Stupid Spherical Harmonics (SH) Tricks

Peter-Pike Sloan
Microsoft Corporation

Abstract

This paper is a companion to a GDC 2008 Lecture with the same title. It provides a brief overview of spherical harmonics (SH) and discusses several ways they can be used in interactive graphics and problems that might arise. In particular it focuses on the following issues: How to evaluate lighting models efficiently using SH, what "ringing" is and what you can do about it, efficient evaluation of SH products and where they might be used. The most up to date version is available on the web at http://www.ppsloan.org/publications

Introduction

Harmonic functions [2], the solutions to Laplace's equation, are used extensively in various fields. Spherical Harmonics are the solutions when restricted to the sphere[1]. They have been used to solve potential problems in physics, such as the heat equation (modeling the variation of temperature over time [5][25]), and the gravitational and electric fields[9]. They have also been used in quantum chemistry and physics to model the electron configuration in atoms and model quantum angular momentum [16][51]. Closer to graphics they have been used to model scattering phenomena [7][17]. In computer graphics they have been extensively used, early uses were in modeling volumetric scattering effects [18], environmental reflections for micro-facet BRDF's without global shadows[6], non-diffuse off-line light transport simulations[40], BRDF representations [53], image relighting[28], image based rendering with controllable lighting [54][55], and modeling light source emission[8]. More recent examples include more work in atmospheric scattering [50] and computer vision [3].

The focus of this article is on techniques related to interactive rendering. The first paper that has been used extensively in games deals with using Spherical Harmonics to represent irradiance environment maps efficiently, allowing for interactive rendering of diffuse objects under distant illumination [35]. This was extended to handle a limited class of BRDF's with the same constraints [36]. Precomputed radiance transfer (PRT) [41][20][24] models a static object/scenes response to a lighting environment, often represented using SH, including complex global illumination effects like soft-shadows and inter-reflections with diffuse and simple glossy materials. This was extended to handle more general BRDF models [20][23] [42],

[1] They are the angular portion of the solution in spherical coordinates.

incorporate sub-surface scattering [42], significantly improve rendering efficiency through compression techniques from machine learning [42], and various techniques to model "local" texture like surface details [43][44][45]. SH have been used to model single scattering from distant lighting environments [49]. Other uses have been in using gradients to lift the distant lighting assumption [1], several techniques addressing dynamic objects [56][37] including support for inter-reflections [46][33], as a representation of visibility to model shadowing of objects with general BRDF models [12], using scaling operators to model shadows from deformable objects [52], as a parameterization of refraction [11] and as a technique to address the level of detail problem with normal maps [15].

More practical papers include covering implementation details for PRT [13], how to integrate these techniques into an engine [30], how to use SH+gradients for irradiance volumes [31][32], practical issues around projection and how to efficiently quantize SH coefficients [21] and a nice paper that projects an analytic skylight model [34] into SH and uses a global polynomial fit to evaluate the SH lightprobe as a function of the models parameters [14]. Numerical techniques for more robustly projection functions defined over the hemisphere using SH have also been investigated [22].

Many of the uses in real-time graphics are just as a convenient way for representing spherical functions – visibility, lighting and reflectance. While there are many other basis functions that can be used, wavelets [39], wavelets on cube maps [27], spherical radial basis functions [9], and others [26], spherical harmonics have some nice properties that will be described in this document. It is important to stress that there are scenarios where these other basis functions are more appropriate.

While spherical harmonics may seem somewhat daunting, they are actually straightforward. They are the spherical analog to the Fourier basis on the unit circle, and are easy to evaluate numerically. Like the Fourier basis used in signal processing, care has to be taken when truncating the series (which will always be done in video games), to minimize the "ringing" artifacts that can occur. This article will describe how to evaluate and represent lights efficiently using spherical harmonics, how to pull conventional lights out of a SH representation, describe "ringing" and mitigation techniques to minimize its impact, and go over products of functions using spherical harmonics, describing where they are useful and special cases that are worth optimizing for.

Background

Definition Spherical Harmonics define an orthonormal[2] basis over the sphere, **S**. Using the parameterization

$$s = (x, y, z) = (\sin\theta\cos\varphi\,, \sin\theta\sin\varphi\,, \cos\theta)$$

[2] An orthonormal basis is one that has the property $\int f_i f_j = \delta_{ij}$, that is one if *i==j*, and zero otherwise.

Where s are simply locations on the unit sphere. The basis functions are defined as

$$Y_l^m(\theta,\varphi) = K_l^m e^{im\varphi} P_l^{|m|}(\cos\theta), l \in \mathbf{N}, -l \le m \le l$$

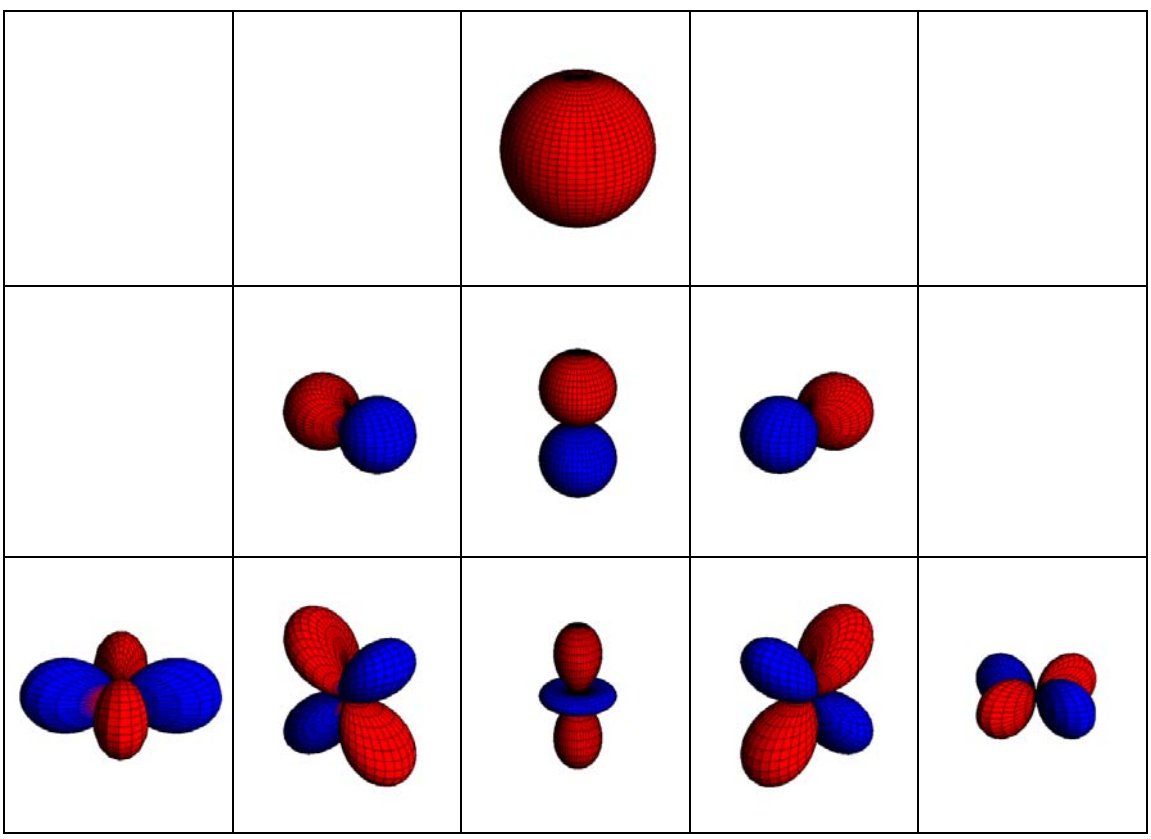
Where P_l^m are the associated Legendre polynomials and K_l^m are the normalization constants

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}$$

The above definition is for the complex form (most commonly used in the non-graphics literature), a real valued basis is given by the transformation

$$y_l^m = \begin{cases} \sqrt{2}\mathrm{Re}(Y_l^m) & m > 0 \\ \sqrt{2}\mathrm{Im}(Y_l^m) & m < 0 \\ Y_l^0 & m = 0 \end{cases} = \begin{cases} \sqrt{2}K_l^m \cos m\varphi \; P_l^m(\cos\theta) & m > 0 \\ \sqrt{2}K_l^m \sin|m|\varphi \; P_l^{|m|}(\cos\theta) & m < 0 \\ K_l^0 P_l^0(\cos\theta) & m = 0 \end{cases}$$

The index *l* represents the "band". Each band is equivalent to polynomials of that degree (so zero is just a constant function, 1 is linear, etc.) and there are 2*l*+1 functions in a given band. While spherical coordinates are convenient when computing integrals, they can also be represented using polynomials, as is commonly done when evaluating them (see Appendix A1 Recursive Rules for Evaluating SH Basis Functions and Appendix A2 Polynomial Forms of SH Basis for details.) An order n SH expansion uses all of the basis functions through degree n-1.



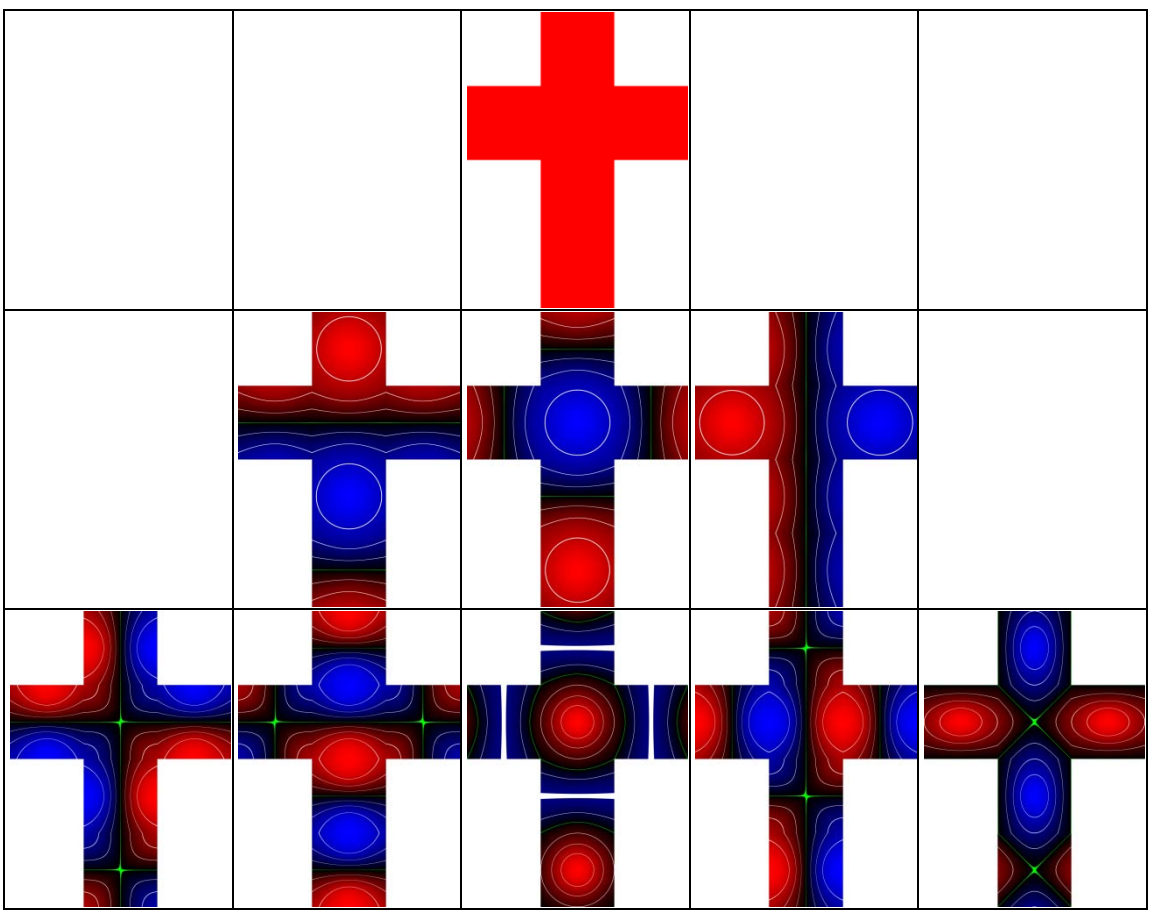
Spherical Harmonics can be visualized in a couple of ways. One standard way is to distort a unit sphere, by scaling each point radialy by the absolute value of the function and coloring it based on the sign (red for positive, blue for negative.) Above are images of the first three bands using this technique.

The functions in the central column (l=0) are called zonal harmonics (ZH) and will be discussed later, these functions have rotational symmetry around the z axis and the zeros (locations where the function is zero) are contours on the sphere parallel to the XY plane. The functions where (l=|m|) are called sectorial harmonics and the zeros define regions like apple slices.

An alternative visualization is to draw them using the parameterization of a cube map unfolded onto the plane. The unfolding of the cube map is as follows:

	+Y	
-X	-Z	+x
	-Y	
	+Z	

Here magnitude is encoded with color (red positive, blue negative, zero green) and iso-intensity contours have been evenly placed (white lines) to give more intuition for the gradient of the function (when they bunch together the function is changing faster, etc.)



Projection and Reconstruction Because the SH basis is orthonormal the least squares projection of a scalar function *f* defined over **S** is done by simply integrating the function you want to project, $f(s)$, against the basis functions (proof in Appendix A6 Least Squares Projection)

$$f_l^m = \int f(s) y_l^m(s)\, ds$$

These coefficients can be used to reconstruct an approximation of the function *f*

$$\tilde{f}(s) = \sum_{l=0}^{n} \sum_{m=-l}^{l} f_l^m y_l^m(s)$$

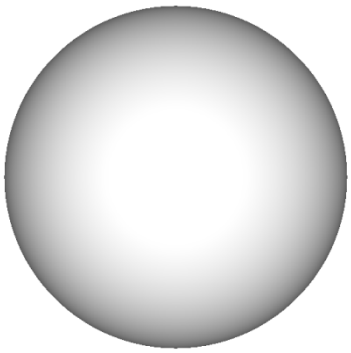
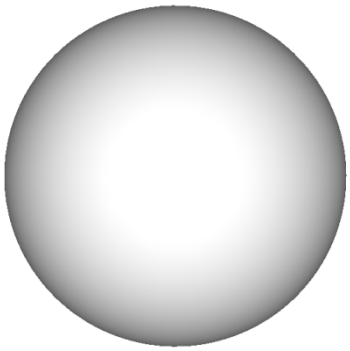
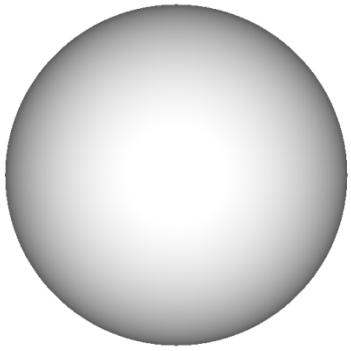
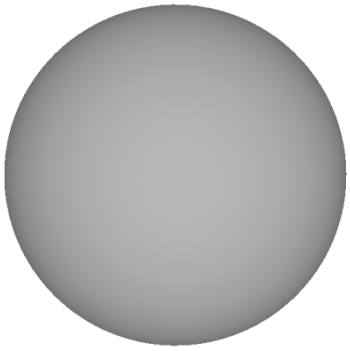
Which is increasingly accurate as the number of bands *n* increases. This paper concentrates on low-frequency approximations to *f*, for higher frequency representations other bases tend to do a better job. Projection to *n*-th order generates n^2 coefficients. It is often convenient to use a single index for both the projection coefficients and the basis function, via

$$\tilde{f}(s) = \sum_{i=0}^{n^2} f_i y_i(s)$$

Where $i=l(l+1)+m$. This formulation makes it clear that evaluating at direction s of the approximate function is simply a dot product between the n^2 coefficient vector f_i and the vector of evaluated basis functions $y_i(s)$. The first coefficient (f_0^0 or f_0 depending on indexing) represents the average value of the function over the sphere and will sometimes be referred to as the DC term.

Basic Properties One important property of SH is how projections interact with rotations. Given a function *g(s)*, which represents a function *f(s)* rotated by a rotation matrix *Q*, so *g(s) = f(Q(s))* the projection of *g* is identical to rotating $\tilde{f}$ and re-projecting it. This rotational invariance is similar to the translational invariance in the Fourier transform. This means that, for example, lighting will be stable under rotations, so there won't be any aliasing artifacts or "wobbling" of the light sources. Below are images of a sphere illuminated by a directional light source, the top row is using SH, the bottom row is using the Ambient Cube basis[3] from Valve [26]. The first column is a best case orientation, and the second is near a worst case one. The image is invariant using SH. This basis is discussed in more details in Appendix A9 Ambient Cube Basis. This will happen to some extent with any other basis defined over the sphere.

[3] This basis is more efficient to evaluate compared to SH, having 6 basis functions with only 3 being non-zero at any point on the sphere.

SH		
HL2		
	"good"	"bad"

Due to the orthonormality of the SH basis, given any two SH functions *a* and *b*, the integral of the product is simply the dot product of the coefficient vectors: $\int \tilde{a}(s)\tilde{b}(s) = \sum_{i=0}^{n^2} a_i b_i$

Convolution Given a kernel function *h*(*z*) that has circular symmetry, you can generate a new SH function that is the result of convolving the kernel with an original function *f*. *h* must have circular symmetry for the result of the convolution to also be represented on the sphere **S**, instead of the rotation group **SO***(3)*. The convolution can be done directly in the frequency domain using the following equation:

$$(h * f)_l^m = \sqrt{\frac{4\pi}{2l+1}} h_l^0 f_l^m$$

This amounts to simply scaling each band of *f* by the corresponding m=0 term from *h*.

Rotation As mentioned before, the SH are closed under rotation. SH rotation matrices are in a block structure, where each band is rotationally independent and has a dense (2l+1)x(2l+1) sub-matrix. There are several ways to compute these rotation matrices, for very small orders (quadratic and less) doing so symbolically is most efficient, but for higher orders it seems to be more efficient to decompose the rotation matrix into zyz Euler angles [19].

Zonal Harmonics Spherical harmonic projections of functions that have rotational symmetry around an axis are called Zonal Harmonics (ZH.) If they are oriented so that this axis is Z, the

zeros of the function will form lines of constant latitude and the functions only depend on θ. The coefficient vector in this orientation only has one non-zero per band, so a *n*-th order function has *n* instead of n^2 coefficients. Zonal Harmonics have been used to approximate transfer [44], and are the common representation of phase functions in scattering theory [7][17], they will be used extensively in this paper when modeling light sources. Rotation of Zonal Harmonics is simpler than general SH, it can be done with what is effectively a diagonal matrix and only requires evaluating the SH basis functions in the new direction *d*. Given the ZH coefficients of a function (only the m=0 terms from an SH projection) z_l it can be rotated to a new direction *d* using this equation:

$$f(s) = \sum_l z_l \sqrt{\frac{4\pi}{2l+1}} \sum_m y_l^m(d) y_l^m(s)$$

So the resulting SH coefficients are:

$$f_l^m = \sqrt{\frac{4\pi}{2l+1}} z_l y_l^m(d)$$

SH Products The kth coefficient of the product of two functions *f* and *g* represented using n^{th} order SH projected into SH has the following form:

$$p_k = \int y_k(s) \left(\sum_{i=0}^{n^2} f_i y_i(s) \right) \left(\sum_{j=0}^{n^2} g_j y_j(s) \right) ds = \sum_{ij} \Gamma_{ijk} f_i g_j$$

Where Γ is the triple product tensor:

$$\Gamma_{ijk} = \int y_i(s)\, y_j(s) y_k(s) ds$$

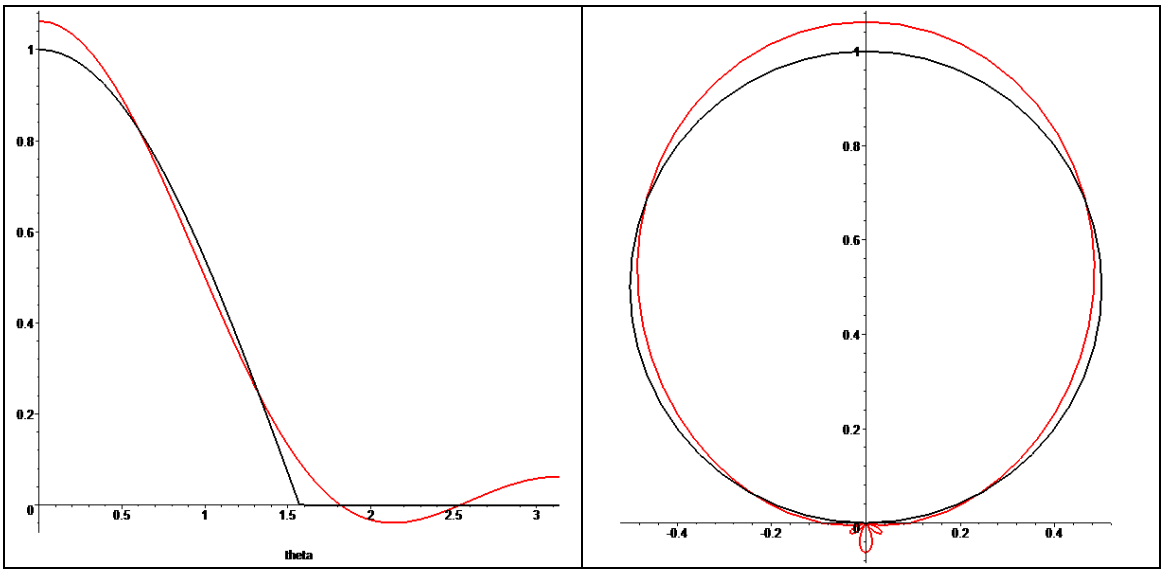
an order 3 sparse symmetric tensor. Since SH are polynomials, the polynomial product will have maximal degree 2*n*-2, which means it will have non-zero coefficients through order 2*n*-1. This becomes unwieldy as the number of functions being multiplied grows, so it is common to truncate the product early [56][37].The number of non-zero coefficients as a function of *n* is quite large [47][37] so care has to be taken when generating efficient code. One special case that is useful to point out is that if the function *f* is fixed (ie: distant lighting) you can compute a "product matrix" $\mathrm{M_f}$ which will significantly reduce the cost. This matrix is symmetric and built using the following equation:

$$(\mathrm{M_f})_{\mathrm{ij}} = \sum_{\mathrm{k}} \Gamma_{ijk} f_k$$

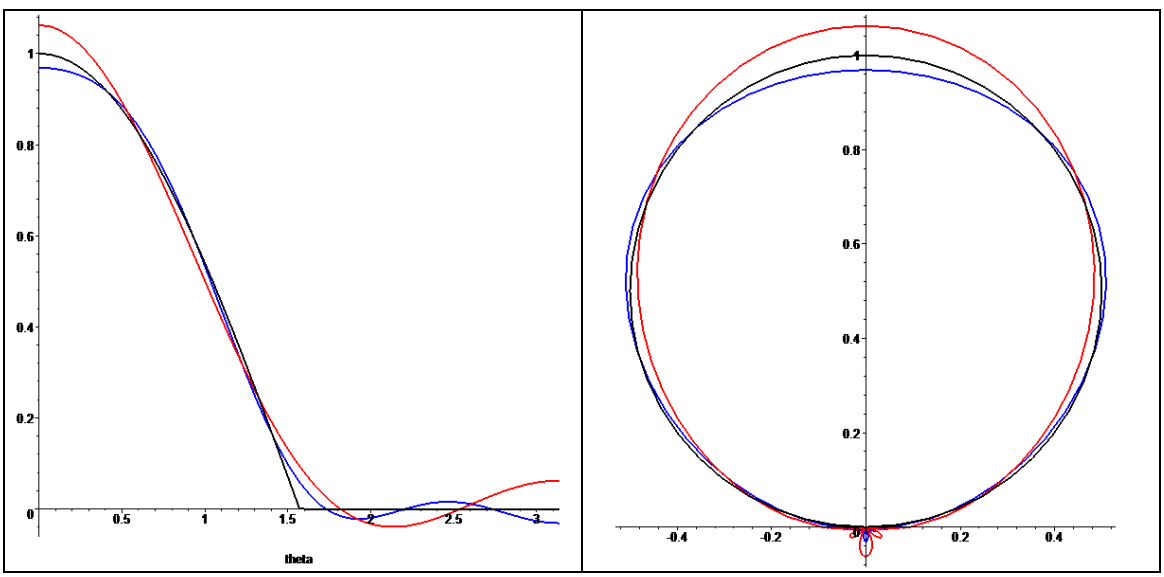
Computing the product with a function *g* in this case is simply a matrix vector product.

Irradiance Environment Maps

An irradiance environment map is created by convolving a light probe with a clamped cosine function; this should be normalized by dividing by π to display radiance. This convolution can be done efficiently using SH [35], and is accurate enough to be efficiently rendered directly from SH as well. Order 3 SH do a good job approximating this kernel, but if HDR light sources are going to be used you might want to consider using Order 5 (the order 4 ZH coefficient is zero so that band can be skipped.) Below are images of the clamped cosine kernel and the order 3 SH approximation, the red curve is the SH approximation, the figure on the left is a plot as a function of theta, on the right a polar plot scaled by the absolute value of the function:



Below are plots that also includes the Order 5 projection (blue):



The Order 3 SH approximation over estimates by 1/16th at theta=0 (north pole) and has a spurious lobe at the south pole with magnitude of 1/16th. A directional light source that would reflect a value of 17 when a normal points at it would reflect a value of 1 pointing in the opposite direction (should be reflecting 0.) The order 5 approximation has a negative lobe, which would reflect -1 with a directional light that would reflect 31 with a normal pointing right at it. While these approximations are accurate, the approximation can cause error, particularly if very bright light sources are being used.

Appendix A10 Shader/CPU code for Irradiance Environment Maps contains shader and CPU code for efficiently evaluating irradiance environment maps.

Lighting Models

There are various ways of representing lighting in SH. The simplest is to just project from a cube map, but there are also analytic models that are inexpensive to evaluate and potentially useful to expose to artists. A recent paper [14] does a nice job of projecting a practical skylight model [34] into SH and fits a global polynomial of the SH coefficients over the parameter space of the model.

Projection from Cube Maps

To project from a cube map you simply need to integrate the SH basis functions against the cube map. This can be done numerically by evaluate the SH basis functions in the direction of each texel center, weight it by the differential solid angle for that texel and normalize the results. In pseudo-code:

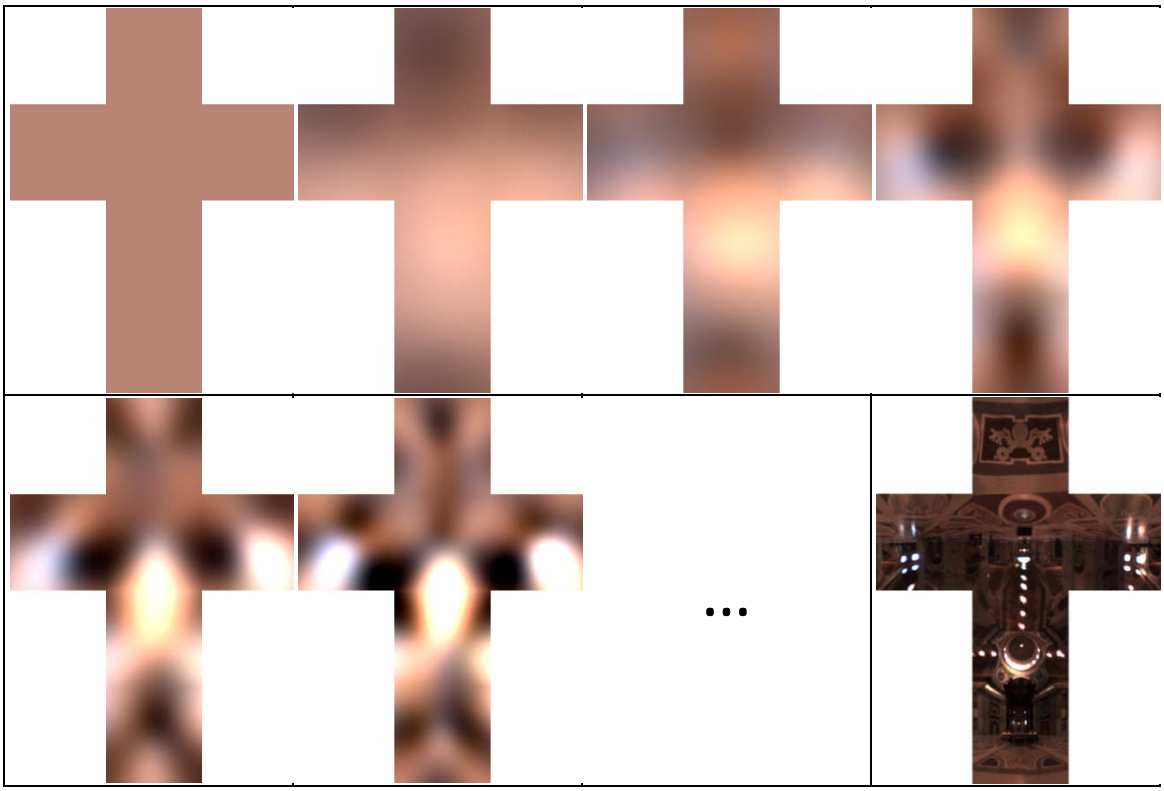
```
float f[],s[];
float fWtSum=0;
Foreach(cube map face)
      Foreach(texel)
            float fTmp = 1 + u^2+v^2;
            float fWt = 4/(sqrt(fTmp)*fTmp);
            EvalSHBasis(texel,s);
            f += t(texel)*fWt*s; // vector
            fWtSum += fWt;
f *= 4*Pi/fWtSum; // area of sphere
```

In the code above u and v represent the two coordinates on the given face that are not +/- 1[4], t(texel) is the texel color. EvalSHBasis needs to normalize the input coordinate (on a cube face), and simply evaluate the SH basis functions in that direction. The last normalization can be omitted (instead you can just divide by the number of samples), since the normalized sum of the

[4] Ie: on the +X face, it would be y/z, etc.

differential solid angles should be 4*Pi, but it tends to be a little off (particularly when using low resolution cube maps.)

Below are images of the reconstruction of a HDR light probe into order 1 to 6 Spherical Harmonics. The final image is the light probe that was projected.



Analytic Models

Directional lights are trivial to compute, you simply evaluate the SH basis functions in the given direction and scale appropriately (see Normalization section.) Spherical Light sources can be efficiently evaluated using zonal harmonics. Below is a diagram showing an example scene, we want to compute the incident radiance, a spherical function, at the receiver point P. Given a spherical light source with center C, radius r, what is the radiance arriving at a point P d units away? The sin of the half-angle subtended by the light source is r/d, so you just need to compute a light source that subtends an appropriate part of the sphere. The ZH coefficients can be computed in closed form as a function of this angle: $z_l = \int_{\theta=0}^{a} \int_{\varphi=0}^{2\pi} y_l^0 \, d\varphi d\theta$ where a is the half-angle subtended. See Appendix A3 ZH Coefficients for Spherical Light Source for the expressions through order 6.

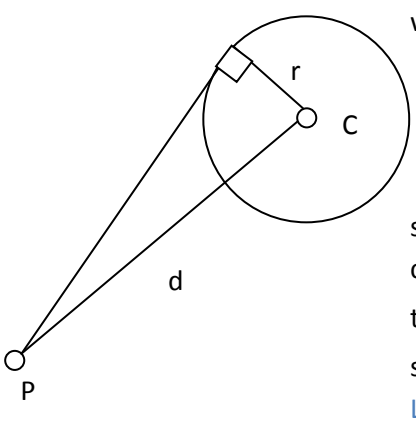
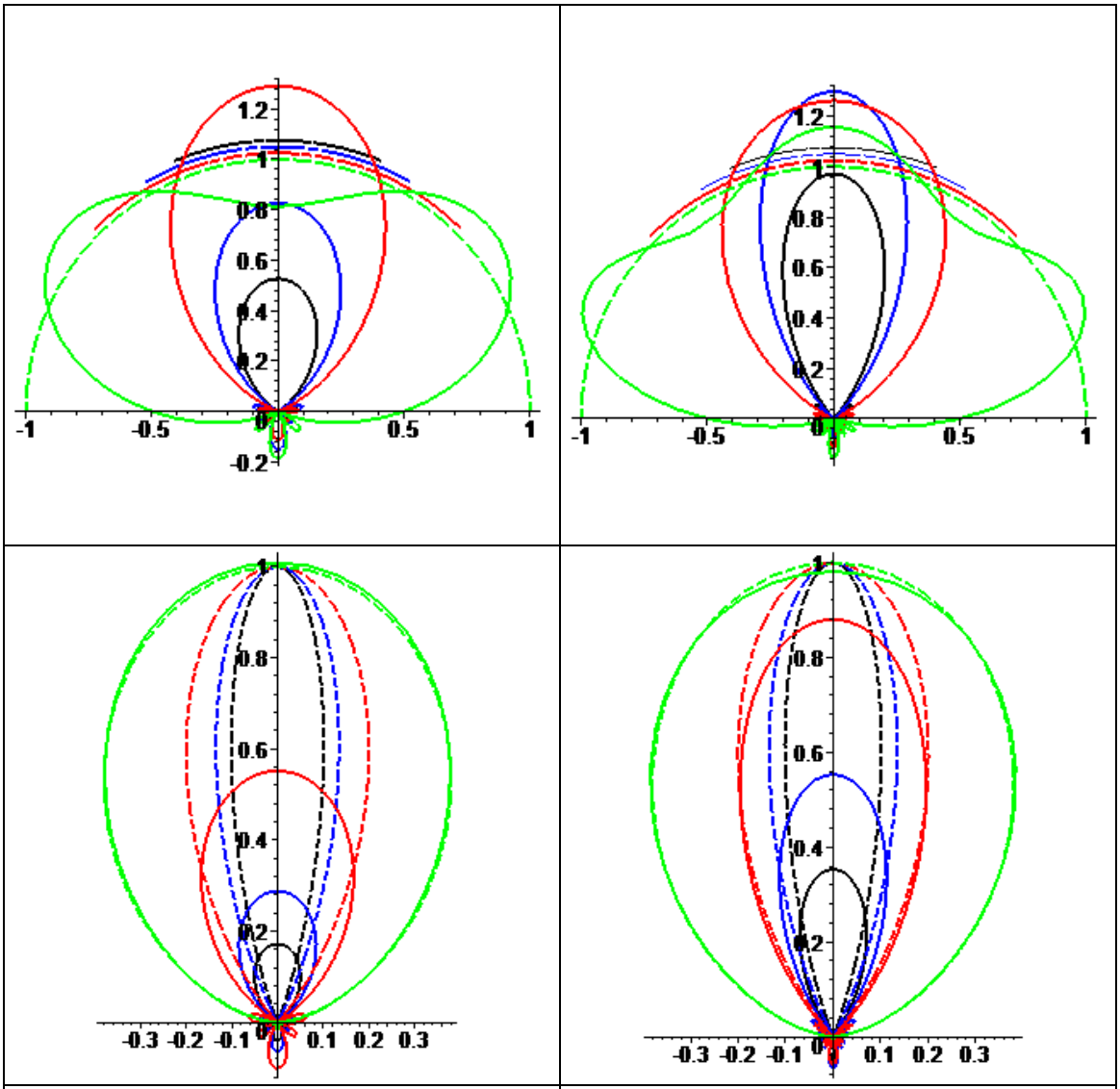


Figure 1 Spherical Light Source

The spherical technique can also be used to model a cone (think of it as a disc at infinity) with constant emission. A softer cone can be modeled that has a smooth fall-off over the visible portion[5] – see Appendix A4 ZH Coefficients for Smooth Cone for the equations.

The columns are for orders 4 and 6, the top row is the projection of a cone and the bottom row is a cone that has a smooth fall off. The angles are 90 (green), 45 (red), 30 (blue), 12.5 (black) degrees. The dashed lines represent the actual function (slightly displaced for the cone so they don't overlap), the solid lines are the SH approximation. In general the soft cone is better behaved. How to address the artifacts that arise in projection is the subject of the section on Ringing below.



Order 4	Order 6

[5] An "ease spline" is used, this is a cubic polynomial *f(x)* that has the constraints *f(0)=1,f'(0)=0,f(a)=0,f'(a)=0*, the unique cubic polynomial is: $f(x) = {2x^3}/{a^3} - {3x^2}/{a^2} + 1$

Normalization

If [0,1] lighting is being used, it is convenient to normalize the radiance vector so that the reflected radiance for an unshadowed receiver with a normal pointing directly at the light source would be 1.0. Mathematically you want to compute a scale factor c, that when multiplied by the lighting vector L will result in unit reflected radiance when integrated against a vector T that represents an unoccluded clamped cosine lobe (projection of normalized clamped cosine into SH.) So you have:

$$1 = \int cL(s)\,T(s)ds \xrightarrow{yields} c = \frac{1}{dot(L,V)}$$

Only the bands that are going to be used for rendering should be used when computing the normalization factor. Aligning T with +Z gives a simple analytic formula; here are the coefficients of T for the first 6 bands:

$$\frac{1}{2\sqrt{\pi}}, \frac{\sqrt{3}}{3\sqrt{\pi}}, \frac{\sqrt{5}}{8\sqrt{\pi}}, 0, \frac{-1}{16\sqrt{\pi}}, 0$$

For analytic lights you can use an analytic normalization term, for the cone light of angle *a* that would be:

$$\frac{1}{\sin^2 a}$$

However the resulting lights will not reflect unit radiance, because the projection error of both the clamped cosine function and the light source will not be taken into account. For directional lights[6] the normalization factor is $\frac{16\pi}{17}$ and for "ambient" lights it is $2\sqrt{\pi}$.

Extracting Conventional Lights from SH

Given an SH lighting vector, it is possible to approximate it as a single directional light source and an ambient light source. This has been used on hardware that does not support vertex shaders. Mathematically we want to compute the intensity of a directional light (*c*) and the intensity of an ambient light source (*a*), so that squared error of reflected radiance is minimized for any surface normal (*N*). Assuming a fixed direction (*d*) for the light source, the error function we want to minimize is:

$$E(c,a) = \int \left(cH_N(d) + a - \int L_e(s)H_N(s)\,ds\right)^2 dN$$

Where $H_{N(x)} = max(dot(N,x)/\pi\,,0)$ is the normalized clamped cosine. If the lighting $L_e(s)$ is expressed in SH, this has a simple solution. The irradiance environment map represented by the new lighting environment should be as close to the input irradiance environment map –

[6] This is assuming no higher than 4th order lighting, for 5th or 6th order the normalization factor is $\frac{32\pi}{31}$

minimizing squared error of these two environment maps is equivalent to minimizing reflected radiance over all normals:

$$E(c,a) = \int \left(cL_d(s) * H_N(s) + aL_a(s) * H_N(s) - L_e(s) * H_N(s)\right)^2 ds$$

Here $L_d(s)$ is a normalized SH directional light in direction *d*, and $L_a(s)$ is a normalized SH constant light (just depends on DC.) The optimal values *a* and *c* are:

$$c = \frac{867}{316\pi}\, dot_0\left(\hat{L}_d, \hat{L}_e\right)$$
$$a = \left(\hat{L}_e[0] - c\frac{8\sqrt{\pi}}{17}\right)\frac{\sqrt{\pi}}{2}$$

The lighting vectors above are all turned into irradiance environment maps through convolving with the normalized clamped cosine kernel. The dot product above ignores the DC term and $\hat{L}_e[0]$ is the DC term of the lighting environment. The above assumes a known direction, a good candidate direction is the "optimal linear" direction [44], formed by normalizing the vector $(\widehat{-L}_e[3], -\hat{L}_e[1], \hat{L}_e[2])$, which are the linear coefficients of the lighting environment.

Extracting Multiple Lights

It is also possible to extract multiple lights from a SH light probe. This might be done to combat ringing (analytic lights won't have negative lobes), to model glossy reflection (just from the light sources pulled out in this manner) or to use a small number of shadow zbuffers (pull the lights out for both diffuse and glossy parts of the BRDF.) The optimal linear direction works well when a light probe is dominated by a single light source, when this isn't the case the directions and intensities of the lights should be optimized for in some fashion. One thing to do is to climb "up hill" to find local maxima of the function. Given how smooth SH are, for a given order there is a finite distance where any point that distance from a distinct peak is guaranteed to reach the point using gradient ascent. A set of points can be generated with the property that the point on the sphere farthest from the center of its Voronoi[7] cell is less then this distance. If you start your search from each of these points you should find all of the point local maxima. These distances can be found by looking at the projection of a delta function and computing the angular distance between peaks and the zeros. Using a conservative estimate of two thirds this radius the number of points needed per order are {1,3,6,10,15,22} for the first 6 orders.

This point set can be computed by computing a simulation: All points have a force with a certain fall-off (say $1/d^2$ where d is the Euclidean distance between them) that acts on every other point. Generate the net force acting on each point, and subtract off the normal component (so it is just a tangential force), move the points by this force with a weight w. If the net sum of

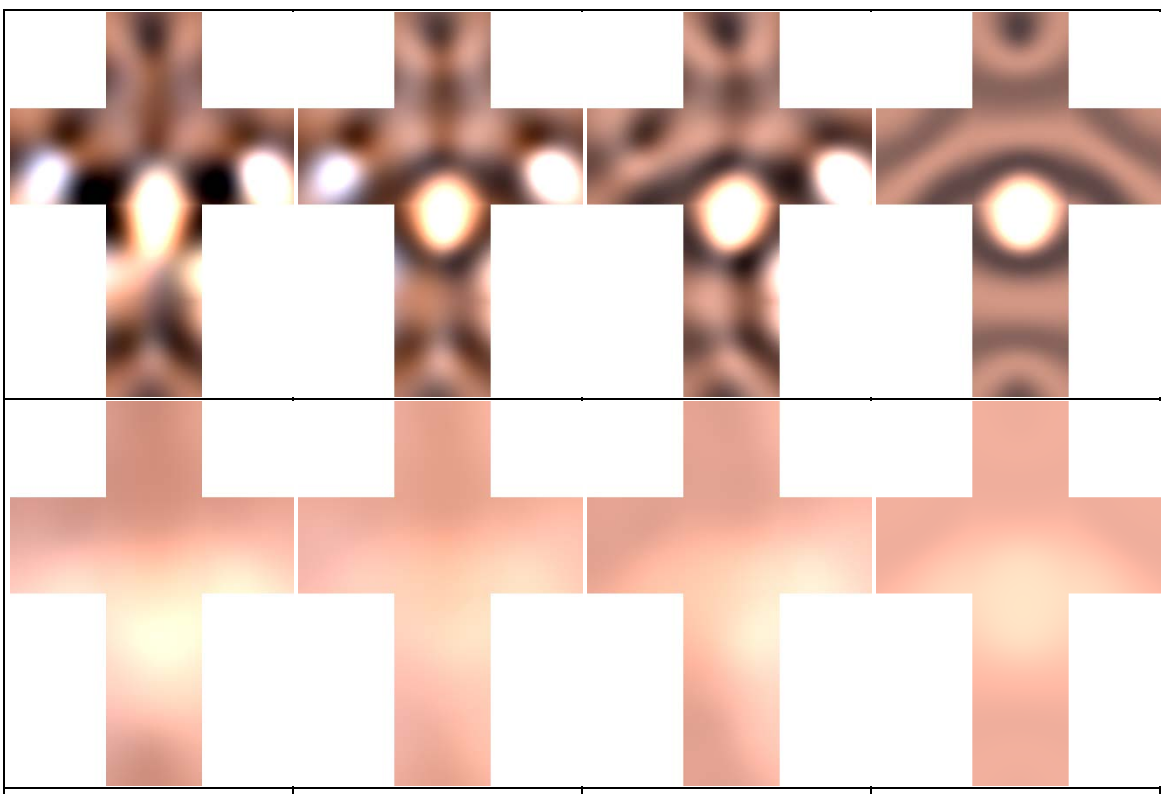
[7] Given a set of points X on the sphere, the Voronoi cell of a given point x_i consists of all the points on the sphere that are closer to x_i than any other point in the set.

forces increases, decrease w by half and try again, if it decreases double w and try again. This is solving for a set of electrons that minimize the sum of the electrostatic charges..

Given a SH lighting environment L(s), you want to find a point on the sphere that is a local maximum, which is the same as finding a local minima of –L(s). This is a non-linear optimization problem, a small number of BFGS iterations [29] converge on the peaks in my experience. The gradients of the basis functions need to be computed when using optimization methods. For a point on the sphere differentiating the polynomials is trivial, but you need to allow the point to go off the sphere when doing a line search, so you want to use symbolic inputs that normalize the coefficients (ie: $f\left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}}\right)$ in place of $f(x,y,z)$). This is done by simply computing the gradient at the normalized location, and multiplying it by the Jacobian of the normalization function:

$$\begin{bmatrix} \frac{L^2-x^2}{L^3} & \frac{-xy}{L} & \frac{-xz}{L} \\ \frac{-xy}{L} & \frac{L^2-y^2}{L^3} & \frac{-yz}{L} \\ \frac{-xz}{L} & \frac{-yz}{L} & \frac{L^2-z^2}{L^3} \end{bmatrix}$$

Where L is $\sqrt{x^2+y^2+z^2}$.



Original	3 direction	2 directions	1 directions

These images compare radiance (top row) and irradiance (bottom row) lighting environments when approximating with 3, 2 or 1 directional lights and a ambient light. Given the N most significant peaks (in terms of magnitude) this solves for directional and ambient intensities in a least squares sense.

The equations to compute the light intensities for two lobes are:

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} \frac{A}{A^2 - B^2} & \frac{-B}{A^2 - B^2} \\ \frac{-B}{A^2 - B^2} & \frac{A}{A^2 - B^2} \end{bmatrix} \begin{bmatrix} L_e \circ L_{d0} \\ L_e \circ L_{d0} \end{bmatrix}$$

Where:

$$A = L_{d0} \circ L_{d0}, B = L_{d0} \circ L_{d1}$$

The ambient term is then:

$$a = \left(L_e[0] - (c_0 L_{d0}[0] + c_1 L_{d1}[0])\right) \frac{\sqrt{\pi}}{2}$$

For three lights the intensities are:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \frac{A^2 - D^2}{E} & \frac{CD - AB}{E} & \frac{BD - AC}{E} \\ \frac{CD - AB}{E} & \frac{A^2 - C^2}{E} & \frac{BC - AD}{E} \\ \frac{BD - AC}{E} & \frac{BC - AD}{E} & \frac{A^2 - B^2}{E} \end{bmatrix} \begin{bmatrix} L_e \circ L_{d0} \\ L_e \circ L_{d1} \\ L_e \circ L_{d2} \end{bmatrix}$$

Where:

$$C = L_{d0} \circ L_{d2}, D = L_{d1} \circ L_{d2,} E = 2BCD + A(A^2 - B^2 - C^2 - D^2)$$

This matrix is symmetric, and the ambient coefficient is:

$$a = \left(L_e[0] - (c_0 L_{d0}[0] + c_1 L_{d1}[0] + c_2 L_{d2}[0])\right) \frac{\sqrt{\pi}}{2}$$

The derivations of the equations are in Appendix A5 Solving for Coefficients to Approximate SH Environment Map with Directional and Ambient Lights.

Adding all of the degrees of freedom (directions and intensities) and using a non-linear solver would generate higher quality results, possibly using the technique above as an initial guess.

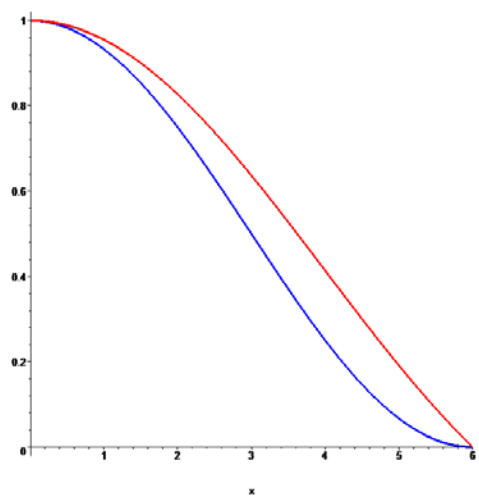
Ringing

Ringing, also called Gibbs Phenomenon, is a common problem in signal processing. When a signal with a discontinuity is projected into a finite Fourier basis (which can only represent continuous functions) overshoot and undershoot will happen around the discontinuity. Functions that don't have discontinuities can exhibit similar behavior if the projection is truncated. We have already seen these problems when looking at lighting models, and in representing irradiance environment maps (projection of clamped cosine function.) A similar problem occurs in surface design, where when trying to satisfy a set of geometric constraints unwanted oscillations can occur. There are two general solutions to these problems:

1) Windowing the truncated projection coefficients using sigma factors. This is the most common solution in signal processing, and can be trivially used with Spherical Harmonics [8][41][37].
2) Minimizing some form of variational function (minimizing a measure of curvature for example), instead of just the standard least-squares error. This is commonly done in computer aided geometric design, but also can be efficiently done using Spherical Harmonics [38].

Windowing[8]

One way to minimize the ringing artifacts is to multiply in the frequency domain (which is a convolution in the spatial domain) by a kernel with projection coefficients that taper to zero as you approach the cut-off frequency. If this function is a sinc[9] that is stretched out so that it reaches zero at the truncated frequency band, it is called using Lanczos sigma factors[10]. Intuitively what this is doing (in 1D) is convolving in the spatial domain with a tight box function, making the function smooth enough to be represented without excessive ringing. There are more sophisticated ways to attack Gibbs phenomena [10][4] but they use the SH coefficients to generate piecewise analytic functions that would not be as convenient for games.



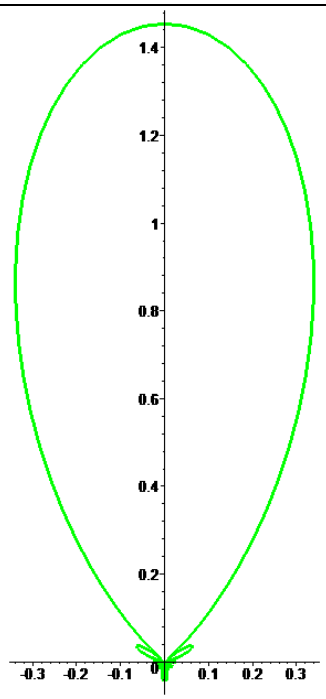
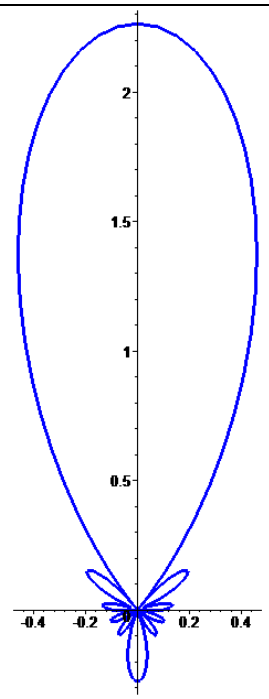
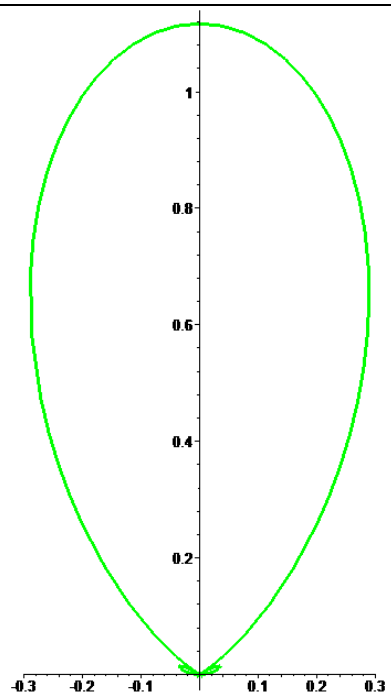
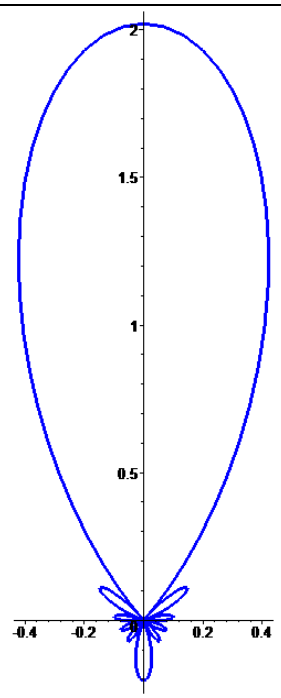
In our experience the choice of windowing function is not as important as having flexibility to trade off between ringing and blurring. The image to the left shows the two windowing functions (red is sinc, blue is raised cosine lobe – called a Hanning window)

[8] The more common use of the term "windowing" in signal processing is used in the spatial domain – when filtering images the spatial extent of the filter is "windowed", and when taking a FFT of a 1D signal the signal might be scaled to make it periodic. In this paper it is being done in the frequency domain.

[9] $\mathrm{sinc}(x) = \frac{\sin x}{x}$ where the limit is 1.0 when x=0.

[10] Sometimes the sigma factors will be raised to a power to more aggressively reduce the Gibbs phenomena – which is the same as repeatedly convolving the signal.

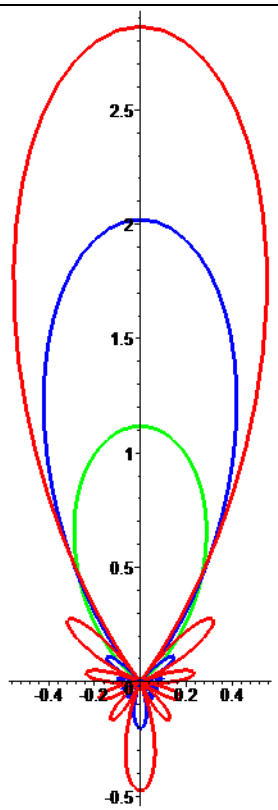
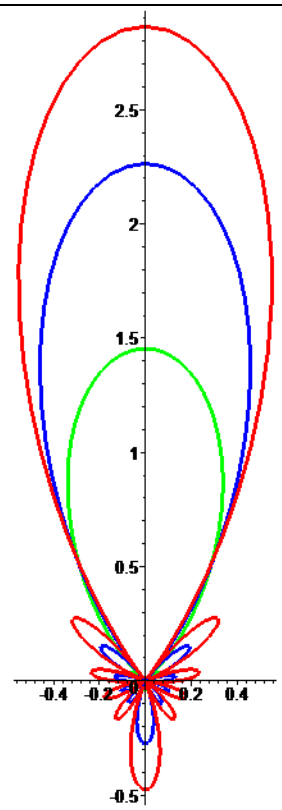
scaled for 6th order SH (so they reach zero for the 7th order band, the last value used would be evaluated at 5 and the functions would only be evaluated at the integers.) The Hanning function decays faster than the Lancosz function, which makes it blur more aggressively[11].

Equation	W=6	W=10
$\frac{\sin\frac{\pi x}{w}}{\frac{\pi x}{w}}$		
$\frac{\left(1+\cos\frac{\pi x}{w}\right)}{2}$		

[11] The Lancosz squared is close to the Hanning function, but decays a bit faster. For the reconstruction of the delta function to be a height of 2.25, the Hanning function needs a window of size 12.0105 and the Lancosz needs one of size 9.8725. The results look visually indistinguishable.

Above we show the results using Lanczos and Hanning sigma factors. The signal being projected is an order 6 delta function, which is the "peakiest" signal you can project into SH and exhibits ringing artifacts. The projection of a delta function is a ZH, so we are showing a cross section of the sphere, phi is fixed. Radial magnitude is plotted, the sign of the lobes alternate.

Looking at all the graphs together (red is the raw projection of the delta function) you can see how windowing blurs the signal while eliminating the rings (visible near the origin in the figures.)

Hanning	Lancosz
	

Minimizing a Functional

An alternative approach is to try and minimize some function besides squared approximation error. One way to do this is to satisfy a set of constraints (for example exact reconstruction at a small number of points) and then use the "slack" variables left over (assuming there are enough degrees of freedom) to minimize some error functional[38]. Given the low SH order often used in games/graphics, this approach does not seem that practical so I will not spend any more time on it. An alternative is to attempt to minimize a norm that penalizes large oscillations. This can be done with Spherical Harmonics in a straightforward fashion. The Laplace operator, or Laplacian, is the divergence of the gradient of a scalar function; equivalently it is the sum of unmixed partial derivatives

$$\Delta f = \frac{df^2}{dx^2} + \frac{df^2}{dy^2} + \frac{df^2}{dz^2}$$

In spherical coordinates on the unit sphere this is

$$\Delta f = \frac{1}{\sin\theta}\frac{d}{d\theta}\left(\sin\theta\frac{df}{d\theta}\right) + \frac{1}{\sin^2\theta}\frac{d^2 f}{d\phi^2}$$

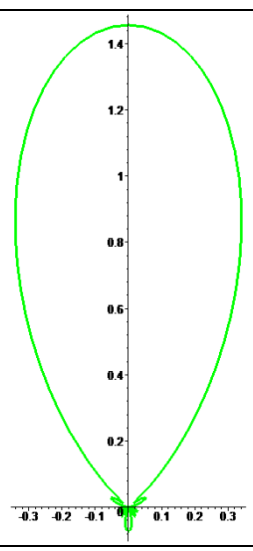
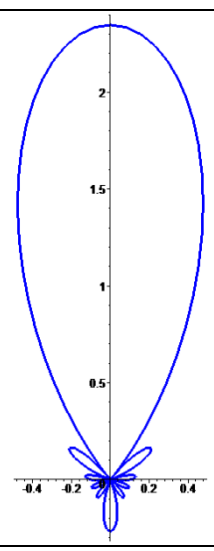
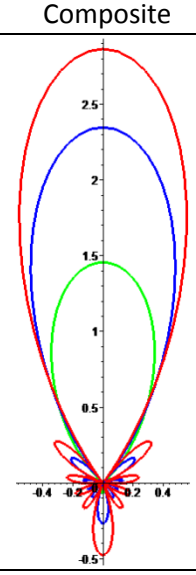
The integral of the squared Laplacian is a curvature measure used on the sphere [38]. The function we are going to minimize is:

$$E(c) = \int \left(\tilde{f}(s) - f(s)\right)^2 ds + \lambda \int \left(\Delta\tilde{f}(s)\right)^2 ds$$

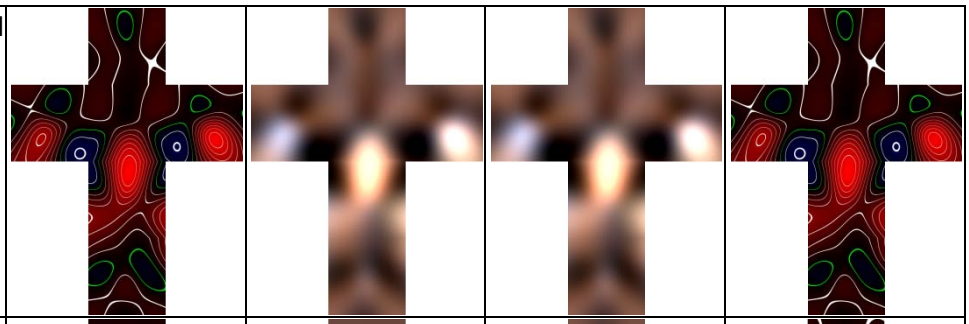
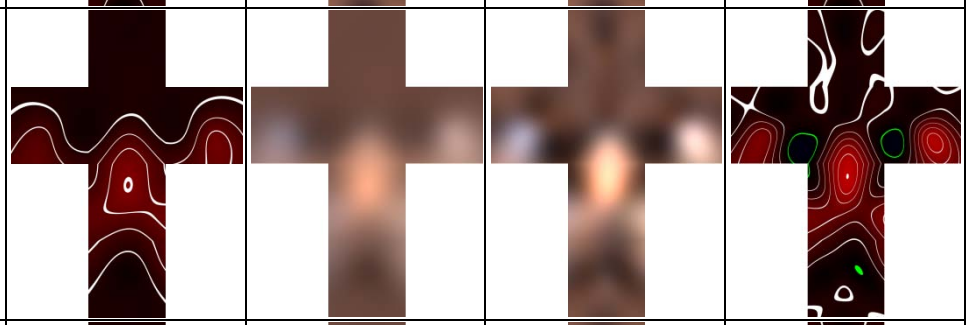
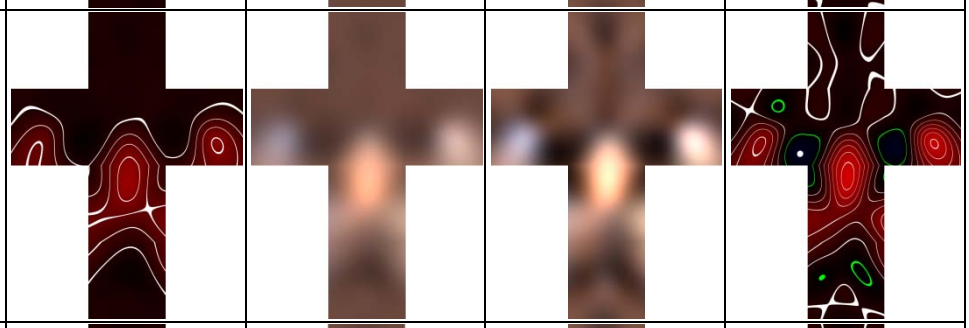
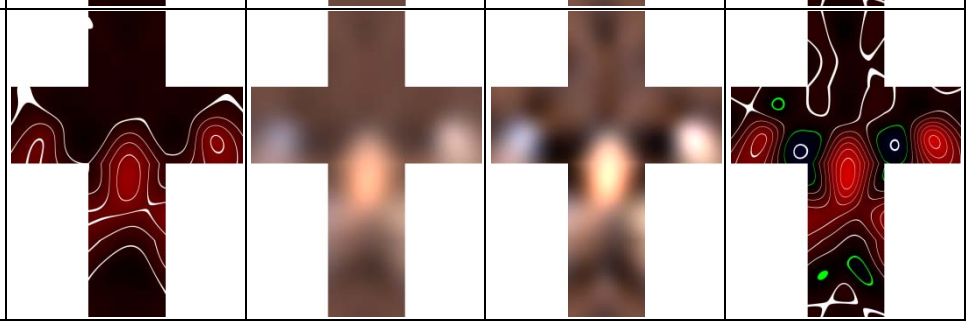
We will tweak this a little bit, we already know the raw projection coefficients f_l^m, so we want to find new coefficients c_l^m that are as close to the least squares result as possible, while also minimizing the weighted squared Laplacian. This can be done in closed form [see Appendix A6 Least Squares Projection], which given a λ results in the following coefficients.

$$c_l^m = \frac{f_l^m}{(1 + \lambda l^2 (l+1)^2)}$$

Note that this amounts to a windowing function that is dependent on λ . When λ is zero, you get the least squares coefficients, when λ is infinity you get just the DC term, which has zero curvature. One approach to choosing λ is to solve that it reduces the squared Lalpacian by a fixed amount, say one half. This can be done using any standard root finding technique. See [Appendix A7 Solving for Lamba to Reduce the Squared Laplacian] for an explanation of how to do this using Newtons method. Below are the results with a 6ht order delta function solving for the squared Laplacian to be 10% ($\lambda = .004209$ green) and 50% ($\lambda = .000632$ blue) of the original. The final plot has them along with the delta function itself:

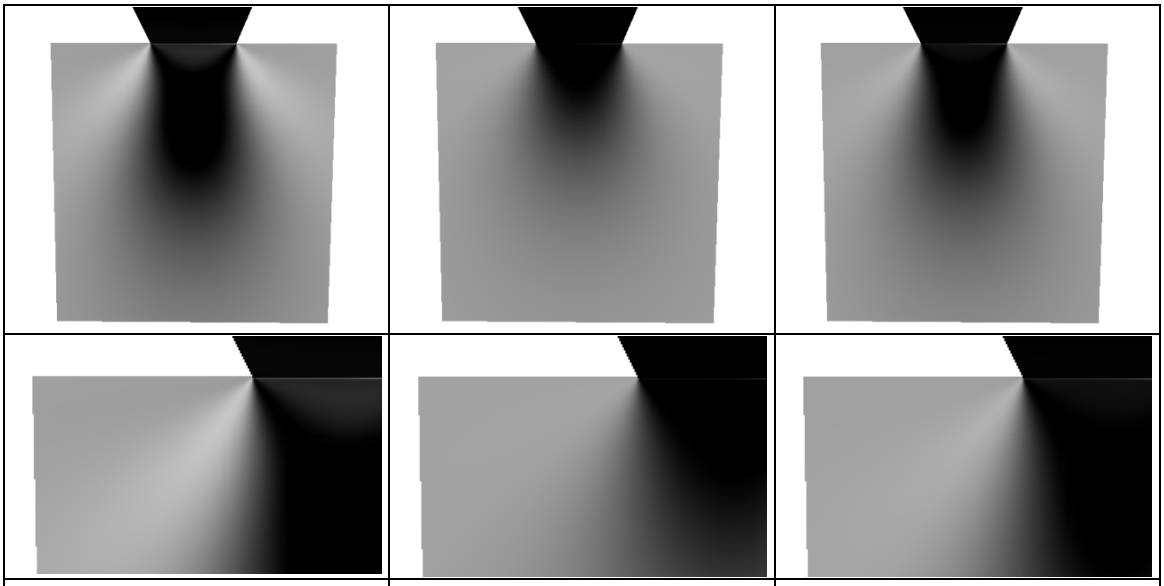
Δ^2=0.1	Δ^2=0.5	Composite
		

Here are images using an actual lighting environment. The first column of images are contour plots (blue is negative) of the second column of images - windowing with a broad (smoother) window. The forth column of images are contour plots of the third column – smoothing with a narrower (less blurry) filter. The top row is the original image – all results are shown at 6th order:

Original				
Cos				
Sinc				
Δ^2				

Windowing should be used judiciously. When using irradiance environment maps, the convolution with the clamped cosine function aggressively attenuates high frequencies and

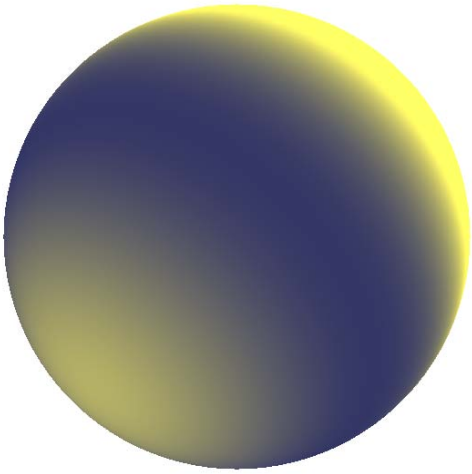
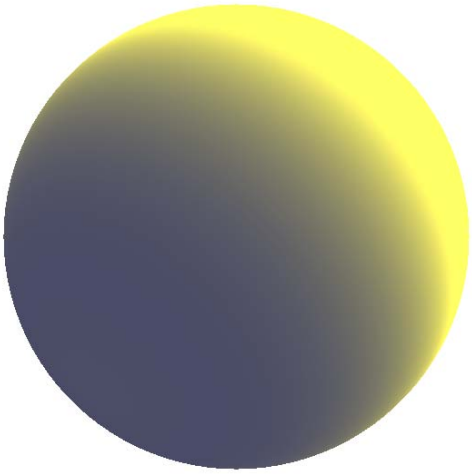
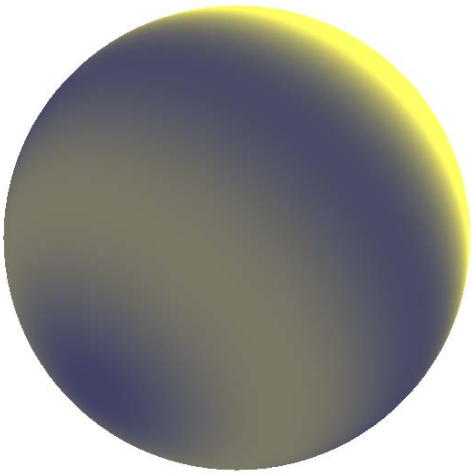
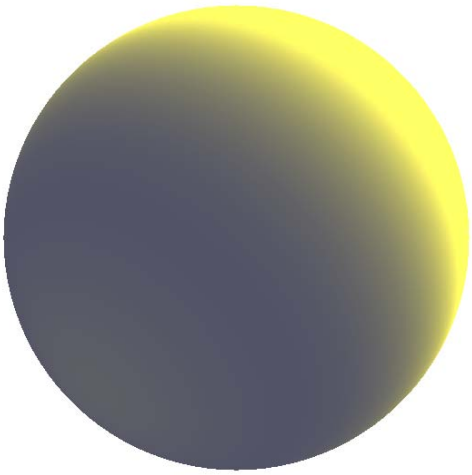
windowing is rarely necessary. Also scenes where there is a lot of normal variation tend to not show off the ringing artifacts as much as ones where the normals of the receiver are smoother. Below are images of a simple scene ("door" and a ground plane) that show how windowing affects the shaded result. An area where ringing occurs is highlighted in the second row.



No Windowing	Hanning 6	Hanning 10

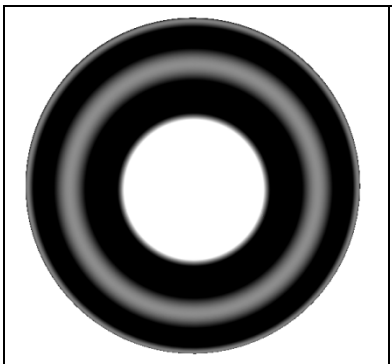
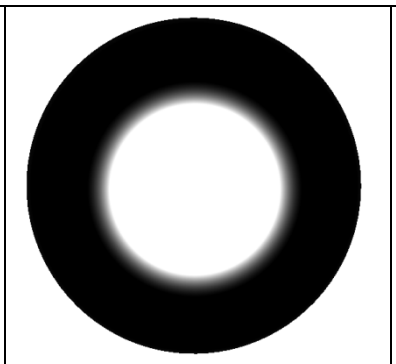
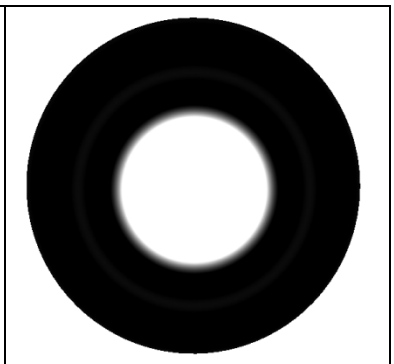
When using HDR, or just moderately bright lights, ringing is a more severe problem and even irradiance environment maps might need some amount of windowing

Another problem that can happen with ringing is color artifacts. In the images below we have a sphere illuminated with a bright directional yellow light (almost rim lit from the upper right, order 6) and a moderate ambient white light. The top row uses irradiance environment maps, the bottom row uses order 6 PRT (so a more accurate approximation of the cosine kernel is used.) The first column is without windowing, the second column uses a Hanning window of order 4 on the top row and order 5 on the bottom. The unwindowed version shows both the positive lobe (order 3, negative order 6) and the blue bands are from the negative rings of the directional light (remove red/green from the ambient light, but leave the blue alone.)

Order 3		
Order 6		
	No Windowing	Hanning

Content Sensitive Windowing

Lighting can be globally windowed, but you can also window depending on the influence the rings will have on the final shaded image. An example is illustrated below. We are rendering a somewhat matte (but not diffuse – phong with power 10) ball using a bright directional light source (the highlight is saturating.) If no windowing is applied, the principle highlight is sharper, but artifacts from the rings are clear. If the lighting is windowed, the rings disappear but the principle highlight is blurred as well. Instead you can look at the angle between the reflection vector and the dominant light direction, if this is small, no windowing is necessary, as it increases you can blend between the windowed and non-windowed light source, preserving the sharp highlight while eliminating the ringing artifacts. This sequence of images is shown below:

		
6th order reflection	Windowed (Hanning 7)	Content sensitive windowing

The equation used to control the blending is:

$$w_a = \left(max\left(0, \frac{\left((n \circ r) - c_t\right)}{1 - c_t}\right)\right)^p$$

Where w_a is the weight of the un-windowed light source (windowed is $1 - w_a$), c_t is a threshold that determines when you completely use the windowed light source and p controls the transition region of the blend. In this figure c_t is 0.07 and p is 0.8. You can experiment with these parameters, the thresholds will largely be based on the amount of windowing and the material properties.

If the light source is not a directional one, you can compute how well it can be approximated by a directional light source and take that into account when deciding how much to blend (the simplest way would be to compute a tensor product between how close the lighting environment is approximated by a directional light and how close the normal/reflection vector is to the dominant light vector.) For more complex shading, say for example PRT, the dominant transfer direction can be used, and how well it approximates the transfer function can be yet another factor in the tensor product (so you window when any one of the terms isn't approximated well.)

If the lighting is very dynamic, you will need to watch out for temporal artifacts that could occur as the lights change (ie: shadows/reflections would become sharper, etc.) For things like static SH light probes this technique should work well though.

SH Products

Computing the SH representation of the product of two functions represented using SH is often useful. Example scenarios are:

3) Punch a hole in a skylight model based on a large flying object (visibility times light), or a simple visibility model of the scene (large buildings, etc.)

4) Multiply visibility functions. This happens when doing dynamic approximate global illumination.
5) Scale or modify a SH light probe. Multiplying by a some constant between zero and 1 can be used to approximate clouds for example.

Computing a product in the frequency domain is quite complex, it boils down to applying a "triple product tensor" times the two SH vectors. Code for this can be generated efficiently [47] and will not be described in this paper. There are several special cases that are worth mentioning though.

Products with a Constant Function

If one of the SH functions is going to be used a lot you can build a dense matrix called the product matrix, this makes the triple product a simple matrix times vector product, which is significantly less expensive. An order 6 product would have 1296 multiplies instead of the 2527 in the code generated by [47].

Products with varying Orders

This is particularly common when the output order is lower, for example quadratic, so that you can represent a local radiance environment. Special casing the code for these cases can significantly decrease the complexity of the code. For example the product of two 6th order SH has 2527/1995 multiplies/adds when computing 6th order results, but only 933/676[12] when computing 3rd order results. Another example is simple ambient occlusion, in this case one of the terms is just DC, and so you simply have to scale the other vector by the DC value. Finally, one of the two functions might be lower order (ie: just multiply by linear visibility) which can also reduce the cost.

Product with Zonal Harmonics

If one of the functions is a Zonal Harmonic, you can rotate the other function into that same frame (less expensive rotation, only need 2 Euler angles due to the symmetry), compute the product and rotate back. The sparsity in the ZH frame eliminates a significant amount of work, which can increase performance. The product of two 6th order functions, one which is a ZH oriented in z, requires only 380/249 multiplies/adds. For arbitrary ZH the time to compute one million products (where one of the functions is always a in an arbitrary orientation) is around 1.2 seconds while the time to compute general 6th order products is just over 3 seconds, so this technique is almost 3 times faster.

Product with Analytic Function

If one of the functions has an analytic form, it is more accurate to analytically compute what amounts to the product matrix. An example is zeroing everything under the horizon (useful if there is a ground plane) or taking a product with a clamped cosine function. Doing this

[12] More efficient code could be generated, these results are using a naïve algorithm applied to the output of [47].

analytically is equivalent to having an infinite order expansion of the analytic function, and in general will be much faster than using a SH expansion in these cases. The code for these two examples is in an appendix [Appendix A8 Code for Multiplying SH by Analytic Functions].

Conclusions

Spherical Harmonics are extremely useful tool, particularly for lighting, in games. Hopefully this article shed some light on how they can be used and how to mitigate some challenges that can arrive when using them. The are several ways to extend the ideas discussed in this paper, windowing coefficients (using any window function) can be solved for that minimize the magnitude of a negative (or positive) lobe, or possibly the magnitude of reflected radiance when pointing in a direction that should be zero. When extracting lights, more rigorous techniques based on non-linear optimization could be used [29][44] and more general lighting models (for example including the cone angles from the light sources discussed) could be extracted. It also might be worth investigating tying windowing into the fitting process. Initially fit with a smoothed version of the function, and then dial back the amount of windowing – effectively steering to better local minima. The content sensitive windowing ideas need to be fleshed out, particularly when integrating with techniques like PRT.

One tool that I've found to be invaluable when playing around with Spherical Harmonics is a symbolic math program. I've used Maple (http://www.maplesoft.com), but other programs like Mathematica (http://www.wolfram.com) would work equally well. The DirectX SDK (http://msdn.microsoft.com/directx) has functions for evaluation, rotation, products and several analytic lighting models, along with samples using both PRT and irradiance environment maps.

Acknowledgements

Much of the work has been done with other collaborators, in particular a long and fruitful series of papers with John Snyder, starting when we were colleagues in the graphics group at Microsoft Research. I've also had the pleasure of discussing Spherical Harmonics with several game developers: Hao Chen (Bungie), Arn Arndt , James Grieves, Clint Hanson (EA at the time), Loren McQuade (while at Blizzard), Dan Baker (MS and Firaxis), Alex Evans (Lionhead at the time), Tom Forsyth (Muckyfoot), Willem de Boer (Muckyfoot), Manchor Ko (Naughty Dog), Naty Hoffman (SCEA) and Chris Oat (ATI). Jason Sandlin, Ben Luna and Jon Steed worked on the samples/test code at Microsoft. Finally I've enjoyed participating in discussions on the GDAlgorithms mailing list on SH and other topics.

References

[1] ANEN, T, KAUTZ, J, DURAND, F, SEIDEL, H. Spherical Harmonic Gradients for Mid-Range Illumination, Eurographics Symposium on Rendering 2004.

[2] AXLER, S, BOURDON, P, RAMEY, W. Harmonic Function Theory, Springer-Verlag, 2000.

[3] BASRI, R, AND JACOBS, D. Lambertian Reflectance and Linear Subspaces, ICCV 2001.

[4] BLAKELY, C, GELB, A, NAVARRA, A. An Automated Method for Recovering Piecewise Smooth Functions on Spheres Free from Gibbs Oscillations, Sampling Theory in Signal and Image Processing Vol 6, No. 3, Sep 2007.

[5] BYERLY, W. An Elementary Treatise on Fourier's Series and Spherical, Cylindrical and Ellipsoidal Harmonics, with Applications to Problems in Mathematical Phyisics, Dover, 1893.

[6] CABRAL, B, MAX, N, SPRINGMEYER, R. Bidirectional Reflection Functions from Surface Bump Maps, SIGGRAPH 1987.

[7] CHANDRASEKHAR, S. Radiative Transfer, Dover, 1960.

[8] DOBASHI, Y, KANEDA, K, NAKATANI, H, YAMASHITA, H. A Quick Rendering Method Using Basis Functions for Interactive Lighting Design, EUROGRAPHICS 1995.

[9] FREEDEN, W, GERVENS, T, SCHREINER, M. Construction Approximation on the Sphere, Clarendon Press, 1998.

[10] GELB, A. The Resolution of the Gibbs Phenomenon for Spherical Harmonics, Mathematics of Computation, Volume 66, Number 218, April 1997.

[11] GENEVAUX, O, LARUE, F, DISCHLER, J. Interactive Refraction on Complex Static Geometry using Spherical Harmonics, Symposium on Interactive 3D Graphics and Games 2006.

[12] GREEN, P, KAUTZ, J, DURAND, F. Efficient Reflectance and Visibility Approximation for Environment Map Rendering, Eurographics 2007.

[13] GREEN, R. Spherical Harmonic Lighting: The Gritty Details, GDC 2003.

[14] HABEL, R, MUSTATA, B, WIMMER, M. Efficient Spherical Harmonics Lighting with the Preetham Skylight Model. Eurographics 2008.

[15] HAN, C, SUN, B, RAMAMOORTHI, R, GRINSPUN, E. Frequency Domain Normal Map Filtering, SIGGRAPH 2007.

[16] INUI, T, TANABE, Y, ONODERA, Y. Group Theory and Its Applications in Physics, Springer-Verlag, 1996.

[17] ISHIMARU, A. Wave Propagation and Scattering in Random Media, IEEE Press, 1978.

[18] KAJIYA, J, von HERZEN, B. Ray Tracing Volume Densities, SIGGRAPH 1984.

[19] KAUTZ, J, SLOAN, P, SNYDER, J. Fast, Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics, EUROGRAPHICS Workshop On Rendering, 2002.

[20] KAUTZ, J, LEHTINEN, J, SLOAN, P. Precomputed Radiance Transfer: Theory and Practice. SIGGRAPH 2005 course. http://www.cs.ucl.ac.uk/staff/j.kautz/PRTCourse/

[21] KO, J, KO, M, ZWICKER, M. Practical Methods for a PRT-based Shader Using Spherical Harmonics, Shader X6: Advanced Rendering Techniques, 2008.

[22] LANN, P, LEUNG, C, WONG, T. Noise-Resistant Fitting for Spherical Harmonics, IEEE Transactions on Visualization and Computer Graphics. 12(2) 2006.

[23] LEHTINEN, J, KAUTZ, J. Matrix Radiance Transfer, Symposium on Interactive 3D Graphics, 2003.

[24] LEHTINEN, J. A Framework for Precomputed and Captured Light Transport. ACM Transactions on Graphics 26(4) 2007.

[25] MACROBERTS, T. Spherical Harmonics, Dover, 1948.

[26] MCTAGGART, G. Half-Life 2 Source Shading, GDC 2004.

[27] NG, R, RAMAMOORTHI, R, HANRAHAN, P. All-Frequency Shadows Using Non-Linear Wavelet Lighting Approximation, SIGGRAPH 2003.

[28] NIMEROFF,J, SIMONCELLI, E, DORSEY, J. Efficient Re-rendering of Natural Environments, EUGROGRAPHICS Workshop on Rendering, 1994.

[29] NOCEDAL, J, WRIGHT, S. Numerical Optimization, Springer-Verlag 1999.

[30] OAT, C. Adding Spherical Harmonic Lighting to the Sushi Engine. GDC 2004. http://ati.amd.com/developer/gdc/Oat-GDC04-SphericalHarmonicLighting.pdf

[31] OAT, C. Irradiance Volumes for Games. GDC 2005. http://ati.amd.com/developer/gdc/GDC2005_PracticalPRT.pdf

[32] OAT, C. Real-Time Irradiance Volumes, ShaderX 5: Advanced Rendering Techniques, 2006.

[33] PAN, M, WANG, R, LIU, X, PENG, Q, HUJUN, B. Precomputed Radiance Transfer Field for Rendering Interreflections in Dynamic Scenes, Eurographics 2007.

[34] PREETHAM, A, SHIRLEY, P, SMITS, B. A Practical Analytic Model for Daylight, SIGGRAPH 1999.

[35] RAMAMOORTHI, R, AND HANRAHAN, H. An Efficient Representation for Irradiance Environment Maps, SIGGRAPH 2001.

[36] RAMAMOORTHI, R, AND HANRAHAN, H. Frequency Space Environment Map Rendering, SIGGRAPH 2002.
[37] REN, Z, WANG, R, SNYDER, J, ZHOU, K, SUN, B, SLOAN, P, BAO, H, PENG, Q, GUO, B. Real-Time Soft Shadows in Dynamic Scenes Using Spherical Harmonic Exponentiation, SIGGRAPH 2006.
[38] RUFFINI, G, MARCO, J, GRAU, C. Spherical Harmonics Interpolation, Computation of Laplacians and Gauge Theory, arXiv:physics/0206007, 2002.
[39] SCHROEDER, P, SWELDENS, W. Spherical Wavelets: Efficiently Representing Functions on the Sphere, SIGGRAPH 1995.
[40] SILLION, F, ARVO, J, WESTIN, S, GREENBERG, D. A Global Illumination Solution for General Reflectance Distributions, SIGGRAPH 1991.
[41] SLOAN, P, KAUTZ, J, SNYDER, J. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments, SIGGRAPH 2002.
[42] SLOAN, P, HALL, J, HART, J, SNYDER, J. Clustered Principal Components for Precomputed Radiance Transfer, SIGGRAPH 2003.
[43] SLOAN, P, LIU, X, SHUM, H-Y, SNYDER, J. Bi-Scale Radiance Transfer, SIGGRAPH 2003.
[44] SLOAN, P, LUNA, B, SNYDER J, Local, Deformable Precomputed Radiance Transfer, SIGGRAPH 2005.
[45] SLOAN, P. Normal Mapping for Precomputed Radiance Transfer, Symposium on Interactive 3D Graphics and Games 2006.
[46] SLOAN, P, GOVINDARAJU, N, NOWROUZEZAHRAI, D, SNYDER, J. Image-Based Proxy Accumulation for Real-Time Soft Global Illumination, Pacific Graphics 2007.
[47] SNYDER, J. Code Generation and Factoring for Fast Evaluation of Low-Order Spherical Harmonic Products and Squares, Microsoft Research Technical Report, MSR-TR-2006-53, 2006.
[48] SNYDER, J. Personal Communication, 2008.
[49] SUN, B, RAMAMOORTHI, R, NARASIMHAN, S, NAYAR, S. A Practical Analytic Single Scattering Model for Real Time Rendering, SIGGRAPH 2005.
[50] STAM, J. Multiple Scattering as a Diffusion Process, EUROGRAPHICS Workshop on Rendering, 1995.
[51] VARSHALOVICH, D, MOSKALEV, A, KHERSONSKII, V. Quantum Theory of Angular Momentum, World Scientific Publishing, 1988.
[52] WANG, J, XU, K, ZHOU, K, LIN, S, HU, S, GUO, B. Spherical Harmonic Scaling, The Visual Computer, 22(9-11), 2006.
[53] WESTIN, S, ARVO, J, TORRANCE, K. Predicting Reflectance Functions from Complex Surfaces", SIGGRAPH 1992.
[54] WONG, T, HENG, P, OR, S, NG, W. Image-Based Rendering with Controllable Illumination, Eurographics Workshop on Rendering, 1997.
[55] WONT, T, FU, C, HENG, P, LEUNG, C. The Plenoptic Illumination Function, IEEE Transactions on Multimedia 4(3) 2002.
[56] ZHOU, K, HU, Y, LIN, S, GUO, B, SHUM, H-Y. Precomputed Shadow Fields for Dynamic Scenes, SIGGRAPH 2005.

Appendix A1 Recursive Rules for Evaluating SH Basis Functions

Recurrence relations [48] can be used to efficiently evaluate the polynomial forms of the SH basis functions. Recall the formulas for the basis functions:

$$y_l^m = \begin{cases} \sqrt{2}\mathrm{Re}(Y_l^m) & m > 0 \\ \sqrt{2}\mathrm{Im}(Y_l^m) & m < 0 \\ Y_l^0 & m = 0 \end{cases} = \begin{cases} \sqrt{2}K_l^m \ \cos m\varphi \ P_l^m(\cos\theta) & m > 0 \\ \sqrt{2}K_l^m \sin|m|\varphi \ P_l^{|m|}(\cos\theta) & m < 0 \\ K_l^0 P_l^0(\cos\theta) & m = 0 \end{cases}$$

Given a point (x,y,z) on the unit sphere, the Associated Legendre polynomials (only depend on Z, divided by $\sin\theta^m$) can be evaluated using these recurrences ($P_0^0 = 1$):

$$
\begin{aligned}
P_m^m &= (1-2m)P_{m-1}^{m-1} \\
P_{m+1}^m &= (2m+1)zP_m^m \\
P_l^m &= \frac{(2l-1)zP_{l-1}^m - (l+m-1)P_{l-2}^m}{l-m}
\end{aligned}
$$

Where you increment l in the inner loop, m in the outer.

The trigonometric addition formula can be used to evaluate the φ dependent terms (multiplied by $\sin\theta^m$ so that they can be computed from (x,y) coordinates on the sphere), where $S(0) = 0, C(0) = 1$:

$$
\begin{aligned}
S(m) &= xS(m-1) + yC(m-1) \\
C(m) &= xC(m-1) - yS(m-1)
\end{aligned}
$$

In the original equation replace $\sin|m|\varphi$ with $S(m)$ and $\cos m\varphi$ with $C(m)$. It is generally more efficient to generate code based on these equations, since it naturally factors common subexpressions.

Appendix A2 Polynomial Forms of SH Basis

The polynomial forms of the SH basis functions are listed below, L is the band index, M is the basis function. Note that Maple randomly changes the order of L and M…

$$\{L = 0, M = 0\}, \frac{1}{2\sqrt{\pi}}$$

$$\{L = 1, M = -1\}, -\frac{\sqrt{3}\,y}{2\sqrt{\pi}}$$

$$\{L = 1, M = 0\}, \frac{\sqrt{3}\,z}{2\sqrt{\pi}}$$

$$\{L = 1, M = 1\}, -\frac{\sqrt{3}\,x}{2\sqrt{\pi}}$$

$$\{L = 2, M = -2\}, \frac{\sqrt{15}\,y\,x}{2\sqrt{\pi}}$$

$$\{M = -1, L = 2\}, -\frac{\sqrt{15}\,y\,z}{2\sqrt{\pi}}$$

$$\{M = 0, L = 2\}, \frac{\sqrt{5}\,(3\,z^2 - 1)}{4\sqrt{\pi}}$$

$$\{M = 1, L = 2\}, -\frac{\sqrt{15}\, x\, z}{2\sqrt{\pi}}$$

$$\{M = 2, L = 2\}, \frac{\sqrt{15}\,(x^2 - y^2)}{4\sqrt{\pi}}$$

$$\{M = -3, L = 3\}, -\frac{\sqrt{2}\,\sqrt{35}\, y\,(3\,x^2 - y^2)}{8\sqrt{\pi}}$$

$$\{M = -2, L = 3\}, \frac{\sqrt{105}\, y\, x\, z}{2\sqrt{\pi}}$$

$$\{M = -1, L = 3\}, -\frac{\sqrt{2}\,\sqrt{21}\, y\,(-1 + 5\,z^2)}{8\sqrt{\pi}}$$

$$\{L = 3, M = 0\}, \frac{\sqrt{7}\, z\,(5\,z^2 - 3)}{4\sqrt{\pi}}$$

$$\{M = 1, L = 3\}, -\frac{\sqrt{2}\,\sqrt{21}\, x\,(-1 + 5\,z^2)}{8\sqrt{\pi}}$$

$$\{L = 3, M = 2\}, \frac{\sqrt{105}\,(x^2 - y^2)\, z}{4\sqrt{\pi}}$$

$$\{M = 3, L = 3\}, -\frac{\sqrt{2}\,\sqrt{35}\, x\,(x^2 - 3\,y^2)}{8\sqrt{\pi}}$$

$$\{L = 4, M = -4\}, \frac{3\sqrt{35}\, y\, x\,(x^2 - y^2)}{4\sqrt{\pi}}$$

$$\{M = -3, L = 4\}, -\frac{3\sqrt{2}\,\sqrt{35}\, y\,(3\,x^2 - y^2)\, z}{8\sqrt{\pi}}$$

$$\{M = -2, L = 4\}, \frac{3\sqrt{5}\, y\, x\,(-1 + 7\,z^2)}{4\sqrt{\pi}}$$

$$\{M = -1, L = 4\}, -\frac{3\sqrt{2}\,\sqrt{5}\, y\, z\,(-3 + 7\,z^2)}{8\sqrt{\pi}}$$

$$\{L = 4, M = 0\}, \frac{3\,(35\,z^4 - 30\,z^2 + 3)}{16\sqrt{\pi}}$$

$$\{M = 1, L = 4\}, -\frac{3\sqrt{2}\sqrt{5}\,x\,z\,(-3 + 7\,z^2)}{8\sqrt{\pi}}$$

$$\{L = 4, M = 2\}, \frac{3\sqrt{5}\,(x^2 - y^2)\,(-1 + 7\,z^2)}{8\sqrt{\pi}}$$

$$\{M = 3, L = 4\}, -\frac{3\sqrt{2}\sqrt{35}\,x\,(x^2 - 3\,y^2)\,z}{8\sqrt{\pi}}$$

$$\{M = 4, L = 4\}, \frac{3\sqrt{35}\,(x^4 - 6\,y^2\,x^2 + y^4)}{16\sqrt{\pi}}$$

$$\{L = 5, M = -5\}, -\frac{3\sqrt{2}\sqrt{77}\,y\,(5\,x^4 - 10\,y^2\,x^2 + y^4)}{32\sqrt{\pi}}$$

$$\{L = 5, M = -4\}, \frac{3\sqrt{385}\,y\,x\,(x^2 - y^2)\,z}{4\sqrt{\pi}}$$

$$\{L = 5, M = -3\}, -\frac{\sqrt{2}\sqrt{385}\,y\,(3\,x^2 - y^2)\,(-1 + 9\,z^2)}{32\sqrt{\pi}}$$

$$\{M = -2, L = 5\}, \frac{\sqrt{1155}\,y\,x\,z\,(-1 + 3\,z^2)}{4\sqrt{\pi}}$$

$$\{M = -1, L = 5\}, -\frac{\sqrt{165}\,y\,(-14\,z^2 + 21\,z^4 + 1)}{16\sqrt{\pi}}$$

$$\{L = 5, M = 0\}, \frac{\sqrt{11}\,z\,(63\,z^4 - 70\,z^2 + 15)}{16\sqrt{\pi}}$$

$$\{L = 5, M = 1\}, -\frac{\sqrt{165}\,x\,(-14\,z^2 + 21\,z^4 + 1)}{16\sqrt{\pi}}$$

$$\{L = 5, M = 2\}, \frac{\sqrt{1155}\,(x^2 - y^2)\,z\,(-1 + 3\,z^2)}{8\sqrt{\pi}}$$

$$\{L = 5, M = 3\}, -\frac{\sqrt{2}\sqrt{385}\,x\,(x^2 - 3\,y^2)\,(-1 + 9\,z^2)}{32\sqrt{\pi}}$$

$$\{L = 5, M = 4\}, \frac{3\sqrt{385}\,(x^4 - 6\,y^2\,x^2 + y^4)\,z}{16\sqrt{\pi}}$$

$$\{L=5, M=5\}, -\frac{3\sqrt{2}\sqrt{77}\,x\,(x^4-10\,y^2\,x^2+5\,y^4)}{32\sqrt{\pi}}$$

Appendix A3 ZH Coefficients for Spherical Light Source

Given a light source that subtends an angle *a* in radians here are the symbolic integrals for the first 6 bands:

L=0 $-\sqrt{\pi}\,(-1+\cos(a))$

L=1 $\frac{1}{2}\sqrt{3}\sqrt{\pi}\,\sin(a)^2$

L=2 $-\frac{1}{2}\sqrt{5}\sqrt{\pi}\,\cos(a)\,(-1+\cos(a))\,(\cos(a)+1)$

L=3 $-\frac{1}{8}\sqrt{7}\sqrt{\pi}\,(-1+\cos(a))\,(\cos(a)+1)\,(5\cos(a)^2-1)$

L=4 $-\frac{3}{8}\sqrt{\pi}\,\cos(a)\,(-1+\cos(a))\,(\cos(a)+1)\,(7\cos(a)^2-3)$

L=5 $-\frac{1}{16}\sqrt{11}\sqrt{\pi}\,(-1+\cos(a))\,(\cos(a)+1)\,(21\cos(a)^4-14\cos(a)^2+1)$

Appendix A4 ZH Coefficients for Smooth Cone

Given a cone that subtends the angle a in radians, the light source has intensity 1 at the north pole, and falls off to zero at angle *a*. At 6th order this function should not be evaluated with an angle less than about 8 degrees using single precision. The derivative of the smoothing function is 0 at the north pole and a. The first 6 bands are:

$$\frac{(a^3+6\,a-12\sin(a)+6\cos(a)\,a)\sqrt{\pi}}{a^3}$$

$$\frac{1}{4}\frac{\sqrt{3}\,(a^3-3\cos(a)\sin(a)+3\cos(a)^2\,a)\sqrt{\pi}}{a^3}$$

$$\frac{1}{9}\frac{\sqrt{5}\,(-6\,a-2\cos(a)^2\sin(a)-9\cos(a)\,a+14\sin(a)+3\cos(a)^3\,a)\sqrt{\pi}}{a^3}$$

$$\frac{1}{256}\sqrt{7}\,(4\,a^3+15\,a-108\cos(a)^2\,a-30\cos(a)^3\sin(a)+63\cos(a)\sin(a)+60\cos(a)^4\,a)\sqrt{\pi}\,/\,a^3$$

$$\frac{1}{1500}\,(-480\,a+742\sin(a)+596\cos(a)^2\sin(a)+225\cos(a)\,a-378\sin(a)\cos(a)^4-1650\cos(a)^3\,a+945\cos(a)^5\,a)\sqrt{\pi}\,/\,a^3$$

$$\frac{1}{3072}\sqrt{11}\,(-63\,a+12\,a^3+350\cos(a)^3\sin(a)-1260\cos(a)^4\,a-15\cos(a)\sin(a)-224\cos(a)^5\sin(a)+672\cos(a)^6\,a+540\cos(a)^2\,a)\sqrt{\pi}\,/\,a^3$$

Appendix A5 Solving for Coefficients to Approximate SH Environment Map with Directional and Ambient Lights

You can compute the intensity *s* that minimizes the approximation error between a directional light in direction *d* and the original lighting environment:

$$E(c) = (L_e - cL_d)^2$$

Where L_e is the SH representation of the lighting environment, L_d is the SH representation of the lighting model in direction *d*[13]. The solution is:

$$c = \frac{L_e \circ L_d}{L_d \circ L_d}$$

If you want to add an ambient light, you need to minimize the error function:

$$E(c,a) = \int \left(cL_d(s) * H_N(s) + aL_a(s) * H_N(s) - L_e(s) * H_N(s)\right)^2 ds$$

Absorbing the convolution into the lighting and differentiate with respect to each variable:

[13] You can optimize using either radiance (leave vectors alone) or irradiance (convolve vectors.)

$$\frac{dE}{dc} = 2\int \left(c\hat{L}_d(s) + a\hat{L}_a(s) - \hat{L}_e(s)\right)\hat{L}_d(s)ds$$

$$\frac{dE}{da} = 2\int \left(c\hat{L}_d(s) + a\hat{L}_a(s) - \hat{L}_e(s)\right)\hat{L}_a(s)ds$$

Now solve for the two equations equal to zero[14] to find the minimum. $\hat{L}_a(s)$ is a scalar function[15], and due to the orthogonality of SH the integrals are all simple dot products of SH vectors. This leaves you with the following equations:

$$\begin{aligned} cA + aB &= D \\ cB + aC &= E \end{aligned}$$

Where:

$$A = \frac{508\pi}{867}, B = \frac{16}{17}, C = \frac{4}{\pi}, D = dot\left(\hat{L}_d, \hat{L}_e\right), E = dot\left(\hat{L}_a, \hat{L}_e\right)$$

Solving for *c* and *a* results in:

$$\begin{aligned} c &= \frac{867}{316\pi}\, dot\left(\hat{L}_d, \hat{L}_e\right) - \frac{51}{79}\, dot\left(\hat{L}_a, \hat{L}_e\right) \\ a &= \frac{127\pi}{316}\, dot\left(\hat{L}_a, \hat{L}_e\right) - \frac{51}{79}\, dot\left(\hat{L}_d, \hat{L}_e\right) \end{aligned}$$

It turns out there is a simpler way to arrive at the same results. Solve for the intensity of a directional light where both the environment and the light don't include the DC term, then computing an ambient term that reconstructs the environment DC term when using the scaled directional light. This results in:

$$\begin{aligned} c &= \frac{867}{316\pi}\, dot_0\left(\hat{L}_d, \hat{L}_e\right) \\ a &= \left(\hat{L}_e[0] - c\frac{8\sqrt{\pi}}{17}\right)\frac{\sqrt{\pi}}{2} \end{aligned}$$

In the above expression the dot product ignores the DC term and $\hat{L}_e[0]$ is the DC term of the environment light. We will use similar techniques for 2 and 3 lights (where the equations get much nastier if you handle *a* at the same time as the light intensities.)

Multiple lights can be done in a similar manner, first by removing DC from all the light vectors, with two lights giving you this error function:

$$E(c_0, c_1) = \int \left(c_0 L_{d0}(s) + c_1 L_{d1}(s) - L_e(s)\right)^2 ds$$

---

[14] The Hessian matrix is $\begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}$ which has positive eigenvalues $(2 \pm \sqrt{2})$, so this is a minimum.

[15] SH vector only has a non-zero in the DC term.

Then differentiate with respect to each of the coefficients and solve for zero. The following expression gives you the intensities:

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} \frac{A}{A^2 - B^2} & \frac{-B}{A^2 - B^2} \\ \frac{-B}{A^2 - B^2} & \frac{A}{A^2 - B^2} \end{bmatrix} \begin{bmatrix} L_e \circ L_{d0} \\ L_e \circ L_{d0} \end{bmatrix}$$

Where:

$$A = L_{d0} \circ L_{d0}, B = L_{d0} \circ L_{d1}$$

It is worth pointing out that *A* is constant independent of direction (it depends on order and if the vectors are convolved or not.) The ambient term is then:

$$a = \left(L_e[0] - (c_0 L_{d0}[0] + c_1 L_{d1}[0])\right) \frac{\sqrt{\pi}}{2}$$

For three lights the intensities are:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \frac{A^2 - D^2}{E} & \frac{CD - AB}{E} & \frac{BD - AC}{E} \\ \frac{CD - AB}{E} & \frac{A^2 - C^2}{E} & \frac{BC - AD}{E} \\ \frac{BD - AC}{E} & \frac{BC - AD}{E} & \frac{A^2 - B^2}{E} \end{bmatrix} \begin{bmatrix} L_e \circ L_{d0} \\ L_e \circ L_{d1} \\ L_e \circ L_{d2} \end{bmatrix}$$

Where:

$$C = L_{d0} \circ L_{d2}, D = L_{d1} \circ L_{d2,} E = 2BCD + A(A^2 - B^2 - C^2 - D^2)$$

This matrix is symmetric, and the ambient coefficient is:

$$a = \left(L_e[0] - (c_0 L_{d0}[0] + c_1 L_{d1}[0] + c_2 L_{d2}[0])\right) \frac{\sqrt{\pi}}{2}$$

Appendix A6 Least Squares Projection

First we will show that for orthonormal basis functions, least squares projection actually is achieved by simply integrating against the basis functions. We want to find the coefficient vector *c* that minimizes this expression:

$$E(c) = \int \left(\sum_i c_i y_i(s) - f(s) \right)^2 ds$$

This can be done by differentiating with respect to each coefficient, and solving the first derivatives for zero[16]

$$\frac{dE}{dc_k} = 2\int\left(\sum_i c_i y_i(s) - f(s)\right) y_k(s)\, ds$$

Because the basis functions are orthonormal we know that $\int y_i(s) y_j(s) ds = \delta_{ij}$ exploiting this fact and solving for zero we get

$$\frac{dE}{dc_k} = 0 \xrightarrow{yields} c_k = \int y_k(s) f(s)\, ds$$

So that direct integration gives you the least squares result.

Now we derive the coefficient vector *g* that minimizes an error functional that includes a penalty based on the squared Laplacian integrated over the sphere. We know from above the coefficient vector *c* that minimizes pure squared error. We introduce a function *h* that is used to simplify indexing

$$\int \left(\Delta \tilde{f}\right)^2 ds = \sum_{l=1}^{n} \sum_{m=-1}^{l} l^2 (l+1)^2 (f_l^m)^2 = \sum_{i=0}^{n^2} h_i (f_i)^2$$

The error function then is

$$E(g) = \left(\sum_i (g_i - c_i)^2\right) + \lambda \sum_i h_i {g_i}^2$$

Differentiating you get

$$\frac{dE}{dg_k} = 2(g_k - c_k) + 2\lambda h_k g_k$$

Solving for zero you get

$$g_k = \frac{c_k}{(1 + \lambda h_k)}$$

Appendix A7 Solving for Lamba to Reduce the Squared Laplacian

This can be done quite easily using Newtons method. The squared Laplacian is:

[16] Second derivative mixed partials are all zero, unmixed partials are all 1, so the hessian is an identity matrix and we have a minima

$$\Delta^2 = \sum_{l=1}^{n} \sum_{m=-1}^{l} l^2(l+1)^2 (f_l^m)^2 = \sum_{l=1}^{n} l^2(l+1)^2 \sum_{m=-1}^{l} (f_l^m)^2 = \sum_{l=1}^{n} L_l B_l$$

The array of *L* values ($L_l = l^2(l+1)^2$) is static, and the array of *B* values can be computed $B_l = \sum_{m=-1}^{l} (f_l^m)^2$. Newtons method takes an initial guess (0 works well for this problem) and refines it using the following recurrence:

$$\lambda_{n+1} = \lambda_n - \frac{f(\lambda_n)}{f'(\lambda_n)}$$

Where *f* is the function we are searching for a root and *f'* is the first derivative. Replacing f_l^m with c_l^m which includes λ and factoring you have:

$$f(\lambda) = \Delta^2 - \sum_{l=1}^{n} \frac{L_l B_l}{(1+\lambda L_l)^2}$$

$$f^{'(\lambda)} = 2\sum_{l=1}^{n} \frac{{L_l}^2 B_l}{(1+\lambda L_l)^3}$$

Where $\Delta^2$is the original squared Laplacian. Iterate until a maximum number of iteration has occurred, or the absolute value of successive approximations is below a threshold (1e-6 seems to work well in practice.) This can be represented as a polynomial (multiplying through by the product of all the denominators) of degree $(n-1)^2$, where *n* is the order. This would only be useful for quadratic and lower orders, where a closed form solution of the roots can be computed and even then it is not clear if it would be much faster than an iterative solution.

Appendix A8 Code for Multiplying SH by Analytic Functions

Below are two functions, both defined over the hemisphere. The first is simply constant, which can be useful in scenes with a dominant ground plane. The second is a clamped cosine in Z.

```
// generates 6th order SH coefficients from multiplying 6th order lighting by hemisphere in Z

void HemiMult(float *R, float *L)
{
    R[0] = 0.433012702f * L[2] - 0.1653594569f * L[12] + 0.1036445247f * L[30] + 0.5f * L[0];
    R[1] = 0.5f * L[1] + 0.4192627457f * L[5] - 0.1711632992f * L[19];
    R[2] = -0.05412658775f * L[20] + 0.5f * L[2] + 0.2420614591f * L[6] + 0.433012702f * L[0];
    R[3] = 0.5f * L[3] - 0.1711632992f * L[21] + 0.4192627457f * L[7];
    R[4] = 0.5f * L[4] - 0.1713860232f * L[28] + 0.4133986423f * L[10];
    R[5] = -0.06477782793f * L[29] + 0.4192627457f * L[1] + 0.5f * L[5] + 0.2614562582f * L[11];
    R[6] = 0.5f * L[6] - 0.1448476267f * L[30] + 0.2420614591f * L[2] + 0.3697549864f * L[12];
    R[7] = -0.06477782793f * L[31] + 0.4192627457f * L[3] + 0.5f * L[7] + 0.2614562582f * L[13];
    R[8] = 0.5f * L[8] - 0.1713860232f * L[32] + 0.4133986423f * L[14];
    R[9] = 0.5f * L[9] + 0.41015625f * L[17];
    R[10] = 0.2685102947f * L[18] + 0.5f * L[10] + 0.4133986423f * L[4];
    R[11] = 0.360244363f * L[19] + 0.5f * L[11] + 0.2614562582f * L[5];
    R[12] = 0.2790440836f * L[20] + 0.5f * L[12] - 0.1653594569f * L[0] + 0.3697549864f * L[6];
```

```
    R[13] = 0.5f * L[13] + 0.360244363f * L[21] + 0.2614562582f * L[7];
    R[14] = 0.2685102947f * L[22] + 0.4133986423f * L[8] + 0.5f * L[14];
    R[15] = 0.41015625f * L[23] + 0.5f * L[15];
    R[16] = 0.408100316f * L[26] + 0.5f * L[16];
    R[17] = 0.2720668773f * L[27] + 0.5f * L[17] + 0.41015625f * L[9];
    R[18] = 0.2685102947f * L[10] + 0.35621916f * L[28] + 0.5f * L[18];
    R[19] = 0.2856107251f * L[29] - 0.1711632992f * L[1] + 0.5f * L[19] + 0.360244363f * L[11];
    R[20] = 0.2790440836f * L[12] + 0.5f * L[20] + 0.3498002708f * L[30] - 0.05412658775f * L[2];
    R[21] = 0.2856107251f * L[31] + 0.360244363f * L[13] + 0.5f * L[21] - 0.1711632992f * L[3];
    R[22] = 0.5f * L[22] + 0.35621916f * L[32] + 0.2685102947f * L[14];
    R[23] = 0.2720668773f * L[33] + 0.5f * L[23] + 0.41015625f * L[15];
    R[24] = 0.5f * L[24] + 0.408100316f * L[34];
    R[25] = 0.5f * L[25];
    R[26] = 0.408100316f * L[16] + 0.5f * L[26];
    R[27] = 0.5f * L[27] + 0.2720668773f * L[17];
    R[28] = 0.5f * L[28] - 0.1713860232f * L[4] + 0.35621916f * L[18];
    R[29] = -0.06477782793f * L[5] + 0.5f * L[29] + 0.2856107251f * L[19];
    R[30] = 0.3498002708f * L[20] + 0.1036445247f * L[0] - 0.1448476267f * L[6] + 0.5f * L[30];
    R[31] = -0.06477782793f * L[7] + 0.2856107251f * L[21] + 0.5f * L[31];
    R[32] = 0.5f * L[32] - 0.1713860232f * L[8] + 0.35621916f * L[22];
    R[33] = 0.5f * L[33] + 0.2720668773f * L[23];
    R[34] = 0.5f * L[34] + 0.408100316f * L[24];
    R[35] = 0.5f * L[35];
}
// generates 6th order SH coefficients from multiplying 6th order lighting by clamped cos in Z

void CosMult(float *R, float *L)
{
    // constants to make code fit
    const float T1 = 0.09547032698f;
    const float T2 = 0.1169267933f;
    const float T3 = 0.2581988897f;
    const float T4 = 0.2886751347f;
    const float T5 = 0.2390457218f;
    const float T6 = 0.2535462764f;
    const float T7 = 0.2182178903f;
    const float T8 = 0.1083940384f;
    const float T9 = 0.2519763153f;
    const float TA = 0.2439750183f;
    const float TB = 0.3115234375f;
    const float TC = 0.2512594538f;
    const float TD = 0.31640625f;

    R[0] = 0.25f * L[0] - 0.03125f * L[20] + T4 * L[2] + 0.1397542486f * L[6];
    R[1] = 0.2236067977f * L[5] + T2* L[11] - 0.02896952533f * L[29] + 0.1875f * L[1];
    R[2] = 0.375f * L[2] + T4 * L[0] + T3 * L[6] - 0.01495979856f * L[30] + T1 * L[12];
    R[3] = 0.2236067977f * L[7] - 0.02896952533f * L[31] + T2* L[13] + 0.1875f * L[3];
    R[4] = 0.101487352f * L[18] + 0.1889822365f * L[10] + 0.15625f * L[4];
    R[5] = 0.2236067977f * L[1] + T5 * L[11] + 0.3125f * L[5] + 0.09568319309f * L[19];
    R[6] = T6 * L[12] + T3 * L[2] + 0.3125f * L[6] + 0.113550327f * L[20] + 0.1397542486f * L[0];
    R[7] = T5 * L[13] + 0.09568319309f * L[21] + 0.2236067977f * L[3] + 0.3125f * L[7];
    R[8] = 0.15625f * L[8] + 0.101487352f * L[22] + 0.1889822365f * L[14];
    R[9] = 0.09068895910f * L[27] + 0.1666666667f * L[17] + 0.13671875f * L[9];
    R[10] = T7 * L[18] + 0.2734375f * L[10] + 0.09068895910f * L[28] + 0.1889822365f * L[4];
    R[11] = T8 * L[29] + TA * L[19] + T5 * L[5] + 0.30078125f * L[11] + T2* L[1];
    R[12] = T9 * L[20] + 0.328125f * L[12] + T6 * L[6] + 0.1028316139f * L[30] + T1 * L[2];
    R[13] = 0.30078125f * L[13] + T8 * L[31] + TA * L[21] + T5 * L[7] + T2* L[3];
    R[14] = 0.2734375f * L[14] + 0.09068895910f * L[32] + 0.1889822365f * L[8] + T7 * L[22];
    R[15] = 0.0906889591f * L[33] + 0.13671875f * L[15] + 0.1666666667f * L[23];
    R[16] = 0.1507556723f * L[26] + 0.123046875f * L[16];
    R[17] = 0.24609375f * L[17] + 0.1666666667f * L[9] + 0.201007563f * L[27];
    R[18] = 0.101487352f * L[4] + T7 * L[10] + 0.2302830933f * L[28] + 0.28125f * L[18];
    R[19] = TD * L[19] + 0.09568319309f * L[5] + TA * L[11] + 0.2461829819f * L[29];
    R[20] = T9 * L[12] + TC * L[30] + 0.113550327f * L[6] - 0.03125f * L[0] + TD * L[20];
    R[21] = 0.2461829819f * L[31] + TA * L[13] + 0.9568319309e-1f * L[7] + TD * L[21];
    R[22] = 0.101487352f * L[8] + 0.28125f * L[22] + T7 * L[14] + 0.2302830933f * L[32];
    R[23] = 0.201007563f * L[33] + 0.1666666667f * L[15] + 0.24609375f * L[23];
    R[24] = 0.1507556723f * L[34] + 0.123046875f * L[24];
    R[25] = 0.1127929688f * L[25];
    R[26] = 0.2255859375f * L[26] + 0.1507556723f * L[16];
    R[27] = 0.9068895910e-1f * L[9] + 0.201007563f * L[17] + 0.2631835938f * L[27];
```

```
    R[28] = 0.2302830933f * L[18] + 0.9068895910e-1f * L[10] + 0.30078125f * L[28];
    R[29] = 0.1083940385f * L[11] + 0.2461829819f * L[19] - 0.02896952533f * L[1] + TB * L[29];
    R[30] = 0.322265625f * L[30] + TC * L[20] - 0.01495979856f * L[2] + 0.1028316139f * L[12];
    R[31] = 0.2461829819f * L[21] - 0.02896952533f * L[3] + TB * L[31] + 0.1083940385f * L[13];
    R[32] = 0.09068895910f * L[14] + 0.2302830933f * L[22] + 0.30078125f * L[32];
    R[33] = 0.09068895910f * L[15] + 0.201007563f * L[23] + 0.2631835938f * L[33];
    R[34] = 0.2255859375f * L[34] + 0.1507556723f * L[24];
    R[35] = 0.1127929688f * L[35];
}
```

Appendix A9 Ambient Cube Basis

The Ambient Cube basis is used by Valve [26]; it consists of 6 basis functions, each defined over a hemisphere:

$$V0 = \begin{cases} x^2 & x > 0 \\ 0 & x \leq 0 \end{cases}, V1 = \begin{cases} 0 & x > 0 \\ x^2 & x \leq 0 \end{cases}$$

$$V2 = \begin{cases} y^2 & y > 0 \\ 0 & y \leq 0 \end{cases}, V3 = \begin{cases} 0 & y > 0 \\ y^2 & y \leq 0 \end{cases}$$

$$V3 = \begin{cases} z^2 & z > 0 \\ 0 & z \leq 0 \end{cases}, V4 = \begin{cases} 0 & z > 0 \\ z^2 & z \leq 0 \end{cases}$$

This basis is not orthogonal, so coefficients cannot be generated by simply integrating against the basis functions. To compute optimal projection coefficients you solve a linear least squares problem (similar to Appendix A6 Least Squares Projection, but without orthonormal basis functions):

$$E(c) = \int \left(\sum_i c_i V_i(s) - f(s) \right)^2 ds$$

Where the $V_i(s)$ are the Ambient Cube basis functions. Differentiating you get:

$$\frac{dE}{dc_k} = 2 \int \left(\sum_i c_i V_i(s) - f(s) \right) V_k(s)\, ds$$

Integration is linear so you can re-arange the terms and solve for zero getting:

$$\sum_i c_i \int V_i(s) V_k(s) ds = \int f(s) V_k(s) ds$$

The left hand side gives you a row of a matrix A, where $A_{ij} = \int V_i(s) V_j(s) ds$ and the right hand side will give you a vector (integrals of function times this basis function.) This leads to a linear system: $Ac = b$ the inverse of A is:

$$\begin{bmatrix} \frac{11}{4\pi} & \frac{1}{4\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} \\ \frac{1}{4\pi} & \frac{11}{4\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} \\ -\frac{3}{8\pi} & -\frac{3}{8\pi} & \frac{11}{4\pi} & \frac{1}{4\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} \\ -\frac{3}{8\pi} & -\frac{3}{8\pi} & \frac{1}{4\pi} & \frac{11}{4\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} \\ -\frac{3}{8\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} & \frac{11}{4\pi} & \frac{1}{4\pi} \\ -\frac{3}{8\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} & -\frac{3}{8\pi} & \frac{1}{4\pi} & \frac{11}{4\pi} \end{bmatrix}$$

To project a function represented using this basis into SH you just integrate the V's against the SH basis, for quadratic spherical harmonics that results in the following matrix:

$$\begin{bmatrix} \frac{\sqrt{\pi}}{3} & \frac{\sqrt{\pi}}{3} & \frac{\sqrt{\pi}}{3} & \frac{\sqrt{\pi}}{3} & \frac{\sqrt{\pi}}{3} & \frac{\sqrt{\pi}}{3} \\ 0 & 0 & -\frac{\sqrt{3}\sqrt{\pi}}{4} & \frac{\sqrt{3}\sqrt{\pi}}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{3}\sqrt{\pi}}{4} & -\frac{\sqrt{3}\sqrt{\pi}}{4} \\ -\frac{\sqrt{3}\sqrt{\pi}}{4} & \frac{\sqrt{3}\sqrt{\pi}}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{\sqrt{\pi}\sqrt{5}}{15} & -\frac{\sqrt{\pi}\sqrt{5}}{15} & -\frac{\sqrt{\pi}\sqrt{5}}{15} & -\frac{\sqrt{\pi}\sqrt{5}}{15} & \frac{2\sqrt{\pi}\sqrt{5}}{15} & \frac{2\sqrt{\pi}\sqrt{5}}{15} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\sqrt{\pi}\sqrt{15}}{15} & \frac{\sqrt{\pi}\sqrt{15}}{15} & -\frac{\sqrt{\pi}\sqrt{15}}{15} & -\frac{\sqrt{\pi}\sqrt{15}}{15} & 0 & 0 \end{bmatrix}$$

The even degrees above quadratic are in the null-space of the Ambient Cube basis, and the odd degrees above quadratic are in the null space of the clamped cosine function, so if irradiance environment maps are being used, no higher order is necessary. The Ambient Cube basis can exactly reconstruct the DC term, approximate the linear term, exactly reconstructs two of the quadratic basis functions (y_2^0, y_2^2) and has three of the quadratic basis functions in the null space ($y_2^{-2}, y_2^{-1}, y_2^1$).

To project from SH into the Ambient Cube basis you need to minimize this error function:

$$E(c) = \int \left(\sum_i c_i V_i(s) - \sum_j l_j y_j(s) \right)^2 ds$$

Where l_j are the SH lighting coefficients (assumed to be convolved.) Once again differentiate with respect to the unkown variables:

$$\frac{dE}{dc_k} = 2 \int \left(\sum_i c_i V_i(s) - \sum_j l_j y_j(s) \right) V_k(s)\, ds$$

And then solve for zero:

$$\sum_i c_i \int V_i(s) V_k(s) ds = \sum_j l_j \int y_j(s) V_k(s) ds$$

This results in a linear system as well, $Ac = Bl$, where A is the same as before B is a matrix where $B_{ij} = \int V_i(s) y_j(s) ds$. The matrix $A^{-1}B$ can be used to move from SH to the Ambient Cube basis:

$$\begin{bmatrix}
\frac{1}{2\sqrt{\pi}} & 0 & 0 & -\frac{5\sqrt{3}}{8\sqrt{\pi}} & 0 & 0 & -\frac{\sqrt{5}}{4\sqrt{\pi}} & 0 & \frac{\sqrt{15}}{4\sqrt{\pi}} \\
\frac{1}{2\sqrt{\pi}} & 0 & 0 & \frac{5\sqrt{3}}{8\sqrt{\pi}} & 0 & 0 & -\frac{\sqrt{5}}{4\sqrt{\pi}} & 0 & \frac{\sqrt{15}}{4\sqrt{\pi}} \\
\frac{1}{2\sqrt{\pi}} & -\frac{5\sqrt{3}}{8\sqrt{\pi}} & 0 & 0 & 0 & 0 & -\frac{\sqrt{5}}{4\sqrt{\pi}} & 0 & -\frac{\sqrt{15}}{4\sqrt{\pi}} \\
\frac{1}{2\sqrt{\pi}} & \frac{5\sqrt{3}}{8\sqrt{\pi}} & 0 & 0 & 0 & 0 & -\frac{\sqrt{5}}{4\sqrt{\pi}} & 0 & -\frac{\sqrt{15}}{4\sqrt{\pi}} \\
\frac{1}{2\sqrt{\pi}} & 0 & \frac{5\sqrt{3}}{8\sqrt{\pi}} & 0 & 0 & 0 & \frac{\sqrt{5}}{2\sqrt{\pi}} & 0 & 0 \\
\frac{1}{2\sqrt{\pi}} & 0 & -\frac{5\sqrt{3}}{8\sqrt{\pi}} & 0 & 0 & 0 & \frac{\sqrt{5}}{2\sqrt{\pi}} & 0 & 0
\end{bmatrix}$$

Appendix A10 Shader/CPU code for Irradiance Environment Maps

Given a quadratic SH representation of a lighting environment, it is fairly simple to generate shader code. In [35] a matrix representation is used, however this turns out to require more instructions[17] (15 vs. 11) and more constants (12 vs. 7) compared to a more direct evaluation of the quadratic spherical harmonics. You fold leading constants of the polynomials into the lighting coefficients, group all but y_2^{-2} by channel (using float4's) and keep y_2^{-2} as a color and

[17] On scalar GPU's this gap is even more significant – 60 vs. 42

fold part of y_2^0 into DC. Shader code for evaluation and CPU code to setup the constants is shown below. The normal passed in to shader code should be normalized and the 4th channel should be 1.0. The CPU code takes an array of 3 float pointers to quadratic radiance SH coefficients and an Effect to bind the constants to.

```
// constants containing irradiance environmne map
float4 cAr;
float4 cAg;
float4 cAb;
float4 cBr;
float4 cBg;
float4 cBb;
float4 cC;

float3 ShadeIrad(float4 vNormal)
{
  float3 x1, x2, x3;

  // Linear + constant polynomial terms
  x1.r = dot(cAr,vNormal);
  x1.g = dot(cAg,vNormal);
  x1.b = dot(cAb,vNormal);

  // 4 of the quadratic polynomials
  float4 vB = vNormal.xyzz * vNormal.yzzx;
  x2.r = dot(cBr,vB);
  x2.g = dot(cBg,vB);
  x2.b = dot(cBb,vB);

  // Final quadratic polynomial
  float vC = vNormal.x*vNormal.x - vNormal.y*vNormal.y;
  x3 = cC.rgb * vC;
  return x1+x2+x3;
}

//-------------------------------------------------------------
void SetSHEMapConstants( float* fLight[3], ID3DXEffect* pEffect)
{
    // Lighting environment coefficients
    D3DXVECTOR4 vCoeff[3];

    static const float s_fSqrtPI = ((float)sqrtf(D3DX_PI));
    const float fC0 = 1.0f/(2.0f*s_fSqrtPI);
    const float fC1 = (float)sqrt(3.0f)/(3.0f*s_fSqrtPI);
    const float fC2 = (float)sqrt(15.0f)/(8.0f*s_fSqrtPI);
    const float fC3 = (float)sqrt(5.0f)/(16.0f*s_fSqrtPI);
    const float fC4 = 0.5f*fC2;

    int iC;
    for( iC=0; iC<3; iC++ )
    {
        vCoeff[iC].x = -fC1*fLight[iC][3];
        vCoeff[iC].y = -fC1*fLight[iC][1];
        vCoeff[iC].z =  fC1*fLight[iC][2];
        vCoeff[iC].w =  fC0*fLight[iC][0] - fC3*fLight[iC][6];
```

```
    }

    pEffect->SetVector( "cAr", &vCoeff[0] );
    pEffect->SetVector( "cAg", &vCoeff[1] );
    pEffect->SetVector( "cAb", &vCoeff[2] );

    for( iC=0; iC<3; iC++ )
    {
        vCoeff[iC].x =      fC2*fLight[iC][4];
        vCoeff[iC].y =     -fC2*fLight[iC][5];
        vCoeff[iC].z = 3.0f*fC3*fLight[iC][6];
        vCoeff[iC].w =     -fC2*fLight[iC][7];
    }

    pEffect->SetVector( "cBr", &vCoeff[0] );
    pEffect->SetVector( "cBg", &vCoeff[1] );
    pEffect->SetVector( "cBb", &vCoeff[2] );

    vCoeff[0].x = fC4*fLight[0][8];
    vCoeff[0].y = fC4*fLight[1][8];
    vCoeff[0].z = fC4*fLight[2][8];
    vCoeff[0].w = 1.0f;

    pEffect->SetVector( "cC", &vCoeff[0] );
}
```

Version History

- 2/14/2008 first public version
- 4/1/2008 Added appendix with recurrence relations when evaluating SH, renumbered appendices
- 3/17/2009 Fixed typos in the recurrence relations
- 11/30/2009 Ambient light and optimal linear direction typos fixed
- 2/10/2010 Typo in defining product matrix (thanks to Don Holden for pointing it out)