Eurographics Symposium on Rendering 2009
Hendrik P. A. Lensch and Peter-Pike Sloan
(Guest Editors)

Volume 28 (2009), Number 4

Fast Global Illumination on Dynamic Height Fields

Derek Nowrouzezahrai
University of Toronto

John Snyder
Microsoft Research

Abstract

We present a real-time method for rendering global illumination effects from large area and environmental lights on dynamic height fields. In contrast to previous work, our method handles inter-reflections (indirect lighting) and non-diffuse surfaces. To reduce sampling, we construct one multi-resolution pyramid for height variation to compute direct shadows, and another pyramid for each indirect bounce of incident radiance to compute inter-reflections. The basic principle is to sample the points blocking direct light, or shedding indirect light, from coarser levels of the pyramid the farther away they are from a given receiver point. We unify the representation of visibility and indirect radiance at discrete azimuthal directions (i.e., as a function of a single elevation angle) using the concept of a "casting set" of visible points along this direction whose contributions are collected in the basis of normalized Legendre polynomials. This analytic representation is compact, requires no precomputation, and allows efficient integration to produce the spherical visibility and indirect radiance signals. Sub-sampling visibility and indirect radiance, while shading with full-resolution surface normals, further increases performance without introducing noticeable artifacts. Our method renders 512x512 height fields (>500K triangles) at 36Hz.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Color, shading, shadowing, and texture—

1. Introduction

The human visual system is adapted to the physical world, where light typically reflects off many different surfaces before entering the eye. Global illumination (GI) thus enhances realism and perceptibility of synthetically rendered scenes. Shadows, the direct component of GI, provide critical cues about spatial relationships between objects but blacken points receiving no contribution from any light source. In reality, such points receive *indirect* light from surrounding geometry. A subtle case is color bleeding where colored geometry reflects correspondingly colored light to nearby geometry. More dramatic is the brightening within surface depressions, which reveals detail in what would otherwise be a black image void.

Though compelling, simulating indirect effects comes with a price: the computation is intensive and difficult to do in real-time. The problem arises because the relationship between lighting and shading becomes non-linear, spatially-varying, dependent on the global geometric configuration, and iterative. The geometry effectively acts as a complex light source that recursively relights itself until reaching an energy balance. Although indirect contributions can be precomputed in the case of static geometry, this approach fails to extend to geometry that changes over time. We present a real-time method for computing direct and indirect illumination on animated height field geometry under varying environmental and directional lighting. Computation of the direct visibility, and indirect lighting for each number of bounces, forms the bottleneck. We make it practical using the same approximation strategy for both types of functions.

Our work has two main contributions. First, we extend the multi-resolution method of [SN08] to handle indirect lighting. The basic idea is to construct a pyramid for geometry and radiance data to reduce aliasing and sampling requirements. This avoids the need to take more and more samples of visibility/indirect radiance as the distance from occluder/illuminator to receiver grows. For diffuse inter-reflections, we pyramidally filter the scalar shading over the height field at each lighting bounce; for glossy inter-reflection, we instead filter the (glossy) exit radiance, represented as a vector in the spherical harmonic (SH) basis.

Second, at each receiver point, we represent 1D functions

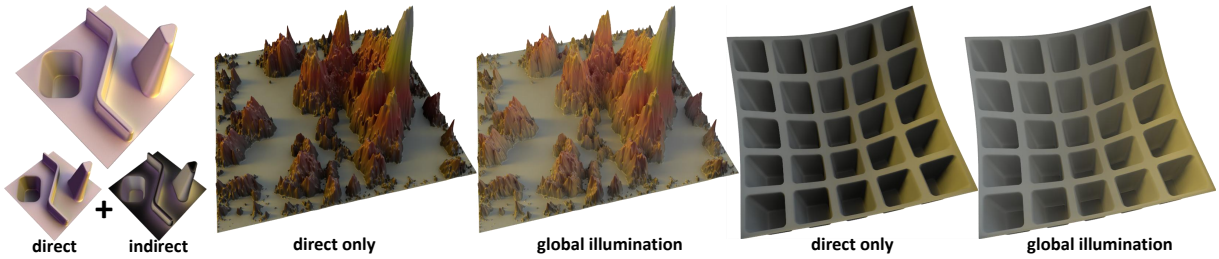

Figure 1: *Real-time global-illumination on dynamic height fields ($>35Hz$), compared to direct illumination only. All images in the paper are captured at high resolution from the real-time demo; readers are encouraged to digitally zoom in to reveal detail.*

of visibility and indirect radiance along each azimuthal ray in a compact, analytic form using the normalized Legendre polynomial (NLP) basis. We then integrate over samples in different azimuthal directions to produce spherical functions in the SH basis by applying a small linear operator on the NLP coefficients. For linear interpolation and order-4 SH, each azimuthal segment applies a 16×8 matrix to the two 4D NLP coefficient vectors at the two adjacent azimuthal samples. We also support B-spline azimuthal interpolation with a 16×32 matrix. Summing over all azimuthal segments yields the total incident radiance at the receiver, which is finally dotted with an SH vector representing its BRDF.

Our approach is fast. It is also simple and implemented completely on the GPU in less than 100 lines of shader code (see the Supplemental Material.) Its performance is independent of the geometry or complexity of the lighting environment and depends only on the height field resolution. It handles arbitrary numbers of bounces off diffuse and glossy surfaces with cost linear in the number of bounces. Lighting, geometry, and viewpoint can all be manipulated dynamically while maintaining framerates of at least 35 Hz for 512×512 height fields. Figure 1 shows the quality we achieve.

All approximations, including the use of multi-resolution pyramids for visibility and indirect radiance, allow control of error. The largest error source arises from our use of low-order (order-4) SH for spherical functions of radiance and visibility, which provides only low-frequency (soft) reflection effects. Highly specular reflections and sharp shadows can not be captured, though shadow sharpness is enhanced by using light sources of restricted extent [SN08].

2. Previous Work

Shadow mapping approaches [Wil78] compute direct illumination under point lighting for arbitrary geometry in real-time; see Hasenfratz et al. [HLHS03] for a survey. Soft shadows are typically approximated by filtering the results of depth comparison over a neighborhood in the shadow map [RSC87]. Recent approaches augment the shadow map with additional data, such as depth mean and variance [DL06] or functions of light-space z variation [AMB*07, AMS*08], to provide more control over shadow softness.

Annen et al. [ADM*08] ignore indirect illumination and use many shadow maps to shadow dynamic geometry under environmental lighting. Their method scales linearly with the number of shadow maps, and thus the number of area lights required to approximate the environment. The approach is restricted to rectangular light sources. It also incurs greater error for larger lights, since it assumes coplanarity of blockers, receivers, and light sources, and computes visibility with respect to the center of each light source.

Ritschel et al. [RGK*08] augment an instant-radiosity approach with *imperfect shadow maps*, which approximate the incident radiance from point and environment lighting. Quality of the indirect illumination, and algorithm performance, depend on the number of secondary lights. While the approach is suitable for small area light sources, larger ones (e.g. environmental) are approximated by many direct lights, which in turn require many more indirect light sources.

Precomputed radiance transfer (PRT) techniques generate soft shadows and GI effects for static scenes lit by large area and environmental sources [SKS02, KSS02, NRH03]. The costly ray-tracing precomputation precludes the use of standard PRT approaches for dynamic geometry.

Spherical Harmonics Exponentiation (SHEXP) [RWS*06, SGNS07] avoids this costly precomputation, by approximating dynamic geometry as a set of spherical blockers, and accumulating the occlusion from each sphere in a logarithmic SH space. In the case of complex and dynamic geometry such as detailed height fields, a blocker sphere approximation can not be precomputed (since we target general animation not described by simple "skinning" of an articulated skeleton) and requires too many spheres.

Height field rendering has a long history in computer graphics. Horizon mapping [Max88, SC00] precomputes horizon angles over discrete azimuthal directions, to render point light shadows. A similar method precomputes a circular aperture at each receiver [OS07]. *Precomputed visibility maps* handle both direct and indirect lighting by storing the location of the first visible point, over all receiver points and a discretization of incident directions [HDKS00]. These methods do not address dynamic geometry. Parallax occlusion mapping [Tat06] can approximate height displacement and soft shadows with a pixel-shader ray-tracer.

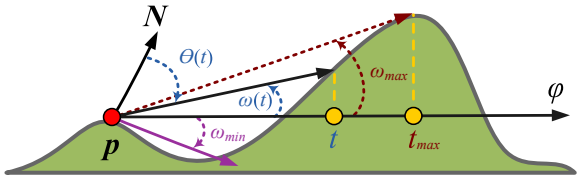

Figure 2: *Height field elevation angle notation.*

Similar techniques have been used in dynamic scenes to approximate ambient occlusion (AO) [SA07, AMA02] and GI in screen space [RGS09]. These approaches render geometry into a depth buffer and treat it as a height field. Shading integrals are approximated by examining a small set of samples nearby in screen space. AO complements standard point lighting with a softer "GI feel", but ignores cast shadows from lighting with strong directionality [RWS*06]. In general, these techniques render plausible images but ignore shadows and indirect light cast by geometry occluded in the view. What's more, the small number and locality of samples approximating the integrals preclude long-range GI effects. Our method accounts for long-range effects, eliminates aliasing, and converges to correct ground truth results.

Snyder and Nowrouzezahrai [SN08] use a multi-resolution pyramid of height variation to compute horizon maps in real-time without aliasing artifacts. We extend [SN08] to include indirect effects and handle non-diffuse surfaces. Indirect radiance is a more challenging function to manipulate compared to simple (binary) visibility. Visibility in a given azimuthal direction is completely captured by a single scalar representing the maximum blocking angle, whereas indirect radiance varies over the height field and is received from all unoccluded points along that direction. Technical innovations developed to deal with this problem include the detection of visible samples for indirect accumulation using the idea of a "casting set", the use of the NLP basis to represent such azimuthal slice functions, the ability to handle visibility and indirect radiance below the z=0 plane, and higher-order interpolation of azimuthal samples to further reduce aliasing.

3. Overview and Terminology

Height fields are scalar functions on the plane defining a surface of points $p = (x, y, f(x,y))$, where f evaluates the height at planar position, (x,y). We denote the unit length normal vector at p as N. We generate height fields with analytic formulae or simple simulation procedures, but any method producing a 2D array of heights can be substituted.

Visibility over the height field is represented by first considering the simplified case of a single point p receiving light along a single azimuthal direction φ. The associated ray is given by $r(t) = (x,y) + t(\cos\varphi, \sin\varphi)$ and 3D points along it are denoted $p(t) = (r(t), f(r(t)))$.

The *blocking angle* at p, formed between the horizon and height field points along $r(t)$, is given by

$$\omega(t) = \tan^{-1}\left(\frac{f(r(t)) - f(r(0))}{t}\right) . \quad (1)$$

This angle is measured from the horizon up. Its associated *visible angle* is defined as $\theta = (\pi/2 - \omega)$ and measured from the zenith down. While θ follows standard spherical parameterization, it is more natural to express visibility and indirect radiance in terms of ω.

We define the *binary visibility function* as

$$v(\omega;\sigma) = \begin{cases} 0, & \text{if } \omega \le \sigma \\ 1, & \text{otherwise} , \end{cases} \quad (2)$$

where σ represents the elevation angle at which the transition from blocked to visible occurs, as one looks increasingly higher from p along the azimuthal direction φ. The corresponding *binary occlusion function* is just the logical inversion of this visibility function, denoted

$$\bar{v}(\omega;\delta) = 1 - v(\omega;\delta) . \quad (3)$$

The transition angle, or *maximum blocking angle* at p, is

$$\omega_{max} = \max_{t\in(0,\infty)} \omega(t) , \quad (4)$$

and occurs at t_{max}. The *minimum blocking angle* is defined by the upper hemisphere about N and given by $\omega_{min} = -\arcsin(N \cdot (\cos\varphi, \sin\varphi, 0))$ (see Figure 2.)

Then the visibility along $r(t)$, as a function of blocking angle ω, is

$$v(\omega) = v(\omega;\omega_{max}) = \begin{cases} 0, & \text{if } \omega \le \omega_{max} \\ 1, & \text{otherwise} . \end{cases} \quad (5)$$

The casting set is defined as the set of points along $p(t)$ that are visible (i.e., not occluded) from p. More formally, $\mathcal{T}$ is the set of values of t whose corresponding blocking angle $\omega(t)$ is larger than that for all smaller t; i.e.,

$$\mathcal{T} = \{t \mid \omega(t') < \omega(t) \;\; \forall\, t' < t\} . \quad (6)$$

Such $t \in \mathcal{T}$ thus appear above all intervening geometry in the height field, as seen from p. Discrete elements of this set are denoted $t_i \in \mathcal{T}$; the actual discretization is explained further in Section 5. Figure 3 illustrates an example.

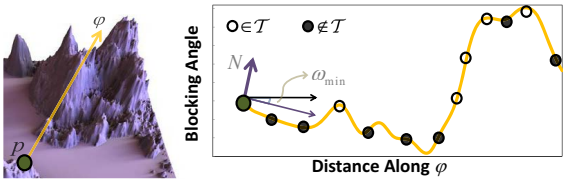

Figure 3: *Monotonically increasing blocking angles above ω_{min}, marked in white, form the casting set. Samples in black are not visible from the receiver p and thus excluded.*

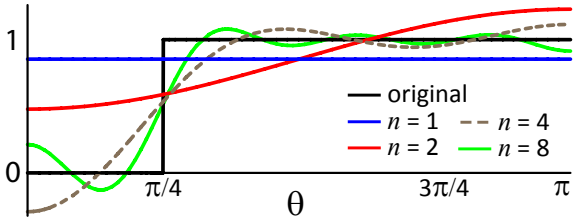

Figure 4: *Normalized Legendre polynomial reconstruction of the occlusion function $\bar{v}(\omega;\pi/4)$, for different orders, n.*

Indirect radiance (IR) towards p from any point along $p(t)$ arrives from direction $s(t) = -(\theta(t),\varphi)$, and is defined as

$$u(t) = u\left(p(t),\, p(t) \to p(0)\right) = u\left(p(t),\, s(t)\right) \,. \qquad (7)$$

When computing the b^{th} bounce of indirect illumination, we require the radiance from the $(b-1)^{th}$ bounce. Assuming piece-wise constant variation of radiance along the elevation angle, in terms of a number of discrete samples $t_i \in \mathcal{T}$, we express IR along $r(t)$ as a function of the blocking angle ω:

$$u(\omega) = \begin{cases} u(t_i), & \text{if } \omega(t_{i-1}) \le \omega < \omega(t_i) \\ 0, & \text{if } \omega < \omega_{min} \text{ or } \omega > \omega_{max} \,. \end{cases} \qquad (8)$$

Piece-wise constant interpolation is justified because adjacent values of $t_i \in \mathcal{T}$ need not be close on the height field; they may arise from entirely different visible surface layers (see Figure 3). This approximation converges to the correct result with increased sampling in t.

4. Compact Representation of 1D Elevation Functions

Equations 5 and 8 can be compactly expressed in the orthogonal NLP basis. This is a natural choice when using the SH basis for spherical functions, since the order-n NLP basis (containing n basis functions) captures all the variation in elevation angle one obtains from one azimuthal slice of an order-n SH expansion (containing n^2 basis functions). Each NLP basis function, denoted $\hat{P}_l(\cos\theta)$, is a degree-l polynomial in $\cos\theta$. NLPs are related to zonal harmonics (ZH) via

$$\mathbf{y}_l(\theta,\phi) = \sqrt{\frac{1}{2\pi}}\,\hat{P}_l(\cos\theta) = \sqrt{\frac{2l+1}{4\pi}}\,P_l(\cos\theta) \,, \qquad (9)$$

where P are the (unnormalized) Legendre polynomials. The vector corresponding to the first $n = 4$ normalized Legendre basis functions evaluated at $z = \cos\theta$ is

$$\hat{\mathbf{P}}(z) = \left[\sqrt{\frac{1}{2}}, \frac{\sqrt{3}z}{\sqrt{2}}, \frac{5\sqrt{2}(3z^2-1)}{4}, \frac{7\sqrt{2}(5z^3-3z)}{4}\right] .$$

The NLP basis function is given by Rodrigues' formula:

$$\hat{P}_l(z) = \sqrt{\frac{2l+1}{2}}\,\frac{1}{2^l\,l!}\,\frac{d^l}{dz^l}[(z^2-1)^l] \,.$$

satisfying orthonormality over the interval $[-1,1]$:

$$\int_{-1}^{1} \hat{P}_i(z)\,\hat{P}_j(z)\,dz = \delta_{ij} \,,$$

where δ_{ij} is the Kroenecker delta function.

An order-n NLP expansion of the binary occlusion function in Equation 3 is then given by

$$\bar{\mathbf{v}}(\sigma) = \int_0^{\pi} \bar{v}(\omega;\sigma)\,\hat{\mathbf{P}}(z)\,dz = \int_{\frac{\pi}{2}-\sigma}^{\pi} \hat{\mathbf{P}}(z)\,dz \,, \qquad (10)$$

with $dz = \sin\theta\,d\theta$. We use Equation 10 to obtain projection coefficients of Equations 5 and 8 in the NLP basis via

$$\boldsymbol{v} = \bar{\mathbf{v}}(\pi/2) - \bar{\mathbf{v}}(\omega_{max}) \qquad (11)$$

$$\begin{aligned} \boldsymbol{u} &= \int_0^{\frac{\pi}{2}} u(\theta)\hat{\mathbf{P}}(\cos\theta)\sin\theta\,d\theta \\ &= \sum_i \Big[u(t_i) - u(t_{i+1})\Big]\,\bar{\mathbf{v}}(\omega(t_i)) - u(t_0)\,\bar{\mathbf{v}}(\omega_{min}), \end{aligned} \qquad (12)$$

where $\boldsymbol{v}$ and $\boldsymbol{u}$ are the coefficient vectors of visibility and IR. Here, $u(t_{i+1}) = 0$ when $i \ge |\mathcal{T}|$. Equation 11 converts occlusion via subtraction to reconstruct visibility in terms of the maximum blocking angle ω_{max}. Equation 12 accumulates indirect radiance over segments of successively increasing blocking angles in the casting set. The last term subtracts out radiance below the lower hemisphere, $\omega < \omega_{min}$.

Reconstruction of continuous functions from these projection vectors requires a simple dot product:

$$v(\theta) = \boldsymbol{v}\cdot\hat{\mathbf{P}}\left(\cos(\theta)\right),\; u(\theta) = \boldsymbol{u}\cdot\hat{\mathbf{P}}\left(\cos(\theta)\right). \qquad (13)$$

This representation unifies the two quantities necessary for determining the direct and indirect illumination along the ray. We can precompute $\bar{\mathbf{v}}$ in a 1D lookup table for many values of σ or use the following order-4 analytic form:

$$\bar{\mathbf{v}}(\sigma) = \Big[\frac{\sin\sigma+1}{\sqrt{2}}, \frac{-3\cos^2\sigma}{2\sqrt{6}}, \frac{-5\sin\sigma\cos^2\sigma}{2\sqrt{10}}, \frac{7\cos^2\sigma(-4+5\cos^2\sigma)}{8\sqrt{14}}\Big] .$$

Figure 4 illustrates the reconstruction of $\bar{v}(\omega;\sigma)$ with increasing order n. Section 6 discusses how these visibility and indirect radiance vectors can be integrated over several azimuthal directions φ to yield spherical functions for direct and indirect shading. Discretization and multi-resolution sampling along the ray parameter t is explained next.

5. Multi-Resolution Sampling

5.1. Uniform Sampling

Uniformly sampling along $r(t)$, the maximum blocking angle and casting set (Equations 4 and 6) can be computed via

$$\omega_{max} \approx \max_i \left[\tan^{-1}\left(\frac{f(x+t_i\cos\varphi, y+t_i\sin\varphi) - f(x,y)}{t_i}\right)\right],$$

$$\mathcal{T} \approx \{t_i \mid \omega(t_j) < \omega(t_i),\; \forall j < i\} \,, \text{ with } t_i = i\,\Delta t,$$

where Δt represents the discrete step size in t. We note a small abuse in notation in that the t_i here include all points along the ray, some of which are occluded and thus not members of $\mathcal{T}$. When scanning to compute $\mathcal{T}$, successive indices denote successive points along the ray, while in scanning

over the casting set, as in Equation 8, they denote successive members of $\mathcal{T}$.

Uniform sampling leads to aliasing unless an impractically large number of samples is taken. This is especially true because we are integrating over 2D domains, parameterized by both distance t and azimuthal direction φ. To ensure all height field features are adequately sampled, sampling must be tied to domain area and so must sample more as the distance to the receiver point increases. Intuitively though, by adequately prefiltering the geometry and indirect radiance, we can avoid this problem.

5.2. Multi-Resolution Height Sampling

As in [SN08], we reduce the sampling requirements with a *height pyramid*. Each pyramid level is denoted $f_i(x,y)$, $i \in \{0,1,\cdots,n_v-1\}$. It is a filtered version of the original height field which reduces its resolution by a factor of $2^{1/k_v}$ in both x and y, using 2D B-spline filtering. The *level step*, k_v, controls how fast levels decrease in resolution as a function of their index i; the total number of pyramid levels is denoted n_v. We typically set $k_v = 4$, corresponding to a pyramid with 4 levels per level-of-2 reduction in resolution, as in [SN08].

Instead of stepping uniformly along the ray $r(t)$, we take steps that increase exponentially via

$$\tau_i = 2^{-(n_v-1-i)/k_v} . \tag{14}$$

As τ_i increases, we access increasingly filtered levels in the pyramid. The blocking angle at a given level is computed by

$$\omega_i = \tan^{-1}\left(\frac{f_i(x+\tau_i\cos\varphi, y+\tau_i\sin\varphi) - f_i(x,y)}{\tau_i}\right) . \tag{15}$$

Equation 4 can now be approximated as the maximum over a 1D B-spline interpolant, sampled at the knots and midpoints, over these ω_i values:

$$\omega_{max} \approx \max_{\tau} \underbrace{\text{B-spline}\left(\tau, \{\omega_0,\cdots,\omega_{n_v-1}\}\right)}_{\omega(\tau)} . \tag{16}$$

A *level offset*, o_v, can be used to bias the pyramid level by replacing f_i in Equation 15 with $f_{i+k_v o_v}$. Increasing o_v reduces the "blurring" of the height field with distance and so produces sharper long-range shadows, but also reduces the ability of the pyramid to control aliasing. We typically set $o_v = 2$ for visibility sampling.

We compute Equation 16 on the GPU, by scanning τ_i over all pyramid levels ($i = 0,\ldots,n_v-1$) and only storing the visible τ_i; i.e, those with blocking angle ω_i larger than that from all previous τ_i, forming a discrete approximation of $\mathcal{T}$

$$\mathcal{T} \approx \{\tau_i \mid \omega(\tau_j) < \omega(\tau_i) \;\; \forall\, j < i\} . \tag{17}$$

Results from scanning over pyramid levels (ω_{max} and $\tau_i \in \mathcal{T}$) are then substituted into Equations 11 and 12 to compute the NLP coefficients of the 1D visibility and IR functions. Evaluating radiance values, $u(\tau_i)$, in Equation 12 depends on the BRDF at the point $p(\tau_i)$, as discussed in the next section.

5.3. Multi-Resolution Radiance Sampling

We use the same multi-resolution strategy to efficiently compute $u(\tau_i)$ at visible points $\tau_i \in \mathcal{T}$, as seen by the receiver point p in the azimuthal direction φ. The quantities necessary for computing u are accessed from a *radiance pyramid* representing exit radiance at each point on the height field.

Diffuse surfaces emit radiance uniformly in all directions, and thus require the storage of only a single (trichromatic) value at each surface point. In order to compute the b^{th} bounce of light, we process diffuse shading results from the $(b-1)^{th}$ bounce and generate a multi-resolution shading pyramid on these values. As before with the height pyramid, this pyramid is generated using 2D B-spline filtering but using n_u levels (level step k_u), and accessed with level offset o_u. The $u(\tau_i)$ are then sampled from this pyramid.

Non-diffuse surfaces require more complicated handling since the outgoing radiance from the $(b-1)^{th}$ bounce of lighting is a full (trichromatic) spherical function, not a scalar. We therefore store and filter the IR vector from the $(b-1)^{th}$ bounce with the shading pyramid. The $u(\tau_i)$ is calculated as the dot product of the SH vector obtained by evaluating the BRDF in the direction $s(t)$ with the SH vector representing IR from the pyramid (see Section 7.) We can also augment vectors sampled from the radiance pyramid with any additional spatially varying information, such as parameters of the BRDF (e.g. Phong exponent) or albedo.

Mixture surfaces, composed of diffuse and glossy components, can be handled by combining the above approaches using two pyramids. Alternatively, we can approximate this result using a single pyramid which filters the sum of the diffuse and glossy scalar shades and then applies the diffuse-only sampling strategy. This is not strictly correct since the outgoing glossy radiance is evaluated in the direction towards the viewer rather than towards p; i.e., in the direction $s(t)$. If the surface is not overly specular, then the error introduced is small. This approximation is similar to the *diffuse indirect bounce* approach in [BAEDR08]. Figure 5 compares results of our two approximation approaches. Most error arises from reduced sampling in t in φ.

6. Constructing Spherical Functions

So far, we have explained how to compute visibility and IR (Equations 5 and 8) only along a single azimuthal direction. Shading requires full spherical functions. We obtain them by sampling at a discrete, uniformly-spaced set of azimuthal directions, $\Phi = \{\varphi_1,\cdots,\varphi_m\}$. Each consecutive pair of azimuthal samples forms an *azimuthal wedge* having angular extent $\Delta\varphi = \varphi_{j+1} - \varphi_j$. Unlike [SN08], we interpolate

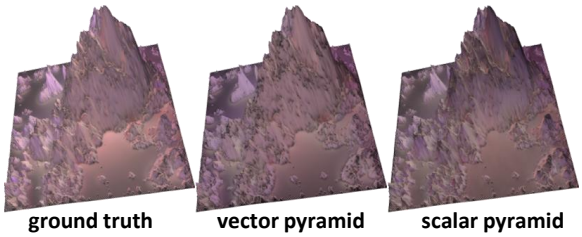

Figure 5: *Approximating glossy indirect illumination (only indirect lighting is illustrated.) Left: ground truth. Middle: multi-res vector (radiance) pyramid (converges to the ground truth; see Section 8). Right: multi-res scalar (shading) pyramid, similar to the approximation in [BAEDR08].*

1D functions rather than scalar blocking angles, over each wedge. Our method also allows higher-order interpolation.

For visibility due to environmental illumination, and all IR computations, Φ spans the complete azimuthal domain, $[0, 2\pi]$. For visibility due to smaller light sources, Φ is restricted to the light's partial azimuthal extent as in [SN08]. Complete domains yield m wedges, with an additional wedge between the first and last samples, while partial ones yield $m-1$ wedges. Another benefit of NLP-based elevation functions is that visibility for smaller light sources can now be restricted in the elevation domain, as well as azimuthally, producing even sharper shadowing effects with low-order SH lighting. To accomplish this, the parameters of the two terms in Equation 11 become the intersection of the angular intervals $[\omega_{max}, \pi/2]$ and $[l_{min}, l_{max}]$, where l_{min} and l_{max} are the elevation ranges of the light source, in terms of ω.

For each azimuthal direction φ_j, we denote the NLP projection of visibility and IR as $\boldsymbol{v}_j$ and $\boldsymbol{u}_j$. The spherical function within the wedge j between azimuthal directions φ_j and φ_{j+1} is expressed using a general interpolation operator applied to the whole set of discrete azimuthal directions, denoted $\mathbb{I}$. Projecting each wedge j into the SH basis yields

$$\boldsymbol{V}_j = \int_0^{\Delta\varphi}\int_0^{\pi} \mathbb{I}\left(\frac{\phi+\varphi_j}{\Delta\varphi}; \boldsymbol{v}_1, \cdots, \boldsymbol{v}_m\right) \boldsymbol{Y}(\theta,\phi)\sin\theta\, d\theta\, d\phi \tag{18}$$

$$\boldsymbol{U}_j = \int_0^{\Delta\varphi}\int_0^{\pi} \mathbb{I}\left(\frac{\phi+\varphi_j}{\Delta\varphi}; \boldsymbol{u}_1, \cdots, \boldsymbol{u}_m\right) \boldsymbol{Y}(\theta,\phi)\sin\theta\, d\theta\, d\phi\,, \tag{19}$$

where $\boldsymbol{Y}(\theta,\phi)$ is the vector of SH basis functions. This formula canonically re-orients each wedge to start at ϕ=0. A continuous integrand in θ is obtained by applying the NLP reconstruction rule in Equation 13.

Integration in Equation 18 and 19 is a linear operator on the NLP projection coefficients $\boldsymbol{v}_j$ and $\boldsymbol{u}_j$. For linear interpolation, the formula reduces to

$$\boldsymbol{V}_j = \mathbb{M}_{lin}\begin{bmatrix}\boldsymbol{v}_j \\ \boldsymbol{v}_{j+1}\end{bmatrix},\quad \boldsymbol{U}_j = \mathbb{M}_{lin}\begin{bmatrix}\boldsymbol{u}_j \\ \boldsymbol{u}_{j+1}\end{bmatrix}. \tag{20}$$

B-spline interpolation is similar but uses a larger matrix which operates on a vector concatenating 4 consecutive azimuthal samples rather than 2. The appendix details how $\mathbb{M}$ is computed for linear and B-spline interpolation.

Finally, to accumulate contributions over all wedges, we perform a fast SH Z-rotation of $\boldsymbol{V}_j$ and $\boldsymbol{U}_j$ into the wedge's actual azimuthal position and sum the resulting vectors:

$$\boldsymbol{V} = \sum_{j=1}^{m-1} \mathrm{Rot}_z(\boldsymbol{V}_j, \varphi_j),\quad \boldsymbol{U} = \sum_{j=1}^{m-1} \mathrm{Rot}_z(\boldsymbol{U}_j, \varphi_j)\,. \tag{21}$$

7. Shading in SH

Shading at a receiver point p is computed as the double product integral (dot product) of SH vectors representing the BRDF evaluated in the view direction ω_o, denoted $\boldsymbol{F}(\omega_o)$, with the incident radiance. We represent BRDF vectors with circularly symmetric functions. For low-frequency reflectance, these are compactly represented with ZH. This basis also provides an efficient rotation algorithm [SLS05].

A diffuse BRDF is symmetric about the normal N and has order-4 ZH vector $\{0.886227, 1.02333, 0.495416, 0\}$. The Phong BRDF is symmetric about the reflection vector and represented in canonical orientation by the ZH approximation $\{1, e^{[-1/(2S)]}, e^{[-2/S]}, e^{[-9/(2S)]}\}$, with Phong exponent S [RH02]. In either case, $\boldsymbol{F}(\omega_o)$ is computed by rotating the ZH vector to the appropriate frame, in which the z axis aligns with the normal or reflection vector. Our method easily extends to general BRDFs, using e.g. [KSS02, NKF09].

For direct illumination, the incident radiance is the SH product of (typically distant) lighting, $\boldsymbol{L}$, and the visibility, $\boldsymbol{V}$, from (21). We compute the SH visibility for each area (key) light and for environmental lighting using the multi-resolution approach discussed in Section 5.2. Environmental lights take samples over the entire azimuth, while key lights concentrate them over the light's azimuthal extent [SN08].

For indirect illumination, the incident radiance is $\boldsymbol{U}$ from (21). Each indirect bounce b is computed as discussed in Section 5.3. In the case of diffuse BRDFs or the approximate (scalar pyramid) mixture approach, the scalar shade from the previous bounce of lighting is forwarded to the IR computation algorithm to compute the next bounce. Results from all bounces are accumulated into the final shaded result. In the case of mixture surfaces, both the diffuse shade and the indirect radiance must be forwarded to the next pass.

Sub-sampling visibility and IR yields significant performance gains. A 2×2 sub-sampling has little effect on shading quality when shading is performed with full-resolution normals, as shown in Figure 7. Stages that can be sub-sampled are shown with *'s in Figure 6. We compute the visibility and IR vectors every frame (see Section 9) but amortize the direct and indirect shading computation between frames when the geometry is static: on even numbered frames the direct shading is computed using the visibility,

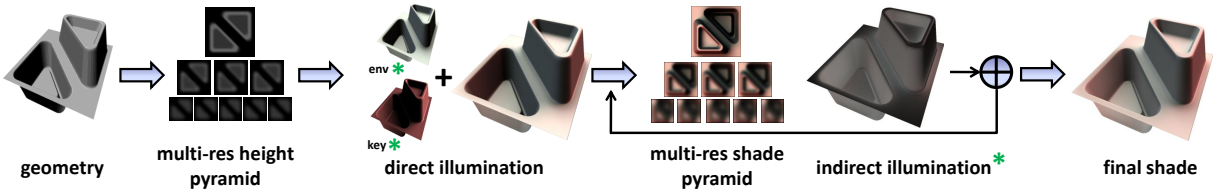

Figure 6: *Rendering pipeline: stages marked * may be sub-sampled. Stages 4 and 5 are accumulated for each indirect bounce.*

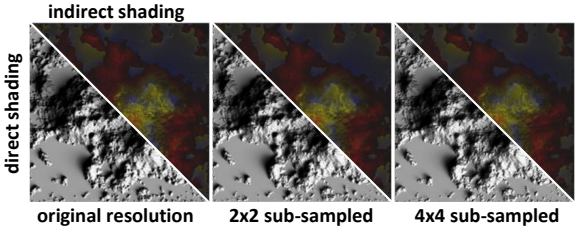

Figure 7: *Illumination sub-sampling. 2×2 yields a good trade-off between performance and quality; at 4×4, slight bleeding of shadows and indirect illumination occurs.*

and on odd numbered frames the indirect shading is computed using the IR. When the geometry changes, both the direct and indirect shade are updated.

8. Approximations and Convergence to Ground Truth

Our algorithm approximates brute-force evaluation of the rendering equation in several ways. It represents spherical functions of visibility and radiance using low-order and thus low-frequency SH, and 1D azimuthal slice functions in the NLP basis. It samples heights and incoming radiance along each ray using multi-resolution pyramids. And it interpolates azimuthal wedges over discrete sets of azimuthal samples. Error in each of these steps can be reduced within any desired limit using simple (but possibly computationally expensive) choices for the algorithm parameters.

Both the SH and NLP bases have the property that they represent any smooth function arbitrarily accurately as the order increases. However, they are an inefficient choice for representing detailed functions of visibility or radiance [NRH03], and better suited for handling low-frequency (soft) GI effects. In the case of multi-resolution sampling, the number of levels in the pyramid can be increased and the filtering reduced (or equivalently, the level bias increased) until sampling becomes equivalent to brute-force ray-marching. The number of azimuthal samples can also be increased, which reduces the significance of azimuthal interpolation. We demonstrate convergence to ground truth, computed with ray-tracing and order-4 SH lighting, in Figure 10.

9. Implementation and Performance

Our GI pipeline is illustrated in Figure 6 and is implemented as a set of GPGPU fragment shader kernels. We tessellate the height field using a vertex at each grid point and a pair of triangles connecting each adjacent "quad" of 4 vertices. A multi-resolution pyramid for the height field geometry is first generated using multiple render passes on the GPU. Repeated bilinear decimation the height field is followed by B-spline synthesis for each pyramid level, using the approach of [SH05]. We maintain a stack of synthesized textures (at the potentially sub-sampled resolution), however an alternative approach is discussed in Section 10. The max blocking angle ω_{max}, and casting set $\mathcal{T}$, are then computed at each receiver point for each azimuthal direction with Equations 16 and 17. We use 32 azimuthal samples for environmental visibility and indirect illumination sampling, and 3 samples for key light visibility. Sampling in τ and over azimuthal directions forms the computational bottleneck of our approach. Note that key lights may be absorbed into the lighting environment for better performance but separating them out yields sharper shadows (see Supplemental Material.) Equations 11 and 12 perform the Legendre projection of visibility and IR at each azimuthal direction. After looping over all azimuthal samples, Equations 20 and 21 then integrate over azimuthal wedges to obtain SH vectors for direct shadowing, $\boldsymbol{V}$, and IR, $\boldsymbol{U}$. Each vertex is shaded as described in Section 7 to yield the shading texture map applied to the geometry.

Images in the paper were computed in real-time, and except for Figure 8, using a single indirect bounce. Computation time scales linearly in the number of azimuthal samples and height field resolution, and sub-linearly with the number of levels used in each pyramid. Shading amortization results in a $5-10\%$ performance gain on low-to-mid range GPUs, and up to 45% on a high-end GPU. Table 1 summarizes performance and memory usage; note that each power-of-2 multiple of sub-sampling affords a memory gain of almost 4x and reported memory usage includes all input and intermediate (16-bit float) textures. The approach in [SN08] was timed using the original executable but it is possible to apply sub-sampling to it as well.

With 2x sub-sampling, our algorithm is faster than the approach of [SN08], but accounts for multi-bounce GI effects, sharper and more accurate key light shadows, and lighting below $z = 0$. Our performance advantage comes mainly from sub-sampling and using an analytic formula as opposed to a costly visibility wedge texture lookup. Figure 9 compares the two techniques. Figure 8 shows multi-bounce results.

10. Conclusion and Future Work

Indirect effects provide realism and enhance visualization cues by revealing shape detail in the umbra. We formulate

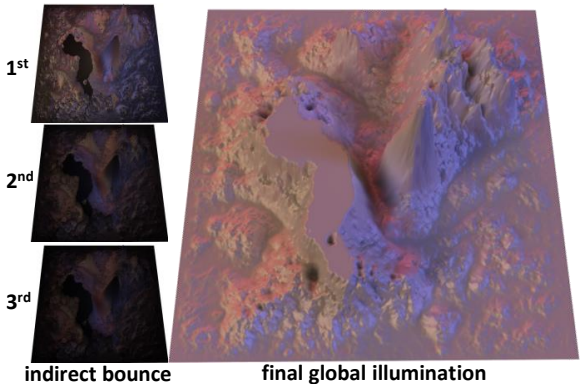

Figure 8: *Real-time multi-bounce results.*

HF Res.	Sub-sample	Mem. (MBs)	FPS for 1, 2, & 3 indirect bounces (Hz)		
			8800 GT	GTX 285	GTX 285*
256^2	1x	17.4	15.7, 13, 11	49.5, 44, 40	100, 86, 79
	2x	4.5	43, 39, 36	125, 107.7, 101	230, 194, 176
	4x	1.65	89, 81, 76	229, 198, 185	363, 303, 276
512^2	1x	76	3.5, 2.8, 2.4	11.6, 10.5, 10	24, 22, 20
	2x	20	11, 10, 8.9	35.5, 32, 30	67, 60, 56
	4x	7.1	26, 23, 21	73, 65.6, 62	124, 110, 104
1024^2	1x	336	0.8, 0.6, 0.4	2.7, 2.4, 2.2	5, 5, 4
	2x	88	2.1, 1.6, 1.2	8.3, 7.6, 7.1	16, 14, 14
	4x	29.5	4.5, 3.5, 2.9	18.9, 17.2, 16.4	32, 30, 28
			Framerate for [SN08] (Hz)		
256^2	1x	5	24.3	129	n/a
512^2	1x	22	5.8	30	n/a
1024^2	1x	96	1.6	7.2	n/a

* performance with shading amortization enabled.

Table 1: *Performance with $k_v = k_u = 4$, 1 env. light ($m = 32$), 1 key light ($m = 3$), and 8X multi-sampling measured on PCs with nVidia 8800 GT (512 MB) and GTX 285 (1 GB) GPUs.*

direct and indirect illumination on height fields in a unified manner, by introducing a compact representation for visibility and indirect radiance at discrete azimuthal directions using the NLP basis. Each azimuthal slice is accumulated over samples in the visible (casting) set using analytic expressions for NLP occlusion. We then efficiently collect these samples into full spherical functions using analytic interpolation operators over azimuthal wedges. All computations are local and map naturally to the GPU, allowing real-time global illumination on dynamically changing geometry with varying light, view, and BRDF. As mentioned in Section 9, we currently synthesize a "stack" of visibility/IR values with pyramid levels "blown out" to high-resolution via B-spline interpolation every frame. Storing an actual pyramid saves memory (>30x for 256^2 and 512^2, and 40x for 1024^2) but requires non-natively supported texture interpolation on-the-fly and so currently reduces performance.

Our algorithm currently exploits properties of height fields in several ways, most notably in that a single direction on the domain maps to just one global azimuthal direction. This is not true for more general geometry parameterizations where all three coordinates vary over the 2D domain. Our technique can also be applied in screen space, to better handle long-range GI effects without aliasing compared to [SA07, AMA02, RGS09]. Currently, only a single height field patch is supported. In the future we plan on investigating the coupling of overlapping height field patches as a *local* representation for more general geometry. Coupling our indirect illumination sampling with other shadowing techniques (e.g. shadow maps) is a straightforward extension.

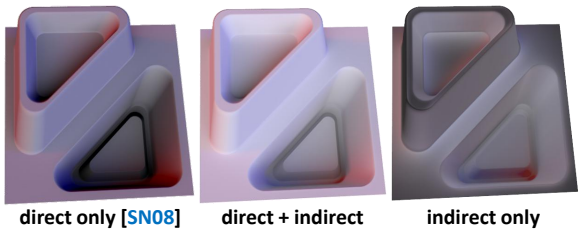

Figure 9: *Comparing direct illumination [SN08] (left, 28Hz) with our global illumination results (middle/right, 36Hz).*

Acknowledgments: The first author is partially funded by the National Sciences and Engineering Research Council of Canada. We thank the anonymous reviewers for their helpful suggestions and comments leading to clarifications included in the final manuscript.

References

[ADM*08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. In *ACM SIGGRAPH papers* (NY, USA, 2008).

[AMA02] AKENINE-MÖLLER T., ASSARSSON U.: Approximate soft shadows on arbitrary surfaces using penumbra wedges. *13th Eurographics Workshop on Rendering* (2002), 297–305.

[AMB*07] ANNEN T., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Convolution shadow maps. In *Eurographics Symposium on Rendering* (Grenoble, France, 2007).

[AMS*08] ANNEN T., MERTENS T., SEIDEL H.-P., FLERACKERS E., KAUTZ J.: Exponential shadow maps. In *Proceedings of Graphics Interface* (Toronto, Canada, 2008).

[BAEDR08] BEN-ARTZI A., EGAN K., DURAND F., RAMAMOORTHI R.: A precomputed polynomial representation for interactive brdf editing with global illumination. *ACM Trans. Graph.* (2008).

[DL06] DONNELLY W., LAURITZEN A.: Variance shadow maps. In *Proceedings on Interactive 3D graphics and games* (2006).

[HDKS00] HEIDRICH W., DAUBERT K., KAUTZ J., SEIDEL H.: Illuminating micro geometry based on precomputed visibility. In *Proceedings of ACM SIGGRAPH 2000* (2000).

[HLHS03] HASENFRATZ J., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadows algorithms. *Computer Graphics Forum 22*, 4 (2003).

[KSS02] KAUTZ J., SLOAN P., SNYDER J.: Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *Proc. of the 13th Eurographics Workshop on Rendering* (2002).

[Max88] MAX N.: Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer 4*, 2 (1988).

[NKF09] NOWROUZEZAHRAI D., KALOGERAKIS E., FIUME

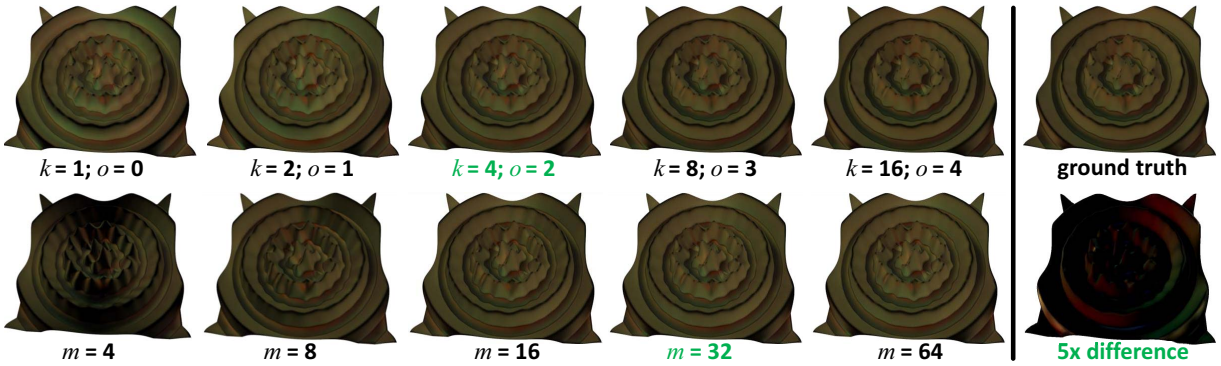

Figure 10: *Comparison of our indirect effects with ray-tracing. Our approach converges to the ground truth as the number of azimuthal samples (m) increases (bottom row, with $k_v = k_u = 4$ and $o_v = o_u = 2$) and as the sampling along each azimuthal direction ($k = k_u = k_v$, $o = o_u = o_v$) increases (top row, with $m = 100$.) Our GPU implementation uses the settings marked in green. The bottom right image shows the difference between these settings and ground truth, scaled by a factor of 5.*

E.: Shadowing dynamic scenes with arbitrary BRDFs. *Computer Graphics Forum (Proc. Eurographics) 28*, 2 (2009).

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph.* (2003).

[OS07] OAT C., SANDER P.: Ambient aperture lighting. In *Proc. on Interactive 3D Graphics and Games* (2007).

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps for efficient computation of indirect illumination. In *SIGGRAPH Asia papers* (New York, NY, USA, 2008), ACM.

[RGS09] RITSCHEL T., GROSCH T., SEIDEL H.-P.: Approximating dynamic global illumination in image space. In *Proc. on Interactive 3D Graphics and Games* (2009).

[RH02] RAMAMOORTHI R., HANRAHAN P.: Frequency environment maps. In *Proc. of ACM SIGGRAPH* (2002).

[RSC87] REEVES W., SALESIN D., COOK R.: Rendering antialiased shadows with depth maps. In *Proceedings of ACM SIGGRAPH* (1987).

[RWS*06] REN Z., WANG R., SNYDER J., ZHOU K., LIU X., SUN B., SLOAN P., BOA H., PENG Q., GUO B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM TOG 25*, 3 (2006).

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on GPUs. In *Proc. on Interactive 3D Graphics and Games* (2007).

[SC00] SLOAN P., COHEN M.: Interactive horizon mapping. In *Eurographics Workshop on Rendering Techniques* (June 2000).

[SGNS07] SLOAN P., GOVINDARAJU N., NOWROUZEZAHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *Pacific Graphics* (2007).

[SH05] SIGG C., HADWIGER M.: Fast third-order texture filtering. In *GPU Gems 2*. Addison-Wesley, 2005.

[SKS02] SLOAN P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM TOG 21*, 3 (2002).

[SLS05] SLOAN P., LUNA B., SNYDER J.: Local, deformable precomputed radiance transfer. *ACM TOG 24*, 3 (2005).

[SN08] SNYDER J., NOWROUZEZAHRAI D.: Fast soft self-shadowing on dynamic height fields. *Computer Graphics Forum: Proc. Eurographics Symposium on Rendering* (2008).

[Tat06] TATARCHUK N.: Dynamic parallax occlusion mapping with approximate soft shadows. In *I3D* (NY, USA, 2006), ACM.

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *Proceedings of ACM SIGGRAPH* (1978).

Appendix A: SH Interpolation Over Azimuthal Wedges

For a wedge spanning azimuthal directions j and $j+1$, we compute the SH projection coefficients of a spherical function $W(\theta,\phi)$ within the wedge (canonically re-oriented to start at $\phi = 0$) as

$$\boldsymbol{W} = \int_0^{\Delta\varphi}\int_0^{\pi} W(\theta,\phi)\,\boldsymbol{Y}(\theta,\phi)\,\sin(\theta)\,d\theta\,d\phi\,. \tag{22}$$

We define W as a continuous azimuthal blending of discrete elevation functions, with NLP projection coefficients $\boldsymbol{w}_j$:

$$W(\theta,\phi) = \mathbb{I}(\alpha_j;\boldsymbol{w}_1,\cdots,\boldsymbol{w}_m)\cdot\hat{\mathbf{P}}(z)\,, \tag{23}$$

where $\alpha_j = \frac{\phi-\varphi_j}{\Delta\varphi}$ and $z = \cos\theta$. In the case of linear interpolation,

$$\mathbb{I}(\alpha_j;\boldsymbol{x}_1,\cdots,\boldsymbol{x}_m) = \mathbb{I}_{lin} = (1-\alpha_j)\,\boldsymbol{x}_j + \alpha_j\boldsymbol{x}_{j+1}\,. \tag{24}$$

Substituting 24 and 23 into 22, we express this integral as a linear operator (matrix) acting on coefficient vectors $\boldsymbol{w}_j$ and $\boldsymbol{w}_{j+1}$,

$$\boldsymbol{W} = \mathbb{M}_{lin}(\Delta\varphi)\,\left[\boldsymbol{w}_j\ \boldsymbol{w}_{j+1}\right]^T,$$
$$\left[\mathbb{M}_{lin}\right]_{r,c} = \int_0^{\Delta\varphi}\int_0^{\pi}\left[\mathbb{I}_{lin}\right]_c\,\hat{P}_c(z)\,\boldsymbol{Y}_r(\theta,\phi)\,\sin(\theta)\,d\theta\,d\phi\,.$$

We can replace Equation 24 with any interpolation scheme. For B-spline interpolation, we have

$$\mathbb{I}(\alpha_j;\boldsymbol{x}_1,\cdots,\boldsymbol{x}_m) = \mathbb{I}_{bspl} = \beta_0(\alpha_j)\,\boldsymbol{w}_{j-1} + \beta_1(\alpha_j)\,\boldsymbol{w}_j + \beta_2(\alpha_j)\,\boldsymbol{w}_{j+1} + \beta_3(\alpha_j)\,\boldsymbol{w}_{j+2}\,,$$

where β_i are the B-spline basis functions (see [SN08]). Then

$$\boldsymbol{W} = \mathbb{M}_{bspl}(\Delta\varphi)\,\left[\boldsymbol{w}_{j-1}\ \boldsymbol{w}_j\ \boldsymbol{w}_{j+1}\ \boldsymbol{w}_{j+2}\right]^T,$$
$$\left[\mathbb{M}_{bspl}\right]_{r,c} = \int_0^{\Delta\varphi}\int_0^{\pi}\left[\mathbb{I}_{bspl}\right]_c\,\hat{P}_c(z)\,\boldsymbol{Y}_r(\theta,\phi)\,\sin(\theta)\,d\theta\,d\phi\,.$$

We compute the elements of these matrices using symbolic integration in Maple. In the case of order-4 SH and order-4 NLP, $\mathbb{M}_{lin}$ has dimension 16×8 and $\mathbb{M}_{bspl}$ has dimension 16×32.