Analytic Spherical Harmonic Coefficients for Polygonal Area Lights

JINGWEN WANG, University of California, San Diego
RAVI RAMAMOORTHI, University of California, San Diego

Spherical Harmonic (SH) lighting is widely used for real-time rendering within Precomputed Radiance Transfer (PRT) systems. SH coefficients are precomputed and stored at object vertices, and combined interactively with SH lighting coefficients to enable effects like soft shadows, interreflections, and glossy reflection. However, the most common PRT techniques assume distant, low-frequency environment lighting, for which SH lighting coefficients can easily be computed once per frame. There is currently limited support for near-field illumination and area lights, since it is non-trivial to compute the SH coefficients for an area light, and the incident lighting (SH coefficients) varies over the object geometry. We present an efficient closed-form solution for projection of uniform polygonal area lights to spherical harmonic coefficients of arbitrary order, enabling easy adoption of accurate area lighting in PRT systems, with no modifications required to the core PRT framework. Our method only requires computing zonal harmonic (ZH) coefficients, for which we introduce a novel recurrence relation. In practice, ZH coefficients are built up iteratively, with computation linear in the desired SH order. General SH coefficients can then be obtained by the recently developed sparse zonal harmonic rotation method.

CCS Concepts: • Computing Methodologies → Computer Graphics; Rendering;

ACM Reference format:

1 INTRODUCTION

Spherical Harmonic (SH) lighting is a popular method in interactive rendering, within Precomputed Radiance Transfer (PRT) systems [Sloan et al. 2002]. While strictly accurate only for low-frequency lighting, the method enables realistic visual effects like soft shadows and glossy reflections at real-time frame rates with dynamic lighting. PRT has been the subject of a substantial body of work (see survey [Ramamoorthi 2009]), and is widely used in real-time applications like video games. In recent years, SH lighting and PRT have also become mainstays of offline rendering, and have been widely used in movie production [Pantaleoni et al. 2010].

The core PRT framework precomputes a transfer function for each vertex on object geometry, taking into account visibility and BRDF effects. This transfer function is then converted (projected) into spherical harmonics and stored on the object, possibly with compression [Sloan et al. 2003]. For soft shadows on a Lambertian surface, this would simply be an SH representation of the visibility in each direction. At run-time, these SH coefficients are combined with the lighting (a simple dot product of SH coefficients of lighting and transfer for visibility). In this paper, we do not modify the pre-computation or core PRT framework at all, but address computation of the SH lighting coefficients. Most PRT methods assume distant, low-frequency environment map lighting. In this case, the incident lighting is the same everywhere in the scene, and SH lighting coefficients can easily be computed once per frame using standard numerical integration, and re-used for each vertex on the object.

On the other hand, there is currently limited support for near-field illumination from area light sources. Indeed, near-field effects imply that the angular distribution of incident radiance is now different at each vertex (location in space), and numerical integration to compute SH lighting coefficients separately for each vertex or pixel is too expensive. In this paper, we address this challenge by developing an efficient closed-form computation for spherical harmonic coefficients of general orders, for an arbitrary polygonal area light.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
0730-0301/2018/8-ART54 $15.00
https://doi.org/10.1145/3197517.3201291
area lights). Our method is exact, does not require numerical integration, is implemented in a few lines of code in a GPU shader, and is fast enough for real-time PRT systems (Fig. 1).

We are inspired by early methods to compute integrals of polynomials against area lights [Arvo 1995; Snyder 1996], although their purpose was different, to enable non-Lambertian effects. Since SH are polynomials, it is possible in principle to use these methods directly. However, there are three concerns. First, they derive recurrence formulae for polynomials, rather than for SH directly. Computing all relevant polynomials and then combining them to find SH coefficients is inefficient, since an SH of order \( n \) may have \( O(n) \) polynomial terms. Second, the recursion can be inefficient for high orders, with the number of recursive steps growing with the order. Third, the simpler PHong formulae of [Snyder 1996] are attractive, but apply only to radially symmetric polynomials, while we need all the SH coefficients. Our contributions, summarized below, address each of these three challenges in order:

**Novel Zonal Harmonic Recurrence Formula:** Our key technical contribution is the derivation of a novel zonal harmonic recurrence formula (Sec. 5) for computing the ZH about a given direction, for a polygonal light. This is a significant generalization of the recurrence for monomials \( x^n \) in [Snyder 1996], and requires keeping track simultaneously of 3 different surface and boundary integrals. While integrals of monomials (as needed for PHong lobes in [Snyder 1996]) naturally lend themselves to recursive formulae due to the repetition of factors in the integrand, such relations are more difficult to find for integrals of polynomials. Our work makes use of the inherent recurrence properties of the Legendre polynomials.

**Iterative Computation of all ZH coefficients:** Unlike previous work, we seek to determine all ZH coefficients up to some order \( n \) (Sec. 4.1). Hence, instead of recursing for each term (which can involve \( O(n) \) complexity for each term), we iteratively compute the ZH terms from constant to lower-order polynomials. Thus, each new ZH coefficient can be computed in \( O(1) \) time, and total computational time and storage is \( O(n) \). While a similar iterative computation is certainly possible for [Snyder 1996] to obtain polynomials up to order \( n \), directly applying that method to compute ZH coefficients would still involve \( O(n^2) \) time, since each ZH of order \( n \) is a linear combination\(^3\) of \( O(n) \) polynomial terms.\(^4\)

**Sparse Zonal Harmonic Rotation for SH coefficients:** Finally, to compute all \((n+1)^2\) SH coefficients at order \( n \) from ZH coefficients up to order \( n \), we leverage the recent sparse zonal harmonic factorization method for efficient SH rotation [Nowrouzezahrai et al. 2012] (Sec. 4.2). This is substantially simpler and more efficient than applying [Arvo 1995] to each polynomial term separately. In particular, we compute the ZH recurrence iteratively for each of the \(2n+1\) lobe directions in [Nowrouzezahrai et al. 2012] (shared among all SH bands). This takes \( O(n^2) \) time, consistent with the total number of SH coefficients. We then apply the fast rotation formula. In principle, this step involves \( O(n^2) \) SH coefficients, each expressed as a linear combination of each ZH lobe direction, and can be \( O(n^2) \). However, there is considerable sparsity, and the rotation is very fast in practice [Nowrouzezahrai et al. 2012], with minimal overhead (Fig. 5). Hence, the algorithm’s effective cost is \( O(n^2) \), linear in the number of SH coefficients.

2 RELATED WORK

We are inspired by recent work on analytic lighting [Heitz et al. 2016], which can efficiently compute direct lighting from area sources. However, they cannot address shadows or other effects. In contrast, traditional PRT systems enable complex light transport, but do not permit near-field area lights. We bridge this gap by computing SH coefficients, thus enabling area light sources with soft shadows and other complex light transport effects. Note that we integrate zonal harmonics, which are more complex functions than BRDF lobes, so we cannot directly apply the methods or transformation matrices in [Dupuy et al. 2017; Heitz et al. 2016].

This paper focuses on SH lighting, which is widely used in practice. All-frequency methods using wavelets [Ng et al. 2003] or radial basis functions [Tsai and Shih 2006] can be more accurate, and require fewer than 100 basis functions at run-time, comparable to SH. However, the full transport matrix, often with tens of thousands of basis functions, must still be precomputed and stored, which has limited practical applications. Note that our method applies to any PRT system using SH lighting, and is orthogonal to the core PRT framework, which need not be modified at all. While we demonstrate our method primarily for visibility precomputation, involving diffuse soft shadows in static scenes, the method can also be included with PRT methods that support glossy reflection and interreflections [Sloan et al. 2003], and dynamic scenes [Sloan et al. 2005].

Near-field illumination in PRT has been included with source radiance fields [Zhou et al. 2005] for all-frequency relighting. However, this requires precomputing a full 5D space-angle radiance field, and limits the ability to change the shape of the light. The affine wavelet method of [Sun and Ramamoorthi 2009] reduces storage by propagating lighting in wavelets, including for textured lights. However, the method is approximate, and lights cannot be rotated out of plane. Another approximate method, in the original paper [Sloan et al. 2002], was to compute incident lighting at a few locations on object geometry and interpolate. In contrast, we compute exact SH area light coefficients at each vertex in a simple GPU shader.

Spherical harmonic exponentiation [Ren et al. 2006] addresses real-time soft shadows in dynamic low-frequency environments by forming spherical approximations of object geometry and lighting, but provides analytic lights only for circular/spherical light sources. Spheres (with circular cross-sections from all views) cannot easily be packed to accurately represent a planar polygon. Since we allow out-of-plane rotations and arbitrary shapes, this is especially true for polygonal lights that have thin/irregular shapes or are at grazing angles to the receiver. Moreover, projection of large numbers of spherical lights might require considering occlusion between spheres. In contrast, our method computes exact SH coefficients.

As noted earlier, we build on [Arvo 1995; Snyder 1996]. Like them, we assume uniform polygonal planar area lights. We do not explicitly handle texture, but in some cases, can do so by breaking the light...
We differ conceptually in simultaneously computing integrals of all while we demonstrate real-time performance. Although our paper focuses on PRT, the various other applications mentioned by [Belcour et al. 2018] apply to our method as well, such as basis function projection, hierarchical sample warping, and use in offline rendering.

3 BACKGROUND

In this section, we introduce some relevant background. We first introduce the basic reflection and PRT equations for direct lighting, focusing on Lambertian surfaces with soft shadowing for simplicity (more complex light transport can also be included, since our method is orthogonal to the core PRT framework). We then briefly introduce the spherical harmonics and integration against an area light, ending with some key SH and Legendre polynomial formulas. Table 1 summarizes notation for key symbols used in the paper.

### 3.1 Precomputed Radiance Transfer

The reflection equation for direct lighting can be written,

\[ L'(x, \omega_r) = \int_{Q} L'(x, \omega_i) V(x, \omega_i) \rho(\omega_i, \omega_r, n) \max(\omega_i \cdot n, 0) \, d\omega_i, \]

where \( x \) is the spatial coordinate of the current vertex or pixel, \( (\omega_i, \omega_r) \) are the global incoming and outgoing directions, and \( n \) is the surface normal. \( L' \) is the reflected radiance or image intensity, while \( L' \) is the incident radiance. \( V \) is the visibility and \( \rho \) is the BRDF.

### 3.2 Area Light Integration

Note that \( T_{lm} \) is precomputed, based on numerical integration in the standard way for PRT methods. The major contribution of this paper is to derive a closed form formula for \( L_{lm} \) from polygonal area lights. In particular, we seek a solution to

\[ L_{lm} = \int_{Q} Y_{lm}(\omega_i) \, d\omega_i, \]

where the integral is over the projection of the polygon \( Q \) onto the unit sphere. Note that we work with the real (not complex) spherical harmonics. In addition, we have dropped the spatial coordinate \( x \) in \( L_{lm} \), since it is easy to translate the polygon so \( x \) is at the origin; we denote this using \( Q_x \), but will drop the subscript in Sec. 4 to simplify notation.

We will first seek to determine the zonal harmonic coefficients \( (m = 0) \) corresponding to a particular central direction \( \omega \),

\[ L_l(\omega) = \int_{Q_x} Y_{lm}(\omega) \, d\omega_i. \]

### 3.3 Spherical Harmonic Formulas

Finally, we briefly discuss the mathematical forms we use for the spherical harmonics and zonal harmonics in this paper, along with relevant identities. Consider the geographic angular parameterization of the sphere \((\theta, \phi)\) with Cartesian components, \((x, y, z) = \)

\[ \begin{align*}
Y_{lm} & \quad \text{The SH basis function of degree } l, \text{ index } m \\
P_l & \quad \text{Legendre polynomial of degree } l \\
Q & \quad \text{The polygonal light (projected to unit sphere)} \\
\omega_e & \quad \text{Projection of vertex } e \text{ onto unit sphere} \\
\gamma_e & \quad \text{Angle between polygonal vertices } e \text{ and } e + 1 \\
\omega & \quad \text{Central axis of zonal harmonic lobe} \\
\omega^m_{lm} & \quad \text{Weight of } d \text{-th ZH lobe used to reconstruct } Y_{lm} \\
S_l & \quad \text{The surface integral of the } l \text{-th Legendre polynomial} \\
P_l(\omega \cdot \omega_i) & \quad \text{The edge integral} \\
B_{l,e} & \quad \text{The intermediate edge integral} \\
C_{l,e} & \quad \text{The intermediate edge integral} \\
D_{l,e} & \quad \text{The intermediate edge integral}
\end{align*} \]

| \( Y_{lm} \) | The SH basis function of degree \( l \), index \( m \) |
| \( P_l \) | Legendre polynomial of degree \( l \) |
| \( Q \) | The polygonal light (projected to unit sphere) |
| \( \omega_e \) | Projection of vertex \( e \) onto unit sphere |
| \( \gamma_e \) | Angle between polygonal vertices \( e \) and \( e + 1 \) |
| \( \omega \) | Central axis of zonal harmonic lobe |
| \( \omega^m_{lm} \) | Weight of \( d \)-th ZH lobe used to reconstruct \( Y_{lm} \) |
| \( S_l \) | The surface integral of the \( l \)-th Legendre polynomial |
| \( P_l(\omega \cdot \omega_i) \) | The edge integral |
| \( B_{l,e} \) | The intermediate edge integral |
| \( C_{l,e} \) | The intermediate edge integral |
| \( D_{l,e} \) | The intermediate edge integral |

Table 1. Table of notation for key symbols.
In this section, we introduce our algorithm for computing the analytic SH coefficients for a polygonal area light source, i.e., determining $L_l$ in Sec. 3 below. The key technical contribution is the novel zonal harmonic recurrence, which is derived later in Sec. 3. While the mathematics and derivation is somewhat involved, the actual implementation is simple, as discussed at the end of Sec. 3 and shown in the pseudocode in Algorithm 4. In practice, the algorithm is implemented in a simple GPU vertex shader, and the source code is publicly available at http://viscomp.ucsd.edu/projects/ash.

4 ALGORITHM

In this section, we introduce our algorithm for computing the analytic SH coefficients for a polygonal area light source, i.e., determining $L_l$ in equation 5. The key technical contribution is the novel zonal harmonic recurrence, which is derived later in Sec. 3. While the mathematics and derivation is somewhat involved, the actual implementation is simple, as discussed at the end of Sec. 3 and shown in the pseudocode in Algorithm 4. In practice, the algorithm is implemented in a simple GPU vertex shader, and the source code is publicly available at http://viscomp.ucsd.edu/projects/ash.

Fig. 2. Illustration of the integrals we keep track of (surface, boundary). In (a), we show the projection of the polygon to the unit sphere with support $Q$. In (b) we show the corresponding surface integral $S_l$, while (c) shows the boundary integral $B_l$. Note that $B_l$ can be broken into edges or arcs $B_{l,e}$ as shown in (d) using a very simple parameterization. We also keep track of intermediate boundary integral $D_{l,e}$, and compute $C_{l,e}$ on the fly.

4.1 Iterative Computation of Zonal Harmonic Coefficients

We first develop an iterative algorithm to compute the zonal harmonics coefficients, $L_l$ (equation 6). Doing so requires iteratively updating the surface integral, as well as two boundary integrals (a third boundary integral is computed on the fly). Figure 2 shows the quantities we keep track of. We start by defining the integrals we need and relevant notation and quantities, then discuss the base cases, and finally state the recurrence relation. Then, we summarize the iterative algorithm with reference to the pseudocode in Alg. 4.

Surface and Boundary Integrals: We assume the polygon has been rotated by $-x$, so the spatial location of the vertex is at the origin, as shown in Fig. 2(a), and omit explicit dependence on $x$ in what follows. We are now primarily interested in the surface integral (Fig. 2(b)),

$$S_l = \int_Q P_l(\omega \cdot \omega) d\omega, \quad L_l = \sqrt{\frac{2l+1}{4\pi}} S_l. \quad (13)$$

$S_l$ simply integrates the projection of the area light on the unit sphere against Legendre polynomials $P_l$ of the dot product $\omega \cdot \omega$. We can multiply in the (precomputed) normalization constant at the end to compute final ZH coefficient $L_l$. Note that $S_l$ is computed separately for each lobe direction $\omega_{l,d}$ (there may be $2n + 1$ such directions). For simplicity of notation, we just use $\omega$, as in equation 6, and omit the explicit dependence of $S_l$ on $\omega$.

We will be deriving a recurrence for $S_l$ in terms of boundary integrals over the contours of the polygon’s projection, denoted by

$$\left(\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta \right).$$

The real spherical harmonics are,

$$Y_{lm}(\theta, \phi) = \frac{(2l+1)(l-m)!}{4\pi(l+m)!} P^m_l(\cos \theta) f(l|m|\phi), \quad (7)$$

where $P_{lm}$ are the associated Legendre polynomials and the trigonometric function $f(l|m|\phi)$ is set to 1 when $m = 0$, given by $\sqrt{2} \cos(m\phi)$ when $m > 0$ and $\sqrt{2} \sin(m\phi)$ when $m < 0$. In particular, the zonal harmonics (ZH) for $m = 0$ are given by,

$$Y_{l0}(\cos \theta) = \sqrt{\frac{2l+1}{4\pi}} P_l(\cos \theta), \quad (8)$$

where we parameterize by $\cos \theta = z$, and $P_{l}(z)$ are simply the Legendre polynomials of degree $l$. For example $P_{0}(z) = 1$, $P_{1}(z) = z$ and $P_{2}(z) = \frac{1}{2}(3z^2 - 1)$. Note that the ZH are radially symmetric and independent of $\phi$, depending only on $z$ in a Cartesian coordinate system. To consider ZH lobes (as a function of incident direction $\omega_i = (\theta, \phi)$) about an arbitrary axis $\omega$, we can simply use a dot product with that axis, defining the ZH as $Y_{l0}(\omega \cdot \omega_i)$ as in equation 6.

A key result relating SH and ZH enables sparse ZH factorization of SH, and allows for fast rotation of ZH into SH. Nowrouzezahrai et al. 2012 show that for any $m$ and any order $l$ for consistency with equation 6 we use $\omega_i = (\theta, \phi),$

$$Y_{lm}(\omega_i) = \sum_d c_{lm,d}^i Y_{l0}(\omega_{l,d} \cdot \omega_i). \quad (9)$$

That is, an SH of degree $l$ can always be written as a sum of rotated ZH basis functions. While $c_{lm,d}$ can in principle range from $-l$ to $l$, involving $2l+1$ lobes, the actual number of lobes needed is usually much smaller. Moreover, the same lobes are used across a band $l$, and in fact one can optimize to share directions $\omega_{l,d}$ across bands. As such, there are only $2n + 1$ unique lobe directions $\omega$ across all bands, and the weights $c_{lm,d}$ are very sparse. Nowrouzezahrai et al. 2012 have already precomputed the lobe directions and weights.

Finally, we will make use of some well-known relations for Legendre polynomials in the subsequent derivations in Sec. 3 below, $P'_{l}(z)$ stands for the derivative $\frac{d}{dz}P_{l}(z)$,

$$z^2 - 1 = P'_{l}(z) = zP_{l}(z) - P_{l-1}(z) \quad (10)$$

$$P_{l}(z) = (2l - 1)zP_{l-1}(z) - (l - 1)P_{l-2}(z) \quad (11)$$

$$(2l + 1)P_{l}(z) = P'_{l+1}(z) - P'_{l-1}(z) \quad (12)$$

**Versions of equation 6 appear earlier in other fields, going back at least to Stern 1965.**
\( \partial Q \) (Fig. 2c). Specifically, we keep track of

\[
B_l = \oint_{\partial Q} P_l(\omega \cdot \omega_l)(\hat{x}_l d\hat{y}_l - \hat{y}_l d\hat{x}_l) = \sum_{e=1}^{M} c_e B_{l,e}. \tag{14}
\]

Note that \( B_l \) is a sum over sub-integrals \( B_{l,e} \) for the \( M \) edges of the polygon. The coefficient \( c_e \) corresponds to the differential measure, which will be shown to be a constant over an edge, and a formula for \( c_e \) is given later in equation \( 20 \).

The differential measure in the integral is expressed in terms of Cartesian components of \( \omega_l \), with respect to \( \omega \). The tildes on \( \hat{x} \) and \( \hat{y} \) denote these are with respect to a coordinate frame where \( \omega \) is the \( Z \) axis. One can also express the differential measure in terms of a cross product between \( \omega_l \) and \( d\omega_l \), taking the component along \( \omega \),

\[
P_l(\omega \cdot \omega_l)(\hat{x}_l d\hat{y}_l - \hat{y}_l d\hat{x}_l) = P_l(\omega \cdot \omega_l)[\omega \cdot (\omega_l \times d\omega_l)],
\]

For the differential measure, \( d\omega_l \) has a simple form in the appropriate parameterization. Let us define \( \omega_e \) as the projection on the unit sphere of \( \omega_l \), so that \( \omega_e \) connects \( \omega_l \) and \( \omega_{l+1} \), as shown in Fig. 2d. We seek to parameterize this edge, as the arc of a great circle on the sphere, to evaluate \( B_{l,e} \). If the (great-circle) arc were in the \( XZ \) plane with \( \omega_l = Z \), a simple parameterization would be \( \omega_l(y) = Z \cos y + X \sin y \) where \( y \) ranges from 0 to the angle between \( \omega_l \) and \( \omega_{l+1} \). In the general case, the vector \( Z \) is simply replaced by \( \omega_e \), while the vector \( X \) is replaced by a new vector \( \lambda_e \), given through cross products by

\[
\lambda_e = \frac{\omega_e \times \omega_{e+1}}{\|\omega_e \times \omega_{e+1}\|} \times \omega_e = \frac{\omega_{e+1} - \omega_e (\omega_e \cdot \omega_{e+1})}{\|\omega_e \times \omega_{e+1}\|} \times \omega_e. \tag{16}
\]

We are simply creating a vector of unit length orthogonal to \( \omega_e \), so that \( \omega_{e+1} \) or any intermediate point along the edge (arc), can be expressed as a linear combination of \( \omega_e \) and \( \lambda_e \).

\[
\omega_e(y) = \omega_e \cos y + \lambda_e \sin y, \tag{17}
\]

where \( y \) ranges from 0 for \( \omega_e \) to \( \cos^{-1}(\omega_e \cdot \omega_{e+1}) \) for \( \omega_{e+1} \). This parameterization for the integral is visualized in Fig. 2d. \( \omega_e \) and \( \lambda_e \) are computed once, and reused for all lobes \( \omega_l \).

We can now express \( B_{l,e} \) from equation 14 using the integrand in equation 12 and the expression for \( \omega_e \) above,

\[
P_l(\omega \cdot \omega_l) = P_l((\omega_e \cdot \lambda_e) \cos y + (\omega_e \cdot \lambda_e) \sin y). \tag{18}
\]

For the differential measure, \( d\omega_l = -\omega_e \sin y dy + \lambda_e \cos y dy \),

\[
\omega \cdot (\omega_l \times d\omega_l) = \omega \cdot (\omega_e \times \lambda_e) dy = \omega_e \cdot \mu_e dy, \tag{19}
\]

where we define \( \mu_e = \omega_e \times \lambda_e = \frac{\omega_e \times \omega_{e+1}}{\|\omega_e \times \omega_{e+1}\|} \) to create a coordinate frame \( (\omega_e, \lambda_e, \mu_e) \). It is possible to derive a number of scalar quantities, corresponding to the dot-products above, with \( (a_e, b_e, c_e) \) the coordinates of \( \omega \) with respect to this coordinate frame,

\[
a_e = \omega \cdot \omega_e \]
\[
b_e = \omega \cdot \lambda_e \]
\[
c_e = \omega \cdot \mu_e \]
\[
z_e = a_e \cos y + b_e \sin y \]
\[
\gamma_e = \cos^{-1}(\omega_e \cdot \omega_{e+1}). \tag{20}
\]

Finally, it is now possible to write the edge integral \( B_{l,e} \) from equations 14 and 15 simply as (using equations 18,20),

\[
B_{l,e} = \int_0^{\gamma_e} P_l(z_e) dy. \tag{21}
\]

Note that \( B_l \) in equation 14 involves an additional factor \( c_e \) for each edge, corresponding to the differential measure.

The following quantities are intermediate edge integral computations used to find the boundary integral \( B_{l,e} \),

\[
C_{l,e} = \int_0^{\gamma_e} z_e P_l(z_e) dy \tag{22}
\]
\[
D_{l,e} = \int_0^{\gamma_e} P_l'(z_e) dy. \tag{23}
\]

**Base Cases:** For iterative computation, we need to establish the base cases, starting with the surface integral. \( S_0 \) is simply the solid angle subtended by the polygon, which can be obtained from the angles between edges [Arvo1995],

\[
S_0 = \sum_{l=1}^M \cos^{-1}((\omega_{l-1} \times \omega_l) \cdot (\omega_l \times \omega_{l+1})) -(M-2)\pi \tag{24}
\]

Note the standard convention that vertex indexing is cyclic, so \( \omega_{M+1} = \omega_0 \) and \( \omega_M = \omega_0 \).

Next, the value of \( S_1 \) just corresponds to the computation of irradiance [Baum et al.1989], which in this case is simply given by the corresponding boundary integral,

\[
S_1 = \frac{1}{2} B_0 = \frac{1}{2} \sum_{e=1}^{M} c_e B_{0,e} = \frac{1}{2} \sum_{e=1}^{M} c_e \gamma_e. \tag{25}
\]

The base case for the boundary integral is a well known result for irradiance, and follows directly from equation 21

\[
B_{0,e} = \gamma_e. \tag{26}
\]

Similarly, noting that \( P_l(z_e) = z_e = a_e \cos y + b_e \sin y \),

\[
B_{1,e} = a_e \sin \gamma_e - b_e \cos \gamma_e + b_e. \tag{27}
\]

Although easy to find, we do not actually need a base case for \( C_{l,e} \) in equation 22 since we derive an explicit non-recursive formula for it in terms of the other boundary integrals. For \( D_{l,e} \) in equation 23 since it involves derivatives, base cases are related to those for \( B_{l,e} \),

\[
D_{0,e} = 0 \quad D_{1,e} = \gamma_e \quad D_{2,e} = 3B_{1,e} = 3(a_e \sin \gamma_e - b_e \cos \gamma_e + b_e) \tag{28}
\]

**Recurrence Relation:** In Sec. 3 we derive the recurrence relations for surface and boundary integrals. Combined with the base cases above, these enable us to start with \( S_0, S_1, B_0, B_1 \) and \( D_0, D_1, D_2 \), iterating to build up higher values of \( S_l \) and \( B_l \) until we reach \( S_M \). Specifically, we derive that

\[
S_l = \frac{2l-1}{l(l+1)} B_{l-1} + \frac{(l-2)(l-1)}{l(l+1)} S_{l-2}. \tag{29}
\]

This is a direct \( O(1) \) iterative expression for \( S_l \) assuming \( B_{l-1} \) and \( S_{l-2} \) have already been computed. However, to compute \( S_{l+1} \) we will also need an iterative formula for the boundary integral, \( B_l \), which is derived for each edge as,

\[
B_{l,e} = \frac{2l-1}{l} C_{l-1,e} - \frac{l-1}{l} B_{l-2,e}. \tag{30}
\]
An explicit formula for the boundary integral $C_{l-1,e}$ is derived in the appendix, 
\[
C_{l-1,e} = \frac{1}{l}((a_e \sin \gamma_e - b_e \cos \gamma_e) P_{l-1}(a_e \cos \gamma_e + b_e \sin \gamma_e) 
+ b_e P_{l-1}(a_e) + (a_e^2 + b_e^2 - 1) D_{l-1,e} + (l-1)B_{l-1,e}) 
\]
(31)

Finally, there is a simple recurrence for $D_l$, 
\[
D_{l,e} = (2l - 1)B_{l-1,e} + D_{l-2,e}. 
\]
(32)

After we compute the base cases, we iterate $2 \leq l \leq n$, storing the values of $S_l$ for increasing orders, while also updating $B_l$ and $D_l$ per the recurrence formulas above. Note that each iteration for increasing $l$ is constant time, and the whole time and storage complexity is $O(n)$. Further note that $C_{l-1,e}$ is computed explicitly on the fly and does not require storage. Moreover, we do not need to store all values of the boundary integrals $B$ and $D$. Indeed, we need only keep the three intermediate values of $D_{l-2,e}, D_{l-1,e}, D_{l,e}$, and can discard older values, with a similar strategy employed for $B$.

**Iterative Computation:** The pseudocode in Algorithm 1 details the iterative computation, given spatial location $x$ and a polygon with vertices $\nu_e$ of $M$ sides. The lobe weights $a_{l,d}^m$ for ZH rotation, along with lobe directions $\omega_{l,d}$ are also inputs. We compute SH coefficients up to order $n$.

First (Line 2), we translate the polygon vertices by $-x$ and project onto the unit sphere, obtaining the vertices $\omega_e$ on the sphere. The quantities $\lambda_e, \mu_e = a_e \times \lambda_e$ and $\gamma_e$ are also pre-calculated at this stage for each edge (Lines 3-7). Finally, the solid angle of the polygon $S_0$ is computed (Line 8). We now enter the iterative computation, which is repeated for each lobe direction $\omega_l$ (Line 9). We calculate $a_e, b_e$ and $c_e$ as per equation (20) (Line 12), and the base cases for $B_0,e, B_1,e$ (Lines 14-15) and $D_1,e, D_2,e$ (Line 16). For the base case $S_l$, we must sum over all lobe edges (Lines 10,13).

Then, we iterate $2 \leq l \leq n$ (Line 18), and update the boundary integrals for each edge (Lines 20-25). Specifically we compute $C_{l-1,e}$ (Line 21), and apply the recurrence formulae for $B_l,e$ (Line 22) and $D_l,e$ (Line 24). The boundary integral $B_l$ is updated by summing over edge integrals $B_{l,e}$ (Lines 19, 23). For clarity, we do not include a number of practical storage optimizations. Only a single variable $C$ needs to be used for $C_{l-1,e}$ since it is computed explicitly. As discussed earlier, we only need to maintain the three most recent values for $B$ and $D$. Finally, we update the surface integral $S_l$ (Line 26) and include the normalization factor to compute the zonal harmonic coefficients $L_l(\omega)$ for each lobe (Line 27).

The last stage is to use the sparse zonal harmonic rotation formula to get all spherical harmonic coefficients $L_{lm}$ from ZH coefficients $L_l(\omega)$, as discussed in Secs. 4.2 (Lines 30-35).

**Discussion:** We briefly compare our recurrence with that derived in [Snyder 1996] for Phong lighting. Ours is a recurrence directly on Legendre polynomials rather than monomials in [Snyder 1996]. Thus, we directly have the ZH coefficients. In contrast, if we had used the monomial recurrence from [Snyder 1996], we would have had to separately compute and sum $O(n)$ monomial integrals for each ZH coefficient for each lobe, with a total complexity $O(n^3)$. Indeed, our direct Legendre polynomial recurrence is the key contribution of the paper. Note that the derivation and recursive formula is more complicated since we are dealing with polynomial rather than monomial terms, requiring us to simultaneously keep track of the surface integral $S$ and the boundary integrals $B$ and $D$, while a third boundary integral $C$ can be evaluated explicitly. Nevertheless, the actual implementation is quite straightforward, involving only very simple formulae, as shown in the equations above, and the relatively simple pseudocode in Algorithm 1.

**4.2 Sparse Zonal Harmonic Rotation**

We now proceed to compute all the SH coefficients $L_{lm}$ given the ZH coefficients $L_l$ from the recurrence relation (recall that $L_l$ is easily obtained from $S_l$ using equation 15). In fact, the same linearity relation in equation (9) also applies to the coefficients, since they involve linear integrals of the spherical harmonics. Therefore, as

```
Algorithm 1 Pseudo code of algorithm for SH coefficients.
1: function COMPUTECOEFFICIENTS(x, ω_{l}, M, n, a_{l,d}, ω_l)
2: for all e: \nu_e = \nu - x; \omega_e = \frac{\nu_e}{\|\nu_e\|} ➔ Project to Sphere
3: for e = 1 to M do ➔ Pre-Calculate per edge
4: λ_e = \frac{\omega_{l} \times \omega_{l+1}}{\|\omega_{l} \times \omega_{l+1}\|} \times \omega_e ➔ Eq. 16
5: µ_e = \omega_e × λ_e ➔ Frame (ω_e, λ_e, µ_e)
6: γ_e = \cos^{-1}(\omega_e \cdot ω_{e+1}) ➔ Angle of edge e ➔ Eq. 20
7: end for
8: S_0 = SolidAngle(ω_{l}, M) ➔ Eq. 24
9: for all ω in ω_l do ➔ Up to 2n+1 for all bands
10: S_l = 0 ➔ Initialize S_l
11: for e = 1 to M do ➔ Base Cases
12: a_e = ω • ω_e; b_e = ω • λ_e; c_e = ω • µ_e ➔ Eq. 20
13: S_l = S_l + \frac{1}{e} c_e \gamma_e ➔ Eq. 29
14: B_{0,e} = γ_e ➔ Eq. 26
15: B_{1,e} = a_e \sin γ_e - b_e \cos γ_e + b_e ➔ Eq. 27
16: D_{0,e} = 0; D_{1,e} = γ_e + D_{2,e} = 3B_{1,e} ➔ Eq. 28
17: end for
18: for l = 2 to n do ➔ Each degree 2 ≤ l ≤ n
19: B_0 = 0 ➔ Initialize B_l
20: for e = 1 to M do ➔ Edge Integrals
21: C_{l-1,e} = C(l, a_e, b_e, B_{l-2,e}, D_{l-1,e}) ➔ Eq. 51
22: B_{l,e} = \frac{l+1}{l+2} C_{l-1,e} - (l-1)B_{l-2,e} ➔ Eq. 30
23: B_l = B_l + c_e B_{l,e} ➔ Eq. 14
24: D_{l,e} = (2l - 1)B_{l-1,e} + D_{l-2,e} ➔ Eq. 32
25: end for
26: S_l = \frac{2l+1}{4l+2} B_{l-1,e} + \frac{(l-2)(l-1)}{4l+2} S_{l-2} ➔ Eq. 29
27: L_l(ω) = \sqrt{\frac{2l+1}{4π}} S_l ➔ Eq. 13
28: end for
29: for each SH basis function (l, m) do ➔ ZH rotation
30: L_{lm} = 0 ➔ Initialize L_{lm}
31: for each d ∈ δ(l) do ➔ Sparse set of lobes ω_{l,d}
32: L_{lm} = L_{lm} + a_{l,d}^m L_l(ω_{l,d}) ➔ Eq. 33
33: end for
34: end for
35: end for
36: end function
```
We now need to convert this to the notation of the previous section. We begin with the derivation of equation 29. Consider the Legendre

\[ P_l(\omega) = \sum_{m=-l}^{l} a_{l,m} \omega^m. \]

While \( d \) can take \( 2l+1 \) values in the general case (and there are a total of \( 2n+1 \) lobe directions \( \omega \) shared among all bands), there is considerable sparsity in practice for the set of lobes \( d(l) \) associated with SH order \( l \). Moreover, the weights \( a_{l,d} \) are known beforehand, as a basic property of spherical harmonics. Therefore, the rotation in equation 33 is very efficient.

In terms of computation and storage, computing all the ZH takes \( O(n) \) time, and is run for each of \( O(n) \) lobe directions \( \omega \) to compute \( L_l(\omega) \), taking \( O(n^2) \) time and space, linear in the total number of SH coefficients \( L_{lm} \), which is also \( O(n^2) \). (Of course, there is also a \( O(M) \) factor for the number of edges of the polygon.) Since the rotation coefficients \( a_{l,d}^m \) are sparse, the final step in equation 33 is very fast, with overall complexity essentially linear in the number of SH coefficients.

5 DERIVATION OF ZONAL HARMONIC RECURRENT

In this section, we derive the zonal harmonic recurrence in equations 29-32. Note that while this section is mathematically involved, the actual algorithm is quite simple, as shown in Algorithm 1. Readers more interested in implementation may want to skip ahead to the results in Sec. 6 on a first reading.

5.1 Main Recurrence

We begin with the derivation of equation 29. Consider the Legendre

\[ P_l(z) = \sum_{m=-l}^{l} a_{l,m} z^m. \]

Integrating over \( Q \) (i.e., with each term now as an integrand), we obtain

\[ \int_Q P_l(z) \, dz = \left( \frac{2l+1}{l} \right) \int_Q z P_{l-1}(z) \, dz - \left( \frac{l-1}{l} \right) \int_Q z P_{l-2}(z) \, dz. \]

We now need to convert this to the notation of the previous section, and in particular the surface integral \( S_l \) in equation 13. The main issue is that the canonical form above uses \( z \) for the argument of \( P_l \), while \( S_l \) uses \( \omega \cdot \omega \). For our derivation, we can assume, without loss of generality, that we have rotated the coordinate system so \( \omega \)

aligned with the Z axis, while explicitly preserving \( \omega \) for the actual algorithm in Sec. 4. We can now view equation 34 as a relation for the surface integral,

\[ S_l = \left( \frac{2l-1}{l} \right) \int_Q z P_{l-1}(z) \, d\omega - \left( \frac{l-1}{l} \right) S_{l-2}. \]

It is clear that the expression on the right is a previously-computed value. Now we show how to compute \( \int_Q z P_{l-1}(z) \, d\omega \). We use Stokes’ Theorem to convert this into a boundary integral. While this approach has been used in many previous works, the form of our integrals and recurrence is more complicated, requiring a much more involved calculation.

Stokes’ theorem states that for a suitable vector \( V \),

\[ \int_Q N \cdot (\nabla \times V) \, d\omega = \oint_{\partial Q} V \cdot d\mathbf{r}, \]

where as usual, \( N \) is simply the normal at \( \omega \) on the region \( Q \) of the sphere, with Cartesian components \( (x, y, z) \), and \( d\mathbf{r} \) is a differential curve segment on the boundary \( \partial Q \). To convert the integral expression in equation 35 to a boundary integral, we need to find the appropriate vector \( V \), and further calculations are still required. Specifically, we use

\[ V = \begin{pmatrix} -y P_{l-1}(z) \\ x P_{l-1}(z) \\ 0 \end{pmatrix}. \]

We take the curl, as in the left-hand side of equation 36,

\[ N \cdot (\nabla \times V) = -x \frac{d}{dz} P_{l-1}(z) - y \frac{d}{dz} P_{l-1}(z) + 2z P_{l-1}(z) \]

\[ = (z^2 - 1) P'_{l-1}(z) + 2z P_{l-1}(z), \]

where we use \( P' \) for \( dP/dz \). We now apply equation 10 to rewrite \( P' \) in the right-hand side of the above expression,

\[ N \cdot (\nabla \times V) = (l + 1) z P_{l-1}(z) - (l - 1) P_{l-2}(z). \]

We can now apply Stokes’ Theorem, converting the surface integral for the quantity above to a boundary integral. For simplicity of notation, we drop the argument \( z \) to the Legendre polynomials in the remainder of this subsection,

\[ (l + 1) \int_Q z P_{l-1}(z) \, d\omega - (l - 1) \int_Q P_{l-2}(z) \, d\omega = \oint_{\partial Q} P_{l-1}(z)(xdy - ydx). \]

We can now view the surface integral \( \int_Q z P_{l-1}(z) \, d\omega \) in equation 35 including the normalization \( \frac{2l-1}{l} \) in equation 35 and rearranging terms in equation 10,

\[ \frac{2l-1}{l} \int_Q z P_{l-1}(z) \, d\omega = \frac{2l-1}{l(l+1)} \int_Q P_{l-1}(z)(xdy - ydx) + \frac{2l-1}{l(l+1)} \int_Q P_{l-2}(z) \, d\omega. \]

Substituting this result back into equation 35 gives

\[ S_l = \frac{2l-1}{l(l+1)} B_{l-1} + \frac{2l-1}{l(l+1)} \frac{(l-1)}{l} S_{l-2} - \frac{(l-1)}{l} S_{l-2}. \]

which is exactly the main recurrence in equation 29.
ware and fetched within the shader program. This effectively allows
the algorithm to be implemented in a simple GPU vertex shader, which
computes SH coefficients separately at each vertex, following the
typical method for SH lighting (Sec. 4.2). We store pre-tabulated values of
Legendre polynomials in 1D textures, which are interpolated in hardware
and fetched within the shader program. This effectively allows the
Legendre polynomial evaluations in equation 31 to be evaluated
in constant time. Once the SH lighting coefficients are computed,
they can directly be used in the PRT system. Moreover, the SH lighting
coefficients can also be added to those from an environment map or from other lights to render images with multiple area light
sources and environment lighting (Figs. 10, 19).

Analysis of Accuracy: The spherical harmonic formula in Sec. 4 is exact. To verify the correctness of the formula and its deriva-
tion, we compare with numerical integration with 146k samples
per vertex using stratified Monte Carlo sampling computed offline
in Fig. 3. We also conducted a number of other similar tests with
different scenes, lights and SH orders. Figure 3 uses SH up to
degree n = 8 (81 spherical harmonic terms) to show that we get
high accuracy for general degree SH. A comparison of analytic and
numerically computed images shows they are identical to within
one gray level. To explore further, we consider the central scanline,
plotting overall intensity, which is again essentially identical; the
differences between the two results are at most one grey level, and displayed
here at 10 times the intensity. For further verification, we also plot the
intensity along the central image scanline, where the analytic and numerical
curves agree almost perfectly.

5.2 Boundary Integrals
Now we derive a recurrence for the boundary integral \( B_l \), which
is simply a sum of edge integrals \( B_{l,e} \), as in equation 21. We are
interested in \( B_{l,e} = \int_0^{\pi/\gamma} P_l(z_e) \, dy \), where as usual we have that
\( z_e = a_e \cos y + b_e \sin y \). We first rewrite \( B_{l,e} \) using equation 11
\[
\int_0^{\pi/\gamma} P_l(z_e) \, dy = \frac{21 - 1}{l} \int_0^{\pi/\gamma} z_e P_{l-1}(z_e) \, dy - \frac{l - 1}{l} \int_0^{\pi/\gamma} P_{l-2}(z_e) \, dy.
\]  
(43)

The integrals correspond respectively to \( B_{l,e} \) on the left-hand side,
and \( C_{l-1,e} \) and \( B_{l-2,e} \) on the right-hand side. Considering the nor-
malization factors, this is exactly the recurrence relation for \( B_{l,e} \) in
equation 50. It is clear that the second term on the right has been
already computed. The first term on the right corresponds to
\( C_{l-1,e} = \int_{z_e} P_{l-1}(z_e) \, d\gamma \). In the appendix, we derive an explicit
formula for \( C_{l,e} \), using integration by parts, deriving equation 31.

Finally, to compute \( D_{l,e} \), we can use equation 12, which we
rewrite as \( (2l - 1) P_{l-1}(z_e) = P_l'(z_e) - P_{l-2}'(z_e) \). Considering the corre-
sponding integrals and rearranging,
\[
\int_0^{\pi/\gamma} P_l'(z_e) \, dy = (2l - 1) \int_0^{\pi/\gamma} P_{l-1}(z_e) \, dy + \int_0^{\pi/\gamma} P_{l-2}'(z_e) \, dy.
\]  
(44)

This can be written as \( D_{l,e} = (2l - 1) B_{l-1,e} + D_{l-2,e} \), which gives the recurrence in equation 32.

6 IMPLEMENTATION AND RESULTS
The algorithm is implemented in a simple GPU vertex shader, which
computes SH coefficients separately at each vertex, following the
pseudocode in Algorithm 1. We store pre-tabulated values of Le-
gende polynomials in 1D textures, which are interpolated in hardware
and fetched within the shader program. This effectively allows the
Legendre polynomial evaluations in equation 31 to be evaluated
in constant time. Once the SH lighting coefficients are computed,
optimized compiles also give roughly (but not exactly) linear performance with 8 SH coefficients, as desired. Finally, Fig. 7 plots normalized runtime time complexity of our method is $O(n)$ total runtime at order $n$, a small overhead, which is negligible at lower orders (not noticeable in the total number of SH coefficients (Sec. 4.1), is the major time complexity is linear in the number of edges (or vertices) of the polygonal area light.

We now discuss actual wall-clock running times, using an NVIDIA GeForce GTX 1080 Ti GPU, with compiler optimizations enabled. For the plant scene in Fig. 5 with 163K triangles and one rectangular area light, at SH order $n = 8$, our method runs at 350 frames per second. We also compared with using the monomial recurrence in [Snyder 1996] (the direct computation per [Arvo 1995] would be even slower). To do so, we used our iterative method to compute the monomials instead of the ZH, followed by combining them to compute the ZH coefficients, and then using our sparse SH rotation. Note that this is not the method of [Snyder 1996], since we only use their recurrence, and only one part differs from ours (computation of monomials rather than directly using the ZH recurrence). That method ran about 4x slower, at 91fps because of the time complexity in linearly combining monomials to obtain ZH. While the compiler can optimize the required additions to make them very fast, the extra $O(n)$ complexity for linear combinations of monomials still adds significant overhead. Table 2 shows additional runtime comparisons on a variety of scenes. The complexity depends on polygon count and the number of lights, achieving 300+fps for models of about 100K triangles and one light (Figs. 5, 8), and still being real-time at 28-39fps for the 1M polygon dragon scene in Fig. 9 as well as the 388K polygon Buddha scene in Fig. 1 with three polygonal lights. In all cases, our algorithm is $2 \times 4 \times 4$ times faster than using the monomial recurrence of [Snyder 1996].

Applications: A number of applications and extensions enabled by our approach can be seen in the accompanying video and are briefly described here. Our method supports general polygonal area lights, and enables editing of the light source, rotations, or translations as shown in Fig. 8 (The scene can also be rotated/translated, corresponding to an inverse transformation of the lights). Moreover, multiple area lights can be combined, simply by adding their SH coefficients (Fig. 7). The time complexity for SH lighting calculation will grow linearly with the number of lights, but the cost of lighting-visibility computation in the PRT framework is not affected, since it need only be done once at the end. We can also include environment maps (Fig. 9), with their SH coefficients computed in the conventional way by numerical integration once per frame, and then simply added to the analytic area light SH coefficients.

Finally, while we do not support textured lights (since the derivation and formulae assume constant emission over the polygon), we can in some cases break the area light into smaller polygons, each with a different color. Since we only use low-order spherical harmonics, detailed textures are not needed. An example is shown in Fig. 10. This result also uses a glossy ground reflector and Buddha.

Fig. 6. Plot of timings for increasing numbers of spherical harmonic coefficients. Note that the rotation step is negligible at lower orders, and the overall method is therefore linear in the number of SH coefficients.

Fig. 7. Plot of timings for increasing the number of edges (vertices) in the polygonal area light, showing the method behaves linearly.

Table 2. Running time for our method in frames per second. We also include a comparison to using the monomial recurrence in [Snyder 1996].

<table>
<thead>
<tr>
<th>Scene</th>
<th>Tris.</th>
<th>Lights</th>
<th>Speed (Us)</th>
<th>Monomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant (Fig. 5)</td>
<td>163k</td>
<td>1</td>
<td>350.0 fps</td>
<td>91.0 fps</td>
</tr>
<tr>
<td>Buddha (Fig. 1)</td>
<td>388k</td>
<td>3</td>
<td>28.4 fps</td>
<td>8.2 fps</td>
</tr>
<tr>
<td>Dragon (Fig. 9)</td>
<td>1M</td>
<td>1</td>
<td>38.5 fps</td>
<td>15.3 fps</td>
</tr>
<tr>
<td>Face (Fig. 8)</td>
<td>98k</td>
<td>1</td>
<td>384.6 fps</td>
<td>150.0 fps</td>
</tr>
</tbody>
</table>

Because this is not the method of [Snyder 1996], since we only use their recurrence, and only one part differs from ours (computation of monomials rather than directly using the ZH recurrence). That method ran about 4x slower, at 91fps because of the time complexity in linearly combining monomials to obtain ZH. While the compiler can optimize the required additions to make them very fast, the extra $O(n)$ complexity for linear combinations of monomials still adds significant overhead. Table 2 shows additional runtime comparisons on a variety of scenes. The complexity depends on polygon count and the number of lights, achieving 300+fps for models of about 100K triangles and one light (Figs. 5, 8), and still being real-time at 28-39fps for the 1M polygon dragon scene in Fig. 9 as well as the 388K polygon Buddha scene in Fig. 1, with three polygonal lights. In all cases, our algorithm is $2 \times 4 \times 4$ times faster than using the monomial recurrence of [Snyder 1996].
leading to appearance that switches from reddish to bluish, in accordance with the light texture, from one end of the ground plane to the other. Similarly, the Buddha has bluish as well as yellowish highlights (combining red and green parts of the light). Glossy reflection is computed using spherical harmonic triple products (Clebsch-Gordan coefficients) to multiply the precomputed visibility and area lighting computed with our method at each vertex, before convolving with a Phong BRDF [Sloan et al. 2002].

7 CONCLUSIONS AND FUTURE WORK

Spherical Harmonic lighting is a popular technique today for real-time applications based on the precomputed radiance transfer method. We enable the use of near-field polygonal area lights, almost as easily as distant environment map lighting, by deriving analytic results for integrating spherical harmonics against planar polygons. The key technical contribution is a novel recurrence directly for zonal harmonic coefficients, which enables all coefficients up to a given degree to be computed iteratively in linear time, followed by fast spherical harmonic rotation. The algorithm is implemented in a simple GPU shader for area lighting within any spherical harmonic lighting method. In the future, we would like to generalize the results to non-uniform or directional emission.

So far, we have not exploited the smoothness of area lighting. This can be done in at least two ways. First, the SH coefficients are smooth spatially. One could compute the SH projections at a much lower-frequency coarse mesh (or even a sparse volume grid for the whole scene) and interpolate, while still representing visibility and performing PRT at vertices of the dense mesh. Moreover, at each vertex, the integral to compute SH coefficients from the area light is smooth, usually without the discontinuities typically present in graphics integrals. As such, an optimized sampling-based numerical integration or quadrature scheme may have low error, while being competitive in speed with our exact analytic approach.

In summary, we have shown that analytic lighting approaches can be combined with methods for complex light transport such as soft shadows, enabling wider availability of near-field effects in future real-time rendering applications.
ACKNOWLEDGMENTS

We thank the reviewers for many helpful suggestions, James Ha for discussion regarding the recurrence and Weilun Sun for implementation suggestions. The Happy Buddha and Dragon models are from the Stanford 3D Scanning Repository. The plant model was provided by Xfrog through TurboSquid. This work was supported in part by NSF grant IIS-1451830, Samsung, the Ronald L. Graham endowed Chair, and the UC San Diego Center for Visual Computing.

REFERENCES


A Appendix: Boundary Integral C

In this appendix, we derive the formula for $C_{l,e}$. For simplicity of notation, we drop the subscript and index $e$ for the edge, and omit the limits of the integral from $0$ to $\gamma_e$ in most places. We start with the formula for $C_{l,e}$ in equation\, and integrate by parts,

$$\int z P_l(z) \, dy = \left( P_l(z) \int z \, dy \right) |_0^{\gamma_e} - \left( \int z \, dy \right) \frac{dP_l(z)}{dy} \, dz = (a \sin \gamma - b \cos \gamma) P_l(z) |_0^{\gamma_e} + \int (a \sin \gamma + b \cos \gamma) P_l(z) \, dy. \quad (45)$$

In the last line, we use $z = a \cos \gamma + b \sin \gamma$, making it easy to find $\int z \, dy$ and $dz \, dy$. It is now convenient to write $C_{l,e} = C_{l,e}^1 + C_{l,e}^2$. The first term above is simple to evaluate directly, leaving us with

$$C_{l,e}^1 = (a_e \sin \gamma_e - b_e \cos \gamma_e) P_l(a_e \cos \gamma_e + b_e \sin \gamma_e) + b_e P_l(a_e). \quad (46)$$

The second term $C_{l,e}^2$ is found by integrating by parts again,

$$\int (\sin \gamma + b \cos \gamma) P_l(z) \, dy = \left( -a \sin \gamma + b \cos \gamma \right) \frac{dP_l(z)}{dy} |_0^{\gamma_e} - \left( \int P_l(z) \, dy \right) \int (\sin \gamma + b \cos \gamma) \, dy. \quad (47)$$

We now perform another integration by parts for the second term,

$$\int (\sin \gamma + b \cos \gamma) \int P_l(z) \, dy + \left( \int P_l(z) \, dy \right) \frac{d \int (\sin \gamma + b \cos \gamma) \, dy}{dz} = \int (\sin \gamma + b \cos \gamma)^2 P_l(z) \, dy = \int z^2 P_l(z) \, dy. \quad (48)$$

Note that we substitute for $z = a \cos \gamma + b \sin \gamma$ to obtain the last line. In the process, we also removed the evaluation from 0 to $\gamma_e$, since $a^2 + b^2$ is a constant.

We now use the identity in equation\, rearranging to obtain $z^2 P_l(z) = (l + 1) z P_l(z) - P_{l-1}(z) + P_l(z)$. Substituting inside the second integral, the above expression becomes,

$$\int z P_l(z) \, dy = \left( l + 1 \right) \int z P_l(z) \, dy - \int P_{l-1}(z) \, dy + \int P_l(z) \, dy \quad (49)$$

Plugging back into equation\ gives us

$$\left( l + 1 \right) \int z P_l(z) \, dy = (a \sin \gamma - b \cos \gamma) P_l(z) |_0^{\gamma_e} + \int z^2 P_l(z) \, dy + \int P_l(z) \, dy \quad (50)$$

We are now ready to obtain a formula for $C_{l,e}$, which corresponds to the integral on the left-hand side. The right-hand side integrals are simply $P_{l-1}$ and $B_{l-1}$, while equation\ can be used for the evaluation at 0 and $\gamma_e$. Putting this together, and re-introducing the index and subscript $e$ for the edge,

$$\left( l + 1 \right) C_{l,e} = (a_e \sin \gamma_e - b_e \cos \gamma_e) P_l(a_e \cos \gamma_e + b_e \sin \gamma_e) + b_e P_l(a_e) + \left( a_e^2 + b_e^2 - 1 \right) D_{l,e} + DB_{l-1,e}. \quad (51)$$

If we now consider the corresponding formula for $C_{l-1,e}$, we obtain

$$C_{l-1,e} = (a_e \sin \gamma_e - b_e \cos \gamma_e) P_{l-1}(a_e \cos \gamma_e + b_e \sin \gamma_e) + b_e P_{l-1}(a_e) + \left( a_e^2 + b_e^2 - 1 \right) D_{l-1,e} + (l - 1) B_{l-2,e}. \quad (52)$$

Upon dividing through by $l$, we obtain equation\.