A SMOOTH, EFFICIENT REPRESENTATION OF REFLECTANCE

By

MASAKI KAMEYA

A dissertation submitted in partial fulfillment of
the requirement for the degree of

Doctor of Philosophy

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

AUGUST 2002

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of MASAKI KAMEYA find it satisfactory and recommend that it be accepted.

_____

Chair

_____

_____

A SMOOTH, EFFICIENT REPRESENTATION OF REFLECTANCE

Abstract

by Masaki Kameya, Ph.D.
Washington State University
August 2002

Chair : Robert R. Lewis

Reflectance plays an important role in computer graphics. It describes the appearance of an object with two directional parameters. Reflectance is critical, because it determines the appearance of the object to be synthesized. Reflectance can be determined either by an analytical model, or by evaluating a fit to a measured reflectance data set. In general, analytical models are complex and computationally expensive to evaluate and it is difficult to control the parameters of the model to obtain a desired appearance. A popular method of fitting data is by using a basis function expansion. However, this method requires many basis functions to represent the strongly-peaked data and the result is computationally expensive.

We propose a method to overcome this problem by using a modified N-dimensional multilevel B-spline approximation. Our method fits various reflectance data very well. Multilevel architecture makes it possible to control the accuracy of the fit. A higher level fit uses a denser control mesh and fits more accurately. In addition, the resulting fit is very smooth and efficient to evaluate. The time complexity of evaluation is a constant regardless of the fit level. A higher level fit requires more storage than a lower level fit. The storage might be a problem on memory intensive applications. To overcome this, we represent a data set with two fits, a diffuse fit and a specular fit and we can successfully compress the

storage for finer fit without losing major performance from the original method. In addition, by utilizing minimal perfect hashing, we can retrieve the value of each control point efficiently from compressed table.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer graphics is one of the most active research area in computer science these days. Results of such research are found in various areas such as entertainment, medicine, industrial design and so on. One objective of computer graphics research is photo-realistic image generation, which produces images that can be used in place of images of the real world. The possible applications include: simulating an experiment in a critical situation such as an explosion or an experimental surgery operation, making an actor/actress move around a synthetic universe in a film, and so on. To draw realistic images on a computer, we start with a geometric model of the object and render it, or we can capture images of the scene, extract geometric information from them, and then render the extracted object. The latter technique is called image-based rendering. In general, this saves expensive computation and one can render the image very fast.

There are many aspects involved in generating such images: how to model an object, how to determine its appearance, how to make it move, how to represent the material it is made from. In this thesis, we focus on how to control the appearance of objects.

The color of a material is the result of an interaction of light with the surface of

the material. Some light is reflected, some is absorbed, some is emitted, and some is transmitted through the surface of an object. Various materials have very different characteristics of light reflection, and the way a material reflects light is described as its reflectance.

Reflectance plays an important role in computer graphics, since it determines the color of materials and it affects the appearance of objects. In principle, we could analytically derive the reflectance from physics. But in general, it is difficult to represent a desired appearance in analytical form and it requires expensive calculation to evaluate the model. Another way to obtain the reflectance is directly measuring it from a sample of real material. When we use measured reflectance, we have to find a way to interpolate the data to render images, as the data set is a collection of discrete samples. The more samples we have, the more accurately we can synthesize images. However, it is time consuming to acquire such data and it is practically impossible to store all of sample data for all possible directions. Thus researchers are trying to model a reflectance by a simple representation using a sparse representation.

In this thesis, we present a representation of reflectance: specifically, measured bidirectional reflectance distribution function (BRDF) data. Our method finds a fit of the given reflectance data efficiently. Unlike the complex representations presented before, our representation can be computed efficiently and the fit can represent the measured data set accurately.

In Chapter 2, we introduce the basic ideas behind this thesis. In Chapter 3, we review previous work on representing reflectance. We present our method in Chapter 4, and show some results in Chapter 5. In Chapter 6, we revise our method to make it more storage-efficient and show the results of revised method in Chapter 7. Chapter 8 concludes this thesis.

Chapter 2

Background

2.1 Illumination

Illumination is a major area of study in computer graphics. It is a study of lighting and shading. How well we can render surfaces of objects depends on how well we can model the reflection of light on the surface. While the physics of reflection is well-understood, it is generally too complicated to use in graphics rendering. Hence, we tend to simplify physics, or find some technique to imitate them so as to make the image look good.

In this chapter, we explain the nomenclature of illumination, fundamental ideas, and simple illumination models.

2.1.1 Radiometry Nomenclature

Radiometry is the science of measuring radiant energy transfer. *Flux* is a rate at which light energy is emitted and is measured in watts(W). *Solid angle* is defined as the area on a unit sphere subtended by an object whose projection rays start at the center of the sphere. A steradian (sr) is the customary unit of solid angle. Spherical coordinates are often used in

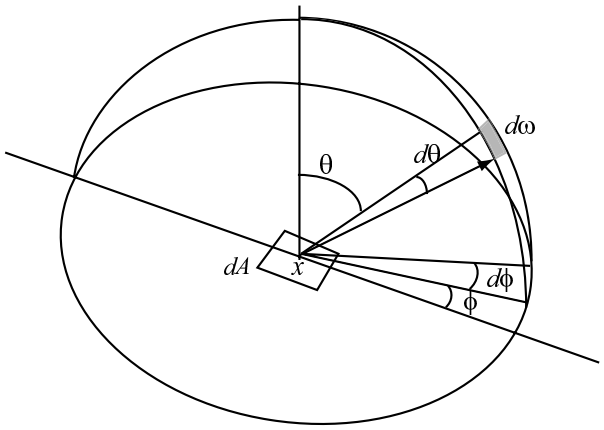


Figure 2.1: Spherical Coordinates and Solid Angle for Small Area.

illumination problems. A direction is expressed by two angles in spherical coordinates: a polar angle θ and an azimuthal angle ϕ (Figure 2.1). A differential solid angle around the direction (θ, ϕ) is expressed by

$$d\omega = \sin\theta d\theta d\phi. \tag{2.1}$$

Radiance is defined as the amount of energy traveling at some point in a specified direction, per unit time, per unit area perpendicular to the direction of travel, per unit solid angle of source. It is measured in W/(sr·m^2). We denote radiance by L. The energy radiated in the small solid angle $d\omega$, from differential area dA on the hemisphere during time interval dt is expressed as

$$L(x, \theta, \phi) dA \cos\theta d\omega dt. \tag{2.2}$$

Irradiance is the incident flux per unit surface area and is measured in W/m^2.

An important property of the radiance is Helmholtz reciprocity. For any two mutually visible points x and y in space, the radiance leaving point x in the direction of

point y is the same as the radiance reflecting on point y from the direction of point x:

$$L(x, \mathbf{S}) = L(y, -\mathbf{S}), \tag{2.3}$$

where $\mathbf{S}$ is the direction from point x to point y.

2.1.2 Simple Illumination Models

An illumination model can be expressed by an illumination equation in variables associated with a position on the object being shaded. For simplicity, we mainly discuss an illumination in terms of monochromatic light. When we consider a colored object, we have to express the illumination equation for the light of each color.

The simplest illumination equation is :

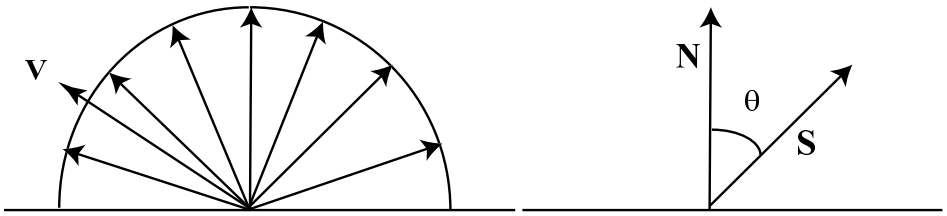
$$L = L_e \tag{2.4}$$

where L is the resulting radiance of the light and L_e is an intrinsic emissivity. This model does not depend on an external light source. It models a luminous object.

An object can be illuminated by light coming from any direction. Some might come directly from light sources, while other light might be indirectly reflected off one or more surfaces before it reaches the object. As indirect light is difficult to represent, it is common to represent it as a constant ambient term. If we assume that an artificial ambient light impinges uniformly on all surfaces from all directions, the illumination equation becomes

$$L = k_a L_a\,, \tag{2.5}$$

where L_a is an radiance of the ad-hoc ambient light. The amount of ambient light reflected



(a)The light reflects equally in all directions.

(b)The amount of light incident upon a small area of the surface is proportional to the cosine of the angle θ.

Figure 2.2: Diffuse Reflection Model

from an object's surface is determined by k_a, which is a coefficient associated with the object's material.

If there is an direct light source in the scene, such as a light bulb, then the brightness of the object's surface varies depending on its orientation. *Diffuse reflection* is a simple approach to model this effect and it is also known as Lambertian reflection[22]. Material with a diffuse reflection property (which is called a diffuse material) reflects the light equally in all directions. Therefore a viewer can observe uniform surface brightness independently of the viewer's position (Figure 2.2 (a)). The brightness of the surface only depends on the angle between the surface normal and the direction to the light source. The amount of light falling onto the surface is proportional to cosine of the angle between the surface normal $\mathbf{N}$ and the direction of light $\mathbf{S}$ (Figure 2.2(b)). The diffuse illumination equation is written as:

$$L = k_d L_d \cos\theta, \tag{2.6}$$

where L_d is the radiance of the incident light and k_d is the coefficient of diffuse reflection which is associated with the material. If the vectors $\mathbf{N}$ and $\mathbf{S}$ are normalized, then (2.6) can

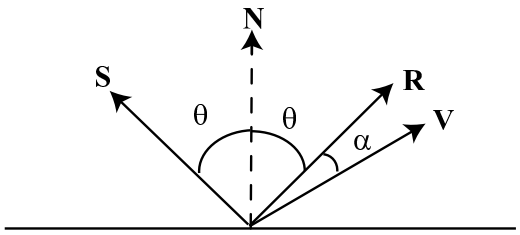


Figure 2.3: Specular Reflection Model. $\mathbf{S}$ is the direction of the light, $\mathbf{R}$ is the mirror direction. $\mathbf{V}$ is the direction of the viewer, $\mathbf{N}$ represents the surface normal.

be rewritten by using the dot product:

$$L = k_d L_d(\mathbf{N} \cdot \mathbf{S}) \tag{2.7}$$

We observe a highlight on a surface when we look at shiny material such as polished plastic or metal. The highlight is caused by a *specular reflection* and the color of the highlight is almost always the color of the light (usually white), independent of the material color. If the viewer changes position, the brightness of the highlight varies. This is because a specular surface reflects the light differently in different directions. An extra case of this phenomenon is a mirror, where the light is only reflected in the mirror direction $\mathbf{R}$ (Figure 2.3).

Phong proposed an illumination model [31] to simulate the specular highlight with a simple form. It assumes that maximum specular reflection occurs when the angle α between the viewing direction and mirror direction is zero and falls off sharply as α increases. Phong modeled it by using $\cos^n \theta$, where n is the material dependent specular exponent:

$$L = k_p L_p \cos^n \theta, \tag{2.8}$$

where L_p is the incident radiance and k_p is the reflectance associated with the material. The

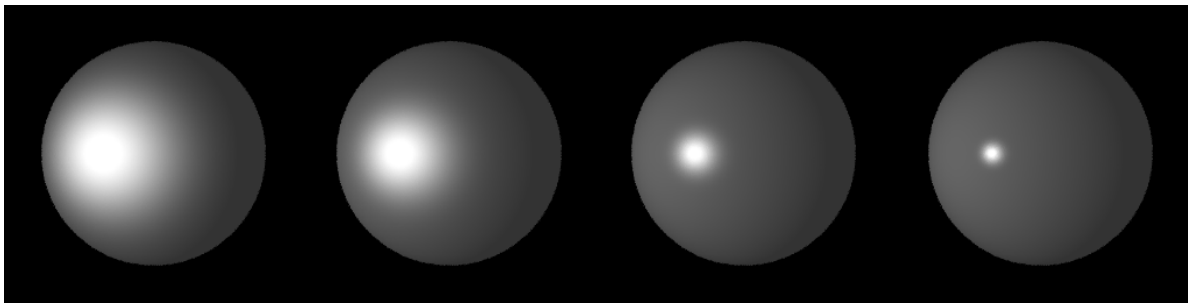


Figure 2.4: A Sphere Shaded using Phong's Illumination Model with Different Exponent n ($n = 5, 10, 50, 200$ from left to light).

value of n varies from 1 to several hundred. Small n such as 1 gives a broad, slow falloff, while a larger value gives a narrow, rapid falloff and a small highlighted area. Figure 2.4 shows a sphere synthesized using Phong's illumination model for various specular exponents. This model has been used for quite long time in computer graphics because it is simple, it is computationally inexpensive and it produces quite a nice appearance.

We can synthesize images with illumination models described above, but we still explore a more sophisticated model, which is modeled by incorporating physics that express the behavior of the reflection of lights. In next subsection, we will model illumination based on the study of radiometry.

2.1.3 Reflectance

Various materials have very different appearances in nature. This is because each material has a different property of light reflection. This property can be described by the concept of *reflectance*. Reflectance is a function of various parameters such as the surface material, the wavelength of the light, the incident direction of the light and viewer's direction. Typically reflectance in computer graphics is represented by a *bidirectional reflectance distribution function*, BRDF.

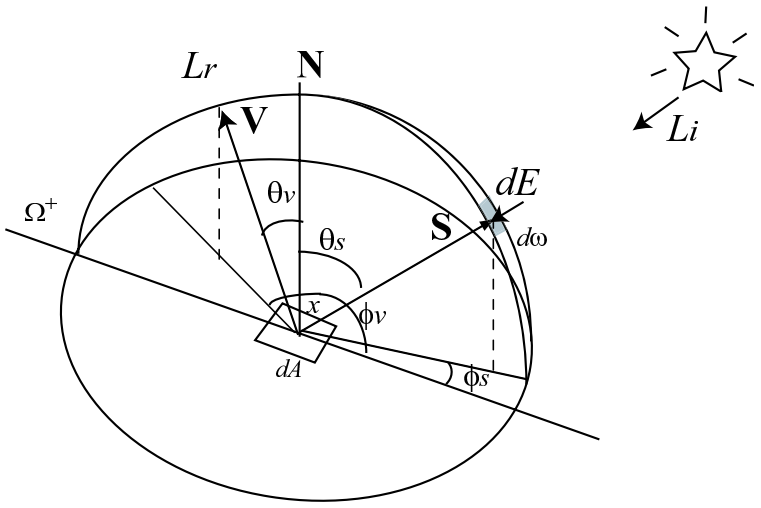


Figure 2.5: Geometry of BRDF over the Hemisphere.

The BRDF is defined as a ratio of the reflected radiance dL_r in a given direction V to the irradiance dE in an incident direction $\mathbf{S}$. It is defined on the upper hemisphere Ω^+ (Figure 2.5). Traditionally, the pair of a polar and azimuthal angle (θ_s, ϕ_s) and (θ_v, ϕ_v) parameterize the direction vector $\mathbf{S}$ and $\mathbf{V}$ respectively. The surface normal at the small area dA is denoted by $\mathbf{N}$. $d\omega$ is a differential solid angle, and dE is the irradiance coming through $d\omega$. We denote the BRDF as $f_r(\mathbf{S}, \mathbf{V})$ and it is defined to be

$$dL_r = f_r(\mathbf{S}, \mathbf{V})dE. \tag{2.9}$$

The irradiance dE for the source direction $\mathbf{S}$ subtending a solid angle $d\omega$ can be represented as:

$$dE(\mathbf{S}) = L_i(\mathbf{S})cos\theta d\omega, \tag{2.10}$$

where $L_i(\mathbf{S})$ is the incident radiance in the direction $\mathbf{S}$. Then the reflected radiance L_r in the direction $\mathbf{V}$ is obtained by integrating the differential radiance dL_r over all directions

on the Ω^+. The total reflected radiance $L_r(\mathbf{V})$ is thus :

$$L_r(\mathbf{V}) = \int_{\Omega_{(+)}} f_r(\mathbf{S}, \mathbf{V}) L_i(\mathbf{S}) \cos\theta d\omega. \tag{2.11}$$

The BRDF plays an important role in computer graphics because it describes what will be seen by the viewer for any incident light distribution. We need to know how light is reflected by the surface for arbitrary viewing and incident light directions to synthesize an arbitrary image.

The simple illumination models described in section 2.1.2 can be expressed as BRDFs. An ideal diffuse reflects the light equally in all direction. Thus, the ideal diffuse BRDF will be constant and does not depend upon an incident nor a reflected direction. Therefore,

$$f_r(\mathbf{S}, \mathbf{V}) = \rho. \tag{2.12}$$

where ρ is some constant associated with a material. An ideal specular reflection reflects the light only to the mirror direction as described in Section 2.1.2. Hence specular BRDF can be expressed by a Dirac delta function,

$$f_r(\mathbf{S}, \mathbf{V}) = \rho \frac{\delta(\theta_s - \theta_v)\delta(\phi_s - \pi - \phi_v)}{\cos\theta_s \sin\theta_s} \tag{2.13}$$

where θ_s and ϕ_s are polar and azimuthal angles of the incident direction and θ_v and ϕ_v are the polar and azimuthal angles of the viewing direction, δ is a normalized Dirac delta function which returns 0 for a non-zero argument.

It is convenient to consider that BRDF is composed of diffuse and specular com-

ponents. Hence, the BRDF f_r can be modeled as :

$$f_r = k_d f_d + k_s f_s, \tag{2.14}$$

where f_d and f_s are diffuse and specular bidirectional reflectances respectively and k_d and k_s are the diffuse and specular reflection coefficients respectively. The ratio of k_d and k_s determines how shiny or dull the surface is.

Variations of this model are possible. We can decompose the specular reflectance into several components and vary the coefficients of each components to control the specularity of the material:

$$f_r = k_d f_d + \sum_i k_{s,i} f_{s,i}. \tag{2.15}$$

In Chapter 3, we will review some popular representations of BRDFs.

2.2 Asymptotic Notation

When we talk about the size of storage or running time of an algorithm, we usually use asymptotic notation to represent them. We could measure the exact size of storage, or the time of computation, but the extra precision is not usually worth the effort of computing them. In many cases, the multiplicative constants to the size of input dominate the resulting size or time. Therefore, we usually use asymptotic notation to descrivbe the efficiency of the algorithm.

2.2.1 Θ-notation

For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions

$$\begin{aligned}\Theta(g(n)) \;=\; &\{f(n) : \text{there exist positive constanst } c_1, c_2, \text{ and } n_0 \text{ such that} \\ &0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0\}.\end{aligned}$$

Suppose a function $f(n)$ represents the running time of some algorithm for input size n. When we say $f(n)$ is $\Theta(n^2)$, then there exist some constants that satisfy:

$$c_1 n^2 \le f(n) \le c_2 n^2. \tag{2.16}$$

The running time $f(n)$ is equal to n^2 to within a constant factor. Thus, $\Theta(n^2)$ gives an *asymptotically tight bound* for $f(n)$.

2.2.2 O-notation

When we have only an *asymptotic upper bound*, we use O-notation. For a given function $g(n)$, we denote by $O(g(n))$ the set of functions

$$\begin{aligned}O(g(n)) \;=\; &\{f(n) : \text{there exist positive constant } c, \text{ and } n_0 \text{ such that} \\ &0 \le f(n) \le c g(n) \text{ for all } n \ge n_0\}.\end{aligned}$$

If we say $f(n)$ is $O(n^2)$, then there is a constant such that

$$f(n) \le c n^2. \tag{2.17}$$

The running time $f(n)$ is less than a constant factor of n^2.

2.3 B-Spline Approximation

Sampled data is by definition incomplete. If the data is reflectance data, we have it for some directions, but our rendering algorithm may require reflectance in any direction, so we must interpolate or approximate from the data we have. One method to do this is with parametric curves.

We apply the multi-dimensional B-spline approximation presented by Lee et al. for this purpose. In this chapter we explain the basic idea of B-Spline approximation. Then we will explain Lee's approximation algorithm. More detailed discussion on B-splines can be found in [1, 12].

2.3.1 B-Splines

A B-spline curve is one of the most popular parametric representations. A spline was originally a flexible strip of metal used by a boat builder to draw a curve for designing a vessel. A curve was made up of a set of such strips that were pulled into shape by attached metal weights. We can mathematically describe such a curve with a piecewise cubic polynomial function whose first and second derivatives are continuous across the curve sections. B-splines consist of curve sections, and the polynomial coefficients of each section depend on a few control points.

An uniform cubic B-spline has some properties that are advantages over other splines. These are local control, convex hull and C^2 continuity. By local control we mean that altering a control point causes only a part of the curve to change. So, if we change the value of control point, or delete or add a control point, we do not need to change the rest of the curve where we do not want to change.

The control points form the boundary, called *convex hull*, by connecting each

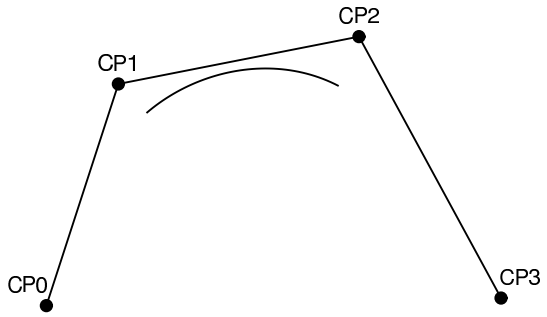


Figure 2.6: Cubic B-spline Curve. The curve is determined by 4 control points (CP0, $\cdots$, CP3) and surrounded by a convex hull formed by the control points.

control points, and the boundary encloses the splines which are defined by those control points (Figure 2.6). The second derivative of the curve is continuous at any point of the cubic B-spline curve, so the curve is fairly smooth.

Cubic B-splines approximate a series of $m + 1$ control points $p_0, p_1, \cdots, p_m$, $m \geq 3$, with a curve consisting of $m - 2$ cubic polynomial curve segments Q_0, Q_1, $\cdots$, Q_{m-3}. The i th segment of a B-spline is represented as Q_i and each segment possesses four control points that we refer to as p_i, p_{i+1}, p_{i+2} and p_{i+3}. The $i + 1$th segment Q_{i+1} possesses control points $p_{i+1}, p_{i+2}, p_{i+3}$,and p_{i+4}. Therefore, these two segments share the control points p_{i+1}, p_{i+2}, and p_{i+3}.

Although a cubic spline might be defined on its own parameter domain $0 \leq\sim t < 1$, we can adjust the parameter so that the parameter domains for the various curve segments are sequential. Thus, segment Q_i can be defined on the interval $t_i \leq t < t_{i+1}$ for $0 \leq i < m - 2$. For each $i \geq 1$, there is a common point or *knot* between Q_{i-1} and Q_i at the parameter value t_i. The parameter value at such a point is called a *knot value*. The term uniform means that the knots are spaced at equal intervals of the parameter t (usually 1). The knots are often expressed in the form of a vector and the vector is called the *knot*

vector. A single cubic B-spline curve is represented as:

$$Q_i(t) = \sum_{k=0}^{3} B_k(t) p_{i+k} \tag{2.18}$$

where B_k is the B-spline basis function. The basis functions B_k are polynomials of degree 3.

$$\begin{aligned} B_0(t) &= \frac{1}{6}(1-t)^3 \\ B_1(t) &= \frac{1}{6}(3t^3 - 6t^2 + 4) \\ B_2(t) &= \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1) \\ B_3(t) &= \frac{1}{6}t^3 \end{aligned} \tag{2.19}$$

Basis functions determine how much each control point contributes to the spline at the given parameter t. Therefore, basis functions are also called *blending functions*. Basis functions for B-splines curves can be computed recursively.

2.3.2 Approximation by Uniform Cubic B-Splines

A B-spline can be used to fit data. Suppose we have a set of data at points $v_0, v_1, \cdots, v_n$ for some n, and want to represent with a smooth cubic B-spline curve $q(t)$. Solving this problem is finding the control points $\{p_i\}$ of $q(t)$ so that $q(t)$ approximates data points $\{v_i\}$. We can determine the value of these control points in such a way that minimizing the sum of square error between data point and the fit; that is, minimize the sum of error e:

$$e = \sum_i (q(t_i) - v_i)^2. \tag{2.20}$$

This is known as the least square error approximation.

Although we can fit the data set by a B-spline, the resulting fit might not be able to represent the data set well in some case. To refine the fit, we can subdivide a B-spline and recompute the fit for each segment. In the next subsection, we describe the notion of subdivision.

2.3.3 Subdivision

The flexibility of controlling a B-spline curve can be increased by either raising the order of the B-spline basis or inserting additional knots. After raising the degree or inserting the knot, the curve will have more control points or the curve is subdivided into smaller segment and it has narrower locality and that makes it easy to manipulate the curve.

If we want to keep evaluation time of the basis functions constant, raising the degree of basis function is not desirable. In this section, we only consider the knot insertion.

Consider the original curve $P(t)$ defined by

$$P(t) = \sum_{i=1}^{n+1} p_i B_{i,k}(t) \tag{2.21}$$

with knot vector $[t_1, t_2, \cdots, t_{n+k+1}]$. After knot insertion, the new curve $R(t)$ is defined by

$$R(t) = \sum_{j=1}^{m+1} r_j C_{j,k}(t) \tag{2.22}$$

with the new knot vector $[l_1, l_2, \cdots, l_{m+k+1}]$, where $m > n$. The objective is to determine the new control points r_j such that $P(t) = R(t)$. By the Oslo algorithm [33] the new r_j's are computed by:

$$r_j = \sum_{i=1}^{n+1} \alpha_{i,j}^k p_i \quad 1 \leq i \leq n, 1 \leq j \leq m \tag{2.23}$$

where the $\alpha_{i,j}^k$'s are given by the recursion relation

$$\alpha_{i,j}^1 = \begin{cases} 1 & t_i \leq l_i < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

$$\alpha_{i,j}^k = \frac{l_{j+k-1} - t_i}{t_{i+k-1} - t_i}\alpha_{i,j}^{k-1} + \frac{t_{j+k} - l_{j+k-1}}{t_{i+k} - t_{i+1}}\alpha_{i+1,j}^{k-1} \tag{2.25}$$

The result is that fewer data points are affected by each control point and it is then easier to fit the data.

So far, we have only described approximation for 2D data by B-spline curves. But it can be used for 3D data with B-spline surfaces. If we allow the control point $\{p_k\}$ to vary in 3D along the B-spline curve with parameter s, then we have

$$Q(t, s) = \sum_{k=0}^{3} B_k(t) p_k(s). \tag{2.26}$$

If s is fixed, then $Q(t, s)$ is simply a curve as before. By changing the parameter s, then we can draw different curves. If the control points $p_k(s)$ vary along the cubic B-spline curve, we have

$$Q(t, s) = \sum_{k=0}^{3} \sum_{l=0}^{3} B_k(t) B_l(s) p_{k,l}. \tag{2.27}$$

(2.27) represents a bi-cubic B-spline surface. This representation can be extended to higher dimensional data.

2.3.4 N-dimensional Multilevel B-Spline Approximation for Scattered Data

Lee et al. presented a B-spline based approximation for image morphing [24] and then applied it to a scattered data set in [25] (hereafter, denoted as LWS). The algorithm presented in LWS is a fast approximation algorithm and it makes use of a coarse-to-fine hierarchy of control lattices to generate a sequence of cubic B-spline functions whose sum approaches the desired fit. Even if the data set is not uniformly sampled, it can find the fit efficiently. The fit has C^2 continuity and the time complexity of fit finding is linear in terms of the size of the control lattice. The evaluation of the fit is constant-time. The algorithm can be applied to any dimensional data, so we refer it as N-dimensional multilevel B-spline approximation algorithm.

In this section, we describe their basic algorithm and the multilevel refinement algorithm which improves the accuracy of the approximation of the basic algorithm without loosing the smoothness of the fit.

Basic algorithm

Paralleling LWS, we explain their algorithm. Although there are no limitations on the data dimensions, for illustration, we assume the data set to be a height field : a scalar range z and a 2D domain (x, y). Let the domain of the data set be $\Psi = \{(x, y)|0 \leq x < m, 0 \leq y < n\}$ be a rectangular domain in the $xy-$plane (see Figure 2.7). Consider a given set of scattered points $P = (x_c, y_c, z_c)$ in R^3, where (x_c, y_c) is a point in Ψ. For this point set, we will find the fit function f. It approximates data points P as a uniform non-parametric bi-cubic B-spline function which is defined by a control lattice Φ overlaid on and lying slightly outside the domain Ψ.

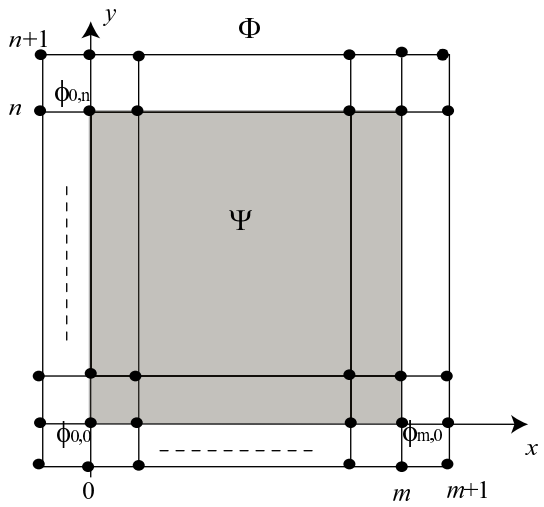


Figure 2.7: Domain Ψ of the Data set (in gray) and Control Lattice Φ.

Let ϕ_{ij} be the control point at (i, j) on the $(m + 3) \times (n + 3)$ lattice Φ, where $-1 \leq i \leq m + 1, -1 \leq j \leq n + 1$. The approximation function f is defined as:

$$f(x, y) = \sum_{k=0}^{3} \sum_{l=0}^{3} \phi_{(k+i)(l+j)} B_k(s) B_l(t), \tag{2.28}$$

where $i = \lfloor x \rfloor - 1, j = \lfloor y \rfloor - 1, s = x - \lfloor x \rfloor, t = y - \lfloor y \rfloor$. B_k and B_l are the cubic B-spline basis functions. The problem of finding f is thus reduced to solving for the control points in Φ that approximate the scattered data points P.

Assume $1 \leq x_c, y_c < 2$, then control points ϕ_{kl} for $k, l = 0, 1, 2, 3$ determine the value of f at (x_c, y_c). Therefore, f must satisfy

$$z_c = \sum_{k=0}^{3} \sum_{l=0}^{3} w_{kl} \phi_{kl}, \tag{2.29}$$

where $w_{kl} = B_k(s) B_l(t)$, $s = x_c - 1$, and $t = y_c - 1$.

There are many values that satisfy (2.29). The basic algorithm (BA) chooses the

one in the least-squared sense that minimizes $\sum_{k=0}^{3}\sum_{l=0}^{3}\phi_{kl}^2$. We can make a geometrical interpretation of this.

Note that (2.29) represents a hyper-plane in a 16-dimensional Euclidean space, with ϕ_{kl} the unknowns and normal components w_{kl}. In vector form,

$$z_c = \mathbf{W}\cdot\mathbf{\Phi}, \tag{2.30}$$

where $\mathbf{W} = [w_{00}, w_{01}, \cdots, w_{32}, w_{33}]$ and $\mathbf{\Phi} = [\phi_{00}, \phi_{01}, \cdots, \phi_{32}, \phi_{33}]$.

If we locate the point $(\phi_{00}, \phi_{01}, \cdots, \phi_{33})$ on the hyper-plane and make the direction of the line $\mathcal{L}$, which passes though the origin and that point, be perpendicular to the hyper-plane (i.e., parallel to the hyper-plane's normal), then we can minimize the distance from the origin to the point $(\phi_{00}, \phi_{01}, \cdots, \phi_{33})$, i.e. minimize the $\sum_{k,l=0,1,2,3}\phi_{kl}^2$.

Since $\mathcal{L}$ is parallel to the hyper-plane's normal and it passes through the origin, we can represent ϕ_{kl} with a scalar parameter u as:

$$\phi_{kl} = uw_{kl}. \tag{2.31}$$

Incorporating this into (2.29), we get

$$u = \frac{z_c}{\sum_{k=0}^{3}\sum_{l=0}^{3}w_{kl}^2}. \tag{2.32}$$

Substituting this into (2.31) we get

$$\phi_{kl} = \frac{w_{kl}z_c}{\sum_{i=0}^{3}\sum_{j=0}^{3}w_{ij}^2}. \tag{2.33}$$

For each data point, (2.33) can be used to determine the set of 4×4 control

points in its neighborhood.

What happens, however, when multiple data points all affect the same control point? Let P_{ij} be the data affecting control point ϕ_{ij} such that

$$P_{ij} = \{(x_c, y_c, z_c) \in P \mid i - 2 \leq x_c < i + 2, j - 2 \leq y_c < j + 2\}.$$

For each point (x_c, y_c, z_c) in P_{ij}, (2.33) gives ϕ_{ij} a different value ϕ_c:

$$\phi_c = \frac{w_c z_c}{\sum_{a=0}^{3} \sum_{b=0}^{3} w_{ab}^2}. \tag{2.34}$$

We choose ϕ_{ij} to minimize the error metric $e(\phi_{ij})$ between the real and expected contributions of ϕ_{ij} to the function f at (x_c, y_c):

$$e(\phi_{ij}) = \sum_c (w_c \phi_{ij} - w_c \phi_c)^2. \tag{2.35}$$

By differentiating (2.35) respect to ϕ_{ij}, setting the result to 0, and solving for ϕ_{ij}, we get a minimum (presumably) of (2.35) at

$$\phi_{ij} = \frac{\sum_c w_c^2 \phi_c}{\sum_c w_c^2}. \tag{2.36}$$

This provides a solution to ϕ_{ij} which minimizes a local least-squared approximation error.

The time and space complexity of this algorithm is $O(p + mn)$, where p is the number of data points and $m \times n$ is the size of the control lattice.

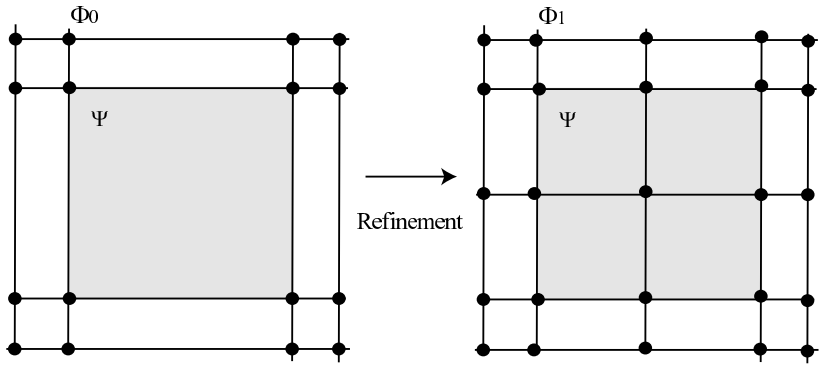


Figure 2.8: Control Mesh Refinement. Double the mesh size within the domain Ψ in all dimensions. (The figure shows 2D case.) Φ_0 is the initial mesh. Φ_1 is refined once from Φ_0. Black dots indicate the position of control points.

Multilevel Refinement

There is a trade off between accuracy and smoothness with BA affected by the coarseness of the grid, which is a given. Because this algorithm uses cubic B-splines, the approximation function f is C^2 continuous and has local support. If we use a coarse control lattice, then the control points are calculated from many data points and f has wide support, so we get a smooth approximation. On the other hand, if we use a finer lattice, then f has smaller local support and a smaller number of data points affecting the calculation of control points. Hence, f achieves greater accuracy.

To circumvent this situation, LWS went on to present the *multilevel* B-spline approximation, which is refereed to as "MBA". This algorithm starts from the coarsest approximation (i.e., $m = 1$), which captures the global shape of f. By setting up a denser lattice on the domain, a finer level of approximation can be obtained. Refinement can be done by doubling the size of the control mesh within the domain Ψ for all dimensions as shown in Figure 2.8.

At the refinement stage, the function approximates the difference between the

approximated data and the original data for the finer lattice. That is, given an initial coarsest fit f_0, it finds the approximation f_1 for

$$\delta^1 z = z_c - f_0(x_c, y_c).$$

Successive finer levels serve to approximate and remove the residual error. The final approximation is defined as the sum of all levels of fits f_k, i.e. $f = \sum_{k=0}^{h} f_k$.

The only user-specified fit parameter is then the maximum level h, and with non-pathological data (e.g., all (x_c, y_c)'s are numerically distinct), the least-squared error of the fit decreases monotonically with increasing h. This makes it possible for an MBA user to specify a given tolerance and have the algorithm find the smallest value of h which satisfies it.

If we simply refine the lattice resolution in a way described above, we need to keep the control points in each level. When we evaluate the fit, we have to evaluate the fit at each level and sum the result of all levels. Thus the evaluation time and storage will be increased at higher levels. LWS also proposed a method to optimize the refinement. With this optimization, we can only store the control points at the highest refinement level. Evaluation is also done with the fit on the highest level. That is, the final fit can be expressed by one B-spline evaluation. We can do this optimization by incorporating the B-spline refinement.

Let $F(\Phi_0)$ be the B-spline function generated by control lattice Φ_0, $|\Phi_0|$ be the size of Φ_0, and f_0 be the fit found with Φ_0. By B-spline refinement, we can derive the control lattice Φ_0' that defines the new B-spline function $F(\Phi_0')$ that satisfies $F(\Phi_0') = f_0$ and $|\Phi_0'| = |\Phi_1|$ where Φ_0 is the lattice at level 0 (the coarsest level) and Φ_1 is the one at level 1 (next finer level). Then, the sum of the functions f_0 and f_1 (the fit at refinement

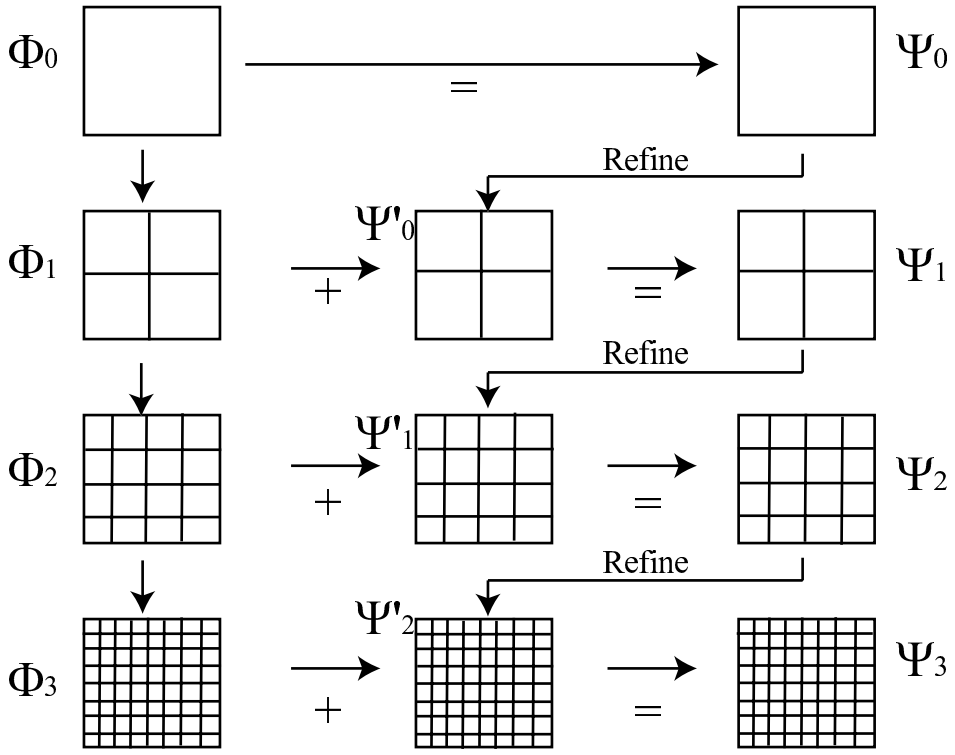


Figure 2.9: MBA Refinement Process.

level 1) can be represented by the control point lattice Ψ_1 which resulted from the addition of each corresponding pair of control points in Φ_0' and Φ_1, i.e. $F(\Psi_1) = g_1 = f_0 + f_1$, where $\Psi_1 = \Phi_0' + \Phi_1$. We can repeat this refinement and addition to represent the fit at level k with one B-spline function. In general, let $g_k = \sum_{i=0}^{k} f_i$ be the partial sum of functions f_i up to level k. Suppose that function g_{k-1} is represented by a control lattice Ψ_{k-1}, where $|\Psi_{k-1}| = |\Phi_{k-1}|$. In the same manner that we refine Φ_k to Φ_k', we can refine Ψ_{k-1} to Ψ_{k-1}'. Then add Ψ_{k-1}' to Φ_k to obtain Ψ_k such that $F(\Psi) = g_k$ and $|\Psi_k| = |\Phi_k|$. This process is depicted in Figure 2.9.

Let $M \times N$ be the size of finest control lattice, p be the number of data points, and h be the maximum level of refinement. Then the time complexity for the fitting part of MBA is $O(hp + MN)$ and its space complexity is $O(p + MN)$. The time complexity of

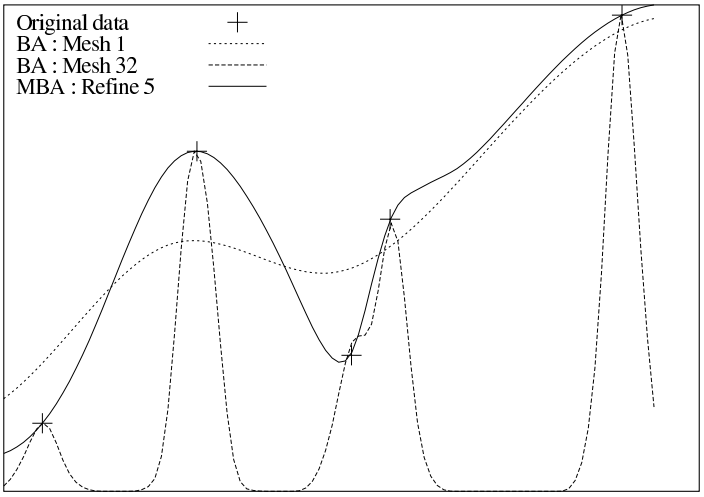


Figure 2.10: The Effect of Refinement. The BA fit with mesh size 32 fits to data, but lacks smoothness. The plot with MBA(refine level 5) shows both accuracy of fit and smoothness.

the fit evaluation is constant , i.e $O(1)$, independent of the level of hierarchy h.

Figure 2.10 shows the effect of MBA refinement. The initial fit for MBA is equivalent to the BA fit with mesh size 1. The figure shows the BA fit with mesh size 32 and the MBA fit with refinement level 5, whose number of control points is equivalent to BA fit with mesh size 32. The BA fit with the finer mesh size fits the data well, but is less smooth. The MBA fit with refinement level 5 also fits well the data but still keeps the smoothness.

We will apply the MBA algorithm to represent reflectance data in Chapter 4.

2.4 Hash Functions

A hash function transforms a key value from a set with many members to a hash value from a set with a fixed number of members, usually fewer than the original one. There are many applications that use hash functions in computer science. In a compiler, for example, the

hash function is used as a part of the searching algorithm for keywords of the language. Instead of comparing the string character by character with all keywords, first the compiler "hashes" the string into an integer hash value and uses this as the index of a "hash table" of keywords. Then it compares the contents of the table with string by character by character to determine if the string is the keyword and not a coincidence. This kind of problem is called the *dictionary problem*. The hash function and hash table give an effective way to solve it. Instead of linearly searching the whole key space, it offers greater efficiency.

In this section, we describe a class of hash functions, called "minimal perfect". In Chapter 6, we will use such a hash function to develop a compact representation of reflectance data.

2.4.1 Basic Hashing

Let U be the universe of keys. The elements can be numbers or strings, and K be a subset of U, consists of n distinct elements, or keys. A *hash function* H is a function that maps the key from the set K into a value from a set M with a fixed number of (usually fewer) members. Let the number of elements in M be m. Given a key k from K, a hash function h computes the index of the storage where the key k is stored. The storage is called *hash table*. There is a case that two distinct keys are hashed to the same value. This is called a *collision*. Since the size of M is typically smaller than the size of K, some keys could be hashed to the same value. Suppose there are two distinct keys k_2 and k_5. Then a collision occurs if $h(k_2) = h(k_5)$ for $k_2 \neq k_5$ (Figure 2.11). There are many methods to deal with a collision.

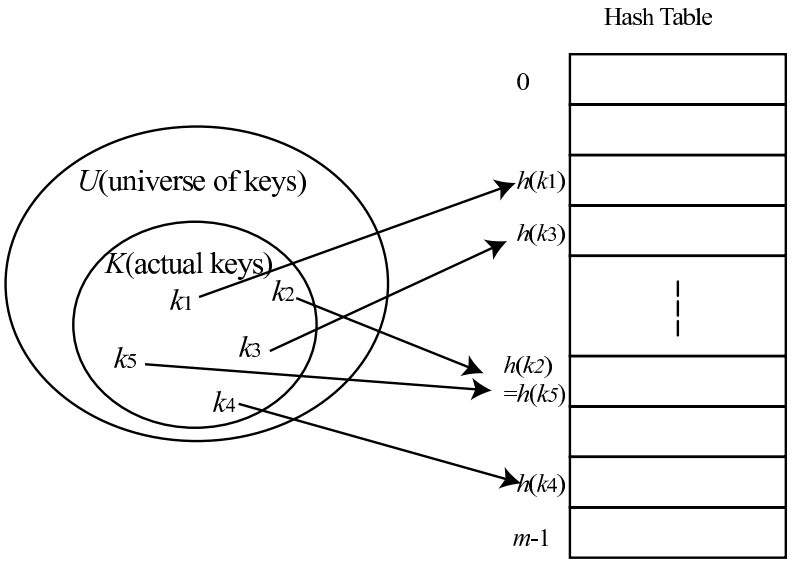


Figure 2.11: Hash Function. Using a Hash function h to map keys from K to hash table index.

2.4.2 Chaining

One method to resolve collisions is called *chaining*: ordering all the elements that hash to the same value in a linked list (Figure 2.12). Slot j contains a pointer to the head of the list of keys that hash to j and their values. If there are no such elements, then slot j is empty (represented as NULL in Figure 2.12).

Given a hash table T with m slots that stores n elements, we define the load factor α for T as n/m. This is the average number of elements stored in a chain. The worst-case searching time for T with chaining is the case where all keys hash to one slot: the length of the list in that slot is n. In this case, the search time will be $\Theta(n)$.

The average performance of hashing depends on how well the hash function distributes the output through the entire space of the hash table index. For a given key, a good hash function is equally likely to hash into any of the m slots. This property is called *uniform hashing*. We can assume that the time for computing the hash function is $O(1)$.

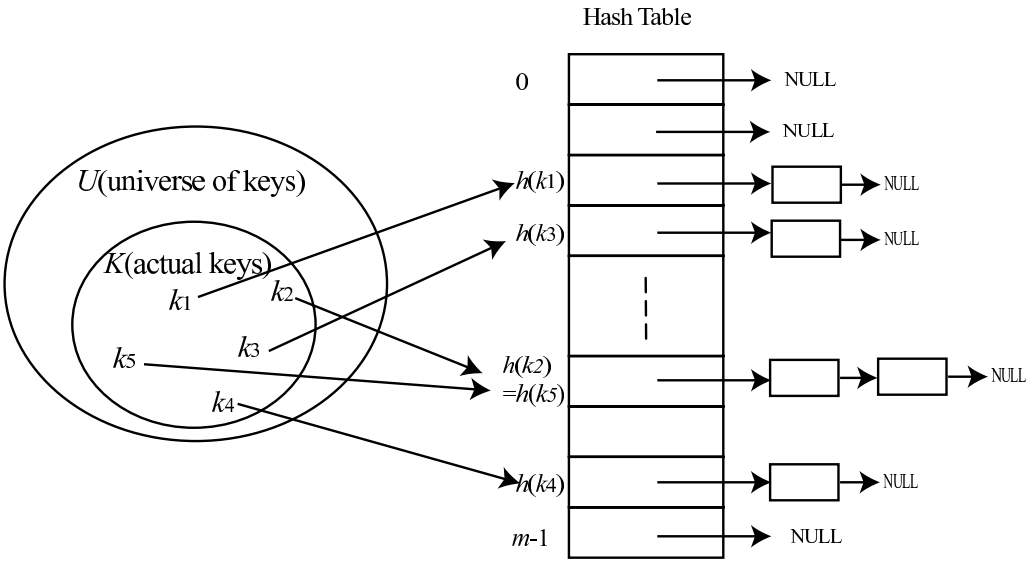


Figure 2.12: Chaining to Resolve the Collision Problem. The elements are stored as linked list in each slot.

Under this assumption, the average time for searching in hash table with chaining takes $\Theta(1+\alpha)$. The detailed discussion can be found in [5].

2.4.3 Minimal Perfect Hashing

The other method to resolve collisions is using perfect hashing. In this section, we describe the minimal perfect hash function which is special type of perfect hash function.

Let K be a set of n distinct keys belonging to some universe U and M be a set with m elements. A *perfect hashing* for K is an *injection* H from K to M: for all $x, y \in K$ such that $x \neq y$, $H(x) \neq H(y)$. A perfect hash function transforms each key in K into a unique index in the hash table. If n and m are the same, we have *minimal perfect hashing*.

Perfect and minimal perfect hashing are suitable only for static sets in which no deletion or insertion of elements occurs. Minimal perfect hash functions are used for memory-efficient storage and fast retrieval of items from a static set, such as keywords

in programming languages, command names in operating system command shells and so on. Owing to their applicability, there has been much research on finding a good minimal perfect hash function algorithms. [9] reviewed these researches.

If the order of keys and their hash values are preserved, i.e. for all $x, y \in K$ such that $x < y$, $H(x) < H(y)$, then the hashing is called *order preserving*. In [8], Czech et al. presented an order preserving minimal perfect hash function. The form of the hash function is :

$$H(k) = (g(f_1(k)) + g(f_2(k))) \mod m \tag{2.37}$$

where f_1 and f_2 are functions that map strings into integers and g is a function that maps integers into $[0, m-1]$. The functions g and f_i are realized as tables, so an implementation must store these tables. The storage required for this is $O(m \log m)$ bits, which is optimal for order-preserving minimal perfect hashing.

For any given key set, at least one minimal perfect hashing algorithm exists. The hash stores a sorted table of all keywords and, as usual, the location of each keyword is its hash value. In practice, finding any kind of perfect hash function, especially for large sets of keys, may not be easy. Only one function in ten million is a perfect hash function for 31 keys mapped into 41 locations [19]. If we consider the random placement of n keys into a hash table of size $m (m \geq n)$, then the probability that no collisions occur is

$$\frac{m(m-1)\cdots(m-n+1)}{m^n} = \frac{m!}{m^n(m-n)!} = \frac{\mathrm{P}_{m:n}}{m^n}, \tag{2.38}$$

where $\mathrm{P}_{m:n}$ represents the number of ordered sequences with length n from m element set (n-permutation of an n-set). The probability of placing n keys into an n element hash table with no collision is $n!/n^n$. For $n = 10$, it is only 0.00036. A crucial issue in perfect hashing is the efficiency of the hashing function. The hash function has to be computed

easily, otherwise there is no reason to use a hash function instead of directly searching the key space. One practical implementation of perfect hashing is gperf [35], which is available under the GNU license.

The time to search a key from the key space using perfect hashing is $O(1)$, because there is no collision. The time to compute a hash function can be assumed to be a constant. Technically, it is $O(n)$ where n is the size of , but since it is being used on a static set, once n is determined, it can be considered a constant.

Chapter 3

Previous Work on BRDF Representation

Many researchers have addressed the problem of representing reflectance. In this chapter, we will review some of the more popular representations.

BRDFs used in computer graphics are produced in one of two ways: analytic derivation or direct measurement. We can further categorize the analytical derivation approach into two subclasses: ad-hoc models and physically-based models. We will discuss each category in roughly chronological order.

3.1 Ad-Hoc Analytical Models

Ad-hoc models are created out of a need to exhibit observed behavior, but their derivation does not involve physical considerations.

Phong[31] presented a model that was designed to capture the behavior of roughened surfaces by raising a cosine to a user-specified power and scaling the result by a user-specified coefficient. This model is relatively old and not physically correct, but is still widely used in computer graphics because it produces fairly good images, is computation-

ally inexpensive, and, with some experience, it is easy for the user to control – to "model" – parameters to obtain a desired material appearance. .

Blinn[2] modified Phong's work by introducing a "halfway vector" and this model is represented as:

$$f_r(\mathbf{S}, \mathbf{V}) = k_d + ks\frac{(\mathbf{N} \cdot \mathbf{H})^n}{\mathbf{N} \cdot \mathbf{S}}, \tag{3.1}$$

where $\mathbf{S}, \mathbf{V}$ are incident and viewing direction, $\mathbf{H} = (\mathbf{S} + \mathbf{V})/|\mathbf{S} + \mathbf{V}|$ is the unit halfway vector between $\mathbf{S}$ and $\mathbf{V}$,and n is the specular reflection exponent, k_d and k_s are the diffuse and specular reflection coefficients respectively. Whether to use a Phong or a Phong-Blinn model is often the choice of the hardware or rendering package implementer.

Lewis[26] derived physically plausible representations of these models. "Physically plausible" is a weaker constraint than "physically based" in that it accepts forms of reflectance models which cannot be ruled out on the basis of energy conservation and reciprocity.

3.2 Physically-Based Analytical Models

As graphics capabilities expanded, so did the need for realism in images. We would, however, assert that physically-based models predate ad-hoc ones, as we must in fairness consider the work of Lambert[22] to be physically-based. Lambert's derivation was based upon reasoning about the nature of reflectance. Lambert's Law, which is that the amount of light reflected on the surface is independent of the viewer's direction and depends only on the angle between the incident light direction and the surface normal, is still widely used for diffuse objects and has even been "extended" by Oren and Nayar[30] to non-diffuse surfaces.

Also for non-diffuse cases, Cook and Torrance[4] derived a model based on geo-

metrical optics, assuming specular V-grooves and incorporating masking and self-shadowing effects. The specular component of their model is written as :

$$f_r(\mathbf{S}, \mathbf{V}) = \frac{F}{\pi} \frac{DG}{(\mathbf{N} \cdot \mathbf{V})(\mathbf{N} \cdot \mathbf{S})}, \tag{3.2}$$

where D is a distribution function of the microfacet orientation, G is the geometrical attenuation factor, and F is the Fresnel term computed by Fresnel's equation. $\mathbf{N}$ is the surface normal, and $\mathbf{S}$, $\mathbf{V}$ are the incident and the reflected directions. Schlick[34] extended this model with a more comprehensive one that has seen considerable use in the literature.

He et al. [15] extended the Cook-Torrance model to the region of physical optics. This takes into account polarization, surface roughness, masking, and shadowing and is given by a single formula which consists of specular, directional diffuse, and uniform diffuse terms. This model is not used much so far, because it requires greater computational cost than a non-physically-based one and it is difficult to control the parameters.

3.3 Data-Driven Models

With the advent of image-based rendering in recent years, another popular way to represent reflectance has arisen: approximating or interpolating measured data or data computed from a more complex analytical model. The mathematical representation used derives neither from qualitative desirability or control needs as in the case of the ad-hoc methods nor from a rigorous physical derivation as in the case of the physically-based methods. Instead, it comes from finding an efficient and compact fit to the data itself: It is "data-driven". Some of these enforce physical plausibility, which is certainly desirable when fitting real-world data.

Ward[23] simplified the geometric attenuation and Fresnel factors from the Cook-

Torrance model into one normalized parameter and incorporated them into a Gaussian lobe model. In addition, he incorporated two roughness parameters into an elliptical Gaussian model and successfully fit data that he had measured on his own apparatus. The isotropic model of Ward is :

$$f_r(S,V) = \frac{\rho_d}{\pi} + \rho_s \cdot \frac{1}{\sqrt{\cos\theta_s \cos\theta_v}} \cdot \frac{\exp[-\tan^2\delta/\alpha^2]}{4\pi\alpha^2}, \tag{3.3}$$

where ρ_d is the diffuse reflectance, ρ_s is the specular reflectance, δ is the angle between surface normal and halfway vector, α is the standard deviation of surface slope. θ_s and θ_v are the polar angles of incident and reflected directions.

Other popular representations use basis functions. As a BRDF is generally represented as a function of spherical coordinates, it is natural to use a basis function defined over spherical coordinates. Westin et al.[40] and Sillon[38] represented a BRDF as a weighted sum of spherical harmonics. A general form of these models is :

$$f_r(S,V) = \sum_j \sum_k Y_j(\theta_s,\phi_s) m_{jk} Y_k(\theta_v,\phi_v), \tag{3.4}$$

where m_{jk} are the coefficients, and Y_j and Y_k are the spherical harmonic basis functions. Westin's model can represent both isotropic and anisotropic BRDFs. To obtain an accurate representation of specular data, however, these models require many terms. They store the coefficients for only half of the hemisphere to save storage. To achieve a smooth representation, they represent the BRDF as $f_r(S,V)\cos\theta_s\cos\theta_v$. Multiplying by $\cos\theta_s$ forces C^1 continuity at the equator of the hemisphere and drastically reduces ringing. Multiplying $\cos\theta_v$ maintains the symmetry of coefficients $m_{jk} = m_{kj}$ which assures the reciprocity of the BRDF.

Schröder and Sweldens[36] show how to build "second-generation wavelets"

which make it possible to define wavelets on nearly arbitrary domains. As an example, they represent a BRDF with wavelets defined on a sphere. Their implementation uses fewer coefficients than spherical harmonics and coefficients can be computed efficiently, but their approach only uses wavelets with two degrees of freedom. The evaluation time of BRDF depends on the wavelet's resolution.

Koenderink et al.[20] constructed an orthonormal basis based on the Cartesian product defined on the hemisphere by mapping Zernike polynomials onto a unit disk. Then they project the BRDF into this vector space. Zernike polynomials appear to offer some advantages over spherical harmonics, but still require more terms to capture greater specularity.

Lafortune et al.[21] represented a BRDF as a non-linear summation of powers of cosine lobes. It may be considered as a generalization of Phong's original formula. This model can represent important BRDF behavior such as off-specular reflection and a retro-reflection with small number of parameters, and it is currently a very popular reflectance model.

Some researchers represent a BRDF as a product, rather than a summation, of functions. Fournier[14] separates a BRDF into two functions, one a function of the incident direction and the other of the reflected direction, using the technique of singular value decomposition. He then sums the products of these two functions with weights and applies the results to a Phong model and to Ward's experimental data[23].

McCool et al.[29] also used this technique, decomposing a BRDF into products of two or more functions of lower dimensionality. The form of their model is :

$$f_r(\mathbf{S}, \mathbf{V}) = \prod_j p_j(\pi_j(\mathbf{S}, \mathbf{V})), \tag{3.5}$$

where the p_js are two dimensional functions which, upon implementation, are stored as 2D texture maps. π_j is a projection function: $\mathbf{R}^4 \rightarrow \mathbf{R}^2$, associated with each map. Their method can take advantage of graphics hardware that accelerates texture mapping.

Chapter 4

Fitting BRDF Data

Our objective in this thesis is to obtain a smooth representation of reflectance data, specifically measured BRDF data. However, the representation presented in this thesis can be applied to any reflectance data, either captured by direct measurement or computed analytically.

BRDFs are smooth and flat for diffuse data and has peak value for specular data. If the material is shiny, then the slope of the peak is sharp. In general, a material has both diffuse and specular. Therefore, a representation of BRDF has to be able to represent smooth shapes as well as high peaks.

Other important property of BRDF representation is efficiency. Evaluation of a fit must be done efficiently. If the evaluation cannot be done efficiently, the representation may not be practical. Although computational power is dramatically increasing, demand for speed in graphics applications is always great. Thus, evaluation time is critical in most graphics applications.

To construct a representation that fulfills these requirements, we apply the MBA algorithm proposed by LWS (see Section 2.3.4) to fit a measured BRDF data set. This

algorithm can be applied to scattered data of arbitrary dimensionality. A measured BRDF data set is not necessarily uniformly sampled, is usually sparse and its dimensionality is three (isotropic) or four (anisotropic). In addition, the efficiency of evaluation is $O(1)$ regardless of the density of the control point mesh or number of samples in a data set.

First, we describe the BRDF databases we will use, then we will describe how we can apply MBA to them.

4.1 Source of Measured BRDF Data

Two on-line databases of BRDF are publicly available.

Foo [13] measured BRDFs using a custom-built gonioreflectometer and his data sets are available from the Cornell measurement web site [6]. He measured BRDFs for 1024 different wavelengths and bandpass filtered them down to 31 or 65 wavelengths. For each wavelength, reflectances were measured for more than 1,300 points. Assuming isotropicity, he fixed the incident azimuth angle as $\phi_s = 0$ and varied the other three parameters $(\theta_s, \theta_v, \phi_v)$. This database contains BRDF data of glossy and diffuse automotive and house paints. The Cornell group also measured human skin BRDF using a digital camera [28] and this data set is also available from same web site.

Dana et al.[10] used a robotic manipulator and CCD camera for their BRDF measurement. They performed radiometric calibrations to obtain radiance from a pixel value that was captured by CCD camera. Their data is available from the Columbia-Utrecht (CUReT) database web site [7]. This database has reflectance measurements for over 60 different samples, with a wide range of materials from very diffuse to highly specular. Each sample was measured for over 200 different combinations of viewing and incident light directions.

Both the Cornell and the CUReT BRDF databases measured BRDFs for only half of the upper hemisphere. We could just use these data as is, however, this causes an artifact at the boundary of the half hemisphere, because MBA tries to fit 0 where there is no data. This would cause a "dent" on a fit along the x axis (i.e. $\phi = 0$ or $\phi = \pi$). The data we represent will not only be isotropic (discussed below), but symmetric: $f_r(\theta_s, \phi_s, \theta_v, \phi_v) = f_r(\theta_s, \phi_s, \theta_v, -\phi_v)$. We must therefore reflect the data to get BRDFs for negative ϕ_v's and fit data over the entire hemisphere. Note that this symmetry is different from Helmholtz reciprocity (cf. [26]), which would hold that $f_r(\theta_s, \phi_s, \theta_v, \phi_v) = f_r(\theta_v, \phi_v, \theta_s, \phi_s)$.

4.2 BRDF Parameterization

In general, four parameters are used to represent a BRDF. In the case of an isotropic surface, the BRDF is invariant under rotation around the surface normal. Therefore, three parameters are enough to represent BRDF as $f_r(\theta_s, \theta_v, \delta\phi)$, where $\delta\phi = \phi_v - \phi_s$. Then, the reflected radiance L_r in the direction (θ_v, ϕ_v) can be represented as:

$$L_r(\theta_v, \phi_v) = \int_{\Omega^+} f_r(\theta_s, \theta_v, \delta\phi)\, L_i(\theta_s, \phi_s)\, cos\theta_s\, d\omega. \tag{4.1}$$

If the incident direction projected onto the positive x axis ($\phi_s = 0$), then $\delta\phi$ would be equal to ϕ_v. If the incident direction were not so aligned, then we can conceptually rotate the (isotropic) surface element around the surface normal until it is. Hence we may denote a BRDF as a function of three parameters: $f_r(\theta_s, \theta_v, \phi_v)$.

We can parameterize a BRDF strictly in terms of (possibly scaled) angles, but this would leave us with a polar anomaly. Our fitting procedure would not realize that $f_r(0, 0, \phi_v)$ refers to the same point for all ϕ_v. Any fit procedure could therefore produce undesirable artifacts near normal incidence and reflection. We need to adopt a coordinate

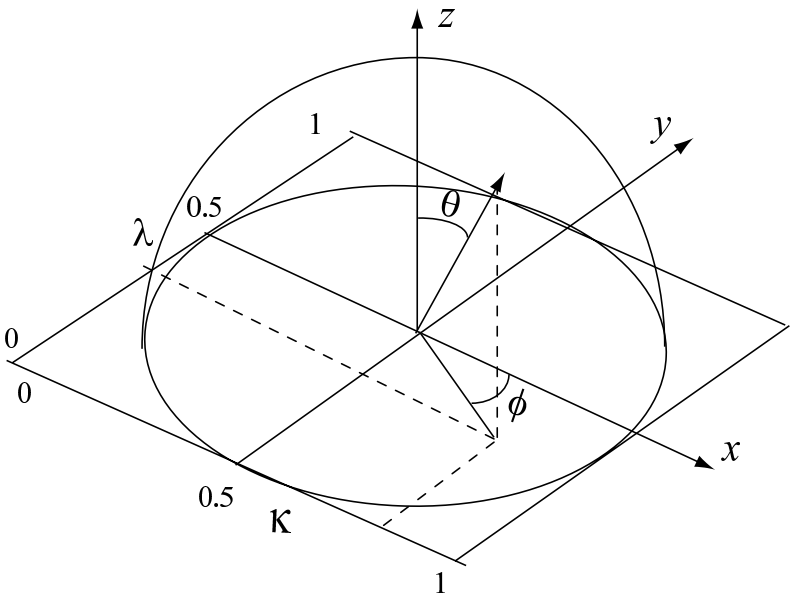


Figure 4.1: Conversion from Polar and Azimuthal Angles (θ, ϕ) to Nusselt Coordinates (κ, λ).

system that avoids this anomaly.

One such system, described in [27], is "Nusselt coordinates". Referring to Figure 4.1, we consider the x, y, and z direction cosines corresponding to a direction (θ, ϕ):

$$\begin{aligned} \mu_x &= \sin\theta\cos\phi \\ \mu_y &= \sin\theta\sin\phi \\ \mu_z &= \cos\theta. \end{aligned} \tag{4.2}$$

We then transform μ_x and μ_y to lie between 0 and 1:

$$\kappa = \frac{\mu_x + 1}{2},\ \lambda = \frac{\mu_y + 1}{2} \tag{4.3}$$

These are the Nusselt coordinates. The mapping $(\theta_s, \theta_v, \phi_v) \Leftrightarrow (\kappa_s, \kappa_v, \lambda_v)$ is unique

and invertible. By applying this coordinate representation, a BRDF can be represented as $f_r(\kappa_s, \kappa_v, \lambda_v)$.

An added benefit of Nusselt coordinates is that μ_x and μ_y (and therefore κ_s, κ_v, and λ_v) are usually readily available as part of the scene geometry during rendering: no inverse trigonometry is required.

4.3 Finding the Fit

Now we apply MBA to fit the measured BRDF data. This process turns out to be straightforward. As in the Cornell and CUReT BRDF databases, most measured BRDF datasets are represented with the polar and the azimuthal angles for incident and reflected direction: $(\theta_s, \phi_s, \theta_v, \phi_v)$. We first convert these four parameters representation to three parameters by computing $\delta\phi = \phi_s - \phi_v$. Then we convert the spherical coordinates to Nusselt coordinates. We apply a three-dimensional version of MBA to the resulting data set to find the fit.

We start from finding fit with the coarsest control mesh which is the size of 1. Then refine the density of the control mesh if necessary. For refinement, we simply double the size of the control mesh for all dimensions. The total number of control points S for a three dimensional MBA at refinement level h is:

$$S = (M + 3)^3 = (2^h + 3)^3 \tag{4.4}$$

The number of control points is 64 at the first level ($h = 0$), 125 at level 1 ($h = 1$) and so on.

The necessary refinement level can be determined based on the accuracy of the resulting fit, which we can be measured by a metric such as root mean square error.

The final output of MBA is the fit represented as a mesh of control points. Since the locations of control points are determined by the extent of the domain and the mesh size, we need only store the values of the control point as an array $\{\phi_{i,j,k}\}$, where $i, j,$ and k are the indices on the control mesh. We do not need to store the basis functions along with the control mesh, as their forms are fixed.

4.4 Evaluation

Once we have the fit, we can evaluate it for an arbitrary direction within the extent of the domain. For a given $(\kappa_s, \kappa_v, \lambda_v)$, the BRDF can be calculated by :

$$f_r^h(\kappa_s, \kappa_v, \lambda_v) = \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(s)B_m(t)B_n(r)\phi^h_{(l+i)(m+j)(n+k)}, \tag{4.5}$$

where B_l's are bi-cubic B-spline basis functions and the ϕ^h's are the control points on Φ_h, a control mesh of a refinement level h. To calculate (4.5), we have to determine which control points $\phi^h_{(i,j,k)}$ are used and the relative distance between a given points and nearest grids. s, t and r represent the distances between the grid point $(i+1, j+1, k+1)$ and the given point $(\kappa_s, \kappa_v, \lambda_v)$. To locate a position on the control mesh for a point, we need to map the Nusselt coordinate to the grid of control points (i, j, k). The control mesh at level h is of resolution $(M+3)^3$, where $M = 2^h$. The gap Δ between each grid point is $1/M = 2^{-h}$ for the level h control mesh. Hence

$$i = \lfloor \kappa_s/\Delta \rfloor - 1, \; s = \frac{\kappa_s}{\Delta} - \left\lfloor \frac{\kappa_s}{\Delta} \right\rfloor. \tag{4.6}$$

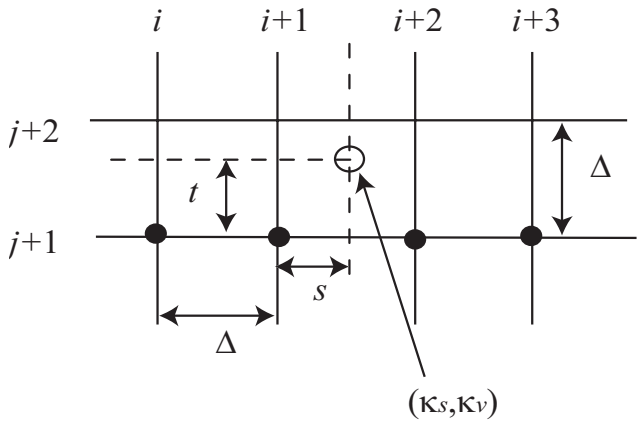


Figure 4.2: Conversion from Nusselt Coordinates to the Location Indices in 2D Case. For given (κ_s, κ_v), locate (i, j), then determine (s, t) that is distance from the grid $(i+1, j+1)$.

Similarly, j, t and k, r can be determined from κ_v and λ_v as follows:

$$j = \lfloor \kappa_v/\Delta \rfloor - 1, \quad t = \tfrac{\kappa_v}{\Delta} - \left\lfloor \tfrac{\kappa_v}{\Delta} \right\rfloor, \tag{4.7}$$

$$k = \lfloor \lambda_v/\Delta \rfloor - 1, \quad r = \tfrac{\lambda_v}{\Delta} - \left\lfloor \tfrac{\lambda_v}{\Delta} \right\rfloor. \tag{4.8}$$

Figure 4.2 shows this conversion in 2D case.

Chapter 5

Fit Results

We applied our method to the BRDF databases introduced in Section 4.1 and measured how well our representation fits the original data. We computed error between the original data and the fit, and measured the evaluation time to gauge the efficiency of our representation. We plotted fits along with original data and also synthesized spheres to evaluate the visual quality of the fit.

We selected BRDF data for felt (No. 1), leather (No. 5), aluminum foil (No. 15) from the CUReT database and latex blue paint from the Cornell database. Felt shows a mostly-diffuse surface with slightly-specular behavior at high incident and reflected polar (grazing) angles. Leather has a little bit of specularity everywhere and has high specular behavior at grazing angles also. Aluminum foil has high specularity. See [7] for images of the sampled materials. Latex blue paint is largely diffuse but it also has high specularity at grazing angles.

In addition, we picked two illumination models previously presented and fit the same data sets to compare with our method. We used the model proposed by Koenderink et al.[20] and fit the measured data with order 8 of their method. In the CUReT paper

[11], they used this model to fit their measured data. The other model is the one proposed by Lafortune et al.[21] with 3 cosine lobes. This model is preferred by many researchers because it is simple and looks good. Hereafter, we will refer to Koenderink method and Lafortune method, respectively. Their exact forms are presented in Appendix A.

5.1 Quantitative Results

5.1.1 Accuracy

One way to measure the accuracy of a fit is root mean square error (here after we will note "RMSE"), which is defined as :

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{N-1}(f_r(\kappa_{ii}, \kappa_{oi}, \lambda_{oi}) - d_i)^2}{N}}, \tag{5.1}$$

where $d_i, \kappa_{ii}, \kappa_{oi}, \lambda_{oi}$ are the i-th measured data from the data set, the incident and the reflected directions for the i-th data respectively. N is a number of samples in the data set.

A high specular BRDF has peaks in the mirror direction, and lower reflectance in other directions. The transition from peak to low value is drastic if the material is really shiny. Modeling this behavior is difficult in general. Hence, seeing how well the representation fits these peaks is important. We measured maximum absolute error to see how a fit represents these high peaks. Maximum absolute error (hereafter, "MAE") is defined as:

$$\text{MAE} = \max_i \left(|d_i - f_r(\kappa_{ii}, \kappa_{oi}, \lambda_{oi})| \right), \tag{5.2}$$

where $\max_i(x_i)$ returns the maximum value of x_i. If the RMSE is very small, then we can say that most of the evaluated values are close to the original data. However, if MAE is

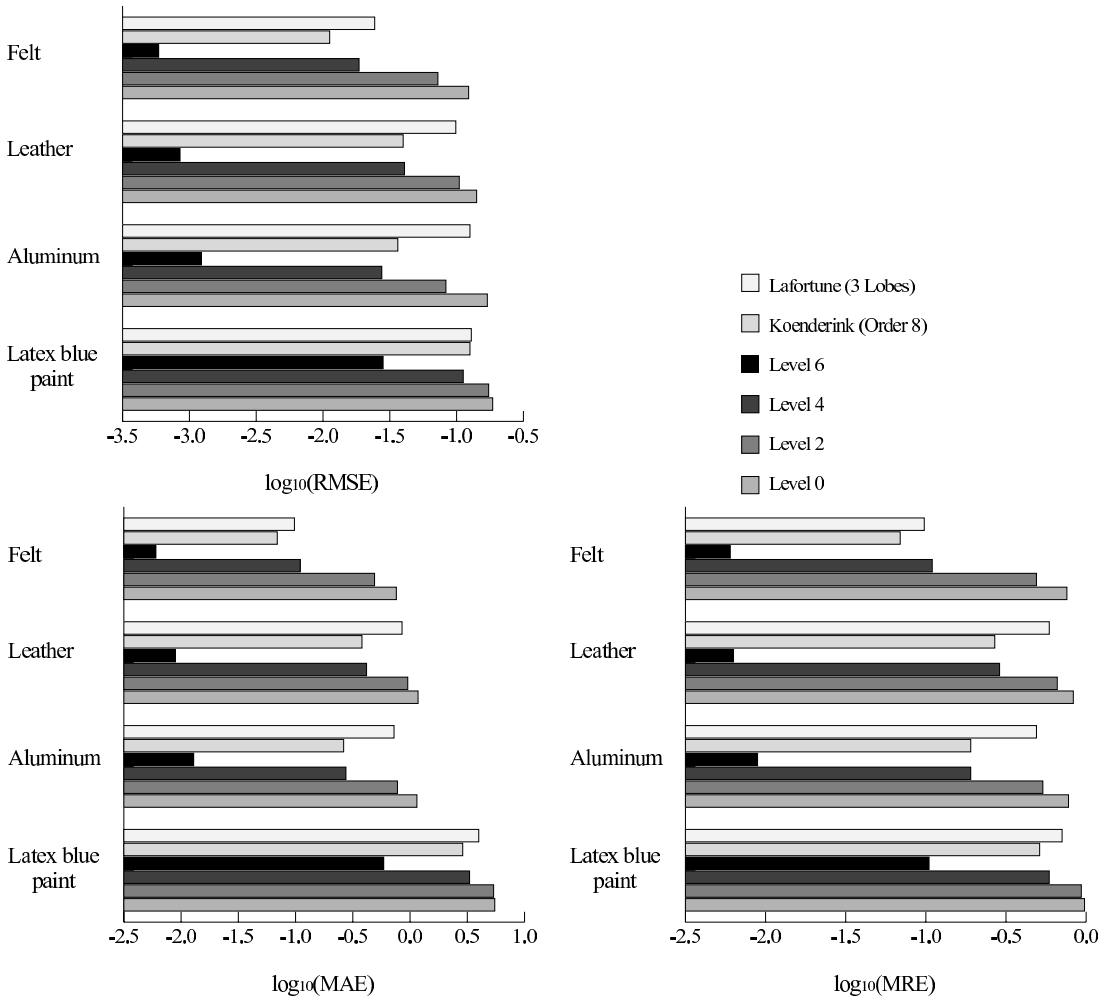


Figure 5.1: Error Measurement Results of Our Fit for Levels 0, 2, 4 and 6, Compared with Koenderink and Lafortune Models.

large while the RMSE is relatively small, then we can say that the fit cannot reproduce the high peaks even if most of the diffuse part can be fit.

We also measured maximum relative error (hereafter, "MRE") which is defined as:

$$\text{MRE} = \frac{\max_i \left(|d_i - f_r(\kappa_{ii}, \kappa_{oi}, \lambda_{oi})|\right)}{\max_i \left(|d_i|\right)}. \tag{5.3}$$

If the data set has only low value of data, then the MAE would be small even if the fit

does not match the original data very well. On the other hand, if the original data have large values, then the MAE would be large. MRE measure the relative scale of error. If the values of data set are small, the it would return the large value if the error is large compared to the maximum value of the data set.

The results of error measurements at refinement levels 0, 4 and 6 are shown in Figure 5.1. The numeric values are tabulated in Table B.1. The errors for other levels up to 5 are tabulated in Table B.2.

The upper left chart in Figure 5.1 shows RMSE. Our level 6 fit achieved great accuracy compared to the other levels and other methods. For the CUReT data, it achieved RMSE of order 10^{-3}. The other two fitting methods show almost the same accuracy as our level 4 fit. The bottom left of Figure 5.1 shows MAE and bottom right shows MRE. Again, the level 6 fit achieves comparatively greater accuracy.

The errors for latex blue paint are larger than for other materials. We believe that the reason for this is that the data is extremely large in grazing angles. All methods cannot fit closely to this high peak, but only level 6 of our method can fit well. As these graphs showed, our fit can improve the accuracy by refining the level.

Scatter plot is an alternative way to see a statistical measurement. The x axis of the plot is a value of measured data and the y axis is a evaluated value. If evaluated values are exactly same value for every measured data, then every point is plotted exactly on the $x = y$.

Figure 5.2 to 5.5 show the scatter plots of fit for each BRDF and their original data. A diagonal line on each plot is $x = y$.

The level 0 fit shows a poor fit: most evaluated values are not close to the data. Instead, they are scattered around a line that is parallel to the x axis. So, the fit yields an average value of the data. However, as the level increases, the scatter points get closer to

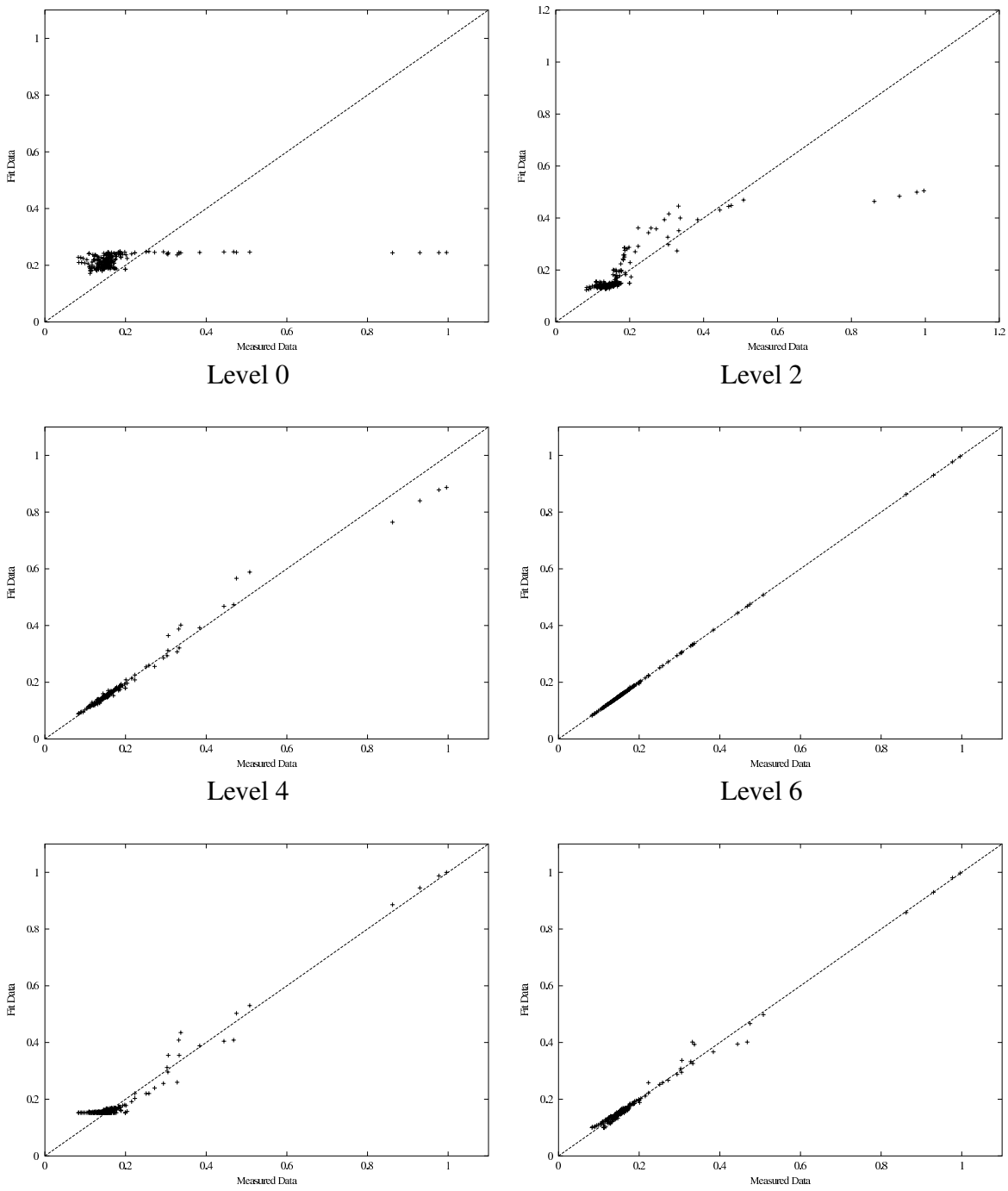


Figure 5.2: Scatter Plot for Felt.

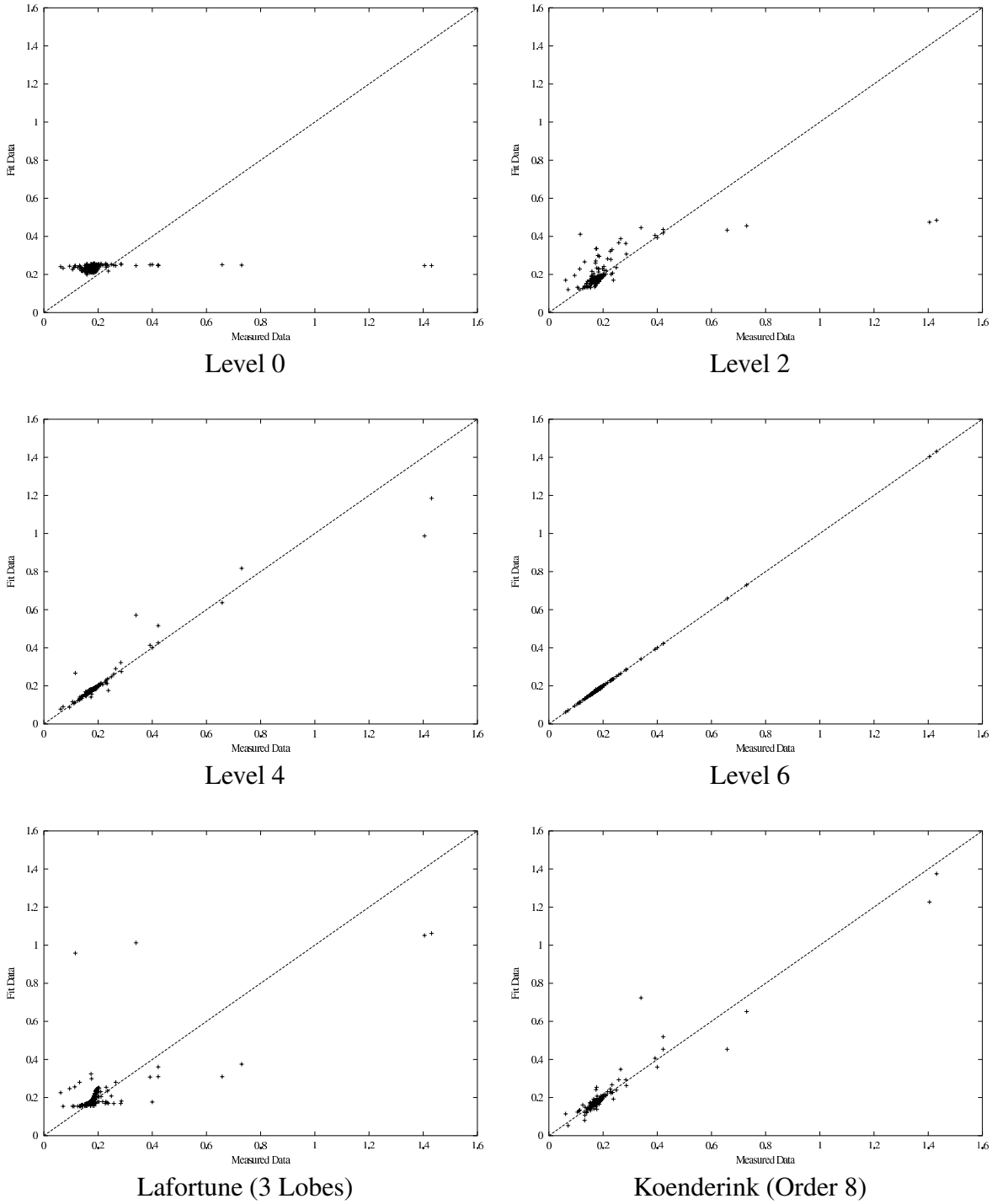


Figure 5.3: Scatter Plot for Leather.

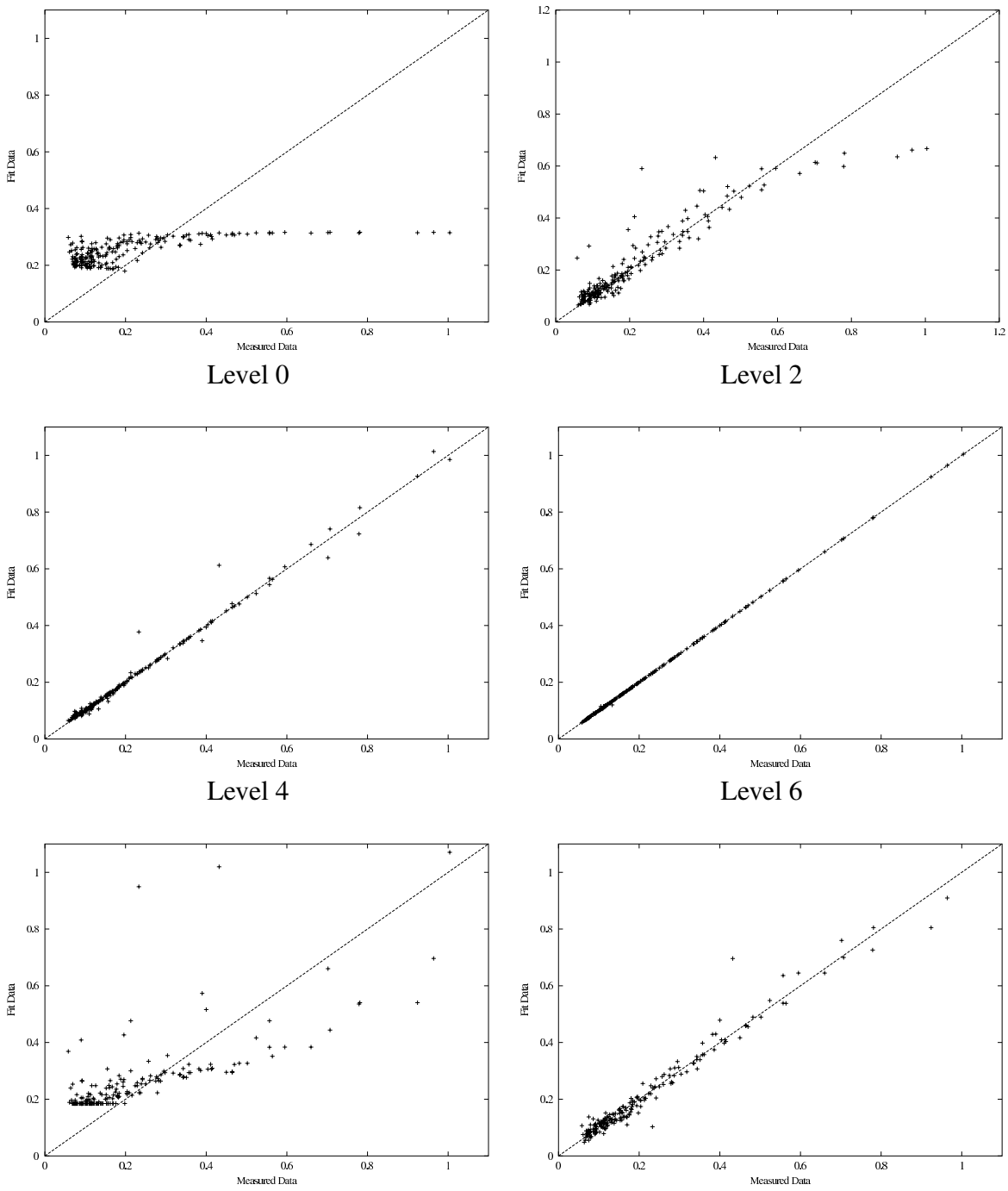


Figure 5.4: Scatter Plot for Aluminum.

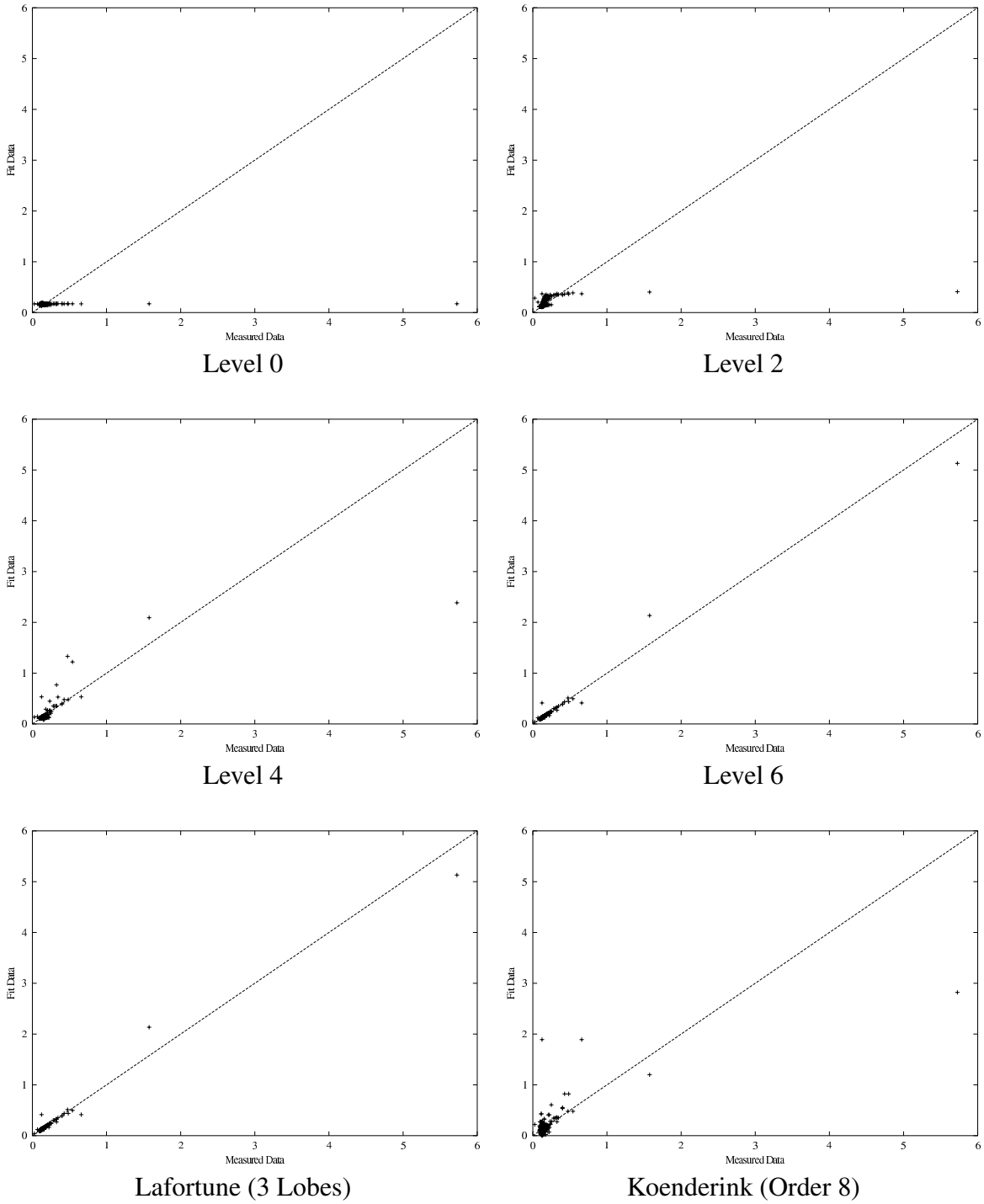


Figure 5.5: Scatter Plot for Latex Blue Paint.

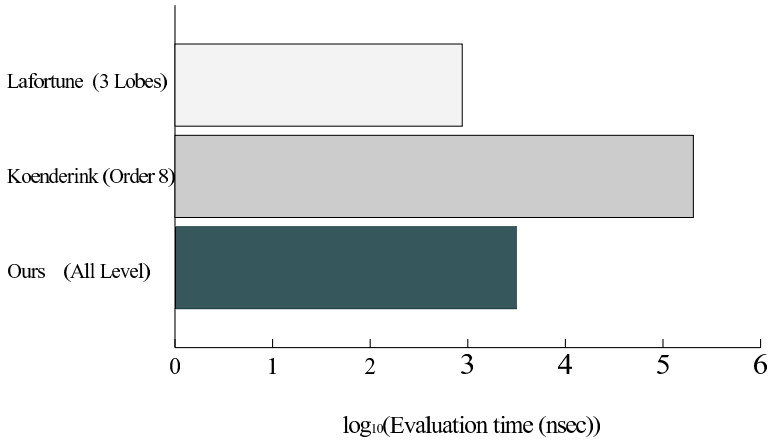


Figure 5.6: Fit Evaluation Time Comparison.

the diagonal line. Finally, at level 6, the plot is very close to the line. Error results of other method are similar as our level 2 or 4 fit, but not as a good match as our level 6 fit.

5.1.2 Speed

If a BRDF representation uses basis functions, it is possible to represent the original data exactly if an infinite number of basis function is used. This is, of course, impractical and we have to limit the number of basis functions. If we, on the other hand, do not use enough basis functions, then the fit will show "ringing" and cannot capture peaks of the data. To eliminate ringing, the high frequency basis functions have to be included, since they represent sudden changes of the data. If a certain high frequency basis is included, then all lower frequency basis functions also have to be included.

Thus, we need many basis functions to represent a BRDF with high specular values and this makes many basis function based representation be slow.

We measured the evaluation time and compared it with Koenderink and Lafortune methods. We used Koenderink's fit with order 8 and Lafortune's fit with 3 lobes. Time

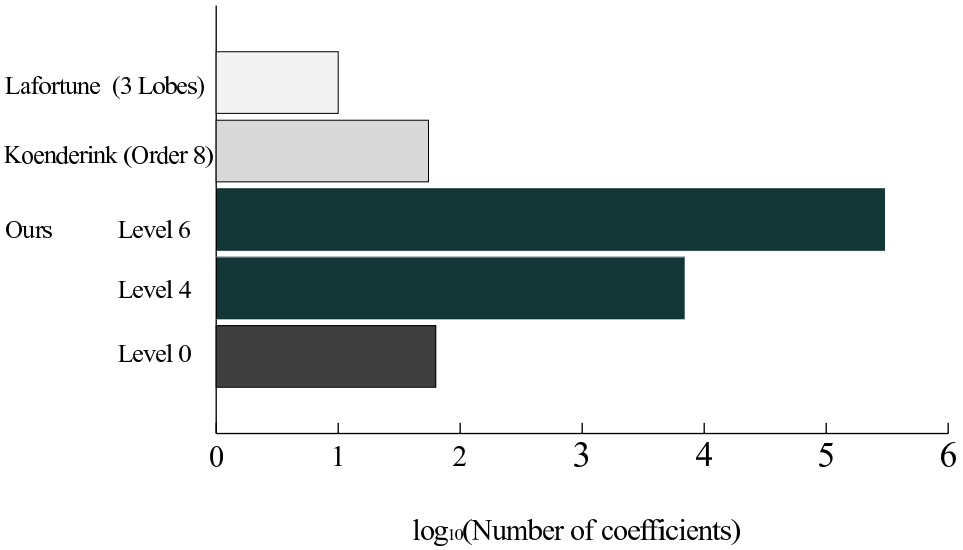


Figure 5.7: Storage Comparison with Koenderink and Lafortune Models.

is measured by computing the means of 100,000 function calls for different combinations of incident and viewing direction. The evaluations were done on a PC running Linux Red Hat 7.2 on an AMD Athlon 1.4 Ghz microprocessor with 256 KB cache and 256 MB RAM. The results are shown in Figure 5.6.

As we can see from (4.5), the evaluation time of our method does not depend on a level, instead it is constant for all levels. It only depends on the dimension of parameters (3 for isotropic BRDF). Therefore we just listed the result for one level in the figure. Lafortune method is very fast because the form is simple and it is a little bit faster than our method.

5.1.3 Storage

Figure 5.7 shows the storage requirements. We counted the number of control points for our method and the coefficients for other methods. These control points or coefficients have to be stored to use each fit.

Our method with level 6 requires far more storage compared to other methods.

Although the errors at this level are negligible, the storage requirements are not. If we use a single precision float value (4 bytes) to represent a control point, we need about 1.2 MB of storage to represent the fit (Remember that this applies to a single spectral channel, so this number would be tripled for the standard three RGB coefficients in a practical application.).

5.1.4 Tradeoffs

As we have shown, the Koenderink method of order 8 is fairly good for some materials, but the computational cost is dramatically increased at this order. Hence it is not suitable for practical use. Some of the previous works that use basis functions suffer from this problem. On the other hand, the Lafortune method is quite fast, since their representation is simple. Our method is not as fast as Lafortune's, but our high level fit can achieve far more accuracy than Lafortune's fit. Since their model is fast, it could in principle use more basis functions (lobes) to increase the accuracy of the fit without increasing evaluation time dramatically. However, it is quite difficult to obtain a good fit with more lobes for their model. We fit the data with Lafortune model by the Levenberg-Marquardt nonlinear fitting algorithm [32], which is also used to find the fit in Lafortune's paper. With 3 lobes, we got similar results as they had shown. To increase the accuracy of the fit, we tried to use more lobes than they had presented, but we could not obtain a good fit. A fit result is highly dependent of the initial value of nonlinear fitting and the program could not converge. For 5 lobes, we have to correctly guess 16 parameters, which is very difficult. W. Heidrich also confirmed [16] that the Lafortune model with a higher number of lobes is numerically instable and cannot yield a good result.

Although our method with higher levels achieves great accuracy and comparatively fast evaluation time, the fit requires far more parameters than others. This storage requirement is a disadvantage for the MBA method. In Chapter 6 we will present a revised

method to reduce the storage requirement.

5.2 Qualitative Results

5.2.1 Smoothness

To visualize how well MBA can match the measured data, we plot the results in 2D along with the measured data, showed in Figure 5.8 and 5.9.

The fits with level 0 are smooth but we cannot say that they fit the data very well. They just capture the average of a data set. However, at level 2 or finer level, they represent the measured data very well and maintain smoothness.

For latex blue paint data, the fit of level 6 shows ringing at $\kappa_i \sim 0.07$ where the BRDF falls rapidly. The measured data at this angle has extremely high values compared to other directions. More samples near this direction could reduce the ringing.

5.2.2 Image

We synthesized spheres with fits produced by our method. Figure 5.10 shows them with a fit of the levels 0,2,4 and 6. These images show the effect of refinement. The sphere in each image is lit by a point light source. Figure 5.11 shows a sphere synthesized with level 6 fit. A sphere is lit by a point light source from various directions.

These images show the expected characteristics of each material. They capture the high specularity and the brightness transition smoothly from specular highlight to mainly off-specular diffuse. As the refinement level increases, the specular region narrows and its brightness increases. This shows that the fit improves as the refinement level increases. However, even in level 2, the images represent material characteristics adequately.

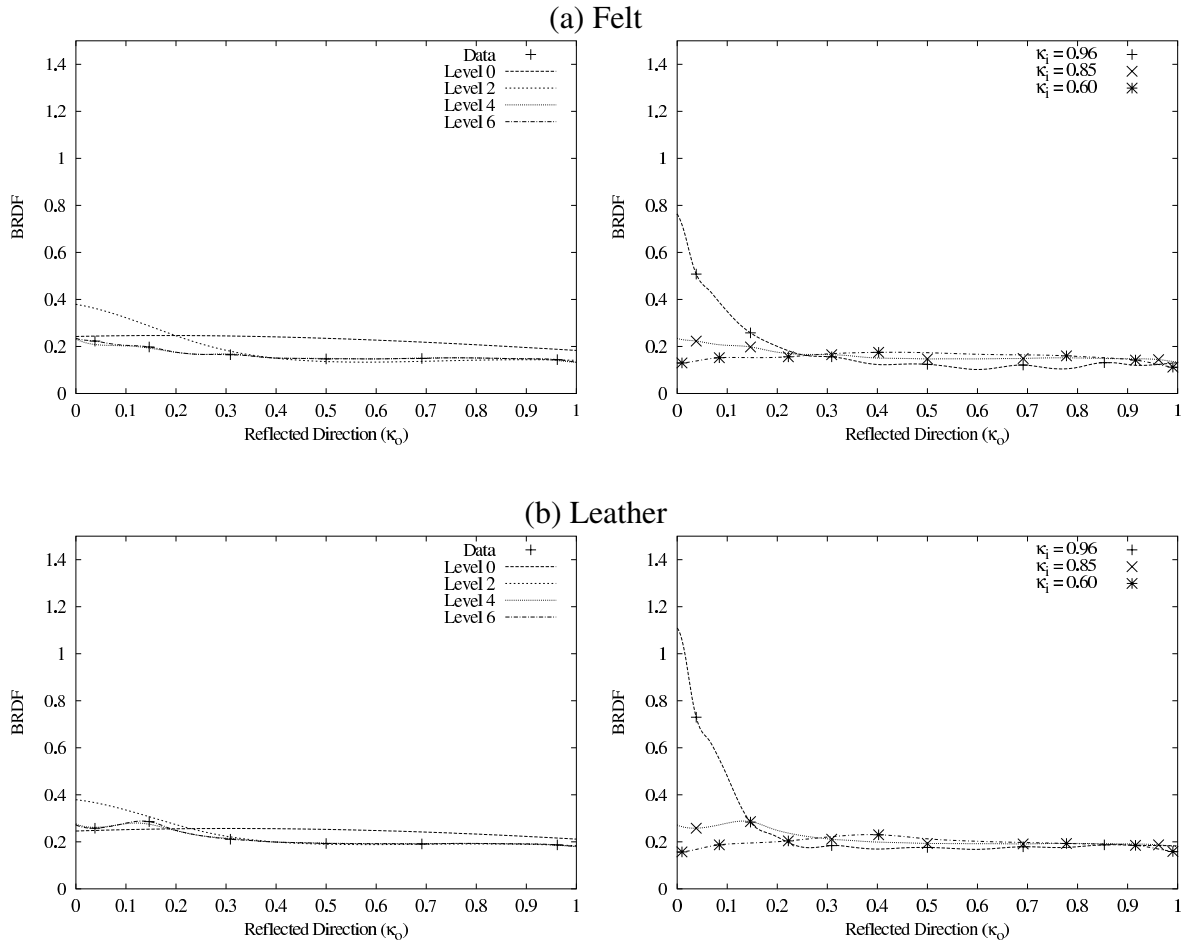
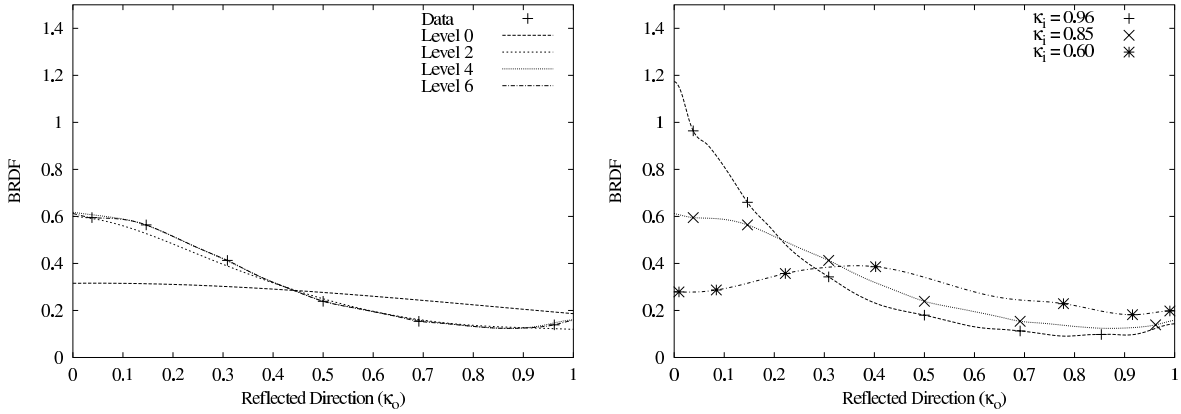


Figure 5.8: BRDF Fit Results (1). Left side show the fits of various refinement levels for aluminum and latex blue paint. Incident angle of these plot is $\kappa_i = 0.85$. Plots on right hand side show the various incident angles for level 6 fit. The abscissae of plots represent the reflected direction κ_o.

So, if we can sacrifice some accuracy, we can fit data with less storage.

With an ad-hoc BRDF model, increasing specularity at grazing angles is difficult to represent [21], but our model correctly represents such specularity. Extremely high values in the latex blue paint data causes ringing in the level 6 fit as we have seen in Figure 5.9 This artifact is also observed in the synthesized sphere in Figure 5.10 and 5.11. We see a dark area around the highlight. We could avoid this situation if we could increase the

(c) Aluminum



(d) Latex blue paint

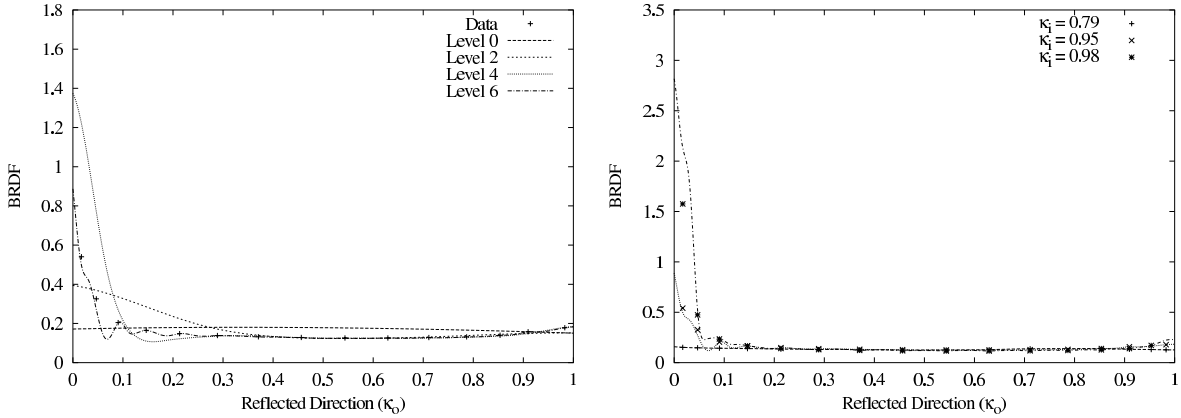


Figure 5.9: BRDF Fit Results (2). Left side show the fits of various refinement levels for aluminum and latex blue paint. Incident angle of these plot is $\kappa_i = 0.85$. Plots on right hand side show the various incident angles for level 6 fit. The abscissae of plots represent the reflected direction κ_o.

density of samples around this area. This is an additional advantage of MBA: it permits non-uniform sampling.

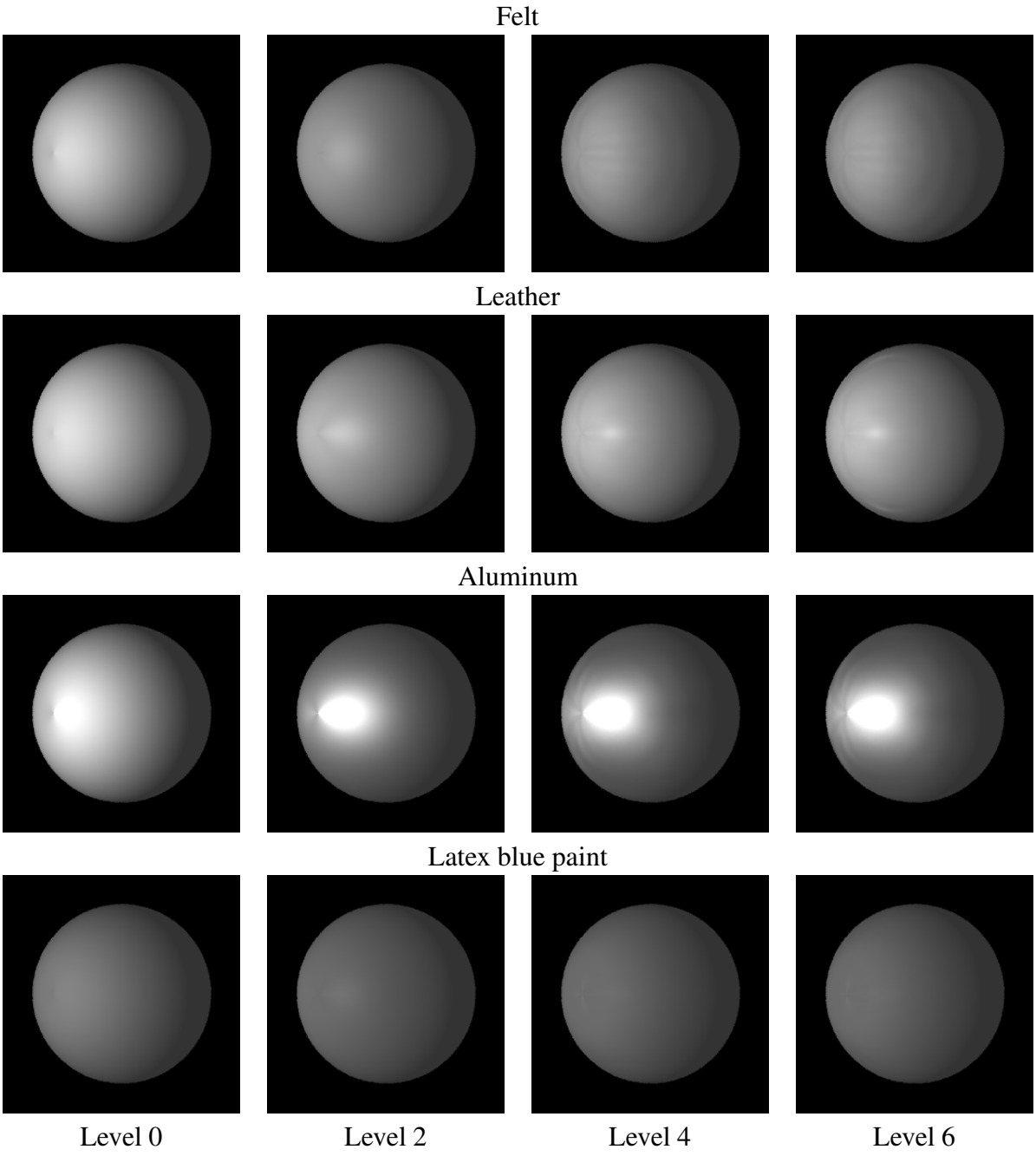


Figure 5.10: Sphere with BRDF Fit with levels 0, 2, 4 and 6. The light source is located at the 30° to the left from the viewpoint.

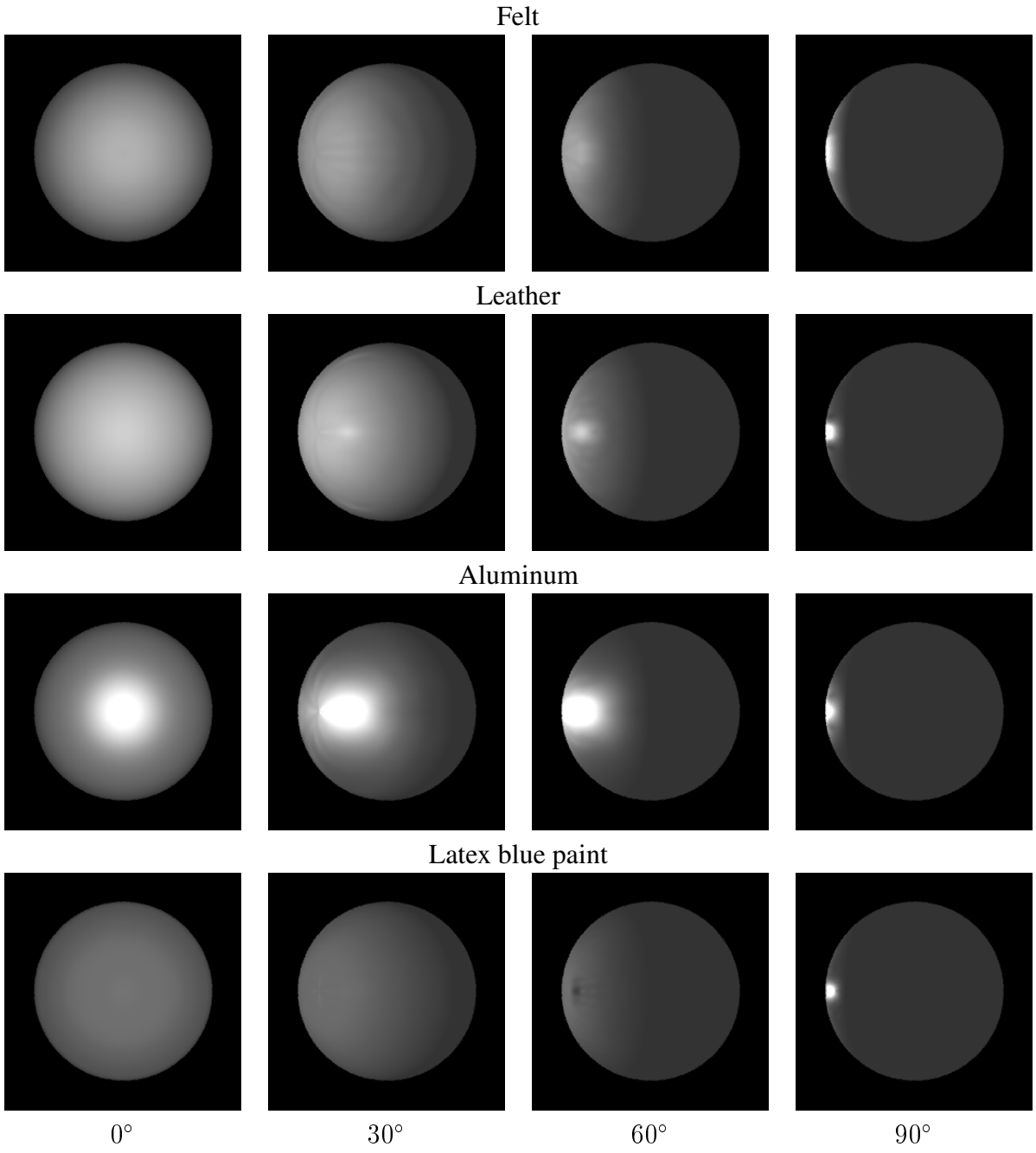


Figure 5.11: Sphere with Level 6 Fit. The light source is rotated $0, 30, 60$ and $90°$ to the left around the center of the sphere from viewing point.

Chapter 6

Compact Bi-level Multiresolution B-Spline Algorithm

In Chapter 4, we developed a smooth and efficient representation of measured reflectance data. However, the storage required to store control points at higher levels is not negligible. For level 6, there are more than 300,000 control points and more than 1.2 MB is needed to store those control points if we use single precision floating point. This amount by itself is not critical with today's memory availability. However, if we compare this requirement with other methods, our method is much more memory-demanding. If we synthesize a complex scene, we might need hundreds of different types of materials and we need to store the control points for all materials. In that case, this amount of memory usage would be a concern. Although other methods cannot achieve both the accuracy and efficiency at the same time that our method can, the required storage amount might be discouraging. We therefore need to investigate ways to reduce the storage. In this chapter, we will develop a revised algorithm, "Compact Bi-level Multiresolution B-Spline Algorithm", to reduce storage requirements.

6.1 Control Points Reduction

To reduce the number of control points, one possible way is simply omitting control points. When we look at the magnitudes of control points, many have very small values. Therefore, omitting those control points might not affect the overall shape of the fit. We can omit control points simply treating its value as 0. Then we can compress these zero-valued control points. We investigated this effect by measuring RMSE.

The RMSE for each material is shown in Table 6.1. We zeroed 5, 50, 87.5 and 93.4 % of the original control points. The evaluation of fit can be performed as before. Table 6.1 shows that this method does not work very well. Even with 5 % of control points omitted, the difference between the original method and the simple compression method is not negligible.

Another possibility to reduce the storage is changing the resolution of the control mesh adaptively. We can assign a denser mesh where specular peaks exist and a much coarser mesh to the rest. However, to distinguish such directions from a high dimensional scattered data would be a complicated task.

CUReT type	Level	Reduction				
		None(0 %)	5 %	50 %	87.5 %	93.4 %
Felt	5	0.0043	0.0121	0.0400	0.1112	0.1272
	6	0.0006	0.0148	0.0440	0.1121	0.1247
Leather	5	0.0154	0.0209	0.0484	0.1361	0.1559
	6	0.0008	0.0169	0.0574	0.1453	0.1572
Aluminum	5	0.0073	0.0272	0.0844	0.1191	0.1358
	6	0.0012	0.0310	0.0898	0.1216	0.1383
Total number	5	42,875	40,732	30,013	5,360	2,830
of control points	6	300,763	285,725	243,618	37,596	19,851

Table 6.1: RMSE of the Fit with Control Point Reduction. Even small number of cut affects the errors.

We can reduce the control points another way by dividing a BRDF into specular and diffuse components. We also utilize a minimal hash function to retrieve the value of control points. In the following section, we will describe how to decompose BRDF into two parts and how to reduce the storage requirement.

6.2 Decomposing the BRDF

Representing a BRDF with two components, specular and diffuse, is a common idea in computer graphics [3, 37, 34]. Motivated by this idea, we divide our fit into two parts. We refer these two fits as the diffuse fit f_d and the specular fit f_s. f_d has a coarser control mesh and f_s has a finer one. The BRDF is represented as the sum of these fits:

$$f_r = f_d + f_s \tag{6.1}$$

To construct these fits, we first fit the data with a low level MBA (i.e., a coarse control point mesh) even though the fit is not accurate. The resulting fit is f_d. We then calculate the error between f_d and the original data, then we take the resulting error as a data set and fit with MBA. The resulting fit is f_s. This process is the same technique as the intermediate process of MBA refinement. At this fitting stage, the initial mesh size for MBA is the same size of that of f_d. That is, if the mesh size of f_d is h_d, then the initial mesh size to find f_s is also h_d. We then refine the mesh size until a satisfactory result is obtained as before (See Figure 6.1).

f_d represents the low-resolution shape of the BRDF data set and is smooth but may not match the data very well, because we do not continue refinement. Hence it would fail to fit a specular BRDF, but it does capture the behavior of a diffuse BRDF. By taking the error between f_d and the original data, we can extract the specular component of the data.

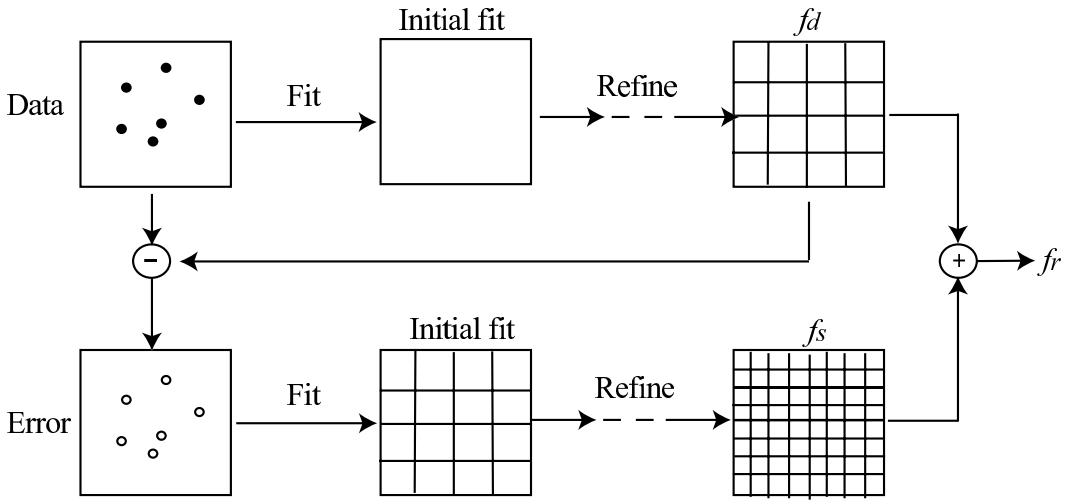


Figure 6.1: Representing BRDF with Two Fits.

The second part of our fit captures this component. Since f_s represents an error between f_d and an original data, the final form of our fit is represented as a sum of both fits as in (6.1)

6.3 Control Points Compression

f_s represents both the high specularity in the data and also high-frequency noise where f_d does not completely fit. If the magnitude of a control point is small, then the effect of that control point on the fit is also small. Hence we can omit those control points without major loss of accuracy of a fit. We omit small control points by treating their values as 0 and omit storage for (effectively) zero-valued control points, result in compression. However, we have to store extra information that tells us which control points are omitted.

For non-zero control points, we store the indices of the contort points (i, j, k) along with their values into the key-value table. The control points that are not in this table are treated as 0. To retrieve the value of control point of index (i, j, k), first we look up the key value in the table to check if it contains the index (i, j, k). If it does, then the value at

the location is returned. If not, then the control points is treated as 0.

The process of searching the specific index from key-value table would be time-consuming. This is same as a dictionary search. If we cannot find it efficiently, it degrades efficiency of our method. We utilize a minimal perfect hashing to do this.

The hash function h takes an index (i, j, k) as input and hashes to an integer a. a is used as the index (location) of key-value table. If the key at the location a in the table is same as (i, j, k), then the value of control point is the value at location a. If not, return 0 as the value of the control point.

The pseudo code for retrieving the value of control point is as follows:

```
a = h(i,j,k)
if key[a] == (i,j,k) then
   return value[a]
else
   return 0
```

In the next section, we describe how to construct the key-value table and minimal perfect hash function.

6.4 Retrieving Control Point Values

Suppose we have N_s control points for f_s, and want to omit N_c control points. Let M be $N_s - N_c$, the number of non-zero control points. After we get f_s, we sort its control points by their magnitude. We then take the M largest control points and store their values and indices in key-value table (Figure 6.2). Then we construct a minimal perfect hash function from the table. The input to h are the indices (i, j, k) of a control point and it returns a

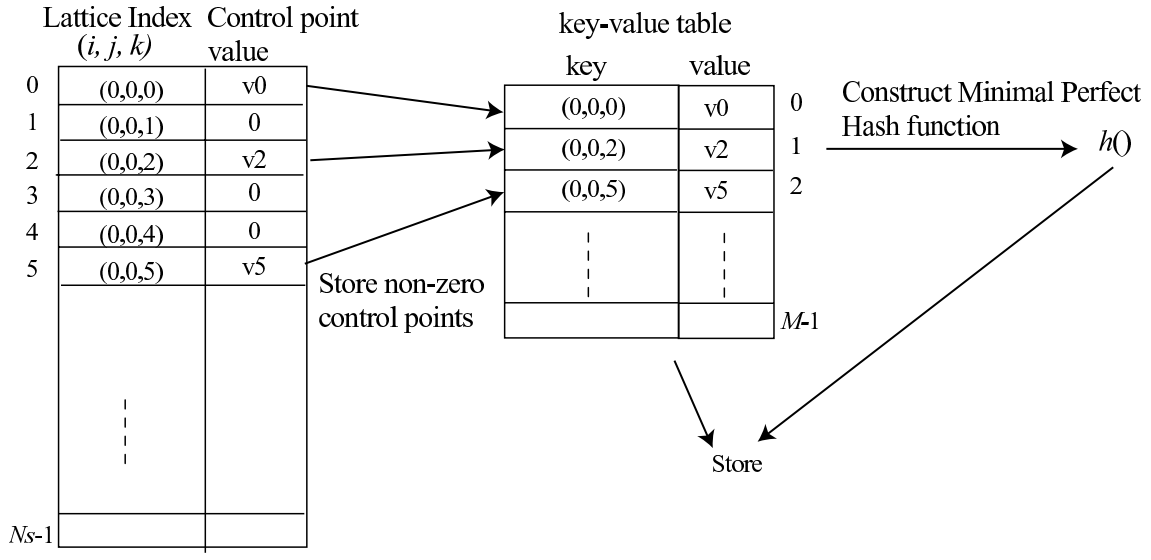


Figure 6.2: Construction of key-value table. Hash function h is constructed from this table and stored with key-value table.

location in key-value table. The reason why we need to use minimal perfect hashing is that it is time and storage efficient. In perfect hashing, no collisions occur. That is :

For all distinct $(i, j, k), (l, m, n)$ in Index_table, $h(i, j, k) \neq h(l, m, n)$.

Therefore we do not need to spend time for resolving collisions.

In general, the size of hash value space is larger than the size of key space. That is, $h(i, j, k)$ might return an integer which is larger than M. Therefore, the size of table would be larger than M. Hence, some slots of key-value table would be empty. We want to avoid this situation because we are trying to reduce the storage requirement, so we use a minimal perfect hash function to avoid empty slots. With minimal perfect hashing, the return value of the hash function is guaranteed to be less than M without collision.

An excellent review of perfect hashing is in Czech et al.[9], from which we find the algorithm to implement minimal perfect hash function. We use an implementation from Jenkins [17]. It is a fast algorithm and the source is available.

For non-zero control points, no collision will occur as described above. However, if we hash the index of zero-valued control points, collision will occur. Therefore, we have to check if the index in the key-value table are the same as the input index. If they are not the same, it means that the index is not stored in the table and its value is omitted, i.e. 0 (See pseudo code in Section 6.3).

Unlike the control points for f_s, entire control points of f_d are stored without compression. In addition to the control points and key-value tables, we also need storage for a minimal perfect hash function. Some minimal perfect hashing techniques, as well as Jenkins', make use of auxiliary tables to realize hash functions. Although the size of these tables is small in general, they are not negligible. Thus the total storage for our new method includes control points for f_d, key-value table and the hash function.

In the next chapter, we evaluate the performance of our revised method. Hereafter, we refer our revised algorithm as Compact Bi-level Multiresolution B-Spline Algorithm (CBMBA) and the fit produced by CBMBA as CBMBA fit.

Chapter 7

Results of CBMBA

The objective of the new algorithm, CBMBA, described in Chapter 6 is to reduce the storage without sacrificing the performance of the fit. In this Chapter, we measure the storage requirements for new method and its effect on the accuracy, speed of fit and visual quality.

7.1 Quantitative results

7.1.1 Storage

The storage required for the CBMBA fit is shown in Table 7.1. We tabulated the results for the fit with diffuse levels of 3 and 4, and specular fits with levels 5 and 6. The omitting rates for the specular fit are 80, 90 and 95 %. Required total storage includes for control points of the diffuse fit f_d, the key-value table for f_s, and the hash function tables. We assume single precision floating point (4 bytes) to represent control points. With a specular fit of level 6, we can compress about 88 % from the original method if we omit 95 % of the control points for the specular fit.

Although CBMBA can compress greatly compared to the original fit, it is of no

	Specular Level 5			Specular Level 6		
Specular omitting rate	80 %	90 %	95 %	80 %	90 %	95 %
Diffuse Level 3						
Total storage (byte)	70,469	39,433	22,378	459,672	232,754	119,295
Total compression (%)	58.9	77.0	87.0	61.8	80.7	90.1
Diffuse Level 4						
Total storage (byte)	92,581	61,545	44,490	481,784	254,866	141,407
Total compression (%)	46.0	64.1	74.1	60.0	78.8	88.2

Table 7.1: Total Storage for CBMBA Fit in Byte and Total Compression Percentage from the Original Method.

use if the result does not perform well. We measured error metrics and also prepared the scatter plots as in Chapter 5.

7.1.2 Accuracy

Figure 7.1 shows the graph of error measurement. Numerical values of errors are tabulated in Table C.1.

To simplify our notation, we will use (D, S, C) to refer a CBMBA fit consisting of a level D diffuse fit and a level S specular fit with C % control point compression. For example, (4,6,95) denotes the fit with a level 4 diffuse fit and a level 6 specular fit with 95 % of its control points omitted. If we do not perform any compression, the error performance of the fit, the $(D, S, 0)$ fit, is as same as level S fit of original method. Therefore, the best error level in the graph is the same as the level 6 fit of the original method. As the graph shows, the result of the (4,6,95) fit is almost identical to the level 6 fit. Other fits are a bit behind from the (4,6,95) fit. From this result, we should use at least a level 4 diffuse fit to achieve an error level comparable to refinement level 6. If we do not need such an accurate fit, we could use a coarser diffuse fit, or a coarser specular fit to save storage.

Figure 7.2 shows comparison of error measurements with Koenderink and Lafor-

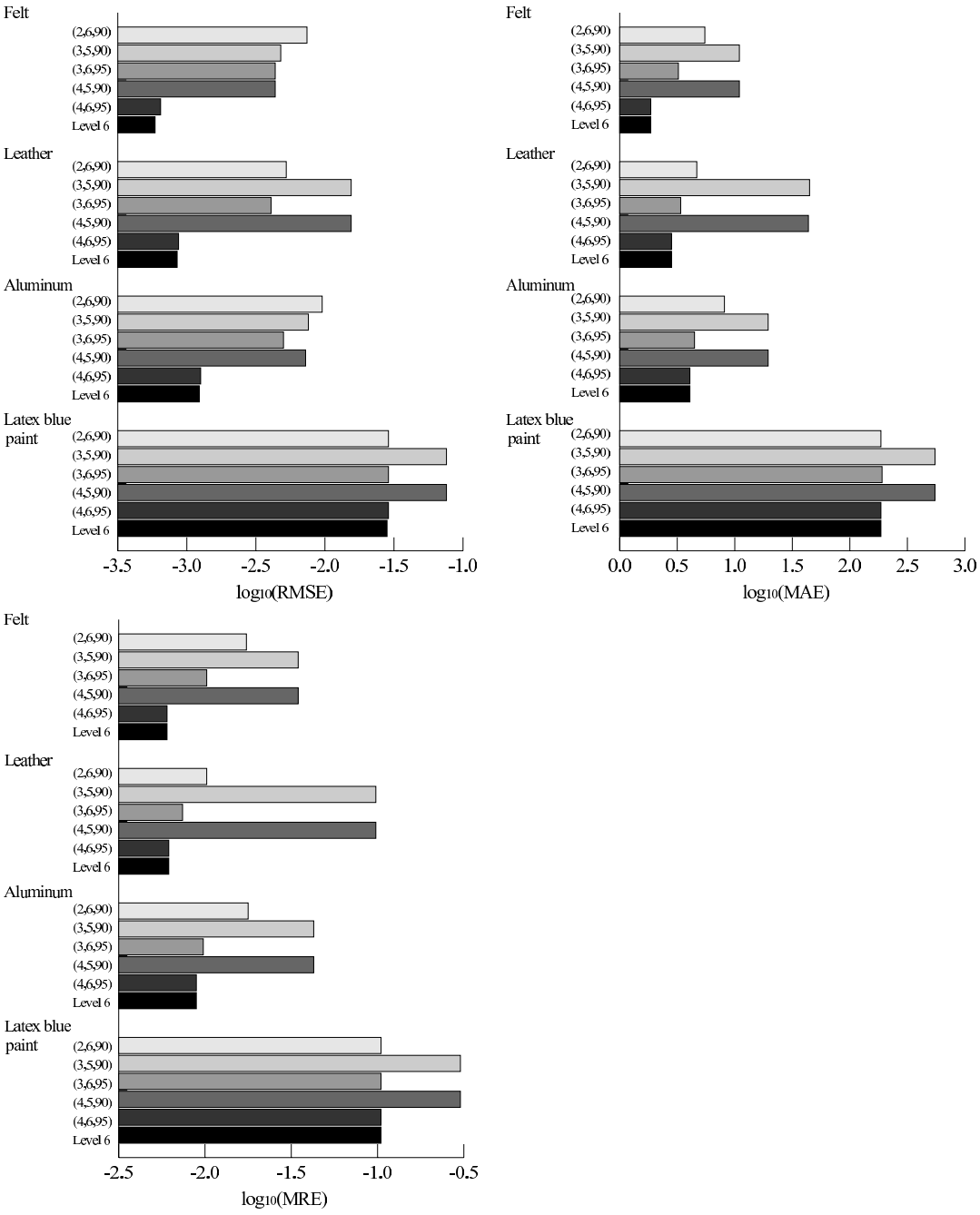


Figure 7.1: Error Measurement for CBMBA Fit. (D,S,C) in the graph indicates the levels and the compression rate of the fit. (diffuse level(D), specular level(S), percentage of control point omit(C)).

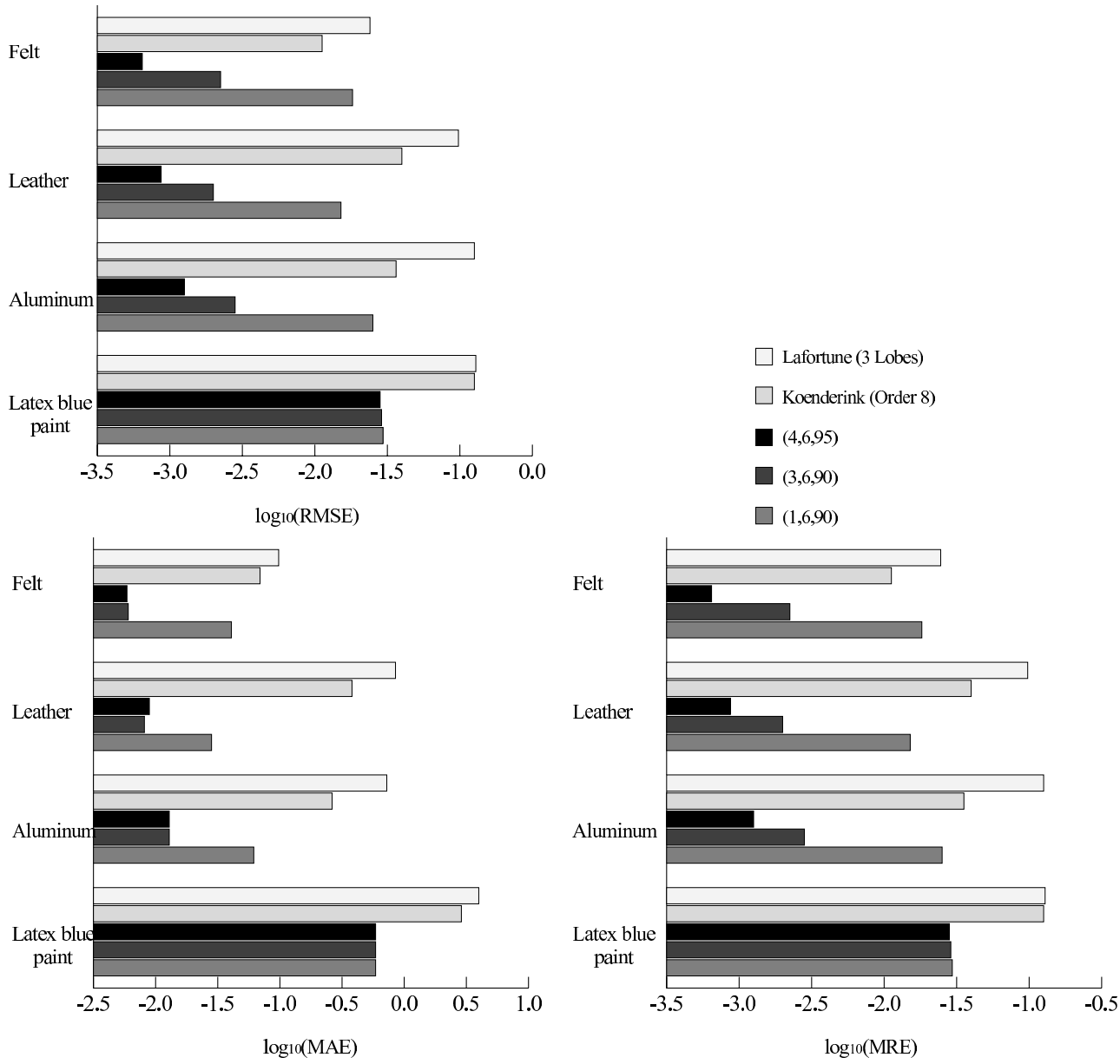


Figure 7.2: Error Comparison for CBMBA Fit.

tune models. The numerical values are tabulated in Table C.2. As we can see from Figure 7.2 and 7.1, some of our fits are not as accurate as original level 6 fit, but they are still better than other methods. Even in the lower diffuse level fit, our method can achieve the same

error level of other two methods.

Figure 7.3 and 7.4. show the scatter plots of CBMBA fits. For the (1,6,90) fit, the data around low values are scattered around, not on, the $x = y$ line, while other plots are almost on the line. However, the large values of the (1,6,90) fit are almost on the line. This means that the fit works for large values in the data set. This is because a specular fit can capture high specular values, but the diffuse fit does not fit the diffuse data well enough. In addition, too much information is removed by control point compression.

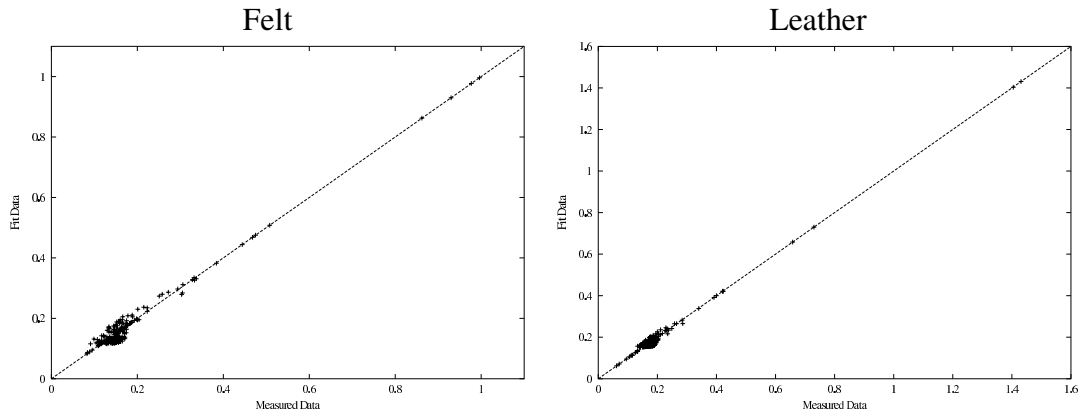
7.1.3 Speed

With CBMBA, we have to include the time needed to compute the hash function and evaluating the specular fit. In the same evaluation environment as in Section 5.1.2, it takes about 8 μsec for evaluating CBMBA fit. The time to compute the hash function is about $1.5 \times 10^{-2}\mu$sec. Compared with the time of original method, about 3 μsec, this is negligible. However, we have to evaluate two fits and have to check if the index is the same as the key in the key-value table. Hence, the evaluation time for CBMBA fit is a little bit more than twice that of the original method.

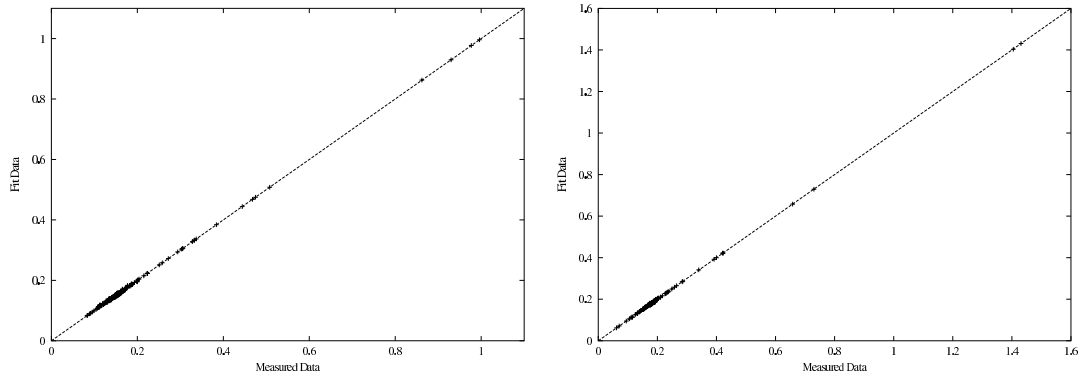
To obtain the same level of fit with the Koenderink method, order 8 of their model has to be used, and its evaluation time takes more than 200 μsec. Our method can still evaluate much faster than this, but is not faster than the Lafortune method, which is about 1 μsec.

7.2 Qualitative Results

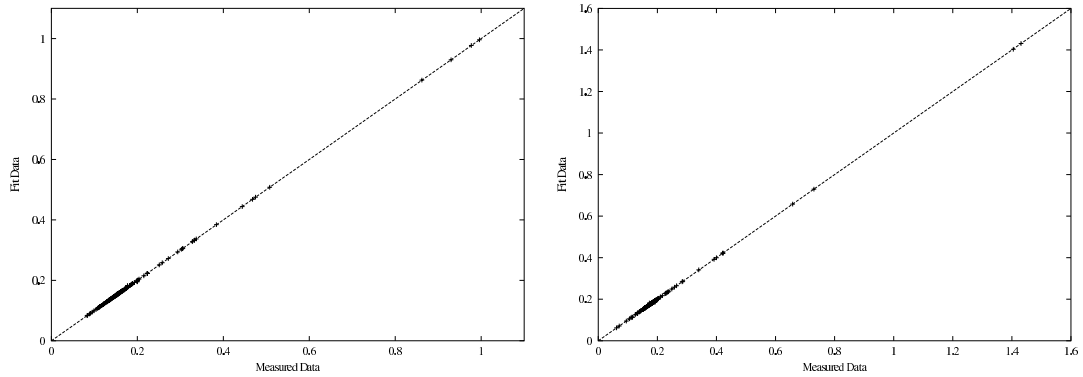
Next, we will show the visual quality of the CBMBA fit. CBMBA fits are plotted to show that they still maintain the smooth shapes. We also synthesize spheres as in Chapter 5. In



(1,6,90) fit



(3,6,90) fit



(4,6,95) fit

Figure 7.3: Scatter Plots of CBMBA Fit (1).

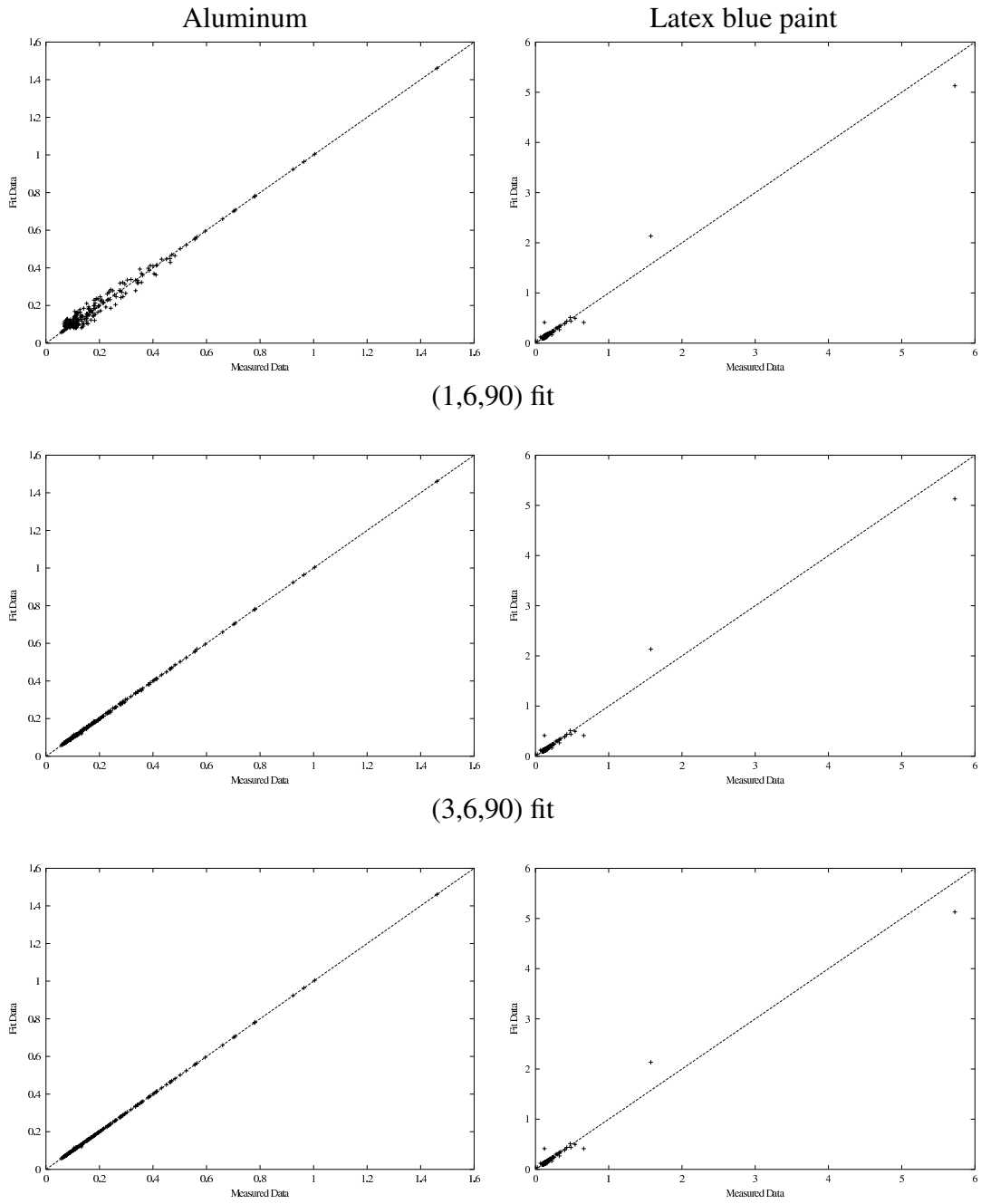


Figure 7.4: Scatter Plot of CBMBA Fit (2).

addition, we synthesize a more complex object, the Bunny, with a multi-channel BRDF.

7.2.1 Smoothness

Figure 7.5 and 7.6 show plots of CBMBA fit, along with their original fits of level 6. With a diffuse fit of level 1, there are some "ringing" where there is no data, although they fit the data well. As we mentioned in the discussion of scatter plots in Section 7.1.2, the level 1 fit could not capture the diffuse part well enough and this information is abandoned by compression, and it resulted in these artifacts. If we use a level 3 or finer level of diffuse fit, even if 95 % of compression of specular fit, no artifact is observed.

7.2.2 Images

Figure 7.7 shows synthesized spheres with a CBMBA fit. A point light source is located at 30° from the viewing position to the left. If we look at the images with the (1,6,90) fit, we can see a little artifact around the specular highlight. It is caused by ringing as the plot in Figure 7.6 shows, but other images show the same quality as the image with the original fit.

Finally, we synthesize a more complex object, the Stanford Bunny, with a multi-channel BRDF. The model was obtained from Stanford university's data archive [39]. We use BRDFs of red, green and blue data for felt, leather and aluminum. For latex blue paint, we took BRDFs of three wavelengths (700nm for red, 510nm for green and 480nm for blue). A point light source is used and rotated 30° to the right from the viewing position. We synthesized the bunny with the original fit and the (4,6,95) fit. All images shows the expected material appearances. We see no visible difference between the images with the original fit and with the CBMBA fit.

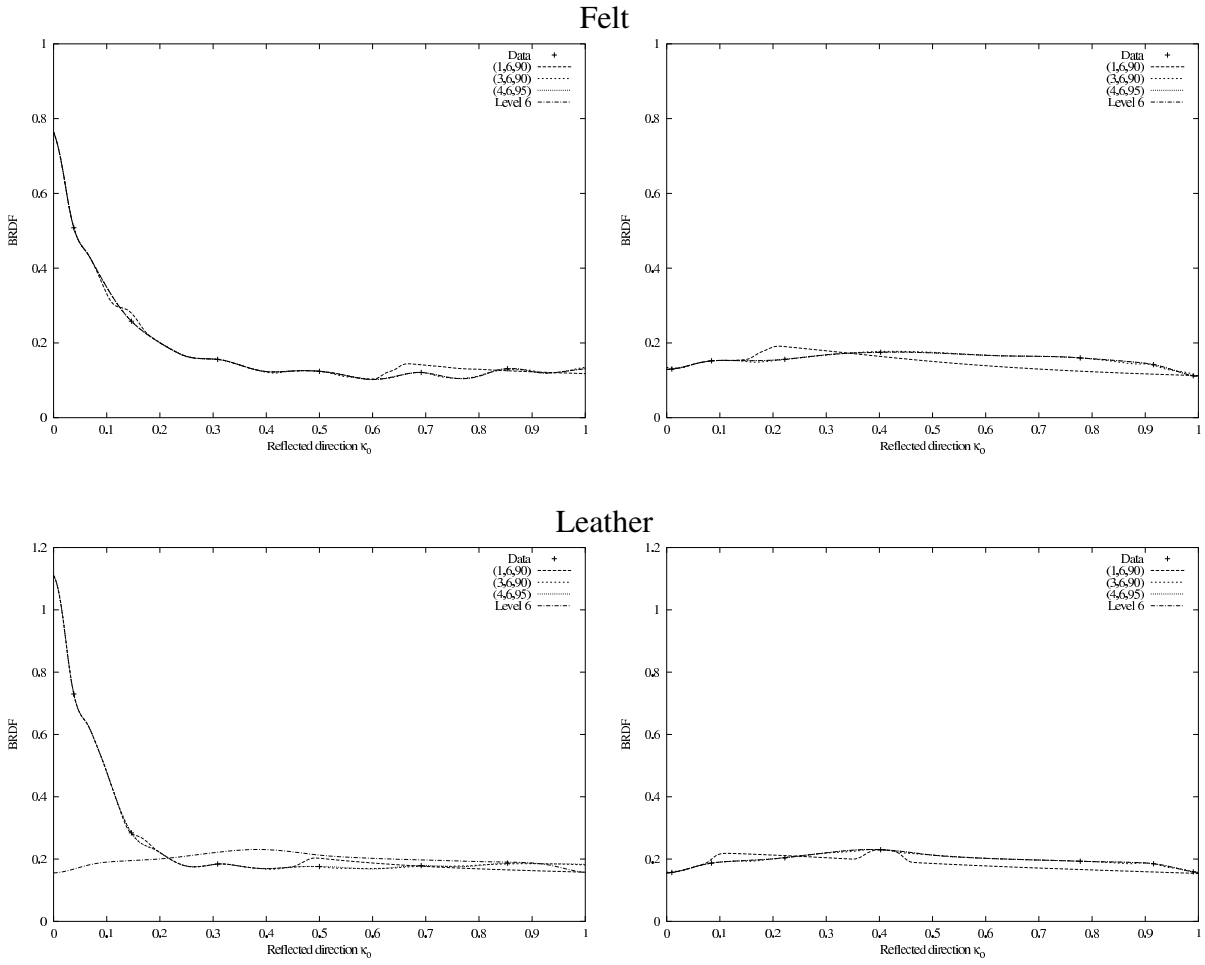


Figure 7.5: Plot for CBMBA Fit with Some Combination of Levels and Compression Percentage (1). Level 6 fit (no compression) is also shown.

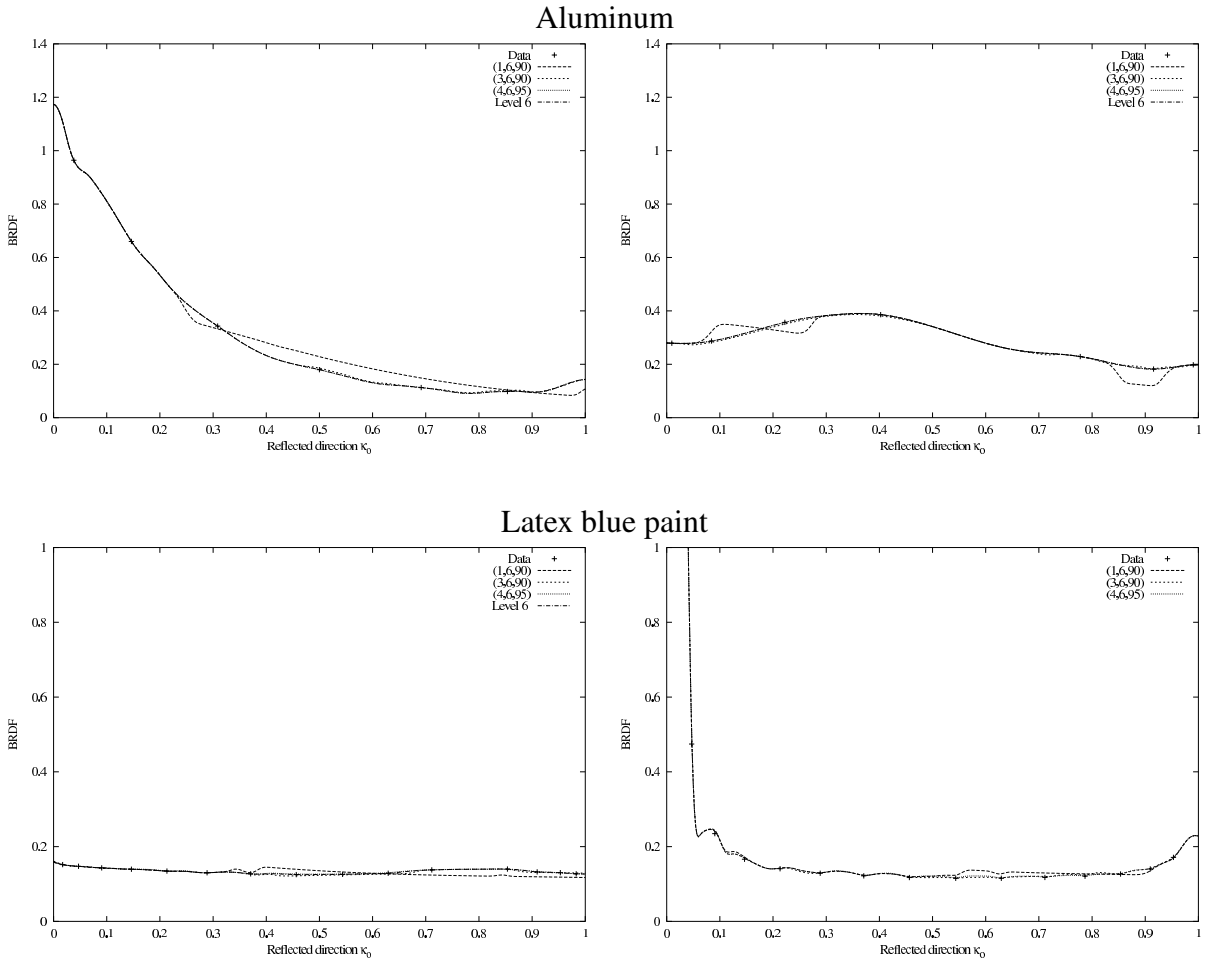


Figure 7.6: Plot for CBMBA Fit with Some Combination of Levels and Compression Percentage (2). Level 6 fit (no compression) is also shown.

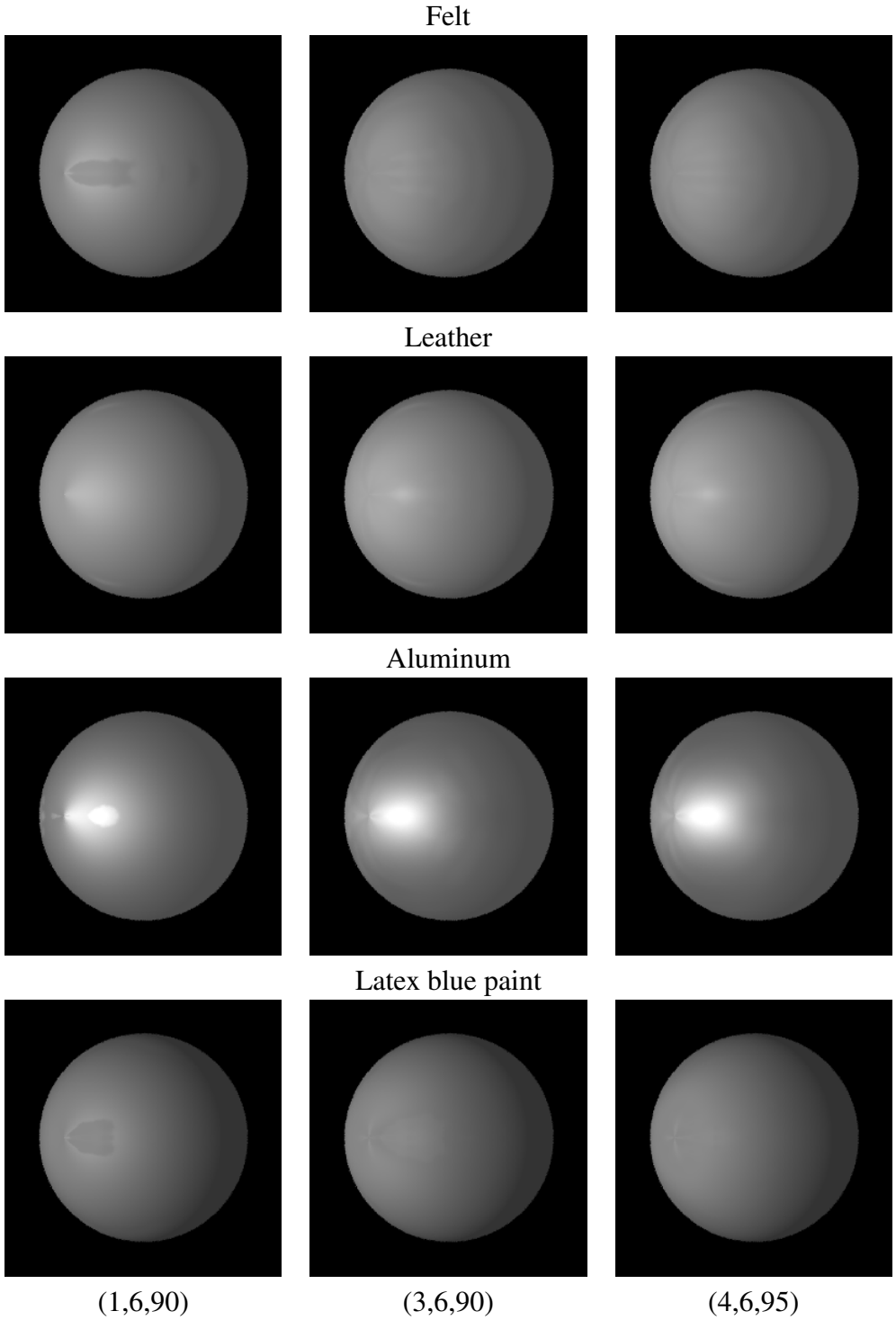
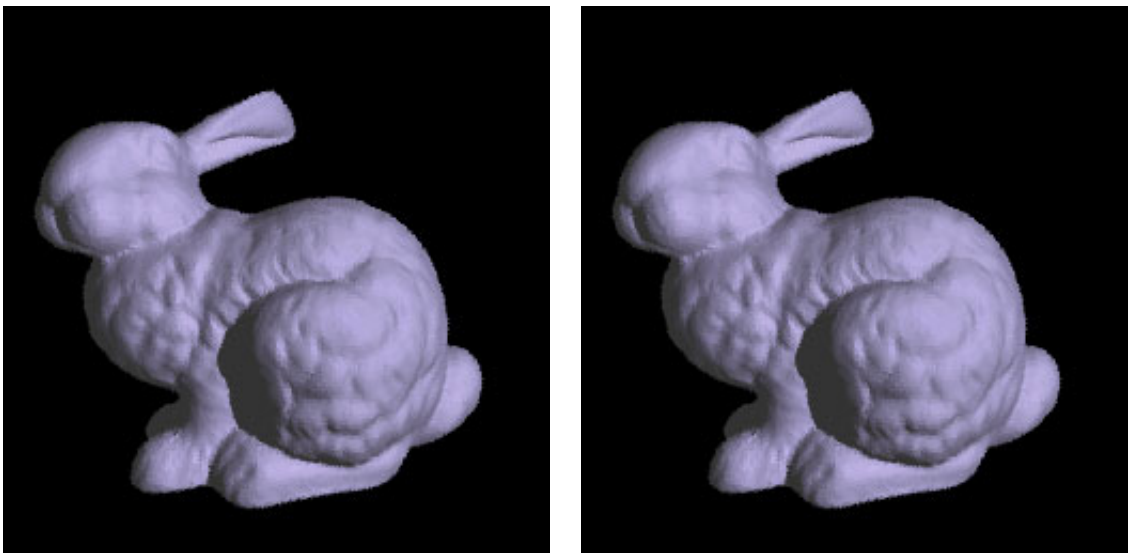


Figure 7.7: Synthesized Sphere using the CBMBA fit. The light source is located 30° to the left form the viewing position. Bottom row shows the (Diffuse level,Specular level,Compression percentage) of CBMBA fit.

Felt



Leather

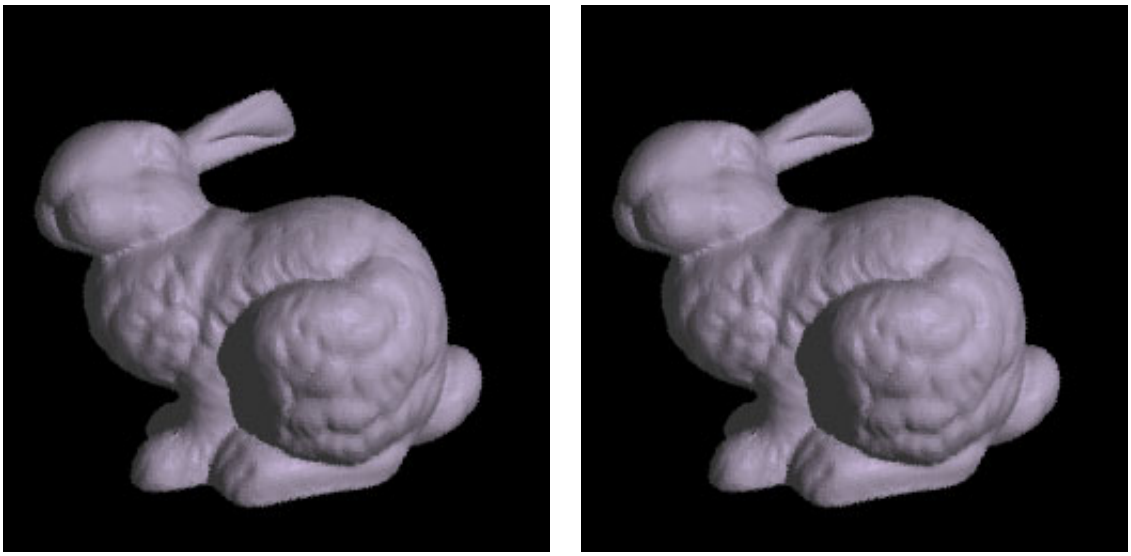
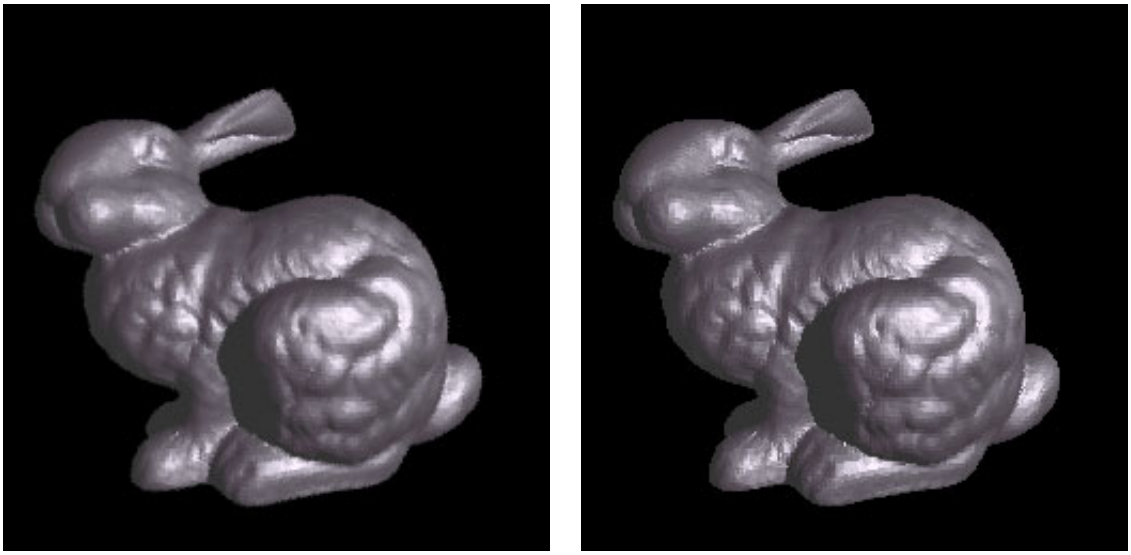


Figure 7.8: Synthesized Bunny with Original (left) and (4,6,95) (right) Fits.

Aluminum



Latex blue paint

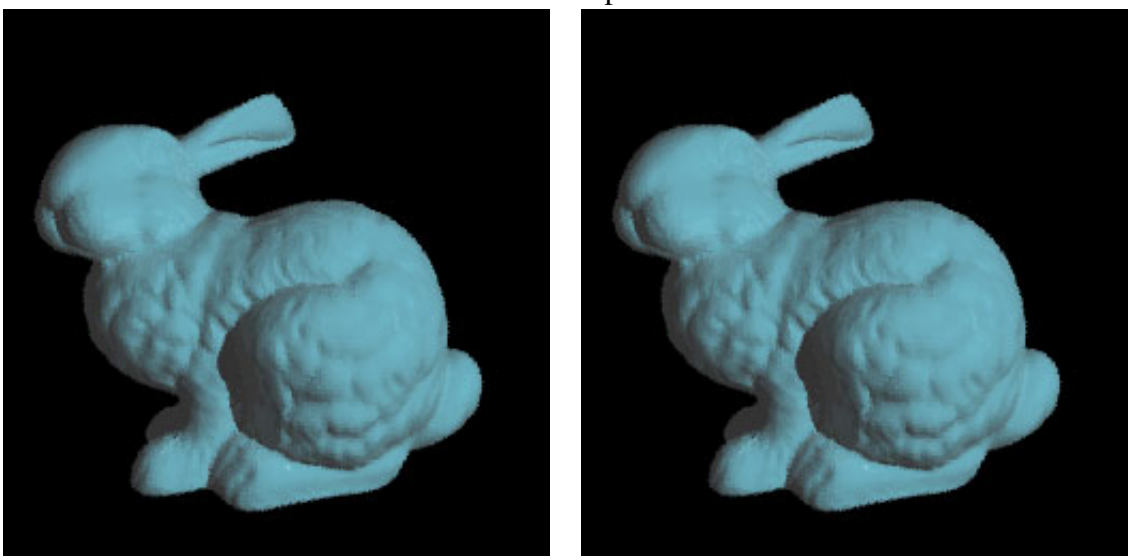


Figure 7.9: Synthesized Bunny with Original (left) and (4,6,95) (right) Fits.

Chapter 8

Conclusions

We presented a smooth, efficient representation of reflectance with a N-dimensional multi-level B-spline approximation. The quantitative and visual result of our simulation showed that the accuracy is quite good at level 6 fit and is much better than previous well-known methods. Even though the evaluation time of our method is not as fast as the fastest previous representation, it is comparable.

The disadvantage of our first method, compared to the others, was an excessive storage requirement. By decomposing a BRDF into diffuse and specular components, we successfully produced a composite fit (CBMBA fit). Then we compressed the storage by omitting the storage of specular control points whose values are small. We applied minimal perfect hashing to retrieve control points from compressed table. Because the time to compute the hash function is negligible, we can evaluate the CBMBA fit without major loss of time efficiency compared to the original method. From the results of the CBMBA fit, we can say that it produced a better representation than previous methods in terms of accuracy and ,to some extent, efficiency, even if we compress the storage by 90 % from the original fit.

In this thesis, we limited ourselves to use isotropic BRDFs. However, we can easily apply our method to anisotropic BRDFs.

In addition, our method can represent not only measured BRDF data, but also can represent any kind of reflectance.

As future work, we can extend this work to subsurface scattering (cf. [18]), and texture representation.

Appendix A

Examples of Reflectance Models

In this appendix, we will briefly describe the form of the reflectance models that we use for performance comparison with our method.

Koenderink Model

Koenderink et al.[20] constructed an orthonomal basis based on the Cartesian product of the hemisphere by mapping Zernike polynomials onto a unit disk. Then they project the BRDF into this vector space. Zernike polynomials appear to offer some advantages over spherical harmonics. In this section, we only describe their model for isotropic reflection. Their model can be written as:

$$\begin{aligned} f(\theta_s, \theta_v, \delta\phi) &= \sum_{nml} a_{nml} S^l_{nm}(\theta_s, \theta_v, \delta\phi) \\ &= \sum_{nml} (\Theta^l_n(\theta_s)\Theta^l_m(\theta_v) + \Theta^l_m(\theta_s)\Theta^l_n(\theta_v)) \cos(l\delta\phi), \end{aligned} \tag{A.1}$$

where $n \geq 0, 0 \leq m \leq nand0 \leq l \leq m.(n-l)$ and $(n-m)$ are even. The function $\Theta_n^l(\theta)$ is defined as:

$$\Theta_n^l(\theta) = \sqrt{\frac{n+1}{2\pi}} R_n^l(\sqrt{2}\sin\frac{\theta}{2}). \tag{A.2}$$

The function R_n^l is closely related to Jacobi's polynomial and is defined as:

$$R_n^{\pm l}(\tau) = \sum_{s=0}^{(n-l)/2} (-1)^s \frac{(n-s)!\tau^{n-2s}}{s!((n+l)/2-s)!((n-l)/2-s)!}. \tag{A.3}$$

We used order $n = 8$ of this model to compare with our method.

Lafortune Model

Lafortune et al. [21] represented a BRDF as a non-linear summation of powers of cosine lobes. It may be considered a generalization of Phong's original formula. This model can represent important BRDF behavior such as off-specular reflection and a retro-reflection with small number of parameters and it is currently a very popular reflectance model.

The model is defined as:

$$f_r(\mathbf{u}, \mathbf{v}) = \rho_s \left[C_x u_x v_x + C_y u_y v_y + C_z u_z v_z\right]^n, \tag{A.4}$$

where ρ_s is a value between 0 to 1, related to the reflectivity of the material, $\mathbf{u}$ is the incident direction, $\mathbf{v}$ is the reflected direction. C_x, C_y and C_z are the coefficients of this model, associated with the reflectance of a material, and n plays an similar role as in Phong's model, controlling the sharpness of the cosine lobe. It is not necessarily an integer. For isotropic reflection, we can set $C_x = C_y$. In practice, the reflectance is represented as a sum of (A.4). By absorbing the coefficients ρ_s into C_x, C_y and C_z, the reflectance is

written as:

$$f_r(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{k} \left[C_{x,i} u_x v_x + C_{y,i} u_y v_y + C_{z,i} u_z v_z \right]^{n,i}. \tag{A.5}$$

We use $k = 3$ lobes model of this model for comparison with our method.

Appendix B

Fitting Results

		Refinement Level (h)			Lafortune	Koenderink
		0	4	6	3 Lobes	Order 8
	Coefficients	64	6,859	300 ,763	10	55
Felt	RMSE	0.123	0.019	0.001	0.024	0.011
	MAE	0.751	0.109	0.060	0.098	0.070
	MRE	0.751	0.110	0.006	0.098	0.070
Leather	RMSE	0.140	0.041	0.001	0.099	0.040
	MAE	1.184	0.418	0.009	0.842	0.384
	MRE	0.828	0.292	0.006	0.589	0.268
Aluminum	RMSE	0.171	0.027	0.001	0.127	0.036
	MAE	1.146	0.277	0.013	0.716	0.264
	MRE	0.785	0.190	0.009	0.490	0.181
Latex blue	RMSE	0.184	0.113	0.029	0.128	0.125
Paint	MAE	5.554	3.340	0.595	4.016	2.905
	MRE	0.970	0.583	0.104	0.701	0.507

Table B.1: Error Measurement

		Refinement Level (h)			
		1	2	3	5
	Coefficients	125	343	1,331	42,875
Felt	RMSE	0.098	0.072	0.044	0.004
	MAE	0.902	0.491	0.268	0.035
	MRE	0.677	0.493	0.269	0.035
Leather	RMSE	0.125	0.105	0.074	0.015
	MAE	1.132	0.947	0.675	0.140
	MRE	0.791	0.662	0.471	0.098
Aluminum	RMSE	0.124	0.084	0.055	0.007
	MAE	1.006	0.782	0.545	0.063
	MRE	0.688	0.535	0.373	0.043
Latex blue paint	RMSE	0.180	0.172	0.154	0.076
	MAE	5.518	5.316	4.699	1.728
	MRE	0.964	0.929	0.821	0.583

Table B.2: Error Measurement for Level 1, 2 ,3 and 5.

Appendix C

CBMBA Results

		(2,6,90)	(3,5,90)	(3,5,95)	(4,5,90)	(4,6,95)
Felt	RMSE	0.007	0.005	0.004	0.004	0.001
	MAE	0.017	0.035	0.010	0.035	0.006
	MRE	0.018	0.035	0.010	0.035	0.006
Leather	RMSE	0.005	0.016	0.004	0.015	0.001
	MAE	0.015	0.140	0.011	0.140	0.009
	MRE	0.010	0.098	0.007	0.098	0.006
Aluminum	RMSE	0.010	0.008	0.005	0.007	0.001
	MAE	0.026	0.063	0.014	0.063	0.013
	MRE	0.018	0.043	0.010	0.043	0.009
Latex blue paint	RMSE	0.029	0.076	0.029	0.078	0.029
	MAE	0.595	1.728	0.596	1.728	0.595
	MRE	0.1040	0.302	0.104	0.302	0.104

Table C.1: Error Measurement for CBMBA Fit. Top row shows the combination of fit level and compression rate for the specular fit. (D, S, C) means the fit of level D diffuse fit, level S specular fit and its C % of control points are omitted.

		(1,6,90)	(3,6,90)	(4,6,95)	Lafortune(3)	Koenderink (8)
Felt	RMSE	0.018	0.002	0.001	0.020	0.011
	MAE	0.041	0.006	0.006	0.098	0.070
	MRE	0.041	0.006	0.006	0.098	0.070
Leather	RMSE	0.015	0.002	0.001	0.099	0.040
	MAE	0.028	0.008	0.009	0.842	0.384
	MRE	0.020	0.006	0.006	0.588	0.268
Aluminum	RMSE	0.025	0.003	0.001	0.127	0.036
	MAE	0.061	0.013	0.013	0.716	0.264
	MRE	0.042	0.009	0.009	0.490	0.181
Latex blue	RMSE	0.030	0.029	0.029	0.128	0.125
paint	MAE	0.595	0.595	0.595	4.016	2.905
	MRE	0.104	0.104	0.104	0.701	0.507

Table C.2: Error Measurement for CBMBA Fit : Compared with 3 lobes Lafortune model and order 8 Koenderink model.

Bibliography

[1] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers, 1986.

[2] J. F. Blinn. Models of light reflection for computer synthesized pictures. *Computer Graphics*, 11(2):192–198, July 1977.

[3] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *Computer Graphics (Proceedings of SIGGRAPH 81)*, 15(3):307–316, August 1981.

[4] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1):7–24, January 1982.

[5] T. H. Cormen, C.E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw Hill, 1989.

[6] Cornell reflectance data,http://www.graphics.cornell.edu/online/-measurements/reflectance/index.html.

[7] Curet reflectance and texture database, http://www.cs.columbia.edu/cave/curet/.

[8] Z. J. Czech, G. Havas, and B. S. Majewski. An optimal algorithm for generating minimal perfect hash functions. *Information Processing Letters*, 43(5):257–264, 1992.

[9] Z. J. Czech, G. Havas, and B. S. Majewski. Perfect hashing. *Theoretical Computer Science*, 182, 1997.

[10] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. Technical report, Columbia University, December 1996.

[11] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999.

[12] G. Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Morgan Kaufmann Publishers, 2001.

[13] S.C. Foo. A gonioreflectometer for measuring the bidirectional reflectance of materials for use in illumination computations. Master's thesis, Cornell University, Ithaca, New York, July 1997.

[14] A. Fournier. Separating reflection functions for linear radiosity. *Eurographics Rendering Workshop 1995*, pages 296–305, June 1995.

[15] X. D. He, K. E. Torrance, F. X. Sillion, and D. P. Greenberg. A comprehensive physical model for light reflection. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):175–186, July 1991.

[16] W. Heidrich. Private communication, 2002.

[17] B. Jenkins. Algorithm alley : Hash functions. *Dr. Dobb's Journal*, September 1997.

[18] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. *Proceedings of SIGGRAPH 2001*, pages 511–518, August 2001.

[19] D. E. Knuth. *The Art of Computer Programming vol 3. / Sorting and Searching*. Addison Wesley, 1997.

[20] J. J. Koenderink, A. J. van Doorn, and M. Stavridi. Bidirectional reflection distribution function expressed in terms of surface scattering modes. *Proc. 4th European Conference on Computer Vision*, pages 28–39, 1996.

[21] E. P. F. Lafortune, S. C. Foo, K. E. Torrance, and D. P. Greenberg. Non-linear approximation of reflectance functions. *Proceedings of SIGGRAPH 97*, pages 117–126, August 1997.

[22] J.H. Lambert. Photometria sive de mensure de gratibus luminis, colorum umbrae. *Eberhard Klett*, 1760.

[23] G. J. Ward Larson. Measuring and modeling anisotropic reflection. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):265–272, July 1992.

[24] S. Lee, K.-Y. Chwa, S.Y. Shin, and G. Wolberg. Image metamorphosis using snakes and free-form deformations. *Computer Graphics (Proc. SIGGRAPH '95)*, 1995.

[25] S. Lee, G. Wolberg, and S. Y. Shin. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244, July - September 1997.

[26] R. R. Lewis. Making shaders more physically plausible. *Computer Graphics Forum*, 13(2):109–120, January 1994.

[27] R. R. Lewis and A. Fournier. Wavelet radiative transfer and surface interaction. *Computer Graphics Forum*, 19(2):135–151, June 2000.

[28] S.R. Marschner, S.H. Westin, E.P.F. Lafortune, K.E. Torrance, and D.P. Greenberg. Image-based brdf measurement including human skin. *Eurographics Rendering Workshop 1999*, June 1999.

[29] M. D. McCool, J. Ang, and A. Ahmad. Homomorphic factorization of brdfs for high-performance rendering. *Proceedings of SIGGRAPH 2001*, pages 171–178, August 2001.

[30] M. Oren and S.K. Nayar. Generalization of the lambertian model and implications for macine vision. *International Journal of Computer Vision*, 14:227–251, 1995.

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

[32] W. H. Press, S. A. Eukolsky, and B. P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. The Press Syndicate of the University of Cambridge, 1992.

[33] D.F. Rogers and J. A. Adams. *Mathematical Elements for Computer Graphics*. McGraw Hill, 1990.

[34] C. Schlick. A customizable reflectance model for everyday rendering. *Fourth Eurographics Workshop on Rendering*, pages 73–84, June 1993.

[35] D. C. Schmidt. Gperf: A perfect hash function generator. *Second USENIX C++ conference Proceedings*, April 1990.

[36] P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. *Proceedings of SIGGRAPH 95*, August 1995.

[37] P. Shirley, H. Hu, B. Smits, and E. P. Lafortune. A practitioners' assessment of light reflection models. *Pacific Graphics '97*, October 1997.

[38] F. X. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg. A global illumination solution for general reflectance distributions. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):187–196, July 1991.

[39] Stanford computer graphics laboratory data archives, http://www.graphics.stanford.edu/data.

[40] S. H. Westin, J. R. Arvo, and K. E. Torrance. Predicting reflectance functions from complex surfaces. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):255–264, July 1992.