Spherical Lighting with Spherical Harmonics Hessian

KEI IWASAKI, Saitama University, Japan and Prometech CG Research, Japan
YOSHINORI DOBASHI, Hokkaido University, Japan and Prometech CG Research, Japan

In this paper, we introduce a second-order derivative of spherical harmonics, *spherical harmonics Hessian*, and *solid spherical harmonics*, a variant of spherical harmonics, to the computer graphics community. These mathematical tools are used to develop an analytical representation of the Hessian matrix of spherical harmonics coefficients for spherical lights. We apply our analytic representation of the Hessian matrix to grid-based SH lighting rendering applications with many spherical lights that store the incident light field as spherical harmonics coefficients and their spatial gradient at sparse grid. We develop a Hessian-based error metric, with which our method automatically and adaptively subdivides the grid whether the interpolation using the spatial gradient is appropriate. Our method can be easily incorporated into the grid-based precomputed radiance transfer (PRT) framework with small additional storage. We demonstrate that our adaptive grid subdivided by using the Hessian-based error metric can substantially improve the rendering quality in equal-time grid construction.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: Spherical Harmonics, Solid Spherical Harmonics, Spherical Harmonics Hessian, Spherical Light

ACM Reference Format:
Kei Iwasaki and Yoshinori Dobashi. 2025. Spherical Lighting with Spherical Harmonics Hessian. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3721238.3730689

1 Introduction

Spherical harmonics (SH) are indispensable mathematical tools for representing signals defined on the unit sphere, playing a crucial role in computer graphics. SH possess several nice properties, especially for rendering applications. They allow for the efficient representation and manipulation of functions over the sphere due to their compactness and orthogonality. In rendering, spherical harmonics are particularly useful for approximating illumination, such as environment maps or diffuse lighting. Their ability to encode low-frequency details with relatively few coefficients makes them well-suited for real-time and precomputed rendering techniques.

Physically-based rendering often requires the integral of spherical functions over specific domains. For example, reflected radiance from direct illumination of spherical lights or polygonal lights involves the integration of cosine-weighted Bidirectional Reflectance Distribution Function (BRDF) over spherical domains or polygonal domains. To compute such integrals, traditional rendering methods mainly rely on Monte-Carlo integration at the cost of introducing noise as a variance.

Recent advancements in SH lighting have shown that SH have closed-form solutions of the integral over spherical/polygonal domains [Belcour et al. 2018; Wang and Ramamoorthi 2018]. Furthermore, analytic representation of the spatial gradient of SH coefficients have also been proposed [Mézières et al. 2022; Wu et al. 2020]. These abilities promote the application of SH lighting to many light rendering that accumulates the SH coefficients and their spatial gradient at grid points. The incoming illumination at points to be shaded (referred to as shading points) is then interpolated by using the accumulated SH coefficients and gradients. While these methods achieve significant speed-ups for scenes with many lights, previous methods [Mézières et al. 2022; Wu et al. 2020] use *uniform* grids to store SH coefficients and gradients and are *unaware of the interpolation errors*. Uniform grids can over-subdivide regions with small interpolation errors (e.g., regions far from area lights), leading to unnecessary computational overhead, and under-subdivide regions with significant errors, leading to visible artifacts. Therefore, developing a mathematical tool to estimate the interpolation errors for SH lighting is desirable.

This paper introduces two mathematical tools to the computer graphics community, *spherical harmonics Hessian* and *solid spherical harmonics (SSH)*. SSH are identical to SH when evaluated for unit directions that are practical use cases in rendering applications. SSH are represented with homogeneous polynomials of the Cartesian coordinates whose first and second derivatives can be derived easily. These properties enable us to compute SH gradient and SH Hessian via SSH efficiently and accurately. Our work builds upon the recent work on SSH in the chemical physics field [Bigi et al. 2023]. Our method explicitly formulates SH Hessian via SSH and proposes a code generator for efficient and accurate computation of SH Hessian matrices, leading to up to 3x speedups compared to the previous method [Bigi et al. 2023].

In this paper, we focus on spherical lights and propose a Hessian-based error metric for SH lighting of spherical lights with uniform emittance profile. Our approach subdivides grids taking into account the distribution of spherical lights and the interpolation errors from them estimated by using analytical SH Hessian. The distribution of shading points is also considered to construct adaptive grids to eliminate unnecessary evaluations of SH Hessian matrices for empty spaces where no shading point exist. Our experimental results show that our adaptive grid can render accurate results for scenes including near and distant spherical lights from shading points, which is difficult for uniform grids.

Authors' Contact Information: Kei Iwasaki, keiiwasaki@acm.org, Saitama University, Saitama, Japan and Prometech CG Research, Tokyo, Japan; Yoshinori Dobashi, doba@ist.hokudai.ac.jp, Hokkaido University, Sapporo, Japan and Prometech CG Research, Tokyo, Japan.

SIGGRAPH Conference Papers '25, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1540-2/2025/08
https://doi.org/10.1145/3721238.3730689

2 Related Work

2.1 Spherical Harmonic Lighting

Sloan et al. [2002] proposed a precomputed radiance transfer framework that encodes distant environmental lighting and the transfer function encompassing the visibility function and the cosine-weighted BRDF in SH. Sloan et al. [2005] introduced Zonal Harmonics (ZHs), which are a subset of SHs, to render deformable objects. To extend the PRT framework, several methods have been developed to handle deformable objects [Wang et al. 2006], dynamic scenes [Iwasaki et al. 2007; Ren et al. 2006], and fast computation of multiple SH products [Xin et al. 2021].

For efficient evaluations of SH, Sloan showed an explicit polynomial representation of SH using Cartesian coordinates [Sloan 2008]. Sloan [2013] derived recurrence formulae of the associated Legendre polynomial for efficient evaluations of SH. This method proposed a code generator that exploits the recurrence formulae and outperforms the previous explicit polynomial representation [Sloan 2008]. Our method follows these lines of research and extends to generate a code to compute SH Hessian efficiently.

Recent advancements in SH lighting enable us to compute SH coefficients for near-field illumination from polygonal and spherical lights analytically. Belcour et al. [2018] proposed an analytic solution for the integral of SH over a spherical polygonal domain using Zonal Harmonics Factorization [Nowrouzezahrai et al. 2012]. Wang et al. [2018] derived a recurrence formula of the integral of Legendre polynomials over a polygonal domain. Zhou et al. [2020] proposed an analytical integration method for SH over spherical caps. While these methods can compute SH coefficients for near-field illumination analytically, their computational costs are proportional to the number of lights, which can be expensive for many-light scenes.

Instead of computing SH coefficients at each shading point for many area lights, several methods exploit the spatial coherence and smoothness of SH coefficients for computational efficiency. Annen et al. [2004] proposed a semi-analytical method for spatial gradients of SHs to interpolate SH coefficients of mid-range illumination. In recent years, efforts have been made to handle many area lights efficiently for spherical harmonic lighting [Mézières et al. 2022; Wu et al. 2020]. These methods prepare a uniform grid encompassing the entire scene and accumulate SH coefficients and spatial gradients at sparse grid points. Then, the SH coefficients at each shading point are interpolated by those stored at neighbor grid points. Wu et al. [2020] developed analytic SH gradients for polygonal lights. Mezieres et al. [2022] proposed an efficient calculation method for SH gradients of spherical lights using recurrence formulae. Unfortunately, these methods construct uniform grids without consideration of the distribution of area lights and lack a mechanism to control errors arising from the gradient-based interpolation. In contrast, our method controls such errors using SH Hessian. Our method subdivides grids adaptively based on Hessian-based error metric, which is analytically calculated using spherical harmonics Hessian.

2.2 Caching-based methods

Caching-based methods record the illumination information (e.g., irradiance and radiance) at sparsely distributed points in the scene, then the illumination information at a shading point is interpolated by those stored at nearby cache points [Jarosz et al. 2008; Krivanek et al. 2005; Křivánek et al. 2006; Ward and Heckbert 2008]. Second-order derivatives (i.e., Hessian) have been used to measure errors of cache-based methods that interpolate radiance/irradiance using first-order derivatives (i.e., spatial gradients). Schwarzhaupt et al. [2012] proposed a Hessian-based error metric for irradiance caching. Marco et al. [2018] derived Hessian-based error metrics taking into account occlusions for volumetric radiance caching. These methods calculate the error metric using Monte-Carlo estimations that can lead to variance, while our method can calculate the Hessian-based error metric analytically. SH Hessian can be used to place cache points adaptively by estimating the interpolation errors, but it remains as future work.

2.3 Solid Spherical Harmonics (SSH)

SSH have been studied in the fields of mathematics, physics, and quantum chemistry [Steinborn and Ruedenberg 1973]. Bigi et al. [2023] proposed an efficient SH evaluation method using SSH and implemented this algorithm in software called *sphericart*. They derived a first-order derivative of SSH in the paper and the second-order derivative of SSH is implemented in sphericart. Our method extends the work of Bigi et al. in the two-fold. Firstly, we propose a code generator for SHs and the first/second-order derivatives. While sphericart also provides a hard-coded implementation, it is limited to sixth order. Our method can generate codes with much higher orders. Secondly, we explicitly formulate the SH Hessian and exploit the similarities between the matrix components. Our method reuses common components that have already been computed, results in up to 3x speedups compared to sphericart.

3 Preliminary

3.1 Spherical Harmonics

The real-valued SH Y_l^m for unit direction $\omega \in \mathbb{S}^2$ are defined as:

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

where θ and ϕ are the spherical coordinates of direction ω, l denotes the order of SH and m is an integer satisfying $-l \le m \le l$, K_l^m is the normalization factor, and P_l^m is the associated Legendre polynomial.

3.2 SH coefficients for spherical lights

The emitted radiance from a spherical light $\mathbf{y}$ with radius r towards a shading point $\mathbf{x}$ is represented with its SH coefficient $L_l^m(\mathbf{x})$ [Mézières et al. 2022] as:

$$L_l^m(\mathbf{x}) = \int_{\Omega(\mathbf{x})} Y_l^m(\omega)d\omega = \Lambda_l Y_l^m(\omega_{\mathbf{xy}})\tilde{L}_l(\mathbf{x}), \quad (2)$$

where $\Omega(\mathbf{x})$ is the projected area of the spherical light $\mathbf{y}$ onto the unit sphere $\mathbb{S}^2$ at $\mathbf{x}$, $\Lambda_l = \sqrt{4\pi/(2l+1)}$, $\omega_{\mathbf{xy}} = (\mathbf{y}-\mathbf{x})/\|\mathbf{y}-\mathbf{x}\|$ is the unit direction vector from the shading point $\mathbf{x}$ to the center of the spherical light $\mathbf{y}$ as shown in Fig. 1. $\tilde{L}_l(\mathbf{x})$ is the zonal harmonics

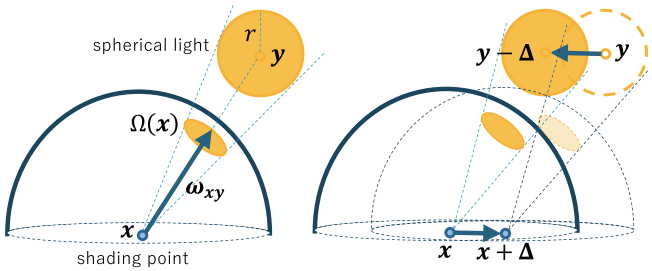


Fig. 1. Illustration of computing SH coefficient for a spherical light $\mathbf{y}$ projected onto the unit sphere centered at a shading point $\mathbf{x}$. Our method computes the spatial gradient with respect to $\mathbf{y}$ since the movement of $\mathbf{x}$ by Δ is equivalent to that of $\mathbf{y}$ by $-\Delta$. This minus sign is shown in Eq. (6).

(ZH) coefficient for the spherical light and calculated as:

$$\tilde{L}_l(\mathbf{x}) = \sqrt{\pi/(2l+1)}(P_{l-1}(\alpha) - P_{l+1}(\alpha)), \tag{3}$$

$$\alpha = \cos(a(\mathbf{x})), \tag{4}$$

$$a(\mathbf{x}) = \arcsin\left(r/\|\mathbf{y}-\mathbf{x}\|\right), \tag{5}$$

where P_l is the l-th Legendre polynomial ($P_{-1}(\alpha) = 1$ is used to compute $\tilde{L}_0(\mathbf{x})$), $a(\mathbf{x})$ is the half angle of the spherical light projected onto the unit sphere at $\mathbf{x}$.

Mezieres et al. [2022] derived an analytic calculation method of the spatial gradient $\nabla L_m^l(\mathbf{x})$ at shading point $\mathbf{x}$ as:

$$\nabla L_l^m(\mathbf{x}) = -\Lambda_l \left(\nabla Y_l^m(\omega_{\mathbf{xy}})\tilde{L}_l(\mathbf{x}) + Y_l^m(\omega_{\mathbf{xy}})\nabla\tilde{L}_l(\mathbf{x})\right), \tag{6}$$

where the spatial gradient of ZH coefficient $\nabla\tilde{L}_l(\mathbf{x})$ is calculated as:

$$\nabla\tilde{L}_l(\mathbf{x}) = \sqrt{(2l+1)\pi}\nabla a(\mathbf{x})\sin(a(\mathbf{x}))P_l(\alpha), \tag{7}$$

$$\nabla a(\mathbf{x}) = -r\omega_{\mathbf{xy}}/(\|\mathbf{y}-\mathbf{x}\|\sqrt{\|\mathbf{y}-\mathbf{x}\|^2 - r^2}). \tag{8}$$

To compute the spatial gradient ∇Y_l^m in Eq. (6), Mezieres et al. [2022] first performed the partial derivatives $\partial_\theta Y_l^m$ and $\partial_\phi Y_l^m$ with respect to the spherical coordinates (θ, ϕ), then the spatial gradient ∇Y_l^m is converted from the partial derivatives as:

$$\nabla Y_l^m(\omega_{\mathbf{xy}}) = \begin{pmatrix} \frac{xz}{\Psi} & \frac{-y}{x^2+y^2} \\ \frac{yz}{\Psi} & \frac{x}{x^2+y^2} \\ \frac{-x^2-y^2}{\Psi} & 0 \end{pmatrix} \begin{pmatrix} \partial_\theta Y_l^m \\ \partial_\phi Y_l^m \end{pmatrix}, \tag{9}$$

where $(x, y, z) = \mathbf{y} - \mathbf{x}$ and $\Psi = (x^2 + y^2 + z^2)/\sqrt{x^2+y^2}$. Hereinafter, we refer this spherical coordinate based derivative calculation method (with the conversion to Cartesian coordinate) to as *spherical coordinate derivative (SCD) method.*

Limitations. While partial derivatives $(\partial_\theta Y_l^m, \partial_\phi Y_l^m)$ are computed efficiently via recurrence formula, the spherical coordinate derivative (SCD) method has some drawbacks. Firstly, the recurrence formulae have a singularity at $\theta = 0$ since they involve division by $\sin\theta$ (see [Mézières et al. 2022] and our supplemental document.). The conversion from spherical coordinates (θ, ϕ) for spatial gradients ∇Y_l^m also has the same singularity, since the conversion matrix in Eq. (9) includes division by $\sqrt{x^2+y^2} = \sin\theta$. Furthermore, this conversion introduces undesirable computational overhead, which becomes more problematic when computing the SH Hessian, as the conversion matrix for the Hessian increases in size (6×5). Since the Hessian is evaluated repeatedly at all grid points, we propose a faster computation method to address this issue.

3.3 Solid Spherical Harmonics

Solid Spherical Harmonics (SSH) $\tilde{Y}_l^m(x, y, z)$ are variants of SH Y_l^m scaled by the radial distance $r = \sqrt{x^2+y^2+z^2}$ as [Bigi et al. 2023]:

$$\tilde{Y}_l^m(x, y, z) = r^l Y_l^m(\omega) = r^l Y_l^m(x/r, y/r, z/r). \tag{10}$$

Specifically, SSH $\tilde{Y}_l^m(x, y, z)$ are defined as the following equation:

$$\tilde{Y}_l^m(x, y, z) = \begin{cases} \sqrt{2}K_l^{-m}Q_l^{-m}(z, r)s_{-m}(x, y) & (m < 0) \\ K_l^0 Q_l^0(z, r) & (m = 0) \\ \sqrt{2}K_l^m Q_l^m(z, r)c_m(x, y) & (m > 0) \end{cases}, \tag{11}$$

where Q_l^m, s_m, and c_m are defined as:

$$\begin{aligned} Q_l^m(z, r) &= r^l r_{xy}^{-m} P_l^m(\cos\theta), \\ s_m(x, y) &= r_{xy}^m \sin(m\phi), \quad c_m(x, y) = r_{xy}^m \cos(m\phi), \end{aligned} \tag{12}$$

where $r_{xy} = \sqrt{x^2+y^2}$. Q_l^m is the polynomial function of z and r, and is calculated from the following recurrence formulae:

$$\begin{aligned} Q_l^l &= (1-2l)Q_{l-1}^{l-1}, \quad (Q_0^0 = 1) \\ Q_l^{l-1} &= -zQ_l^l, \\ Q_l^m &= \frac{(2l-1)zQ_{l-1}^m - (l+m-1)r^2Q_{l-2}^m}{l-m}. \end{aligned} \tag{13}$$

s_m and c_m are also the polynomial functions of x and y, and are calculated from the following recurrence formulae:

$$s_m = xs_{m-1} + yc_{m-1}, \quad c_m = xc_{m-1} - ys_{m-1}, \tag{14}$$

where $s_0 = 0$ and $c_0 = 1$ are used as the initial values.

4 Proposed Method

The key to efficiently computing the Hessian matrix is the introduction of SSH, which also eliminates the singularity problem mentioned in Sec. 3.2. The Hessian matrix $H_l^m \in \mathbb{R}^{3\times3}$ of SH coefficient $L_l^m(\mathbf{x})$ for the spherical light $\mathbf{y}$ is defined as (arguments are omitted):

$$H_l^m = \Lambda_l \left(\tilde{L}_l\mathcal{H}Y_l^m + \nabla Y_l^m\nabla^\top\tilde{L}_l + \nabla\tilde{L}_l\nabla^\top Y_l^m + Y_l^m\mathcal{H}\tilde{L}_l\right), \tag{15}$$

where $\mathcal{H}f = \nabla\nabla^\top f$ is the Hessian operator of a scalar function f. Since the gradient $\nabla\tilde{L}_l$ is already derived in Sec. 3.2, we focus on the derivation of the gradient ∇Y_l^m, Hessians $\mathcal{H}\tilde{L}_l$ and $\mathcal{H}Y_l^m$.

We employ SSH to compute the gradients and Hessians of SH because SSH is defined in Cartesian coordinates.This Cartesian formulation allows the spatial gradient to be computed directly, bypassing the need for conversion from spherical coordinates, which improves computational efficiency. Moreover, this approach avoids the singularity problem encountered in the previous method [Mézières et al. 2022], thereby ensuring more robust computations.

4.1 SH gradient ∇Y_l^m using SSH

By applying the gradient operator to both sides of Eq. (10), we obtain the following expression:

$$\nabla \tilde{Y}_l^m(\omega) = l r^{l-1} Y_l^m(\omega)\omega + r^l \nabla Y_l^m(\omega), \tag{16}$$

where $\omega = (x, y, z)$. By putting $r = 1$, we obtain the expression for computing the gradient of SH using SSH as:

$$\nabla Y_l^m(\omega) = -l Y_l^m(\omega)\omega + \nabla \tilde{Y}_l^m(\omega). \tag{17}$$

The gradient of SSH for positive $m > 0$ is calculated as:

$$\nabla \tilde{Y}_l^m = \sqrt{2} K_l^m \begin{pmatrix} x Q_{l-1}^{m+1} c_m + m Q_l^m c_{m-1} \\ y Q_{l-1}^{m+1} c_m - m Q_l^m s_{m-1} \\ (l+m) Q_{l-1}^m c_m \end{pmatrix}. \tag{18}$$

The gradient of SSH $\tilde{Y}_l^m$ for $m \leq 0$ are represented in the similar way as shown in the supplemental material.

As mentioned in the previous section, SSH is defined in Cartesian coordinates. Thus, the gradient of SH, computed using Eq. (17), can be evaluated directly in Cartesian coordinates. For computing the gradient of SH coefficients (see Eq. (6)), we assume that the SH component $Y_l^m(\omega)$ has already been computed. Consequently, the additional cost of computing the gradient of SH is limited to the computation of the gradient of SSH, $\nabla Y_l^m(\omega)$, which reduces the overall computational cost.

4.2 SH Hessian $\mathcal{H} Y_l^m$ using SSH

In the similar way to the gradient of SH, we apply the Hessian operator, $\mathcal{H}$, to Eq. (17). Then, SH Hessian $\mathcal{H} Y_l^m$ is calculated from the SSH Hessian $\mathcal{H} \tilde{Y}_l^m$ and the identity matrix I as:

$$\mathcal{H} Y_l^m = \mathcal{H} \tilde{Y}_l^m - l(Y_l^m(I + (l-2)\omega\omega^\top) + \nabla Y_l^m \omega^\top + \omega \nabla^\top Y_l^m).$$

The second derivative of SSH in Cartesian coordinates is computed analytically. For example, the second derivative of SSH $\partial_{xx} \tilde{Y}_l^m$ for positive m is calculated as:

$$\begin{aligned} \partial_{xx} \tilde{Y}_l^m = \sqrt{2} K_l^m (x^2 Q_{l-2}^{m+2} c_m + Q_{l-1}^{m+1}(2mx c_{m-1} + c_m) \\ + m(m-1) Q_l^m c_{m-2}). \end{aligned} \tag{19}$$

Other components are computed by using Q_l^m in the similar way and the details are shown in the supplemental material. Note that we can reuse Q_l^m that have already been calculated to evaluate $\tilde{Y}_l^m$ and $\nabla \tilde{Y}_l^m$ for evaluating $\mathcal{H} \tilde{Y}_l^m$ to save the computational cost.

4.3 SH Hessian Code Generator

We propose a code generator to compute the SH Hessian matrix $\mathcal{H} Y_l^m$. Our code generator aims to reuse the previously computed components for efficient computation. Let us explain the implementation details for $m = 0$ case, and other cases are shown in the supplemental material. The explicit formulae for the first/second derivatives of SH Y_l^0 are represented as:

$$\begin{aligned} \partial_x Y_l^0 &= (K_l^0 Q_{l-1}^1 - l Y_l^0)x, \quad \partial_y Y_l^0 = (K_l^0 Q_{l-1}^1 - l Y_l^0)y, \\ \partial_{xx} Y_l^0 &= (K_l^0 Q_{l-2}^0 - l(l-2) Y_l^0)x^2 - 2(\partial_x Y_l^0) l x + K_l^0 Q_{l-1}^1 - l Y_l^0, \\ \partial_{xy} Y_l^0 &= (K_l^0 Q_{l-2}^0 - l(l-2) Y_l^0)xy - (\partial_x Y_l^0) l y - (\partial_y Y_l^0) l x, \\ \partial_{yy} Y_l^0 &= (K_l^0 Q_{l-2}^0 - l(l-2) Y_l^0)y^2 - 2(\partial_y Y_l^0) l y + K_l^0 Q_{l-1}^1 - l Y_l^0. \end{aligned}$$

The coefficient $K_l^0 Q_{l-1}^1 - l Y_l^0$ for the first derivatives $\partial_x Y_l^0$ and $\partial_y Y_l^0$ can be shared, and reused for the constant term in $\partial_{xx} Y_l^0$ and $\partial_{yy} Y_l^0$. Our code generator reuses the coefficient $K_l^0 Q_{l-2}^0 - l(l-2) Y_l^0$ for the monomial terms x^2, xy, and y^2.

4.4 Hessian matrix of ZH coefficient $\mathcal{H} \tilde{L}_l$

The computation of the Hessian matrix $\mathcal{H} \tilde{L}_l$ for ZH coefficients in spherical lights is rather straightforward. It is calculated by differentiating $\nabla \tilde{L}_l$ in Eq. (3) as:

$$\begin{aligned} \mathcal{H} \tilde{L}_l = \sqrt{\pi(2l+1)} \Big((\alpha P_l(\alpha) + (\alpha^2 - 1) P_l'(\alpha)) \nabla a(\mathbf{x}) \nabla^\top a(\mathbf{x}) \\ + \sin(a(\mathbf{x})) P_l(\alpha) \mathcal{H} a(\mathbf{x}) \Big), \end{aligned} \tag{20}$$

where the Hessian matrix $\mathcal{H} a(\mathbf{x})$ is calculated as:

$$\mathcal{H} a(\mathbf{x}) = r \left(\frac{\omega_{\mathbf{xy}} \omega_{\mathbf{xy}}^\top}{(\|\mathbf{y} - \mathbf{x}\|^2 - r^2)^{3/2}} - \frac{I - 2\omega_{\mathbf{xy}} \omega_{\mathbf{xy}}^\top}{\sqrt{\|\mathbf{y} - \mathbf{x}\|^2 - r^2} \|\mathbf{y} - \mathbf{x}\|^2} \right).$$

5 Adaptive Grid using Hessian-based Error Metrics

We apply SSH Hessian to the grid-based interpolation method of SH coefficients for spherical lights [Mézières et al. 2022]. Our method uses the Hessian matrix H_l^m in Eq. (15) to estimate the interpolation errors of SH coefficients $L_l^m(\mathbf{x})$. Let us consider that a voxel V in the grid and each grid point $\mathbf{x}$ stores the SH coefficient $L_l^m(\mathbf{x})$ and the spatial gradient ∇L_l^m. The SH coefficient at a point $\mathbf{x} + \Delta\mathbf{x}$ in the voxel V is interpolated by using $L_l^m(\mathbf{x}) + \nabla L_l^m \Delta\mathbf{x}$. The difference between the true SH coefficient $L_l^m(\mathbf{x} + \Delta\mathbf{x})$ at $\mathbf{x} + \Delta\mathbf{x}$ and the interpolated SH coefficient $L_l^m(\mathbf{x}) + \nabla L_l^m(\mathbf{x})\Delta\mathbf{x}$ can be calculated by the Hessian matrix H_l^m as:

$$|L_l^m(\mathbf{x} + \Delta\mathbf{x}) - L_l^m(\mathbf{x}) - \nabla L_l^m(\mathbf{x})\Delta\mathbf{x}| \approx \frac{1}{2} |\Delta\mathbf{x}^\top H_l^m \Delta\mathbf{x}|. \tag{21}$$

By diagonalizing the Hessian matrix H_l^m, the difference is bounded by the maximum absolute eigenvalue λ_l^m of H_l^m. Our method estimates the interpolation error e_l^m of SH coefficient L_l^m as $e_l^m = |\lambda_l^m| \|\Delta\mathbf{x}\|^2 / 2$. The absolute error $E_{abs}(\mathbf{x})$ and the relative error $E_{rel}(\mathbf{x})$ at the grid point $\mathbf{x}$ are calculated as

$$E_{abs}(\mathbf{x}) = \frac{\|\boldsymbol{\lambda}\| \|\Delta\mathbf{x}\|^2}{2}, \quad E_{rel}(\mathbf{x}) = \frac{\|\boldsymbol{\lambda}\| \|\Delta\mathbf{x}\|^2}{2\|\boldsymbol{L}\|}, \tag{22}$$

where $\|\boldsymbol{\lambda}\|$ is the L2-norm of a vector $\boldsymbol{\lambda}$ that consists of the maximum absolute eigenvalues $\boldsymbol{\lambda} = [\lambda_0^0, \lambda_1^{-1}, \cdots]$, and $\boldsymbol{L}$ is the SH coefficient vector $\boldsymbol{L} = [L_0^0, L_1^{-1}, \cdots]$.

The absolute error and the relative error for each voxel V are the mean values of the corresponding errors stored at its adjacent eight grid points with $\Delta\mathbf{x}$ the diagonal length of the voxel V. Our method subdivides the voxel when the interpolation error of the voxel exceeds the user-defined threshold. Please note that our method only stores the L2-norm $\|\boldsymbol{\lambda}\|$ at each grid point that requires the additional storage $O(1)$, instead of storing the Hessian matrices with the maximum order l_{max} that requires the additional storage $O(l_{max}^2)$.

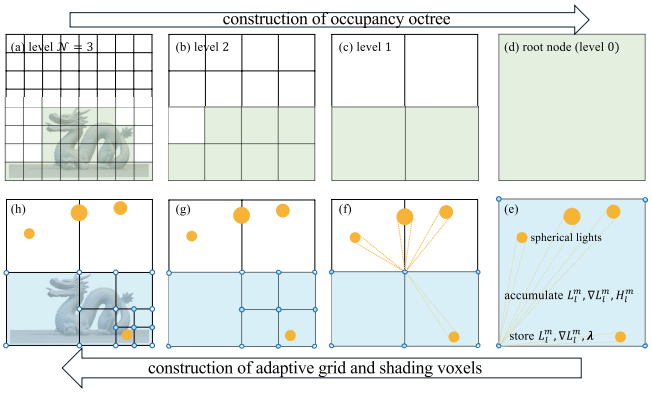


Fig. 2. Construction of an occupancy octree (top) and an adaptive grid (bottom) with shading voxels (blue). Nodes (green) including shading points are marked as *occupied*. Voxels whose corresponding nodes in the occupancy octree are marked as occupied are shading voxels.

5.1 Implementation Details

To estimate the interpolation errors of SH coefficients, our method requires the calculation of the Hessian matrix H_l^m that incurs additional computational costs to the previous gradient-based method [Mézières et al. 2022]. To reduce the computational costs, our method first detects the voxels including the shading points (referred to as *shading voxels*), then SH coefficients L_l^m, the gradients ∇L_l^m, and the Hessian matrices H_l^m of all spherical lights are accumulated only for the grid points of the shading voxels.

Specifically, to construct an adaptive grid with shading voxels, our method prepares an auxiliary grid referred to as the *occupancy octree*, as shown in Fig. 2. The occupancy octree is constructed in a bottom-up approach as shown in Fig. 2 (top). Our method first calculates the bounding box of the entire scene and subdivides the bounding box with the user-defined finest grid resolution $2^{\mathcal{N}}$. Then, for each shading point, the node that contains the shading point is detected and marked as occupied, as shown in Fig. 2(a). The parent node is marked as occupied when one of the child nodes is marked as occupied.This process is repeated in a bottom-up manner.

Next, our method constructs an adaptive grid in a top-down approach as shown in Fig. 2 (bottom). From the root voxel encompassing the entire scene, our method calculates the SH coefficients L_l^m, SH gradients ∇L_l^m, and the Hessian matrices H_l^m for all spherical lights at the grid points of the root voxel as shown in Fig. 2(e). Then the interpolation error E_{abs} (E_{rel}) is computed for the root voxel. If the error exceeds the use-defined threshold ϵ_{abs} (ϵ_{rel}), the root voxel is subdivided into eight child voxels as shown in Fig. 2. If the error of the voxel exceeds the threshold, but the corresponding node in the occupancy octree is not marked as occupied, the voxel is not further subdivided as shown in Figs. 2(f)(g). By repeating this process, an adaptive grid with shading voxels is constructed.

6 Results

In this section, we first validate our derivation of analytic SH Hessian $\mathcal{H}Y_l^m$ in Sec. 6.1, then the results of our application to SH lighting with adaptive grids are shown in Sec. 6.2.

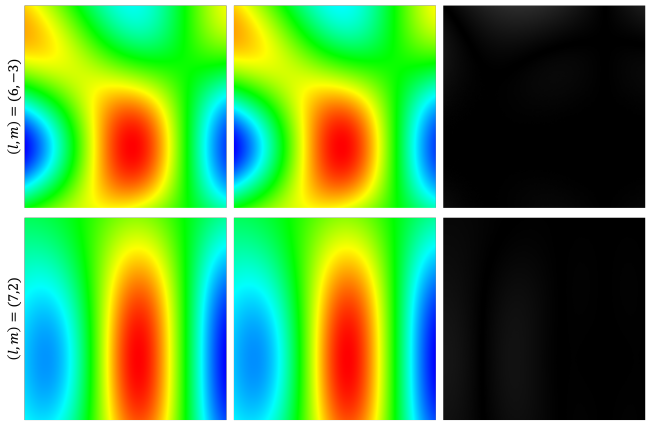


Fig. 3. Visualization of the Hessian matrix H_l^m of SH coefficients. Top row visualizes ∂_{xx} component of H_6^{-3} evaluated by using (a) our method and (b) finite difference, and bottom row visualizes ∂_{zz} component of H_2^7. Elements of Hessian matrices are normalized for better visualization. The rightmost column (c) shows the absolute difference scaled by ten. As shown in the rightmost images, our method can accurately calculate the Hessian matrices.

6.1 Validation and evaluation of SH Hessian $\mathcal{H}Y_l^m$

Fig. 3 shows comparisons of the Hessian matrix H_l^m of SH coefficient L_l^m evaluated using our analytic formula in Eq. (15) and finite difference of SH coefficients L_l^m in Eq. (2). To render Fig. 3, a spherical light with radius $r = 0.4$ is located at $(0.2, 1.0, 0.3)^\top$, and the Hessian matrix H_l^m are evaluated in a square whose top-left and bottom-right corners are $(0.5, 0, 0.5)^\top$ and $(-0.5, 0, -0.5)^\top$, respectively. As shown in Fig. 3, our formula can provide accurate results.

Fig. 4(a) shows performance comparisons between our method, SCD, and the *sphericart* library [Bigi et al. 2023]. Please note that the previous SCD method [Mézières et al. 2022] does not propose the calculation method for the Hessian matrix. We derive the recurrence formulae for the second derivative of the associated Legendre Polynomial P_l^m with respect to the spherical coordinates (θ, ϕ), then the second derivatives with respect to the Cartesian coordinate (x, y, z) are converted from them. The details of the derivation are described in the supplemental material. Note that Bigi et al. [2023] do not provide the second order derivatives of SSH and SH explicitly in the paper, while the sphericart library implements the computation for the Hessian matrix, and their method can generate hard-coded implementations up to sixth-order SHs, while our method can generate codes to evaluate SH, the gradient, and the Hessian with much higher orders.

We measured the computational times to evaluate our SH Hessian $\mathcal{H}Y_l^m$ using SSH $\mathcal{H}\tilde{Y}_l^m$, spherical coordinate derivative (SCD) method, and the *sphericart* library. The computational times are measured on Apple M2 Ultra CPU (24 Cores). For each method, the same set of random 10^4 directions are used and the average computational times over 100 executions (single thread) are measured in Fig. 4. As shown in Fig. 4, our method is the fastest and achieves 3× speed-ups against the state-of-the-art method [Bigi et al. 2023] for

Table 1. Mean Square Errors (MSE) ($\times 10^{-3}$) of the Hessian matrices evaluated by using our method and the SCD method with different SH orders. Our method provides no errors up to 16-th SH orders, while SCD provides errors even for low-order SHs.

Algorithm	4	8	12	16	20
Ours	0.000	0.000	0.000	0.000	0.505
SCD	0.067	0.281	0.637	1.135	2.279

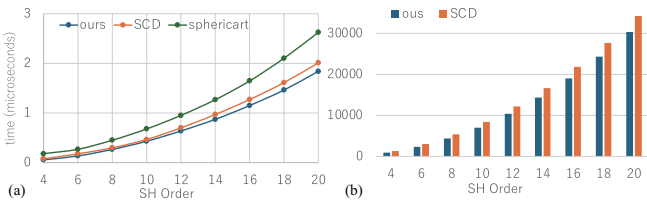


Fig. 4. Plots of (a) SH Hessian $\mathcal{H}Y_l^m$ computation times for our method, SCD, and the sphericart library [Bigi et al. 2023]. Our method achieves 3×/30% for fourth order, and 30%/10% for 20-th order speed-ups against sphericart/SCD, respectively. The numbers of (b) total operations (addition/subtraction/multiplication) for ours and SCD.

fourth order SH (0.05549μs for ours and 0.18093μs for the sphericart) and 30% speed-ups against the SCD method (0.07625μs). For 20-th order SH, our method consistently achieves 30% and 10% speed-ups against the sphericart library and the previous method, respectively. Fig. 4(b) compares the numbers of arithmetic operations that are counted in the generated codes with different SH orders. The number of operations in the SCD method is about 1.46× for fourth-order SH and 1.13× for 20-th order SH compared with that of our method, which corresponds with the computational times in Fig. 4(a).

Table 1 shows the mean square errors (MSEs) of SH Hessian $\mathcal{H}Y_l^m$ using our method and SCD compared with the ground truth evaluated using the sphericart library. As shown in Table 1, our method can accurately evaluate SH Hessian $\mathcal{H}Y_l^m$ while SCD suffers from numerical errors arising from the division by $\sin\theta$, which makes the computation of the associated Legendre polynomial P_l^m and the conversion from spherical coordinates to Cartesian coordinates unstable around the poles ($\theta = 0, \pi$).

6.2 Application to SH lighting with adaptive grids

We have implemented our SH Hessian-based error metric for adaptive grids on top of a PRT framework. Our PRT framework calculates the reflected radiance at each vertex as the shading point. Currently, our PRT framework is implemented on the CPU, and its GPU implementation is left for our future work. We tested our adaptive grid method on a *dragon scene* (937K triangles) and a *living room scene* (1.88M triangles). The SH coefficients L_l^m for spherical lights and those for the transfer function (i.e., the product of cosine-weighted BRDF and the visibility function) are represented by fourth-order SHs, unless otherwise stated. All the images with 1024×768 resolutions are rendered using Apple M1 Max CPU (10 Cores). The reference images were rendered by computing the SH coefficients L_l^m for all the spherical lights at each shading point in the same way as the previous methods [Mézières et al. 2022; Wu et al. 2020]. The SH coefficients L_l^m for spherical lights at each shading point are interpolated by using the cubic Hermite interpolation method. Table 2 summarizes the scene configurations and the performance statistics of our adaptive grid and uniform grids used in the previous method [Mézières et al. 2022].

Table 2. Computational Statistics. *Grid* denotes the grid type, ours indicates adaptive grid and *uni* indicates uniform grid. $\mathcal{N}$ denotes the finest grid resolution $2^{\mathcal{N}}$. N_p indicates the number of grid points of the shading voxels in thousands. For uniform grids, the number of grid points of all voxels/shading voxels are listed. T_g and T_r represent the computational times (ms) for grid construction and rendering. Equal construction times between ours and uniform grids are highlighted in bold. The best and second PSNR values are highlighted in bold and underlined, respectively.

Scene	Grid	$\mathcal{N}$	N_p	T_g	T_r	PSNR(↑)
dragon	ours	6	2.68	1532	56.8	**59.0**
dragon	uni.	3	0.73/0.40	227/128	48.4	21.0
dragon	uni.	4	4.91/1.53	**1532**/481	48.6	35.2
dragon	uni.	5	35.94/4.79	11208/**1542**	48.6	<u>52.8</u>
living	ours	6	5.76	4.36	87.5	<u>70.2</u>
living	uni.	3	0.73/0.47	0.23/0.15	76.8	26.8
living	uni.	4	4.91/2.17	1.61/0.77	76.9	31.3
living	uni.	5	35.94/8.84	13.05/**4.70**	77.5	52.5
living	uni.	6	274.63/38.39	122.93/30.36	78.2	**70.9**

Fig. 5 shows comparisons between our adaptive grid and the uniform grid with 2516 spherical lights. The computational time for the reference image (Fig. 5(a)) is 6586 ms. Fig. 5(b) shows the rendering result using our adaptive grid. Fig. 5(c) shows the adaptive grid that is constructed by using our SH Hessian-based error metric with the relative error threshold $\epsilon_{rel} = 5.8$. The number of grid points of shading voxels is 2683 in Figs. 5(b)(c), and the construction time of this adaptive grid is 1532 ms. The computational time for Fig. 5(b) is 56.77 ms (17.6 fps). Once the adaptive grid is constructed, our method achieves interactive rendering, and the total computational time, including the grid construction (1589 ms), is 4.14× faster than the traditional method that computes SH coefficients for each shading point.

Figs. 5(e)(f)(g) show the rendering results using uniform grids with different resolutions. Previous grid-based methods [Mézières et al. 2022; Wu et al. 2020] typically use $8^3 (\mathcal{N} = 3)$ grids. While a low-resolution grid works well for spherical lights distant from the grid, it fails for spherical lights on the floor as shown in the visualization of Mean Absolute Percentage Errors (MAPEs) (Fig. 5(d)) and the rendering image (Fig. 5(e)). The interpolation errors on the floor are prominent in Figs. 5(d) top-right corner and (e). This is because when the spherical light is located in the voxel, and closer to the shading point than the grid points, the interpolation error occurs even if accurate Hermite interpolation is used.

Fig. 5(f) and (g) are rendered by using $16^3 (\mathcal{N} = 4)$ and $32^3 (\mathcal{N} = 5)$ grids, respectively. We first compare our result in Fig. 5(b) with Fig. 5(f). The grid construction time T_g for our adaptive grid (the number of grid points $N_p = 2683$) is almost equal to that for 16^3

uniform grid where all the grid points ($N_p = 4913$) are used since the previous methods [Mézières et al. 2022; Wu et al. 2020] compute the SH coefficients at all the grid points of a uniform grid. As shown in Table 2, the computational time T_g to construct 16^3 uniform grid ($T_g = 1532$ ms) is equal to that for our adaptive grid. As shown in Fig. 5(b) and (f), our method provides accurate rendering results as indicated by Mean Square Errors (MSEs) and Peak Signal-to-Noise Ratio (PSNR). Our method reduces MSE to 1/30 and increases PSNR by about 24 dB.

Next, we apply our occupancy octree to the uniform grids and compare our method with 32^3 grids ($T_g = 1542$ ms, $N_p = 4786$). While the interpolation errors in the floor are reduced as shown in Fig. 5(g), MSE and PSNR in Fig. 5(g) are worse than those rendered by using the adaptive grid. As shown in the bottom right corner of Fig. 5(d), MAPEs on the dragon model are almost zero, while those on the floor remain. This indicates that 32^3 resolution is excessive for the dragon model, while that is insufficient for the floor. On the other hand, imperceptibly small MAPEs (< 2%) are distributed entirely in our method as shown in Fig. 5(d) top-left corner, which indicates that our Hessian-based error metrics work well both for distant and near spherical lights.

Fig. 6 shows a comparison of the living room scene between our adaptive grid and uniform grids with two spherical lights on the floor. In this scene, the absolute error E_{abs} is used as the error metric, and the absolute error threshold $\epsilon_{abs} = 0.15$ is used. The computational time for Fig. 6(b) is 87.46 ms (11.4 fps). For equal grid construction time comparison with shading voxels enabled, our adaptive grid ($N_p = 5760$) is compared with 32^3 uniform grid with $N_p = 8844$. The computational times T_g for our method and the 32^3 uniform grid are 4.36 ms and 4.70 ms, respectively. As shown in Fig. 6(b) and (c), our method provides more accurate results and increases PSNR by 18 dB in this scene. $64^3(\mathcal{N} = 6)$ uniform grid with $N_p = 38392$ grid points can improve PSNR to 70.9 dB, which is almost the same as ours, but the computational time $T_g(= 30.36ms)$ is about 7× slower than that of our method.

Fig. 7 shows a comparison between (a) the Hermite interpolation method and (b) the trilinear interpolation method. The Hermite interpolation is effective in our adaptive grids and reduces MAPE two orders of magnitude, as shown in Fig. 7.

Fig. 8 compares the rendering results of the living room scene rendered by using uniform grids (first/second rows) and our adaptive grids (third/fourth rows). As shown in Fig. 8, uniform grids with 8^3 and 16^3 resolutions produce visible artifacts.Even if the grid resolution is increased to 32^3, visible artifacts cannot be eliminated. By increasing the grid resolution to 64^3, artifacts can be removed at the cost of the increase in the grid construction time. The third and the fourth rows show the rendering results using our adaptive grids with different error thresholds ϵ_{abs}, as shown in Fig. 8. Our adaptive grid with $\epsilon_{abs} = 4$ produces better results than the uniform grid with 32^3 resolution in terms of the visual quality, and the grid construction time is reduced by a factor of about four.

Fig. 9 compares the rendering results of the dragon scene rendered by using our adaptive grids with different finest resolution parameter $\mathcal{N}$ and relative error threshold parameter ϵ_{rel}. The first column shows the results with $\epsilon_{rel} = 10$, and the second column shows the results with $\epsilon_{rel} = 1$. As shown in the visualization of MAPE images, our method can generate adaptive grids with the similar errors, indicating that the excessive subdivision is avoided. On the other hand, for smaller relative error threshold $\epsilon_{rel} = 1$, $\mathcal{N} = 5$ is insufficient and the subdivision of the adaptive grid is limited to $\mathcal{N}$, as shown in $(\mathcal{N} = 5, \epsilon_{rel} = 1)$ case. This can be alleviated by increasing $\mathcal{N}$, as shown in $\mathcal{N} = 6$ and 7.

Fig. 10 shows the performance and error analysis of the living room scene with different absolute error thresholds ϵ_{abs}. Fig. 10(a) shows the log-log plots of the number of grid points N_g and the computational time T_g with respect to ϵ_{abs}. As shown in Fig. 10(a), N_p and T_g decrease linearly with respect to ϵ_{abs} in log-scale. In this scene, by halving ϵ_{abs}, MAPE and MAE also halve, as shown in Fig. 10(b) and (c). This indicates that our error metric can estimate and control the interpolation errors appropriately. The absolute error metric tends to subdivide bright regions excessively and dark regions moderately. It is the opposite of the relative error metric. Therefore, the relative error metric is recommended for use in bright scenes, and the absolute error metric is better used for other scenes.

6.3 Discussion

Our method requires the evaluations of the SH Hessian $\mathcal{H}Y_l^m$ for all the spherical lights, which incurs additional computational costs compared with the previous gradient-based method [Mézières et al. 2022]. From Table. 2, the computational time per grid point for our method incurs about 44% to 77% additional costs compared with the previous method that requires the evaluations of SH and SH gradient, while our method reduced the evaluation costs by using SSH as described in Sec. 6.1. Skipping the evaluation of SH Hessians for distant spherical lights would be a possible solution to reduce the computational costs, but it remains as our future work.

Rendering with uniform grids is faster than that with our adaptive grid since, for each shading point, finding the voxel that contains the shading point is in constant time for uniform grids. In contrast, our method requires the traversal from the root voxel to the leaf voxel. From Table. 2, our adaptive grid incurs the additional computational cost about 13% to 17%, but our method achieves interactive frame rates as described before.

7 Conclusions

We have introduced spherical harmonics Hessian and solid spherical harmonics to estimate the interpolation errors of SH coefficients for spherical lights. We proposed a code generator to compute SH Hessian efficiently and accurately. We showed that our Hessian-based error metrics can construct an adaptive grid for SH lighting, and our method can provide much more accurate results than the uniform grid-based method. SH Hessian and SSH open up new possibilities for SH operations and applications in computer graphics.

As for future work, we plan to extend our method to estimate the interpolation errors of SH coefficients for general light sources, such as polygonal lights. We also want to investigate the applications and operations of SSH, such as the efficient calculation of triple product or analytical integral of SSH over the polygonal domain by exploiting the properties of Cartesian coordinate representation.

Acknowledgments

We thank all the reviewers for their constructive comments and helpful suggestions. This project was partially supported by JSPS KAKENHI Grant Numbers 23K21737 and 23K28204, and The Telecommunications Advancement Foundation.

References

Thomas Annen, Jan Kautz, Frédo Durand, and Hans-Peter Seidel. 2004. Spherical harmonic gradients for mid-range illumination. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques (EGSR'04)*. 331–336.

Laurent Belcour, Guofu Xie, Christophe Hery, Mark Meyer, Wojciech Jarosz, and Derek Nowrouzezahrai. 2018. Integrating Clipped Spherical Harmonics Expansions. *ACM Trans. Graph.* 37, 2, Article 19 (March 2018), 12 pages. https://doi.org/10.1145/3015459

Filippo Bigi, Guillaume Fraux, Nicholas J. Browning, and Michele Ceriotti. 2023. Fast evaluation of spherical harmonics with sphericart. *J. Chem. Phys.* 159 (2023), 064802.

Kei Iwasaki, Yoshinori Dobashi, Fujiichi Yoshimoto, and Tomoyuki Nishita. 2007. Precomputed radiance transfer for dynamic scenes taking into account light interreflection. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques (EGSR'07)*. 35–44. https://doi.org/10.2312/EGWR/EGSR07/035-044

Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. 2008. Radiance caching for participating media. *ACM Trans. Graph.* 27, 1, Article 7 (March 2008), 11 pages. https://doi.org/10.1145/1330511.1330518

Jaroslav Krivanek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. 2005. Radiance Caching for Efficient Global Illumination Computation. *IEEE Transactions on Visualization and Computer Graphics* 11, 5 (Sept. 2005), 550–561. https://doi.org/10.1109/TVCG.2005.83

Jaroslav Křivánek, Kadi Bouatouch, Sumanta Pattanaik, and Jiří Žára. 2006. Making radiance and irradiance caching practical: adaptive caching and neighbor clamping. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques* (Nicosia, Cyprus) *(EGSR '06)*. Eurographics Association, Goslar, DEU, 127–138.

Julio Marco, Adrian Jarabo, Wojciech Jarosz, and Diego Gutierrez. 2018. Second-Order Occlusion-Aware Volumetric Radiance Caching. *ACM Trans. Graph.* 37, 2, Article 20 (July 2018), 14 pages. https://doi.org/10.1145/3185225

Pierre Mézières, Nicolas Mellado, Loïc Barthe, and Mathias Paulin. 2022. Recursive Analytic Spherical Harmonics Gradient for Spherical Lights. *Computer Graphics Forum* 41, 2 (2022), 393–406. https://doi.org/10.1111/cgf.14482

Derek Nowrouzezahrai, Patricio Simari, and Eugene Fiume. 2012. Sparse zonal harmonic factorization for efficient SH rotation. *ACM Trans. Graph.* 31, 3, Article 23 (June 2012), 9 pages. https://doi.org/10.1145/2167076.2167081

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. *ACM Trans. Graph.* 25, 3 (July 2006), 977–986. https://doi.org/10.1145/1141911.1141982

Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. 2012. Practical Hessian-based error control for irradiance caching. *ACM Trans. Graph.* 31, 6, Article 193 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366212

Peter Pike Sloan. 2008. Stupid Spherical Harmonics (SH) Tricks. In *Proceedings of the Game Developers Conference*.

Peter-Pike Sloan. 2013. Efficient Spherical Harmonic Evaluation. *Journal of Computer Graphics Techniques (JCGT)* 2, 2 (2013), 84–83. http://jcgt.org/published/0002/02/06/

Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.* 21, 3 (July 2002), 527–536. https://doi.org/10.1145/566654.566612

Peter-Pike Sloan, Ben Luna, and John Snyder. 2005. Local, deformable precomputed radiance transfer. *ACM Trans. Graph.* 24, 3 (July 2005), 1216–1224. https://doi.org/10.1145/1073204.1073335

E.O. Steinborn and K. Ruedenberg. 1973. Rotation and Translation of Regular and Irregular Solid Spherical Harmonics. Advances in Quantum Chemistry, Vol. 7. Academic Press, 1–81. https://doi.org/10.1016/S0065-3276(08)60558-4

Jingwen Wang and Ravi Ramamoorthi. 2018. Analytic spherical harmonic coefficients for polygonal area lights. *ACM Trans. Graph.* 37, 4, Article 54 (July 2018), 11 pages. https://doi.org/10.1145/3197517.3201291

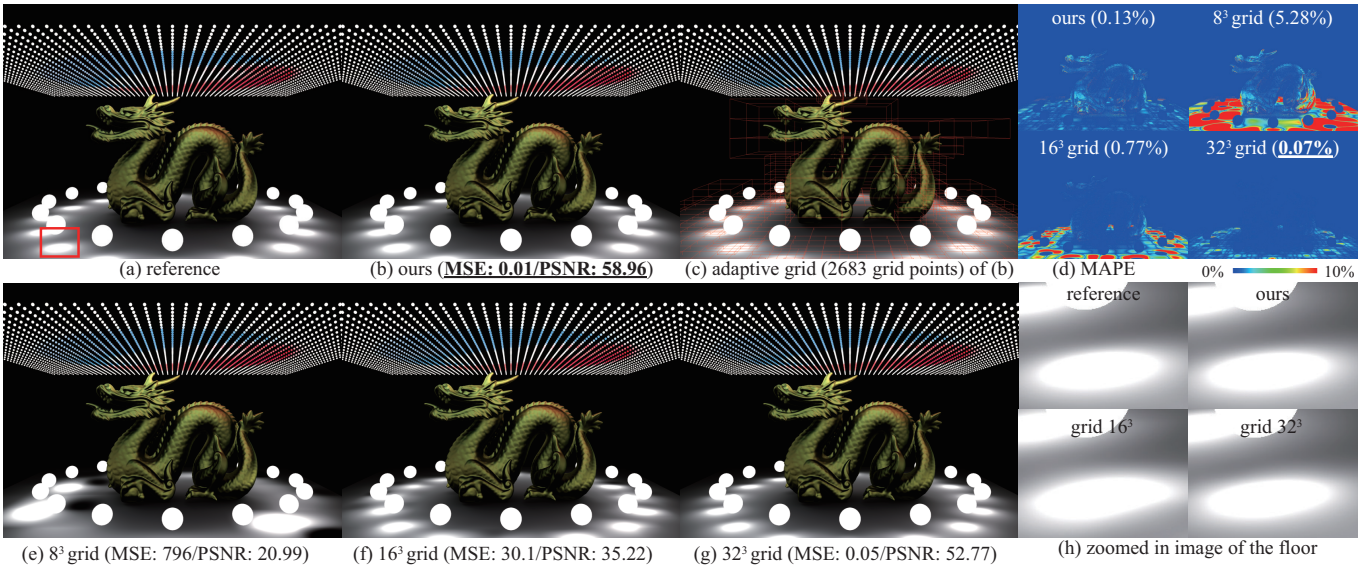
Jiaping Wang, Kun Xu, Kun Zhou, Stephen Lin, Shimin Hu, and Baining Guo. 2006. Spherical harmonics scaling. *Vis. Comput.* 22, 9 (Sept. 2006), 713–720. https://doi.org/10.1007/s00371-006-0057-8

Gregory J. Ward and Paul S. Heckbert. 2008. Irradiance gradients. In *ACM SIGGRAPH 2008 Classes* (Los Angeles, California) *(SIGGRAPH '08)*. Article 72, 17 pages. https://doi.org/10.1145/1401132.1401225

Lifan Wu, Guangyan Cai, Shuang Zhao, and Ravi Ramamoorthi. 2020. Analytic spherical harmonic gradients for real-time rendering with many polygonal area lights. *ACM Trans. Graph.* 39, 4, Article 134 (Aug. 2020), 14 pages. https://doi.org/10.1145/3386569.3392373

Hanggao Xin, Zhiqian Zhou, Di An, Ling-Qi Yan, Kun Xu, Shi-Min Hu, and Shing-Tung Yau. 2021. Fast and accurate spherical harmonics products. *ACM Trans. Graph.* 40, 6, Article 280 (Dec. 2021), 14 pages. https://doi.org/10.1145/3478513.3480563

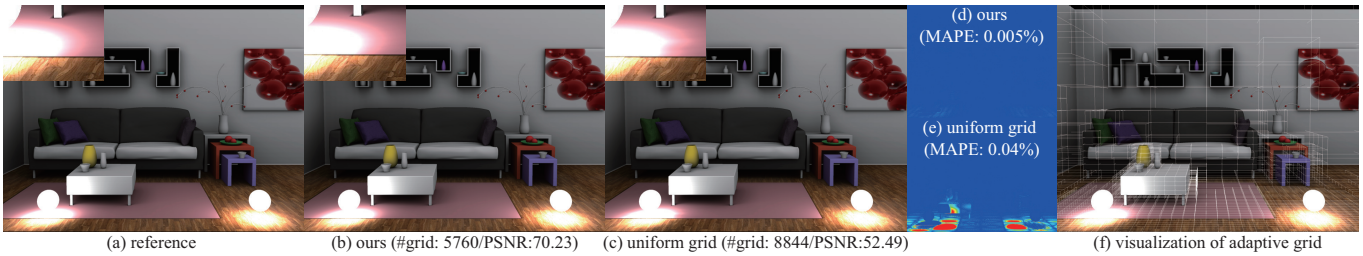
Zihong Zhou and Li-Yi Wei. 2020. Spherical Light Integration over Spherical Caps via Spherical Harmonics. In *SIGGRAPH Asia 2020 Technical Communications*. Article 14, 4 pages. https://doi.org/10.1145/3410700.3425427



(a) reference (b) ours (**MSE: 0.01/PSNR: 58.96**) (c) adaptive grid (2683 grid points) of (b) (d) MAPE

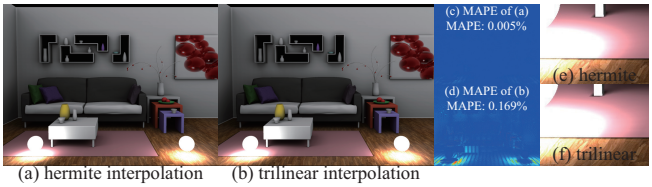
(e) 8^3 grid (MSE: 796/PSNR: 20.99) (f) 16^3 grid (MSE: 30.1/PSNR: 35.22) (g) 32^3 grid (MSE: 0.05/PSNR: 52.77) (h) zoomed in image of the floor

Fig. 5. Comparison of a *dragon scene* between (b) adaptive grid (ours) and (c) uniform grid with 2516 spherical lights. The best performance is highlighted in bold with underlined. Our adaptive grid can increase PSNR by 6 dB and reduce MSE to 1/5 compared with 32^3 uniform grid in equal-time grid construction.



(a) reference (b) ours (#grid: 5760/PSNR:70.23) (c) uniform grid (#grid: 8844/PSNR:52.49) (f) visualization of adaptive grid

Fig. 6. Comparison of a *living room scene* between (b) adaptive grid (ours) and (c) 32^3 uniform grid. The grid construction times T_g for our method and the 32^3 uniform grid are 4.36 ms and 4.70 ms, respectively. Our adaptive grid can increase PSNR by 18 dB and reduce MAPE an order of maginutude in this scene.



(a) hermite interpolation (b) trilinear interpolation

Fig. 7. Comparison between (a) Hermite interpolation and (b) trilinear interpolation for our adaptive grids. The Hermite interpolation (c)(e) can provide more accurate results than the trilinear interpolation (d)(f).

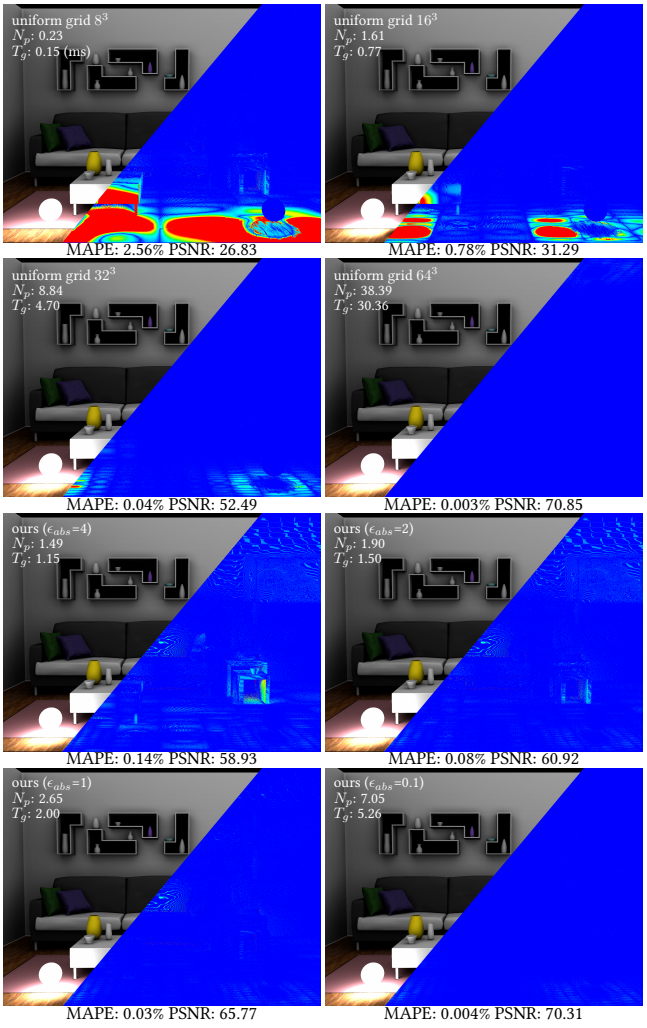


Fig. 8. Rendering results of a living room scene using uniform grids and our adaptive grids with different parameter settings. The first and the second rows shows the rendering results using uniform grids with different resolutions. The third and the fourth rows show the rendering results using our adaptive grids with different absolute error thresholds ϵ_{abs}. N_p and T_g are the number of grid points and the computational time for construction, where shading voxels are used for both uniform grids and adaptive grids. Uniform grids with low resolutions ($\mathcal{N} = 3$ and 4) suffer from visible artifacts due to the interpolation errors. In this scene, by halving the absolute error threshold ϵ_{abs}, MAPE also halves, indicating that our error metric can estimate and control the interpolation errors appropriately.

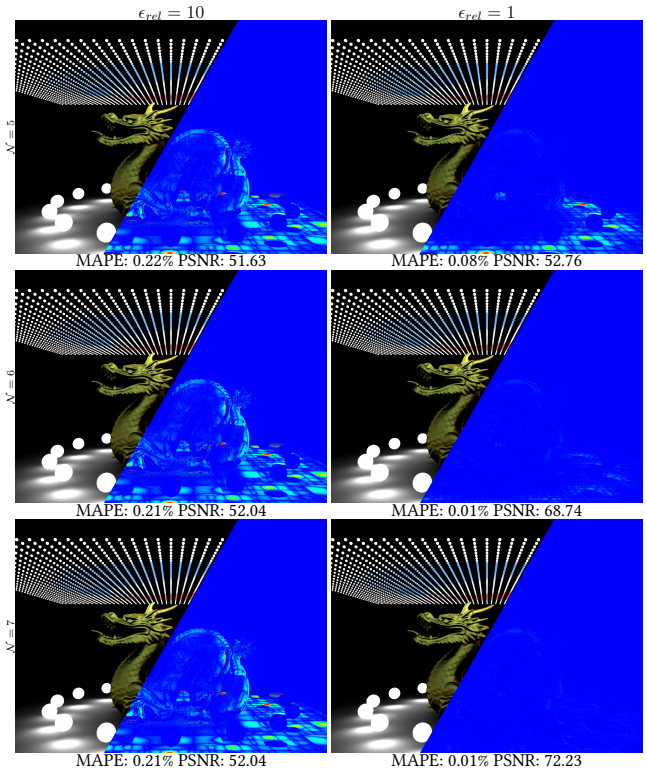


Fig. 9. Rendering results of a dragon scene using our adaptive grid with different parameter settings of the finest grid resolution $\mathcal{N}$ and the relative error threshold ϵ_{rel} (from top to bottom rows, $\mathcal{N} = 5, 6, 7$, and from left to right columns, $\epsilon_{rel} = 10$ and $\epsilon_{rel} = 1$ are used).

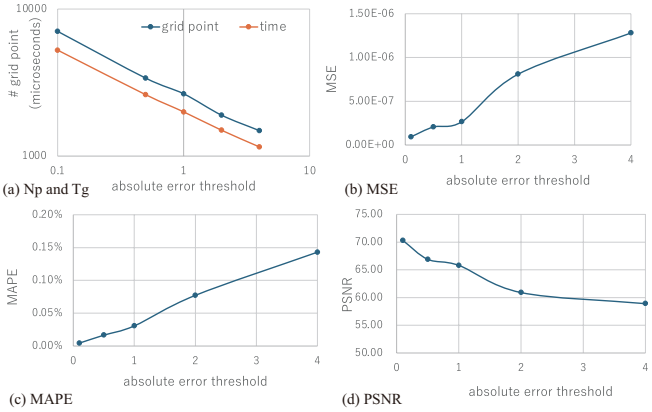


Fig. 10. Performance and error analysis of the living room scene with different absolute error thresholds ϵ_{abs}. (a) log-log plots of the number of grid points N_g and the computational time T_g with respect to ϵ_{abs}. The relationships between ϵ_{abs} and (b) MSE, (c) MAPE, and (d) PSNR are shown.