Rendering Surfaces Under Photographically Acquired Illumination

Ron Dror

December 3, 1999

Abstract

I use images acquired by a camera with a fisheye lens to represent illumination incident from all directions at a point in the real world. Using the lighting visualization package *Radiance*, I render simple objects with a variety of surface properties under such complex illumination. I develop two different techniques to accomplish this task, each of which performs best in certain situations.

1 Introduction

The bulk of traditional computer graphics work assumed simple surface illumination models which took into account only direct lighting and limited indirect specular lighting. Recently, the development of radiosity and enhanced ray-tracing techniques has enabled renderings to take global illumination, including diffuse reflections from other surfaces, into account. Such techniques are still limited in their ability to simulate natural lighting because natural primary light sources such as a partially cloudy sky are difficult to model and because the global illumination of a single surface requires the accurate modeling of all surfaces visible from that point. In fact, to ensure that the illumination of a single surface is correct under such an approach, one must solve the global illumination problem for the entire surrounding world.

This project explores the alternative possibility of capturing illumination models directly from the physical world. One can measure the illumination of a small patch of a surface by photographically recording the radiance incident on that surface from each direction in the hemisphere surrounding its normal. If all visible objects are relatively distant, the illumination will not change significantly as one moves over a small area of the surface. Thus, I use photographic information to render surfaces under physically accurate illumination.

Similar approaches have been utilized recently by Debevec [1] and Sato [8] to render synthetic objects into real scenes. Debevec rendered the local scene explicitly, and captured illumination from the distant scene by taking photographs of a reflective metal ball. Sato and colleagues used a pair of fisheye lenses and a stereo algorithm to measure both the illumination from each direction and the distance to direct and indirect visible light sources. Such techniques might be used to create special effects, to generate augmented reality environments, or to simulate the appearance of a piece of furniture or artwork before physically adding it to a real scene.

2 Goals

The basic goal of this project is to use an illumination model represented by photographic images to render surfaces as they would appear in the real world. Such a rendering task relies on two essential models – one representing the light incident on the surface, and the other representing the bidirectional reflectance distribution function (BRDF) of the surface. To measure illumination, I use a camera with a fisheye lens, which measures incident radiance over more than a hemisphere. Each pixel of the resulting image maps to a direction in the physical world. I assume initially that the surface to be rendered is small relative to the distance to the nearest objects in the photographs. Under this assumption, illumination from any physical direction is the same at any point on the surface unless occluded by the rendered surface itself. I also assume that the surface is opaque and convex, and that reflectance does not vary over the surface. I followed my proposed goal of implementing the rendering process first in this relatively simple case, and then relaxing several of these assumptions.

I also proposed using a BRDF representation more realistic than the Phong model. In the actual implementation, I did not use the Phong model at all, adhering instead to models which are physically plausible in the sense that they obey energy conservation laws. I experimented with a moderately simple empirical reflectance model as well as a more complex physically-derived model, both from the computer graphics literature.

3 *Radiance* Rendering Algorithms

When all the initial assumptions described above hold, rendering each point on the surface requires performing an appropriately weighted integration over a hemisphere of the full illumination sphere. Although expensive, a naive implementation of this integration would not be difficult to code. I chose, however, to use the lighting visualization tool kit *Radiance*, developed by Greg Ward at Lawrence Berkeley National Laboratory over the past 15 years. The primary advantage of using this software is that once I have a basic implementation, I can draw on the existing capabilities of the software package to easily extend my results. Also, *Radiance* provides a ready interface and a fast implementation for certain rendering tasks.

Radiance differs from most rendering packages in that it aims to simulate the effects of physical illumination rather than to simply produce photorealistic images. That is, *Radiance* is designed as a predictive tool rather than as a means of producing artwork that looks like it might have been a photograph. *Radiance* accomplishes this with a hybrid stochastic and deterministic ray-tracing technique. It comprises a collection of programs which interact on the Unix model rather than through a graphical user interface. One might think of *Radiance* as a graphical programming environment, with input specified in text files. The remainder of this section describes the details of the rendering algorithms most relevant to this report, based on [7, 6, 5] and on personal e-mails from Greg Ward. Note that this description is far from complete; I read at least three hundred pages of documentation relevant to this project. Several of the algorithms involved were published by Greg Ward as SIGGRAPH papers.

Radiance breaks the complete radiance equation, an integration of a complicated function over a hemisphere, into three parts. One must understand each of these algorithms in order to properly set a long list of algorithmic parameters for rendering and to fully understand the results.

- **Direct component.** *Radiance* handles the direct contributions of physical light sources and virtual light sources due to mirrors separately from those of indirect light sources. Light sources in *Radiance* always have nonzero surface area. A jittered sample is chosen at random from this surface area and used to compute the radiance contribution of that light source.

  This simple routine works for light sources of finite size because *Radiance* precedes the jittered sampling by an adaptive light source subdivision process whose termination condition depends on the ratio of the source's maximum side length to the distance between the source and the point being rendered. Both the adaptive subdivision process and the jittering process are designed for rectangular light sources. In fact, light sources of certain shapes, such as spheres and sources (discussed below), are not subdivided at all. If a non-rectangular light is subdivided and direct jittering is enabled, certain samples may miss the light completely, causing rendering errors.

- **Indirect specular component.** *Radiance* assumes a default reflectance model, discussed later in this section, whose specular lobe is specified by a Gaussian model. At each rendered point, *Radiance* uses a Monte Carlo technique to choose a single sample ray direction from a probability distribution proportional to this lobe. The contribution of that ray is computed recursively by standard ray tracing techniques. While one can specify an arbitrary BRDF for a material, the indirect specular contribution is computed only for variants of the default model, because the algorithm depends on an analytical Monte Carlo inversion formula specific to the Gaussian form of the specular lobe.

- **Indirect diffuse component.** Computation of the indirect diffuse component poses a challenge to ray-tracing algorithms because it requires one to cast rays over an entire hemisphere for each surface point. *Radiance* deals with the formidable putative expense of this computation by using a sophisticated caching mechanism to avoid casting these rays at every point. The details of this algorithm, which involve a method for extrapolating diffuse illumination from that of nearby surface points as well as an octree separate from that used for ray intersection, are beyond the scope of this report, even though this is the most interesting part of the *Radiance* package.

While *Radiance* allows specification of arbitrary surface bidirectional reflectance-transmittance distribution functions (BTRDFs), the fast indirect lighting algorithms described above depend on a particular form of the reflectance. The indirect diffuse lighting

Figure 1: Two sample fisheye photographs representing illumination at a given point. One is taken indoors, the other outdoors.

algorithm assumes simple Lambertian reflectance. The indirect specular algorithm assumes anisotropic reflectance which, in the isotropic case, simplifies to [7, 5]

$$f(\theta_i, \phi_i; \theta_r, \phi_r) = \rho_s \frac{1}{\sqrt{cos\theta_i cos\theta_r}} \frac{\exp[-\tan^2 \delta/\alpha^2]}{4\pi\alpha^2} \tag{1}$$

where θ_i and phi_i represent the incident angle, θ_r and ϕ_r represent the reflected angle, ρ_s is the specular reflectance, δ is the angle between the surface normal and the bisector of the incident and mirrored reflection directions, and α is the roughness measured as the RMS standard deviation of the surface slope. Although empirical, this model satisfies physical conservation laws and fits measured BRDF data closely [**?**].

4 Algorithms and Results (Achievements)

4.1 Image acquisition and geometry

I deduced distant global illumination at several indoor and outdoor locations from images acquired by Ted Adelson using a Nikon COOLPIX 950 digital camera with its standard fisheye lens adapter. Two sample images are shown in Figure 1. Adelson took these photographs with the center of the lens pointed straight upwards. The distance of a given pixel from the center of an ideal fisheye lens image is proportional to the angle between the direction corresponding to that pixel and that corresponding to the center of the lens. While the administrators of Nikon's technical support forum refused to disclose data on additional geometric distortions introduced by their lenses, they did inform me that the field of view of the fisheye lens is 183 degrees. I assumed an ideal fisheye lens with a 183 degree field of view and wrote a *Radiance* function to map illumination directions to the image.

While the CCD array of a digital camera has an inherently linear luminance response, the pixel values output by the Nikon camera are not proportional to luminance. First, they suffer from saturation. Three eight-bit values represent the color at each image pixel. I used Matlab to compute histograms of each component of the color values for each image, and found significant peaks at the highest intensity values in all the images due to saturation. Second, the camera passes luminance values through a compressive nonlinearity, so that pixel values are not proportional to luminance even within the dynamic range. Nikon refused to share this nonlinearity (the gamma function of their camera). Debevec and Malik [2] demonstrated a method for obtaining luminance values over a high dynamic range using multiple images at different exposures, but I have not yet collected the necessary data or implemented this optimization. I also ignored for the time being the luminance nonlinearities inherent in display devices. Radiance provides utilities to deal with these, but they require data on the specific display device to be used.

Although a hemispherical image suffices to describe the illumination of a small surface patch, capturing the illumination of an object such as a sphere requires a full 360 degree field of view. This might be captured by taking two fisheye lens photographs in opposite directions and fusing them together. I have found software specifically for this purpose [3], but I have not had a chance to collect the necessary pairs of photographs because the Nikon camera has been out of town for some time recently. I therefore rendered objects using only the upper natural illumination hemisphere. I created the renderings from a viewpoint 45 degrees from a direct top view, which somewhat ameliorates this shortcoming, but portions of the resulting images are nevertheless inaccurate.

4.2 Indirect illumination approach

The recommended method for dealing with distant wide-angle light sources in *Radiance* is to force their contributions exclusively into the indirect lighting computation by representing them as "glow" materials, a special luminous material type ignored in the direct lighting computation. This leads to a highly efficient algorithm which takes advantage of the speed of the specular and diffuse indirect lighting computations described in Section 3. In addition to polygons, spheres, and cones, *Radiance* supports a special surface type, "source," representing infinitely distant solid-angle light sources. My first approach was to map the fisheye photographs onto sources slightly large than hemispheres with "glow" characteristics and then to adjust the parameters of the indirect lighting computation to obtain acceptable accuracy in the renderings.

Figure 2 shows a sphere of oxidized copper, generated using the standard metal reflectance type in *Radiance*. I obtained the color, specularity, and roughness parameters for this surface from a data file of standard material properties included with the *Radiance* distribution. I originally rendered an image at a resolution of 1024 by 1024, then down-sampled it to 512 by 512 using the pfilt program. This is standard *Radiance* technique; pfilt reduces aliasing within the image and at its boundaries by filtering and down-sampling, and also automatically adjusts image exposure. Note that the specularities in this image

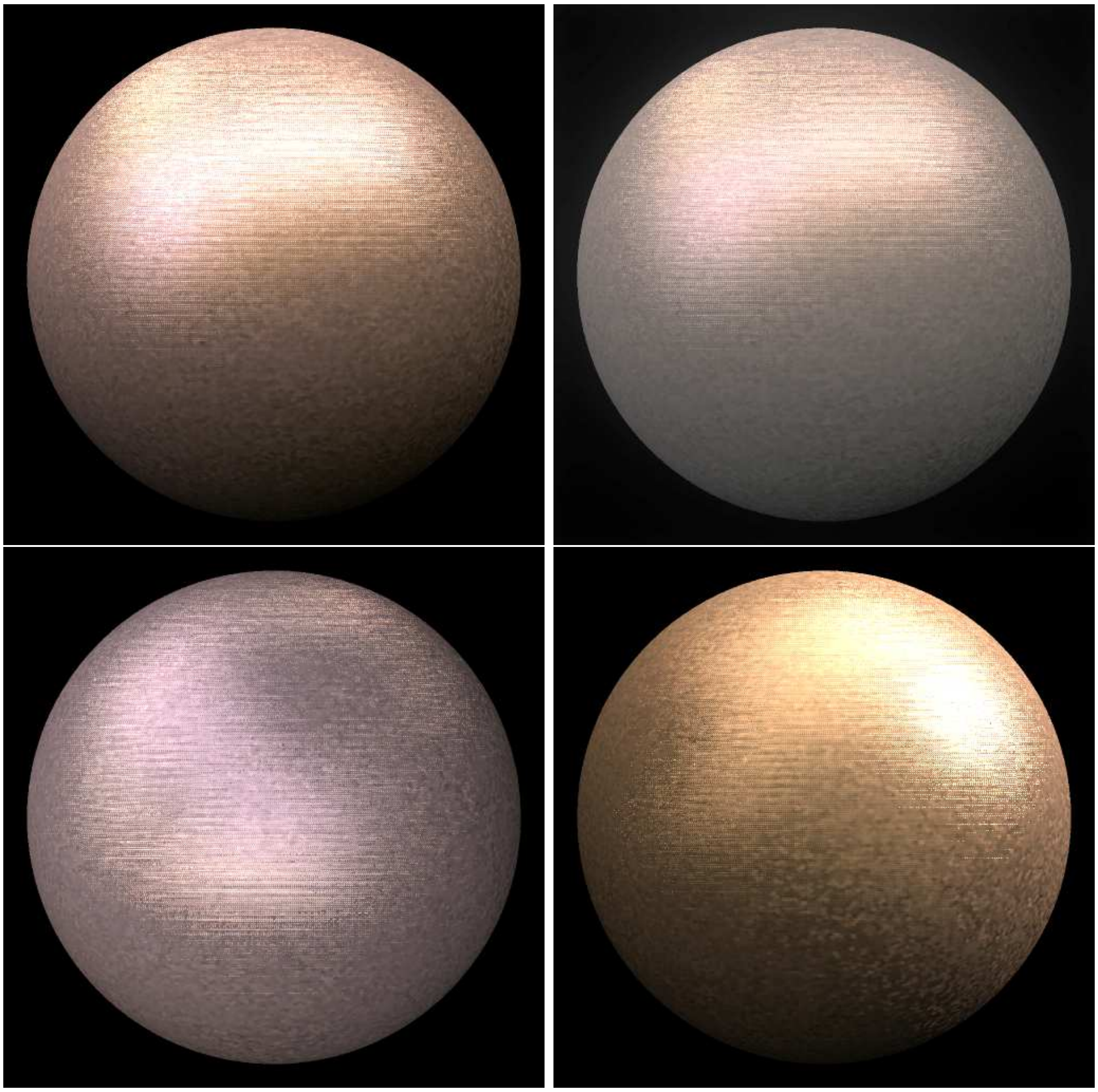

Figure 2: The upper left image is a *Radiance* rendering of an oxidized copper sphere under the illumination represented by the left-hand image in Figure 1, generated at 1024 by 1024 resolution and then filtered and down-sampled to 512 by 512. The upper right image is the same rendering after post-processing with the *Radiance* pcond function, which mimics the human visual response by defocusing dark regions, adding glare to very bright regions, using human contrast sensitivity to determine exposure, and shifting to a blue-dominant response function in scotopic regions. All other images in this report show images before processing by pcond. The two lower images are similar to the upper left image except that their illuminations are derived from different indoor and outdoor photographs. For the purposes of comparison, all other images in this report use the illumination map pictured on the left of Figure 1.

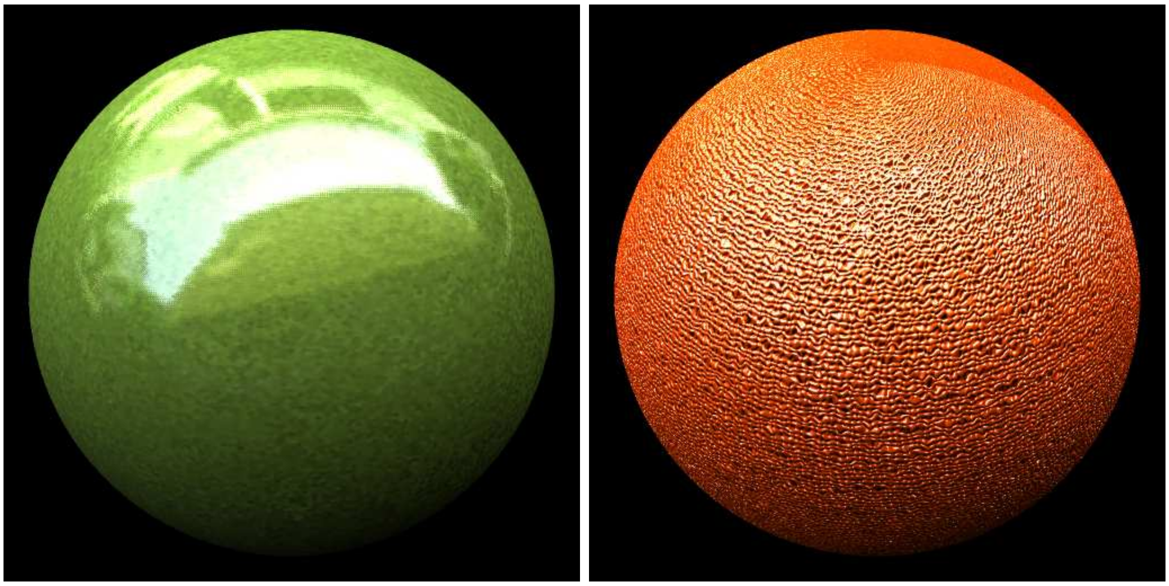

Figure 3: At left, a sphere coated by emerald green paint. At right, an orange whose surface is rendered using a bump map. Both images were generated using photographically acquired illumination at 1024 by 1024 and down-sampled to 512 by 512.

are somewhat speckled. This artifact results from the indirect specular evaluation routine, which sends out a single, jittered ray for each image pixel. If jittering is turned down, the surface will appear less speckled, but the simulation will be less accurate. One can improve results by generating a larger image and using pfilt to low-pass filter and down-sample it. Of course, this increases computational cost. Generating the 1024 by 1024 image takes between 5 and 10 minutes on a fast Sparc Ultra. Also, note that in each image certain areas do not exhibit any specular reflectance. The specular lobes for these regions fall in the lower illumination hemisphere, outside the range of the fisheye lens image.

Figure 3 shows two more spheres with different reflectance properties. One is coated with slightly specular, smooth, emerald green paint. Although the specularity is weak, the reflection of the incident light sources is well-defined due to the smoothness of the surface. The image of an orange illustrates *Radiance*'s capabilities to represent texture as a bump map. One defines a texture as a *Radiance* function, then maps it onto a surface. In this case, I found a texture function in a texture library and estimated color, specularity, and roughness myself.

Using *Radiance*, I can easily extend my rendering capabilities to arbitrary 3D geometries. Figure 4 shows a few examples. The tile at left has a purely diffuse reflectance, with an added pattern or color map (or, to use the standard computer graphics misnomer, "texture"). This particular wooden texture has a three-dimensional description, so that the grain runs in consistent directions on the different surfaces. The coffee table rendering, on the other hand, must take accurate account of inter-reflections. Note that different parts of the table cast shadows on each other, although the external scene does not cast shadows on the table because the photographed objects are modelled as infinitely distant.

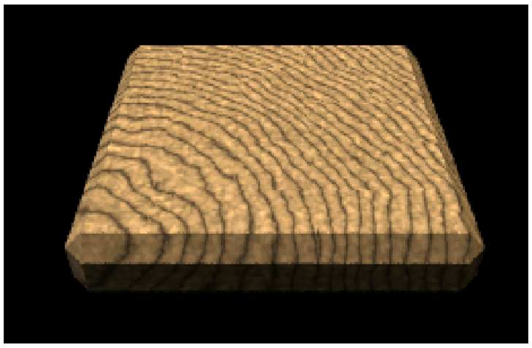 

Figure 4: On the left, a patterned wooden tile. On the right, a table based on a model in [7]. Both images were rendered at 512 by 512 and down-sampled to 256 by 256.

The basic *Radiance* reflection model also includes anisotropic reflectors, as well as transparent and translucent materials. While I have not included example images in this report, all of these could be easily rendered in *Radiance* under photographically acquired lighting using the indirect illumination approach.

4.3 Direct illumination approach

Using only indirect lighting leads to several limitations. As mentioned in Section 3, the indirect specular calculation is automatically disabled when a specular lobe other than *Radiance*'s general anisotropic default is specified, and the indirect diffuse calculation assumes that the diffuse reflectance has a Lambertian form. *Radiance* allows the user to disable the diffuse reflectance computation and specify an arbitrary BRDF as an analytic function or as a data file, but in this case only the direct lighting contribution will be computed. A second limitation of the indirect lighting approach was mentioned in Section 4.2. Since the specular computation uses only a single sample at each point, rough specular images are prone to speckle of the type observed in Figure 2.

I solved both of these problems by representing the photographically acquired illumination as a direct light source. The simplest approach to accomplishing this would be to simple convert the “source” surface from a “glow” material to a “light” material so that it will be included in the direct computation. This fails because *Radiance* does not apply adaptive source subdivision to “source” surfaces (Section 3); the whole hemisphere would effectively be concentrated into a tiny disk at its center. Perhaps the next most obvious strategy is to replace the source by a large sphere tessellated by polygons, each of which *Radiance* will subdivide. I implemented this scheme initially, but it produces poor results

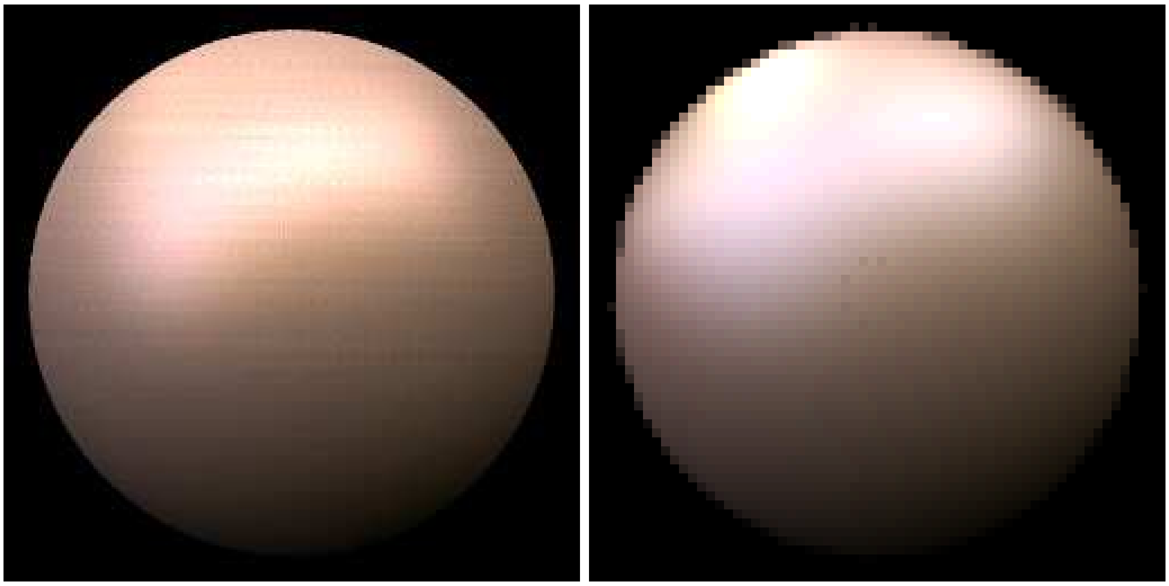

Figure 5: At left, the copper sphere of the previous section rendered using the direct lighting approach. This image was generated at a 512 by 512 resolution and down-sampled to 256 by 256. At right, a similar sphere with reflectance given by the He-Torrance model. Due to the expensive rendering process, this image was generated at 128 by 128. It was not down-sampled by pfilt, leading to the jagged appearance of the edges.

because many of the polygons are non-rectangular, so that many of the jittered samples miss their intended surface entirely (Section 3). A much better strategy is to simply map the photographic illumination map onto the faces of a large cube surrounding the scene to be rendered. *Radiance* then handles the subdivisions automatically, and all the regions to be subdivided and jittered are rectangular. Note that regions near the corners of the cube will be subdivided at a higher resolution than necessary because the termination condition of the subdivision algorithm ignores the relative orientation of the light source polygon with respect to the rendered surface (Section 3).

This method works accurately, but at great computational expense. The copper sphere of Figure 5 has the same material properties and the same illumination as that in the upper left of Figure 2. The speckle artifacts have been eliminated. However, this image took over ten times as long to compute at half the resolution. The direct lighting computation does not utilize the clever caching algorithm of the indirect diffuse lighting calculation (Section 3); it does something more akin to a simple integration over the hemisphere.

I also used the direct lighting method to render a sphere with a reflectance given by the He-Torrance model [4], one of the most accurate available physically-derived reflectance models. The He-Torrance model includes a diffuse component, a specular component, and a directional diffuse component. The standard *Radiance* distribution includes a complete implementation of the BRDF predicted by this model, as well as a slightly simplified version which ignores some of the variation of reflectance between color bands at great computational savings. Even with the simplified model, the combination of a computationally complex reflectance model and complex direct lighting leads to renderings at least three orders of magnitude slower than in the indirect lighting case. Figure 5 also shows an example of a rendering using this spectrally simplified He-Torrance model. I chose model parameters to generate a surface similar to that of Figure 2. Specifically, this sphere has the same diffuse color as that of Figure 2. I used a surface height standard deviation $\sigma_0 = .4\mu m$ and an autocorrelation length $\tau = 3\mu m$, based on parameters used in the examples of He's paper [4] [1]. I produced other renderings for this surface using simple light sources instead of photographically acquired lighting. Renderings generated with the He-Torrance model do appear more realistic to me than those generated with *Radiance*'s default model, but the difference is relatively small given the amount of additional computational expense involved.

5 Lessons Learned

I learned a great deal from this project. I spent over half my time learning to use *Radiance* rather than working on my specific application. During that time, however, I simultaneously learned to use a powerful rendering and visualization package and learned in much more detail the algorithms used to compute surface luminances. Once I passed the learning phase, *Radiance* enabled me to accomplish some of the goals I did not expect would be possible during the course of this project.

This project exposed me directly to a large portion of the concepts we covered in class, including ray tracing, radiosity, illumination and material models, visibility, octrees, texture mapping, bump mapping, and geometrical modeling. It helped me understand important tradeoffs in rendering and the importance of fast computation in graphics. I also gained a detailed physical understanding of certain reflectance models, and an appreciation for the importance of dynamic range in illumination.

6 Acknowledgements

Greg Ward, now at UC Berkeley, proved an incredible resource. On each of the two occasions I e-mailed him, he responded within an hour with accurate answers to all my questions. I am also grateful to Takehiko Nagakura and Haldane Liew of the MIT Architecture Department for instructing me to e-mail Ward, and to Max Chen of the LCS Graphics Group for instructing me to contact Nagakura's group.

Kari Anne Hoier Kjolaas also made a number of helpful suggestions.

7 Disacknowledgements

Nikon Corporation refused to share data on their COOLPIX 950 camera. Specifically, they would not share information on the nonlinear mapping (gamma) from intensities to pixel

[1] I also used an index of refraction of 1.5. This was a mistake which I don't have time to fix due to the long rendering times involved. Metals have very large, mostly imaginary refractive indices, because an electromagnetic wave dies very quickly in a metal.

values introduced in the imaging process, or on geometrical distortions due to the fisheye lens. According to other users of Nikon's technical support forum, Nikon's competitors share similar data because it can always be measured through calibration experiments by camera owners. I also faxed Nikon's engineering staff, who simply ignored me.

References

[1] P. E. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. *Computer Graphics (SIGGRAPH)*, 1998.

[2] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. *Computer Graphics (SIGGRAPH)*, pages 369–78, 1997.

[3] H. Dersch. Panorama tools. http://www.fh-furtwangen.de/ dersch/, 1999.

[4] X. D. He, K. E. Torrance, F. S. Sillion, and D. P. Greenberg. A comprehensive physical model for light reflection. *Computer Graphics (SIGGRAPH)*, 25(4):175–86, 1991.

[5] G. W. Larson. Behavior of materials in radiance. http://radsite.lbl.gov/radiance/refer/materials.pdf, June 1996.

[6] G. W. Larson. The radiance 3.1 synthetic imaging system. http://radsite.lbl.gov/radiance/refer/ray.html, 1997.

[7] G. W. Larson and R. Shakespeare. *Rendering with Radiance: The Art and Science of Lighting Visualization.* Morgan Kaugmann, San Francisco, 1998.

[8] I. Satio, Y. Sato, and K. Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):1–12, 1999.