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*;

Additional Key Words and Phrases: PRT, spherical harmonics, area lighting

ACM Reference Format:
Jingwen Wang and Ravi Ramamoorthi. 2018. Analytic Spherical Harmonic Coefficients for Polygonal Area Lights. *ACM Trans. Graph.* 37, 4, Article 54 (August 2018), 11 pages. https://doi.org/10.1145/3197517.3201291

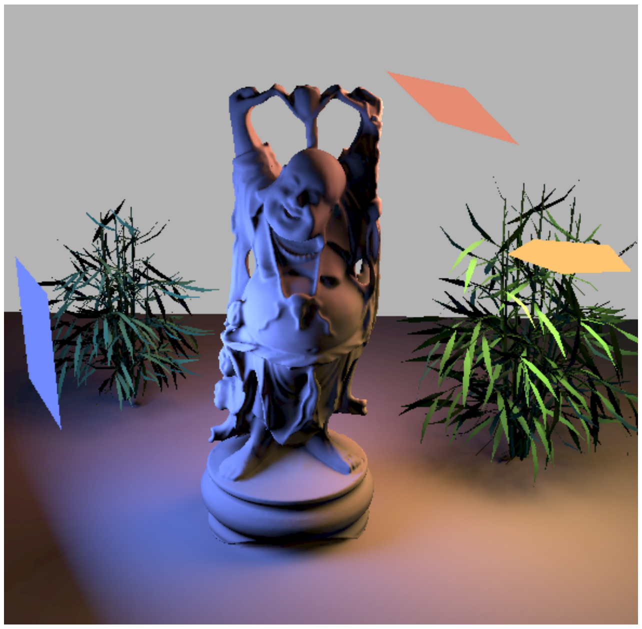


Fig. 1. Images rendered in a PRT system with multiple polygonal area lights, with analytic SH lighting coefficients computed in real-time with our method. This scene contains 388K triangles and is rendered at 28.4 frames per second on an NVIDIA GeForce GTX 1080 Ti GPU.

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.[1] 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 precomputation 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. As in most previous work, we assume the light source emission is

[1]PRT was at one time part of the DirectX API; modern games use a variety of real-time techniques, including SH irradiance, volumetric transport, and variants of PRT.

Authors' addresses: Jingwen Wang, University of California, San Diego, jiw524@eng.ucsd.edu; Ravi Ramamoorthi, University of California, San Diego, ravir@cs.ucsd.edu.

0730-0301/2018/8-ART54 $15.00
https://doi.org/10.1145/3197517.3201291

uniform across its surface (and do not explicitly support textured 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[2] 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 z^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.

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 build up 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 $O(n^3)$. However, there is considerable sparsity, and the rotation is very fast in practice [Nowrouzezahrai et al. 2012], with minimal overhead (Fig. 6). Hence, the algorithm's effective cost is $O(n^2)$, linear in the number of SH coefficients.

[2]We use the terms order and degree of the polynomial interchangeably.
[3]While the addition/linear combination of polynomial coefficients to compute ZH coefficients is very fast in practice, it still adds enough overhead to be 2×-4× slower than our method, as discussed in Sec. 6.
[4]This analysis is for a single ZH lobe direction. As discussed next, we actually need $2n+1$ lobe directions, so our approach has cost $O(n^2)$, while applying the recurrence in [Snyder 1996] would take $O(n^3)$ time.

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 results 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.

Table 1. Table of notation for key symbols.

Y_{lm}	The SH basis function of degree l, index m
L_{lm}	The SH coefficient for the polygonal light
P_l	Legendre polynomial of degree l
Q	The polygonal light (projected to unit sphere)
$\boldsymbol{\omega}_e$	Projection of vertex e onto unit sphere
γ_e	Angle between polygonal vertices e and $e+1$
$\boldsymbol{\omega}$	Central axis of zonal harmonic lobe
$\alpha^m_{l,d}$	Weight of d-th ZH lobe used to reconstruct Y_{lm}
S_l	The surface integral of the l-th Legendre polynomial $\int_Q P_l(\boldsymbol{\omega}\cdot\boldsymbol{\omega}_i)d\omega_i$
$B_{l,e}$	The edge integral $\int_0^{\gamma_e} P_l(z_e)d\gamma$
$C_{l,e}$	The intermediate edge integral $\int_0^{\gamma_e} z_e P_l(z_e)d\gamma$
$D_{l,e}$	The intermediate edge integral $\int_0^{\gamma_e} P_l'(z_e)d\gamma$

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 into smaller polygons, each of which has a uniform color (Fig. 10). In future, it may be possible to leverage [Chen and Arvo 2001] for linearly-varying luminaires and angularly-varying lighting.

Finally, the concurrent work of [Belcour et al. 2018] computes a *single* integral of a particular SH expansion against a polygon. We differ conceptually in simultaneously computing integrals of all SH coefficients, enabling PRT with any SH expansion. In practice, the methods are quite similar, and [Belcour et al. 2018] also uses a zonal harmonic decomposition and combines closed-form monomial integrals [Arvo 1995; Snyder 1996]. We differ in deriving a novel zonal harmonic recurrence, rather than combining monomials, and 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,

$$L^r(\boldsymbol{x},\boldsymbol{\omega}_r)=\int_\Omega L^i(\boldsymbol{x},\boldsymbol{\omega}_i)V(\boldsymbol{x},\boldsymbol{\omega}_i)\rho(\boldsymbol{\omega}_i,\boldsymbol{\omega}_r,\boldsymbol{n})\max(\boldsymbol{\omega}_i\cdot\boldsymbol{n},0)\,d\boldsymbol{\omega}_i, \tag{1}$$

where $\boldsymbol{x}$ is the spatial coordinate of the current vertex or pixel, $(\boldsymbol{\omega}_i,\boldsymbol{\omega}_r)$ are the global incoming and outgoing directions, and $\boldsymbol{n}$ is the surface normal. L^r is the reflected radiance or image intensity, while L^i is the incident radiance.[5] V is the visibility and ρ is the BRDF (which includes the normal as an argument, since it is expressed in terms of incoming and outgoing directions in global coordinates).

For simplicity of notation, we combine the visibility and BRDF terms into a transport function $T(\boldsymbol{x},\boldsymbol{\omega}_i)$, which is precomputed for each spatial location $\boldsymbol{x}$ and incident angle $\boldsymbol{\omega}_i$. We have dropped the dependence on $\boldsymbol{\omega}_r$ for simplicity, but that can also be expressed in spherical harmonics [Sloan et al. 2002], and glossy reflections are demonstrated in Fig. 10. We may now write,

$$L^r(\boldsymbol{x})=\int_\Omega L^i(\boldsymbol{x},\boldsymbol{\omega}_i)T(\boldsymbol{x},\boldsymbol{\omega}_i)\,d\boldsymbol{\omega}_i. \tag{2}$$

To proceed further, we project the lighting and transport function into spherical harmonics $Y_{lm}(\boldsymbol{\omega}_i)$ with $l\geq 0$ the order, and $-l\leq m\leq l$. We will discuss the spherical harmonic basis function forms [MacRobert 1948] briefly at the end of this section. We use spherical harmonics up to degree n with $(n+1)^2$ terms; we demonstrate results for fairly high order spherical harmonics with $n=8$ or $n=14$, and our results are general for even higher orders if needed,

$$\begin{aligned} L^i(\boldsymbol{x},\boldsymbol{\omega}_i) &= \sum_{l=0}^{n}\sum_{m=-l}^{l} L_{lm}(\boldsymbol{x})\,Y_{lm}(\boldsymbol{\omega}_i) \\ T(\boldsymbol{x},\boldsymbol{\omega}_i) &= \sum_{l=0}^{n}\sum_{m=-l}^{l} T_{lm}(\boldsymbol{x})\,Y_{lm}(\boldsymbol{\omega}_i). \end{aligned} \tag{3}$$

Given these coefficients, by orthonormality of the basis functions,

$$L^r(\boldsymbol{x})=\sum_{l=0}^{n}\sum_{m=-l}^{l} L_{lm}(\boldsymbol{x})\,T_{lm}(\boldsymbol{x})=\boldsymbol{L}(\boldsymbol{x})\cdot\boldsymbol{T}(\boldsymbol{x}), \tag{4}$$

where $\boldsymbol{L}$ and $\boldsymbol{T}$ are vector representations of L_{lm} and T_{lm}, and the integral can be reduced to a simple dot product.

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

$$L_{lm}=\int_{Q_{\boldsymbol{x}}} Y_{lm}(\boldsymbol{\omega}_i)\,d\boldsymbol{\omega}_i, \tag{5}$$

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 $\boldsymbol{x}$ in L_{lm}, since it is easy to translate the polygon so $\boldsymbol{x}$ is at the origin; we denote this using $Q_{\boldsymbol{x}}$, 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 $\boldsymbol{\omega}$,

$$L_l(\boldsymbol{\omega})=\int_{Q_{\boldsymbol{x}}} Y_{l0}(\boldsymbol{\omega}\cdot\boldsymbol{\omega}_i)\,d\boldsymbol{\omega}_i. \tag{6}$$

[5]We use superscripts for L^r and L^i to avoid confusion with subscripts such as L_{lm} for spherical harmonic coefficients.

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 (θ, ϕ) with Cartesian components, $(x, y, z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$. The real spherical harmonics are,

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

where P_m^l are the associated Legendre polynomials and the trigonometric function $f(|m|\phi)$ 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,

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

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 ϕ, depending only on z in a Cartesian coordinate system. To consider ZH lobes (as a function of incident direction $\boldsymbol{\omega}_i = (\theta, \phi)$) about an arbitrary axis $\boldsymbol{\omega}$, we can simply use a dot product with that axis, defining the ZH as $Y_{l0}(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_i)$ 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[6] that for any m and any order l (for consistency with equation 6 we use $\boldsymbol{\omega}_i = (\theta, \phi)$),

$$Y_{lm}(\boldsymbol{\omega}_i) = \sum_d \alpha_{l,d}^m Y_{l0}(\boldsymbol{\omega}_{l,d} \cdot \boldsymbol{\omega}_i). \tag{9}$$

That is, an SH of degree l can always be written as a sum of rotated ZH basis functions. While 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 $\boldsymbol{\omega}_{l,d}$ across bands. As such, there are only $2n + 1$ unique lobe directions $\boldsymbol{\omega}$ across all bands, and the weights $\alpha_{l,d}$ 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. 5 (below, $P_l'(z)$ stands for the derivative $\frac{d}{dz}P_l(z)$),

$$\frac{z^2-1}{l}P_l'(z) = zP_l(z) - P_{l-1}(z) \tag{10}$$

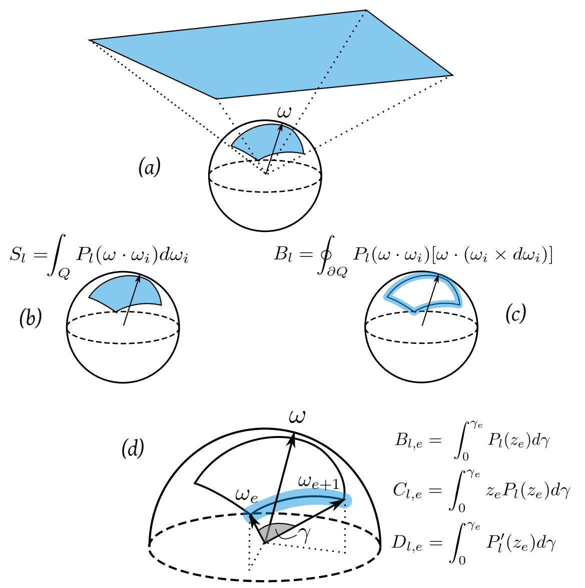
$$lP_l(z) = (2l-1)zP_{l-1}(z) - (l-1)P_{l-2}(z) \tag{11}$$

$$(2l+1)P_l(z) = P_{l+1}'(z) - P_{l-1}'(z) \tag{12}$$

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_{lm} in equation 5. The key technical contribution is the novel zonal harmonic recurrence, which is derived later in Sec. 5. While the mathematics and derivation is somewhat involved, the actual implementation is simple, as discussed at the end of Sec. 4.1 and shown in the pseudocode in Algorithm 1. 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.

[6] Versions of equation 9 appear earlier in other fields, going back at least to [Stern 1965].



ω
(a)
$S_l = \int_Q P_l(\omega \cdot \omega_i)d\omega_i$
$B_l = \oint_{\partial Q} P_l(\omega \cdot \omega_i)[\omega \cdot (\omega_i \times d\omega_i)]$
(b)
(c)
ω
(d)
ω_e
ω_{e+1}
γ
$B_{l,e} = \int_0^{\gamma_e} P_l(z_e)d\gamma$
$C_{l,e} = \int_0^{\gamma_e} z_e P_l(z_e)d\gamma$
$D_{l,e} = \int_0^{\gamma_e} P_l'(z_e)d\gamma$

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. 1.

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

$$S_l = \int_Q P_l(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_i)d\omega_i \qquad L_l = \sqrt{\frac{2l+1}{4\pi}}S_l. \tag{13}$$

S_l simply integrates the projection of the area light on the unit sphere against Legendre polynomials P_l of the dot product $\boldsymbol{\omega} \cdot \boldsymbol{\omega}_i$. 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 $\boldsymbol{\omega}_{l,d}$ (there may be $2n + 1$ such directions). For simplicity of notation, we just use $\boldsymbol{\omega}$, as in equation 6, and omit the explicit dependence of S_l on $\boldsymbol{\omega}$.

We will be deriving a recurrence for S_l in terms of boundary integrals over the contours of the polygon's projection, denoted by ∂Q (Fig. 2(c)). Specifically, we keep track of

$$B_l = \oint_{\partial Q} P_l(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_i)(\tilde{x}_i d\tilde{y}_i - \tilde{y}_i d\tilde{x}_i) \quad = \quad \sum_{e=1}^{M} c_e B_{l,e}. \qquad (14)$$

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 $\boldsymbol{\omega}_i$, with respect to $\boldsymbol{\omega}$. The tildes on $\tilde{x}$ and $\tilde{y}$ denote these are with respect to a coordinate frame where $\boldsymbol{\omega}$ is the $\boldsymbol{Z}$ axis.[7] One can also express the differential measure in terms of a cross product between $\boldsymbol{\omega}_i$ and $d\boldsymbol{\omega}_i$, taking the component along $\boldsymbol{\omega}$,

$$P_l(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_i)(\tilde{x}_i d\tilde{y}_i - \tilde{y}_i d\tilde{x}_i) = P_l(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_i)\left[\boldsymbol{\omega} \cdot (\boldsymbol{\omega}_i \times d\boldsymbol{\omega}_i)\right]. \qquad (15)$$

In fact, the integrand above for $B_{l,e}$ has a simple form in the appropriate parameterization. Let us define $\boldsymbol{\omega}_e$ as the projection on the unit sphere of vertex e of the polygon, so that edge e connects $\boldsymbol{\omega}_e$ and $\boldsymbol{\omega}_{e+1}$, as shown in Fig. 2(d). 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 $\boldsymbol{\omega}_e = \boldsymbol{Z}$, a simple parameterization would be $\boldsymbol{\omega}_i(\gamma) = \boldsymbol{Z}\cos\gamma + \boldsymbol{X}\sin\gamma$ where γ ranges from 0 to the angle between $\boldsymbol{\omega}_e$ and $\boldsymbol{\omega}_{e+1}$. In the general case, the vector $\boldsymbol{Z}$ is simply replaced by $\boldsymbol{\omega}_e$, while the vector $\boldsymbol{X}$ is replaced by a new vector $\boldsymbol{\lambda}_e$, given through cross products by

$$\boldsymbol{\lambda}_e = \frac{\boldsymbol{\omega}_e \times \boldsymbol{\omega}_{e+1}}{\| \boldsymbol{\omega}_e \times \boldsymbol{\omega}_{e+1} \|} \times \boldsymbol{\omega}_e \quad = \quad \frac{\boldsymbol{\omega}_{e+1} - \boldsymbol{\omega}_e(\boldsymbol{\omega}_e \cdot \boldsymbol{\omega}_{e+1})}{\| \boldsymbol{\omega}_e \times \boldsymbol{\omega}_{e+1} \|}. \qquad (16)$$

We are simply creating a vector of unit length orthogonal to $\boldsymbol{\omega}_e$ so that $\boldsymbol{\omega}_{e+1}$, or any intermediate point along the edge (arc), can be expressed as a linear combination of $\boldsymbol{\omega}_e$ and $\boldsymbol{\lambda}_e$,

$$\boldsymbol{\omega}_i(\gamma) = \boldsymbol{\omega}_e \cos\gamma + \boldsymbol{\lambda}_e \sin\gamma, \qquad (17)$$

where γ ranges from 0 for $\boldsymbol{\omega}_e$ to $\cos^{-1}(\boldsymbol{\omega}_e \cdot \boldsymbol{\omega}_{e+1})$ for $\boldsymbol{\omega}_{e+1}$. This parameterization for the integral is visualized in Fig. 2(d). $\boldsymbol{\omega}_e$ and $\boldsymbol{\lambda}_e$ are computed once, and reused for all lobes $\boldsymbol{\omega}$.

We can now express $B_{l,e}$ from equation 14, using the integrand in equation 15, and the expression for $\boldsymbol{\omega}_i$ above,

$$P_l(\boldsymbol{\omega} \cdot \boldsymbol{\omega}_i) = P_l\left((\boldsymbol{\omega} \cdot \boldsymbol{\omega}_e)\cos\gamma + (\boldsymbol{\omega} \cdot \boldsymbol{\lambda}_e)\sin\gamma\right). \qquad (18)$$

For the differential measure , $d\boldsymbol{\omega}_i = -\boldsymbol{\omega}_e \sin\gamma\, d\gamma + \boldsymbol{\lambda}_e cos\gamma\, d\gamma$,

$$\boldsymbol{\omega} \cdot (\boldsymbol{\omega}_i \times d\boldsymbol{\omega}_i) = \boldsymbol{\omega} \cdot (\boldsymbol{\omega}_e \times \boldsymbol{\lambda}_e)\, d\gamma = \boldsymbol{\omega} \cdot \boldsymbol{\mu}_e\, d\gamma, \qquad (19)$$

where we define $\boldsymbol{\mu}_e = \boldsymbol{\omega}_e \times \boldsymbol{\lambda}_e = \frac{\boldsymbol{\omega}_e \times \boldsymbol{\omega}_{e+1}}{\|\boldsymbol{\omega}_e \times \boldsymbol{\omega}_{e+1}\|}$ to create a coordinate frame $(\boldsymbol{\omega}_e, \boldsymbol{\lambda}_e, \boldsymbol{\mu}_e)$. It is convenient to define a number of scalar quantities, corresponding to the dot-products above, with (a_e, b_e, c_e) the coordinates of $\boldsymbol{\omega}$ with respect to this coordinaate frame,

$$\begin{aligned} a_e &= \boldsymbol{\omega} \cdot \boldsymbol{\omega}_e \\ b_e &= \boldsymbol{\omega} \cdot \boldsymbol{\lambda}_e \\ c_e &= \boldsymbol{\omega} \cdot \boldsymbol{\mu}_e \\ z_e &= a_e \cos\gamma + b_e \sin\gamma \\ \gamma_e &= \cos^{-1}(\boldsymbol{\omega}_e \cdot \boldsymbol{\omega}_{e+1}). \end{aligned} \qquad (20)$$

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

$$B_{l,e} = \int_0^{\gamma_e} P_l(z_e)\, d\gamma. \qquad (21)$$

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 $B_{l,e}$,

$$\begin{aligned} C_{l,e} &= \int_0^{\gamma_e} z_e P_l(z_e)\, d\gamma & (22) \\ D_{l,e} &= \int_0^{\gamma_e} P_l'(z_e)\, d\gamma. & (23) \end{aligned}$$

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 [Arvo 1995],

$$S_0 = \left(\sum_{e=1}^{M} \frac{\cos^{-1}\left((\boldsymbol{\omega}_{e-1} \times \boldsymbol{\omega}_e) \cdot (\boldsymbol{\omega}_e \times \boldsymbol{\omega}_{e+1})\right)}{\| \boldsymbol{\omega}_{e-1} \times \boldsymbol{\omega}_e \| \, \|\boldsymbol{\omega}_e \times \boldsymbol{\omega}_{e+1} \|} \right) - (M-2)\pi \qquad (24)$$

Note the standard convention that vertex indexing is cyclic, so $\boldsymbol{\omega}_{M+1} = \boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_0 = \boldsymbol{\omega}_M$.

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,

$$S_1 = \frac{1}{2} B_0 = \frac{1}{2} \sum_{e=1}^{M} c_e B_{0,e} = \frac{1}{2} \sum_{e=1}^{M} c_e \gamma_e. \qquad (25)$$

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

$$B_{0,e} = \gamma_e. \qquad (26)$$

Similarly, noting that $P_1(z_e) = z_e = a_e \cos\gamma + b_e \sin\gamma$,

$$B_{1,e} = a_e \sin\gamma_e - b_e \cos\gamma_e + b_e. \qquad (27)$$

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 $B_{l,e}$,

$$D_{0,e} = 0 \quad D_{1,e} = \gamma_e \quad D_{2,e} = 3B_{1,e} = 3\left(a_e \sin\gamma_e - b_e \cos\gamma_e + b_e\right). \qquad (28)$$

[7] This form of differential measure is common when applying the Stokes theorem to area light integration, and follows from the test function in Sec. 5 (equation 37).

Recurrence Relation. In Sec. 5, 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_n. Specifically, we derive that

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

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,

$$\begin{aligned} C_{l-1,e} = \frac{1}{l}((a_e \sin\gamma_e - b_e \cos\gamma_e)\, P_{l-1}(a_e \cos\gamma_e + b_e \sin\gamma_e) \\ + b_e P_{l-1}(a_e) + (a_e^2 + b_e^2 - 1)\, D_{l-1,e} + (l-1) B_{l-2,e}) \end{aligned} \tag{31}$$

Finally, there is a simple recurrence for D,

$$D_{l,e} = (2l-1) B_{l-1,e} + D_{l-2,e}. \tag{32}$$

After we compute the base cases, we iterate $2 \le l \le n$, 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 $\boldsymbol{x}$ and a polygon with vertices $\boldsymbol{v}_e$ of M sides. The lobe weights $\alpha_{l,d}^m$ for ZH rotation, along with lobe directions $\boldsymbol{\omega}_{l,d}$ are also inputs. We compute SH coefficients up to order n.

First (Line 2), we translate the polygon vertices by $-\boldsymbol{x}$ and project onto the unit sphere, obtaining the vertices $\boldsymbol{\omega}_e$ on the sphere. The quantities $\boldsymbol{\lambda}_e$, $\boldsymbol{\mu}_e = \boldsymbol{\omega}_e \times \boldsymbol{\lambda}_e$ and γ_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 $\boldsymbol{\omega}$ (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_1, we must sum over all edges (Lines 10,13).

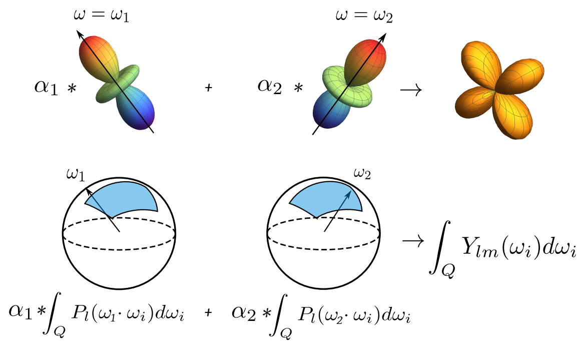
Then, we iterate $2 \le l \le n$ (Line 18), and update the boundary integrals for each edge (Lines 20-25). Specifically we compute $C_{l-1,e}$ (Line 21), and apply the recurrence formulae for $B_{l,e}$ (Line 22) and $D_{l,e}$ (Line 24). The boundary integral B_l is updated by summing over edge integrals $B_{l,e}$ (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 $L_l(\boldsymbol{\omega})$ 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 $L_l(\boldsymbol{\omega})$, as discussed in Sec. 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

Algorithm 1 Pseudocode of algorithm for SH coefficients.

```
 1: function ComputeCoefficients($\boldsymbol{x}$, $\boldsymbol{v}$[], M, n, $\alpha$[], $\boldsymbol{\omega}$[])
 2:     forall $e$: $\boldsymbol{v}_e = \boldsymbol{v}_e - \boldsymbol{x}$ ; $\boldsymbol{\omega}_e = \frac{\boldsymbol{v}_e}{\|\boldsymbol{v}_e\|}$     ▷ Project to Sphere
 3:     for $e = 1$ to $M$ do     ▷ Pre-Calculate per edge
 4:         $\boldsymbol{\lambda}_e = \frac{\boldsymbol{\omega}_e \times \boldsymbol{\omega}_{e+1}}{\|\boldsymbol{\omega}_e \times \boldsymbol{\omega}_{e+1}\|} \times \boldsymbol{\omega}_e$     ▷ Eq. 16
 5:         $\boldsymbol{\mu}_e = \boldsymbol{\omega}_e \times \boldsymbol{\lambda}_e$     ▷ Frame $(\boldsymbol{\omega}_e, \boldsymbol{\lambda}_e, \boldsymbol{\mu}_e)$
 6:         $\gamma_e = \cos^{-1}(\boldsymbol{\omega}_e \cdot \boldsymbol{\omega}_{e+1})$     ▷ Angle of edge $e$ Eq. 20
 7:     end for
 8:     $S_0$ = SolidAngle($\boldsymbol{\omega}_e$[], $M$)     ▷ Eq. 24
 9:     for all $\boldsymbol{\omega}$ in $\boldsymbol{\omega}$[] do     ▷ Up to 2n+1 for all bands
10:         $S_1 = 0$     ▷ Initialize $S_1$
11:         for $e = 1$ to $M$ do     ▷ Base Cases
12:             $a_e = \boldsymbol{\omega} \cdot \boldsymbol{\omega}_e$ ; $b_e = \boldsymbol{\omega} \cdot \boldsymbol{\lambda}_e$ ; $c_e = \boldsymbol{\omega} \cdot \boldsymbol{\mu}_e$     ▷ Eq. 20
13:             $S_1 = S_1 + \frac{1}{2} c_e \gamma_e$     ▷ Eq. 25
14:             $B_{0,e} = \gamma_e$     ▷ Eq. 26
15:             $B_{1,e} = a_e \sin\gamma_e - b_e \cos\gamma_e + b_e$.     ▷ Eq. 27
16:             $D_{0,e} = 0; D_{1,e} = \gamma_e$ ; $D_{2,e} = 3B_{1,e}$     ▷ Eq. 28
17:         end for
18:         for $l = 2$ to $n$ do     ▷ Each degree $2 \le l \le n$
19:             $B_l = 0$     ▷ 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. 31
22:                 $B_{l,e} = \frac{2l-1}{l} C_{l-1,e} - (l-1) B_{l-2,e}$     ▷ 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:             $S_l = \frac{2l-1}{l(l+1)} B_{l-1} + \frac{(l-2)(l-1)}{l(l+1)} S_{l-2}$     ▷ Eq. 29
27:             $L_l(\boldsymbol{\omega}) = \sqrt{\frac{2l+1}{4\pi}} S_l$     ▷ Eq. 13
28:         end for
29:     end for
30:     for each SH basis function $(l, m)$ do     ▷ ZH rotation
31:         $L_{lm} = 0$     ▷ Initialize $L_{lm}$
32:         for each $d \in d(l)$ do     ▷ Sparse set of lobes $\boldsymbol{\omega}_{l,d}$
33:             $L_{lm} = L_{lm} + \alpha_{l,d}^m L_l(\boldsymbol{\omega}_{l,d})$     ▷ Eq. 33
34:         end for
35:     end for
36: end function
```



$\omega = \omega_1$
$\omega = \omega_2$
$\alpha_1 *$
$+$
$\alpha_2 *$
$\rightarrow$
ω_1
ω_2
$\rightarrow \int_Q Y_{lm}(\omega_i) d\omega_i$
$\alpha_1 * \int_Q P_l(\omega_1 \cdot \omega_i) d\omega_i$
$+$
$\alpha_2 * \int_Q P_l(\omega_2 \cdot \omega_i) d\omega_i$

Fig. 3. Sparse zonal harmonic rotation. The SH and equivalently integral for SH coefficients can be viewed as a sparse sum of integrals for ZH coefficients. Thus, we may sum a small number of ZH coefficients (here, illustrated as only two lobes) for each SH coefficient.

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 13). 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 shown in Fig. 3, we simply compute,

$$L_{lm} = \sum_{d \in d(l)} \alpha^m_{l,d} L_l \left(\boldsymbol{\omega}_{l,d}\right). \tag{33}$$

While d can take $2l + 1$ values in the general case (and there are a total of $2n + 1$ lobe directions $\boldsymbol{\omega}$ 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 $\alpha^m_{l,d}$ 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 $\boldsymbol{\omega}$ to compute $L_l(\boldsymbol{\omega})$, 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 $\alpha^m_{l,d}$ 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 RECURRENCE

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 polynomial identity in equation 11. Integrating over Q (i.e., with each term now as an integrand), we obtain

$$\int_Q P_l(z)\, d\omega_i = \left(\frac{2l-1}{l} \int_Q zP_{l-1}(z)\, d\omega_i\right) - \left(\frac{l-1}{l} \int_Q P_{l-2}(z)\, d\omega_i\right). \tag{34}$$

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 $\boldsymbol{\omega} \cdot \boldsymbol{\omega}_i$. For our derivation, we can assume, without loss of generality, that we have rotated the coordinate system so $\boldsymbol{\omega}$ is aligned with the $\boldsymbol{Z}$ axis, while explicitly preserving $\boldsymbol{\omega}$ for the actual algorithm in Sec. 4. We can now view equation 34 as a relation for the surface integral,

$$S_l = \left(\frac{2l-1}{l} \int_Q zP_{l-1}(z)\, d\omega_i\right) - \frac{(l-1)}{l} S_{l-2}. \tag{35}$$

It is clear that the expression on the right is a previously-computed value. Now we show how to compute $\int_Q zP_{l-1}(z)\, d\omega_i$. 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 $\boldsymbol{V}$,

$$\int_Q \boldsymbol{N} \cdot (\nabla \times \boldsymbol{V})\, d\omega_i = \oint_{\partial Q} \boldsymbol{V} \cdot \boldsymbol{dr}, \tag{36}$$

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

$$\boldsymbol{V} = \begin{pmatrix} -yP_{l-1}(z) \\ xP_{l-1}(z) \\ 0 \end{pmatrix} \tag{37}$$

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

$$\begin{aligned} \boldsymbol{N} \cdot (\nabla \times \boldsymbol{V}) &= -x^2 \frac{d}{dz} P_{l-1}(z) - y^2 \frac{d}{dz} P_{l-1}(z) + 2zP_{l-1}(z) \\ &= (z^2 - 1)P'_{l-1}(z) + 2zP_{l-1}(z), \end{aligned} \tag{38}$$

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

$$\boldsymbol{N} \cdot (\nabla \times \boldsymbol{V}) = (l+1)zP_{l-1}(z) - (l-1)P_{l-2}(z). \tag{39}$$

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,

$$(l+1) \int_Q zP_{l-1} d\omega_i - (l-1) \int_Q P_{l-2} d\omega_i = \oint_{\partial Q} P_{l-1}(xdy - ydx). \tag{40}$$

We can now to evaluate the surface integral $\int_Q zP_{l-1}d\omega_i$ in equation 35. Including the normalization $\frac{2l-1}{l}$ in equation 35 and rearranging terms in equation 40,

$$\frac{2l-1}{l}\int_Q zP_{l-1}d\omega_i = \tag{41}$$
$$\frac{2l-1}{l(l+1)}\oint_{\partial Q} P_{l-1}(xdy-ydx) + \frac{(2l-1)(l-1)}{l(l+1)}\int_Q P_{l-2}d\omega_i.$$

Substituting this result back into equation 35 gives

$$S_l = \frac{2l-1}{l(l+1)}B_{l-1} + \frac{(2l-1)(l-1)}{l(l+1)}S_{l-2} - \frac{(l-1)}{l}S_{l-2}$$
$$S_l = \frac{2l-1}{l(l+1)}B_{l-1} + \frac{(l-2)(l-1)}{l(l+1)}S_{l-2}, \tag{42}$$

which is exactly the main recurrence in equation 29.

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 $B_{l,e} = \int_0^{\gamma_e} P_l(z_e)\,d\gamma$, where as usual we have that $z_e = a_e\cos\gamma + b_e\sin\gamma$. We first rewrite $B_{l,e}$ using equation 11,

$$\int_0^{\gamma_e} P_l(z_e)\,d\gamma = \frac{21-1}{l}\int_0^{\gamma_e} z_eP_{l-1}(z_e)\,d\gamma - \frac{l-1}{l}\int_0^{\gamma_e} P_{l-2}(z_e)\,d\gamma. \tag{43}$$

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 normalization factors, this is exactly the recurrence relation for $B_{l,e}$ in equation 30. It is clear that the second term on the right has been previously computed. The first term on the right corresponds to $C_{l-1,e} = \int z_eP_{l-1}(z_e)\,d\gamma$. 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 $(2l-1)P_{l-1}(z) = P'_l(z) - P'_{l-2}(z)$. Considering the corresponding integrals and rearranging,

$$\int_0^{\gamma_e} P'_l(z_e)\,d\gamma = (2l-1)\int_0^{\gamma_e} P_{l-1}(z_e)\,d\gamma + \int_0^{\gamma_e} P'_{l-2}(z_e)\,d\gamma. \tag{44}$$

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 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. 1,9).

Analysis of Accuracy. The spherical harmonic formula in Sec. 4 is exact. To verify the correctness of the formula and its derivation, we compare with numerical integration with 146k samples per vertex using stratified Monte Carlo sampling computed offline in Fig. 4. We have also conducted a number of other similar tests with different scenes, lights and SH orders. Figure 4 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 red numerical integration curve is almost indistinguishable (and therefore invisible) from the blue analytic computation.

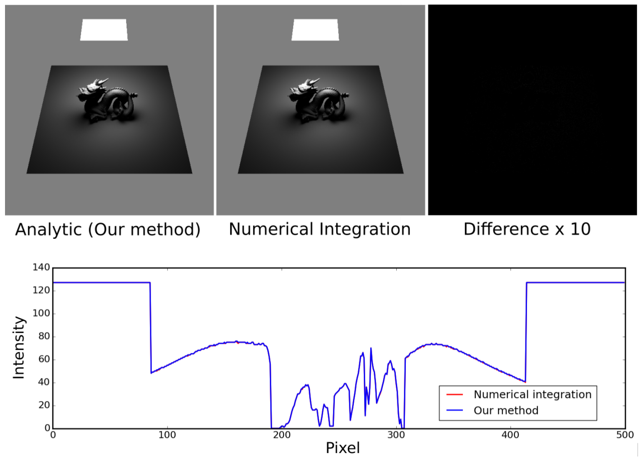


Fig. 4. We show image comparisons using our analytic SH computation, numerical integration (slow), and the pixel-wise difference. Note that the differences between the two results are at most one grey level, and displayed here at 10 times the intensity. For further verification, we also plot the intensity along the central image scanline, where the analytic and numerical curves agree almost perfectly.

Our method allows us to compute the spherical harmonics up to any order. In Fig. 5 we show the effects of increasing the order, obtaining sharper, more accurate shadows. We also show one example with glossy reflections later in Fig. 10.

Timings. We now discuss the time required to compute the SH lighting coefficients for increasing SH orders, shown in Fig. 6. For our method, we also break out the time for iterative zonal harmonic computation (Sec. 4.1) and the sparse zonal harmonic rotation (Sec. 4.2). We profiled the GPU time for a single draw call using NVIDIA's Nsight, performing our lighting method on a scene with 400k vertices. To obtain timings for the different steps of our method and accurately observe performance impact of increasing orders, we disabled compiler optimizations for these measurements.[8] Because of this, as well as the slight additional performance overhead from using the profiler, we show normalized time in Fig. 6.

It can be seen that the iterative ZH computation, which is linear in the total number of SH coefficients (Sec. 4.1), is the major time complexity of the method. Sparse zonal harmonic rotation adds only a small overhead, which is negligible at lower orders (not noticeable at all until order $n = 12$ or 169 SH terms). It is a maximum of 15% of

[8] Optimized compiles also give roughly (but not exactly) linear performance with increasing coefficients. Differences are due to e.g. load balancing at various GPU stages.

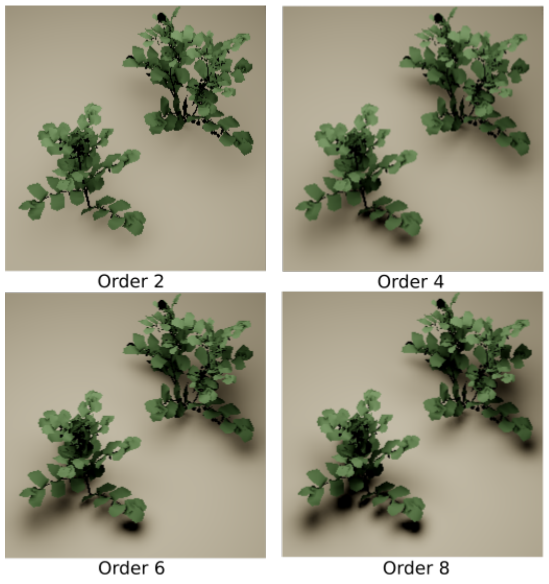


Fig. 5. Area lighting with SH orders 2, 4, 6, and 8. Note increasing accuracy in terms of sharpness of shadows with higher orders.

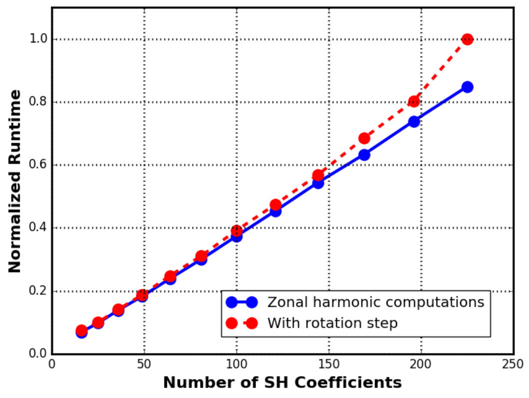


Fig. 6. Plot of timings for increasing numbers of spherical harmonic coefficients. Note that the rotation step is negligible at lower orders, and the overall method is therefore linear in the number of SH coefficients.

total runtime at order $n = 14$ (225 SH terms). Thus, in practice, the time complexity of our method is $O(n^2)$ or linear in the number of SH coefficients, as desired. Finally, Fig. 7 plots normalized runtime vs. the number of light polygon edges M. As expected the 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.

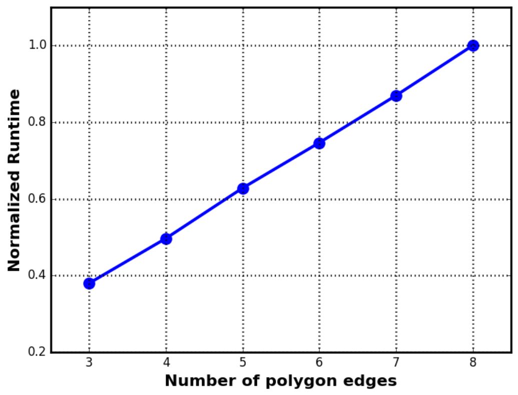


Fig. 7. Plot of timings for increasing the number of edges (vertices) in the polygonal area light, showing the method behaves linearly.

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

Scene	Tris.	Lights	**Speed (Us)**	Monomial
Plant (Fig. 5)	163k	1	**350.0 fps**	91.0 fps
Buddha (Fig. 1)	388k	3	**28.4 fps**	8.2 fps
Dragon (Fig. 9)	1M	1	**38.5 fps**	15.3 fps
Face (Fig. 8)	98k	1	**384.6 fps**	150.0 fps

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 4× 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×−4× 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. 1). 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.

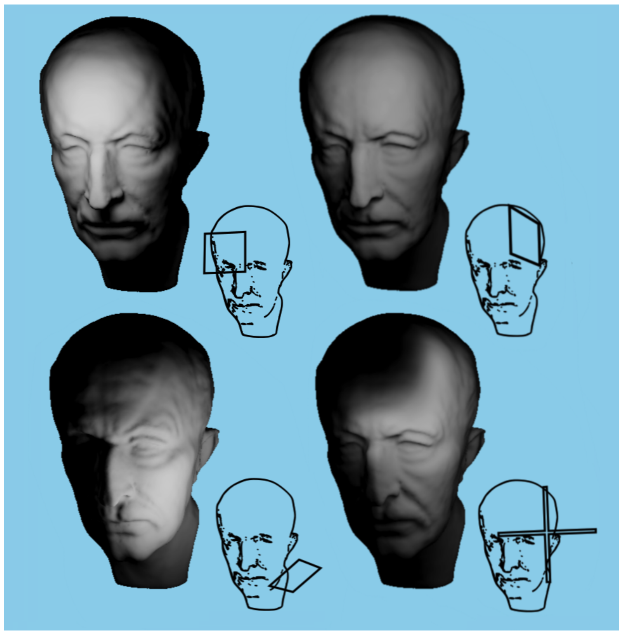


Fig. 8. Our method supports translation, in/out of plane rotations, and editing of the shape of the light source. Insets show area light positions. In this example, we show SH order $n = 8$.

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, 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.

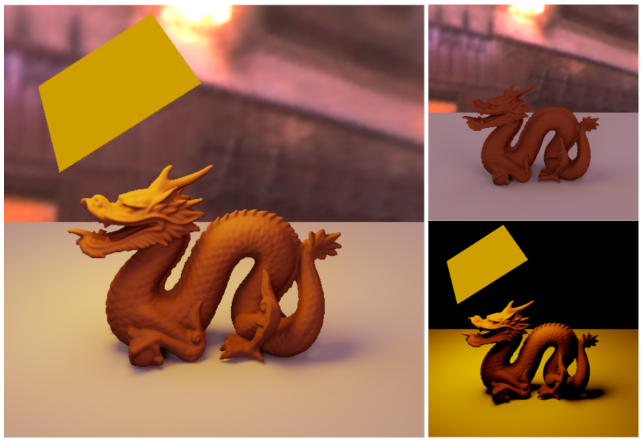


Fig. 9. Our method can support traditional relighting with environment maps in addition to area lights. Both the area light and environment can be rotated/changed interactively at real-time framerates.

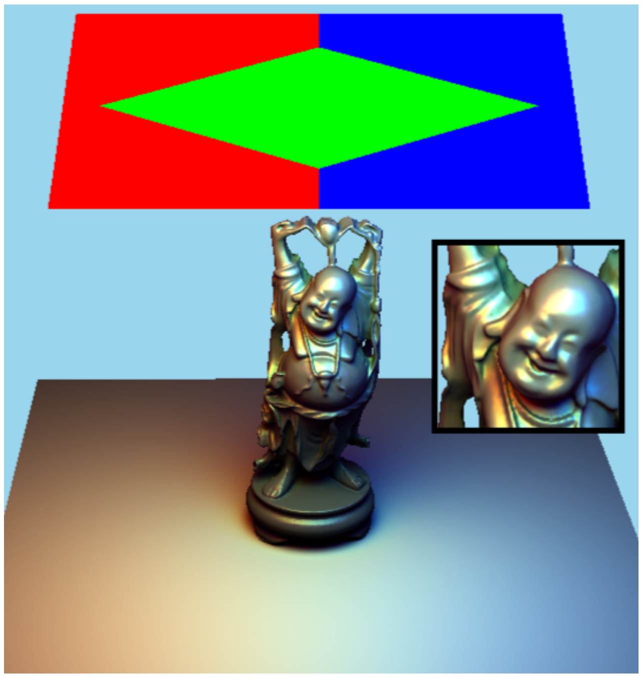


Fig. 10. More complex light sources with different patterns and colors can be broken into constituent polygons and used within our system. This example also shows glossy reflections.

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

J Arvo. 1995. Applications of irradiance tensors to the simulation of non-Lambertian phenomena. In *SIGGRAPH 95*. 335–342.

D. Baum, H. Rushmeier, and J. Winget. 1989. Improving Radiosity Solutions through the use of analytically determined form-factors. In *SIGGRAPH 89*. 325–334.

L. Belcour, G. Xie, C. Hery, M. Meyer, W. Jarosz, and D. Nowrouzezahrai. 2018. Integrating Clipped Spherical Harmonics Expansions. *ACM Transactions on Graphics* 37, 2 (2018), 19:1–19:12.

M Chen and J Arvo. 2001. Simulating Non-Lambertian Phenomena Involving Linearly-Varying Luminaires. In *Eurographics Workshop on Rendering*. 25–38.

J. Dupuy, E. Heitz, and L. Belcour. 2017. A Spherical Cap Preserving Parameterization for Spherical Distributions. *ACM Transactions on Graphics (Proc. SIGGRAPH 17)* 36, 4 (2017), 139:1–139:12.

E. Heitz, J. Dupuy, S. Hill, and D. Neubelt. 2016. Real-Time Polygonal-Light Shading with Linearly Transformed Cosines. *ACM Transactions on Graphics (Proc. SIGGRAPH 16)* 35, 4 (2016), 41:1–41:8.

T MacRobert. 1948. *Spherical harmonics: an elementary treatise on harmonic functions with applications.* Dover Publications.

R Ng, R Ramamoorthi, and P Hanrahan. 2003. All-Frequency Shadows using Non-Linear Wavelet Lighting Approximation. *ACM Transactions on Graphics (Proc. SIGGRAPH 03)* 22, 3 (2003), 376–381.

D. Nowrouzezahrai, P. Simari, and E. Fiume. 2012. Sparse zonal harmonic factorization for efficient SH rotation. *ACM Transactions on Graphics* 31, 3 (2012), 23:1–23:9.

J Pantaleoni, L Fascione, M Hill, and T Aila. 2010. PantaRay: fast ray-traced occlusion caching of massive scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH 10)* 29, 4 (2010).

R Ramamoorthi. 2009. Precomputation-Based Rendering. *Foundations and Trends in Computer Graphics and Vision* 3, 4 (2009), 281–369.

Z Ren, R Wang, J Snyder, K Zhou, X Liu, B Sun, P Sloan, H Bao, Q Peng, and B Guo. 2006. Real-time Soft Shadows in Dynamic Scenes using Spherical Harmonic Exponentiation. *ACM Transactions on Graphics (Proc. SIGGRAPH 06)* 25, 3 (2006), 977–986.

P Sloan, J Hall, J Hart, and J Snyder. 2003. Clustered Principal Components for Precomputed Radiance Transfer. *ACM Transactions on Graphics (Proc. SIGGRAPH 03)* 22, 3 (2003), 382–391.

P Sloan, J Kautz, and J Snyder. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Transactions on Graphics (Proc. SIGGRAPH 02)* 21, 3 (2002), 527–536.

P Sloan, B Luna, and J Snyder. 2005. Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics (Proc. SIGGRAPH 05)* 24, 3 (2005), 1216–1224.

J. Snyder. 1996. *Area Light Sources for Real-Time Graphics.* Technical Report MSR-TR-96-11. Microsoft Research.

D. Stern. 1965. Classification of Magnetic Shells. *Journal of Geophysics Research* 70, 15 (1965), 3629–3634.

B Sun and R Ramamoorthi. 2009. Affine double and triple product wavelet integrals for rendering. *ACM Transaction on Graphics* 28, 2 (2009).

Y Tsai and Z Shih. 2006. All-Frequency Precomputed Radiance Transfer using Spherical Radial Basis Functions and Clustered Tensor Approximation. *ACM Transactions on Graphics (Proc. SIGGRAPH 06)* 25, 3 (2006), 967–976.

K Zhou, Y Hu, S Lin, B Guo, and H Shum. 2005. Precomputed shadow fields for dynamic scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH 05)* 24, 3 (2005), 1196–1201.

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 γ_e in most places. We start with the formula for $C_{l,e}$ in equation 22, and integrate by parts,

$$\begin{aligned}\int zP_l(z)\,d\gamma &= \left(P_l(z)\int z d\gamma\right)\Big|_0^{\gamma_e} - \int\left(\int z d\gamma\right) d\gamma P_l'(z)\frac{dz}{d\gamma}\\ &= (a\sin\gamma - b\cos\gamma)P_l(z)\big|_0^{\gamma_e} + \int(-a\sin\gamma + b\cos\gamma)^2 P_l'(z)d\gamma. \quad (45)\end{aligned}$$

In the last line, we use $z = a\cos\gamma + b\sin\gamma$, making it easy to find $\int z\,d\gamma$ and $dz/d\gamma$. 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

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

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

$$\begin{aligned}\int(-a\sin\gamma + b\cos\gamma)^2 P_l'(z)d\gamma &= \left((-a\sin\gamma + b\cos\gamma)^2\int P_l'(z)d\gamma\right)\Big|_0^{\gamma_e}\\ &- \int\left(\int P_l'(z)d\gamma\right)(-2(b\cos\gamma - a\sin\gamma))(a\cos\gamma + b\sin\gamma)\,d\gamma\\ = \left((-a\sin\gamma + b\cos\gamma)^2\int P_l'(z)d\gamma\right)\Big|_0^{\gamma_e} &- \int\left(\int P_l'(z)d\gamma\right)(-2zdz). \quad (47)\end{aligned}$$

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

$$\begin{aligned}&\left((-a\sin\gamma + b\cos\gamma)^2\int P_l'(z)d\gamma + \left(\int P_l'(z)d\gamma\right)z^2\right)\Big|_0^{\gamma_e} - \int z^2P_l'(z)d\gamma\\ &= \left(((-a\sin\gamma + b\cos\gamma)^2 + z^2)\left(\int P_l'(z)d\gamma\right)\right)\Big|_0^{\gamma_e} - \int z^2P_l'(z)d\gamma\\ &= (a^2+b^2)\left(\int P_l'(z)d\gamma\right) - \int z^2P_l'(z)d\gamma. \quad (48)\end{aligned}$$

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 γ_e, since $a^2 + b^2$ is a constant.

We now use the identity in equation 10, rearranging to obtain $z^2P_l'(z) = l\,(zP_l(z) - P_{l-1}(z)) + P_l'(z)$. Substituting inside the second integral, the above expression becomes,

$$\begin{aligned}&(a^2+b^2)\left(\int P_l'(z)d\gamma\right) - \left(l\int zP_l(z)d\gamma\right)\\ &+\left(l\int P_{l-1}(z)d\gamma\right) - \left(\int P_l'(z)d\gamma\right) \quad (49)\\ &= (a^2+b^2-1)\left(\int P_l'(z)d\gamma\right) - \left(l\int zP_l(z)d\gamma\right) + \left(l\int P_{l-1}(z)d\gamma\right).\end{aligned}$$

Plugging back into equation 45 gives us

$$\begin{aligned}(l+1)\int zP_l(z)d\gamma = (a\sin\gamma - b\cos\gamma)P_l(z)\big|_0^{\gamma_e} +\\ (a^2+b^2-1)\left(\int P_l'(z)d\gamma\right) + \left(l\int P_{l-1}(z)d\gamma\right) \quad (50)\end{aligned}$$

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 $D_{l,e}$ and $B_{l-1,e}$, while equation 46 can be used for the evaluation at 0 and γ_e. Putting this together, and re-introducing the index and subscript e for the edge,

$$\begin{aligned}(l+1)C_{l,e} &= (a_e\sin\gamma_e - b_e\cos\gamma_e)P_l(a_e\cos\gamma_e + b_e\sin\gamma_e) + b_eP_l(a_e)\\ &+ (a_e^2 + b_e^2 - 1)D_{l,e} + lB_{l-1,e}. \quad (51)\end{aligned}$$

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

$$\begin{aligned}lC_{l-1,e} &= (a_e\sin\gamma_e - b_e\cos\gamma_e)P_{l-1}(a_e\cos\gamma_e + b_e\sin\gamma_e) + b_eP_{l-1}(a_e)\\ &+ (a_e^2 + b_e^2 - 1)D_{l-1,e} + (l-1)B_{l-2,e}. \quad (52)\end{aligned}$$

Upon dividing through by l, we obtain equation 31.