Chapter 4

Irradiance Environment Maps

In the introduction to this dissertation, we noted that complex realistic lighting environments are rarely used in either forward or inverse rendering. We also stated our thesis that a deeper understanding of the computational nature of reflection and illumination helps to address these difficulties and restrictions in a number of areas in computer graphics and vision. Subsequently, in chapters 2 and 3, we have developed a new way of looking at reflection, formalizing the idea of reflection as a spherical convolution of the incident illumination and BRDF. The insights from these two chapters lead to the possibility of attacking a number of difficult forward and inverse rendering problems in the frequency domain.

Chapters 4, 5 and 6 of this dissertation are devoted to practical applications of the signal-processing ideas developed theoretically in the previous two chapters. Chapters 4 and 5 deal with efficient representations and algorithms for forward rendering using environment maps, which are representations of the (distant) incident illumination distribution at a point. This chapter considers the case of *Irradiance Environment Maps*, corresponding to the reflection from diffuse or Lambertian objects. We show that frequency-space analysis can be used to reduce the effects of arbitrarily complex (but distant) incident illumination to a simple analytic low-dimensional formula. In the next chapter, we will extend these ideas, using similar methods for general BRDFs, further showcasing the practical benefits of frequency-space concepts like sampling rate analysis and efficient frequency domain convolutions. Finally, chapter 6 presents practical algorithms for inverse rendering—estimation of illumination and material properties.

In this chapter, we consider the rendering of diffuse objects under distant illumination, as specified by an environment map. Using an analytic expression for the irradiance in terms of spherical harmonic coefficients of the lighting, derived in chapter 3.2.5, we show that one needs to compute and use only 9 coefficients, corresponding to the lowest-frequency modes of the illumination, in order to achieve average errors of only 1%. In other words, the irradiance is insensitive to high frequencies in the lighting, and is well approximated using only 9 parameters. In fact, we show that the irradiance can be procedurally represented simply as a quadratic polynomial in the cartesian components of the surface normal, and give explicit formulae. These observations lead to a simple and efficient procedural rendering algorithm amenable to hardware implementation, a prefiltering method up to three orders of magnitude faster than previous techniques, and new representations for lighting design and image-based rendering.

The rest of this chapter is organized as follows. After an introduction to the specific problem of interest here, in section 1, we briefly describe the relevant background and practical details from the previous theoretical analysis required here in section 2. Section 3 discusses practical implementation of our algorithms. Finally, section 4 concludes this paper and suggests directions for future work. This chapter corresponds to our SIGGRAPH paper on *An Efficient Representation for Irradiance Environment Maps* [71].

4.1 Introduction and Previous Work

Lighting in most real scenes is complex, coming from a variety of sources including area lights and large continuous lighting distributions like skylight. But current graphics hardware only supports point or directional light sources. One reason is the lack of simple procedural formulas for general lighting distributions. Instead, an integration over the upper hemisphere must be done for each pixel.

One approach to using general lighting distributions is the method of environment maps. Environment maps are representations of the incident illumination at a point. Blinn and Newell [5] used them to efficiently find reflections of distant objects. Miller and Hoffman [59], and Greene [22] *prefiltered* environment maps, precomputing separate reflection

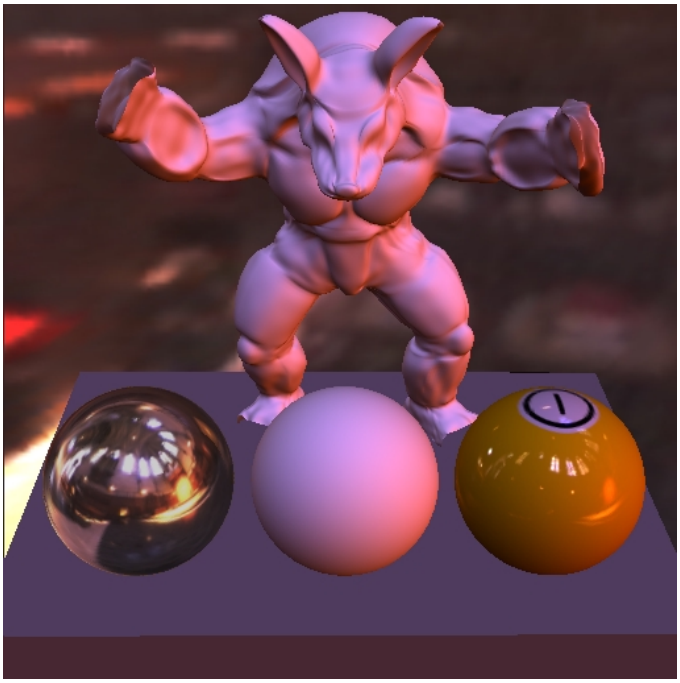

Figure 4.1: *The diffuse shading on all the objects is computed procedurally in real-time using our method. The middle sphere, armadillo, and table are white diffuse reflectors. The colors come from the environment—owing to a variety of colored sources, including blue stained-glass windows. Our method can also be combined with standard texture mapping—used to modulate the albedo of the pool-ball on the right—and reflection mapping—used for specular highlights on the pool-ball, and for the mirror sphere on the left. The environment is a light probe of the Grace Cathedral. Tone mapping is used to convey high dynamic range for the background and the mirror sphere; the remaining objects are shaded using a linear scale.*

maps for the diffuse and specular components of the BRDF. Cabral et al. [8] handled general BRDFs by using a 2D set of prerendered images. Prefiltering is generally an offline, computationally expensive process. After prefiltering, rendering can usually be performed at interactive rates with graphics hardware using texture-mapping.

Of course, environment maps, and the relevant techniques presented in this dissertation, are only an approximation and do not account for near-field illumination, cast shadows, or interreflection. Nevertheless, they have proven an effective tool for interactive rendering with realistic lighting effects.

This chapter focuses on the Lambertian component of the BRDF. We use the term *irradiance environment map* for a diffuse reflection map indexed by the surface normal,

since each pixel simply stores the irradiance for a particular orientation of the surface. For applications like games, irradiance maps are often stored directly on the surface, instead of as a function of the normal vector, and are called *light maps*. Irradiance environment maps can also be extended to spatially varying illumination by computing an *irradiance volume*, as done by Greger et al. [23]. Many of the same ideas can be applied to speeding up global illumination algorithms. The slowly varying nature of irradiance has led to Ward and Heckbert [85] proposing interpolation using irradiance gradients, while the idea of storing irradiance as a function of surface orientation in *orientation lightmaps* has been proposed by Wilkie et al. [87].

Our approach relies on the rapid computation of an analytic approximation to the irradiance environment map. For rendering, we demonstrate a simple procedural algorithm that runs at interactive frame rates, and is amenable to hardware implementation. The procedural approach is preferable to texture-mapping in some applications. Since irradiance varies slowly with orientation, it need only be computed per-vertex and interpolated across triangles. Further, we require only a single texturing pass to render textured objects with irradiance environment maps, since the irradiance is computed procedurally. On the other hand, the standard approach requires a separate texture for the irradiance, and needs *multitexturing* support or multiple texturing passes. In other applications, where per-fragment texture-mapping is relatively inexpensive, our method can be used to very efficiently compute the irradiance environment map texture. Our novel representation also suggests new approaches to lighting design and image-based rendering.

4.2 Background

Empirically, it is well known that the reflected intensity from a diffuse surface varies slowly as a function of surface orientation. This qualitative observation has been used to justify representing irradiance environment maps at low resolutions [59], and in efficiently computing the shading hierarchically [39, 45]. Our goal is to use an analytic quantitative formula for the irradiance, derived in section 3.2.5, which formalizes these observations, and allows for principled approximations.

Let L denote the distant lighting distribution. As is common with environment map algorithms, we neglect the effects of cast shadows and near-field illumination. The irradiance E is then a function of the surface normal only and is given by an integral over the upper or visible hemisphere,

$$E(\alpha, \beta) = \int_{\theta_i'=0}^{\pi/2} \int_{\phi_i'=0}^{2\pi} L\left(R_{\alpha,\beta}(\theta_i', \phi_i')\right) \cos\theta_i' \, d\theta_i' d\phi_i'. \tag{4.1}$$

We must scale E by the surface albedo[1], which may be dependent on position $\vec{X}$ and be described by a texture $T(\vec{X})$, to find the reflected light field B, which corresponds directly to the image intensity,

$$B(\vec{X}; \alpha, \beta) = T(\vec{X}) E(\alpha, \beta). \tag{4.2}$$

Our main concern will be approximating the irradiance E. A texture map $T(\vec{X})$ may be used later to simply modulate the reflected light field computed. Note that the form of equation 4.1 is simply a special case of the reflection equation 2.11 for isotropic surfaces with no outgoing angular dependence. The limits of the θ_i' integral range from 0 to $\pi/2$ because we consider only the front hemisphere, where the cosine of the incident angle is positive. The transfer function corresponding to the Lambertian BRDF is the *clamped cosine* function $\hat{\rho}(\theta_i') = \max(\cos\theta_i', 0)$.

In section 3.2.5 (a more detailed version of which is published in [72]), we have been able to derive an analytic formula for the irradiance by determining the spherical harmonic filter coefficients for the Lambertian clamped-cosine function. Similar results have been obtained independently by Basri and Jacobs [2] in simultaneous work on face recognition.

For the purposes of implementation, it is often convenient to use real-valued functions where possible, rather than the complex forms of the spherical harmonics given in equation 2.27. It is easy to define real forms of the spherical harmonics, simply by considering the real and complex parts separately. For this purpose, we define the real form of the

[1]Technically, for Lambertian objects, the BRDF is given by $1/\pi$ times the albedo, so the textures should be multiplied by $1/\pi$.

spherical harmonics as follows (c.f. equation 2.26),

$$\begin{aligned} N_{lm} &= \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} \\ Y_{lm}(\theta,\phi) &= N_{lm}P_{lm}(\cos\theta)az_m(\phi), \end{aligned} \tag{4.3}$$

where the azimuthal basis functions are defined by

$$\begin{aligned} az_{+m}(\phi) &= \sqrt{2}\cos\phi \\ az_0(\phi) &= 1 \\ az_{-m}(\phi) &= \sqrt{2}\sin\phi. \end{aligned} \tag{4.4}$$

While this is essentially the standard definition of the real form of the spherical harmonics, the sign conventions used are not always consistent. For that reason, we will make explicit the numerical values used here to fix the precise conventions used by us.

Recall that the spherical harmonics may be written as polynomials of the cartesian components (x, y, z). Below, we give the numeric values of the the first 9 spherical harmonics (with $l \leq 2$), which are simply constant ($l = 0$), linear ($l = 1$), and quadratic ($l = 2$) polynomials (c.f. figure 2.3),

$$\begin{aligned} (x,y,z) &= (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) \\ Y_{00}(\theta,\phi) &= 0.282095 \\ (Y_{11};Y_{10};Y_{1-1})\,(\theta,\phi) &= 0.488603\,(x;z;y) \\ (Y_{21};Y_{2-1};Y_{2-2})\,(\theta,\phi) &= 1.092548\,(xz;yz;xy) \\ Y_{20}(\theta,\phi) &= 0.315392\left(3z^2-1\right) \\ Y_{22}(\theta,\phi) &= 0.546274\left(x^2-y^2\right). \end{aligned} \tag{4.5}$$

Note that these basis functions are closely related to the spherical polynomials used by Arvo [1] in his irradiance tensor formulation.

$E(\alpha,\beta)$ and $L(\theta,\phi)$ can be represented by the coefficients—E_{lm} and L_{lm}—in their

spherical harmonic expansion,

$$\begin{aligned} L(\theta,\phi) &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l} L_{lm}Y_{lm}(\theta,\phi) \\ E(\alpha,\beta) &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l} E_{lm}Y_{lm}(\alpha,\beta). \end{aligned} \tag{4.6}$$

We may also expand the Lambertian transfer function $\hat{\rho}(\theta_i') = \max(\cos\theta_i', 0)$, i.e. the clamped cosine, in terms of spherical harmonics. Since $\hat{\rho}$ has no azimuthal dependence, $m = 0$ and we use only the l index,

$$\hat{\rho}(\theta) = \max\left[\cos\theta, 0\right] = \sum_{l} \hat{\rho}_l Y_{l0}(\theta). \tag{4.7}$$

With these definitions, one can directly apply equation 2.62 or equation 3.10,

$$E_{lm} = \Lambda_l \hat{\rho}_l L_{lm}, \tag{4.8}$$

where $\Lambda_l = \sqrt{4\pi/(2l+1)}$. The only difference in equation 3.28 in section 3.2.5 is that we used there the reflected light field B, which is simply a scaled version of the irradiance E for Lambertian surfaces.

It will be convenient to define a new variable A_l by

$$A_l = \Lambda_l \hat{\rho}_l, \tag{4.9}$$

and to expand out the irradiance for rendering,

$$E(\alpha,\beta) = \sum_{l,m} A_l L_{lm} Y_{lm}(\alpha,\beta). \tag{4.10}$$

An analytic formula for $\hat{\rho}_l$ (and hence A_l) has been derived in section 3.2.5. It can be shown that A_l vanishes for odd values of $l > 1$, and even terms fall off very rapidly as $l^{-5/2}$. The

analytic formulae are given by (c.f. equation 3.32)

$$\begin{array}{ll} l = 1 & A_l = \frac{2\pi}{3} \\ l > 1, \mathbf{odd} & A_l = 0 \\ l \ \mathbf{even} & A_l = (-1)^{\frac{l}{2}-1} \frac{2\pi}{(l+2)(l-1)} \left[\frac{l!}{2^l \left(\frac{l}{2}!\right)^2} \right]. \end{array} \tag{4.11}$$

Numerically, the first few terms are

$$\begin{aligned} A_0 = 3.141593 \quad A_1 = 2.094395 \quad A_2 = 0.785398 \\ A_3 = 0 \quad A_4 = -0.130900 \quad A_5 = 0 \quad A_6 = 0.049087. \end{aligned} \tag{4.12}$$

Approximation: For rendering, we make use of the observation that A_l decays so fast that we need consider only low-frequency lighting coefficients, of order $l \leq 2$. Equivalently, **the irradiance is well approximated by only 9 parameters**—1 for $l = 0, m = 0$, 3 for $l = 1, -1 \leq m \leq 1$, and 5 for $l = 2, -2 \leq m \leq 2$. By working in frequency-space, we exploit the low-frequency character of the Lambertian BRDF filter, using a few coefficients instead of a full hemispherical integral. The simple form of the first 9 spherical harmonics, given in equation 4.5, makes implementation straightforward.

4.3 Algorithms and Results

In this section, we discuss three applications of this result. First, we show how to rapidly prefilter the lighting distribution, computing the coefficients L_{lm}. Next, we develop a simple real-time procedural shader for rendering that takes these coefficients as inputs. Finally, we discuss other applications of our representation.

4.3.1 Prefiltering

For a given environment map, we first find the 9 lighting coefficients, L_{lm} for $l \leq 2$, by integrating against the spherical harmonic basis functions. Each color channel is treated

separately, so the coefficients can be thought of as RGB values,

$$L_{lm} = \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} L(\theta, \phi) Y_{lm}(\theta, \phi) \sin\theta \, d\theta d\phi. \tag{4.13}$$

The expressions for the Y_{lm} are found in equation 4.5. The integrals are simply sums of the pixels in the environment map L, weighted by the functions Y_{lm}. The integrals can also be viewed as moments of the lighting, or as inner-products of the functions L and Y_{lm}.

Since we compute 9 numbers, the prefiltering step takes O(9S) time, where S is the size (total number of pixels) of the environment map. By comparison, the standard method of computing an irradiance environment map texture takes $O(\mid T \mid \cdot S)$ time, where $\mid T \mid$ is the number of texels in the irradiance environment map. Our method will therefore be approximately $\mid T \mid /9$ times faster[2]. Even if a conventional irradiance environment map is computed at a very low resolution of 64×64, corresponding to $\mid T \mid = 4096$, our method will be nearly 500 times faster.

We have implemented prefiltering as a preprocessing step for a given environment map. Values of L_{lm} for a few light probes are tabulated in figure 4.1. The computation time for a 300x300 environment map was less than a second. This indicates that our approach might be able to handle scenes with dynamic lighting in the future. By contrast, the standard method of performing a hemispherical integral for each pixel to compute the irradiance environment map took approximately two hours. In fact, if an explicit representation of the irradiance environment map texture is required, we believe the best way of computing it is to first compute the 9 coefficients L_{lm} using our method, and then use these to very rapidly generate the irradiance environment map using the rendering method described below.

It is important to know what errors result from our 9 parameter approximation. The maximum error for any pixel, as a fraction of the total intensity of the illumination, is 9% and corresponds to the maximum error in the order 2 approximation of the clamped cosine function. Furthermore, the average error over all surface orientations can be shown to be under 3% for any physical input lighting distribution [2]. For the environment maps used in our examples, corresponding to complex natural illumination, the results are somewhat

[2]It may be possible to use a hierarchical integration scheme, as demonstrated by Kautz et al. [39] for Phong BRDFs, to speed up both our method and the conventional approach. Hardware acceleration may also be possible.

	Grace Cathedral			Eucalyptus Grove			St. Peters Basilica		
L_{00}	.79	.44	.54	.38	.43	.45	.36	.26	.23
L_{1-1}	.39	.35	.60	.29	.36	.41	.18	.14	.13
L_{10}	-.34	-.18	-.27	.04	.03	.01	-.02	-.01	-.00
L_{11}	-.29	-.06	.01	-.10	-.10	-.09	.03	.02	.01
L_{2-2}	-.11	-.05	-.12	-.06	-.06	-.04	.02	.01	.00
L_{2-1}	-.26	-.22	-.47	.01	-.01	-.05	-.05	-.03	-.01
L_{20}	-.16	-.09	-.15	-.09	-.13	-.15	-.09	-.08	-.07
L_{21}	.56	.21	.14	-.06	-.05	-.04	.01	.00	.00
L_{22}	.21	-.05	-.30	.02	-.00	-.05	-.08	-.06	.00

Table 4.1: *Scaled RGB values of lighting coefficients for a few environments. These may be used directly for rendering, and for checking the correctness of implementations.*

better than the worst-case bounds—the average error is under 1%, and the maximum pixel error is under 5%. Finally, figure 4.2 provides a visual comparison of the quality of our results with standard prefiltering, showing that our method produces a perceptually accurate answer.

4.3.2 Rendering

For rendering, we can find the irradiance using equation 4.10. Since we are only considering $l \leq 2$, the irradiance is simply a quadratic polynomial of the coordinates of the (normalized) surface normal. Hence, with $\vec{N}^t = (x\ y\ z\ 1)$, we can write

$$E(\vec{N}) = \vec{N}^t M \vec{N}. \tag{4.14}$$

M is a symmetric 4x4 matrix. Each color has an independent matrix M. Equation 4.14 is particularly useful for rendering, since we require only a matrix-vector multiplication and a dot-product to compute E. The matrix M is obtained by expanding equation 4.10,

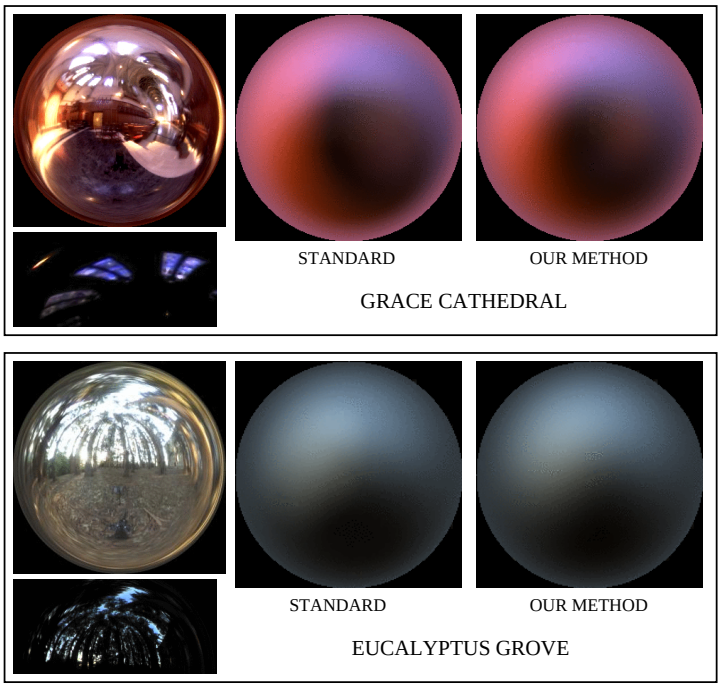

Figure 4.2: *A comparison of irradiance maps from our method to standard prefiltering. The irradiance map resolutions are 256x256. For each light probe, the left image is a tone-mapped version of the environment. Below that, we show the brightest parts of the environment on a linear scale. Both environments have bright bluish lights—from stained-glass windows, and the sky respectively—which are not apparent in the tone-mapped images. This accounts for the bluish portions of the irradiance maps. It can be seen that our method produces a result very close to the correct answer. Note that our rendering algorithm does not actually use irradiance maps; we computed them here solely for the purposes of the quality comparison. The coordinate mapping in the images is such that the center of the image is straight forward ($\theta = 0$, the north pole or +Z), the circumference of the image is straight backwards ($\theta = \pi$, the south pole or -Z), and θ varies uniformly in the radial direction from 0 to π. The azimuthal angle ϕ corresponds to the image polar angle.*

$$M = \begin{pmatrix} c_1 L_{22} & c_1 L_{2-2} & c_1 L_{21} & c_2 L_{11} \\ c_1 L_{2-2} & -c_1 L_{22} & c_1 L_{2-1} & c_2 L_{1-1} \\ c_1 L_{21} & c_1 L_{2-1} & c_3 L_{20} & c_2 L_{10} \\ c_2 L_{11} & c_2 L_{1-1} & c_2 L_{10} & c_4 L_{00} - c_5 L_{20} \end{pmatrix}$$

$$c_1 = 0.429043 \quad c_2 = 0.511664$$

$$c_3 = 0.743125 \quad c_4 = 0.886227 \quad c_5 = 0.247708. \tag{4.15}$$

The entries of M depend[3] on the 9 lighting coefficients L_{lm} and the expressions for the spherical harmonics. The constants come from the numerical values of A_l given in equation 4.12, and the spherical harmonic normalizations given in equation 4.5.

On systems not optimized for matrix and vector operations, it may be more efficient to explicitly write out equation 4.10 for the irradiance as a sum of terms,i.e. expand equation 4.15,

$$\begin{aligned} E(\vec{N}) &= c_1 L_{22}\left(x^2 - y^2\right) + c_3 L_{20} z^2 + c_4 L_{00} - c_5 L_{20} \\ &+ 2c_1\left(L_{2-2}xy + L_{21}xz + L_{2-1}yz\right) \\ &+ 2c_2\left(L_{11}x + L_{1-1}y + L_{10}z\right). \end{aligned} \tag{4.16}$$

We implemented equations 4.14 and 4.16 as procedural shaders in the Stanford real-time programmable shading system [68]. We used the ability of that system to perform computations per-vertex. Since E varies slowly, this is adequate and the shading is insensitive to how finely the surfaces are tessellated. The irradiance computations may be performed in software or compiled to vertex programming hardware, if available. The simple forms of equations 4.14 and 4.16 indicate that a per-fragment method could also be implemented in programmable hardware.

We were able to achieve real-time frame rates on PCs and SGIs. As shown in the SIGGRAPH 2001 conference proceedings videotape, we can interactively rotate objects and move our viewpoint, with the irradiance being procedurally recomputed at every frame. We can also rotate the lighting by applying the inverse rotation to the normal $\vec{N}$. Images rendered using our method look identical to those obtained by texture-mapping after precomputing irradiance environment maps.

4.3.3 Representation

Conceptually, the final image is composed of a sum of spherical harmonic basis functions, scaled by the lighting coefficients L_{lm}. These 3D irradiance basis functions depend on the surface normal and are defined over the entire object, making it possible to generate an

[3] A symmetric 4x4 matrix has 10 degrees of freedom. One additional degree is removed since $\vec{N}$ lies on the unit sphere.

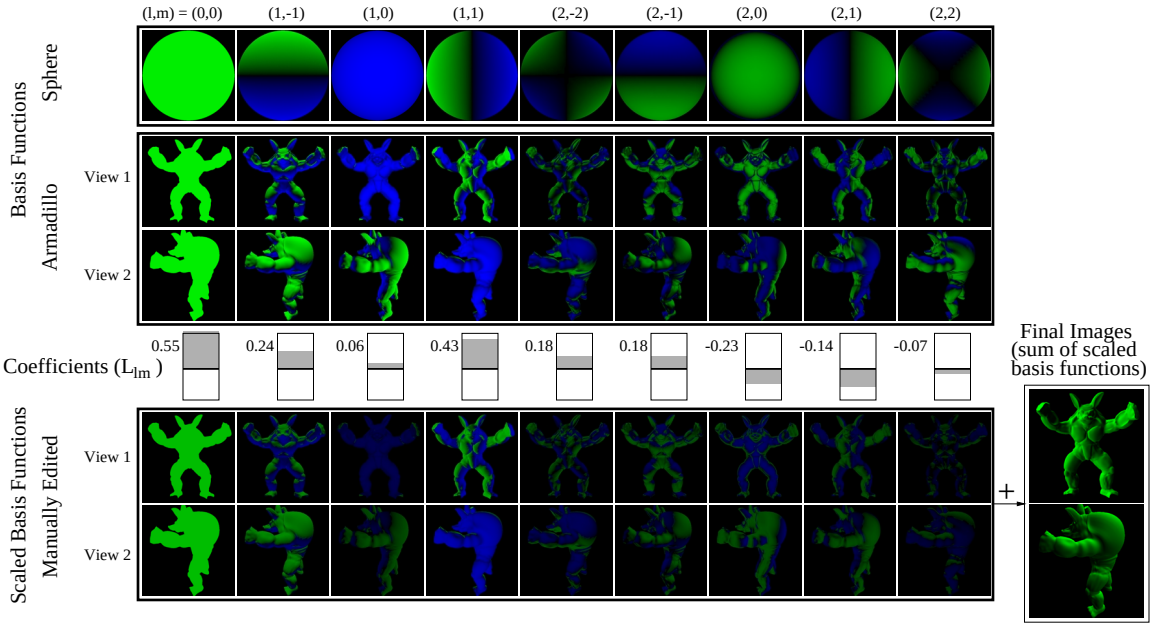

Figure 4.3: *Illustration of our representation, and applications to control appearance. The basis functions have both positive values, shown in green, and negative values, shown in blue. Topmost, we show the spherical harmonic basis functions on a sphere—note that these are actual images, not the coordinate mappings of figure 4.2—and the armadillo. The basis functions are defined over the entire object surface; we show only two views. The rightmost 5 functions are dimmer since they have the highest frequency* $(l = 2)$ *and contribute the least. Conceptually, the basis functions are then scaled by the lighting coefficients* L_{lm} *and added to produce renderings.* L_{lm} *are actually RGB values; for simplicity, we show the coefficients for only one color (green). The coefficients* L_{lm} *may be adjusted manually to manipulate appearance. This editing can be fairly intuitive—for instance, we make* L_{11} *large and positive to darken the right side (with respect to us) and left arm of the armadillo image, since the basis function* $(1, 1)$ *is negative in that region.*

image from any viewpoint. We may also manually adjust the 9 lighting coefficients L_{lm} to directly control appearance, as shown in figure 4.3. The lighting coefficients can often be assigned intuitive meanings. For instance, L_{1-1} is the moment about the vertical or y-axis, and measures the extent to which the upper hemisphere is brighter than the lower hemisphere. As can be seen from figure 4.1, L_{1-1} is usually large and positive, since most scenes are lit from above. By making this value negative, we could give the appearance of the object being lit from below.

Our representation may also be useful in the future for *image-based rendering with varying illumination*. Hallinan [30] and Epstein et al. [18] have observed empirically that,

for a given view, images of a matte object under variable lighting lie in a low-dimensional subspace. Our theory explains this observation, and indicates that a 9D subspace suffices. Basri and Jacobs [2] have obtained similar theoretical results. To synthesize images of a diffuse object under arbitrary illumination, we therefore need only the 9 basis functions, which could be computed from a small number of photographs. Such an approach would significantly speed up both acquisition and rendering in a method such as Debevec et al. [15].

4.4 Conclusions and Future Work

We have described a novel analytic representation for environment maps used to render diffuse objects, and have given explicit formulae for implementation. Our approach allows us to use an arbitrary illumination distribution for the diffuse component of the BRDF, instead of the limitation of current graphics hardware to point or directional sources. We simply specify or compute the first 9 moments of the lighting. Even where more conventional texture-mapping methods are desired, our approach allows us to very efficiently compute irradiance environment map textures.

It should be noted that environment mapping in general, and the methods described in this and the next chapter in particular, are restricted to distant illumination without cast shadows or interreflection. An obvious question is how we can modify and utilize our results when the theoretical assumptions don't exactly hold, i.e. we want to account for some spatially-varying illumination, cast shadows, and interreflection. It is our belief that low-dimensional subspaces may still be appropriate, even if they are not specified by simple analytic formulae in terms of spherical harmonics, or can be described using only 9 parameters. Some preliminary work in this area has already been demonstrated by Sloan et al. [66].

Another interesting question is whether we can use similar frequency-space methods for the specular BRDF component, and more general non-Lambertian reflection functions. One solution to this problem will be presented in the next chapter. In the future, we would also like to further explore the applications to lighting design and image-based rendering discussed in this chapter.