Offset Surface Light Fields

by

Jason Ang

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Technical Report Number CS-2003-22

Waterloo, Ontario, Canada, 2003

©Jason Ang 2003

Author’s Declaration for Electronic Submission of a Thesis

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

For producing realistic images, reflection is an important visual effect. Reflections of the environment are important not only for highly reflective objects, such as mirrors, but also for more common objects such as brushed metals and glossy plastics. Generating these reflections accurately at real-time rates for interactive applications, however, is a difficult problem. Previous works in this area have made assumptions that sacrifice accuracy in order to preserve interactivity.

I will present an algorithm that tries to handle reflection accurately in the general case for real-time rendering. The algorithm uses a database of prerendered environment maps to render both the original object itself and an additional bidirectional reflection distribution function (BRDF). The algorithm performs image-based rendering in reflection space in order to achieve accurate results. It also uses graphics processing unit (GPU) features to accelerate rendering.

Acknowledgements

Thank you Professor Michael McCool for your support, guidance, and encouragement. Your knowledge makes me question why I don't know more and your enthusiasm motivates me to learn more. This thesis would not have been possible without your help. Also thanks to Professors William Cowan and Richard Mann for taking time from their busy schedules to read my thesis and provide valuable comments.

Thanks to the professors and students of the Computer Graphics Lab for providing a fun and social learning environment. Special thanks to Kevin Moule for giving me pointers on software and other issues. I'm also indebted to those who have contributed to my thesis in ways they or I may not be aware.

To my cheerleaders and financial investors, I am forever indebted to you all. I hope I can return the favour someday.

This thesis uses models from Anis Ahmed, Kevin Moule, Martin Newell, Mira Imaging, and Viewpoint Engineering. Paul Debevec provided the eucalyptus grove high-dynamic range light probe. Furthermore, my colour conversion algorithm was based on Mark Ruzon's source code.

This work was funded by the Ontario Graduate Scholarship Program (OGS), Professor McCool's Communications and Information Technology Ontario (CITO) and Natural Sciences and Engineering Research Council of Canada (NSERC) grants, and the University of Waterloo.

Dedication

For my family and Elsa, for their unconditional love and support.

Trademarks

3Dlabs is a registered trademark of 3Dlabs, Inc., Ltd.

Adaptec is a registered trademark of Adaptec, Inc.

AIC is a trademark of Adaptec, Inc.

Alias|Wavefront is a registered trademark of Alias|Wavefront, a division of Silicon Graphics Limited.

Apple Computer is a registered trademark of Apple Computer, Inc.

ATI is a registered trademark of ATI Technologies, Inc.

Atlas is a trademark of Quantum Corporation.

BMRT is a registered trademark of Larry Gritz.

Casino de Monte-Carlo is a registered trademark of Société de Bains de Mer et du Cercle des Etrangers à Monaco, Limited.

DirectX is a registered trademark of Microsoft Corporation.

Exluna is a registered trademark of Exluna, Inc.

GeForce is a registered trademark of NVIDIA Corporation.

Intel is a registered trademark of Intel Corporation.

Intel Xeon is a registered trademark of Intel Corporation.

Maya is a registered trademark of Silicon Graphics, Inc., exclusively used by Alias|Wavefront, a division of Silicon Graphics Limited, and Maya Unlimited is a trademark of Alias|Wavefront, a division of Silicon Graphics Limited.

Microsoft is a registered trademark of Microsoft Corporation.

NVIDIA is a registered trademark of NVIDIA Corporation.

NVIDIA Cg is a registered trademark of NVIDIA Corporation.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Pixar is a registered trademark of Pixar Corporation.

Quantum is a registered trademark of Quantum Corporation.

QuickTime is a registered trademark of Apple Computer, Inc.

RenderMan is a registered trademark of Pixar Corporation.

RenderMonkey is a registered trademark of ATI Technologies, Inc.

Wildcat is a registered trademark of 3Dlabs, Inc., Ltd.

All other company and product names mentioned are trademarks or registered trademarks of their respective owners.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Calculating the colour of an object at each point on its surface is an important part of achieving photo-realistic images. In the rendering process, shading is responsible for performing this calculation. Shading determines the light leaving the surface of an object as a function of the light striking it. There are many variables which affect how this calculation is performed but the most important factor in photo-realistic rendering is the object's bidirectional reflectance distribution function or BRDF. Essentially, this function determines the object's material properties—whether the object looks like a fabric, a matte wall, or a metal is all encapsulated in this function. Calculating this function accurately and integrating it against the incoming light at each point is vital in achieving photo-realistic images.

Reflection is one visual effect factored into an object's BRDF. Varying degrees of this effect can be seen all around us: from sharp reflections in mirrors to blurry reflections in glossy plastics to more subtle and diffused reflections in brushed metals, reflections play an important role in how we perceive everyday objects. Incorporating accurate reflections is therefore necessary to produce physically accurate images.

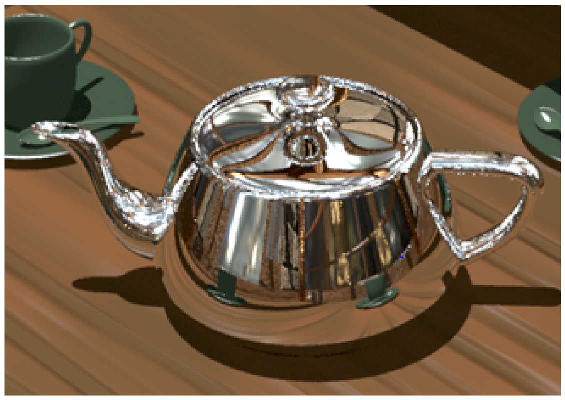
Traditionally, offline rendering techniques, such as ray tracing, have excelled at producing photo-realistic images which incorporate reflections and other complex shading [7, 40, 52, 104, 118]. However, recent works in real-time rendering [6, 50, 78, 82, 91, 93, 94] have closed the gap and brought photo-realistic rendering into the realm of interactive applications. These applications usually employ specialized [4, 31, 86, 88, 111] or consumer grade [2, 34, 68, 87, 113] graphics processor units (GPUs) to accelerate their rendering processes. The advent of the GPU also ushered in a whole new class of graphics algorithms which use the hardware in unique and unusual ways [95, 117]. However, common to these approaches are that they all solve or approximate the rendering equation [55] in some way.

Compared to their slower, offline rendering cousins, real-time hardware-accelerated algorithms have improved the workflow in many industries. For example, architectural, automotive, and industrial designers are able to interactively manipulate and display their designs for clients without the long turnover times associated with generating animations. Furthermore, directors and cinematographers are able to previsualize their camera shots quickly. An example of this is given by *A.I. Artificial Intelligence*, where the blue-screen shots in the Rouge City scenes were visualized in real-time for the director, so he could see how the actors interacted with the environment. Finally, game designers can increase the realism of their worlds without sacrificing interactivity. In many application areas, it is often desirable to have real-time feedback for users. Using a hardware accelerator can result in a speedup of 10 to 1000 times depending on how well the algorithm matches the capabilities of the accelerator.

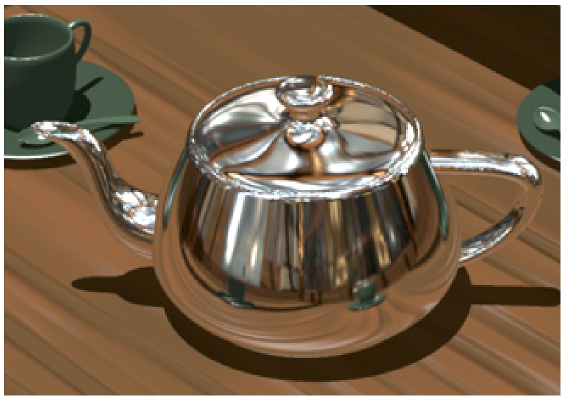
When migrating software algorithms to hardware-accelerated implementations, usually compromises have to be made; this typically comes in the form of sacrificing accuracy for interactivity. The two main sources of error are errors inherent in the hardware rendering pipeline and errors due to the rendering algorithm. An example of the former is accuracy error due to hardware numerical precision. Besides improving the hardware, there is not much that can be done in this case. However, improvements to the rendering algorithm itself can be made.

In terms of shading, software based ray tracing approaches [40, 124] can generate very accurate inter-object reflections such as in Figure 1.1(a) without difficulty. Here, the software shoots a ray through the image plane of the camera. If the ray intersects a specular object, another ray is generated in the reflection direction and this process continues recursively until the ray hits a diffuse object. Finally, the results of all the hits are accumulated to determine the first object's final colour. Although hardware implementations of ray tracing algorithms are available [45], they are not geared for real-time display. Recently, ray tracing of static scenes has been demonstrated on a GPU simulator [95] which is based on proposals from upcoming API specifications [1, 76]. This work uses the CPU for preprocessing and still has major memory consumption and performance issues to resolve.

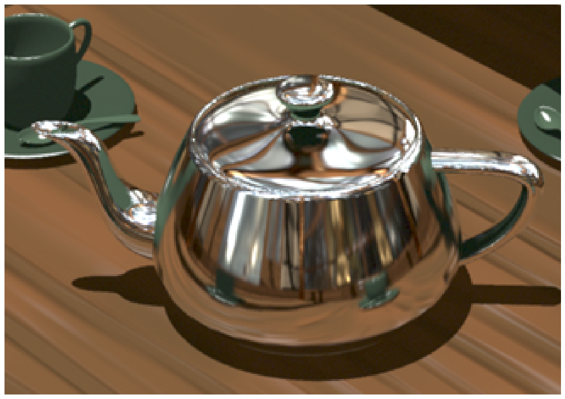
For interactive display, hardware designers have implemented reflection mapping (also known as environment mapping) [13] into their GPUs. This technique, developed by Blinn and Newell, captures the light flow around a chosen centre of projection onto a spherical reflection map. This map is then indexed by the reflection vector off the object's surface to produce specular reflection. Reflection mapping, however, breaks down



(a) Ray trace



(b) Environment map



(c) Offset surface light field

Figure 1.1: Specular component of a highly reflective surface rendered using three methods.

for non-convex objects and for environments with near-field objects. Figure 1.1(b) shows the deficiencies of this approach—the reflections are inaccurate, blurry, and there is no object self-reflection. Nevertheless, for some applications the advantage of interactive display outweighs the accuracy of the result and so environment map based approaches are commonly used today.

The deficiencies of environment mapping and related approaches motivated me to investigate ways of obtaining more accurate results while preserving the real-time dis-

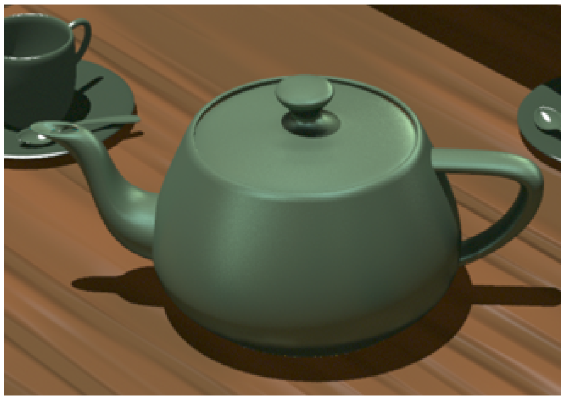


Figure 1.2: Ashikhmin and Shirley's BRDF model rendered using offset surface light fields.

play component. Any solution to this problem should visibly improve the accuracy in highly reflective surfaces—where the problem is most prominently evident—and also be extendible to the case of more general glossy BRDFs, such as in Figure 1.2. Finally, the algorithm should be amenable to a high-performance hardware implementation.

1.1 Major Thesis Contributions

I will present an algorithm which tries to handle reflection accurately in the general case for real-time rendering. The algorithm blends together concepts from reflection mapping and image-based rendering to achieve its accuracy. It is capable of accurately reproducing the original object—composed of curved surfaces—from its database of light fields. Furthermore, it can generate reflections which are qualitatively and quantitatively more accurate than environment map based approaches. In particular, the new technique permits reflections which are dependent on the position in space of the surface point being shaded. The solution can be generalized to handle arbitrary BRDFs for local illumination. Finally, the algorithm is evaluable at runtime, amenable to a hardware-accelerated implementation, and is texture cache friendly.

As this thesis improves real-time reflections, previous work using the environment map approach can take advantage of it to improve accuracy. Furthermore, this thesis opens up many possible extensions to the class of rendering algorithms possible within the offset surface light field paradigm—some of which are listed in Section 8.1.

1.2 Thesis Organization

The remainder of the thesis is organized as follows: Chapter 2 reviews relevant work upon which my research draws, including reflection maps and light fields. Chapter 3 introduces offset surface light fields and demonstrates how to reconstruct the original object accurately. Chapter 4 describes how specular reflections are generated and Chapter 5 generalizes this for evaluating local illumination. The results are discussed in detail in Chapter 6 and Chapter 7 then describes how the algorithm can be targeted to hardware. Finally, Chapter 8 summarizes the results and proposes future research topics to pursue.

Chapter 2

Background

My thesis draws from two areas of research: reflection mapping and image-based rendering. I will start by reviewing the theoretical framework in both areas before progressing onto a survey of related work. This chapter concludes by putting this thesis in perspective relative to prior work.

2.1 Reflection Mapping

Reflection mapping based approaches are best understood and formalized within the context of the rendering equation.

2.1.1 The Rendering Equation

The rendering equation, first described by Kajiya [55] in graphics literature, borrows from radiative heat transfer and neutron transport literature and has been adapted for use in computer graphics. The rendering equation describes how light is scattered off an object's surface and can be used to characterize many known rendering algorithms. Although it is an approximation, neglecting for example wave effects, it has proven useful to formalize the rendering process in computer graphics. Omitting occlusions and using notation from Kautz et al. [60], the rendering equation is:

$$L(\mathbf{x};\vec{v}) = L_e(\mathbf{x};\vec{v}) + \int_{\Omega} f_r(\vec{\omega}(\vec{v},\vec{n},\vec{t}),\vec{\omega}(\vec{l},\vec{n},\vec{t}))L_i(\mathbf{x};\vec{l})\langle\vec{n},\vec{l}\rangle d\vec{l}, \tag{2.1}$$

where L is the reflected exitant radiance in the viewing direction $\vec{v}$ from point $\mathbf{x}$ (with coordinate frame $\{\vec{n},\vec{t},\vec{n}\times\vec{t}\}$), L_e is the outgoing emitted radiance from point $\mathbf{x}$ in direction $\vec{v}$, f_r is the surface's BRDF, $\vec{\omega}(\vec{v},\vec{n},\vec{t})$ is the viewing direction and $\vec{\omega}(\vec{l},\vec{n},\vec{t})$ is the light direction with respect to the given coordinate frame, and L_i is the incoming radiance at $\mathbf{x}$ from direction $\vec{l}$ (Figure 2.1). The integration is performed for all direc-

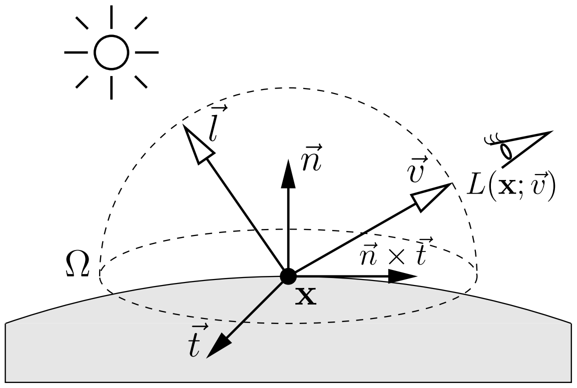


Figure 2.1: Geometry of local illumination.

tions $\vec{l}$ over the hemisphere Ω above $\mathbf{x}$. Equation 2.1 further assumes that the BRDF is shift-invariant (i.e. it does not depend on surface position) otherwise we will have to incorporate the dependence on $\mathbf{x}$ into f_r. The discussion that follows will also assume there is no emission term (i.e. $L_e = 0$) and omit wavelength dependence (this equation is typically evaluated only for the required wavelengths—usually red, green, and blue). Derivation details can be found in Kajiya [55] or Foley et al. [36]. The rendering equation basically says that the outgoing radiance is equal to the emitted radiance added to the sum of all incoming radiance (over the hemisphere) modulated with the BRDF projected onto the surface with differential solid angle $d\vec{l}$.

The BRDF, f_r, plays an important role in an object's appearance. It describes how light is reflected when it comes in contact with various materials and is a function of the incoming light direction and the outgoing reflected light direction. When these two vectors are expressed in spherical coordinates, with respect to the local coordinate frame of the point, the BRDF becomes a four-dimensional function:

$$f_r(\vec{\omega}(\vec{v},\vec{n},\vec{t}),\vec{\omega}(\vec{l},\vec{n},\vec{t})) \equiv f_r(\theta_{\vec{v}},\phi_{\vec{v}},\theta_{\vec{l}},\phi_{\vec{l}}). \tag{2.2}$$

To be physically accurate, f_r must satisfy the first law of thermodynamics (conservation of energy) and Helmholtz's reciprocity principle [122] for mirrors or Rayleigh's reciprocity principle [99] for arbitrary surfaces. BRDFs are divided into two classes: isotropic and anisotropic. The former class has reflectance properties which are invariant with respect to surface rotation around the normal vector (e.g. smooth plastics) whereas the latter class has reflectance properties which change with respect to surface rotation around $\vec{n}$ (e.g. brushed metal and satin). Although tabulated BRDFs have been used, the high dimensionality of the function has caused many researchers to search for more compact representations. One such approach is to represent the BRDF as an computable arithmetic function and many such analytic BRDFs were developed to model specular reflection [8, 10, 12, 16, 27, 46, 54, 90, 102, 115, 116, 123]. However, unless the function was simple, the BRDF could not be evaluated efficiently for real-time display. For interactive purposes, three main approaches have been developed: basis summation approaches represent the BRDF as a sum of simpler basis functions [17, 61, 123]; environment mapping approaches preintegrate the BRDF with the lighting environment [18, 58, 60]; and factorization approaches represent the BRDF as lower-dimensional functions which are added or multiplied together [37, 50, 57, 79].

The rendering equation and BRDFs model purely local illumination; real world effects such as volumetric scattering and fluorescence are not modelled. In spite of this, the rendering equation still serves as a good starting point. Even so, evaluating Equation 2.1 accurately would be too computationally intensive and thus many approximation methods have been investigated. Ray tracing [40, 124], as described in Chapter 1, is ill-suited for reproducing the complex effects encapsulated in the equation. Distribution ray tracing [26] extends ray tracing by casting multiple rays at each reflection point. By further perturbing the rays to have a distribution function which represents the surface's reflection properties, this method better approximates the integral. However, as many rays are needed to produce good results, this method is still too computationally intensive. Path tracing [55] traces random paths of rays from the camera into the scene to sample outgoing radiance. By accumulating multiple samples with the principles of Monte Carlo integration [105, 119], the renderer can obtain an estimate for the integral. Monte Carlo integration will be discussed further in Section 5.1. Light tracing [32], on the other hand, traces rays from the lights to the camera. This approach, however, suffers from uneven sampling of the image plane which decreases its accuracy. Bidirectional path tracing [62, 120] casts rays from both the camera and the lights and later connects the rays together. Although this method is less expensive and has lower error compared to distribution ray tracing, it produces very noisy images, and cannot in fact sample all forms of light transport. This method was later improved by Jensen and Christensen's photon mapping [52, 53]. Metropolis light transport [121] further improves error and noise by taking a path from a light to the camera and perturbing it to see if it still reaches the camera. The advantages it achieves, however, are offset by its complex implementation. All these methods are offline software based algorithms and to date there is no way to evaluate Equation 2.1 accurately for general scenes for real-time display. Path tracing, for instance, requires hundreds of samples per pixel and can take hours to render even a simple scene.

2.1.2 Environment Mapping

For simple BRDFs and single objects or simple environments, environment mapping based approaches can approximate Equation 2.1 at interactive display rates. This technique was first introduced by Blinn and Newell [13] to approximate mirror reflections.

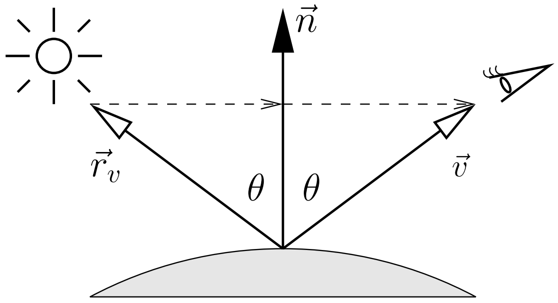


Figure 2.2: Geometry of reflection.

For a purely reflective surface, such as an ideal mirror, the outgoing light to the camera in direction $\vec{v}$ is equal in intensity to the incoming light from direction $\vec{r}_v$. Here, $\vec{r}_v$

is calculated by reflecting $\vec{v}$ in the surface normal, $\vec{n}$ (Figure 2.2). In functional notation,

$$L(\mathbf{x}; \vec{v}) = L_i(\mathbf{x}; \vec{r}_v) \tag{2.3}$$

where

$$\vec{r}_v = 2\vec{n}\langle\vec{n}, \vec{v}\rangle - \vec{v}. \tag{2.4}$$

From Equation 2.3 we can observe that calculating the reflected radiance would be greatly sped up if we could index into a mapping with the reflection vector to obtain the reflected radiance. This map can be created by choosing a centre of projection at $\mathbf{x}$ (usually the geometric centre of the reflective object) and projecting the environment onto it. Such a map is commonly referred to as a spherical environment map. To be physically accurate, this map should use high-dynamic range radiance values. This map is then indexed by spherical projection relative to the view vector (Figure 2.3). Another

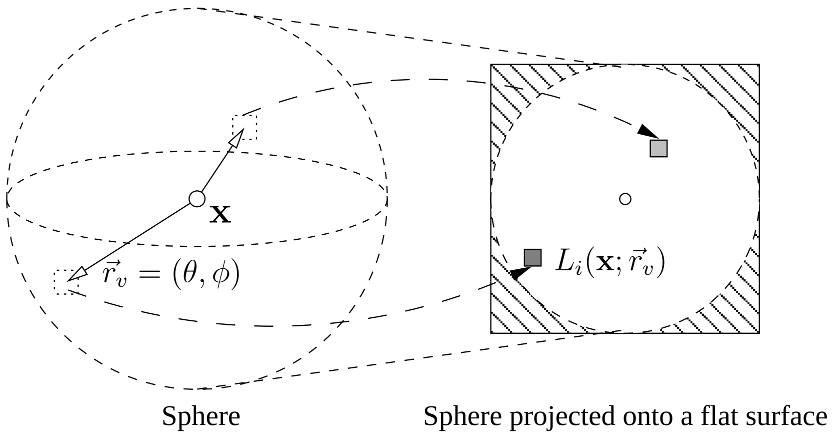

$\mathbf{x}$
$\vec{r}_v = (\theta, \phi)$
$L_i(\mathbf{x}; \vec{r}_v)$
Sphere
Sphere projected onto a flat surface

Figure 2.3: Geometry of environment maps.

possible parameterization of this map is given by latitude-longitude maps [118]. However, both these parameterizations are view-dependent: they are valid only for the view for which they were originally generated. Two view-independent environment map parameterizations are dual parabolic maps [49] and cube maps [43]. Cube maps, which capture the environment using six faces of an enclosing cube, have since been standardized into hardware [92] which allow for efficient look ups.

Environment mapping assumes the enclosing environment is infinitely far from the rendering surface. Therefore, it is only valid for environments with distant objects and lights. Furthermore, planar and concave surfaces present problems because the map is only indexed by $\vec{r}_v$. Planar surfaces can be dealt with using multi-pass techniques [36]. Concave surface problems can be partially alleviated by using the surface position and the centre of projection to determine a corrected reflection direction. However, problems are still visible when compared to accurate ray traced reflections. Comparing Figures 1.1(a) and 1.1(b), distortion is apparent in the roof beams and the table texture; furthermore, the teacups in the environment mapped version appear more distant. Moreover, this approach does not capture the teapot spout, handle, and lid self-reflecting onto the body. Regardless of their shortcomings, environment maps are used widely for non-critical applications, such as games, because of their readily available hardware-accelerated implementations.

Since environment maps capture the lighting around a point, researchers noticed that in some circumstances they could perform the integration in Equation 2.1 beforehand, generating a prefiltered environment map for that point, and then use the GPU to accelerate rendering. For instance, using a purely diffuse BRDF (also known as the Lambertian BRDF),

$$f_r(\vec{v},\vec{l}) := k_d, \tag{2.5}$$

where k_d describes the surface's absorption, Miller and Hoffman [85] perform the following preintegration:

$$\begin{aligned} L_{diffuse}(\mathbf{x}; \vec{v}, \vec{n}, \vec{t}) &= \int_{\Omega} k_d L_i(\mathbf{x}; \vec{l}) \langle \vec{n}, \vec{l} \rangle d\vec{l} \\ L_{diffuse}(\mathbf{x}; \vec{n}) &= k_d \int_{\Omega} L_i(\mathbf{x}; \vec{l}) \langle \vec{n}, \vec{l} \rangle d\vec{l}. \end{aligned} \tag{2.6}$$

This results in an environment map indexed by the two-dimensional vector $\vec{n}$. Miller and Hoffman [85] and Heidrich [50] also use the Phong BRDF [16],

$$f_r(\vec{v}, \vec{l}) := k_s \frac{\langle \vec{r}_v, \vec{l} \rangle^N}{\langle \vec{n}, \vec{l} \rangle}, \tag{2.7}$$

where k_s and N control the shape and size of the Phong lobe, to prefilter the environment map:

$$\begin{aligned} L_{Phong}(\mathbf{x}; \vec{v}, \vec{n}, \vec{t}) &= \int_{\Omega} k_s \frac{\langle \vec{r}_v, \vec{l} \rangle^N}{\langle \vec{n}, \vec{l} \rangle} L_i(\mathbf{x}; \vec{l}) \langle \vec{n}, \vec{l} \rangle d\vec{l} \\ L_{Phong}(\mathbf{x}; \vec{r}_v) &= k_s \int_{\Omega} \langle \vec{r}_v, \vec{l} \rangle^N L_i(\mathbf{x}; \vec{l}) d\vec{l}. \end{aligned} \tag{2.8}$$

This produces an environment map indexed by $\vec{r}_v$. For isotropic BRDFs, Kautz and McCool [58] and Cabral et al. [18] have generated two and four-dimensional prefiltered environment maps respectively for real-time rendering. Kautz et al. [60] further used the anisotropic Banks BRDF model [10], without self-shadowing, to generate a prefiltered three-dimensional environment map.

Using spherical harmonics [17, 71, 89, 97, 106], Ramamoorthi and Hanrahan [96] managed to further reduce the storage requirements for prefiltered environment maps, when the filter kernel is broad, by representing them using the nine lowest-order spherical harmonic coefficients. Spherical harmonics are the analogue on the sphere to the Fourier basis on the line or circle and can represent functions over the sphere compactly—thus, they are ideally suited for representing the integral in Equation 2.1. The drawback of Ramamoorthi and Hanrahan's original approach is that only the diffuse reflection component is encoded. However, they later extended this work into frequency space environment map rendering [98] which allowed real-time rendering of objects with isotropic BRDFs under distant illumination. In this work, they calculate the first nine spherical harmonic coefficients for each texel in the environment map and use them to represent the preintegrated BRDF at each of those texel points. As integration in spatial domain is equivalent to convolution in frequency domain, Equation 2.1 can be evaluated very efficiently in frequency space for interactive rendering. However, the main limitation in their work was that the first nine spherical harmonic coefficients can only represent a low frequency lighting environment or a broad glossy BRDF.

In a related work, Sloan et al. [109] used the spherical harmonic basis functions to encode object self-shadowing and self-reflection for diffuse and glossy objects. Lighting is sampled sparsely near the object and projected onto the spherical harmonic basis. Then a radiance transfer function is precomputed for a dense sampling over the object and stored into vectors or matrices, which are used for real-time shading. Their approach, however, is limited to isotropic BRDFs. In addition, the transfer functions are defined only for diffuse self-shadowing and diffuse inter-reflection, and so only low-frequency lighting environments can be handled. Furthermore, once the transfer function is precomputed, no change to the object's BRDF can be made. Kautz et al. [59] later extended this work for rendering arbitrary, but fixed, BRDFs; however, due to current GPU limitations, their implementation requires a fixed view or fixed lighting in order to achieve interactive performance.

Recently, McAllister et al. [77] performed interactive rendering of spatially varying BRDFs. They used the Lafortune BRDF [61] as basis lobes to approximate arbitrary BRDFs. This approach is ideally suited for surfaces with multiple BRDFs, which traditional methods require multiple rendering passes, because they usually can be approximated with fewer lobes than BRDFs. The parameters of each Lafortune BRDF lobe were stored in a series of texture maps and used to evaluate the BRDF at runtime. Their shader renders at real-time frame rates for a fixed number of hardware lights. For a full evaluation of the rendering equation, however, they still made the approximation of using a common point of projection and stored the preconvolved BRDF in an environment map.

The downside of these reflection map methods is that they are technically only accurate for one point: the pre-chosen centre of projection. Therefore, the environment captured is only valid for distant objects and illumination. Moreover, concave and planar objects will exhibit incorrect illumination. However, for low frequency lighting environments and for the prefiltered environment map case, the results are often blurred which hides the distortion and errors present. Object self-reflection and self-shadowing (due to local illumination evaluation) are also not captured in these approaches.

2.2 Image-Based Rendering

Image-based rendering is a relatively new rendering paradigm introduced into the computer graphics community in the 1990s [22, 23, 81]. Compared to traditional geometry-based rendering systems, the central idea behind this rendering approach is that images are the underlying data types and we can shuffle existing pixels around to make new images, rather than creating them from scratch. Early image-based rendering systems like Apple Computer's *QuickTime VR* system [22] and McMillan and Bishop's plenoptic modeller [81] used outward looking environment maps at fixed positions which allow users to change their viewing direction. Other systems use blending, interpolation, warping, or a combination of these techniques to render their final image. The theoretical framework behind these systems is the plenoptic function.

2.2.1 The Plenoptic Function

Formalized by Adelson and Bergen [3], the plenoptic function (from the Latin *plenus*, complete or full, and *optic*) describes pencils of rays through points in space, at any time, over any wavelengths. By fixing a point (V_x, V_y, V_z) in space, a direction in spherical coordinates (θ, ϕ), a time frame (t), and a wavelength (λ), the plenoptic function yields the intensity of light with those parameters: $P(V_x, V_y, V_z, \theta, \phi, t, \lambda)$. This records every visible phenomenon anywhere, at any time, at any wavelength—making it practically impossible to tabulate this function. However, the dimensionality of the full plenoptic function is very high. In computer graphics, the time dimension is usually dropped (resulting in a static scene with fixed illumination) and λ is assumed to be evaluated at a three colour value such as red, green, and blue. Furthermore, by restricting the space to be free of occluding objects (free space), the position dimension drops to two (u and v); this reduction in dimensionality can be performed because the radiance along two points in space does not change unless blocked. These simplifications yield a more manageable four-dimensional function:

$$P : (u, v) \times (\theta, \phi) \rightarrow incident\ radiance. \tag{2.9}$$

In this form, the plenoptic function can be easily encoded into a two-dimensional array of two-dimensional images. In fact, a reflection map can be seen as a special case of the plenoptic function because it records incident radiance at a point. If we let $\mathbf{x} = (u, v)$ and $\vec{r}_v = (\theta, \phi)$, and compare to Equation 2.3, we get:

$$P(u, v, \theta, \phi) \equiv P(\mathbf{x}; \vec{r}_v) \equiv L_i(\mathbf{x}; \vec{r}_v). \tag{2.10}$$

The plenoptic function will be related to the rendering equation, $L(\mathbf{x}; \vec{v})$, in Section 2.2.3.

The reason behind using captured images to generate new ones is that there are still many phenomena which computer graphics cannot reproduce or reproduce well. By using images from the physical world, we can incorporate those effects into our virtual world. The key though is in keeping the function manageable while still producing novel views of the scene, which may not have been recorded originally.

2.2.2 Light Fields

A light field [67] encodes Equation 2.9 as two-dimensional slices called a light slab or a two-plane parameterization of the plenoptic function (Figure 2.4). The first plane has Cartesian coordinates (u, v) and the second plane has Cartesian coordinates (s, t). To acquire an image data set, the camera is placed at each (u, v) point and an image is recorded of the scene coinciding with the (s, t) plane. Rendering then just involves looking up the

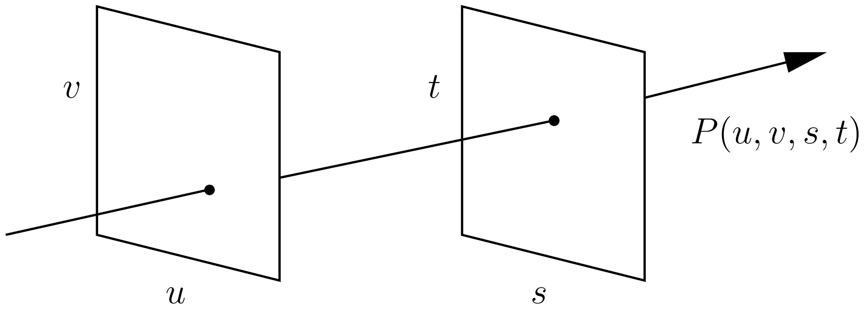


Figure 2.4: Geometry of light fields.

value at (u, v, s, t) for each pixel in the new view. To reduce data redundancy, Levoy and Hanrahan employ vector quantization (VQ) [38] to vectors of the two-dimensional or four-dimensional image tiles. Entropy coding is also performed using the Lempel-Ziv [127] compression algorithm. Decompression then involves decoding the Lempel-Ziv code (during loading) and a table look up into the VQ codebook (during rendering). Light field display can be accelerated by rendering tiles using projective texture mapping hardware. Furthermore, mip-mapping hardware can be used to interpolate between the four-dimensional samples to avoid aliasing artifacts and missing data.

Gortler et al. [42] parameterize Equation 2.9 similarly (calling it a lumigraph) and project the function onto a quadralinear basis function. Furthermore, they use geometric information to incorporate a depth correction term into their basis function; by doing this, they reduce the blurriness visible in Levoy and Hanrahan's final renderings. Finally, they use a hole filling algorithm to fill out missing information from the acquisition stage. These holes are caused by scene areas not visible from camera positions in the data set. The final images are rendered with the aid of texture mapping hardware.

From these two seminal papers, many other parameterizations [19, 51, 63, 69], compression issues [72, 73, 74, 114], and rendering techniques [48, 100, 101, 107, 108] were investigated. Furthermore, the theory behind the sampling and reconstruction of the plenoptic function were solidified using signal processing theory by Chai et al. [21] and Ramamoorthi and Hanrahan [97]. As an aside, this function has also been studied in computer vision literature in the areas of stereo disparity, epipolar volume, and optical flow analysis [9, 14, 41, 73]. Three important papers relating to modelling reflection properties of surfaces using light fields were provided by Neto and Bishop [29], Heidrich et al. [47], and Cabral et al. [18]. Neto and Bishop added dynamic shading to their image-based rendering system by estimating the three-dimensional geometry and surface normals from the acquired images. When rendering, they warped the reference images to the desired camera view and performed an additional shading step to account for new or moved light sources. On the other hand, Heidrich et al. chose to add reflection and refraction effects to light field rendering by decoupling illumination from scene geometry. Instead of storing radiance as in Equation 2.9, a ray direction is stored:

$$P_r : (u, v) \times (\theta, \phi) \rightarrow (\theta_r, \phi_r). \tag{2.11}$$

Depending on the effect desired, (θ_r, ϕ_r) encodes either the reflection or refraction direction. This vector is then used to look up the radiance value in a dual parabolic environment map [49] using the pixel texture extension [103]. In work similar to mine, Cabral et al. use a hybrid environment map and image-based rendering scheme to render isotropic surface reflectance. They first acquire prefiltered environment maps at vertices of an icosahedron surrounding a chosen centre of projection. At rendering time, a warp dependent on the surface's BRDF is applied to the three environment maps nearest to the shaded point (assuming the object being shaded is enclosed by the icosahedron).

Spherical barycentric interpolation is used to blend between the three resulting warped environment maps. Common to these three approaches is they all use geometric information to improve the final rendering's accuracy.

Although light field methods can reproduce images with stunning results, these methods have several limitations. First, to avoid blurriness, the plenoptic function has to be sampled densely. This in turn produces huge data sets; image data sets with the magnitude of several gigabytes are not unheard of. Some sort of compression must be employed to make the data manageable for real-time rendering. Also inherent in the data acquisition process are the limitations of free space, static scene, fixed illumination, and fixed surface BRDFs. Techniques such as hole filling (or multi-dimensional scattered data approximation) and three-dimensional shape approximation have to be employed if these limitations are to be circumvented.

2.2.3 Surface Light Fields

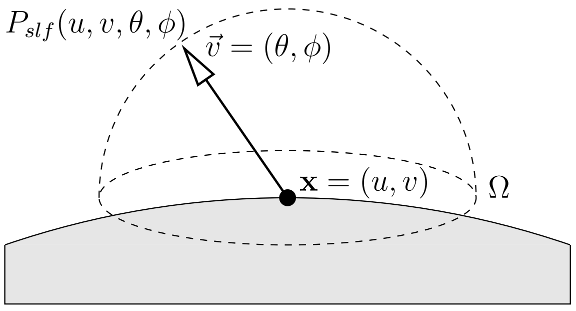


Figure 2.5: Geometry of surface light fields.

Surface light fields (SLFs) are light fields parameterized on the surface being rendered. In terms of Equation 2.9, (u,v) represents the position on the object's surface mesh (which can be achieved using known parameterization methods [33, 65]) and (θ, ϕ) represents a vector originating at (u, v) (Figure 2.5). Furthermore, if we let $\mathbf{x} = (u, v)$, and $\vec{v} = (\theta, \phi)$, and compare to Equation 2.1, we get:

$$P_{slf}(u, v, \theta, \phi) \equiv P_{slf}(\mathbf{x}; \vec{v}) \equiv L(\mathbf{x}; \vec{v}). \tag{2.12}$$

Instead of incident radiance in Equation 2.9, we have:

$$P_{slf} : (u, v) \times (\theta, \phi) \rightarrow exitant\ radiance. \tag{2.13}$$

SLFs are ideally suited for efficient reproduction of many surface reflectance properties as long as they can be recorded.

SLFs store the precomputed rendering equation for points on the surface. Miller et al. [84] exploits this property and uses it to solve global illumination problems. To reduce storage requirements, they apply a JPEG-like compression technique. Wood et al. [125] took a different approach to their research by capturing physical models by range-scanning and photography methods, reparameterizing them into the SLF form, and rendering them at interactive rates. They use a data filling algorithm to cover holes in the data, and function quantization and principal function analysis to compress it. They further demonstrate that small changes to the lighting environment, surface reflectance properties, and mesh shape can be made while still maintaining a believable result. The accuracy of the result, however, has not been investigated.

Noticing that Equation 2.12 is a four-dimensional function similar to Equation 2.2, Chen et al. [24] tried applying BRDF factorization techniques to compress the SLF. First, they partitioned the SLF around each vertex of the surface mesh. Next, the SLF around each vertex is factorized and compressed using principal component analysis [11] or non-negative matrix factorization [66]. Finally, the resulting SLF is rendered using multi-pass texture mapping hardware. In related work, Latta and Kolb [64] applied BRDF homomorphic factorization techniques [79] to factor the rendering equation with an image-based lighting environment. Their approach, however, only works for fixed isotropic BRDFs and static lighting environments.

The image-based SLF techniques have produced some of the most realistic renderings to date. Effects such as anisotropy and sub-surface light scattering can be observed in the final results because they were originally captured in the data set. SLF approaches, however, inherit limitations from light field methods. Except for Wood et al.'s parameterization, the scene acquired and rendered is static with fixed illumination. Furthermore, the BRDFs of the surface are not easily modified.

2.3 Summary

Reflection mapping methods have been around for some time and their strengths and weaknesses are well understood. In particular, they are easily implemented and can be efficiently evaluated in hardware. However, reflection maps are technically only valid for one point, or when the lights and objects in the environment can be assumed to be infinitely distant. Distortion and the absence of self-reflection and self-shadowing are also evident—especially in environments with high-frequency information. On the other hand, image-based rendering approaches are able to reconstruct surface reflection properties relatively accurately. Nevertheless, they have significant storage requirements, need to deal with missing data, and are limited to static scenes with fixed illumination and fixed surface BRDFs.

This thesis blends together concepts from reflection mapping, light fields, and surface light fields to try and address some of these shortcomings. Specifically, environment maps are used for simple and efficient evaluation of surface reflectance. To increase accuracy, an image-based approach is utilized to improve reflections for near-field objects, and to add a single level of self-reflection and self-shadowing effects. Furthermore, changes to the surface's BRDF can be performed. Although not implemented in this thesis, other algorithmic extensions are discussed in Section 8.1.

Chapter 3

Offset Surface Light Fields

Offset surface light fields are based on concepts from reflection maps, light fields, and surface light fields. Since SLFs are well suited to encode the rendering equation accurately, I will try to retain their parameterization. This will help in generating accurate reflections at rendering time. However, I would like to make my parameterization more flexible by allowing for reflectance property changes. By moving the sample points from the surface (as in the case of traditional SLFs) to points slightly above the the surface, the BRDF of the surface becomes semi-decoupled from the lighting environment (Figure 3.1). Doing so allows accurate surface reconstruction, using the original SLF, and additional reflections and material changes, using the acquired lighting environment. Furthermore, moving the sample points above the surface also allows more sample points to be used for each point on the surface. In addition, it also permits a reasonable setting for the near plane when capturing images. The light field at these sample points are captured into environment maps to take advantage of simple hardware-accelerated evaluation. To avoid overly large storage costs, only enough sample points are acquired

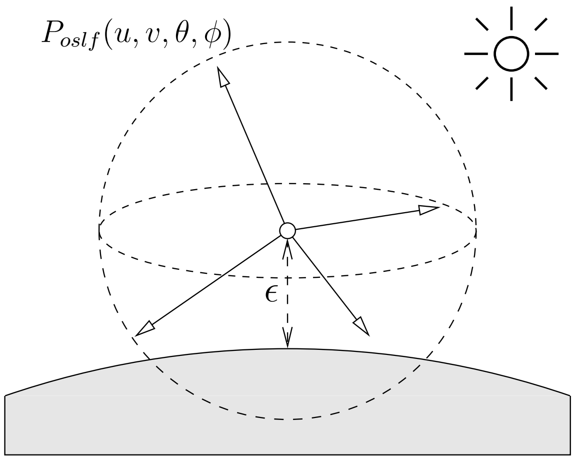


Figure 3.1: Geometry of offset surface light fields.

in order to reconstruct the original object accurately. Holes in the data are covered by blending between these sample points. I call this representation offset surface light fields (OSLFs) because the sample points are offset from the surface by a small amount.

This chapter will discuss the representation in more detail as well as how it is used to reconstruct the original object's SLF. Since I'm primarily concerned with generating reflections, I make the simplification and assumption that the acquired object has a Lambertian material. Chapter 4 will show how relatively accurate reflections can be achieved using this representation while Chapter 5 will show how to generalize OSLFs to evaluate local illumination.

3.1 Parameterization

To parameterize the lighting environment around an object into the OSLF representation, a parameterization of the surface must first be obtained. This can be done using known methods for triangular meshes [33, 65]. To simplify things, I have chosen bicubic tensor product Bézier patches [35] as my surface representation. Bézier patches are simply bivariate polynomials of third degree with the Bernstein polynomials, $B_i^n(t)$, as their basis functions. The patches are specified using a two-dimensional set of control points, $CP_{i,j}$, called a control polygon. Evaluation of a surface point, $\mathbf{x} = \mathbf{x}(u, v)$, just requires successive linear affine combinations of the control points:

$$\mathbf{x}(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} CP_{i,j} B_{i,j}^3(u, v) \tag{3.1}$$

where

$$CP_{i,j} \in \Re^3, \tag{3.2}$$

$$B_{i,j}^3(u, v) = B_i^3(u) B_j^3(v), \tag{3.3}$$

and

$$B_i^3(t) = \left(\begin{array}{c} 3 \\ i \end{array} \right) t^i (1 - t)^{n-i}. \tag{3.4}$$

This can be done efficiently using various algorithms [35, 75] which I will not present in detail here. Bézier patches not only yield a surface parameterized by (u, v), but also a surface normal and tangent plane at $\mathbf{x}$. Moreover, they also allow for a smoother surface representation compared to triangular meshes.

Next, a sampling of the surface is performed with controlled coverage and the resulting surface sample points offset from the surface by ϵ units. These points are enumerated in an array for fast and easy indexing. At each of these OSLF sample points, an environment map is captured—recording a pencil of rays through each point. These pencils represent the light field at the sample points.

As mentioned previously, I would like to emulate the SLF parameterization as closely as possible while avoiding its huge storage costs and bandwidth. So instead of storing a SLF at every $\mathbf{x}(u, v)$ point on the surface, indices to a finite set of OSLF sample points which best represent the light field at $\mathbf{x}(u, v)$ is stored. For fast access by GPUs, these indices are encoded in the form of a texture map associated with the surface patch. As current texture map formats allow up to four channels per texel, I store four sample point indices in each texel. I use texture maps with 16 bits per channel which allows a maximum of 65536 indexed sample points. I will refer to these maps as sample maps.

Compared to SLFs, my parameterization remains as:

$$P_{oslf} : (u, v) \times (\theta, \phi) \rightarrow exitant\ radiance, \tag{3.5}$$

where

$$\begin{aligned} P_{oslf}(u, v, \theta, \phi) &= \mathcal{F}(L_i(\mathbf{spt}_0(u, v); \theta, \phi), L_i(\mathbf{spt}_1(u, v); \theta, \phi), \\ &\quad L_i(\mathbf{spt}_2(u, v); \theta, \phi), L_i(\mathbf{spt}_3(u, v); \theta, \phi)), \end{aligned} \tag{3.6}$$

$$\mathbf{spt}_j : (u, v) \rightarrow \textbf{OSLF sample point}, \tag{3.7}$$

and

$$L_i : \textbf{OSLF sample point} \times (\theta, \phi) \rightarrow incident\ radiance. \tag{3.8}$$

Here $\mathbf{spt}_j$ is the mapping between surface coordinates (u, v) and the jth best sample point (stored in channel j). The intermediate mapping from index to sample point is omitted for simplicity. $L_i(\mathbf{spt}_j(u, v); \theta, \phi)$ represents the light field at the jth best sample point (compare to Equation 2.3) and $\mathcal{F}$ is a function which reconstructs the outgoing radiance for directions (θ, ϕ) from the incoming radiance at the four sample points. The function $\mathcal{F}$ is described in more detail in Sections 3.4, 4.1, 4.2, and 5.3.

3.2 Data Acquisition

I tested this parameterization using a virtual data set acquired by a ray tracer (Section 8.1.1 describes what can be done to acquire physical data). Newell's teapot was chosen as the test object; this teapot is composed of 28 bicubic tensor product Bézier patches. The teapot was placed in a gazebo which was generated using Alias|Wavefront's *Maya Unlimited* and exported to Pixar's *RenderMan* file format. Then, a high-dynamic range environment (Debevec's eucalyptus grove light probe [30]), lights, and surface shaders were added. The teapot was assigned a medium-green Lambertian material. Figure 3.2 shows three views of this environment. Next, the teapot's object description was

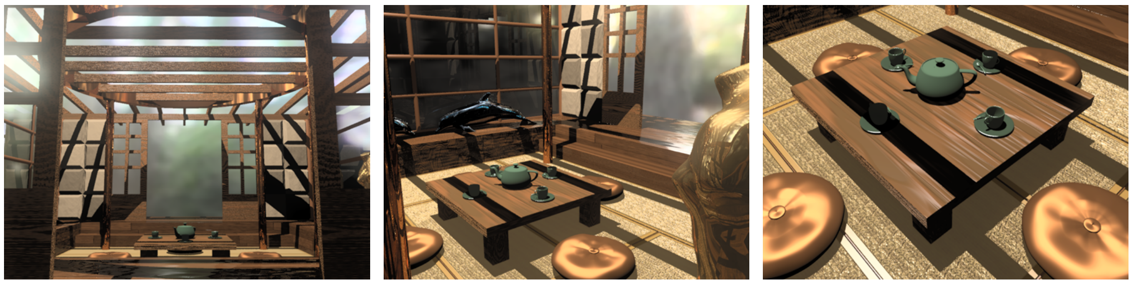

(a) View 1 (b) View 2 (c) View 3

Figure 3.2: Three views of the gazebo scene.

parsed and sample points were generated on its surface at fixed intervals (Figure 3.3). These points were then offset from the surface by $\epsilon = 0.1$ and 0.25 units; for comparison, the diameter of the teapot's rim is approximately three units. Next, environment

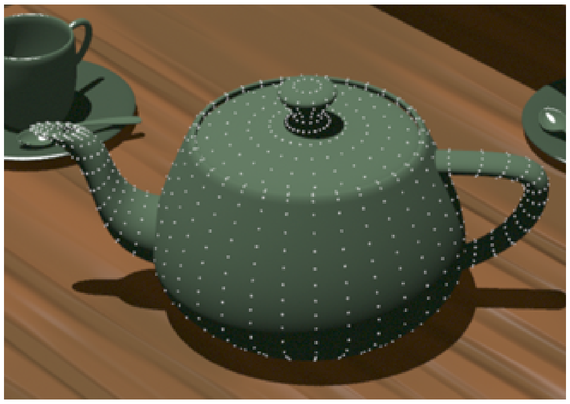


Figure 3.3: Teapot surface sample points.

maps were acquired at each OSLF sample point using Exluna's *Blue Moon Rendering Tools* (*BMRT*) renderer. The axes of the environment map were taken to be the canonical world axes. The renderer produces 128×128 and 256×256 LZW-compressed, 32 bits per channel, IEEE floating-point images in TIFF format which were then assembled into cube maps (Figure 3.4). Along with each cube map, the OSLF sample point's world

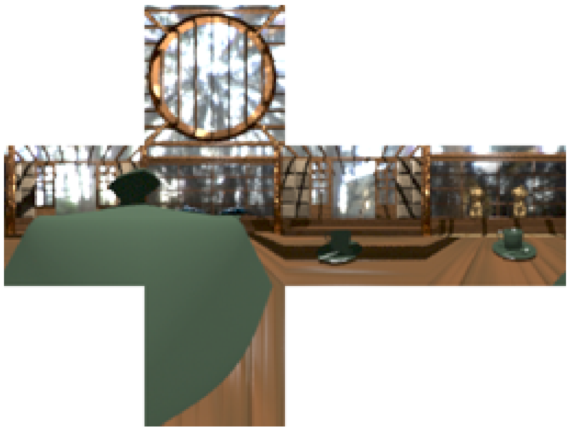


Figure 3.4: Example of an acquired cube map.

coordinates are stored. Finally, the geometry of the teapot is stored just as in the case of traditional SLFs. There were 1156 OSLF sample points generated for the teapot and the size of the resulting 128×128 data set was approximately 1.325 gigabytes (GBs) while the 256×256 data set was approximately 5.179 GBs. These sizes may seem large at first, but previous representations which acquire images with 8 bits instead of 32 bits per channel achieve similar or much larger data sets. Unless otherwise stated, all renderings shown in this thesis use the 128×128 data set.

The free space requirement for light fields must be obeyed for the space between the OSLF sample point and the teapot's surface itself. Since this distance, ϵ, is small, this usually is not a problem. If this precondition is violated, artifacts will appear in the original teapot's reconstruction. Since I'm primarily concerned with adding reflection effects, I have selected a Lambertian material for my test object. Thus, it is important to note that the surface reconstruction algorithm presented in Section 3.4 is only valid for objects with a diffuse texture. In Chapters 4 and 5, I will show how an additional mirror reflection or BRDF can be added onto the original object's diffuse texture. This is similar to adding reflections with an environment map, although my method produces more accurate results. The technique described could also be applied recursively, although I did not attempt this in my implementation. The only other issue in data acquisition is to ensure the ray tracer's near clipping plane value is less than ϵ otherwise the teapot surface near the OSLF sample point will not be visible.

3.3 Preprocessing

Preprocessing involves generating the 16 bits per channel sample maps which store indices to the enumerated OSLF sample points. The pseudocode for generating these sample maps is as follows:

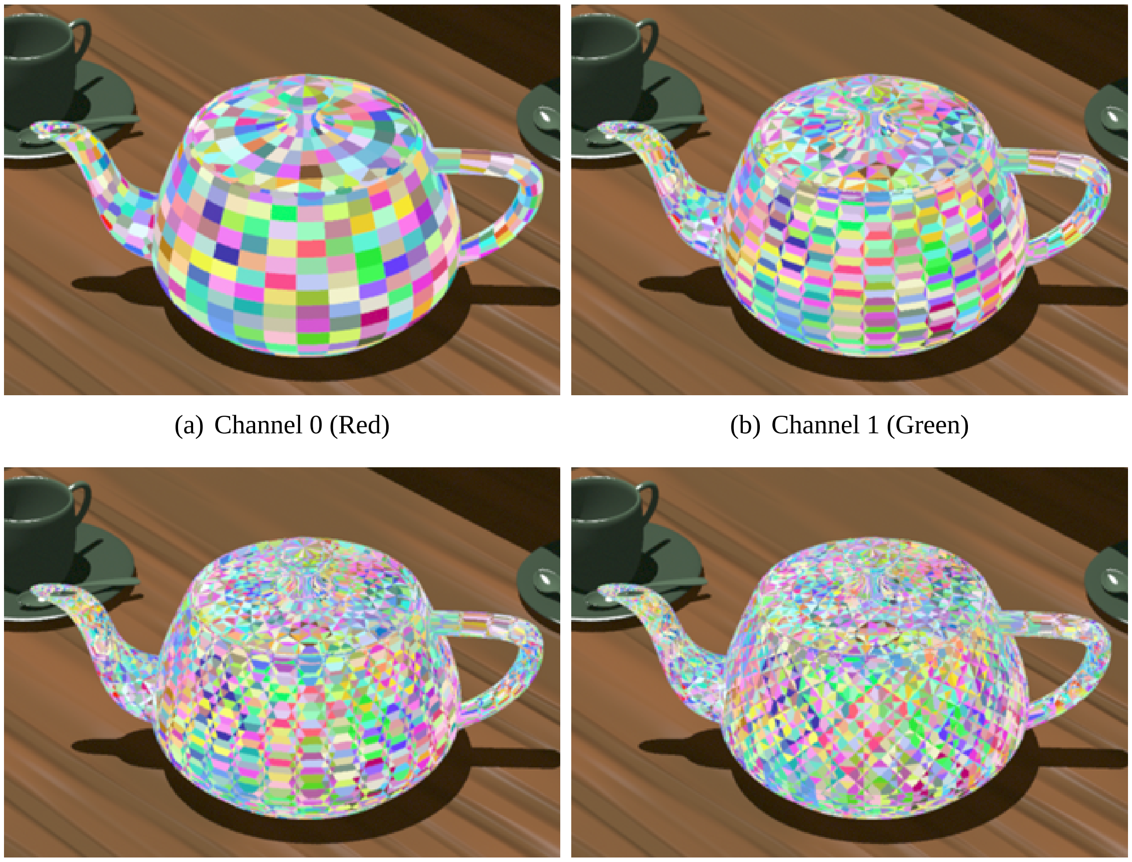
```
for each Bézier patch $B$ {
    for each texel in $B$'s sample map {
        Evaluate corresponding point, $\mathbf{t}$, on $B$'s surface.
        Find four sample points that best represents $\mathbf{t}$.
        Store indices to these points in $B$'s sample map.
    }
}
```

Sample maps are mapped onto Bézier patches such that edges of the sample maps correspond to edges of the Bézier patches. This means that each texel in a sample map corresponds to a finite region in a Bézier patch. To evaluate $\mathbf{t}$, simply take the coordinates of the texel, map them between zero and one, and evaluate Equation 3.1. In my implementation, I used sample maps of size 256×256 for each patch.

The four best points are chosen to be the four closest sample points which are visible from $\mathbf{t}$. To do this, perform a search through the OSLF sample point array and pick the four sample points closest to $\mathbf{t}$ using the standard Euclidean distance measure on the sample points and $\mathbf{t}$. To satisfy the visibility constraint, these points must lie in the upper hemisphere of $\mathbf{t}$. That is,

$$\langle \vec{n}_{\mathbf{t}}, \|(OSLF\ sample\ point) - \mathbf{t}\| \rangle > 0, \tag{3.9}$$

where $\vec{n}_{\mathbf{t}}$ is the normal at $\mathbf{t}$ and $\|\cdot\|$ represents the norm in three-space. Furthermore, sample points near the horizon should be excluded because their view of $\mathbf{t}$ can be obstructed due to aliasing in the environment map. Introducing a tolerance angle accounts


(a) Channel 0 (Red) (b) Channel 1 (Green)

(c) Channel 2 (Blue) (d) Channel 3 (Alpha)

Figure 3.5: Teapot sample maps displayed with false colours to indicate different indices.

for this:

$$\langle \vec{n}_{\mathbf{t}}, \|(OSLF\ sample\ point) - \mathbf{t}\| \rangle > \cos(\phi_{samplemap}). \quad (3.10)$$

I used $\phi_{samplemap} = 85^\circ$ in my implementation. Finally, these four points are sorted in ascending order by distance and stored into the RGBA channels of the sample map. Figure 3.5 shows the results of this step.

One last implementation detail is for objects with self-intersections, points inside the surface should be removed before the sample map generation is done. This, however, is not an issue for the teapot model.

3.4 Surface Reconstruction

Reconstruction of the original object's surface is very easy because every texel in the sample map stores a constellation of points visible from it. This means the surface's diffuse texture can be reconstructed accurately by sampling in the direction of the surface point (Figure 3.6).

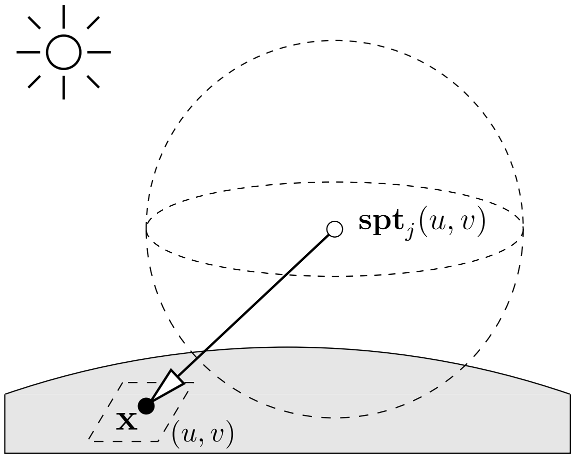


Figure 3.6: Geometry of surface reconstruction for offset light fields.

Formally, to shade the point $\mathbf{x}$ on the teapot, first obtain the four point indices associated with $\mathbf{x}$. Texture mapping hardware gives these indices automatically, by performing the mapping: $\mathbf{x} \rightarrow (u,v) \rightarrow indices$. The GPU has to be set to nearest-neighbour texel interpolation because any sort of interpolation on point indices is invalid. From these indices, we obtain the corresponding sample points: $\mathbf{spt}_0(u,v)$, $\mathbf{spt}_1(u,v)$, $\mathbf{spt}_2(u,v)$, and $\mathbf{spt}_3(u,v)$. By indexing into the light field in the direction of the surface point,

$\vec{v}_{recon} = (\mathbf{x} - \mathbf{spt}_j(u,v))$, we obtain the surface reflectance (Equation 3.8):

$$L_i(\mathbf{spt}_j(u,v); \vec{v}_{recon}) = L_i(\mathbf{spt}_j(u,v); \theta_{\vec{v}_{recon}}, \phi_{\vec{v}_{recon}}). \tag{3.11}$$

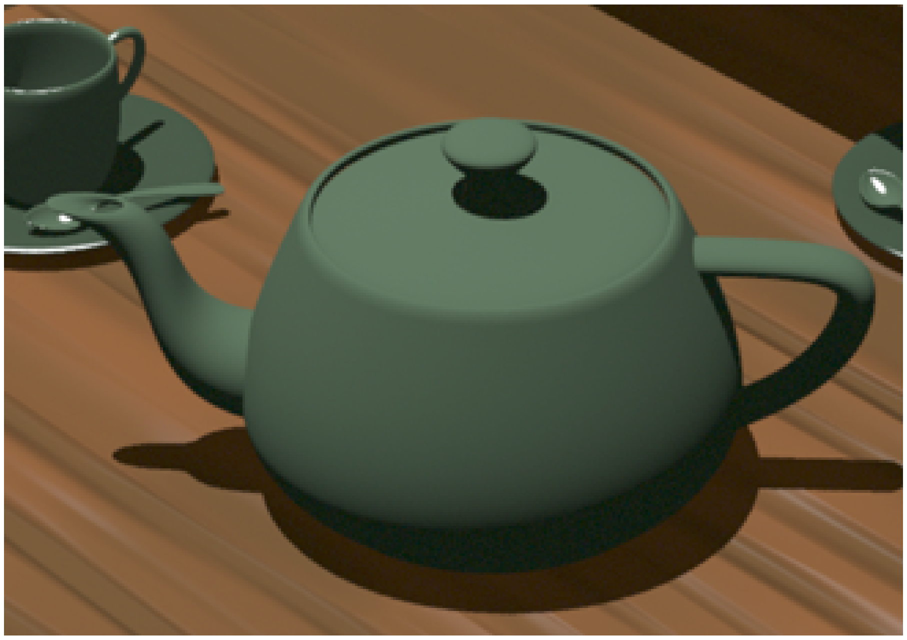
This indexing is performed using bilinear interpolation of the cube map.

Although a blend of the four resulting samples would make sense, I have found that the sample from the closest OSLF sample point, $\mathbf{spt}_0(u,v)$, is sufficient to produce an accurate result. Blends such as averaging the results produced no visible advantage at a slightly increased evaluation cost. For non-diffuse objects, a blend between sample points is necessary and this is discussed in Section 8.1.1. The final equation in the case of surface reconstruction is:

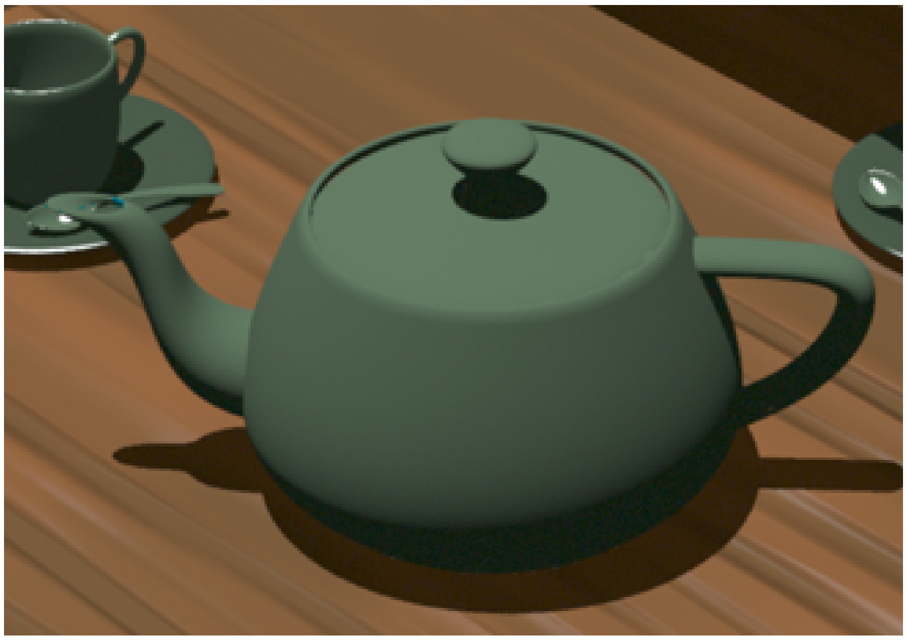
$$\begin{aligned} \mathcal{F}_{recon}(L_i(\mathbf{spt}_0(u,v);\theta,\phi), L_i(\mathbf{spt}_1(u,v);\theta,\phi), &= L_i(\mathbf{spt}_0(u,v);\theta_{\vec{v}_{recon}},\phi_{\vec{v}_{recon}}). \\ L_i(\mathbf{spt}_2(u,v);\theta,\phi), L_i(\mathbf{spt}_3(u,v);\theta,\phi)) & \end{aligned} \tag{3.12}$$

Figure 3.7 shows results of this surface reconstruction stage. Finally, if the original surface reconstruction is all that is needed, then

$$P_{oslf} = \mathcal{F} = \mathcal{F}_{recon}. \tag{3.13}$$



(a) Ray trace



(b) Offset surface light field

Figure 3.7: Comparison of surface reconstruction results.

Chapter 4

Specular Reflections

For objects acquired without reflections, such as the diffuse textured teapot in my test scene, the shading algorithm can add reflections at runtime. Accurate real-time reflections are difficult to achieve using reflection mapping techniques. In the OSLF representation, however, the cube maps at sample points $\mathbf{spt}_j(u, v)$ actually describe an accurate light field through $\mathbf{spt}_j(u, v)$. The closer the sample points are to the point being shaded, the more accurate a reflection can be obtained. Since the preprocessing stage already stores sample points closest to the surface, a blend between incident radiance at these points will give a fairly accurate reflection. This chapter discusses how to reproduce mirror reflections, which, due to high-frequency information, are generally difficult to reproduce accurately for real-time rendering. The camera view in Figure 1.1 will be used as a basis for comparison because it shows the effects of near-field reflections (the table texture and teacups), far-field reflections (the roof beams and gazebo environment), teapot self-reflection, and teapot self-shadowing effects. These are the critical cases to examine when comparing accuracy between various methods.

4.1 Mirror Reflection

Traditional mirror reflections are accomplished by adding the surface's diffuse term ($L_{diffuse}$) to a specular reflection term ($L_{specular}$). The result is then modulated by the surface's colour (Cs):

$$L(\mathbf{x};\vec{v}) = Cs * (L_{diffuse}(\mathbf{x};\vec{v}) + L_{specular}(\mathbf{x};\vec{v})), \tag{4.1}$$

where $\mathbf{x}$ is the point being shaded and $\vec{v}$ is the viewing direction. $L_{specular}$ is the component shown in Figure 1.1. The specular term for mirror reflections is equal to the incoming radiance from direction $\vec{r_v}$ (Equation 2.4):

$$L_{specular}(\mathbf{x};\vec{v}) = L_i(\mathbf{x},\vec{r_v}). \tag{4.2}$$

In the OSLF paradigm, points $\mathbf{spt}_j(u,v)$ were predetermined to be closest to $\mathbf{x}$. Since the mirror reflection at $\mathbf{x}$ can be approximated by the mirror reflection at a point close to $\mathbf{x}$, say $\mathbf{spt}_0(u,v)$, we obtain the following:

$$\begin{aligned} L_{specular}(\mathbf{x};\vec{v}) &\approx L_{specular}(\mathbf{spt}_0(u,v),\vec{r_v}) \\ &\approx L_i(\mathbf{spt}_0(u,v),\vec{r_v}). \end{aligned} \tag{4.3}$$

The result of this approximation is shown in Figure 4.1. Another possible approximation uses the sample point closest to the reflection vector in angular space:

$$L_{specular}(\mathbf{x};\vec{v}) \approx L_i(\mathbf{spt}_k(u,v),\vec{r_v}), \tag{4.4}$$

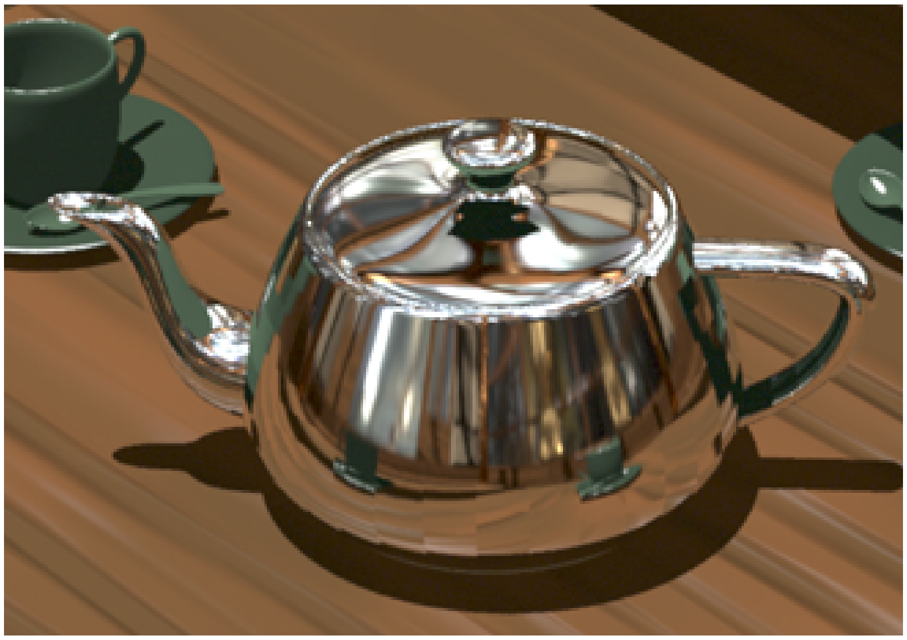


Figure 4.1: $L_{specular}$ term for mirror reflections rendered using the closest sample point in Euclidean space.

where

$$\langle \|\mathbf{spt}_k(u,v) - x\|, \vec{r}_v\rangle \geq \langle \|\mathbf{spt}_j(u,v) - x\|, \vec{r}_v\rangle \ \forall j. \tag{4.5}$$

This approximation is shown in Figure 4.2 Already, the reflections generated by these two approximations are visibly more accurate than environment map reflections. However, the structure of the sample map is evident in both images. These blocky artifacts, which are prominent in the table texture and the teapot knob reflections, are caused by abrupt transitions between points in the sample map. To smooth out the transition, I perform a blend between the resulting radiance:

$$\begin{aligned} L_{specular}(\mathbf{x};\vec{v}) \quad \approx \quad & \mathcal{F}_{refl}(L_i(\mathbf{spt}_0(u,v);\theta,\phi), L_i(\mathbf{spt}_1(u,v);\theta,\phi), \\ & L_i(\mathbf{spt}_2(u,v);\theta,\phi), L_i(\mathbf{spt}_3(u,v);\theta,\phi)), \end{aligned} \tag{4.6}$$

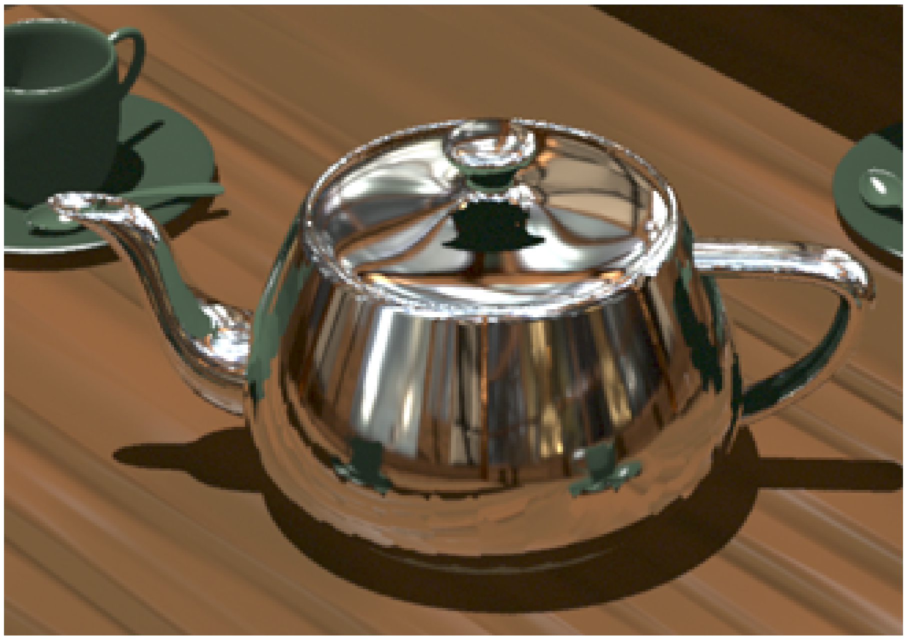


Figure 4.2: $L_{specular}$ term for mirror reflections rendered using the closest sample point in angular space.

where

$$\begin{aligned}\mathcal{F}_{refl}(L_i(\mathbf{spt}_0(u,v);\theta,\phi), L_i(\mathbf{spt}_1(u,v);\theta,\phi), &= \sum_{j=0}^{3} \mathcal{B}_j L_i(\mathbf{spt}_j(u,v);\theta_{\vec{r}_v},\phi_{\vec{r}_v}),\\ L_i(\mathbf{spt}_2(u,v);\theta,\phi), L_i(\mathbf{spt}_3(u,v);\theta,\phi)) & \qquad (4.7)\end{aligned}$$

for some basis function $\mathcal{B}_j$.

4.2 Blending Basis Functions

In selecting a basis function for blending, we must ensure the basis functions form a partition of unity:

$$\sum_j \mathcal{B}_j = 1. \tag{4.8}$$

The first obvious blend to try is an average of the incident radiance:

$$\mathcal{B}_j^{avg} = \frac{1}{4}. \tag{4.9}$$

The result is shown in Figure 4.3.

Similar to early lumigraphs [42, 48, 67, 101], this basis function produced blurring and ghosting artifacts, especially in the teacup and table texture reflections, because all

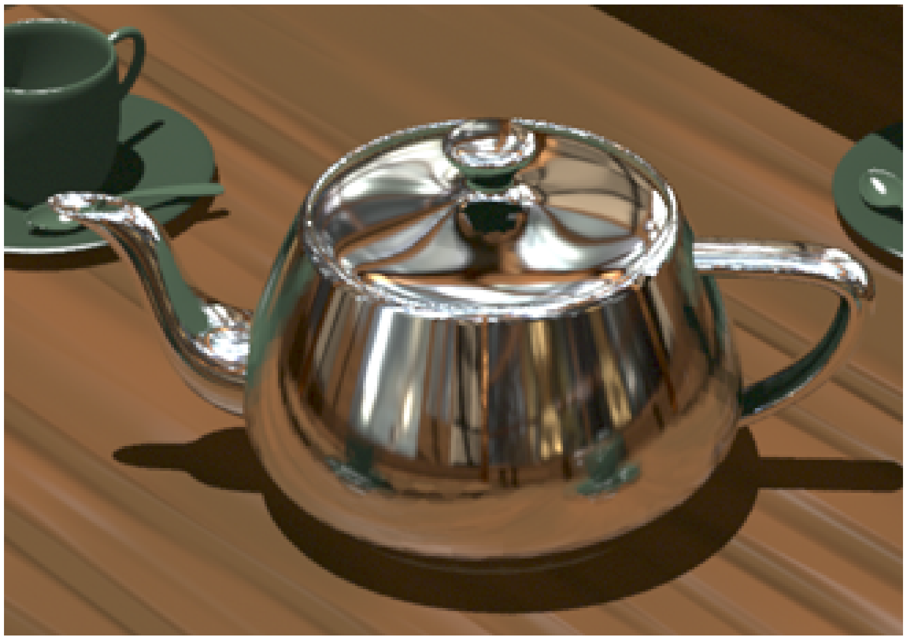


Figure 4.3: $L_{specular}$ term for mirror reflections rendered using $\mathcal{B}^{avg}$.

radiance samples were given equal weight. It is desirable then to give more weight to samples with higher accuracy. Note that another cause for the blurriness is the data set's resolution: the 256×256 data set produces clearer reflections than the 128×128 one (Section 6.4).

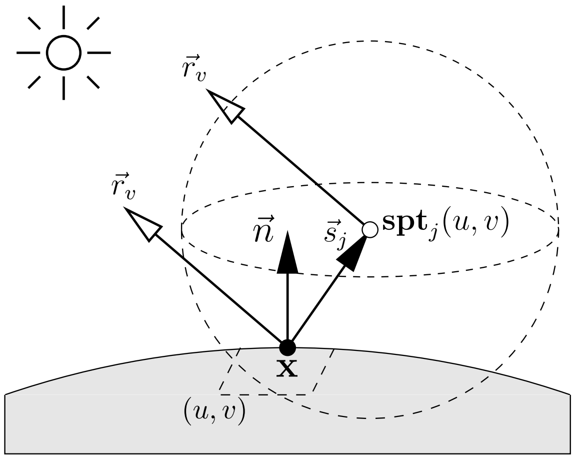


Figure 4.4: Geometry of reflection for offset surface light fields.

Next, I tried a basis function based on the cosine lobe. Let $\vec{s}_j = \|\mathbf{spt}_j(u,v) - \mathbf{x}\|$ (Figure 4.4). Then the blending function is defined by:

$$\mathcal{B}_j^{\vec{r}} = \frac{\max(\langle \vec{r}_v, \vec{s}_j \rangle^N, 0)}{\sum_j \max(\langle \vec{r}_v, \vec{s}_j \rangle^N, 0)}, \tag{4.10}$$

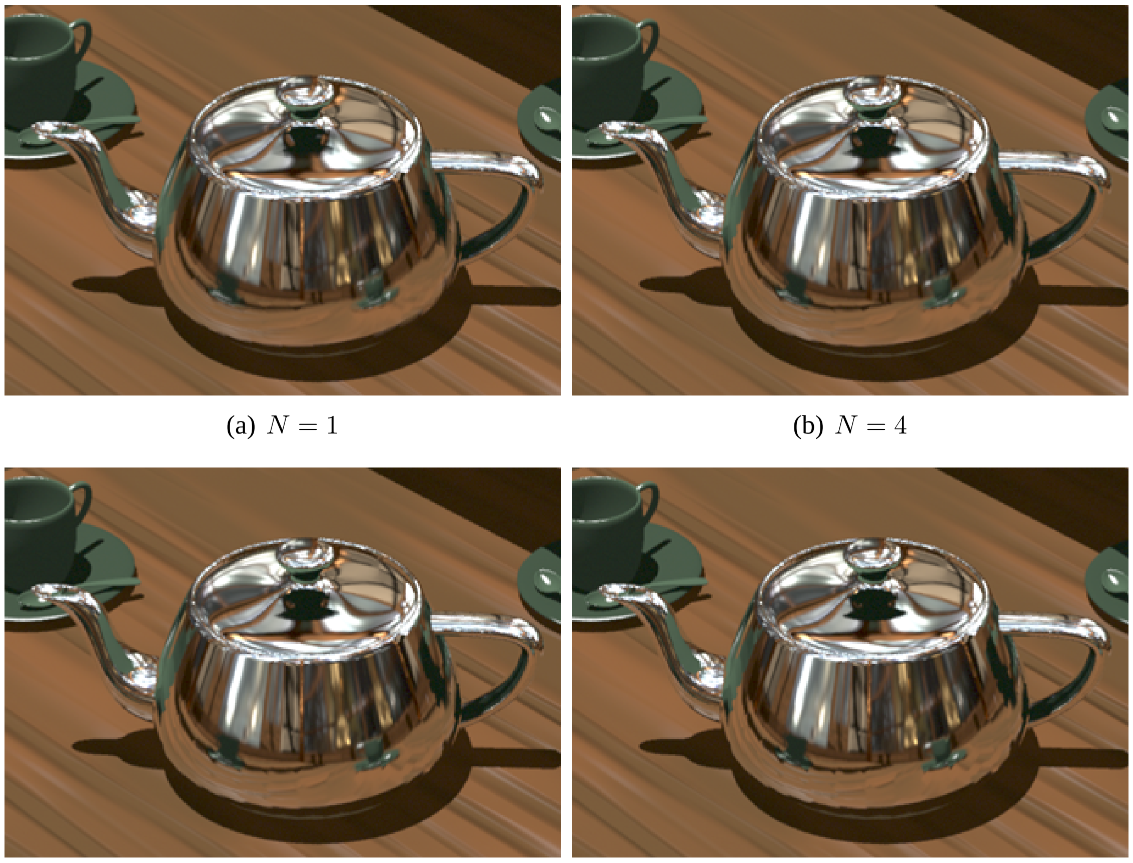
where $\vec{r}_v$ is the reflection vector. This blending function is similar to a Phong lobe [16] and its shape is controlled by the parameter N. The inner product achieves a maximum value when $\vec{s}_j = \vec{r}_v$. When this occurs, indexing into $\mathbf{spt}_j(u,v)$'s cube map in direction $\vec{r}_v$ produces a 100% accurate reflection: $L_i(\mathbf{spt}_j(u,v); \vec{r}_v)$. Furthermore, the inner product decreases to zero as the angle between $\vec{s}_j$ and $\vec{r}_v$ increases. This means that points which give more accurate reflection samples are given more weight. The basis function is clamped below by zero to avoid using points in the opposite hemisphere of $\vec{r}_v$ (since $\langle \vec{r}_v, \vec{s}_j \rangle < 0$ if and only if the angle between $\vec{r}_v$ and $\vec{s}_j$ is greater than $90°$). Points in this hemisphere will usually give a more erroneous radiance sample unless they are closer to $\mathbf{x}$ than the other sample points and the angle between $\vec{r}_v$ and $\vec{n}$ is small. To simplify the algorithm, the weights of these points are set to zero (as an aside, I also tried $\mathcal{B}_j^{|\vec{r}|} = \left| \frac{\langle \vec{r}_v, \vec{s}_j \rangle^N}{\sum_j \langle \vec{r}_v, \vec{s}_j \rangle^N} \right|$ but this basis function produced more artifacts). Finally, the basis function is normalized by the sum of all the inner products so they partition unity. Note that when $N = 0$, this basis function reduces to $\mathcal{B}^{avg}$.

The result of using $\mathcal{B}^{\vec{r}}$ is shown in Figure 4.5. Compared to the previous results, blurriness and ghosting artifacts are reduced but still visible. As N increases, so does the sharpness of the reflections. The structure of the sample maps is also visible, especially for higher values of N. Lastly, the reflections are still more accurate than the environment map result.

Although the results are quite accurate, they do not look smooth and this is predominantly due to the sample maps' structure. To reduce the visibility of this structure, I used the following basis function instead:

$$\mathcal{B}_j^{\vec{n}} = \frac{\langle \vec{n}, \vec{s}_j \rangle^N}{\sum_j \langle \vec{n}, \vec{s}_j \rangle^N}, \tag{4.11}$$

where $\vec{n}$ is the normal at $\mathbf{x}$ and N controls the shape of the cosine lobe. This basis function simplifies the implementation because $\langle \vec{n}, \vec{s}_j \rangle \geq 0$ by construction of the sample maps and the $\max(\ldots)$ function in $\mathcal{B}^{\vec{r}}$ can be avoided. Using the same analysis as before, points further away from the normal in angular space will contribute less to the result. This implies more weight is given to points closer to $\mathbf{x}$ and therefore the basis function's



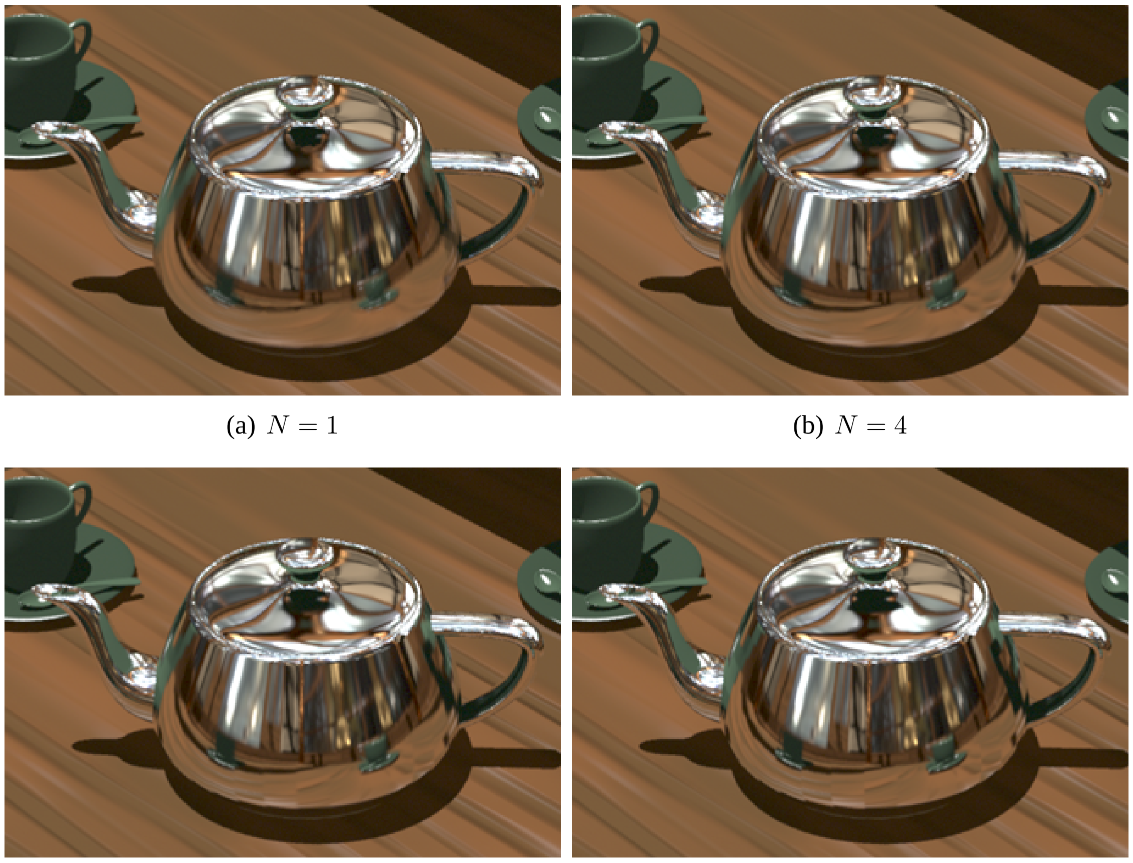
(a) $N = 1$ (b) $N = 4$

(c) $N = 8$ (d) $N = 16$

Figure 4.5: $L_{specular}$ term for mirror reflections rendered using $\mathcal{B}^{\vec{r}}$ for various N.

support is more centred around $\mathbf{x}$. This produces a smoother blend between radiance samples, especially across texel boundaries in the sample map. However, this smoothing comes at a cost of losing some accuracy, because sampling the reflection from L_i using $\vec{s}_j = \vec{n}$ is only accurate if $\vec{r}_v = \vec{n}$.

The result of using $\mathcal{B}^{\vec{n}}$ is shown in Figure 4.6. As expected, transition between texel boundaries appear more continuous at the expense of some accuracy. My own observations, performed on a CRT monitor, show that $N = 4$ results in the most visually



(a) $N = 1$ (b) $N = 4$

(c) $N = 8$ (d) $N = 16$

Figure 4.6: $L_{specular}$ term for mirror reflections rendered using $\mathcal{B}^{\vec{n}}$ for various N.

smooth image. I defined the most visually smooth image as the image which has the least artifacts due to visual discontinuities, as perceived by the observer. Even though the images are less accurate than in the previous result, it is still visibly more accurate than environment map reflections.

Putting together the diffuse and specular components to construct the OSLF in Equation 3.6, with added reflections, we get:

$$P_{oslf} = \mathcal{F} = \mathcal{F}_{recon} + Cs * \mathcal{F}_{refl}. \tag{4.12}$$

Here, $\mathcal{F}_{recon}$ is the diffuse component of the teapot and $\mathcal{F}_{refl}$ is its mirror reflection. The result of this evaluation is shown in Section 6.4, Figure 6.3. This construction can also be applied to any object for which additional mirror reflections are desired. In the latter case, $\mathcal{F}_{recon}$ will be the object's original material, which could in itself incorporate many complex effects, and $\mathcal{F}_{refl}$ is the mirror reflection obtained by the above algorithm.

4.3 Texture Cache

Most consumer systems today do not have enough main memory, let alone texture memory, to store the 1.325 GBs required for the smallest data set. That means most of the data must be stored on slow access hard disks. Even with an Ultra3 SCSI (also known as Ultra160 Wide) controller, software rendering times for reflections were around half an hour per frame. To get around this bottleneck, a texture caching system was implemented. As the shader is meant to be executed on a GPU, this texture caching system must be compatible with graphics hardware for it to be useful.

I implemented three caching systems based on the least recently used (LRU) eviction strategy [56, 112]. The LRU cache replacement algorithm simply chooses the cache entry that is least recently used as the eviction candidate. Although there are better performing algorithms [56], such as least frequently used (LFU), LRU is one of the most popular in commercial applications because of its simple implementation. LRU is easily implemented in my software shader: a 32-bit counter is initialized to zero and incremented every time the shader is called. Every time a texture is accessed, an entry in the cache is created and associated with the current counter value. If the entry already exists in the cache, then its counter value is updated with the current shader counter. When the cache is full, a search of the cache line is performed and the entry with the smallest counter value is replaced. Two separate cache lines are maintained: one for the sample maps and one for the cube maps.

Sample maps are stored on disk in an uncompressed TIFF format which allows random access between image rows; because of this, I decided to store only a single scanline in its texture cache buffer. This means a cache miss will occur every time a different scanline had to be accessed. I found that a 2 item cache produced a 71.8% miss rate whereas a 64 item cache produced a 15.6% miss rate. However, the actual rendering time only differed by a second. This shows the bottleneck lies in cube map access.

Cube maps are stored on disk in a LZW-compressed TIFF format which only allows sequential access; because of this, three different read strategies were employed: read on demand, read half threshold, and read full image. The read on demand strategy reads into the cache buffer everything from the start of the image to the requested texel. The read half threshold strategy is the same as read on demand, except when the requested texel lies in the second half of the image; in this case, the whole image is read into the cache buffer. The reasoning is that future reads are likely to lie in the second half of the image because of the way bilinear interpolation works. In the read on demand case, if a second cache miss is produced at this stage, the algorithm would have read more data in total than if it brought in the full image on the first access, avoiding a cache miss. Finally, the read full image strategy reads in the whole image whenever a new cube map face is requested.

Results (Section 6.6) show that the read full image strategy has the lowest miss rate and shortest rendering time. Since this read strategy parallels the *OpenGL* [103] and *DirectX* [82] texture management systems, the miss rates on a hardware-accelerated implementation should be similar. However, the data set sizes are still larger than I would like. I discuss possible strategies to deal with the data set size in Section 8.1.1.

Chapter 5

Bidirectional Reflection Distribution Functions

Showing that OSLFs can handle accurate reflection is clearly not enough. Any representation for surface reflectance should not be reflection model limited. In this chapter, I will show how OSLFs can be extended to handle arbitrary BRDFs. I will use Monte Carlo integration to evaluate the rendering equation at the shaded point. Monte Carlo integration is well suited for this job because OSLFs already store results of Equation 2.1 in cube maps, for points close to the shaded point. These cube maps act as an illumination cache that the integration algorithm can use. I begin by reviewing the key concepts from Monte Carlo integration. Next, I will demonstrate evaluation of local illumination using the anisotropic Ashikhmin and Shirley BRDF model [8]. This chapter concludes with a description on how to use arbitrary BRDF models.

5.1 Monte Carlo Integration

Monte Carlo methods have been around since the 1940s. The principle behind Monte Carlo integration is the use of random sampling to estimate integrals (hence the reference to the infamous *Casino de Monte-Carlo*). I will quickly review the main results of Monte Carlo methods but more detail can be found in Shreider [105] and Veach [119].

To evaluate an integral using the Monte Carlo technique, it is first converted into an equivalent expected value problem using probability theory:

$$I = \int f(x)dx = \int \frac{f(x)}{p(x)}p(x)dx = E\left[\frac{f(x)}{p(x)}\right], \tag{5.1}$$

where $p(x)$ is some arbitrary probability density function (PDF) which satisfies $p(x) > 0$ whenever $f(x) \neq 0$. The expected value is then estimated from M random samples generated with the PDF $p(x)$, making uniform sampling unnecessary. This gives an estimate for the integral:

$$I = E\left[\frac{f(x)}{p(x)}\right] \approx \frac{1}{M}\sum_{i=1}^{M}\frac{f(x_i)}{p(x_i)}, \tag{5.2}$$

where x_i's are the random samples. Monte Carlo methods are unbiased because the estimator $\frac{f(x)}{p(x)}$ is unbiased (Equation 5.1). Finally, it can be proved that basic Monte Carlo methods converge at a rate of $O\left(\frac{1}{\sqrt{M}}\right)$ [119].

5.2 The Ashikhmin and Shirley BRDF Model

I chose the Ashikhmin and Shirley model as the BRDF for demonstrating local illumination because it has an anisotropic specular term which produces a Phong-style specular lobe. Anisotropic BRDFs are generally more difficult to handle than isotropic BRDFs. Other properties of this BRDF that make it attractive are: it has intuitive control parameters, it satisfies the energy conservation and reciprocity principles, it incorporates Fresnel effects, it allows for a non-Lambertian diffuse term (although I do not use their diffuse term), and it is Monte Carlo friendly. Recently, Steigleder and McCool have demonstrated a hardware-accelerated implementation of this BRDF [110].

The Ashikhmin and Shirley BRDF model incorporates ideas from Neumann and Neumann [90], Schlick [102], and Ward [123]. At the simplest level, the BRDF decomposes into specular and diffuse components:

$$f_r(\vec{v},\vec{l}) = f_{sr}(\vec{v},\vec{l}) + f_{dr}(\vec{v},\vec{l}), \tag{5.3}$$

where f_{sr} is the specular reflection component and f_{dr} is the diffuse reflection component. This model is controlled by four parameters: R_s and R_d specify the specular and diffuse reflectance at normal incidence, and n_u and n_v are two Phong-like exponents that control the shape of the specular lobe. The model's diffuse term is

$$f_{dr}(\vec{v},\vec{l}) = \frac{28R_d}{23\pi}(1-R_s)\left(1-\left(1-\frac{\langle\vec{n},\vec{v}\rangle}{2}\right)^5\right)\left(1-\left(1-\frac{\langle\vec{n},\vec{l}\rangle}{2}\right)^5\right) \tag{5.4}$$

and its specular term is

$$f_{sr}(\vec{v},\vec{l}) = \frac{\sqrt{(n_u+1)(n_v+1)}}{8\pi} \frac{\langle \vec{n},\vec{h}\rangle^{\frac{n_u\langle \vec{h},\vec{t}\rangle^2 + n_v\langle \vec{h},\vec{n}\times\vec{t}\rangle^2}{1-\langle \vec{h},\vec{n}\rangle^2}}}{\langle \vec{h},\vec{k}\rangle \max(\langle \vec{n},\vec{v}\rangle, \langle \vec{n},\vec{l}\rangle)} F(\langle \vec{k},\vec{h}\rangle), \tag{5.5}$$

where

$$\vec{h} = \|\vec{v}+\vec{l}\| \tag{5.6}$$

is the halfway vector between $\vec{v}$ and $\vec{l}$,

$$F(\langle \vec{k},\vec{h}\rangle) = R_s + (1-R_s)(1-\langle \vec{k},\vec{h}\rangle)^5 \tag{5.7}$$

is Schlick's approximation of the Fresnel fraction [102], and $\vec{k}$ can be either $\vec{v}$ or $\vec{l}$.

To use the Monte Carlo method to estimate the rendering equation's integral with this BRDF model, first generate the random vector $\vec{h}$ using the PDF

$$p_h(\vec{h}) = \frac{\sqrt{(n_u+1)(n_v+1)}}{2\pi} \langle \vec{n},\vec{h}\rangle^{\frac{n_u\langle \vec{h},\vec{t}\rangle^2 + n_v\langle \vec{h},\vec{n}\times\vec{t}\rangle^2}{1-\langle \vec{h},\vec{n}\rangle^2}}. \tag{5.8}$$

This can be done by the following formula:

$$\begin{aligned} \vec{h} &= (\theta,\phi) \\ &= \left(\arccos\left((1-\xi_2)^{\frac{1}{\left(\frac{n_u\langle \vec{h},\vec{t}\rangle^2 + n_v\langle \vec{h},\vec{n}\times\vec{t}\rangle^2}{1-\langle \vec{h},\vec{n}\rangle^2}\right)+1}} \right), \arctan\left(\sqrt{\frac{n_u+1}{n_v+1}} \tan\left(\frac{\pi\xi_1}{2}\right) \right) \right), \end{aligned} \tag{5.9}$$

where (ξ_1,ξ_2) are two random numbers uniformly distributed in $[0,1)$ and $[0,1]$ respectively. This results in a vector $(\theta, \phi) \in [0, \frac{\pi}{2}] \times [0, \frac{\pi}{2})$. However, note the integration in Equation 2.1 is over the hemisphere Ω. To ensure full coverage and stratification of Ω, ξ_1 is mapped to one of four functions depending on where it lies in $[0, 1)$. For example, if $\xi_1 \in [0.5, 0.75)$ then evaluate $\phi(1 - 4(0.75 - \xi_1))$ using Equation 5.9 and rotate it about the $\phi = \pi$ axis. Next, $\vec{l}$ is calculated using Equation 2.4,

$$\vec{l} = 2\vec{h}\langle\vec{h}, \vec{v}\rangle - \vec{v}, \tag{5.10}$$

and it has the PDF

$$p(\vec{l}) = \frac{p_h(\vec{h})}{4\langle\vec{v}, \vec{h}\rangle}. \tag{5.11}$$

Finally, using Monte Carlo integration, the rendering equation can be estimated as:

$$\begin{aligned} L(\mathbf{x}; \vec{v}) &= \int_\Omega f_r(\vec{\omega}(\vec{v}, \vec{n}, \vec{t}), \vec{\omega}(\vec{l}, \vec{n}, \vec{t}))L_i(\mathbf{x}; \vec{l})\langle\vec{n}, \vec{l}\rangle d\vec{l} \\ &\approx \frac{1}{M}\sum_{i=1}^{M} \frac{f_r(\vec{v}, \vec{l})L_i(\mathbf{x}; \vec{l})\langle\vec{n}, \vec{l}\rangle}{p(\vec{l})}. \end{aligned} \tag{5.12}$$

As the teapot in my scene already has a diffuse material applied to it, I chose to use OSLF surface reconstruction (Section 3.4) to obtain the diffuse term. The other option is to use f_{dr} from Equation 5.4. The specular component, on the other hand, is calculated using Equation 5.5.

There are two ways to apply the Monte Carlo method for OSLFs. The first method performs Monte Carlo integration at each of the sample points $\mathbf{spt}_j(u, v)$ separately and then blends the results together. This is illustrated in the pseudocode below:

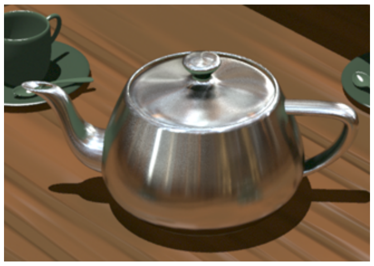
```
for each OSLF sample point {
    Generate $M$ random vectors ($h$).
    for each $h$ {
        Look up radiance from cube map.
    }
    Accumulate using Equation 5.12.
}
Blend results using Equation 4.7.
```

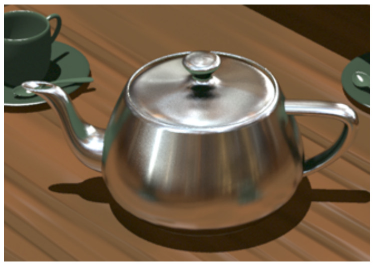
On the other hand, the second method's pseudocode is:

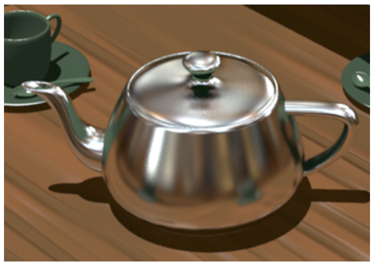
```
Generate $M$ random vectors ($h$).
for each $h$ {
    for each OSLF sample point {
       Look up radiance from cube map.
    }
    Blend results using Equation 4.7.
}
Accumulate using Equation 5.12.
```

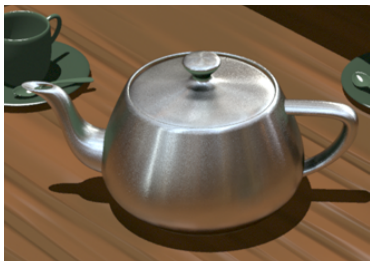
This second method is similar to doing a specular reflection calculation for each random vector and then accumulating the results an the outer loop. For large M, I expect the results of both methods to be similar. However, the first method uses more random vectors. I chose to implement the first method because it was the simplest to code and understand.

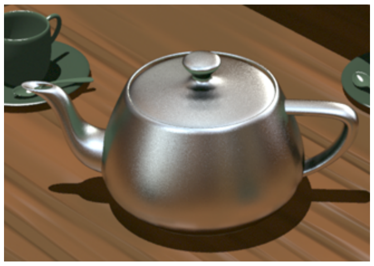
Just as in the case of specular reflection, using $\mathcal{B}^{\vec{n}}$ with $N = 4$ produced the smoothest looking results. Figure 5.1 shows the results using the $\mathcal{B}^{\vec{n}}$ blending function for

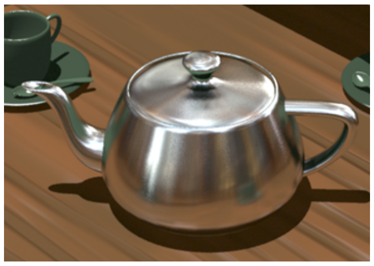

(a) $n_u = 10, n_v = 1000$

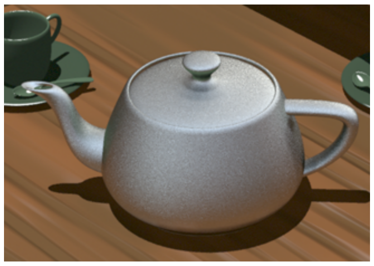

(b) $n_u = 100, n_v = 1000$

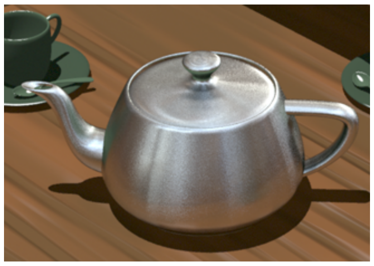

(c) $n_u = 1000, n_v = 1000$


(d) $n_u = 10, n_v = 100$


(e) $n_u = 100, n_v = 100$


(f) $n_u = 1000, n_v = 100$


(g) $n_u = 10, n_v = 10$


(h) $n_u = 100, n_v = 10$

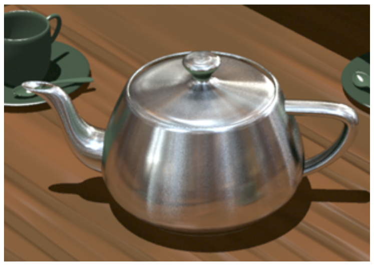

(i) $n_u = 1000, n_v = 10$

Figure 5.1: $L_{specular}$ term for Ashikhmin and Shirley's BRDF rendered using $\mathcal{B}^{\vec{n}}$ with $N = 4$, $R_s = 1$, $R_d = 0$, and for various exponents n_u and n_v. Monte Carlo integration was used with $M = 100$ samples for each OSLF sample point.

various values of n_u and n_v. The results for the other basis functions mirror the results from the mirror reflection case and are not included here. Accuracy is not as important in these renderings compared to the case of mirror reflections. This is because of the low-frequency result produced when the lighting environment is convolved with the BRDF. Very high values of n_u and n_v, however, will produce sharp mirror-like reflections and

$\mathcal{B}^{\vec{r}}$ may be used instead if accuracy is needed.

5.3 Arbitrary BRDF Models

As mentioned before, the Ashikhmin and Shirley BRDF model was chosen because it is a fairly general model with anisotropic properties. This makes it a good test case for OSLFs. This section shows how OSLFs can handle arbitrary BRDFs.

There are two ways to generalize this approach for arbitrary BRDFs. The first one performs Monte Carlo integration to evaluate the arbitrary BRDF at each of the OSLF sample points, $\mathbf{spt}_j(u, v)$, separately and then blends the results together to approximate the BRDF at $\mathbf{x}(u, v)$. The second one evaluates the arbitrary BRDF for each random $\vec{l}$ at each $\mathbf{spt}_j(u, v)$ separately, blends the results to approximate the incoming radiance at $\mathbf{x}(u, v)$ for that particular $\vec{l}$, and then accumulates all the results using Monte Carlo integration. These methods are listed in pseudocode at the end of Section 5.2.

Implementing either of these methods will allow the use of arbitrary BRDFs. This shows my representation is not reflection model limited and can be used to evaluate local illumination generally. Performing the evaluation this way can also be thought of as using the OSLF as an illumination cache which stores the illumination at every sample point. Rendering involves looking up lighting information from this illumination cache instead of going out to the environment to gather radiance—as in the case of the computationally intensive methods listed in Section 2.1.1.

To put everything together, Equation 4.12 is evaluated. For my case, $\mathcal{F}_{recon}$ is the diffuse component of the teapot and $\mathcal{F}_{refl}$ is the newly applied Ashikhmin and Shirley specular BRDF model. The result of this evaluation for the Ashikhmin and Shirley BRDF model is shown in Section 6.5, Figure 6.4. This construction can also be applied to any object for which an additional BRDF model needs to be imposed. In the latter case, $\mathcal{F}_{recon}$ will be the object's original material, which could in itself incorporate many complex effects, and $\mathcal{F}_{refl}$ is the additional BRDF evaluated using the above algorithm.

Chapter 6

Results

This chapter begins with a description of error metrics used to evaluate the results and proceeds to discuss the results of surface reconstruction, mirror reflections, and the Ashikhmin and Shirley BRDF in detail. Furthermore, it examines sources of error and describes minor implementation tweaks that were made to remove visible artifacts. Statistics for the texture cache are also presented and the chapter concludes with a summary of the results. This chapter can be skimmed over if the amount of detail presented here is not required.

6.1 Error Metrics

The difference between the final rendered images can be described qualitatively and quantitatively. Qualitative evaluation is based on observations by the human eye and tends to be subjective. On the other hand, quantitative evaluation is more objective because of its reliance on mathematically computable quantities. I will use both methods to evaluate OSLF results.

As previously mentioned, the camera view was chosen specifically so various effects can be observed. Near-field reflection effects can be observed in the teacup and table texture reflections while far-field reflection effects can be observed in the gazebo wall and roof beam reflections. Object self-reflection occurs mainly for the teapot spout, handle, and knob. Finally, object self-shadowing can be observed for the teapot handle and knob. Qualitative evaluation involves critical examination of areas with these effects.

Since qualitative evaluation depends on viewing conditions such as ambient light, angle and distance between the eye and the display, and display characteristics, quantitative evaluation presents a more objective measure of image differences. The metrics used for quantitative evaluation are divided into two categories: distortion metrics and fidelity metrics. Distortion metrics describe a mathematically measurable physical change between images. Examples are colour, intensity, and noise. Fidelity metrics, on the other hand, describe a mathematically computable perceptible difference between images using models of human vision and perception. Examples are the Daly Visual Differences Predictor [28] and Sarnoff Visual Discrimination Model [70]. I will use a simpler metric based on the CIEL*a*b* space [126]. The CIEL*a*b* space is a perceptually uniform colour space defined by the *Commission Internationale de L'Éclairage* (CIE) for subtractive colours (in contrast to the CIEL*u*v* space which is for additive colours). In this space, two colours which are perceived to be equally distant by viewers in specific viewing conditions are also numerically equidistant. The L* channel represents luminance relative to a reference white, the a* channel represents the red-green continuum, and the b* channel represents the yellow-blue continuum. I will perform a root mean-squared error (RMSE) calculation in this space and tabulate the results. Although the RMSE distortion metric is not suitable for measuring difference in many areas [39], performing RMSE in the perceptually based CIEL*a*b* space avoids some of its shortcomings. However, this measure still does not capture differences due to geometric distortions; defining a measure to compare two images perceptively is still an open research area. I've also included the RMSE in RGB space as a reference. To maintain numerical precision, 655×460, 32 bits per channel, IEEE floating-point images were used for the error calculation.

6.2 Sample Maps

Preprocessing results have already been shown in Section 3.3. However, there is an unavoidable point registration error introduced at this step. This error is the difference between the coordinates of surface points generated by my surface evaluation algorithm [75] and *BMRT*'s surface intersection algorithm. When considering the same point on the surface, I have found the error is usually between 0.01 and 0.0075 units; for comparison, the diameter of the teapot's rim is approximately three units. Therefore, the points generated to represent sample map texels are not as accurate as I would like.

6.3 Surface Reconstruction

When comparing surface reconstruction results of ray tracing versus OSLFs (Figure 3.7), the final images are virtually identical. Object self-shadowing is also reproduced because the shadows were present at the data acquisition stage. However, early surface

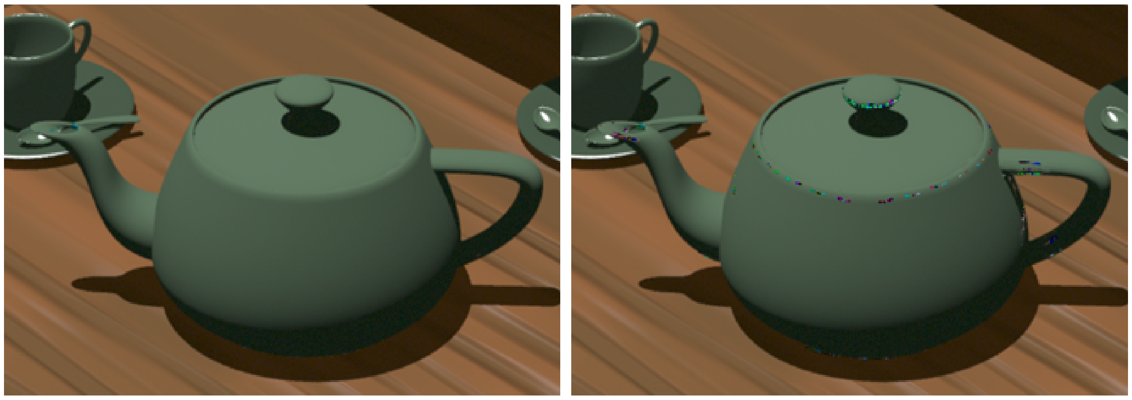

(a) Final rendering. (b) Early rendering.

Figure 6.1: Surface reconstruction results.

reconstruction renderings (Figure 6.1(b)) contained errors in the image, especially on the teapot rim and knob, due to sampling the cube map near the horizon (i.e. ϕ close to 90°). At these glancing angles, the area around the shaded point occupies a very small portion of view. To see this, take a piece of paper and look at it with the paper's plane perpendicular to your line of sight. Then slowly rotate the paper 90° so that the paper's plane becomes almost parallel to your line of sight. As you rotate the paper, the visible surface area decreases. This effect, coupled with aliasing artifacts in the cube maps, resulted in an incorrect radiance being sampled. Note that this problem does not occur when sampling radiance in the reflection or local lighting evaluation cases: since the sample points are above the surface, we can sample in directions close to $\phi = 90^\circ$ without problems, although with reduced accuracy.

There are many solutions to the sampling near the horizon problem. First, by increasing the resolution of the acquired cube maps, aliasing artifacts will be reduced and so does the occurrence of this problem. Second, by increasing the sampling density of the surface, the likelihood of the preprocessing algorithm choosing sample points near the horizon decreases. As the proportion of chosen sample points near the horizon decreases, the effect of these points on surface reconstruction is reduced and so are the artifacts in Figure 6.1(b). Third, decreasing the angle tolerance $\phi_{samplemap}$ (Section 3.3) in sample map generation also helps reduce the occurrence of this problem. The third solution is the most attractive because it does not involve regenerating or reacquiring a new the data set. However, by decreasing $\phi_{samplemap}$, the preprocessing algorithm might not find four OSLF sample points within the angle of tolerance for each texel. So instead, at runtime the shader finds the closest OSLF sample point within a specified angle of tolerance, ϕ_{shader}, from the shaded point's normal and uses that point to sample the surface. I used $\phi_{shader} = 65°$ for my renderings. Note that this angle is distinct from the $\phi_{samplemap}$ angle used in sample map generation because $\phi_{samplemap}$ is specified with respect to the point in the middle of the sample map texel whereas ϕ_{shader} is specified with respect to the point shaded at runtime.

This brings up the issue of finding enough sample points for each sample map given $\phi_{samplemap}$. One solution is to allow for less than four sample points per texel by storing an invalid index; in fact, I have tried this solution with excellent results. However, this adds to the runtime shader complexity, for example, by having to assigning zero weight to invalid sample points. Instead, I elected to base the surface sampling density on $\phi_{samplemap}$. For a smaller value of $\phi_{samplemap}$, a higher surface sampling density is needed to obtain four sample points per texel. This implies that areas with high surface curvature will have a higher sampling density. The sampling density is fixed per Bézier patch but is allowed to vary across patches. Finally, I would like to keep $\phi_{samplemap}$ as large as possible to decrease data set size and also because reflections are not affected by this issue as much. This is why the value $\phi_{samplemap} = 85°$ was chosen.

One final surface reconstruction implementation issue concerns the use of curved surfaces. This issue does not occur for triangular mesh objects. When shading high curvature curved surfaces, a sample point below the horizon might be used. In Figure 6.2,

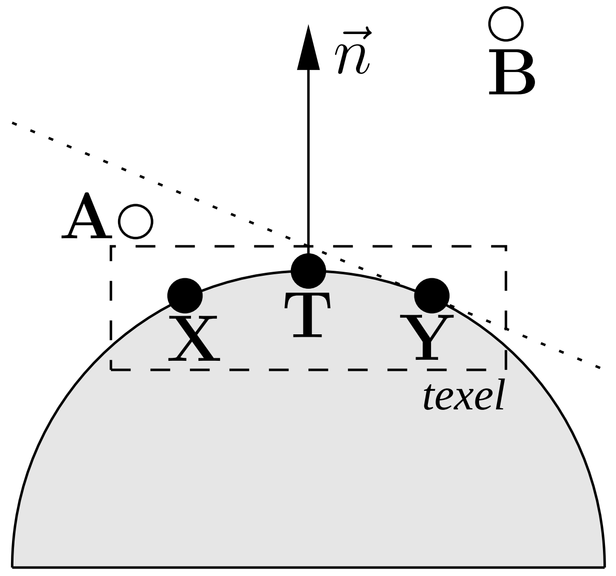


Figure 6.2: Geometry of the curved surfaces problem.

assume the preprocessing algorithm selects point $\mathbf{T}$ to represent a texel. Further assume that sample points $\mathbf{A}$ and $\mathbf{B}$ are stored in that texel and that $\mathbf{A}$ is closer to $\mathbf{T}$ than $\mathbf{B}$. When shading point $\mathbf{X}$, sample point $\mathbf{A}$ is used to determine the reflected radiance at $\mathbf{X}$ because it is closest to $\mathbf{T}$. The same is true for shading point $\mathbf{Y}$. However, using $\mathbf{A}$ in this case is incorrect because $\mathbf{Y}$ is not visible from $\mathbf{A}$: $\mathbf{A}$ lies in the lower hemisphere of $\mathbf{Y}$. Fortunately, ϕ_{shader} accounts for this problem as well.

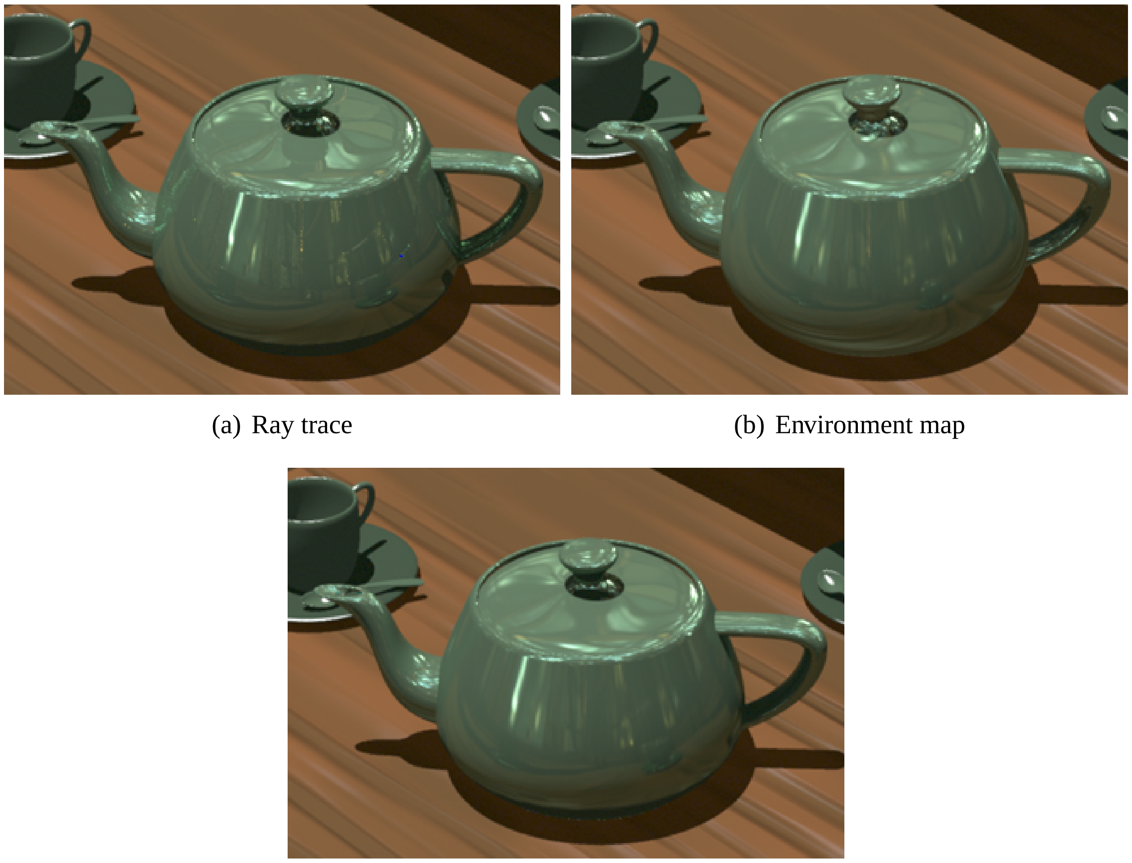
After accounting for these sources of error, one artifact remains when comparing the final results (Figure 3.7). Examining the teapot's lid near the rim, there are some small patches of lighter shade—especially near the teapot handle. Violation of the free space precondition causes these lighter patches to appear. At these regions, the sample map generation routine determined that the closest points lie on the other side of the rim. Therefore, the outer rim's lighter colour is sampled when sampling in the direction of the shaded point, resulting in these lighter regions. By decreasing the distance, ϵ, to the surface, this artifact is reduced and this was observed when I compared the $\epsilon = 0.25$ and $\epsilon = 0.1$ data sets. Figure 3.7(b) shows the teapot rendered with the $\epsilon = 0.1$ data set although some of these artifacts still remain. Another solution is to incorporate a depth correction term similar the original lumigraph paper [42]. This would be easy to do, for example, by storing cube maps in RGBZ format instead of RGB and then using the z coordinate to check for point concordance. As *BMRT* does not output the z coordinate (due to a bug in its RGBZ 32-bit IEEE floating-point output), I did not try this option. To completely eliminate this artifact, the free space condition has to be checked for every point in preprocessing. This problem is not unique to OSLFs as other image-based rendering methods, such as light fields and SLFs, also assume free space data acquisition.

Method	*Root Mean-Squared Error*					
	L*	a*	b*	R	G	B
OSLF$_{128}$	0.005351	0.012321	0.034318	0.000590	0.000952	0.004827
OSLF$_{256}$	*0.004679*	*0.007938*	*0.021619*	*0.000537*	*0.000892*	*0.003098*

Table 6.1: Comparison of OSLF surface reconstruction to a ray traced result using RMSE.

Table 6.1 compares the quantitative surface reconstruction quality between OSLF and ray trace methods; the subscripts indicate the resolution of cube maps used. Since RMSE roughly indicates the average change in a pixel between the two rendering methods, these results show that there is very little difference between ray tracing and OSLFs for surface reconstruction. As expected, using a higher resolution cube map improves rendering quality.

6.4 Specular Reflections


(a) Ray trace (b) Environment map

(c) Offset surface light field with $\mathcal{B}^{\vec{n}}$, $N = 4$.

Figure 6.3: Full surface reconstruction with added specular reflection rendered using three methods.

Qualitatively, OSLF approaches produce more accurate mirror reflections than the environment map approach. Near-field objects, such as the teapot, look more distant when using environment mapped reflections. In OSLF approaches, the teacups appear in the same areas as the ray traced reflections. When using the $\mathcal{B}^{\vec{r}}$ blending function, the shapes of the teacups are very accurate compared with the slightly enlarged teacups when using $\mathcal{B}^{\vec{n}}$. Also, while the $\mathcal{B}^{\vec{n}}$ renderings exhibit more blurriness than the $\mathcal{B}^{\vec{r}}$'s, the latter blending function makes the structure of the sample maps more visible (e.g. in the knob's reflection and table texture). The distortion of the table texture near the bottom of the teapot in the environment map rendering is also reduced in the OSLF renderings.

Similar comments can be made about far-field objects such as the gazebo structure and roof beams. One obvious flaw in the environment map rendering is the roof beam distortion near the knob. The OSLF renderings reproduces this reflection accurately.

Teapot self-reflection is notably absent in the environment map rendering whereas they appear accurately in the OSLF renderings. Reflections of the teapot spout, handle, knob, and teapot shadow appear in the same general area as in the ray traced rendering. Note that the reflection of the handle and knob appear on the teapot body as shadowed areas, because these areas were in shadow when the scene was acquired. Only one level of self-reflection is possible in a single pass with OSLFs, whereas a ray traced rendering can have many levels. Furthermore, the reflections are not as sharp as the ray traced counterpart. This is to be expected because of the blending functions used and also because OSLFs sample from preacquired, fixed-resolution cube maps. Increasing the resolution of the cube maps helps to reduce blurriness. I have observed, however, that OSLFs match environment maps in terms of resolving texture detail for the same resolution cube maps.

Table 6.2 compares the quantitative specular reflection quality between environment

Method	*Root Mean-Squared Error*					
	L*	a*	b*	R	G	B
EnvMap_{128}	0.198457	0.267233	0.301128	0.074420	0.039586	0.036783
EnvMap_{256}	*0.204308*	*0.268992*	*0.301899*	*0.074699*	*0.041149*	*0.037367*
Distance_{128}	**0.136457**	**0.248841**	**0.294089**	**0.072282**	**0.023559**	**0.031592**
Distance_{256}	***0.123106***	***0.245717***	***0.292770***	***0.071869***	***0.019287***	***0.030598***
Angle_{128}	0.138536	0.249629	0.294369	0.072318	0.024135	0.031722
Angle_{256}	*0.125873*	*0.246529*	*0.293082*	*0.071922*	*0.020223*	*0.030760*
$\mathcal{B}_{128}^{avg}$	0.137826	0.249274	0.294176	0.072320	0.023969	0.031726
$\mathcal{B}_{256}^{avg}$	*0.124964*	*0.246195*	*0.292920*	*0.071915*	*0.019981*	*0.030739*
$\mathcal{B}_{128}^{\vec{r}}, N = 1$	0.137538	0.249262	0.294211	0.072296	0.023922	0.031665
$\mathcal{B}_{128}^{\vec{r}}, N = 4$	0.137685	0.249351	0.294255	0.072292	0.023942	0.031650
$\mathcal{B}_{256}^{\vec{r}}, N = 4$	*0.125065*	*0.246274*	*0.292982*	*0.071902*	*0.020026*	*0.030705*
$\mathcal{B}_{128}^{\vec{r}}, N = 8$	0.137939	0.249434	0.294283	0.072295	0.023991	0.031662
$\mathcal{B}_{128}^{\vec{r}}, N = 16$	0.138215	0.249515	0.294307	0.072301	0.024058	0.031681
$\mathcal{B}_{128}^{\vec{n}}, N = 1$	0.137068	0.248988	0.294078	0.072287	0.023745	0.031642
$\mathcal{B}_{128}^{\vec{n}}, N = 4$	0.136870	0.248949	0.294102	0.072283	0.023725	0.031616
$\mathcal{B}_{256}^{\vec{n}}, N = 4$	*0.124759*	*0.246009*	*0.292878*	*0.071914*	*0.019957*	*0.030716*
$\mathcal{B}_{128}^{\vec{n}}, N = 8$	0.136788	0.248932	0.294104	0.072283	0.023730	0.031609
$\mathcal{B}_{128}^{\vec{n}}, N = 16$	0.136858	0.248949	0.294114	0.072286	0.023778	0.031617

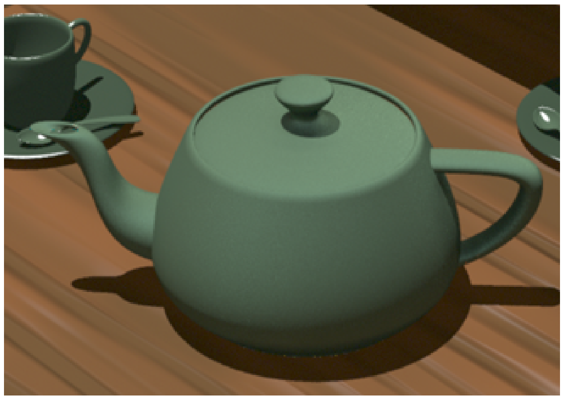
Table 6.2: Comparison of OSLF mirror reflections to a ray traced result using RMSE.

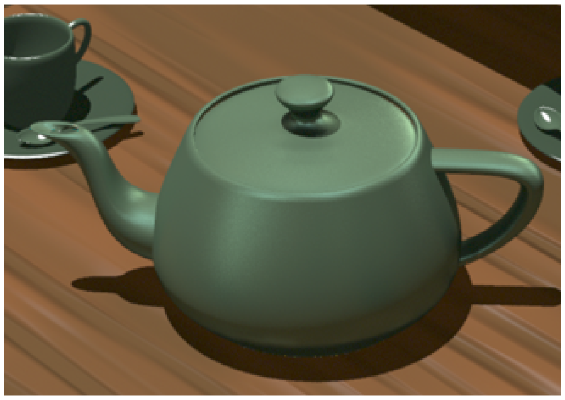
map and various OSLF blending functions with ray trace methods; the subscripts indicate the resolution of cube maps used, “EnvMap” represents reflections generated using environment mapping, “Distance” represents reflections generated using the closest point in Euclidean space, and “Angle” represents reflections generated using the closest point in angular space. The bold numbers indicate the lowest RMSE in each channel, for 128×128 and 256×256 sized cube maps. In all cases, OSLF methods produce more accurate reflections than environment mapping. While the RMSE metric indicates that the closest point method produces the most accurate reflections, the discontinuities across sample map texel boundaries do not make its images visually pleasing—this is a defect of the RMSE measure. While other techniques also produce results which are quantitatively close to the lowest RMSE, the $\mathcal{B}^{\vec{n}}$ blending function with $N = 4$ produces the most visually smooth result. Finally, while OSLF methods become more accurate as cube map resolution is increased, environment mapping actually becomes less accurate: this is because the blurriness in the lower resolution environment map hide many errors which appear at a higher resolution.

6.5 Bidirectional Reflection Distribution Functions

Due to time constraints, I did not implement a ray tracing, Monte Carlo evaluation of local illumination for the Ashikhmin and Shirley BRDF model in a *RenderMan* shader. Therefore, a ray traced result is not available for comparison. However, when this BRDF is convolved with the lighting environment, a lower frequency result is usually produced (Figure 6.4). Therefore, accuracy is not as much a factor in this case compared to mirror reflections. Since the mirror reflection results show OSLFs are more accurate than environment maps but slightly less accurate than the ray traced result, this comparison will also hold when evaluating local illumination with a BRDF.

In evaluating local illumination at a point (Equation 2.1), radiance is sampled from directions over the hemisphere and then convolved with the BRDF. Specular reflection results show OSLFs can sample radiance in the reflection direction more accurately than the environment map approach (and almost as accurately as a ray tracer). This implies it can also sample radiance in any direction over the hemisphere more accurately than the environment map approach (and almost as accurate as a ray tracer). Therefore, I expect local lighting evaluation using OSLFs to be more accurate than environment map based


(a) $n_u = 10, n_v = 10$


(b) $n_u = 100, n_v = 100$

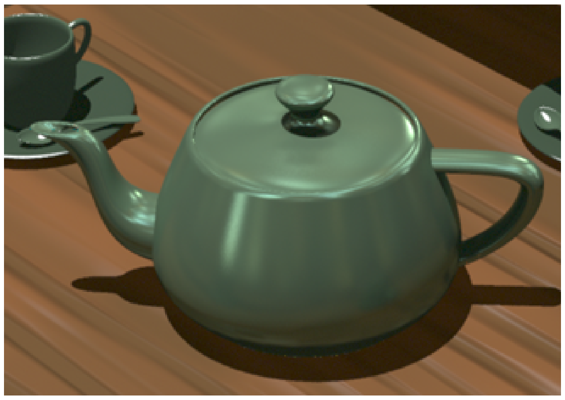

(c) $n_u = 1000, n_v = 1000$

Figure 6.4: Full surface reconstruction with added Ashikhmin and Shirley BRDF.

approaches but slightly less accurate than ray tracing approaches. We can very roughly validate my lighting computation by comparing Figure 5.1 to the metallic spheres in Ashikhmin and Shirley [8], which were generated with Monte Carlo-based ray tracing. Examining both set of images show the surface reflectance properties of the teapot is similar to the metallic spheres.

Finally, OSLFs also incorporate object self-shadowing due to local illumination evaluation. These shadows come naturally because evaluating the rendering equation at points on the surface takes this factor into account automatically. This self-shadowing effect is distinct from the preacquired shadows reproduced by surface reconstruction (Figure 6.1(a)) and the effect can seen more clearly by examining the specular term of the BRDF: Figure 5.1 reveals shadows of the teapot's handle and knob on its body. These shadows, which are really due to one level of radiance transfer at these areas, cannot be generated by prefiltered environment map based approaches unless specific provisions are made to encode the radiance transfer function [59, 109]. Furthermore, since prefiltered environment map based approaches are only valid for the chosen centre of projection, concave objects such as the teapot will display obvious lighting inaccuracy in their concavities. OSLFs do not suffer from this flaw because local lighting is computed based on the position in space of the surface point being shaded.

6.6 Texture Cache

This section presents statistics for the texture cache. All images were rendered on a dual Intel *Xeon* 2 GHz processor, with 2 GBs of shared main memory, an Adaptec AIC7899 Ultra160 SCSI adapter, and a Quantum *Atlas* WLS 10K3 10,000 RPM 36 GBs Ultra160 SCSI drive. *BMRT*, however, is a single threaded renderer. Timings are in hh:mm:ss format which indicate actual processor time used and were obtained using *BMRT*'s statistics output option.

Table 6.3 shows the statistics for the sample map cache. These timings only reflect the time spent indexing into the sample maps and does not take in account the rest of the shading process. These results show that sample map indexing is not a bottleneck, even though the maps are stored on hard disk.

Cache Size	*Time*	*Miss Rate*
2	00:00:28	71.80%
4	00:00:28	59.41%
8	00:00:27	44.03%
16	00:00:27	31.65%
32	00:00:27	22.20%
64	00:00:27	15.60%

Table 6.3: Sample map cache statistics.

Cache Size	*Time*	*Miss Rate*
0	00:07:26	100.00%
2	00:01:51	24.92%
4	00:01:30	18.67%
6	00:01:22	16.43%
8	00:01:19	15.51%
16	00:01:14	13.99%
32	00:01:11	13.15%

Table 6.4: Cube map cache statistics for surface reconstruction.

Table 6.4 shows the statistics for surface reconstruction using the read on demand strategy. As expected, the rendering times decrease as the size of the cache increases. Even with a two item cache, the cube map cache provides an impressive improvement in the rendering times. However, these results show that there is not much benefit going beyond a 16 item cache as the rendering times start to level out.

Cache Size	*Time*	*Miss Rate*
0	00:27:19	100.00%
4	00:04:04	16.71%
8	00:02:13	8.32%
16	00:01:34	5.19%
32	00:01:19	4.08%

Table 6.5: Cube map cache statistics for specular reflections.

Table 6.5 shows the statistics for specular reflection using the read on demand strategy. Only $L_{specular}$ is computed in this case. The effects of increasing the cache size are seen more dramatically here because generating reflections involves many more cube map accesses compared to reconstructing the object's surface. Again, there is not much improvement going from a 16 item cache to a 32 item cache.

M	n_u	n_v	*Cache Size*	*Time*	*Miss Rate*
100	10	10	0	07:36:10	100.00%
			8	00:07:27	0.95%
			16	00:04:45	0.46%
			32	00:04:00	0.32%
100	100	100	0	08:58:10	100.00%
			8	00:04:03	0.30%
			16	00:03:30	0.21%
			32	00:03:21	0.17%
100	1000	1000	0	09:18:11	100.00%
			8	00:03:21	0.20%
			16	00:03:08	0.16%
			32	00:03:05	0.14%

Table 6.6: Cube map cache statistics for Ashikhmin and Shirley's BRDF model.

Table 6.6 shows the statistics for evaluating the Ashikhmin and Shirley BRDF model for the teapot surface using the read on demand strategy. Again, only $L_{specular}$ is computed. A huge improvement in rendering time is obtained when using a cache size of eight items compared to not using a cache. However, the improvements in rendering time when increasing the cache size beyond eight items are not as dramatic. These results also show that a 16 item cache seems to be sufficient for this scene.

Finally, Table 6.7 compares the various image reading strategies. The read full image strategy performs the best. Since most graphics APIs, such as *OpenGL* [103] and *DirectX* [82], manage textures on an image level (although individual driver implementations may do something different), these texture cache statistics should be similar when the

Image Reading Strategy	*Cache Size*	*Time*	*Miss Rate*
Read on demand	0	09:19:59	100.00%
	8	00:05:19	0.59%
	16	00:04:26	0.35%
	32	00:04:08	0.29%
Read half threshold	32	00:03:00	0.14%
Read full image	32	00:02:45	0.05%

Table 6.7: Cube map cache statistics for various image reading strategies.

algorithm is migrated to a hardware-accelerated platform.

6.7 Summary

Offset surface light fields can reproduce the original surface's diffuse material accurately if the free space precondition is met. In addition, reflections generated with OSLFs are more accurate than the environment map approach but less accurate than ray tracing. However, OSLFs still have some disturbing visual discontinuities even though their measured error is lower. OSLFs can model one level of radiance transfer (e.g. object self-reflection) in a single pass, which is generally missing in environment map approaches. However, ray tracing approaches do better by allowing multi-level self-reflection. For OSLFs, the choice between which blending function to use is a choice between trading off reflection accuracy against visual smoothness. OSLFs can also evaluate local illumination efficiently by using the cube maps as an illumination cache. Furthermore, direct object self-shadowing can be reproduced accurately if the shadows were preacquired. Indirect object self-shadowing, due to local illumination, is generated for free because evaluating the rendering equation already takes this effect into account. These effects are not possible with environment map based approaches. Although there was insufficient time to implement a ray tracing, Monte Carlo evaluator for the Ashikhmin and Shirley BRDF model in the *RenderMan* shading language, the reasoning in Section 6.5 can convince us that this method will be more accurate than single point of projection approaches. The statistics in Section 6.6 further show that this approach is texture cache friendly. Lastly, the downsides of OSLFs compared to ray tracing are counterbalanced by the fact that OSLFs can be used for real-time rendering whereas ray tracing approaches currently cannot. Real-time rendering of OSLFs are, however, constrained by memory usage.

Chapter 7

Hardware Implementation

"All processors aspire to be general-purpose", *Tim van Hook*.

This chapter sketches a hardware-accelerated implementation for offset surface light fields. At the time of this writing, the NVIDIA *GeForce FX* seems to be a promising candidate as an implementation platform. However, cards based on this GPU are only slated for launch next year. My algorithm targets a GPU with more general programming features than those available today. The trend in GPU design is to move from fixed-function graphics pipelines, with feature-based interfaces, to those with limited programmability and assembly-language like interfaces. Upcoming graphics hardware will push this trend even further to more general purpose architectures, with higher precision, a convergence of vertex and fragment pipelines [5], along with high-level programming interfaces.

I implemented my software shader using *RenderMan*'s shading language API version 3.1 and C++, and tested it using *BMRT*. Note that the shader is fully implementable using the 3.2 version API. However, since *BMRT* does not support this version (specifically the array data type), I had to revert to writing portions of code using *RenderMan*'s C/C++

DSO framework. As Proudfoot et al. have already demonstrated a partial implementation of the *RenderMan* shading language on current GPUs [94], I expect future GPU and compiler advancements will be able to compile my shader to a hardware-accelerated platform. Current high-level shading languages such as ATI's *RenderMonkey* [20], Microsoft's *DirectX 9 HLSL* (High-Level Shading Language) [83], NVIDIA's *Cg* [91], and *OpenGL* 2.0's shading language [1], already show some of this progress.

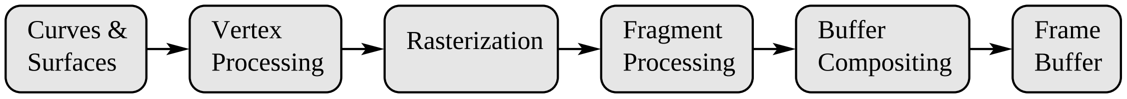


Figure 7.1: General architecture for GPU rendering pipelines.

The general architecture for GPUs today is shown in Figure 7.1 [80]. High-level surface descriptions are converted into triangles which can then be modified by per-vertex operations. Next, the triangles are rasterized and passed on to the fragment processor where texture operations and per-pixel shading is performed. Finally, the results are combined in the compositing stage and written to the frame buffer for display.

OSLFs should be easy to implement on a capable hardware-accelerated platform because of its simple parameterization. My implementation does most of its work in the fragment processing stage, at the pixel level. The Bézier patches passed into the rendering pipeline are tessellated (e.g. with the `GL_NV_evaluators` *OpenGL* extension) and rasterized. At the fragment processing stage, a texture shader looks up the four 16-bit indices in the sample map corresponding to the patch being shaded. These indices are in turn used to look up the four cube maps which store the light field at the corresponding OSLF sample points. This step is referred to as dependent texturing because the results of one texture map are used to index into another texture map. Note that these steps can be compressed into one three-dimensional dependent texture look up but with a slightly more complex implementation. To reconstruct the surface, the cube maps are indexed in the direction of the surface point. To construct a mirror reflection, the cube maps are indexed in the reflection direction. The results are then combined using blending basis functions and accumulated with the results of the reconstruction stage. To apply a BRDF, a loop construct is needed which generates random vectors to evaluate the local illumination using Monte Carlo integration about the sample points. The results are then combined using the blending basis functions and accumulated with the results of the reconstruction stage.

There are many complications in implementing this algorithm on current hardware. First, sample maps are stored in 16-bit integer format and cube maps in 32-bit floating-point format so the rendering pipeline must be able to handle and perform arithmetic operations with this level of precision throughout the pipeline. To my knowledge, only the upcoming NVIDIA *GeForce FX* has this capability. Second, a more general instruction set is needed at the fragment processing stage. At this stage, the sample map look up returns 16-bit indices which are used to look up into a 32-bit cube map. Current hardware and the *GeForce FX*, however, cannot index into arrays of textures but this can be partially remedied by storing the acquired environment maps, as dual parabolic maps, into a large texture. Then, indexing into the array of cube maps would be replaced by indexing into a sub-square of a square texture. To perform the environment map look up, the texture shader needs the shaded point's coordinates and the stored coordinates of the OSLF sample points corresponding to the cube maps. The shaded point's coordinates are usually obtained by interpolating between the triangle's three vertices which are generated at tessellation time. The OSLF sample point coordinates can be obtained using another dependent texture look up to a texture which stores XYZ point coordinates in RGB channels. Next, a point subtraction is performed and the resulting vector used to bilinearly interpolate into the cube maps. However, to perform the horizon angle check described in Section 6.3, a dot product of the shaded point's normal (obtained by an interpolation step similar to the shaded point's coordinates) and the index vector must be computed. The result of this dot product can be stored in a four component vector or in temporary GPU registers. Then the fragment program needs to check each component of the vector to see if it satisfies the angle of tolerance criteria—returning a one if it does and a zero otherwise into another four component vector. The program then scans this vector to find the closest sample point which satisfies the angle of tolerance and uses it to reconstruct the surface. The use of SIMD instruction sets [68] accelerate the calculations at this stage. Similar operations are needed for generating specular reflections. Furthermore, the fragment program needs to calculate the basis functions for each sample point and blend the results of the cube map look up. The specular reflection results are then combined with the surface reconstruction results using presently available combiners (e.g. the `GL_NV_register_combiners*` extensions). For BRDF and local lighting evaluation, a general loop construct is needed to accumulate the results of Monte Carlo integration. Support for mathematical functions, such as $\arctan()$ and $\arccos()$, are also needed to evaluate the Ashikhmin and Shirley BRDF. For current hardware, however, a popular approach to calculate these functions is to approximate them with a texture look up into a precomputed look up table. Although a pseudo-random number generator can be implemented in a fragment program with appropriate operations, a more likely approach is to store a precomputed set of random numbers in a texture map and use that instead.

Unfortunately, current pixel shader extensions, such as the `GL_NV_texture_shader*` extensions, do not have the capability (and instruction sets) to perform the operations described above. However, I am confident that OSLFs will be implementable in future GPU designs as more general purpose instruction sets appear. This will probably be true as functionality from vertex and fragment pipelines are combined, as GPUs evolve towards more generality. One issue that might affect the performance of the hardware-accelerated algorithm is the cost of transferring textures down the AGP bus to texture memory. However, architectures which incorporate a virtual memory system, such as the 3Dlabs *Wildcat VP* [2], will address this performance issue as it can make more efficient use of memory and reduce texture swapping. Because of its relatively simple implementation and potential for SIMD hardware-acceleration, OSLFs should be able to render at real-time frame rates.

Chapter 8

Conclusion

In this thesis I have presented a new parameterization of the lighting surrounding an object. This offset surface light field representation allows for an accurate reconstruction of the original object's diffuse surface light field. Furthermore, additional reflections can be incorporated into the surface's material with increased accuracy over traditional reflection mapping approaches. Local illumination for points on the surface can be evaluated efficiently using the OSLFs as an illumination cache and, if further desired, a completely new bidirectional reflectance distribution function can be imposed onto the surface. The simplicity of the algorithm and its texture cache friendliness lends itself to a real-time, hardware-accelerated implementation. Such an implementation was sketched out for future general-purpose graphics processors.

This new technique can be potentially used for visualizing various data sets in the areas such as architecture, automotive design, industrial design, three-dimensional photography, museum display, motion picture production, and game design. Although not as accurate as ray tracing approaches, a high-performance hardware-accelerated implementation will allow real-time display rates and interactive manipulation not achievable using traditional software based methods, although the approach presented is limited to fixed illumination. Finally, the image-based roots of this approach allow it to accurately portray effects from the physical world, which can be difficult to do using current algorithms.

8.1 Future Work

OSLFs are based on work from reflection mapping and image-based rendering. Future research topics to pursue include applying various refinements from those areas to this parameterization and extending the class of renderings possible within the OSLF paradigm.

8.1.1 Refining OSLFs

Basis Functions

I have presented basis functions which trade off smoothness for accuracy and vice versa. Perhaps a blend of these two basis functions or a completely new basis function would produce more accurate and visually smoother results.

Sensitivity Analysis

Besides sample map resolution, cube map resolution, and surface sampling density, the accuracy of surface reconstruction and reflection sampling is sensitive to the distance of OSLF sample points from the surface, ϵ, and the angles of tolerance, ϕ_{shader} and

$\phi_{samplemap}$. Perhaps further analysis of these parameters will yield a basis function which incorporates them to produce a better blending function.

Sampling Issues and Real World Data

Currently, sampling density is based on the simple criteria of finding enough samples to fill the sample map given $\phi_{samplemap}$. This does not necessarily ensure that the sample points chosen are the best possible representations for the texel considered. An analysis of surface curvature could potentially produce a better sampling pattern than the current one. Moreover, OSLFs should be tested with data sets acquired from the physical world. This would involve creating a rig which can record position accurately, within a small tolerance, attached with a small rotating camera to acquire environment maps—the tiny spy cameras come to mind. Gonioradiometers can also be adapted for acquiring physical data.

Surface Reconstruction

I have presented an algorithm to reconstruct surfaces which is only valid for objects with a diffuse texture. However, since the OSLF implicitly stores the object's SLF for a fixed number of outgoing vectors, we can roughly reconstruct a specular object's reflectance properties by adapting Cabral et al.'s [18] method or by applying a blending function which favours the viewing direction, $\vec{v}$. In this case, data acquisition will involve rendering the object's surface with the same BRDF to be used in the final real-time rendering, and with reflections as required. However, it is to be determined how well these methods will work.

Compression and Storage

Light fields and SLFs have been shown to be readily compressible into forms which allow random access [24, 25, 84, 125]. OSLFs contain similar data redundancy and should be compressible using previously known methods. Since OSLFs contain less data than SLFs, the resulting data set should be smaller than an equivalent SLF data set. Furthermore, representations other than cube maps can be used to store the OSLFs. For example, it might be worthwhile to try storing the OSLFs in spherical harmonic form as in Ramamoorthi and Hanrahan's spherical harmonic reflection maps (SHRMs) [98]. This would allow more efficient evaluation of local illumination in the frequency domain. However, using their SHRMs naïvely would result in an explosion of storage requirements, as they store nine spherical harmonic coefficients per cube map texel. Furthermore, their SHRMs are only valid for prefiltering with isotropic BRDFs. Increasing the order of SHRMs to handle high-frequency lighting is probably not a good idea as the number of basis functions increases rapidly.

8.1.2 Extending OSLFs

Bump Mapping

Bump mapping is a technique to simulate surface detail without increasing the number of triangles rendered. It relies on normals stored in a texture map to define bumps on the surface. A small modification to the lighting calculation simulates the appearance of bumps. Since all the information required for this calculation is available at shader runtime, this technique can also be applied within the OSLF framework.

Multiple Levels of Reflection

As mentioned before, the specular reflection generation algorithm can be applied recursively to produce multiple levels of reflection. Specifically, we can reacquire cube maps at the OSLF sample points after the initial reflection is calculated. By using these new cube maps to perform another specular reflection calculation, we obtain an additional level of reflection.

BRDFs and Interactive Materials

Various BRDFs can be used when evaluating local illumination at runtime. However, if the GPU does not have the capabilities to evaluate complicated BRDFs, then another approach is to preintegrate the cube maps with the BRDF (as discussed in Section 2.1.2) and then sample from these prefiltered cube maps. This would produce smoother results, making the sample map structure less visible. The general idea of this approach is similar to Cabral et al.'s algorithm [18] except our evaluation process is much simpler.

If the BRDF can be evaluated at runtime, we have the option of completely replacing the surface material and letting users edit its parameters interactively. This is possible by either ignoring the acquired SLF or by acquiring an object with a purely diffuse material and incorporating a specular BRDF on top of it. Another possibility is to acquire the object with a modified blue-screen technique which displays one colour where the object is completely illuminated and another where the object is in shadow (a special shader can accomplish this for virtual data sets). Evaluating local illumination at runtime then allows for interactive manipulation of the surface material.

Object Replacement

Instead of changing the surface material, we can completely replace the object. To do this, we densely sample the space which the object is to reside in. Then preprocessing will create sample maps for a number of objects using only relevant sample points. Rendering different objects just involves switching between the different object geometries and sample maps. This technique, however, cannot produce self-reflection and self-shadowing effects unless special provisions are made to handle them. However, reflections and local illumination evaluation will remain reasonably accurate. This idea is related to Greger et al.'s irradiance volumes [44].

Relighting

We can perform simple manipulation of the lighting environment by rotating the cube maps before indexing. This can be done by changing the associated texture matrix at runtime. To get around the fixed illumination assumption, it is possible to add additional light sources to illuminate the object in real-time, as this data is readily available at shader runtime and already hardware accelerated. This is almost the same approach used by Neto and Bishop's software implementation [29]. One problem with doing this is that the added lights do not affect the environment in the preacquired cube maps. However, this problem is about reproducing a second-order effect. All first-order effects will be captured correctly.

Thin Lens Refractions

Refractions can be simulated with OSLFs. Instead of indexing in the reflection direction, we simply index in the refraction direction. Refractions generated this way, however, are technically only valid for thin surfaces where the thin lens approximation holds [15] and where there are no internal reflections.

Bibliography

[1] 3Dlabs. 3Dlabs' OpenGL 2.0 specifications and white papers resource site. http://www.3dlabs.com/support/developer/ogl2/index.htm, June 2002. 3, 76

[2] 3Dlabs. 3Dlabs P10 Visual Processing Unit and Wildcat VP documentation. http://www.3dlabs.com/support/developer/index.htm, June 2002. 2, 79

[3] Edward H. Adelson and James R. Bergen. The plenoptic function and the elements of early vision. In M. Landy and J. A. Movshon, editors, *Computation Models of Visual Processing*, chapter 1, pages 3–20. MIT Press, Cambridge, MA, 1991. 17

[4] Kurt Akeley. RealityEngine graphics. In James T. Kajiya, editor, *Proceedings of SIGGRAPH 1993*, Computer Graphics Proceedings, Annual Conference Series, pages 109–116. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1993. Anaheim, CA, Aug 1-6, 1993. 2

[5] Kurt Akeley and Pat Hanrahan. Real-time graphics architectures. CS448A Course Slides, September 2001. http://www.graphics.stanford.edu/courses/cs448a-01-fall/. 75

[6] Tomas Akenine-Möller and Eric Haines. *Real-Time Rendering*. A K Peters, Ltd, Natick, MA, second edition, 2002. 2

[7] Anthony A. Apodaca and Larry Gritz. *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann, New York, NY, 2000. 2

[8] Michael Ashikhmin and Peter Shirley. An anisotropic Phong BRDF model. *Journal of Graphics Tools*, 5(2):22–32, 2002. 9, 49, 69

[9] Harlyn H. Baker and Robert C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1988*, pages 2–9, Washington, D.C., June 1988. SRI, Computer Society Press. Ann Arbour, MI, June 5-9, 1988. 20

[10] David C. Banks. Illumination in diverse codimensions. In Andrew S. Glassner, editor, *Proceedings of SIGGRAPH 1994*, Computer Graphics Proceedings, Annual Conference Series, pages 327–334. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 1994. Orlando, FL, July 24-29, 1994. 9, 14

[11] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, NY, 1995. 23

[12] James F. Blinn. Models of light reflection for computer synthesized pictures. In James George, editor, *Proceedings of SIGGRAPH 1977*, Computer Graphics Proceedings, Annual Conference Series, pages 192–198. ACM SIGGRAPH, ACM Press, July 1977. San Jose, CA, July 20-22, 1977. 9

[13] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(1):542–546, October 1976. 3, 11

[14] Robert C. Bolles and Harlyn H. Baker. Epipolar-plane image analysis: A technique for analyzing motion sequences. In L. S. Baumann, editor, *Proceedings: Image Understanding Workshop*, pages 137–148, San Mateo, CA, 1985. DARPA / SRI, Morgan Kaufmann. Miami Beach, FL, December 9-10, 1985. 20

[15] Max Born and Emil Wolf. *Principles of Optics*. Cambridge University Press, Cambridge, UK, seventh edition, 1999. 86

[16] Phong Bui-Tuong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, June 1975. 9, 14, 42

[17] Brian Cabral, Nelson Max, and Rebecca Springmeyer. Bidirectional reflection functions from surface bump maps. In Maureen C. Stone, editor, *Proceedings of SIGGRAPH 1987*, Computer Graphics Proceedings, Annual Conference Series, pages 273–281. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 1987. Anaheim, CA, July 27-31, 1987. 10, 14

[18] Brian Cabral, Marc Olano, and Philip Nemec. Reflection space image based rendering. In Alyn Rockwood, editor, *Proceedings of SIGGRAPH 1999*, Computer Graphics Proceedings, Annual Conference Series, pages 165–170. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1999. Los Angeles, CA, August 8-13, 1999. 10, 14, 20, 82, 84

[19] Emilio Camahort, Apostolos Lerios, and Don Fussell. Uniformly sampled light

fields. In George Drettakis and Nelson Max, editors, *Rendering Techniques '98*, Proceedings of the Eurographics Workshop on Rendering, pages 117–130, New York, NY, June 1998. Eurographics, Springer Wien. Vienna, Austria, June 29-July 1, 1998. 19

[20] Drew Card and Jason L. Mitchell. RenderMonkey. SIGGRAPH 2002 presentation slides, August 2002. http://www.ati.com/developer/techpapers.html. 76

[21] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 307–318. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 2000. New Orleans, LA, July 23-28, 2000. 20

[22] Shenchang Eric Chen. QuickTime VR: An image-based approach to virtual environment navigation. In Robert L. Cook, editor, *Proceedings of SIGGRAPH 1995*, Computer Graphics Proceedings, Annual Conference Series, pages 29–38. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1995. Los Angeles, CA, August 6-11, 1995. 16, 17

[23] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In James T. Kajiya, editor, *Proceedings of SIGGRAPH 1993*, Computer Graphics Proceedings, Annual Conference Series, pages 279–288. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1993. Anaheim, CA, Aug 1-6, 1993. 16

[24] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk.

Light field mapping: Efficient compression and hardware rendering of surface light fields. *ACM Transactions on Graphics*, 21(3):447–456, July 2002. SIGGRAPH 2002, San Antonio, TX, July 21-26, 2002. 22, 83

[25] Wei-Chao Chen, Radek Grzeszczuk, and Jean-Yves Bouguet. Light field mapping: Hardware-accelerated visualisation of surface light fields. In Eugene Fiume, editor, *SIGGRAPH 2001 Course 46 Notes*, Computer Graphics Proceedings, Annual Conference Series, pages 410–416. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press, August 2001. Los Angeles, CA, August 12-17, 2001. 83

[26] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In Hank Christiansen, editor, *Proceedings of SIGGRAPH 1984*, Computer Graphics Proceedings, Annual Conference Series, pages 137–145. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 1984. Minneapolis, MN, July 23-27, 1984. 10

[27] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1):7–24, January 1982. 9

[28] Scott Daly. The visible differences predictor: An algorithm for the assessment of image fidelity. In Andrew B. Watson, editor, *Digital Images and Human Vision*, pages 179–206. MIT Press, Cambridge, MA, 1993. 59

[29] Manuel M. de Oliveira Neto and Gary Bishop. Dynamic shading in image-based rendering. Technical Report TR98-023, University of North Carolina at Chapel Hill, May 1998. 20, 85

[30] Paul Debevec. Eucalyptus grove light probe. http://www.debevec.org/Probes/, 1998. 29

[31] Michael Deering and David Naegle. The SAGE graphics architecture. *ACM Transactions on Graphics*, 21(3):683–692, July 2002. SIGGRAPH 2002, San Antonio, TX, July 21-26, 2002. 2

[32] Philip Dutré, Eric P. Lafortune, and Yves D. Willems. Monte Carlo light tracing with direct computation of pixel intensities. In *Proceedings of Compugraphics '93*, pages 128–137, December 1993. Alvor, Portugal. 10

[33] Matthias Eck, Tony DeRose, Tom Duchamp, Hughes Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In Robert L. Cook, editor, *Proceedings of SIGGRAPH 1995*, Computer Graphics Proceedings, Annual Conference Series, pages 173–182. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1995. Los Angeles, CA, August 6-11, 1995. 22, 26

[34] Gordon Elder. Radeon 9700. In Thomas Ertl, editor, *Hot3D Presentations in Graphics Hardware 2002*, ACM SIGGRAPH / Eurographics Workshop on Graphics Hardware. ACM SIGGRAPH / Eurographics, ACM SIGGRAPH / ACM Press, August 2002. Saarbrüken, Germany, September 1-2, 2002. 2

[35] Gerald E. Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Academic Press, Toronto, ON, fourth edition, 1998. 27

[36] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Com-*

puter Graphics, Principles and Practice. Addison-Wesley, Don Mills, ON, second edition, 1996. Second Edition in C. 9, 13

[37] Alain Fournier. Separating reflection functions for linear radiosity. In Pat Hanrahan and Werner Purgathofer, editors, *Rendering Techniques '95*, Proceedings of the Eurographics Workshop on Rendering, pages 383–392, New York, NY, June 1995. Eurographics, Springer Wien. Dublin, Ireland, June 12-14, 1995. 10

[38] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, MA, 1992. 19

[39] Bernd Girod. What's wrong with mean-squared error? In Andrew B. Watson, editor, *Digital Images and Human Vision*, pages 207–220. MIT Press, Cambridge, MA, 1993. 60

[40] Andrew S. Glassner. *An Introduction to Ray Tracing*. Academic Press, San Diego, CA, 1989. 2, 3, 10

[41] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley, Don Mills, ON, 1992. 20

[42] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In Holly Rushmeier, editor, *Proceedings of SIGGRAPH 1996*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1996. New Orleans, LA, August 4-9, 1996. 19, 41, 64

[43] Ned Greene. Applications of world projections. In *Proceedings of Graphics Interface 1986*, pages 108–114. Canadian Information Processing Society, Canadian Human-Computer Communications Society / Morgan Kaufmann, May 1986. Vancouver, BC, May 26-30, 1986. 13

[44] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, March 1998. 85

[45] Daniel Hall. The AR350: Today's ray trace rendering processor. In Peter N. Glaskowsky, editor, *Hot3D Presentations in Graphics Hardware 2001*, ACM SIGGRAPH / Eurographics Workshop on Graphics Hardware, pages 13–19. ACM SIGGRAPH / Eurographics, ACM SIGGRAPH / ACM Press, August 2001. Los Angeles, CA, August 12-13, 2001. 3

[46] Xiao D. He, Kenneth E. Torrance, François X. Sillion, and Donald P. Greenberg. A comprehensive physical model for light reflection. In Thomas W. Sederberg, editor, *Proceedings of SIGGRAPH 1991*, Computer Graphics Proceedings, Annual Conference Series, pages 175–186. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 1992. Las Vegas, NV, July 28-August 2, 1991. 9

[47] Wolfgang Heidrich, Hendrik P. A. Lensch, Michael F. Cohen, and Hans-Peter Seidel. Light field techniques for reflections and refractions. In Dani Lischinski and Gregory Ward Larson, editors, *Rendering Techniques '99*, Proceedings of the

Eurographics Workshop on Rendering, pages 187–196, New York, NY, June 1999. Eurographics, Springer Wien. Granada, Spain, June 21-23, 1999. 20

[48] Wolfgang Heidrich, Hartmut Schirmacher, Hendrik Kück, and Hans-Peter Seidel. A warping-based refinement of lumigraphs. Technical Report 20, Universität Erlangen-Nürnberg, 1998. 19, 41

[49] Wolfgang Heidrich and Hans-Peter Seidel. View-independent environment maps. In Arie Kaufman and Wolfgang Strasser, editors, *Proceedings of SIGGRAPH / Eurographics Workshop on Graphics Hardware*, ACM SIGGRAPH / Eurographics Workshop on Graphics Hardware, pages 39–45. ACM SIGGRAPH / Eurographics, ACM SIGGRAPH / ACM Press, August 1998. Lisbon, Portugal, August 31-September 1, 1998. 13, 20

[50] Wolfgang Heidrich and Hans-Peter Seidel. Realistic, hardware-accelerated shading and lighting. In Alyn Rockwood, editor, *Proceedings of SIGGRAPH 1999*, Computer Graphics Proceedings, Annual Conference Series, pages 171–178. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1999. Los Angeles, CA, August 8-13, 1999. 2, 10, 14

[51] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 297–306. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 2000. New Orleans, LA, July 23-28, 2000. 19

[52] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A K Peters, Ltd, Natick, MA, 2001. 2, 11

[53] Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bi-directional Monte Carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, March 1995. 11

[54] James T. Kajiya. Anisotropic reflection models. In Brian A. Barsky, editor, *Proceedings of SIGGRAPH 1985*, Computer Graphics Proceedings, Annual Conference Series, pages 15–21. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 1985. San Francisco, CA, July 22-26, 1985. 9

[55] James T. Kajiya. The rendering equation. In David C. Evans and Russell J. Athay, editors, *Proceedings of SIGGRAPH 1986*, Computer Graphics Proceedings, Annual Conference Series, pages 143–150. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1986. Dallas, TX, August 18-22, 1986. 2, 7, 9, 10

[56] Ramakrishna Karedla, J. Spencer Love, and Bradley G. Wherry. Caching strategies to improve disk system performance. *IEEE Computer*, 27(3):38–46, March 1994. 46

[57] Jan Kautz and Michael D. McCool. Interactive rendering with arbitrary BRDFs using separable approximations. In Dani Lischinski and Gregory Ward Larson, editors, *Rendering Techniques '99*, Proceedings of the Eurographics Workshop on Rendering, pages 281–292, New York, NY, June 1999. Eurographics, Springer Wien. Granada, Spain, June 21-23, 1999. 10

[58] Jan Kautz and Michael D. McCool. Approximation of glossy reflection with prefiltered environment maps. In *Proceedings of Graphics Interface 2000*, pages 119–126. Canadian Information Processing Society, Canadian Human-Computer Communications Society / Morgan Kaufmann, May 2000. Montreal, PQ, May 15-17, 2000. 10, 14

[59] Jan Kautz, Peter-Pike Sloan, and John Synder. Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In Paul Debevec and Simon Gibson, editors, *Rendering Techniques '02*, Proceedings of the Eurographics Workshop on Rendering, pages 301–308, New York, NY, June 2002. Eurographics, Springer Wien. Pisa, Italy, June 26-28, 2002. 15, 70

[60] Jan Kautz, Pere-Pau Vázquez, Wolfgang Heidrich, and Hans-Peter Seidel. A unified approach to prefiltered environment maps. In Bernard Péroche and Holly Rushmeier, editors, *Rendering Techniques '00*, Proceedings of the Eurographics Workshop on Rendering, pages 185–196, New York, NY, June 2000. Eurographics, Springer Wien. Brno, Czech Republic, June 26-28, 2000. 8, 10, 14

[61] Eric P. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In Turner Whitted, editor, *Proceedings of SIGGRAPH 1997*, Computer Graphics Proceedings, Annual Conference Series, pages 117–126. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1997. Los Angeles, CA, August 3-8, 1997. 10, 16

[62] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceed-*

ings of Compugraphics '93, pages 145–153, December 1993. Alvor, Portugal. 10

[63] Paul Lalonde and Alain Fournier. Interactive rendering of wavelet projected light fields. In *Proceedings of Graphics Interface 1999*, pages 170–114. Canadian Information Processing Society, Canadian Human-Computer Communications Society / Morgan Kaufmann, June 1999. Kingston, ON, June 2-4, 1999. 19

[64] Lutz Latta and Andreas Kolb. Homomorphic factorization of BRDF-based lighting computation. *ACM Transactions on Graphics*, 21(3):509–516, July 2002. SIGGRAPH 2002, San Antonio, TX, July 21-26, 2002. 23

[65] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In Michael F. Cohen, editor, *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, pages 95–104. ACM SIGGRAPH, ACM Press, July 1998. Orlando, FL, July 19-24, 1998. 22, 26

[66] David D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999. 23

[67] Marc Levoy and Pat Hanrahan. Light field rendering. In Holly Rushmeier, editor, *Proceedings of SIGGRAPH 1996*, Computer Graphics Proceedings, Annual Conference Series, pages 31–42. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1996. New Orleans, LA, August 4-9, 1996. 18, 41

[68] Erik Lindholm, Mark J. Kilgard, and Henry Moreton. A user-programmable ver-

tex engine. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 149–157. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press, August 2001. Los Angeles, CA, August 12-17, 2001. 2, 78

[69] Dani Lischinski and Ari Rappoport. Image-based rendering for non-diffuse synthetic scenes. In George Drettakis and Nelson Max, editors, *Rendering Techniques '98*, Proceedings of the Eurographics Workshop on Rendering, pages 301–314, New York, NY, June 1998. Eurographics, Springer Wien. Vienna, Austria, June 29-July 1, 1998. 19

[70] Jeffery Lubin. A visual discrimination model for image system design and evaluation. In Eli Peli, editor, *Vision Models for Target Detection and Recognition*, pages 245–283. World Scientific, New Jersey, NY, 1995. 59

[71] Thomas Murray MacRobert. *Spherical Harmonics; An Elementary Treatise on Harmonic Functions, with Applications*. Pergamon Press, New York, NY, 1967. 14

[72] Marcus Magnor and Bernd Girod. Adaptive block-based light field coding. In *Proceedings of the 3rd International Workshop on Synthetic and Natural Hybrid Coding and 3-D Imaging 1999*, pages 140–143, September 1999. Santorini, Greece, September 1999. 19

[73] Marcus Magnor and Bernd Girod. Hierarchical coding of light fields with disparity maps. In *Proceedings of the IEEE International Conference on Image Processing*

1999, volume 3, pages 334–338, October 1999. Kobe, Japan, October, 1999. 19, 20

[74] Marcus Magnor and Bernd Girod. Data compression for light field rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):338–343, April 2000. 19

[75] Stephen Mann and Tony DeRose. Computing values and derivatives of Bézier and B-spline tensor products. *Computer Aided Geometric Design*, 12(1):107–110, February 1995. 27, 60

[76] Brian Marshall. DirectX graphics future. Meltdown 2001 Conference, July 2001. http://www.microsoft.com/mscorp/corpevents/meltdown2001/ppt/DXG9.ppt. 3

[77] David K. McAllister, Anselmo Lastra, and Wolfgang Heidrich. Efficient rendering of spatial bi-directional reflectance distribution functions. In Wolfgang Heidrich and Michael Doggett, editors, *Proceedings of SIGGRAPH / Eurographics Workshop on Graphics Hardware*, ACM SIGGRAPH / Eurographics Workshop on Graphics Hardware, pages 79–88. ACM SIGGRAPH / Eurographics, ACM SIGGRAPH / ACM Press, September 2002. Saarbrüken, Germany, September 1-2, 2002. 16

[78] Michael D. McCool. SMASH: A next-generation API for programmable graphics accelerators. Technical Report CS-2000-14, University of Waterloo, August 2000. 2

[79] Michael D. McCool, Jason Ang, and Anis Ahmed. Homomorphic factorization

of BRDFs for high-performance rendering. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 171–178. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press, August 2001. Los Angeles, CA, August 12-17, 2001. 10, 23

[80] Michael D. McCool and Mauro Steigleder. Graphics accelerators: State of the art. University of Waterloo Institute for Computer Research talk slides, November 2002. http://www.cgl.uwaterloo.ca/Projects/rendering/Talks/. 76

[81] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In Robert L. Cook, editor, *Proceedings of SIGGRAPH 1995*, Computer Graphics Proceedings, Annual Conference Series, pages 39–46. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1995. Los Angeles, CA, August 6-11, 1995. 16, 17

[82] Microsoft Corporation. *DirectX 8.1 SDK Documentation*. Seattle, WA, November 2002. http://msdn.microsoft.com/directx/. 2, 48, 72

[83] Microsoft Corporation. *DirectX 9.0 SDK Documentation*. Seattle, WA, May 2002. http://www.betaplace.com/. 76

[84] Gavin Miller, Steven Rubin, and Dulce Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In George Drettakis and Nelson Max, editors, *Rendering Techniques '98*, Proceedings of the Eurographics Workshop on Rendering, pages 281–292, New York, NY, June 1998. Eurographics, Springer Wien. Vienna, Austria, June 29-July 1, 1998. 22, 83

[85] Gene S. Miller and Robert Hoffman. Illumination and reflection maps: Simulated objects in simulated and real environments. In Hank Christiansen, editor, *SIGGRAPH 1984 Course Notes: Advanced Computer Graphics Animation*, Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 1984. Minneapolis, MN, July 23-27, 1984. 14

[86] Steve Molnar, John Eyles, and John Poulton. PixelFlow: High-speed rendering using image composition. In James J. Thomas, editor, *Proceedings of SIGGRAPH 1992*, Computer Graphics Proceedings, Annual Conference Series, pages 231–240. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 1992. Chicago, IL, July 26-31, 1992. 2

[87] John Montrym and Henry Moreton. GeForce4. In Thomas Ertl, editor, *Hot3D Presentations in Graphics Hardware 2002*, ACM SIGGRAPH / Eurographics Workshop on Graphics Hardware. ACM SIGGRAPH / Eurographics, ACM SIGGRAPH / ACM Press, August 2002. Saarbrüken, Germany, September 1-2, 2002. 2

[88] John S. Montrym, Daniel R. Baum, David L. Dignam, and Christopher J. Migdal. InfiniteReality: A real-time graphics system. In Turner Whitted, editor, *Proceedings of SIGGRAPH 1997*, Computer Graphics Proceedings, Annual Conference Series, pages 293–302. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1997. Los Angeles, CA, August 3-8, 1997. 2

[89] Claus Müller. *Spherical Harmonics*. Springer-Verlag, New York, NY, 1966. 14

[90] László Neumann, Attila Neumann, and László Szirmay-Kalos. Compact metallic reflectance models. *Computer Graphics Forum*, 18(3):161–172, September 1999. Proceedings of Eurographics '99. 9, 51

[91] NVIDIA. *NVIDIA Cg Language Specification*. Santa Clara, CA, August 2002. http://developer.nvidia.com/view.asp?IO=cg_specification. 2, 76

[92] NVIDIA. *NVIDIA OpenGL Extension Specifications*. Santa Clara, CA, October 2002. http://developer.nvidia.com/view.asp?IO=nvidia_opengl_specs. 13

[93] Mark S. Peercy, Mark Olano, John Airey, and P. Jeffery Ungar. Interactive multi-pass programmable shading. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 425–432. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 2000. New Orleans, LA, July 23-28, 2000. 2

[94] Kekoa Proudfoot, William R. Mark, Svetoslav Tzvetkov, and Pat Hanrahan. A real-time procedural shading system for programmable graphics hardware. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 159–170. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press, August 2001. Los Angeles, CA, August 12-17, 2001. 2, 76

[95] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, 21(3):703–712, July 2002. SIGGRAPH 2002, San Antonio, TX, July 21-26, 2002. 2, 3

[96] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 497–500. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press, August 2001. Los Angeles, CA, August 12-17, 2001. 14

[97] Ravi Ramamoorthi and Pat Hanrahan. A signal processing framework for inverse rendering. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 117–128. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press, August 2001. Los Angeles, CA, August 12-17, 2001. 14, 20

[98] Ravi Ramamoorthi and Pat Hanrahan. Frequency space environment map rendering. *ACM Transactions on Graphics*, 21(3):517–526, July 2002. SIGGRAPH 2002, San Antonio, TX, July 21-26, 2002. 15, 83

[99] B. John William Strutt Rayleigh. On the law of reciprocity in diffuse reflexion. *Philosophical Magazine*, 49:324–325, 1900. Reprinted Scientific Papers, volume 4. Dover Publications, New York, NY, 1964. 9

[100] Matthew J. P. Regan, Gavin S. P. Miller, Steven M. Rubin, and Chris Kogeinik. A real-time low-latency hardware light-field renderer. In Alyn Rockwood, editor, *Proceedings of SIGGRAPH 1999*, Computer Graphics Proceedings, Annual Conference Series, pages 287–290. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1999. Los Angeles, CA, August 8-13, 1999. 19

[101] Hartmut Schirmacher, Wolfgang Heidrich, and Hans-Peter Seidel. High-quality

interactive lumigraph rendering through warping. In *Proceedings of Graphics Interface 2000*, pages 87–94. Canadian Information Processing Society, Canadian Human-Computer Communications Society / Morgan Kaufmann, May 2000. Montreal, PQ, May 15-17, 2000. 19, 41

[102] Christophe Schlick. An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum*, 13(3):233–246, September 1994. Proceedings of Eurographics '94. 9, 51, 52

[103] SGI. *OpenGL 1.4 Specification and OpenGL Extension Specifications*, 2002. http://www.opengl.org/developers/documentation/index.html. 20, 48, 72

[104] Peter Shirley. *Realistic Ray Tracing*. A K Peters, Ltd, Natick, MA, 2000. 2

[105] Yu A. Shreider. *The Monte Carlo Method*. Pergamon Press, Oxford, 1966. Also The Method of Statistical Trials (The Monte Carlo method), Fizmatgiz, Moscow, 1962. 10, 50

[106] François X. Sillion, James Avro, Stephen H. Westin, and Donald P. Greenberg. A global illumination solution for general reflectance distributions. In Thomas W. Sederberg, editor, *Proceedings of SIGGRAPH 1991*, Computer Graphics Proceedings, Annual Conference Series, pages 187–196. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 1992. Las Vegas, NV, July 28-August 2, 1991. 14

[107] Peter-Pike Sloan, Michael F. Cohen, and Steven J. Gortler. Time critical lumigraph rendering. In *Proceedings of 1997 Symposium on Interactive 3D Graphics*, pages

17–ff, New York, NY, April 1997. ACM SIGGRAPH, ACM Press. Providence, RI, April, 1997. 19

[108] Peter-Pike Sloan and Charles Hansen. Parallel lumigraph reconstruction. In *Proceedings of Parallel Visualization and Graphics Symposium 1999*, pages 7–14, October 1999. San Francisco, October, 1999. 19

[109] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21(3):527–536, July 2002. SIGGRAPH 2002, San Antonio, TX, July 21-26, 2002. 15, 70

[110] Mauro Steigleder and Michael D. McCool. Factorization of the Ashikhmin BRDF for real-time rendering. *Journal of Graphics Tools*, Special Issue on Hardware-Accelerated Rendering Techniques, 2002. To be published. 51

[111] Gordon Stoll, Matthew Eldridge, Dan Patterson, Art Webb, Steven Berman, Richard Levy, Chris Caywood, Milton Taveira, Stephen Hunt, and Pat Hanrahan. Lightning-2: A high-performance display subsystem for PC clusters. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 141–148. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press, August 2001. Los Angeles, CA, August 12-17, 2001. 2

[112] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, Toronto, ON, 1992. 46

[113] Gary Tarolli. Real-time cinematic effects on the PC: The 3Dfx T-buffer. In

Peter N. Glaskowsky, editor, *Hot3D Presentations in Graphics Hardware 1999*, ACM SIGGRAPH / Eurographics Workshop on Graphics Hardware. ACM SIGGRAPH / Eurographics, ACM SIGGRAPH / ACM Press, August 1999. Los Angeles, CA, August 8-9, 1999. 2

[114] Xin Tong and Robert M. Gray. Compression of light fields using disparity compensation and vector quantization. In M. H. Hamza, editor, *Proceedings of IASTED International Conference on Computer Graphics and Imaging 1999*, pages 300–305. ACTA Press, October 1999. Palm Springs, CA, October 1999. 19

[115] Kenneth E. Torrance and Ephraim M. Sparrow. Theory of off-specular peak reflection from roughened surfaces. *Journal of Optical Society of America*, 57(9):1105–1114, September 1967. 9

[116] Kenneth E. Torrance, Ephraim M. Sparrow, and R. C. Birkebak. Polarization, directional distribution, and off-specular peak phenomena in light reflected from roughened surfaces. *Journal of Optical Society of America*, 56(7):916–925, July 1966. 9

[117] Chris Trendall and A. James Stewart. General calculations using graphics hardware with application to interactive caustics. In Bernard Péroche and Holly Rushmeier, editors, *Rendering Techniques ’00*, Proceedings of the Eurographics Workshop on Rendering, pages 387–298, New York, NY, June 2000. Eurographics, Springer Wien. Brno, Czech Republic, June 26-28, 2000. 2

[118] Steve Upstill. *The RenderMan Companion: A Programmer's Guide to Realistic Computer Graphics*. Addison-Wesley, Don Mills, ON, 1990. 2, 12

[119] Eric Veach. Introduction to Monte Carlo integration. CS 448 Mathematical Models for Computer Graphics Lecture Notes, October 1997. http://www-graphics.stanford.edu/courses/cs448-97-fall/notes.html. 10, 50

[120] Eric Veach and Leonidas J. Guibas. Bi-directional estimators for light transport. In Georgios Sakas, Peter Shirley, and S. Müller, editors, *Rendering Techniques '94*, Proceedings of the Eurographics Workshop on Rendering, pages 147–162, New York, NY, June 1994. Eurographics, Springer Wien. Darmstadt, Germany, June 12-14, 1994. 10

[121] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In Turner Whitted, editor, *Proceedings of SIGGRAPH 1997*, Computer Graphics Proceedings, Annual Conference Series, pages 65–76. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, August 1997. Los Angeles, CA, August 3-8, 1997. 11

[122] Hermann von Helmholtz. *Helmholtz's Treatise on Physiological Optics*, volume 1. Dover Publications, New York, NY, 1962. Translated from the third German edition (1909), which includes the entire verbatim text of the first edition (1856). 9

[123] Gregory J. Ward. Measuring and modeling anisotropic reflection. In James J. Thomas, editor, *Proceedings of SIGGRAPH 1992*, Computer Graphics Proceedings, Annual Conference Series, pages 265–272. ACM SIGGRAPH, ACM SIG-

GRAPH / ACM Press / Addison-Wesley, July 1992. Chicago, IL, July 26-31, 1992. 9, 10, 51

[124] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 26(6):342–249, June 1980. 3, 10

[125] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3D photography. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 287–296. ACM SIGGRAPH, ACM SIGGRAPH / ACM Press / Addison-Wesley, July 2000. New Orleans, LA, July 23-28, 2000. 22, 83

[126] Günther Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & Sons, New York, NY, second edition, 1982. 59

[127] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3):337–343, May 1977. 19