A Tighter Quantization Bound for the Spherical Harmonic Projection of Non-Negative Functions

Technical Memo ATVI-TR-24-02

TYLER WIEDERIEN, Gibson Ek High School
PETER-PIKE SLOAN, Activision Publishing, Inc.

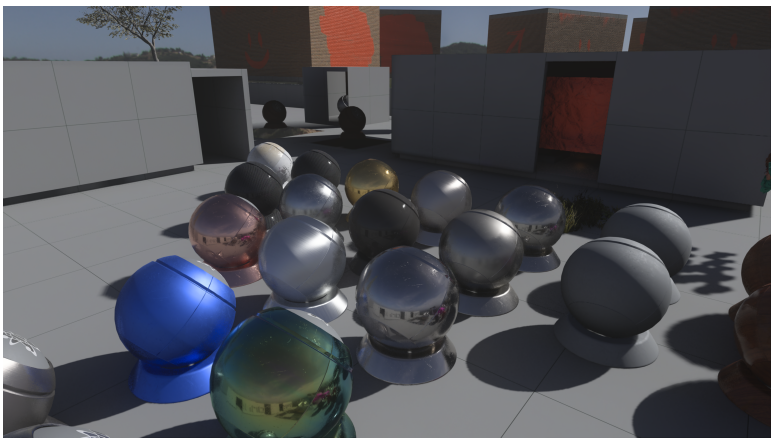

Fig. 1. *Test scene used. The new encoding reduces L^1 error by over 4%.*

Spherical harmonics are a common function space used to encode lighting and other signals represented over the sphere. While some signals, like visibility, have limited dynamic range, the most common signals represented are indirect or direct light. For non-negative signals, there is an analytic upper bound on the ratio of a basis function divided by the first coefficient which is simply the scaled average over the sphere. This upper bound can be used to encode the higher order coefficients using reduced precision. This article derives and discusses an earlier bound, and shows that for the quadratic band a tighter bound can be used that reduces quantization error.

1 INTRODUCTION

Spherical harmonics (SH) are extensively used in interactive applications to represent lighting [1], visibility [2] and other signals on the sphere. This results in a set of coefficients that in the general case, like transfer functions from precomputed radiance transfer [4] need to be encoded with floating point values. Some signals, like visibility, have bounded values on the sphere, so encoding the coefficients used lower precision representations is fine, but what about the most common signal, light? It turns out that for any non-negative function, a bound can be derived [4] on the ratio of the higher order coefficients divided by DC[1], enabling those to be encoded using lower precision textures/buffers independent of the actual intensity of the lighting. While the previous bound is tight for the linear band, 4 of the 5 quadratic coefficients have a slightly tighter bound. This paper derives and discusses the previous bound, and also shows both why a tighter bound exists for higher orders and derives it as well.

[1]the 1st coefficient, which is a scaled average value over the sphere.

Authors' addresses: Tyler Wiederien, tlrwiederien@gmail.com., Gibson Ek High School; Peter-Pike Sloan, ppsloan@activision.com, Activision Publishing, Inc.

2 BACKGROUND

2.1 Spherical Harmonics

The complex spherical harmonics are commonly used in non-graphics applications, we focus on the real spherical harmonics. Expressed in spherical coordinates:

$$Y_l^m \begin{cases} \sqrt{2}K_l^m \cos(m\phi)P_l^m(\cos\theta) & m > 0 \\ \sqrt{2}K_l^m \sin(|m|\phi)P_l^{|m|}(\cos\theta) & m < 0 \\ K_l^m P_l^m(\cos\theta) & m = 0 \end{cases} \tag{1}$$

where

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

and P_l^m are the associated Legendre polynomials. We focus on the $l = 2$ band:

Y_l^m	Spherical (θ, ϕ)	Cartesian (x, y, z)
Y_2^{-2}	$\frac{\sqrt{15}}{2\sqrt{\pi}} \sin\phi \cos\phi \sin^2\theta$	$\frac{\sqrt{15}}{2\sqrt{\pi}} xy$
Y_2^{-1}	$\frac{-\sqrt{15}}{2\sqrt{\pi}} \sin\phi \sin\theta \cos\theta$	$\frac{-\sqrt{15}}{2\sqrt{\pi}} yz$
Y_2^0	$\frac{\sqrt{5}}{4\sqrt{\pi}}(3\cos^2\theta - 1)$	$\frac{\sqrt{5}}{4\sqrt{\pi}}(3z^2 - 1)$
Y_2^1	$\frac{-\sqrt{15}}{2\sqrt{\pi}} \cos\phi \sin\theta \cos\theta$	$\frac{-\sqrt{15}}{2\sqrt{\pi}} xz$
Y_2^2	$\frac{\sqrt{15}}{4\sqrt{\pi}} \sin^2\theta(\cos^2\phi - \sin^2\phi)$	$\frac{\sqrt{15}}{4\sqrt{\pi}}(x^2 - y^2)$

Table 1. *The quadratic ($l = 2$) Spherical Harmonics [3]*

Projection into SH is simple due to the fact that they are an orthogonal basis, a function $f(s)$ is approximated with a coefficient vector:

$$b_l^m = \int f(s) Y_l^m(s) ds \tag{3}$$

Where the domain of integration is the surface of the sphere.

Reconstruction is then:

$$\tilde{f}(s) = \sum_{l=0}^{l=N} \sum_{m=-l}^{m=1} b_l^m Y_l^m(s) \tag{4}$$

Functions that have circular symmetry in the Z-axis project only to a subset of the basis functions, the zonal harmonics (ZH), which has a single non-zero function per-band, the $m = 0$ function.

SH are closed under rotation, and each band is closed under rotation, no energy moves between bands. One important implication of this is that norm of the coefficients in a band is also closed under rotation.

2.2 Prior Bound

Projection of a function $f(s)$ into orthogonal basis $B_k(s)$ with Monte Carlo integration, leads to a sum of scaled delta functions:

$$b_j = \frac{1}{N} \sum_{i=1}^{N} \frac{B_j(x_i) f(x_i)}{p(x_i)} \tag{5}$$

So each coefficient is created by summing the basis function over the domain $B_j(x_i)$ and scaling it by the value $\frac{f(x_i)}{p(x_i)}$, if $f(s)$ is non-negative, this scale factor is also non-negative. The projection coefficients have a mixture of constructive interference, locations on the sphere where the basis functions is positive, and destructive interference, locations where the basis function is negative. The DC function only has constructive interference for non-negative functions, while the rest of the basis functions can have both types of interference. This is why a bound on the ratio can only be found/used when projecting non-negative functions.

Using the facts that the SH basis functions are closed under rotation, and projection is always a sum of delta functions, we can focus on delta functions in the Z frame, and limit the analysis to the ZH basis functions. They always have a maximal absolute value in $+/-Z$, so maximal constructive interference comes from simply summing the same direction, any other direction will be at most equal.

The SH projection of a delta function in Z scaled by s has ZH coefficients for band l that is $s\sqrt{\frac{2l+1}{4\pi}}$, if we divide this by the projection of DC, $l = 0$ the scale factor and denominators cancel out and you are left with $\sqrt{2l+1}$, which is the previous upper bound [4]. Since the norm of the coefficients in a band is rotationally invariant we have a bound of the length of each band when divided by DC: $\sum_{m=-l}^{m=l}(\frac{b_l^m}{b_0^0})^2 <= 2l+1$, this is a useful constraint to use when computing SH not through projection, for example when solving for SH that when interpolated explain lighting over a surface, you can use this constraint to limit the SH to be ones that come from non-negative lighting, the only ones you are trying to represent. It is important to note that the resulting SH can still be negative, the only constraint is that they are derived from the projection of non-negative functions.

An analogy to see why this works is think of a photograph or a room under different lighting conditions. Dividing by the average value over the image (DC) is effectively picking an exposure that is reasonable-ish for each of the lighting conditions. The analogy works better if you think of a low pass filtered version of the image, this filtering will bound the derivatives in a manner closer to how SH have limits on the derivatives. What happens if we have negative values in the image? That can drive the average value arbitrarily small, and increase the dynamic range of the pixels, the opposite of what we are striving for.

The maximal value comes when projecting a delta function, not common when encoding indirect lighting. In that same presentation it is mentioned that taking the square root after $|x|$ is in $[0, 1]$ and then squaring preserving the sign when reconstructing reduces the L^1 error by 50%, this is only used when not interpolating the coefficients due to the stronger non-linearity in reconstruction. The reason this improves accuracy is that the histograms of values is clumped towards zero.

So what was missing from the earlier bound? It turns out that while it is tight for the linear functions, in that every basis function can reach the maximum, for higher orders this is not the case. One way to think about this is that the per band rotation matrices for SH induced by a rotation in $\mathbb{R}^3$ is a sub-space for all bands above linear. So there is no rotation of a delta function in the Z frame that has a single non-zero value for the other basis functions. Similarly the only points on the sphere where the quadratic basis functions have only one non-zero value are $+/-Z$. This fact lead us to search for a tighter bound for the non-zonal quadratic coefficients.

3 NEW BOUND

3.1 Numerical Bound

We initially investigated the bound numerically. This was done by evaluating the Spherical harmonics at 90301 positions on the unit sphere using a python script. The maximum values after dividing by DC are shown in Table 2. This results in a numerical upper bound, roughly 1.9365

for the non-zonal quadratic functions. Next we will compute an exact value for this tighter upper bound.

X	Y	Z	Y_2^{-2}	Y_2^{-1}	Y_2^0	Y_2^1	Y_2^2
0.7075	0.0000	-0.7067	0.0000	0.0000	0.5569	1.9365	0.9695
0.0000	0.7075	-0.7067	0.0000	1.9365	0.5569	0.0000	-0.9695
-0.7075	0.0000	-0.7067	-0.0000	0.0000	0.5569	-1.9365	0.9695
-0.0000	-0.7075	-0.7067	0.0000	-1.9365	0.5569	-0.0000	-0.9695
0.7075	-0.0000	-0.7067	-0.0000	-0.0000	0.5569	1.9365	0.9695
1.0000	0.0000	0.0000	0.0000	-0.0000	-1.1180	-0.0000	1.9365
0.6997	0.7145	0.0000	1.9361	-0.0000	-1.1180	-0.0000	-0.0406
0.0000	1.0000	0.0000	0.0000	-0.0000	-1.1180	-0.0000	-1.9365
-0.6997	0.7145	0.0000	-1.9361	-0.0000	-1.1180	0.0000	-0.0406
-0.7145	0.6997	0.0000	-1.9361	-0.0000	-1.1180	0.0000	0.0406
-1.0000	0.0000	0.0000	-0.0000	-0.0000	-1.1180	0.0000	1.9365
-0.7145	-0.6997	0.0000	1.9361	0.0000	-1.1180	0.0000	0.0406
-0.6997	-0.7145	0.0000	1.9361	0.0000	-1.1180	0.0000	-0.0406
-0.0000	-1.0000	0.0000	0.0000	0.0000	-1.1180	0.0000	-1.9365
0.6997	-0.7145	0.0000	-1.9361	0.0000	-1.1180	-0.0000	-0.0406
0.7145	-0.6997	0.0000	-1.9361	0.0000	-1.1180	-0.0000	0.0406
1.0000	-0.0000	0.0000	-0.0000	0.0000	-1.1180	-0.0000	1.9365

Table 2. *The results of a python implementation of Cartesian coordinate spherical harmonic evaluation divided by DC.*

3.2 Analytic Bound

These numerical values are close to where two values equal $\frac{1}{\sqrt{2}}$. Plotting the spherical harmonics in spherical coordinates as shown in Figure 2 each function has a maximum at these positions. This can be proven using the derivative test. Y_2^{-2} at $\theta = \frac{\pi}{2}, \phi = \frac{\pi}{4}$. The first derivatives are 0, the determinant of the Hessian is $\frac{15}{2\pi}$, and the unmixed second partial derivatives are both negative, indicating a maximum.

A maximum of $\frac{\sqrt{15}}{2}$ for Y_2^1 can be shown analytically by evaluating $[\frac{1}{\sqrt{2}} \frac{-1}{\sqrt{2}}, 0]$ in the Cartesian coordinate version of Y_2^1 after dividing by DC. This process can be similarly executed for Y_2^{-2} and Y_2^{-1}, or by evaluating Y_2^2 at $[1, 0, 0]$.

3.3 Results

The new bound was tested on PBR_White_Box showing improvement between 2% and 9% as shown in table 3. Figure 3 shows this is due to the more even distribution of data after compression. A Shader Toy https://www.shadertoy.com/link with the compression and decompression code is online.

4 CONCLUSIONS AND FUTURE WORK

Encoding non-negative SH projections dividing by DC enables the use of lower precision formats. While a data driven bound could be used, it would require extra constant slots and more shader instructions during reconstruction. A previous bound was tight for the linear SH projection coefficients, but only tight for the zonal quadratic function. We derived a tighter analytic bound for the non-quadratic coefficients.

Function	$\sqrt{5}$	$\frac{\sqrt{15}}{2}$	Δ error	sqrt $\sqrt{5}$	sqrt $\frac{\sqrt{15}}{2}$	Δ error
L^1 Error	1291.382	1232.067	4.593%	720.100	704.648	2.146%
L^2 Error	27.757	26.269	5.361%	9.717	9.508	2.151%
RMSE	0.067	0.066	1.493%	0.040	0.039	5%

Table 3. *The errors after compression from PBR_White_Box. Comparing the old bound and new bound, with and without square root quantization.*

Listing 1. *glsl compression and decompression function.*

```
// pack 4 floating point numbers that are in [-1,1] to be in [0,2*bitVal], optionally computing a sqrt, s
uint Pack4( in vec4 v, in float bv, bool isSqrt )
{
    if ( isSqrt ) v = sign(v)*sqrt(abs(v));
    v = vec4(bv+0.5f, bv+0.5f, bv+0.5f, bv+0.5f ) + v * bv; // round
    return uint(v.x) | (uint(v.y)<<8) | (uint(v.z)<<16) | (uint(v.w)<<24);
}

void CompressSH( in float sh[9], in int bits, in float nonZbound,
                 in bool sq, in bool isIrrad, out uint bo[2] )
{
    float bv = float((1<<(bits-1))) - 1.0f;
    float lc = 1.0f/(sh[0] * sqrt(3.0f) * (isIrrad?2.0f/3.0f:1.0f));
    float qz = 1.0f/(sh[0] * sqrt(5.0f) * (isIrrad?0.25f:1.0f));
    float nz = 1.0f/(sh[0] * nonZbound * (isIrrad?0.25f:1.0f));

    bo[0] = Pack4(vec4(sh[1],sh[2],sh[3],sh[4]) * vec4(lc,lc,lc,nz), bv, sq);
    bo[1] = Pack4(vec4(sh[5],sh[6],sh[7],sh[8]) * vec4(nz,qz,nz,nz), bv, sq);
}

// unpack the results from pack4 - squaring on reconstruction
vec4 Unpack4( in uint r, in float bv, bool isSqrt )
{
    const uint m8 = 0xFFu;
    vec4 tmp = vec4(r&m8, (r>>8)&m8, (r>>16)&m8, (r>>24) )/bv;
    tmp -= vec4(1.0f,1.0f,1.0f,1.0f);
    if ( isSqrt ) tmp = sign(tmp)*(tmp*tmp);
    return tmp;
}

void UncompressSH( in uint shBits[2], in float DC, in int bits, in float nzb,
                   in bool isSqrt, in bool isIrrad, out float shOut[9])
{
    float bv = float((1<<(bits-1))) - 1.0f;
    float lc = DC * sqrt(3.0f) * (isIrrad?2.0f/3.0f:1.0f);
    float qz = DC * sqrt(5.0f) * (isIrrad?0.25f:1.0f);
    float nz = DC * nzb * (isIrrad?0.25f:1.0f);
    vec4 shA = Unpack4( shBits[0], bv, isSqrt) * vec4(lc,lc,lc,nz);
    vec4 shB = Unpack4( shBits[1], bv, isSqrt) * vec4(nz,qz,nz,nz);;

    shOut[0] = DC;
    shOut[1] = shA.x; shOut[2] = shA.y; shOut[3] = shA.z; shOut[4] = shA.w;
    shOut[5] = shB.x; shOut[6] = shB.y; shOut[7] = shB.z; shOut[8] = shB.w;
}
```

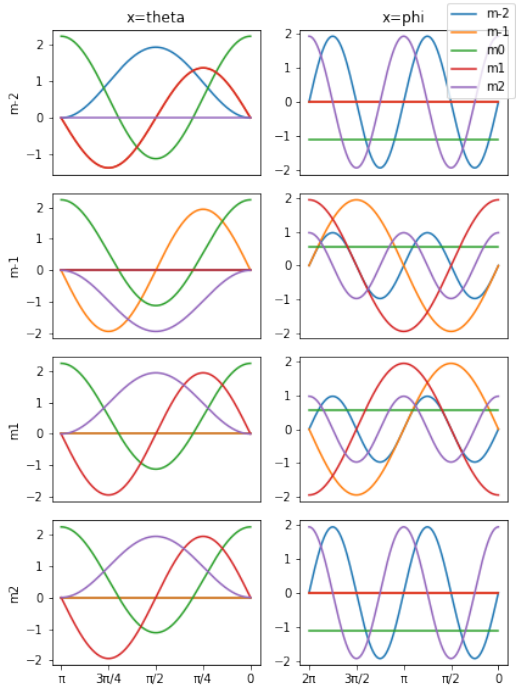

Fig. 2
A plot of values of quadratic SH in spherical coordinates.

The square root encoding is used in our code base for SH loaded from buffers, but due to concerns around potential interpolation artifacts is not used in representations that interpolate textures. The divide by DC is itself a non-linearity, and studying the impact of that more carefully is a potential avenue for future work. While bands above quadratic are rarely used in video games, deriving a tighter upper bound for those is also of interest, along with more ways to exploit the total L^2 bound across a band. That is currently used when solving for SH, to not generate functions that are not possible from non-negative projection, but possibly has other uses.

ACKNOWLEDGMENTS

Ari Silvennoinen pointed out that a tighter quadratic bound exists. Michal Iwanicki for feedback on an earlier draft.

REFERENCES

[1] Ravi Ramamoorthi and Pat Hanrahan. 2001. An efficient representation for irradiance environment maps. In *SIGGRAPH 2001 Conference Proceedings*.

[2] Zhong Ren, Rui Wang, John Snyder, Kun Zhou, Xinguo Liu, Bo Sun, Peter-Pike Sloan, Hujun Bao, Qunsheng Peng, and Baining Guo. 2006. Real-Time Soft Shadows in Dynamic Scenes Using Spherical Harmonic Exponentiation. In *ACM SIGGRAPH 2006 Papers*. 977–986.

[3] Peter-Pike Sloan. 2008. Stupid Spherical Harmonics (SH) Tricks. In *Game Developers Conference*.

[4] Peter-Pike Sloan and Ari Silvennoinen. 2020. Precomputed Lighting Advances in Call of Duty: Modern Warfare. In *SIGGRAPH 2020 Course: Advances in Real-Time Rendering in 3D Graphics and Games*.

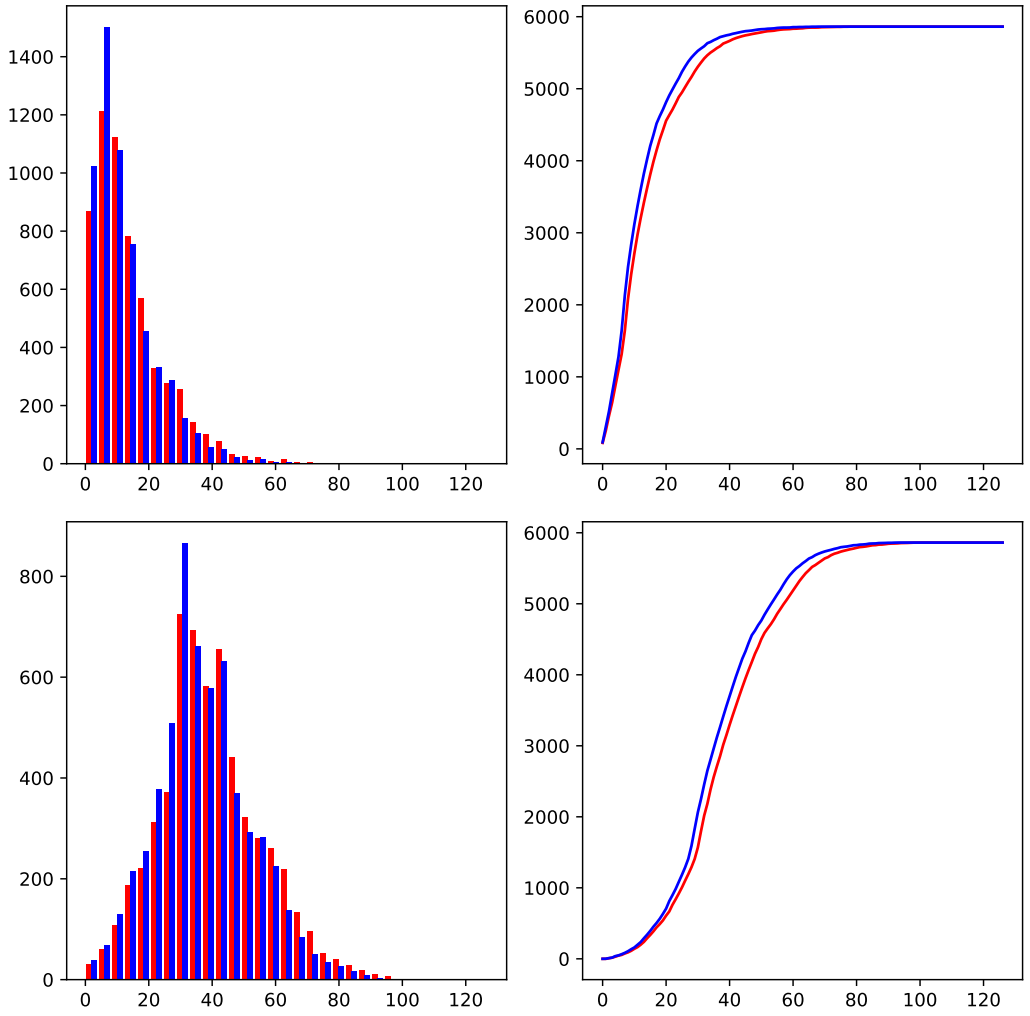

Fig. 3. *The red channel of the Y_2^{-2} from PBR_White_Box. Old bound is blue, new is red. Left is histogram of values, right is prefix sum. Bottom uses square root quantization.*