Reflectance Model for Computer Graphics. Proc. of SIGGRAPH '81. 307-316.

Crow, F. C. (1984). Summed-Area Tables for Texture Mapping. Proc. of SIGGRAPH '84. 207-212.

Dana, K. J., B. v. Ginneken, et al. (1999). "Reflectance and texture of real-world surfaces." ACM Transactions on Graphics **18**(1): 1-34.

Debevec, P., T. Hawkins, et al. (2000). Acquiring the Reflectance Field of a Human Face. Proc. of SIGGRAPH '00. 145-156.

Debevec, P. E. and J. Malik (1997). Recovering High Dynamic Range Radiance Maps from Photographs. Proc. of SIGGRAPH '97, Los Angeles, CA. 369-378.

Foo, S.-C. (1997). A gonioreflectometer for measuring the bidirectional reflectance of material for use in illumination computation. Program of Computer Graphics. Cornell, Cornell University.

Fournier, A. (1995). Separating Reflection Functions for Linear Radiosity. Rendering Techniques '95 (Proc. of Eurographics Workshop on Rendering), Springer. 383-392.

Gershun, A. (1936). "The Light Field." Translated by P. Moon and G. Timoshenko, Journal of Mathematics and Physics **18**: 51-151.

Gortler, S. J., R. Grzeszczuk, et al. (1996). The Lumigraph. Proc. of SIGGRAPH '96, New Orleans, LA. 43-54.

Gösele, M., W. Heidrich, et al. (2000). Building a Photo Studio for Measurement Purposes. Proceedings of Vision, Modeling, and Visualization.

Green, N. (1986). "Environment Mapping and Other Applications of World Projections." Computer Graphics and Applications **6**(11): 21-29.

Greene, N. and P. S. Heckbert (1986). "Creating Raster Omnimax Images from Multiple Perspective Views Using the Elliptical Weighted Average Filter." IEEE Computer Graphics and Applications **6**(6): 21-27.

Hanrahan, P. (1993). Rendering Concepts. Radiosity and Realistic Image Synthesis. M. F. Cohen and J. R. Wallace**:** 13-40.

Hanrahan, P. and P. Haeberli (1990). Direct WYSIWYG Painting and Texturing on 3D Shapes. Proc. of SIGGRAPH '90, Dallas, TX. 215-223.

Hanrahan, P. and W. Krueger (1993). Reflection from Layered Surfaces Due to Subsurface Scattering. Proc. of SIGGRAPH '93, Anaheim, CA. 165-174.

Hanrahan, P. and J. Lawson (1990). A language for shading and lighting calculations. Proc. of SIGGRAPH '90. 289-298.

He, X., K. Torrance, et al. (1991). A Comprehensive Physical Model for Light Reflection. Proc. of SIGGRAPH '91, Chicago, IL. 175-186.

Heckbert, P. S. (1986). "Survey of Texture Mapping." IEEE Computer Graphics and Applications **6**(11): 56-67.

Heidrich, W. and H.-P. Seidel (1999). Realistic, Hardware-accelerated Shading and Lighting. Proc. of SIGGRAPH '99, Los Angeles, CA. 171-178.

Jensen, H. W., S. R. Marschner, et al. (2001). A Practical Model for Subsurface Light Transport. Proc. of SIGGRAPH '01, Los Angeles, CA. 511-518.

Kajiya, J. (1986). The Rendering Equation. Proc. of SIGGRAPH '86. 143-150.

Karner, K. F., H. Mayer, et al. (1996). "An Image based Measurement System for Anisotropic Reflection." Computer Graphics Forum **15**(3): 119-128.

Kautz, J. and M. D. McCool (1999). Interactive Rendering with Arbitrary BRDFs using Separable Approximations. Rendering Techniques '99 (Proc. of Eurographics Workshop on Rendering), Granada, Spain. 247-260.

Kautz, J. and M. D. McCool (2000). Approximation of Glossy Reflection with Prefiltered Environment Maps. Graphics Interface '00. 119-126.

Kautz, J. and H.-P. Seidel (2000). Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs. Proc. of Eurographics/SIGGRAPH Workshop on Graphics Hardware. 51-58.

Kautz, J., P.-P. Vázquez, et al. (2000). A Unified Approach to Prefiltered Environment Maps. Rendering Techniques '00 (Proc. of Eurographics Workshop on Rendering), Springer. 185-196.

Kilgard, M. J. (1999). NVIDIA OpenGL Cube Map Texturing. http://www.nvidia.com.

Koenderink, J. J., A. J. v. Doorn, et al. (1996). Bidirectional Reflection Distribution Function Expressed in Terms of Surface Scattering Modes. ECCV. 28-39.

Lafortune, E. P. F., S.-C. Foo, et al. (1997). Non-Linear Approximation of Reflectance Functions. Proc. of SIGGRAPH '97. 117-126.

Lalonde, P. and A. Fournier (1997). "A Wavelet Representation of Reflectance Functions." IEEE Transactions on Visualization and Computer Graphics **3**(4): 329-336.

Lensch, H., J. Kautz, et al. (2001). Image-Based Reconstruction of Spatially Varying Materials. Rendering Techniques '01 (Proc. of Eurographics Workshop on Rendering), London, England.

Levoy, M. and P. Hanrahan (1996). Light Field Rendering. Proc. of SIGGRAPH '96, New Orleans, LA. 31-42.

Lewis, R. R. (1993). Making Shaders More Physically Plausible. Rendering Techniques '93 (Proc. of Eurographics Workshop on Rendering), Paris, France. 47-62.

Liu, X., H.-Y. Shum, et al. (2001). Synthesizing Bidirectional Texture Functions for Real-World Surfaces. Proc. of SIGGRAPH '01, Los Angeles, CA. 97-106.

Lu, R., J. J. Koenderink, et al. (1998). "Optical properties (bidirectional reflection distribution functions) of velvet." Applied Optics **37**(25): 5974-5984.

Malzbender, T., D. Gelb, et al. (2001). Polynomial Texture Maps. Proc. of SIGGRAPH '01, Los Angeles, CA. 519-528.

Mark, W. R. and K. Proudfoot (2001). Compiling to a VLIW Fragment Pipeline. Proc. of Eurographics/SIGGRAPH Workshop on Graphics Hardware, Los Angeles, CA. 47-56.

Marschner, S. R. (1998). Inverse Rendering for Computer Graphics. Program of Computer Graphics. Ithaca, New York, Cornell University**:** 147.

Marschner, S. R., S. H. Westin, et al. (1999). Image-based BRDF Measurement Including Human Skin. Rendering Techniques '99 (Proc. of Eurographics Workshop on Rendering), Granada, Spain.

McCool, M., J. Ang, et al. (2001). Homomorphic Factorization of BRDFs for High-Performance Rendering. Proc. of SIGGRAPH '01, Los Angeles, CA. 171-178.

Miller, G. and R. Hoffman (1984). Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments. SIGGRAPH '84 Course Notes - Advanced Computer Graphics Animation.

Neider, J., T. Davis, et al. (1993). OpenGL Programming Guide, Addison Wesley.

Nicodemus, F. E. (1970). "Reflectance Nomenclature and Directional Reflectance and Emissivity." Applied Optics **9**: 1474-1475.

NVIDIA (2002). NVIDIA GeForce 4. http://www.nvidia.com/view.asp?PAGE=geforce4ti.

Olano, M. and A. Lastra (1998). A Shading Language on Graphics Hardware: The PixelFlow Shading System. Proc. of SIGGRAPH '98, Orlando, FL. 159-168.

Palmer, J. (1999). Radiometry and Photometry FAQ, Optical Sciences Center. http://www.optics.arizona.edu/Palmer/rpfaq/rpfaq.htm.

Parker, S., P. Shirley, et al. (1998). Interactive Ray Tracing for Isosurface Rendering. Proc. of IEEE Visualization '98, Research Triangle Park, NC, IEEE Computer Society. 233-238.

Peercy, M. S., J. Airey, et al. (1997). Efficient Bump Mapping Hardware. Proc. of SIGGRAPH '97, Los Angeles, CA. 303-306.

Peercy, M. S., M. Olano, et al. (2000). Interactive multi-pass programmable shading. Proc. of SIGGRAPH '00, New Orleans, LA. 425-432.

Phong, B. T. (1975). "Illumination for Computer Generated Pictures." Communications of the ACM **18**: 311-317.

Porter, T. and T. Duff (1984). Compositing digital images. Proc. of SIGGRAPH '84, Minneapolis, MN. 253-259.

Poulin, P. and A. Fournier (1990). A Model for Anisotropic Reflection. Proc. of SIGGRAPH '90, Dallas, TX. 273-282.

Poynton, C. (1999). Frequently Asked Questions About Color. www.inforamp.net/~poynton.

Presse, W. H., B. P. Flannery, et al. (1988). Numerical Recipies in C, Press Syndicate of the University of Cambridge.

Proudfoot, K., W. R. Mark, et al. (2001). A Real-Time Procedural Shading System for Programmable Graphics Hardware. Proc. of SIGGRAPH '01, Los Angeles, CA. 159-170.

Pulli, K., M. Cohen, et al. (1997). View-based Rendering: Visualizing Real Objects from Scanned Range and Color Data. Rendering Techniques '97 (Proc. of Eurographics Workshop on Rendering), St. Etienne, France.

Rhoades, J., G. Turk, et al. (1992). Real-time Procedural Textures. Proc. of Symposium on Interactive 3D Graphics. 95-100.

Rushmeier, H., F. Bernardini, et al. (1998). Acquiring Input for Rendering at Appropriate Levels of Detail: Dgitizing a Pieta. Rendering Techniques '98 (Proc. of Eurographics Workshop on Rendering).

Sato, Y., M. D. Wheeler, et al. (1997). Object Shape and Reflectance Modeling From Observation. Proc. of SIGGRAPH '97, Los Angeles, FL. 379-387.

Schröder, P. and W. Sweldens (1995). Spherical Wavelets: Efficiently Representing Functions on the Sphere. Proc. of SIGGRAPH '95. 161-172.

Segal, M., C. Korobkin, et al. (1992). Fast Shadows and Lighting Effects Using Texture Mapping. Proc. of SIGGRAPH '92. 249-252.

Shirley, P. (1991). Physically Based Lighting Calculations for Computer Graphics. Department of Computer Science. Urbana, Illinois, University of Illinois at Urbana-Champaign.

Shirley, P. (2000). Realistic Ray Tracing, A. K. Peters.

Shirley, P., H. Hu, et al. (1997). A Practitioners' Assessment of Light Reflection Models. Pacific Graphics.

Smith, A. R. (2001). "Digital Paint Systems: An Anecdotal and Historical Overview." Annals of the History of Computing **23**(2): 4-30.

Stam, J. (1999). Diffraction Shaders. Proc. of SIGGRAPH '99, Los Angeles, CA. 101-109.

Taylor, P. (2001). Per-Pixel Lighting, Microsoft Developer Network. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndrive/html/directx11192001.asp.

Torrance, K. and E. M. Sparrow (1967). "Theory for off-specular reflection from roughened surfaces." Journal of Optical Society of America **57**(9).

Verhoef, W. (1984). "Light scattering by Leaf Layers with Application to canopy Reflectance Modeling SAIL Model." Remote Sensing of Environment **16**: 125-141.

Voorhies, D. and J. Foran (1994). Reflection Vector Shading Hardware. Proc. of SIGGRAPH '94. 163-166.

Ward, G. (1992). Measuring and Modeling Anisotropic Reflection. Proc. of SIGGRAPH '92, Chicago, IL. 265-272.

Ward, G. J. (1994). The RADIANCE Lighting Simulation and Rendering System. Proc. of SIGGRAPH '94, Orlando, FL. 459-472.

Westin, S. H., J. R. Arvo, et al. (1992). Predicting Reflectance Functions from Complex Surfaces. Proc. of SIGGRAPH '92, Chicago, IL. 255-263.

Whitted, J. T. (1980). "An Improved Illumination Model for Shaded Display." Communications of the ACM **23**(6): 343-349.

Williams, L. (1983). Pyramidal Parametrics. Proc. of SIGGRAPH '83. 1-11.

Wolberg, G. (1990). Digital Image Warping, IEEE Computer Society Press.

Wood, D., D. Azuma, et al. (2000). Surface Light Fields for 3D Photography. Proc. of SIGGRAPH '00, New Orleans, LA. 287-296.

Yu, Y., P. Debevec, et al. (1999). Inverse Global Illumination: Recovering Reflectance Models of Real Scenes from Photographs. Proc. of SIGGRAPH '99, Los Angeles, CA. 215-224.

Zhang, Z. (2000). "A flexible new technique for camera calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(11): 1330-1334.