Spherical Harmonics Lighting Estimation

Jing Yang, Joe Yu-Ho Chang and Junying Wang

Fig. 1. Above are pictures of final rendering outputs, we can compare these pictures pair by pair (left to right). Based on SH method, our light engine can render the loaded sphere model successfully . We can see the lights projected on the surface of the shpere, and they will change when we rotate the environment.

Abstract

Spherical harmonics (SH) are special functions defined on the surface of a sphere. They are often employed in solving partial differential equations in many scientific fields. In our project, we used spherical harmonics to implement spherical light estimation. Generally, reconstructing environmental lights is time consuming, since we need to sample all the points from all the direction for each vertex of our target model. The cost would be expensive for large scenes because lighting on each vertex may differ from one another. Compared to this traditional lighting and rendering method, our lighting engine based on SH can calculate, reproduce environmental light and render the target models in real time, which is an intuitive method to do real time global illumination.

Keywords—spherical harmonics, light estimation, real-time rendering, global illumination.

I. INTRODUCTION

Spherical Harmonics (SH) lighting is one of the real-time rendering techniques, which can be used for producing highly realistic shading and shadowing with comparatively little overhead. All SH lighting techniques involve replacing parts of standard lighting equations with spherical functions that have been projected into frequency space using the spherical harmonics as a basis. In our basic SH engine, we used cubemaps for environment mapping. We simplified the expensive environmental lights calculation by reducing the environmental light data into SH coefficients if preserving high-frequency detail is not a concern. With our lighting engine, we can render our target model with desired environmental lights effectively.

II. RELATED WORK

Spherical Harmonics have always been around for quite some time. In 1992, Westin et al.[1] first presented a method of predicting reflectance functions from complex surfaces. He caught the public's eyes since his introduction was an efficient way of creating realistic and interactive indirect lighting rendering via Pre-computed Radiance Transfer (PRT). In the early 2000's, Ravi Ramamoorthi[2] and Peter-Pike Sloan[4] introduced a new powerful and amazing method, which made great contributions to the Computer Graphics society: Spherical Harmonics.

SH have interesting properties regarding their orthogonality, parity, symmetry and rotation. An excellent source of information is "Spherical Harmonics Lighting: the Gritty Details" by Robin Green[5] that actually covers the practical use of SH for Computer Graphics, it's a well-explained extension of the original work done by Peter-Pike Sloan about Pre-computed Radiance Transfer (PRT). From Ramamoorthi and Hanrahan's paper, we got to know that both radiance and irradiance are related in terms of SH. Then, it was shortly followed by another seminal paper called "An Efficient Representation for Irradiance Environment Maps"[3] that extends on the first paper and shows

that order 2 SH (9 coefficients) is often enough to properly represent the irradiance field surrounding an object since additional order disappear very quickly.

By searching and following these related paper, we present an implementation of building a lighting engine based on SH. For the implementation part, we first construct the SH coefficients and encode, decode, convolve signals using SH.

III. IMPLEMENTATION

A. Algorithm

Spherical Harmonics, also known as Laplace's Spherical Harmonics, are special functions defined on the surface of a sphere. They satisfy Laplace's equation and are represented in the spherical coordinate system.

The functions are denoted as: $Y_l^m(\theta, \phi)$ where l represents degree and m is a range from l to $-l$.

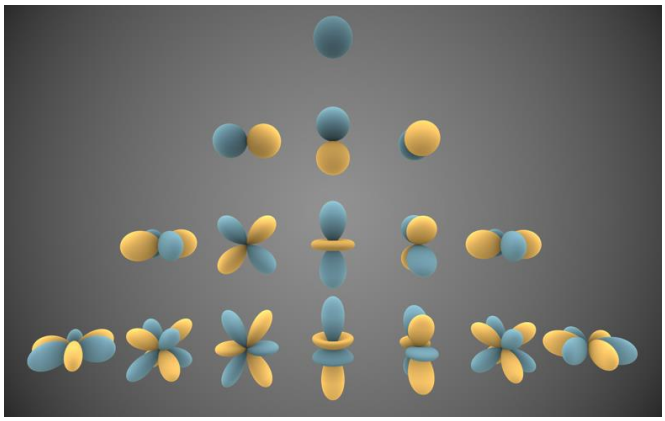

Fig. 2. Visual representations of the first few real spherical harmonics. Blue portions represent regions where the function is positive, and yellow portions represent where it is negative.

The real spherical harmonics can be expressed in spherical coordinates. The formulas are as follows:

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

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

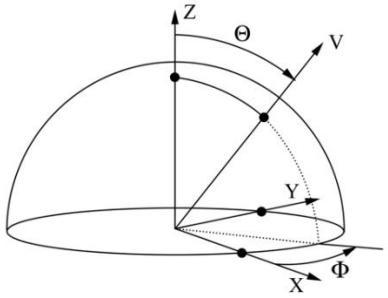

Fig. 3. Shows what is the sphere coordinates

As we can see from the formulati above: K_l^m is a normalization factor and P_l^m is the associated Legendre polynomial.

B. Pipeline

1) Sampling

- Get samples from the cubemap

First of all, in our project, we prepared a set of different cubemaps, all the light samples are based on these cubemaps. These cubemaps are as follows:

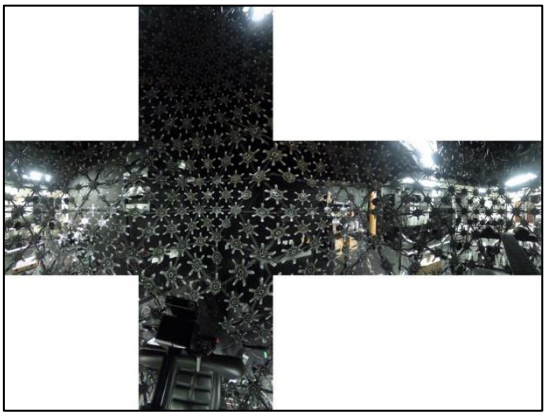

Fig. 4. Skybox of the lightstage

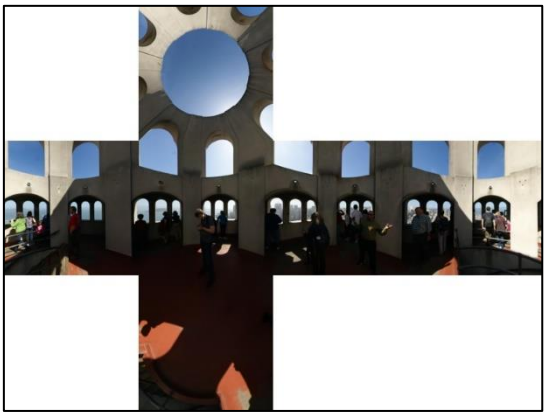

Fig. 5. Skybox of Coti Tower.

Secondly, we sample RGB light values from our cubemap. The number of samples ranged from 100 to 100,000. The following pictures demonstrate the sampled points from the cubemap.

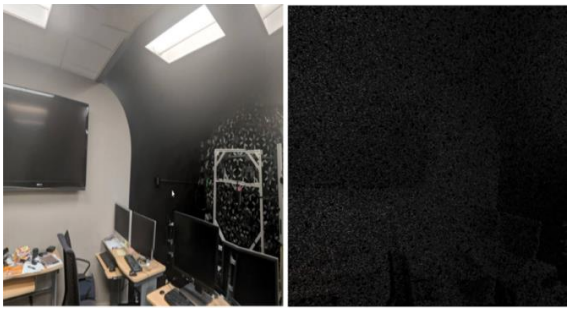

Fig. 6. Left side shows one part of the original cubemap, the right side shows the sample points that we can from the cubemap.

2) Coefficients Evaluation

- Calculate Spherical Harmonics Lighting coefficients using samples and SH equation

$$c_i = \frac{4\pi}{N}\sum_{j=1}^{N} L(x_j)Y_i(x_j)$$

The equation above is the core of our evaluation. In the formula, N represents the number of samples we selected, function L is the light data we retrieved, and function Y is the basis coefficient function.

For example: $Y_0^0(\theta,\phi) = \frac{1}{2}\sqrt{\frac{1}{\pi}}$ is the basis when degree is 0.

When the degree is 1 and 2, we can get these formulas as follows:

$$\begin{cases} Y_1^{-1} = \frac{1}{2}\sqrt{\frac{3}{\pi}}y & m = -1 \\ Y_1^0 = \frac{1}{2}\sqrt{\frac{3}{\pi}}z & m = 0 \\ Y_1^1 = \frac{1}{2}\sqrt{\frac{3}{\pi}}x & m = 1 \end{cases}$$

Where:

$$\begin{cases} x = \sin(\theta)\cos(\phi) \\ y = \sin(\theta)\sin(\phi) \\ z = \cos(\theta) \end{cases}$$

3) Shading

- Pass the SH coefficients into the fragment shader
- Estimate the original environmental lighting

$$L'(s) = \sum_{i=1}^{n^2} c_i Y_i(s)$$

In order to retrieve the estimated lighting for shading, we used the equation above. Function L means the reconstructed light and it is the summation of the coefficient from the previous step times the basis coefficient function from 1 to n^2, where n is 4 in our project since we used a degree of 3.

4) Rendering

- Render the 3D model with the estimated lighting

IV. RESULT

Via user studies, our lighting engine can render different models in high qualities. We tested our engine with different number of samples, different backgrounds and differnet models. As you can see as follows:

A. Test on Different Models

We tested our method on different 3D models, such as Sphere, Cube, Donut, Suzanne the Monkey and the Poke ball. The final reprojected lights are in high qualities and we can render different models in real time.

As *Fig 6* shows, we can compare the final outputs of differnet models line by line: we tested our 3D models in same light environment, and our engine can render all these models with corret lighting in high qualities. And the whole process can be achieved in real time.

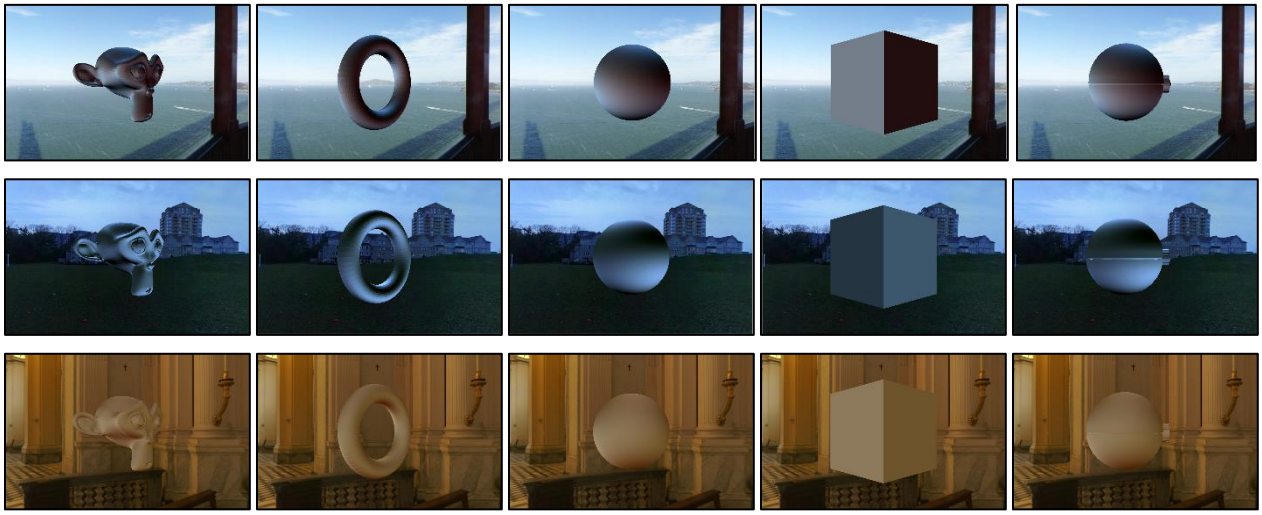

Fig. 7. These pictures above present the final outputs of different test models.

B. *Test with Different Number of Samples and Different Backgrounds*

We test our engine with different number of samples, the more samples we use, the higher rendering quality we get.

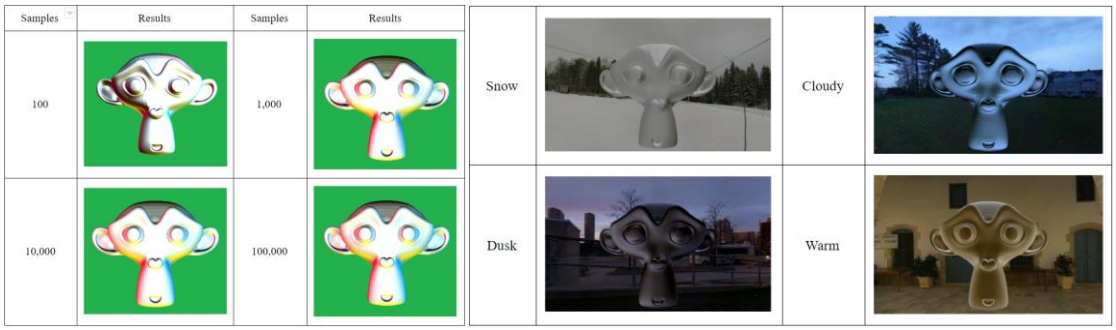

Fig. 8. Left side shows the final results with different number of samples, while right side shows the same model tested in different backgrounds(same sampling)

V. LIMITATIONS AND FUTUREWORK

Firstly, even though SH functions can be used to approximate the rendering equation with only a few coefficients and a simple dot product to evaluate lighting during run time, they also disadvantage -- SH can only approximate low frequency function as this method needs a large number of bands to represent high frequency details.

Secondly, SH lighting using preprocessed coefficients produces results that are beautiful but limited. Typically the lighting can change, or the lit mesh can be rotated, but the mesh can't be translated or deformed without requiring a new set of per-vertex coefficients. More recent techniques split the lighting equation into more parts and introduce techniques for updating SH components in real time.

ACKNOWLEDGMENT

Thanks to Prof. Ulrich Neumann's instructions and great teaching. CSCI 580 is definitely a prefect and well-organized graphics class for doing some more practical and fancier things. Thanks to our team for the great contributions. And also thanks to all the classmates enrolled in CSCI 580 class.

REFERENCES

[1] (1992) "Predicting Reflectance Functions from Complex Surfaces" Stephen Westin et al.

[2] (2001) "On the relationship between radiance and irradiance: determining the illumination from images of a convex Lambertian object" Ravi Ramamoorthi and Pat Hanrahan

[3] (2001) "An Efficient Representation for Irradiance Environment Maps" Ravi Ramamoorthi and Pat Hanrahan

[4] (2002) "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments" Peter-Pike Sloan et al.

[5] (2003) "Spherical Harmonics Lighting: the Gritty Details" Robin Green