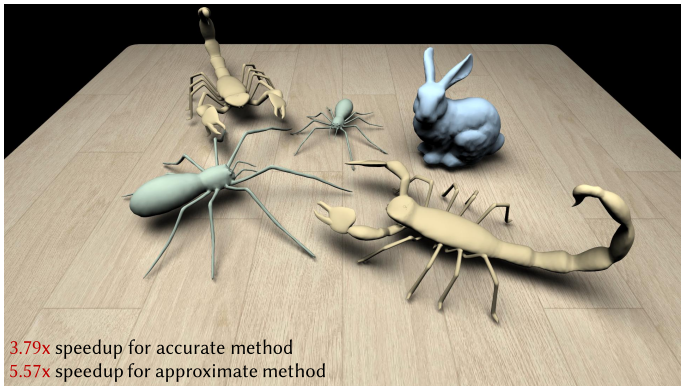
Generalized Spherical Harmonics Products using Spherical Grids

DI AN, BNRist, MOE Key Laboratory of Pervasive Computing, Department of CS&T, Tsinghua University, China
JIAQI WU, BNRist, MOE Key Laboratory of Pervasive Computing, Department of CS&T, Tsinghua University, China
BOWEN XU, Zhili College, Tsinghua University, China
LINGQI YAN, Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates
KUN XU*, BNRist, MOE Key Laboratory of Pervasive Computing, Department of CS&T, Tsinghua University, China

(a) generalized SH triple product integral



(b) specialized SH multiple product integral

Fig. 1. We propose a generalized Spherical Harmonics (SH) product method that supports computing SH products with arbitrary input and output orders. (a) demonstrates glossy relighting using precomputed radiance transfer. This requires evaluating *generalized SH triple product integrals* involving lighting, BRDFs, and visibility—each represented with different SH orders—a case not explicitly addressed by prior methods. (b) shows an example of shadow fields, which involve *specialized SH multiple product integrals* where all input SH orders are the same (order 12). Even in such specialized scenarios, both our accurate and approximate solutions significantly outperform the state-of-the-art method [Xin et al. 2021] in computational speed.

Spherical Harmonics (SH) are fundamental mathematical tools for compactly representing low-frequency spherical functions in computer graphics. Evaluating products of SH-represented functions is a core operation in many rendering algorithms and often becomes a major computational bottleneck. Existing SH product methods face two key challenges: insufficient computational efficiency, and the inflexible requirement that all input functions share the same SH order. To overcome these limitations, we introduce the *Generalized Spherical Harmonics Product*, a novel formulation that supports inputs with different SH orders and allows flexible truncation of the output.

We further propose an efficient and practical computation approach based on Spherical Grids (SphGrid). We establish explicit bidirectional conversions between SphGrid representations and SH coefficients, thus enabling SphGrid to serve as an intermediate representation. Under this framework, SH product computation is reduced to simple point-wise multiplications on the grid. We then derive sufficient conditions on the grid resolution to guarantee exact computation. Using lower resolutions yields an approximate variant that provides a favorable trade-off between accuracy and efficiency.

Our approach is not only more general but also demonstrates superior performance even for the specialized case of identical input orders. In particular, our accurate method achieves a 3.5–6.0× speedup over the state-of-the-art method in a practical setting for graphics, while our approximate variant yields lower errors and a 2.5–6.5× speedup over the state-of-the-art approximate method. We rigorously analyze the correctness and efficiency of our method and demonstrate its effectiveness in real-time rendering applications.

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

ACM Reference Format:
Di An, Jiaqi Wu, Bowen Xu, Lingqi Yan, and Kun Xu. 2026. Generalized Spherical Harmonics Products using Spherical Grids. 1, 1 (April 2026), 18 pages. https://doi.org/10.1145/3811322

1 Introduction

Spherical Harmonics (SH) are widely used basis functions in computer graphics for representing spherical functions. Their ability to efficiently capture low-frequency components makes them well suited for encoding directional signals in rendering, such as visibility [Zhou et al. 2005], environment lighting [Ramamoorthi and

*Corresponding author.

Authors' Contact Information: Di An, and21@mails.tsinghua.edu.cn, BNRist, MOE Key Laboratory of Pervasive Computing, Department of CS&T, Tsinghua University, Beijing, China; Jiaqi Wu, wujiaqi22@mails.tsinghua.edu.cn, BNRist, MOE Key Laboratory of Pervasive Computing, Department of CS&T, Tsinghua University, Beijing, China; Bowen Xu, xu-bw22@mails.tsinghua.edu.cn, Zhili College, Tsinghua University, Beijing, China; Lingqi Yan, lingqi.yan@mbzuai.ac.ae, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates; Kun Xu, xukun@tsinghua.edu.cn, BNRist, MOE Key Laboratory of Pervasive Computing, Department of CS&T, Tsinghua University, Beijing, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2026 ACM.
ACM XXXX-XXXX/2026/4-ART
https://doi.org/10.1145/3811322

Hanrahan 2001], view directions [Müller et al. 2022] and directionally varying radiance [Kerbl et al. 2023].

SH products are fundamental operations in spherical harmonics, particularly in graphics applications. In rendering, it is common to compute the integral of the product of multiple spherical functions, such as lighting, BRDFs, and visibility, all represented using SH coefficients, which often makes this operation a major or even the primary computational bottleneck.

However, current research on SH products faces two key challenges. First, the computational efficiency of SH products remains insufficient. For instance, traditional methods for triple products require a runtime complexity of $O(n^5)$ [Ng et al. 2004]. Although recent state-of-the-art approaches [Xin et al. 2021] reduce the cost to $O(n^3)$ for triple products, the reliance on Fourier series introduces costly FFT computations, especially for low-order SH, limiting their practical utility. Second, existing methods require all input functions to share the same SH order, which restricts flexibility. In practical rendering applications, different functions, such as lighting, BRDFs, and visibility, often exhibit varying frequency content. As a result, one must either pad all inputs to the highest SH order, leading to inefficient computation, or uniformly limit them to a certain lower order, inevitably introducing approximation errors.

To address these limitations, we introduce the concept of *Generalized Spherical Harmonics Product*. This formulation allows for the SH product of k band-limited functions, where each input function may be represented using a different SH order n_i. The output can be truncated to an arbitrary target order p ($1 \le p \le n_G$), where $n_G = \sum_{i=1}^{k} n_i - k + 1$ is the full order of the product function, as will be demonstrated in Section 4. This formulation offers significant flexibility. By setting $p = n_G$, one obtains the full SH representation of the product. By setting $p = 1$, it directly computes the product integral. Thus, our generalized product unifies the computation of SH products and their integrals within a single, flexible framework, overcoming the rigidity of traditional methods.

We further propose a unified, efficient, and practical method for computing the generalized SH product. Our method is based on *Spherical Grids* (SphGrid), a concept from other scientific disciplines [Dai and Xu 2013; Wieczorek and Meschede 2018] that we introduce to computer graphics. To address the generalized SH product problem, we extend the formulation of SphGrid to support arbitrary grid resolutions and use it as an intermediate representation. A SphGrid of resolution $N_\theta \times N_\phi$ consists of points on the sphere defined by Gauss-Legendre nodes in the polar angle θ and uniform samples in the azimuthal angle ϕ. This representation offers two key advantages. First, it supports efficient and exact conversion to and from SH coefficients for band-limited functions when the grid resolution is chosen appropriately. Second, computing the product of functions on a SphGrid reduces to simple point-wise multiplication of their sampled values at each grid point. Consequently, our pipeline for the generalized SH product is straightforward: 1) convert all input SH coefficients to their SphGrid representations on a common grid, 2) perform point-wise multiplication of the grid values, and 3) convert the resulting product grid back to the desired SH coefficients of order p.

By controlling the grid resolution, our method can produce either exact or approximate results, offering a tunable trade-off between accuracy and performance. Specifically, we prove that exact results are guaranteed when the grid resolution meets the condition $N_\theta \ge \lceil (n_G + p - 1)/2 \rceil$ and $N_\phi \ge n_G + p - 1$. Using lower grid resolutions introduces approximation but significantly improves computational efficiency.

Notably, our method is not only more general, as it supports inputs with different SH orders, a scenario not explicitly addressed by previous methods, but also highly competitive in the specialized case where all input orders are identical. Even in this context, our method outperforms state-of-the-art methods [Xin et al. 2021], achieving a speedup of 3.5–6.0× for accurate computation and 2.5–6.5× for approximate computation in real-time rendering applications, while maintaining lower approximation error. Our method achieves this by significantly reducing the constant factor while maintaining competitive asymptotic complexity, which is crucial for performance-critical real-time rendering applications.

In summary, our main contributions are:

(1) **Generalized SH Product Formulation.** We introduce a novel formulation of SH products that supports input functions with arbitrary SH orders and enables flexible truncation of the output.
(2) **SphGrid Introduction and Extension.** We introduce Spherical Grids from other scientific disciplines into computer graphics. To address the generalized SH product problem, we extend the conversion between SH coefficients and SphGrid to support arbitrary grid resolutions, including deriving explicit bidirectional conversion formulas and proving the resolution conditions for exact computation.
(3) **Unified Computation Framework.** We present a practical computation framework for generalized SH product based on Spherical Grids. It unifies the computation of SH products and their integrals.
(4) **Efficiency and Flexibility.** Our method offers a controllable trade-off between accuracy and performance through grid resolution selection, and achieves significantly higher efficiency than existing state-of-the-art specialized methods.
(5) **Broader Applicability.** Our framework naturally supports inputs with different SH orders, a common requirement in rendering applications that is not explicitly addressed by previous approaches.

2 Related Work

2.1 Precomputed Radiance Transfer

Sloan et al. [2002] introduced Precomputed Radiance Transfer (PRT) in real-time rendering applications, which supports dynamic lighting of diffuse and glossy materials by projecting environment lighting and light transport into the SH basis [Ramamoorthi and Hanrahan 2001].

Since then, substantial progress has been made both in exploring the properties of SH and expanding the applicability of PRT. Studies on SH such as SH rotation [Nowrouzezahrai et al. 2012; Sloan et al. 2005b], SH scaling [Wang et al. 2006], SHEXP [Ren et al. 2006], SH gradients [Wu et al. 2020], SH products [Ng et al. 2004;

Xin et al. 2021] and Spin-Weighted SH [Yi et al. 2024] have made SH more practical for the graphics community as a mathematical tool. Meanwhile, PRT has been extended to support dynamic shadows [Zhou et al. 2005], dynamic geometry [Sloan et al. 2005b; Wang et al. 2006], bi-scale rendering effects [Sloan et al. 2003], radiance cache [Krivánek et al. 2005; Xing et al. 2024], glossy materials [Liu et al. 2004], translucent materials [Hao and Varshney 2004; Xu et al. 2007], woven fabric rendering [Yang et al. 2026] and polygonal lights [Wang and Ramamoorthi 2018]. For a comprehensive overview of PRT, we refer readers to the surveys [Lehtinen 2007; Ramamoorthi et al. 2009; Sloan et al. 2005a]. These studies highlight the significance of SH in real-time rendering and motivate our exploration of more practical SH product computations.

2.2 Spherical Harmonics Triple and Multiple Products

The problem of SH triple and multiple products was first studied by Ng et al. [2004]. They proposed an accurate algorithm for evaluating SH triple products by precomputing Clebsch–Gordan coefficients. Their method enables accurate computation of SH triple products with a time complexity of $O(n^5)$, under the assumption that all input SH coefficients share the same order n. However, extending this approach to multiple products is impractical: evaluating the integral of the product of k SH-represented functions leads to a complexity of $O(n^{2k})$ [Sun and Mukherjee 2006].

The Reproducing Kernel Hilbert Space (RKHS) framework proposed by Lessig et al. [2014] also supports accurate SH triple and multiple products. Although these cases are not explicitly discussed in their work, the corresponding time complexities can be derived as $O(n^4)$ for triple products and $O(k^3n^4)$ for multiple products.

More recently, Xin et al. [2021] proposed a state-of-the-art method in terms of time complexity. Their approach transforms SH coefficients into spherical Fourier series, performs products using Fast Fourier Transform (FFT), and converts the result back to SH coefficients. This pipeline reduces the time complexity to $O(n^3)$ for triple products and $O(kn^3 + k^2n^2\log(kn))$ for multiple products. While both SHFFT and our approach accelerate SH products by transforming to an intermediate domain, SHFFT converts to spherical Fourier series and relies on FFT for convolution, introducing non-negligible overhead that leads to suboptimal performance at low SH orders commonly used in real-time rendering. In contrast, our method converts directly to spatial grid values, where products reduce to simple point-wise multiplication, avoiding the overhead of FFT-based processing.

In addition to accurate methods for computing SH triple and multiple products, several approximation techniques have been proposed. A widely used approach in prior work [Sun and Mukherjee 2006; Zhou et al. 2005] recursively multiplies spherical functions expressed in the SH basis and truncates the result back to order n at each step. However, repeated truncation in the frequency domain leads to accumulated approximation errors as the number of products increases. Ren et al. [2006] proposed the SHEXP method, which reformulates multiplication as addition via logarithm and exponential operations in the SH domain. However, this method relies on assumptions of specific kinds of scene geometry and lacks generality. For arbitrary spherical functions, the single logarithmic operation incurs a computational complexity of $O(n^6)$, making it inefficient for practical use.

Notably, existing methods assume that all input SH coefficients share the same order, and the output is restricted to either the same order or the product integral. In contrast, we generalize the formulation of SH products to support arbitrary input and output orders, and propose a more efficient and flexible computation framework with comparable time complexity, improved practical performance, and reduced approximation error.

2.3 Spherical Grids and Spherical Harmonics Transform

Our generalized SH product framework introduces *Spherical Grids* (SphGrid) from other scientific disciplines [Dai and Xu 2013; Wieczorek and Meschede 2018] into computer graphics. However, these works only consider SH transforms between order-n coefficients and a fixed SphGrid resolution of $n \times (2n-1)$, which is insufficient for addressing the generalized SH product problem. We extend the formulation to support arbitrary resolutions and apply it as an intermediate representation for generalized SH products. Specifically, we derive bidirectional conversion formulas at arbitrary resolutions, prove the resolution conditions for exact conversion, and establish a complete computation framework for generalized SH products.

There also exist studies on SH transforms from other forms of grids. Ramamoorthi and Hanrahan [2002] proposed a hybrid representation called Spherical Harmonic Reflection Maps (SHRM), which combines SH coefficients with a grid-based point representation. It provides a practical approach to select SH order based on error tolerance and uses SH transforms for prefiltering. However, it lacks further theoretical analysis of the grid structure and thus cannot guarantee lossless conversion.

Beyond computer graphics, Driscoll and Healy [1994] proved that an order-n band-limited SH function can be accurately reconstructed from a grid of $4n^2$ samples, formed by $2n$ uniform samples in both the polar (θ) and azimuthal (ϕ) directions. They also proposed an $O(n^2\log^2 n)$ forward transform and an $O(n^3)$ inverse transform based on recurrence relations of associated Legendre functions. Healy et al. [2003] later improved both transforms to $O(n^2\log^2 n)$ while enhancing numerical stability. Mohlenkamp [1999] proposed two SH transform algorithms using triangular decompositions of the associated Legendre matrix, achieving $O(n^{5/2}\log n)$ and $O(n^2\log^2 n)$ complexity. Suda and Takami [2002] proposed an approximate transform with $O(n^2\log n)$ complexity using the Fast Multipole Method (FMM). Rokhlin and Tygert [2006] proposed an adaptive approximate transform with $O(n^2\log n/\epsilon)$ complexity, where ϵ is the desired precision. These techniques have also been widely applied in geophysics [Gruber et al. 2011; Sakuraba and Roberts 2009; Schaeffer 2013], where high-order SH transforms are essential for numerical simulations of phenomena such as solar dynamics and the Earth's liquid core.

In principle, these alternative grids could be integrated into our generalized SH product framework. However, they use more spherical points than SphGrid, and these SH transform methods are primarily designed for applications involving very high SH orders, often in the thousands or more. Consequently, they involve large

constant costs, making them impractical for the low-order cases (e.g., $n \leq 20$) commonly used in computer graphics.

3 Preliminaries

To facilitate understanding and make our paper self-contained, we briefly review the fundamentals of spherical harmonics (SH), band-limited spherical functions, and SH products in this section.

3.1 SH and Band-limited Functions

3.1.1 Spherical Harmonics (SH). Spherical harmonics form an orthogonal basis defined on the unit sphere. The real-valued order-n SH basis functions $y_l^m(\theta,\phi)$ ($-l \leq m \leq l$, $0 \leq l < n$) are defined as:

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

where $\theta \in [0,\pi]$ is the polar angle and $\phi \in [0, 2\pi)$ is the azimuthal angle. N_l^m are the normalization factors:

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

and $P_l^m(\cdot)$ are the associated Legendre polynomials, defined as:

$$P_l^m(x) = (-1)^m (1-x^2)^{\frac{m}{2}} \sum_{k=0}^{\left\lfloor \frac{l-m}{2} \right\rfloor} \frac{(-1)^k (2l-2k)!}{2^l k!(l-k)!(l-2k-m)!} x^{l-2k-m}. \tag{3}$$

3.1.2 Projection and Reconstruction. Given a spherical function F, it can be approximately reconstructed from the order-n SH coefficients $\mathbf{f} = \{f_l^m \mid -l \leq m \leq l,\ 0 \leq l < n\}$ through a linear combination of the SH basis weighted by the coefficients:

$$F(\theta,\phi) \approx \widetilde{F}(\theta,\phi) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} f_l^m\, y_l^m(\theta,\phi). \tag{4}$$

The order-n SH coefficients $\mathbf{f}$ consist of n^2 values, which provide the SH representation of the given function F. These coefficients can be obtained by projecting the function onto the SH functions:

$$f_l^m = \int_{\mathbb{S}^2} F(\theta,\phi)\, y_l^m(\theta,\phi)\, \sin\theta \,\mathrm{d}\theta\mathrm{d}\phi. \tag{5}$$

3.1.3 Band-limited Functions. We refer to a spherical function F as an order-n *band-limited function* if it can be exactly reconstructed from its order-n SH coefficients $\mathbf{f}$:

$$F(\theta,\phi) = \widetilde{F}(\theta,\phi) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} f_l^m\, y_l^m(\theta,\phi). \tag{6}$$

3.2 SH Products and Product Integrals

3.2.1 Integral. The integral of a spherical function F over the unit sphere can be obtained directly from the zeroth-order SH coefficient f_0^0 according to Eq. 5:

$$\int_{\mathbb{S}^2} F(\theta,\phi) \sin\theta \,\mathrm{d}\theta\mathrm{d}\phi = 2\sqrt{\pi} f_0^0. \tag{7}$$

3.2.2 Double Product Integral. Given two order-n band-limited functions F_1 and F_2 represented by SH coefficients $\mathbf{f}_1$ and $\mathbf{f}_2$, respectively, the integral of their product function F_1F_2 over the unit sphere can be simply computed by the dot product of their SH coefficients due to the orthogonality of SH basis functions:

$$\int_{\mathbb{S}^2} F_1\, F_2 \,\mathrm{d}\Omega = \langle \mathbf{f}_1, \mathbf{f}_2 \rangle. \tag{8}$$

3.2.3 Triple Product. Given three order-n band-limited functions F_1, F_2 and F_3 represented by order-n SH coefficients $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$, respectively. The *triple product integral* is defined as the integral over the unit sphere of their multiplication:

$$\int_{\mathbb{S}^2} F_1\, F_2\, F_3 \,\mathrm{d}\Omega = \int_{\mathbb{S}^2} \underbrace{(F_1\, F_2)}_{G} \cdot F_3 \,\mathrm{d}\Omega = \langle \mathbf{g}, \mathbf{f}_3 \rangle. \tag{9}$$

As shown in the above equation, computing the triple product integral can be divided into two steps: computing the product of two functions $G = F_1\, F_2$, followed by a dot product of the product function G and the third function F_3 in SH space. The SH coefficients $\mathbf{g}$ of function G are computed from $\mathbf{f}_1$ and $\mathbf{f}_2$ through a *triple product* operation:

$$\mathbf{g} = \mathbf{f}_1 \otimes \mathbf{f}_2. \tag{10}$$

3.2.4 Multiple Product. Considering k order-n band-limited functions $F_1, F_2, ..., F_k$ ($k > 3$) represented by order-n SH coefficients $\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k$, respectively. The *multiple product integral* is defined as:

$$\int_{\mathbb{S}^2} F_1\, F_2 \,...\, F_k \,\mathrm{d}\Omega = \int_{\mathbb{S}^2} \underbrace{(F_1 \cdots F_{k-1})}_{G} \cdot F_k \,\mathrm{d}\Omega = \langle \mathbf{g}, \mathbf{f}_k \rangle, \tag{11}$$

where the SH coefficients $\mathbf{g}$ of function G are computed through a *k-multiple product* operation:

$$\mathbf{g} = \otimes_k(\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{k-1}). \tag{12}$$

Existing works usually employ a *recursive approximation* to approximately compute the multiple product by recursively computing triple products [Sun and Mukherjee 2006; Xin et al. 2021; Zhou et al. 2005]:

$$\otimes_k(\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{k-1}) \approx (\cdots((\mathbf{f}_1 \otimes \mathbf{f}_2) \otimes \mathbf{f}_3) \otimes \cdots \otimes \mathbf{f}_{k-1}). \tag{13}$$

Although the input functions are all order-n band-limited functions, it is worth noting that their product function G, computed by either triple product or multiple product, is no longer order-n band-limited. Specifically, the result of a triple product has an order of $2n-1$, while the result of a $k-$multiple product has an order of $kn - n - k + 2$ [Xin et al. 2021]. However, existing rendering applications usually truncate the output SH coefficients $\mathbf{g}$ within order n. This is because such truncation will not affect the accuracy of the triple (or multiple) product integral, where the final step requires a dot product between $\mathbf{g}$ with order-n SH coefficients $\mathbf{f}_3$ (or $\mathbf{f}_k$).

4 Generalized Spherical Harmonics Products

Existing definitions of SH product operations (see Section 3.2) restrict each input band-limited function to have the same SH order, and we would like to relax this requirement to a more generalized case.

Given k band-limited spherical functions $F_1, \ldots, F_k (k \geq 2)$, where the SH order of the i-th function is denoted as n_i (note that the SH orders may differ among the functions), our *generalized spherical harmonics product* operation computes the order-p truncated SH coefficients of the product function $G = F_1 F_2 \ldots F_k$:

$$\mathbf{g}^{(p)} = \otimes_k^p(\mathbf{f}_1^{(n_1)}, \mathbf{f}_2^{(n_2)}, \ldots, \mathbf{f}_k^{(n_k)}), \text{ where } 1 \leq p \leq n_G. \quad (14)$$

In the above equation, $\mathbf{f}_i^{(n_i)}$ denotes the order-n_i SH coefficient of the i-th band-limited function F_i $(1 \leq i \leq k)$. The output truncation order p can be chosen freely between 1 and n_G, i.e., the full SH order of the product function G. As proved in Appendix A, n_G is given by:

$$n_G = (\sum_{i=1}^{k} n_i) - k + 1. \quad (15)$$

The generalized SH product operation will be a powerful tool for efficiently and flexibly handling spherical function products in a variety of applications. For instance, in rendering, the integrand for shading often involves a product of multiple terms—such as lighting, visibility, and the BRDF—each of which may have different frequency content. A diffuse BRDF may be well-represented with a low SH order, while a specular BRDF or a high-frequency environment map requires a much higher order. Our method allows these components to be processed in a unified manner, in contrast, traditional methods would assume all terms to share the same SH order.

Furthermore, by flexibly adjusting the truncation order p, our framework seamlessly unifies two fundamental operations: computing the SH multiple product and computing the multiple product integral. Specifically, by setting $p = n_G$, the method yields the exact full-order SH representation of the product G; by setting $p = 1$, it directly computes the integral of the product $\int_{\mathbb{S}^2} G \mathrm{d}\Omega$ (i.e., a constant factor to the computed order-1 SH coefficient of G, see Eq. 7), which is the core of many rendering integrals. Thus, within a single formulation, the same operation can compute both the product's detailed harmonic expansion or its integral.

The traditional approaches (Section 3.2), which we distinguish as the *specialized SH product*, are simply the special cases of our generalized product, where all input orders are equal ($n_1 = \cdots = n_k = n$) and the output order is set to $p = n$ (for product) or $p = 1$ (for product integral). Please be aware that the parameter k may differ by 1 in the definitions of specialized and generalized SH products. In specialized SH products, k refers to the total number of functions in the final product integral, even though the product operation itself is applied to only $k - 1$ SH coefficient vectors, e.g., $k = 3$ corresponds to $F_1 F_2$ (triple product) and $\int F_1 F_2 F_3 d\Omega$ (triple product integral). In contrast, since we have unified the definitions of products and product integrals, k is defined more directly to indicate the number of functions taken as input to the product operation, e.g., $k = 3$ corresponds to $F_1 F_2 F_3$ and $\int F_1 F_2 F_3 d\Omega$.

In the following, we first introduce the concept of spherical grids and describe the bidirectional conversion procedures between SH coefficients and spherical grid representations in Section 5. Building on this representation, we present a unified and efficient framework for computing generalized SH products in Section 6.

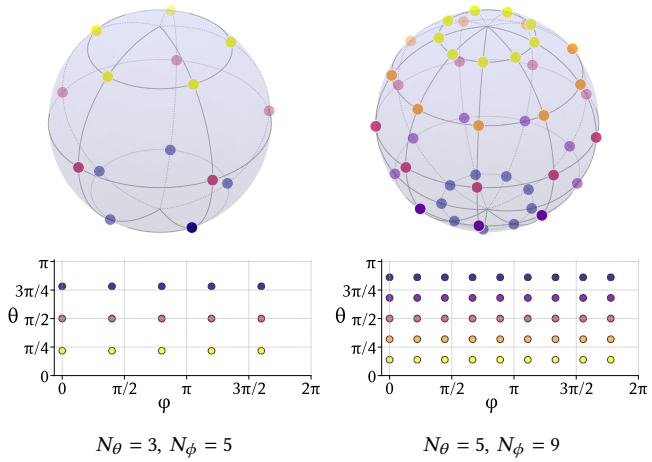


Fig. 2. Visualization of the grid points on two spherical grids. Left: $N_\theta = 3, N_\phi = 5$; right: $N_\theta = 5, N_\phi = 9$.

5 Spherical Grids

For efficient product of band-limited spherical functions, in this section, we introduce the concept of *Spherical Grids* (short for *SphGrid*) from other disciplines [Dai and Xu 2013; Wieczorek and Meschede 2018] into computer graphics.

5.1 Definition and Representation

5.1.1 Definition. A spherical grid of resolution $N_\theta \times N_\phi$ is defined as a grid of points on the unit sphere:

$$\mathcal{P}_{N_\theta \times N_\phi} = \{(\theta_i, \phi_j) \mid 0 \leq i < N_\theta,\ 0 \leq j < N_\phi\}, \quad (16)$$

where (θ_i, ϕ_j) denotes a point on the grid in spherical coordinates.

The polar angles $\{\theta_i\}$ $(0 \leq i < N_\theta)$ satisfy that each $\cos\theta_i$ corresponds to the i-th root of the Legendre polynomial $P_{N_\theta}(x)$:

$$P_{N_\theta}(x) = \sum_{k=0}^{\lfloor N_\theta/2 \rfloor} \frac{(-1)^k (2N_\theta - 2k)!}{2^{N_\theta} k!(N_\theta - k)!(N_\theta - 2k)!} x^{N_\theta - 2k}. \quad (17)$$

The roots are also known as Gauss–Legendre nodes [Press 2007].

The azimuthal angles $\{\phi_j\}$ $(0 \leq j < N_\phi)$ are uniformly sampled over the interval $[0, 2\pi)$, i.e., $\phi_j = 2\pi j / N_\phi$ for $j = 0, 1, \ldots, N_\phi - 1$.

Figure 2 illustrates two examples of spherical grids. For a given resolution, the grid is fixed, with both the number and locations of sampling points uniquely determined.

5.1.2 Representation. Given a spherical function F, its SphGrid representation $\mathcal{F}_{N_\theta \times N_\phi}$ of resolution $N_\theta \times N_\phi$ is defined as the set of function values F evaluated at all points on the SphGrid:

$$\mathcal{F}_{N_\theta \times N_\phi} = \{\, F(\theta_i, \phi_j) \mid (\theta_i, \phi_j) \in \mathcal{P}_{N_\theta \times N_\phi} \,\}. \quad (18)$$

The SphGrid representation offers two key advantages. First, it is straightforward to obtain by simply sampling the spherical function on the grid. This makes it particularly well-suited for pointwise operations like multiplication. For example, to compute the product of two functions, one can directly multiply the function values

at each sample point in a piecewise manner. Second, as will be shown in the following subsections, for band-limited functions, the SphGrid representation can be converted losslessly and efficiently to and from spherical harmonics coefficients, if an appropriate grid resolution $N_\theta \times N_\phi$ is chosen.

5.2 Forward Conversion from SH coefficients to SphGrid representation

Given an order-n band-limited function F represented by its SH coefficients $\mathbf{f}^{(n)} = \{f_l^m\}$ and a target grid resolution $N_\theta \times N_\phi$, the goal of forward conversion is to compute the corresponding SphGrid representation $\mathcal{F}_{N_\theta \times N_\phi} = \{F(\theta_i, \phi_j)\}$ of the target resolution.

5.2.1 Derivations. Certainly, the SphGrid representation can be obtained by reconstructing the band-limited function F from its SH coefficients and evaluating it at the SphGrid sampling points:

$$F(\theta_i, \phi_j) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} f_l^m \, y_l^m(\theta_i, \phi_j). \tag{19}$$

Such a straightforward reconstruction and evaluation scheme requires $O(n^2 N_\theta N_\phi)$ time, i.e., there are totally $N_\theta \times N_\phi$ grid points to evaluate, and reconstructing the function values from SH coefficients at a certain grid point requires $O(n^2)$ operations.

Below, we introduce a more efficient way to obtain the SphGrid representation. Notice that SH basis (see definition in Eq. 1) can be decomposed into a product of two separable components:

$$y_l^m(\theta, \phi) = C_l^m(\theta) \, R_m(\phi), \tag{20}$$

where $C_l^m(\cdot)$ is a function of the polar angle θ and $R_m(\cdot)$ is a function of the azimuth angle ϕ, defined as:

$$C_l^m(\theta) = N_l^{|m|} \, P_l^{|m|}(\cos\theta), \quad R_m(\phi) = \begin{cases} 1, & m = 0 \\ \sqrt{2}\,\cos(m\phi), & m > 0 \\ \sqrt{2}\,\sin(|m|\phi), & m < 0. \end{cases} \tag{21}$$

By replacing the SH basis in Eq. 19 with the decomposed form (Eq. 20) and swapping the summation order, we obtain:

$$F(\theta_i, \phi_j) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} f_l^m \, C_l^m(\theta_i) \, R_m(\phi_j) \tag{22}$$

$$= \sum_{m=-n+1}^{n-1} \sum_{l=|m|}^{n-1} f_l^m C_l^m(\theta_i) R_m(\phi_j). \tag{23}$$

In the above equation, notice that the term $R_m(\phi_j)$ is independent of the band l, so that it could be pulled out of the inner summation:

$$F(\theta_i, \phi_j) = \sum_{m=-n+1}^{n-1} R_m(\phi_j) \sum_{l=|m|}^{n-1} f_l^m C_l^m(\theta_i). \tag{24}$$

5.2.2 Computational Procedure. Hence, to obtain the SphGrid representation, we could first compute and store all values of the inner summation and then compute $F(\theta_i, \phi_j)$ using the stored values of inner summation:

$$F(\theta_i, \phi_j) = \sum_{m=-n+1}^{n-1} R_m(\phi_j) s_m(\theta_i), \text{ where } s_m(\theta_i) = \sum_{l=|m|}^{n-1} f_l^m C_l^m(\theta_i). \tag{25}$$

5.2.3 Time Complexity. There are totally $(2n-1) \times N_\theta$ different inner summations ($s_m(\theta_i)$) and computing each inner summation takes $O(n)$ operations on average. Similarly, we have $N_\theta \times N_\phi$ grid points and evaluating the function value $F(\theta_i, \phi_j)$ at each grid point using stored inner summations takes $O(n)$ operations.

In summary, the forward conversion from order-n SH coefficients to the SphGrid representation of resolution $N_\theta \times N_\phi$ requires $O(n^2 N_\theta + n N_\theta N_\phi)$ time.

5.3 Inverse Conversion from SphGrid Representation to SH Coefficients

Given an order-n band-limited function F represented by its SphGrid representation $\mathcal{F}_{N_\theta \times N_\phi} = \{F(\theta_i, \phi_j)\}$ of a given resolution $N_\theta \times N_\phi$ and a target truncation order p ($1 \le p \le n$), the inverse conversion aims to compute the order-p truncated SH coefficients $\mathbf{f}^{(p)} = \{f_l^m\}$ of the band-limited function F.

5.3.1 Derivations. The SH coefficients can be obtained by projecting the function F onto the SH basis (Eq. 5):

$$f_l^m = \int_{\mathbb{S}^2} F(\theta, \phi) \, y_l^m(\theta, \phi) \, \sin\theta \, \mathrm{d}\theta \mathrm{d}\phi. \tag{26}$$

By using the decomposition of SH basis (Eq. 20) and swapping the order of integration, we have:

$$f_l^m = \int_{\mathbb{S}^2} F(\theta, \phi) \, C_l^m(\theta) \, R_m(\phi) \, \sin\theta \mathrm{d}\theta \mathrm{d}\phi \tag{27}$$

$$= \int_0^\pi C_l^m(\theta) \underbrace{\left(\int_0^{2\pi} F(\theta, \phi) \, R_m(\phi) \, \mathrm{d}\phi \right)}_{s_m(\theta)} \sin\theta \, \mathrm{d}\theta \tag{28}$$

$$= \int_0^\pi C_l^m(\theta) \, s_m(\theta) \, \sin\theta \, \mathrm{d}\theta \tag{29}$$

$$\doteq \sum_{i=0}^{N_\theta - 1} \omega_i \, C_l^m(\theta_i) \, s_m(\theta_i). \quad \text{(equality when } 2N_\theta \ge n + l) \tag{30}$$

In the last step of derivation (Eq. 30), we evaluate the integral through a discrete summation using the Gauss–Legendre quadrature rule over θ. Its equality holds when $2N_\theta \ge n + l$, and a proof of equality is provided in the Appendix C.

The coefficients $\{\omega_i\}$ in Eq. 30 are the Gauss-Legendre weights [Press 2007]:

$$\omega_i = \frac{2}{(1 - x_i^2)[P'_{N_\theta}(x_i)]^2}, \quad 0 \le i < N_\theta, \tag{31}$$

where $\{x_i\}$ are the Gauss-Legendre nodes, i.e., roots of the Legendre polynomial $P_{N_\theta}(x)$ (Eq. 17), and $P'_{N_\theta}(x_i)$ denotes its derivative evaluated at x_i.

The inner integral $s_m(\theta)$ can also be evaluated through a discrete summation using discrete Fourier transform over ϕ:

$$s_m(\theta) = \int_0^{2\pi} F(\theta, \phi) \, R_m(\phi) \, \mathrm{d}\phi \tag{32}$$

$$\doteq \frac{2\pi}{N_\phi} \sum_{j=0}^{N_\phi - 1} F(\theta, \phi_j) \, R_m(\phi_j). \quad \text{(equality when } N_\phi \ge n + |m|) \tag{33}$$

The equality of the last step (Eq. 33) holds when $N_\phi \geq n + |m|$, and the proof of equality is also provided in the Appendix B.

5.3.2 Condition for exact inverse conversion. By combining the equality conditions of both Eq. 30 and Eq. 33 for all coefficients f_l^m ($0 \leq l < p$ and $-l \leq m \leq l$), and taking into account that N_θ and N_ϕ are integers, the recovered order-p truncated SH coefficients $\mathbf{f}^{(p)}$ are *exact* if the grid resolution satisfies the following conditions:

$$N_\theta \geq \left\lceil \frac{n+p-1}{2} \right\rceil, \quad N_\phi \geq n + p - 1. \tag{34}$$

According to the general conditions Eq. 34, we have two important special cases. If we would like to exactly recover the full-order SH representation of function F, the minimal resolution of the grid would be (by setting $p = n$):

$$N_\theta \geq n, \quad N_\phi \geq 2n - 1. \tag{35}$$

If we would like to compute the integral of function F over the unit sphere using Eq. 7, the minimal resolution of grid would be (by setting $p = 1$):

$$N_\theta \geq \left\lceil \frac{n}{2} \right\rceil, \quad N_\phi \geq n. \tag{36}$$

5.3.3 Computational Procedure. By combining Eq. 30 and Eq. 33, To obtain SH coefficients from the SphGrid representation, we could first compute and store the values of inner summations $s_m(\theta_i)$, and then compute SH coefficients using the stored inner summations:

$$f_l^m \doteq \sum_{i=0}^{N_\theta - 1} \omega_i \, C_l^m(\theta_i) \, s_m(\theta_i), \text{ where } s_m(\theta_i) \doteq \frac{2\pi}{N_\phi} \sum_{j=0}^{N_\phi - 1} F(\theta_i, \phi_j) \, R_m(\phi_j). \tag{37}$$

5.3.4 Time Complexity. Similar to the forward conversion, the overall time complexity of the inverse conversion is $O(p^2 N_\theta + p N_\theta N_\phi)$, which is $O(pn^2)$ when N_θ and N_ϕ are set to their minimal values required for exact inverse conversion in Eq. 34.

6 Generalized SH products using SphGrid representations

In this section, we present our unified approach for generalized SH products using SphGrid as an intermediate representation (Section 6.1), followed by the parameter configurations for different computation tasks (Section 6.2).

6.1 A Unified Approach

Given a specific grid resolution $N_\theta \times N_\phi$, our approach for computing the generalized SH product consists of three steps:

(1) **Forward Conversion.** We convert all input SH coefficients $\mathbf{f}_1^{(n_1)}, \ldots, \mathbf{f}_k^{(n_k)}$ into their SphGrid representations $\mathcal{F}_1, \ldots, \mathcal{F}_k$ of the same resolution $N_\theta \times N_\phi$, using the forward conversion (Section 5.2).

(2) **Multiplications using SphGrid representations.** Since SphGrid representations are essentially sampled function values at fixed grid points, the SphGrid representation of the product

$$\mathcal{G}_{N_\theta \times N_\phi} = \{G(\theta_i, \phi_j) \mid (\theta_i, \phi_j) \in \mathcal{P}_{N_\theta \times N_\phi}\} \tag{38}$$

can be obtained through point-wise multiplications on the grid points:

$$G(\theta_i, \phi_j) = F_1(\theta_i, \phi_j) \cdots F_k(\theta_i, \phi_j). \tag{39}$$

(3) **Inverse Conversion.** We obtain the order-p truncated SH coefficients $\mathbf{g}^{(p)}$ of the product function G from its SphGrid representation $\mathcal{G}_{N_\theta \times N_\phi}$ using the inverse conversion procedure in Section 5.3.

6.1.1 Overall Time Complexity. The first step requires $O(N_\theta \sum_{i=1}^k n_i^2 + N_\theta N_\phi \sum_{i=1}^k n_i)$ time, the second step takes $O(k N_\theta N_\phi)$ time, and the third step costs $O(p^2 N_\theta + p N_\theta N_\phi)$ time. The overall time complexity of the unified approach is:

$$O(N_\theta \sum_{i=1}^k n_i^2 + N_\theta N_\phi \sum_{i=1}^k n_i + p^2 N_\theta). \tag{40}$$

6.1.2 Minimal Grid Resolution for Exact Results. To obtain the *exact* order-p truncated SH coefficients of the product, according to the condition for exact inverse conversion (see Section 5.3.2 and Eq. 34), the minimal grid resolution is:

$$N_\theta \geq \left\lceil \frac{n_G + p - 1}{2} \right\rceil, \quad N_\phi \geq n_G + p - 1. \tag{41}$$

When the grid resolution is set to be lower than this threshold, approximation errors are introduced in the high-order components due to insufficient sampling. In practice, considering performance and memory constraints, it may be acceptable to employ a lower-resolution grid, which trades the loss of accuracy in the higher-frequency components for improved computational efficiency and reduced memory consumption.

Our method is unified. For different tasks, such as triple (or multiple) product or triple (multiple) product integrals, we simply adjust the parameters k and p to handle each case (e.g., setting $p = 1$ for product integrals). By configuring different grid resolutions, we can provide accurate results, or different levels of approximation with reduced computational cost.

6.2 Parameter Configurations for Different Tasks

For simplicity and convenience in configurations, we use only a single parameter t to control the grid resolution, setting:

$$N_\theta = \left\lceil \frac{t}{2} \right\rceil, \quad N_\phi = t. \tag{42}$$

Table 1 shows the parameter configurations and time complexity for different tasks. For generalized tasks (a-b), as analyzed in Sec. 6.1.1 and 6.1.2, setting $t = n_G + p - 1$ with a time complexity of $O(n_G^3)$ leads to an exact output while lower values of t introduce approximation, with a time complexity of $O(t \sum_{i=1}^k n_i^2 + t^2 \sum_{i=1}^k n_i + p^2 t)$.

Regarding the specialized tasks (i.e., all input orders are equal: $n_1 = \cdots = n_k = n$), that traditional methods can handle, we also provide their parameter settings and complexities (c-j).

The triple product (tasks (c–e)) requires a grid resolution of $t = 3n - 2$ for both truncated and integral computations, and $t = 4n - 3$ for the full-order result. For all cases, the overall time complexity is $O(n^3)$, and the results are ensured to be exact.

Table 1. Task configurations with different parameters and grid resolutions.

Task	Parameters	Grid Resolution	Time Complexity	Accuracy
(a) Generalized tri./mult. product (integral)	$1 \le p \le n_G$	$t = n_G + p - 1$	$O(n_G^3)$	Accurate
(b) Generalized tri./mult. product (integral)	$1 \le p \le n_G$	$t < n_G + p - 1$	$O(t \sum_{i=1}^{k} n_i^2 + t^2 \sum_{i=1}^{k} n_i + p^2 t)$	Approximate
(c) Specialized truncated triple product	$k = 2, p = n$	$t = 3n - 2$	$O(n^3)$	Accurate
(d) Specialized triple product integral	$k = 3, p = 1$	$t = 3n - 2$	$O(n^3)$	Accurate
(e) Specialized full-order triple product	$k = 2, p = 2n - 1$	$t = 4n - 3$	$O(n^3)$	Accurate
(f) Specialized truncated multiple product	$k \ge 3, p = n$	$t = nk + n - k$	$O(k^3 n^3)$	Accurate
(g) Specialized multiple product integral	$k \ge 4, p = 1$	$t = nk + 1 - k$	$O(k^3 n^3)$	Accurate
(h) Specialized full-order multiple product	$p = nk - k + 1$	$t = 2nk + 1 - 2k$	$O(k^3 n^3)$	Accurate
(i) Specialized truncated multiple product	$k \ge 3, p = n$	$t = 3n - 2$	$O(kn^3)$	Approximate
(j) Specialized multiple product integral	$k \ge 4, p = 1$	$t = 3n - 2$	$O(kn^3)$	Approximate

For exact multiple product computations (tasks (f–h)), the required resolution increases with k: $t = nk + n - k$ for truncated output, $t = nk + 1 - k$ for the integral, and $t = 2nk + 1 - 2k$ for the full-order result. The overall time complexity is $O(k^3 n^3)$.

Finally, for the approximated multiple product (tasks (i-j)), we find that using specialized truncated triple product's resolution $t = 3n - 2$ strikes an excellent balance, maintaining low errors while significantly improving performance, leading to a more favorable time complexity of $O(kn^3)$.

In the subsequent experiments section, we verify that our method supports generalized SH products with different input orders and enables a flexible trade-off between accuracy and performance via the resolution parameter t. For both accurate and approximate specialized product computations, our method is more efficient than state-of-the-art approaches, while also achieving lower errors for approximate computations.

7 Experiments

We performed a series of experiments under various settings to evaluate our method. We implement our method on both CPU and GPU. The CPU implementation is written in C++ 17 and compiled with Clang++, while the GPU version is using CUDA 12.8. All experiments are performed on a PC with an AMD R9-9950X CPU and an NVIDIA RTX 4090 GPU. In our implementation, we observed no numerical issues across all tested SH orders ($n \le 20$).

7.1 Performance Comparison

Our method provides a unified formulation for generalized SH products with potentially different input and output orders. In contrast, previous methods [Ng et al. 2004; Xin et al. 2021] focus on specialized SH products where all input orders are assumed to be identical, and do not explicitly address the case of different input orders.

To highlight the performance advantages of our method, we compare it with previous methods on specialized truncated triple and multiple product tasks. In these settings, all k input SH coefficients share the same order n, and the output is truncated to order n. Below, we briefly describe the baseline methods used in our comparisons.

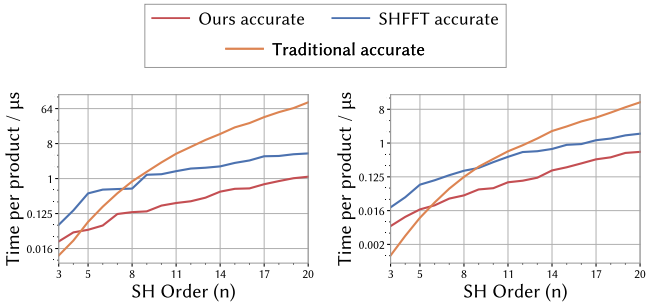
7.1.1 Baselines. We compare our method against several baseline approaches to evaluate performance comprehensively.

- The traditional method based on precomputed Clebsch–Gordan coefficients for accurately computing the specialized truncated triple product and triple product integral [Ng et al. 2004], which we refer to as *traditional accurate*. It has a time complexity of $O(n^5)$.
- The traditional recursive method [Sun and Mukherjee 2006; Zhou et al. 2005] that employs recursive approximations to compute the specialized truncated multiple product and multiple product integral, which we refer to as *traditional recursive*. It has a time complexity of $O(kn^5)$.
- The state-of-the-art FFT-based method [Xin et al. 2021], which we refer to as *SHFFT*. It provides accurate computation of the specialized truncated triple product and triple product integral in $O(n^3)$ time, and accurate computation of the specialized truncated multiple product and multiple product integral in $O(kn^3 + k^2 n^2 \log(kn))$ time; we refer to these variants as *SHFFT accurate*. In addition, SHFFT also provides an approximate solution for the specialized truncated multiple product and multiple product integral based on recursive approximations with a time complexity of $O(kn^3)$, which we refer to as *SHFFT recursive*.

For clarity, We also refer to our accurate method as *ours accurate* and our approximate method as *ours approximate*. The parameters used in our accurate method correspond to the configurations (c)–(h) in Table 1, while those for our approximate method correspond to (i)–(j), where $t = 3n - 2$.

7.1.2 Specialized Truncated Triple Product ($k = 2, p = n$, Table 1(c)). We compare the runtime performance of *ours accurate* with *traditional accurate* [Ng et al. 2004] and *SHFFT accurate* [Xin et al. 2021]. As shown in Fig. 3, we evaluate performance on both CPU and GPU for SH orders in the range $3 \le n \le 20$, which are commonly used in graphics applications.

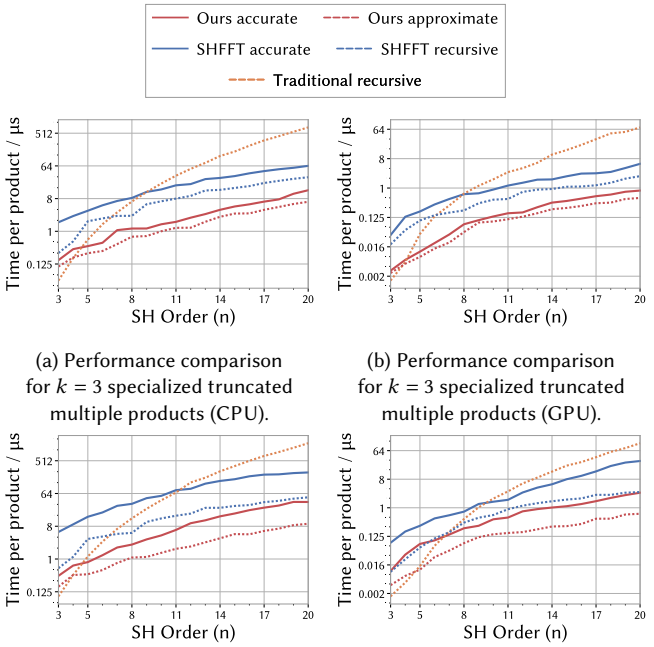
Our method has a lower time complexity than *traditional accurate*, becoming faster for $n \ge 5$ on the CPU and $n \ge 6$ on the GPU. For larger n, our method runs significantly faster. Across all tested SH orders, our method consistently outperforms *SHFFT accurate*,



(a) Performance comparison for specialized truncated triple products (CPU).

(b) Performance comparison for specialized truncated triple products (GPU).

Fig. 3. Comparison of our accurate method (red) with *traditional accurate* (orange) and *SHFFT accurate* (blue) for specialized truncated triple products. Our method outperforms *traditional accurate* for $n \geq 5$ on CPU and $n \geq 6$ on GPU, and is consistently faster than *SHFFT accurate* across all tested SH orders.



(a) Performance comparison for $k = 3$ specialized truncated multiple products (CPU).

(b) Performance comparison for $k = 3$ specialized truncated multiple products (GPU).

(c) Performance comparison for $k = 5$ specialized truncated multiple products (CPU).

(d) Performance comparison for $k = 5$ specialized truncated multiple products (GPU).

Fig. 4. Comparison of accurate methods (ours accurate: red solid; SHFFT accurate: blue solid) and approximate methods (ours approximate: red dashed; traditional recursive: orange dashed; SHFFT recursive: blue dashed). Our accurate method outperforms SHFFT accurate in all tested cases, and is even faster than SHFFT recursive in most cases. Our approximate method becomes faster than traditional recursive when $n \geq 5$, and is consistently faster than SHFFT recursive.

achieving an average speedup of 5.50× on the CPU and 4.03× on the GPU.

Although our method shares the same asymptotic complexity as *SHFFT accurate*, we achieve significant performance improvements at low SH orders ($n \leq 20$) commonly used in graphics applications. This is primarily due to the improved constant factor: *SHFFT accurate* requires FFT-based convolutions which introduce non-negligible overhead, while our method has a simpler formulation with lower constant factors, making it particularly suitable for real-time rendering applications.

7.1.3 Specialized Truncated Multiple Product ($k \geq 3, p = n$, Table 1(f)(i)). In Fig. 4, we compare the runtime performance of both accurate and approximate methods for specialized truncated multiple products. The accurate methods include *ours accurate* (Table 1(f)) and *SHFFT accurate* [Xin et al. 2021], while the approximate methods include *ours approximate* (Table 1(i)), *traditional recursive* [Sun and Mukherjee 2006; Zhou et al. 2005] and *SHFFT recursive* [Xin et al. 2021].

Our accurate method consistently outperforms *SHFFT accurate* across all tested settings, despite having a slightly higher asymptotic complexity. This is again attributed to our improved constant factor: *SHFFT accurate* requires complex divide-and-conquer algorithms and multiple FFT passes, leading to substantial computational overhead, while our method has a simpler formulation with lower constant factors, significantly reducing the actual computational cost at low SH orders.

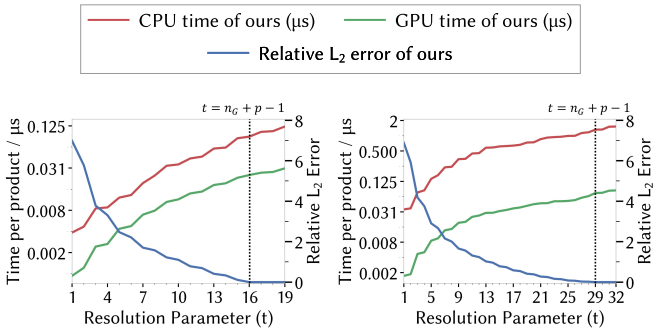
For approximate methods, *ours approximate* exhibits clear advantages over both recursive baselines. Compared to *traditional recursive*, *ours approximate* has a lower time complexity and becomes faster for $n \geq 5$ under both $k = 3$ and $k = 5$, with the speedup increasing as the SH order grows. Compared to *SHFFT recursive*, our approximate method is consistently faster across all tested settings. Notably, *ours accurate* is even faster than *SHFFT recursive* in most cases, despite the latter being designed as an approximate method. This further highlights the efficiency and practical effectiveness of our method.

7.2 Error Analysis of Approximate Methods

In this section, we analyze and visualize the approximation error of our method and compare it with previous approaches. Although *traditional recursive* and *SHFFT recursive* differ in their computational pipelines, they theoretically produce identical approximate results. For clarity, we therefore collectively refer to the results of both methods as *recursive approximation* in the following analysis.

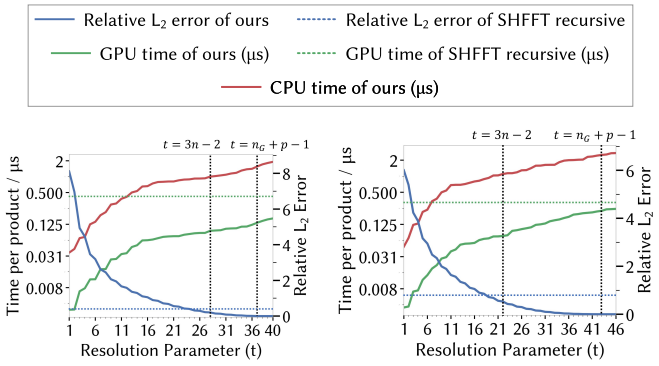
7.2.1 Quantitative Error and Performance Analysis. We first evaluate the relative L_2 error of the output SH coefficients together with the corresponding CPU and GPU runtimes under varying resolution parameters t. As shown in Fig. 5, we consider four representative SH product tasks. For each task, we randomly generate 1 million sets of input SH coefficients, where each coefficient is uniformly sampled from $[-1, 1]$.

The results confirm that exact computation is achieved when $t \geq n_G + p - 1$. For $t < n_G + p - 1$, the approximation error increases as t decreases, while the computational cost is reduced



(a) Generalized triple product: $k = 2, n_1 = 3, n_2 = 10, p = 5$

(b) Generalized multiple product: $k = 3, n_1 = 5, n_2 = 7, n_3 = 10, p = 10$



(c) Specialized truncated multiple product: $k = 3, n = p = 10$

(d) Specialized truncated multiple product: $k = 5, n = p = 8$

Fig. 5. Relative L_2 error and CPU/GPU runtime as functions of the resolution parameter t for various SH product tasks. When $t \geq n_G + p - 1$, marked by the black dotted line, the computation becomes exact and the approximation error is negligible. For the specialized truncated multiple product (c) (d), we additionally annotate the error of the recursive approximation (blue dotted line) and the GPU runtime of SHFFT recursive (green dotted line). The CPU runtime of the SHFFT recursive–6.74 μs in (c) and 5.25 μs in (d)—exceeds the maximum scale of the plot and is therefore omitted. We also mark $t = 3n - 2$ for our approximate method. At this setting, our method achieves both lower error and better performance than SHFFT recursive.

accordingly. This trade-off aligns with the theoretical analysis in Sec. 6.2, demonstrating that the resolution parameter t provides an effective mechanism for balancing accuracy and performance in practice.

Fig. 5 (a) and (b) evaluate generalized SH product tasks with different input and output orders. Fig. 5 (c) and (d) correspond to specialized truncated multiple product tasks. Here we also report the error and performance of the state-of-the-art SHFFT recursive method. Consistent with the discussion in Sec. 6.2, when $t = 3n - 2$, our approximate method achieves both lower error and better performance than *SHFFT recursive*, making this choice of t a practical and effective configuration for such tasks.

7.2.2 Visual Comparison. To provide an intuitive understanding of the approximation error, we visualize representative results in Fig. 6 and Fig. 7. In Fig. 6, the input SH coefficients are uniformly sampled from $[-1, 1]$, while in Fig. 7, the inputs are band-limited visibility functions extracted from real scenes. We consider two types of tasks, generalized SH products and specialized truncated SH products, to comprehensively illustrate the behavior of our approximate method.

For generalized products, we show approximate results computed with two different resolution parameters t. These examples demonstrate that our method naturally supports multiple approximation levels, enabling a flexible trade-off between accuracy and performance by adjusting t.

For specialized truncated products, we compare our approximate results computed with $t = 3n - 2$ against those produced by the recursive approximation. The visual comparisons show that our method introduces noticeably smaller errors, consistent with the quantitative results reported in Fig. 5.

7.3 Rendering Applications

Existing methods for computing spherical harmonics products [Ng et al. 2004; Xin et al. 2021; Zhou et al. 2005] are constrained by the assumption that all input SH orders must be identical. This limitation is suboptimal for rendering applications, where lighting, BRDFs, and visibility often exhibit distinct frequency characteristics. Adapting the SH order for each component individually enables a more efficient trade-off between accuracy and computational cost. Our method removes this constraint by introducing a generalized formulation of SH products that supports arbitrary input and output orders. By simply replacing traditional SH product modules with our method, existing rendering pipelines can achieve improved performance. Moreover, they gain the flexibility to assign appropriate SH orders to different scene components.

To demonstrate these practical benefits, we integrate our method into two SH-based real-time rendering pipelines: glossy relighting [Ng et al. 2004; Sloan et al. 2002] and shadow fields [Zhou et al. 2005]. Both pipelines require runtime evaluation of SH triple or multiple product integrals, which can be directly computed by setting $p = 1$ in our method.

7.3.1 Glossy Relighting using PRT. In this application, SH triple product integral is the main computational bottleneck, as triple product integrals of lighting, BRDFs, and visibility are evaluated at runtime.

For the specialized triple product integral, we compare *ours accurate* (Table 1(d)) against *traditional accurate* and *SHFFT accurate* on two scenes with dynamic lighting and BRDFs, including one with environment lighting [Ramamoorthi and Hanrahan 2001] and another with area lights [Wang and Ramamoorthi 2018]. All three methods produce numerically identical results. The rendering results are shown in Fig. 8, and the corresponding performance is summarized in Table 2(a). Compared to the best performance of previous methods, our approach achieves a 3.46× speedup for Fig. 8(a) and a 3.59× speedup for Fig. 8(b).

7.3.2 Shadow Fields. Shadow fields [Zhou et al. 2005] extend the PRT framework to support dynamic visibility by precomputing an object occlusion field (OOF) for each object in the scene. Each OOF stores SH coefficients that encode visibility in a look-up table and

can be efficiently queried at runtime for a given shading point. In this application, SH multiple product integral is the main computational bottleneck, as multiple product integrals of lighting, self-visibility and visibility queried from the OOFs are evaluated at runtime.

For the specialized multiple product integral, we compare multiple methods, including *ours accurate*, *ours approximate*, *SHFFT accurate*, *SHFFT recursive* and *traditional recursive* on several dynamic scenes. The rendering results are shown in Fig. 1 (b) and Fig. 9, and the performance is summarized in Table 2(b). Our accurate method consistently outperforms *SHFFT accurate* across all tested scenes, achieving speedups ranging from 3.79× to 5.96×. Meanwhile, *ours approximate* produces results that are nearly indistinguishable from the accurate method, while outperforming the best prior approximate method with speedups ranging from 3.79× to 5.96× and achieving better quantitative metrics.

7.3.3 Generalized SH products for glossy relighting and shadow fields. Moreover, our method supports generalized SH product integrals with arbitrary input orders, allowing lighting, BRDFs, and visibility to be represented at different SH orders for flexible control. It is important to note that such generalized cases are not explicitly supported by existing specialized SH product methods, which assume identical input orders. We show several rendering examples under these generalized settings in Fig. 1(a), Fig. 10, and Fig. 11, and provide the performance in Table 2(c). All results are computed using our accurate method with the parameter configuration from Table 1(a). In Fig. 1, the teapot uses a higher SH order (11) for its BRDF, while the dragon uses a lower order (5), resulting in a glossier appearance for the teapot. In Fig. 11, the blue sphere uses a higher SH order (9) for its object occlusion field (OOF), while the pink sphere uses a lower SH order (3), producing sharper shadows beneath the blue sphere and softer shadows beneath the pink one. These examples highlight an important observation: different rendering components, such as BRDFs and visibility, often exhibit distinct frequency characteristics and can benefit from being represented at different SH orders. However, previous work has not explicitly addressed the product of SH inputs with different orders, limiting the flexibility of rendering applications that rely on SH products. By introducing the generalized SH product formulation, our work explicitly identifies and supports this need, improving both the efficiency and flexibility of the representation.

8 Discussions and Conclusion

Conclusion. In this work, we introduce the concept of generalized spherical harmonics products, which support arbitrary input SH orders and enable flexible truncation of the output. By introducing Spherical Grids from other scientific disciplines into computer graphics and extending them as intermediate representations, we propose a unified and efficient framework for computing generalized SH products. We also derive sufficient grid resolution conditions for exact evaluation and present an approximate variant that enables a tunable trade-off between accuracy and efficiency. Even in specialized cases with identical input SH orders, our method significantly outperforms state-of-the-art approaches in speed while achieving lower approximation error. The ability to handle varying input orders further enhances the flexibility of our method.

Grid Efficiency. Currently, our method utilizes the SphGrid representation, which requires a grid resolution of $n \times (2n-1)$ to fully reconstruct order-n SH coefficients (See Eq. 35). It is worth noting, however, that there are only n^2 independent SH coefficients, while the grid contains $n(2n-1)$ samples. The oversampling creates data redundancy, which is most obvious near the poles where points are packed much more densely than at the equator. While this introduces overhead, the grid structure allows us to separate the latitude and longitude coordinates, which significantly speeds up the computation. This is precisely why we adopt this grid in our method. Alternative spherical sampling schemes, such as the Spherical Fibonacci Grid (SFG) [Marques et al. 2013] and HEALPix [Gorski et al. 2005], can achieve more uniform point distributions on the sphere. However, the SFG lacks iso-latitude structure, leading to an SH transform complexity of $O(n^4)$, compared to the $O(n^3)$ complexity of our SphGrid. HEALPix provides a hierarchical equal-area grid but does not offer a known analytic exact SH transform, whereas our SphGrid supports exact bidirectional conversion with proven resolution conditions. Exploring ways to further reduce the number of required samples while maintaining the efficiency of product computation presents a promising direction for future research.

Time complexity and practical performance. While asymptotic complexity characterizes how algorithm runtime scales with problem size, actual performance depends on many factors, such as constant factors and memory access patterns. Although our method and SHFFT share competitive asymptotic complexity, our simpler formulation leads to lower constant factors, resulting in substantial speedups for the low SH orders ($n \leq 20$) commonly used in graphics. For very high SH orders, where asymptotic complexity becomes the dominant factor, the SH transform algorithms from mathematics and geophysics (Section 2.3), which have slightly lower asymptotic complexity, could potentially replace our $O(n^3)$ bidirectional conversion to achieve better performance.

SH vs. SphGrid representations. The SphGrid representation introduced in our work may inspire future research on point-based representations as primary representations for spherical signals. Compared to SHs, SphGrid offers significant advantages in product computation but lacks closed-form solutions for rotation, scaling, gradients, and spherical/polygonal lighting. This limitation restricts its applicability as a primary representation in real-time rendering and merits further investigation. Our generalized SH product method leverages the strengths of both representations. It uses SphGrid as an intermediate representation to accelerate product computation while maintaining closure in the SH domain, where both inputs and outputs are SH coefficients that can seamlessly integrate with existing SH operations. We hope our method not only provides a practical solution to generalized SH products, but also inspires further research on efficient representations and computations for spherical signals in graphics.

Acknowledgments

We would like to thank all anonymous reviewers for their insightful comments and valuable suggestions. This work is supported by the National Natural Science Foundation of China (Project No. 62372257).

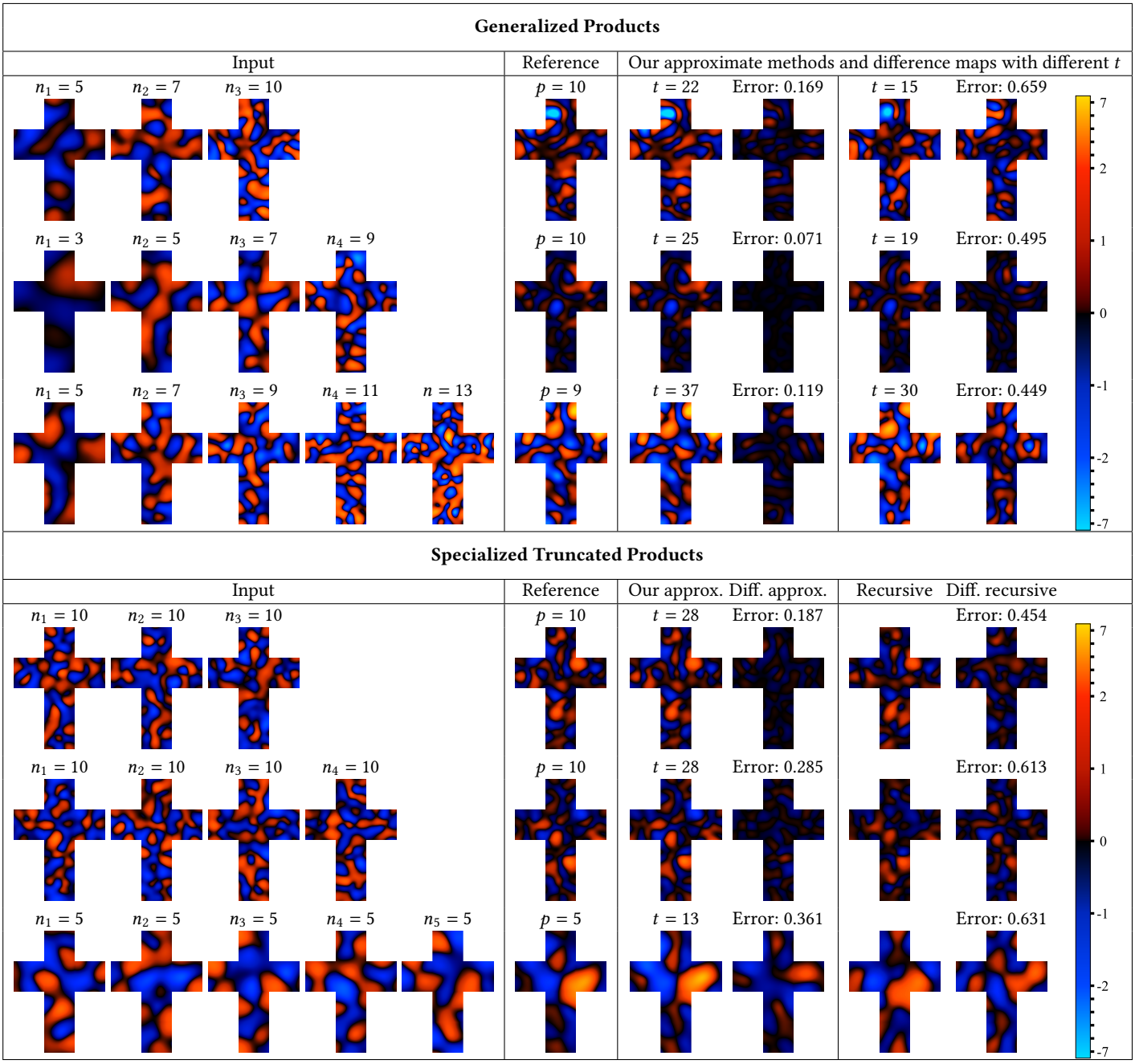


Fig. 6. Visualization of SH product examples on randomly generated inputs of band-limited functions. From left to right, we show the input functions and the ground truth. For generalized SH products, we show the outputs of our approximate method and the corresponding difference maps under different resolution parameters t. For specialized truncated SH products, we show the output of our approximate method with $t = 3n - 2$ and its difference map, as well as the output of the recursive method and its difference map. The relative L_2 errors of SH coefficients between the approximate results and the reference are also reported.

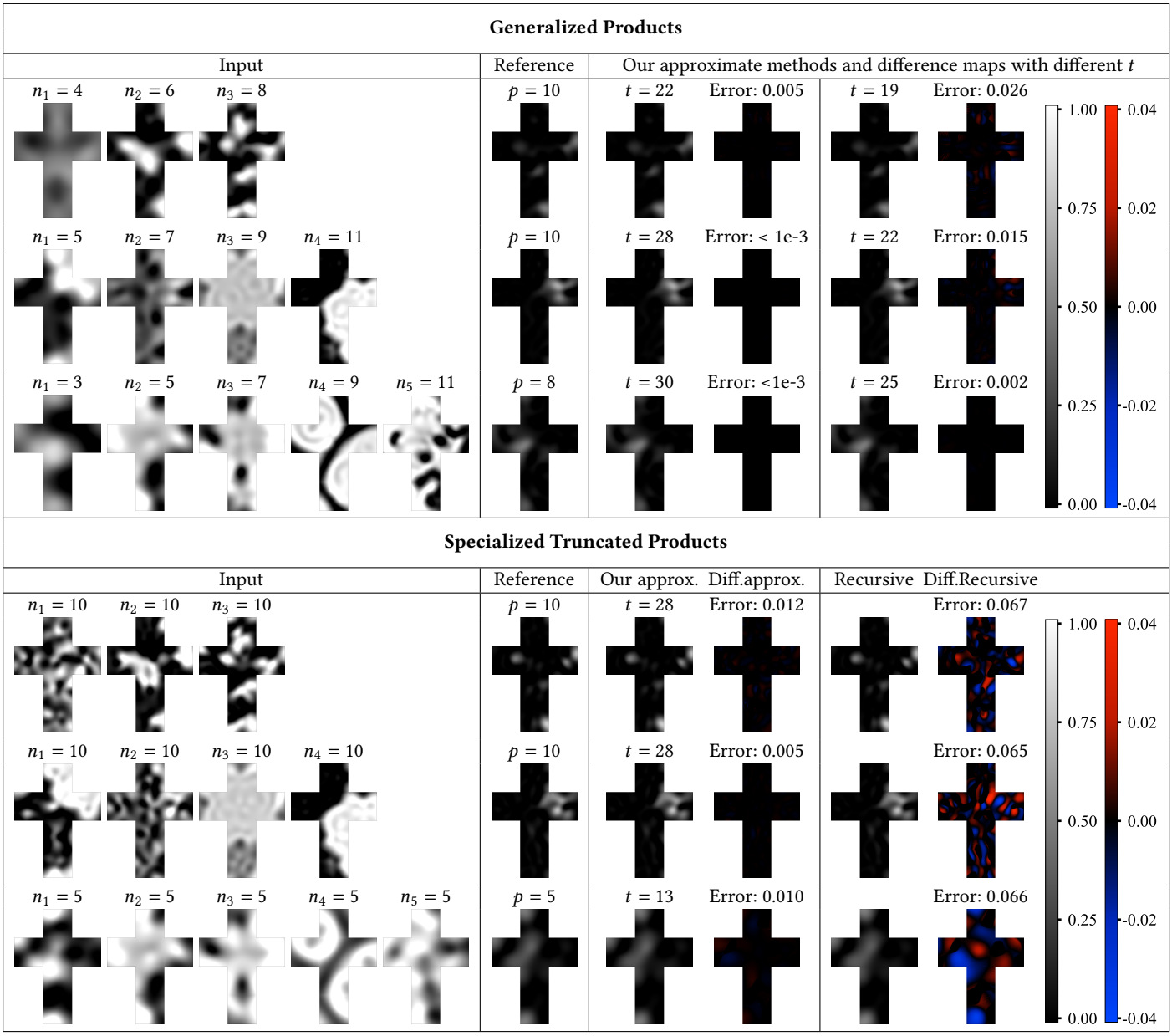


Fig. 7. Visualization of SH product examples on band-limited visibility function inputs from real scenes. From left to right, we show the input functions and the ground truth. For generalized SH products, we show the outputs of our approximate method and the corresponding difference maps under different resolution parameters t. For specialized truncated SH products, we show the output of our approximate method with $t = 3n - 2$ and its difference map, as well as the output of the recursive method and its difference map. Visibility functions are visualized using a grayscale colorbar, while difference maps are visualized using a blue–red colorbar. The relative L_2 errors of SH coefficients between the approximate results and the reference are also reported.

References

Feng Dai and Yuan Xu. 2013. *Cubature Formulas on Spheres.* Springer New York, New York, NY, 127–153. doi:10.1007/978-1-4614-6660-4_6

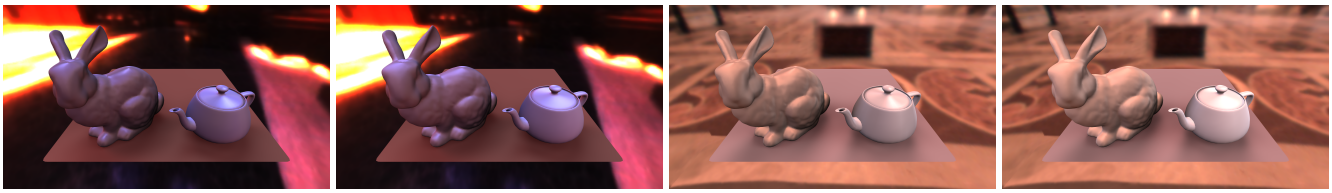
JR Driscoll and DM Healy. 1994. Computing Fourier Transforms and Convolutions on the 2-Sphere. *Advances in Applied Mathematics* 15, 2 (1994), 202–250.

K. M. Gorski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. 2005. HEALPix: A FRAMEWORK FOR HIGH-RESOLUTION DISCRETIZATION AND FAST ANALYSIS OF DATA DISTRIBUTED ON THE SPHERE. *Astrophysical Journal* 622, 2Pt1 (2005), p.759–771.

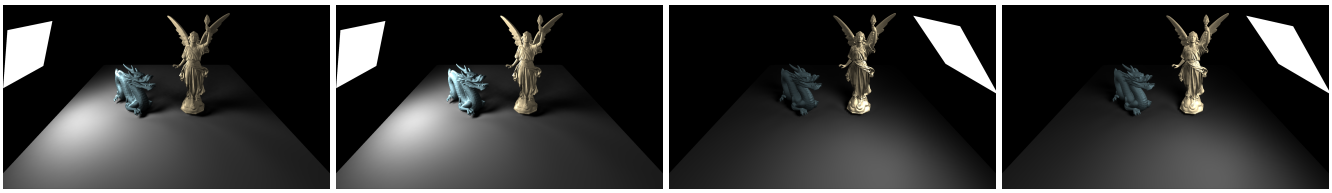
Christian Gruber, Pavel Novák, and Josef Sebera. 2011. FFT-based high-performance spherical harmonic transformation. *Studia Geophysica et Geodaetica* 55 (2011), 489–500.

Xuejun Hao and Amitabh Varshney. 2004. Real-time rendering of translucent meshes. *ACM Transactions on Graphics* 23, 2 (2004), 120–142.

Dennis M Healy, Daniel N Rockmore, Peter J Kostelec, and Sean Moore. 2003. FFTs for the 2-sphere-improvements and variations. *Journal of Fourier analysis and applications* 9 (2003), 341–385.

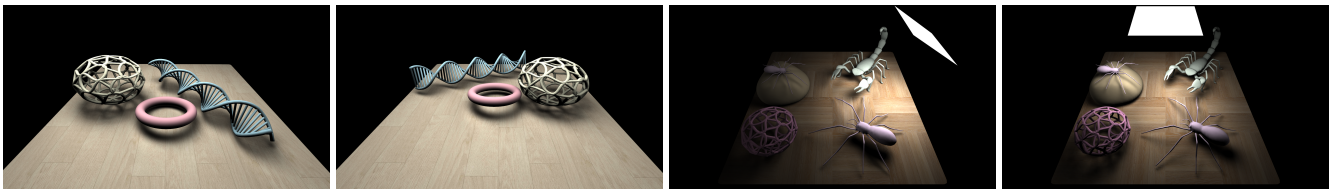


(a) Rendered with SH order $n = 10$. Framerates: *traditional accurate*: 26.43 FPS, *SHFFT accurate*: 31.29 FPS, *ours accurate*: 108.29 FPS



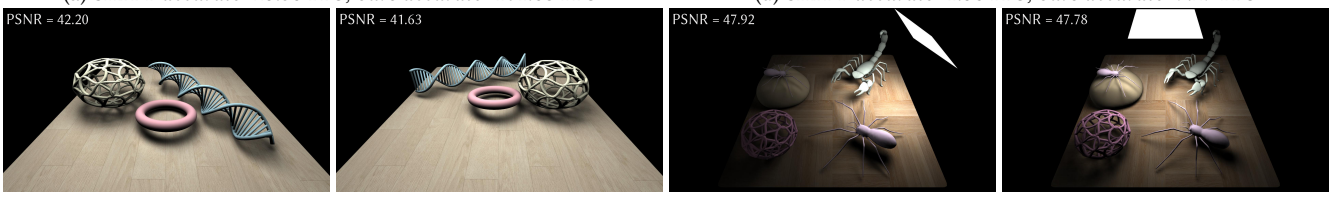
(b) Rendered with SH order $n = 15$. Framerates: *traditional accurate*: 1.45 FPS, *SHFFT accurate*: 4.41 FPS, *ours accurate*: 15.83 FPS

Fig. 8. Examples of glossy relighting using PRT with different methods for the specialized triple product integral. Our method is significantly faster than the state-of-the-art method *SHFFT accurate*.



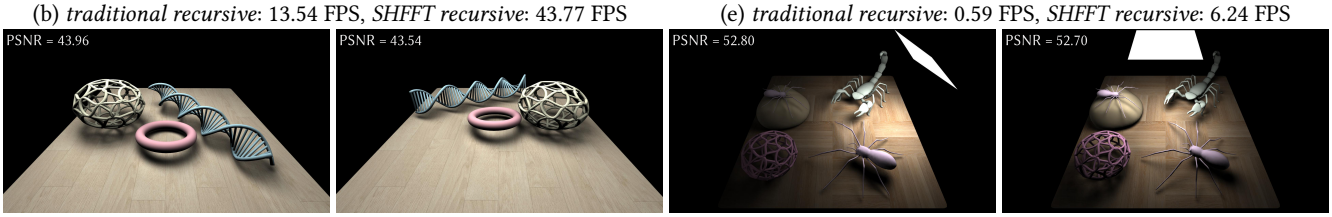
(a) *SHFFT accurate*: 25.35 FPS, *ours accurate*: 101.68 FPS

(d) *SHFFT accurate*: 1.56 FPS, *ours accurate*: 9.29 FPS



(b) *traditional recursive*: 13.54 FPS, *SHFFT recursive*: 43.77 FPS

(e) *traditional recursive*: 0.59 FPS, *SHFFT recursive*: 6.24 FPS



(c) *ours approximate*: 119.15 FPS

(f) *ours approximate*: 39.16 FPS

Fig. 9. Examples of shadow fields with different methods for the specialized multiple products integral. (a–c) are rendered with SH order $n = 10$, and (d–f) with $n = 15$. Both our accurate and approximate methods are significantly faster than the best performance of previous methods. Besides, the rendering results of our approximate method achieve better quantitative metrics (i.e., higher PSNR values) than the recursive approximation.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

Jaroslav Krivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. 2005. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics* 11, 5 (2005), 550–561.

Jaakko Lehtinen. 2007. A framework for precomputed and captured light transport. *ACM Transactions on Graphics (TOG)* 26, 4 (2007), 13–es.

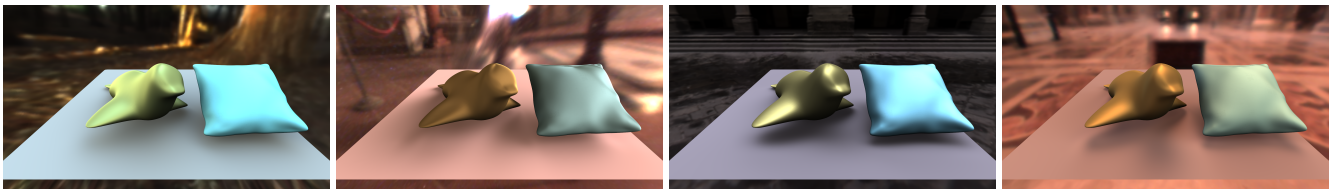
Christian Lessig, Mathieu Desbrun, and Eugene Fiume. 2014. A constructive theory of sampling for image synthesis using reproducing kernel bases. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–14.

Xinguo Liu, Peter-Pike J Sloan, Heung-Yeung Shum, and John Snyder. 2004. All-Frequency Precomputed Radiance Transfer for Glossy Objects. *Rendering Techniques* 2004 (2004).

R. Marques, C. Bouville, M. Ribardière, L. P. Santos, and K. Bouatouch. 2013. Spherical Fibonacci Point Sets for Illumination Integrals. *Computer Graphics Forum* 32, 8 (2013), 134–143.

Martin J Mohlenkamp. 1999. A fast transform for spherical harmonics. *Journal of Fourier analysis and applications* 5 (1999), 159–184.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.



(a) Lighting SH order = 5 (b) Lighting SH order = 7 (c) Lighting SH order = 9 (d) Lighting SH order = 11

Fig. 10. Examples of glossy relighting using PRT with our accurate method for the generalized triple product integral. The seal BRDF is represented with SH order 11, while the pillow BRDF uses order 5. All visibility terms use SH order 9.

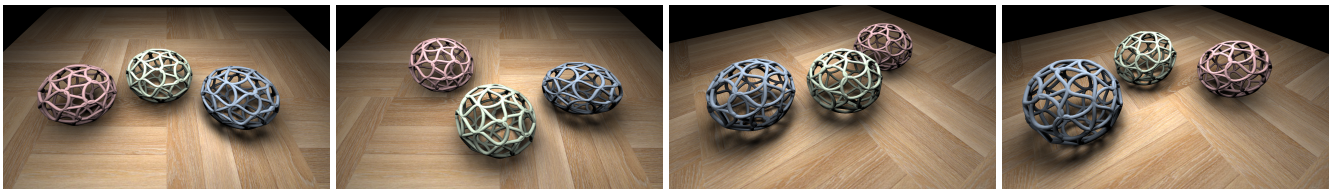


Fig. 11. Examples of shadow fields with our accurate method for the generalized multiple product integral. The lighting is represented with SH order 9. The OOF terms of the three spheres use SH orders 3 (pink), 5 (green) and 9 (blue).

Table 2. Performance in real-time rendering applications involving SH product integrals. (a) Specialized triple product integral: glossy relighting using PRT [Sloan et al. 2002], where triple SH product integrals of lighting, BRDFs and visibility are evaluated at runtime. (b) Specialized multiple product integral: shadow fields [Zhou et al. 2005] for dynamic scenes, where multiple SH product integrals of lighting, self-visibility and visibility queried from the OOFs are evaluated at runtime. (c) Generalized triple or multiple product integral: glossy relighting and shadow fields with different input SH orders, which uses our accurate method as Table 1 (a). For the specialized product integral, We report frame per second (FPS) for baselines and our methods. Speedups are measured relative to the best performance of prior approach. For the generalized product integral, we report FPS only for our accurate method, as existing methods cannot handle this case.

(a) Glossy Relighting using PRT (Specialized Triple Product Integral)

Scene	Figure	Vertices	Parameters	FPS (Trad.)	FPS (SHFFT)	FPS (Ours)	Speedup
Bunny & Teapot	Fig. 8 (a)	45,488	$n = 10$	26.43	31.29	108.29	3.46×
Dragon & Lucy	Fig. 8 (b)	98,699	$n = 15$	1.45	4.41	15.83	3.59×

(b) Shadow Fields (Specialized Multiple Product Integral)

Scene	Figure	Vertices	Parameters	Accurate (FPS)			Approximate (FPS)			
				SHFFT	Ours	Speedup	Trad.	SHFFT	Ours	Speedup
Bunny & Scorpions	Fig. 1 (b)	101,687	$k = 6, n = 12$	3.17	12.02	3.79×	1.66	8.03	44.75	5.57×
DNA & Ring	Fig. 9 (a) (b) (c)	36,125	$k = 4, n = 10$	25.35	101.68	4.01×	13.54	43.77	119.15	2.72×
Spider & Scorpion	Fig. 9 (d) (e) (f)	78,223	$k = 6, n = 15$	1.56	9.29	5.96×	0.59	6.24	39.16	6.28×

(c) Glossy Relighting and Shadow Fields (Generalized Triple or Multiple Product Integral)

Task	Scene	Figure	Vertices	Lighting SH order	Ours accurate (FPS)
Generalized triple product integral	Dragon & Lucy & Teapot	Fig. 1 (a)	100,781	9	171.18
	Seal and Pillow	Fig. 10 (a)	37,581	5	176.73
		Fig. 10 (b)		7	168.92
		Fig. 10 (c)		9	161.76
		Fig. 10 (d)		11	155.03
Generalized multiple product integral	Three balls	Fig. 11	36,945	9	145.56

Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. 2004. Triple product wavelet integrals for all-frequency relighting. *ACM Transactions on Graphics* 23, 3 (2004), 477–487.

Derek Nowrouzezahrai, Patricio Simari, and Eugene Fiume. 2012. Sparse zonal harmonic factorization for efficient SH rotation. *ACM Transactions on Graphics (TOG)* 31, 3 (2012), 1–9.

William H Press. 2007. *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge university press.

Ravi Ramamoorthi et al. 2009. Precomputation-based rendering. *Foundations and Trends® in Computer Graphics and Vision* 3, 4 (2009), 281–369.

Ravi Ramamoorthi and Pat Hanrahan. 2001. An Efficient Representation for Irradiance Environment Maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01).* 497–500.

Ravi Ramamoorthi and Pat Hanrahan. 2002. Frequency space environment map rendering. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques.* 517–526.

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 Transactions on*

Graphics 25, 3 (2006), 977–986.

Vladimir Rokhlin and Mark Tygert. 2006. Fast algorithms for spherical harmonic expansions. *SIAM Journal on Scientific Computing* 27, 6 (2006), 1903–1928.

Ataru Sakuraba and Paul H Roberts. 2009. Generation of a strong magnetic field using uniform heat flux at the surface of the core. *Nature Geoscience* 2, 11 (2009), 802–805.

Nathanaël Schaeffer. 2013. Efficient spherical harmonic transforms aimed at pseudospectral numerical simulations. *Geochemistry, Geophysics, Geosystems* 14, 3 (2013), 751–758.

Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 527–536.

Peter-Pike Sloan, Jaakko Lehtinen, and Jan Kautz. 2005a. Precomputed Radiance Transfer: Theory and Practice. *SIGGRAPH courses* (2005).

Peter-Pike Sloan, Xinguo Liu, Heung-Yeung Shum, and John Snyder. 2003. Bi-scale radiance transfer. *ACM Transactions on Graphics* 22, 3 (2003), 370–375.

Peter-Pike Sloan, Ben Luna, and John Snyder. 2005b. Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 1216–1224.

Reiji Suda and Masayasu Takami. 2002. A fast spherical harmonics transform algorithm. *Math. Comp.* 71, 238 (2002), 703–715.

Weifeng Sun and Amar Mukherjee. 2006. Generalized wavelet product integral for rendering dynamic glossy objects. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 955–966.

Jingwen Wang and Ravi Ramamoorthi. 2018. Analytic spherical harmonic coefficients for polygonal area lights. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.

Jiaping Wang, Kun Xu, Kun Zhou, Stephen Lin, Shimin Hu, and Baining Guo. 2006. Spherical harmonics scaling. *The Visual Computer* 22 (2006), 713–720.

Mark A Wieczorek and Matthias Meschede. 2018. SHTools: Tools for working with spherical harmonics. *Geochemistry, Geophysics, Geosystems* 19, 8 (2018), 2574–2592.

Lifan Wu, Guangyan Cai, Shuang Zhao, and Ravi Ramamoorthi. 2020. Analytic spherical harmonic gradients for real-time rendering with many polygonal area lights. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 134–1.

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 Transactions on Graphics (TOG)* 40, 6 (2021), 1–14.

Youxin Xing, Gaole Pan, Xiang Chen, Ji Wu, Lu Wang, and Beibei Wang. 2024. Real-time all-frequency global illumination with radiance caching. *Computational Visual Media* 10, 5 (2024), 923–936.

Kun Xu, Yue Gao, Yong Li, Tao Ju, and Shi-Min Hu. 2007. Real-time homogenous translucent material editing. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 545–552.

Tianyi Yang, Meng Duan, Zixuan Li, and Beibei Wang. 2026. Real-time woven fabric rendering using SGGX fitting. *Computational Visual Media* 12, 1 (2026), 159–171.

Shinyoung Yi, Donggun Kim, Jiwoong Na, Xin Tong, and Min H Kim. 2024. Spin-Weighted Spherical Harmonics for Polarized Light Transport. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–24.

Kun Zhou, Yaohua Hu, Stephen Lin, Baining Guo, and Heung-Yeung Shum. 2005. Precomputed shadow fields for dynamic scenes. *ACM Transactions on Graphics* 24, 3 (2005), 1196–1201.

A Proof of Eq. 15

Consider the product of k band-limited spherical functions $F_1, F_2, \ldots, F_k$, where each F_i is exactly represented by order-n_i SH coefficients $\mathbf{f}_i^{(n_i)}$. We first show the case for $k = 2$.

Case $k = 2$. Let F_1 and F_2 have SH orders n_1 and n_2, respectively:

$$\begin{aligned} &F_1(\theta,\phi)F_2(\theta,\phi) \\ &= \left(\sum_{l_1=0}^{n_1-1}\sum_{m_1=-l_1}^{l_1} f_{1,l_1}^{m_1}\, y_{l_1}^{m_1}(\theta,\phi)\right)\left(\sum_{l_2=0}^{n_2-1}\sum_{m_2=-l_2}^{l_2} f_{2,l_2}^{m_2}\, y_{l_2}^{m_2}(\theta,\phi)\right) \\ &= \sum_{l_1=0}^{n_1-1}\sum_{m_1=-l_1}^{l_1}\sum_{l_2=0}^{n_2-1}\sum_{m_2=-l_2}^{l_2} f_{1,l_1}^{m_1} f_{2,l_2}^{m_2}\, y_{l_1}^{m_1}(\theta,\phi)\, y_{l_2}^{m_2}(\theta,\phi). \end{aligned} \tag{43}$$

The product of two SH basis functions can be expanded as [Driscoll and Healy 1994]:

$$\begin{aligned} &y_{l_1}^{m_1}(\theta,\phi)\, y_{l_2}^{m_2}(\theta,\phi) \\ &= \sum_{L=|l_1-l_2|}^{l_1+l_2} \sqrt{\frac{(2l_1+1)(2l_2+1)}{4\pi(2L+1)}}\, W_{0,0,0}^{l_1,l_2,L}\, W_{m_1,m_2,m_1+m_2}^{l_1,l_2,L}\, y_L^{m_1+m_2}(\theta,\phi), \end{aligned} \tag{44}$$

where W denotes Wigner 3-j symbols, and $y_L^{m_1+m_2}$ is defined to be zero if $|m_1 + m_2| > L$.

From Eq. (44), the maximum degree $L_{\max}$ in the product satisfies:

$$L_{\max} = l_1 + l_2 \le (n_1 - 1) + (n_2 - 1) = n_1 + n_2 - 2.$$

Therefore, the product F_1F_2 is band-limited with full SH order

$$n_G = n_1 + n_2 - 1. \tag{45}$$

General Case $k \ge 2$. By recursively applying the above argument, the product of k band-limited functions $F_1, \ldots, F_k$, with SH orders $n_1, \ldots, n_k$, is also band-limited. Its full SH order is given by

$$n_G = \left(\sum_{i=1}^{k} n_i\right) - k + 1, \tag{46}$$

which proves Eq. 15.

B Proof of Eq. 33

Given an order-n band-limited function F, we aim to show that for any fixed polar angle θ and integer m ($|m| < n$), the following identity holds when $N_\phi \ge n + |m|$:

$$\int_0^{2\pi} F(\theta,\phi)R_m(\phi)\,\mathrm{d}\phi = \frac{2\pi}{N_\phi}\sum_{j=0}^{N_\phi-1} F(\theta,\phi_j)R_m(\phi_j), \tag{47}$$

where the azimuthal angles $\{\phi_j\}$ are uniformly sampled over $[0, 2\pi)$:

$$\phi_j = \frac{2\pi j}{N_\phi}, \quad j = 0, 1, \ldots, N_\phi - 1. \tag{48}$$

Left-hand side. Since F is order-n band-limited, it admits the exact reconstruction with order-n SH coefficients $\mathbf{f}$

$$F(\theta,\phi) = \sum_{l'=0}^{n-1}\sum_{m'=-l'}^{l'} f_{l'}^{m'}\, y_{l'}^{m'}(\theta,\phi) = \sum_{l'=0}^{n-1}\sum_{m'=-l'}^{l'} f_{l'}^{m'}\, C_{l'}^{m'}(\theta)\, R_{m'}(\phi). \tag{49}$$

Substituting Eq. (49) into the left-hand side of Eq. (47) and interchanging summation and integration yields

$$\int_0^{2\pi} F(\theta,\phi)R_m(\phi)\,\mathrm{d}\phi = \sum_{l'=0}^{n-1}\sum_{m'=-l'}^{l'} f_{l'}^{m'} C_{l'}^{m'}(\theta)\int_0^{2\pi} R_m(\phi)R_{m'}(\phi)\,\mathrm{d}\phi. \tag{50}$$

The products of trigonometric functions satisfy the following orthogonality property:

$$\int_0^{2\pi} \sin(m\phi)\sin(m'\phi)\,\mathrm{d}\phi = \begin{cases} \pi, & m = m' \ne 0, \\ 0, & m = m' = 0, \\ 0, & m \ne m', \end{cases} \tag{51}$$

$$\int_0^{2\pi} \cos(m\phi)\cos(m'\phi)\,\mathrm{d}\phi = \begin{cases} \pi, & m = m' \ne 0, \\ 2\pi, & m = m' = 0, \\ 0, & m \ne m', \end{cases} \tag{52}$$

$$\int_0^{2\pi} \sin(m\phi)\cos(m'\phi)\,\mathrm{d}\phi = 0. \tag{53}$$

Recall that $R_m(\phi)$ is a trigonometric function of frequency $|m|$ with normalization factor $\sqrt{2}$ for $m \ne 0$ (Eq. 21), we have

$$\int_0^{2\pi} R_m(\phi)R_{m'}(\phi)\,\mathrm{d}\phi = \begin{cases} 2\pi, & m = m', \\ 0, & m \ne m'. \end{cases} \tag{54}$$

Therefore, only the terms with $m = m'$ contribute, and Eq. (50) simplifies to

$$\int_0^{2\pi} F(\theta,\phi)R_m(\phi)\,\mathrm{d}\phi = 2\pi\sum_{l'=|m|}^{n-1} f_{l'}^{m} C_{l'}^{m}(\theta). \tag{55}$$

Right-hand side. Substituting the expansion in Eq. (49) into the discrete summation on the right-hand side of Eq. (47) gives

$$\sum_{j=0}^{N_\phi-1} F(\theta,\phi_j)R_m(\phi_j) = \sum_{l'=0}^{n-1}\sum_{m'=-l'}^{l'} f_{l'}^{m'} C_{l'}^{m'}(\theta)\sum_{j=0}^{N_\phi-1} R_m(\phi_j)R_{m'}(\phi_j). \tag{56}$$

Since the azimuthal samples $\{\phi_j\}$ are uniformly distributed over $[0, 2\pi)$, for any integer t satisfying $|t| < N_\phi$, the discrete sums of $\sin(t\phi_j)$ and $\cos(t\phi_j)$ exhibit exact symmetry over a full period:

$$\sum_{j=0}^{N_\phi-1} \sin(t\phi_j) = 0, \qquad \sum_{j=0}^{N_\phi-1} \cos(t\phi_j) = \begin{cases} N_\phi, & t = 0, \\ 0, & t \ne 0. \end{cases} \tag{57}$$

When $N_\phi \ge n + |m|$, since $|m'| < n$, we have $|m \pm m'| < n + |m| \le N_\phi$. By applying the product-to-sum identities, we obtain

$$\sum_{j=0}^{N_\phi-1} \sin(m\phi_j)\sin(m'\phi_j) = \begin{cases} N_\phi/2 & m = m' \ne 0 \\ 0 & m = m' = 0 \\ 0 & m \ne m' \end{cases} \tag{58}$$

$$\sum_{j=0}^{N_\phi-1} \cos(m\phi_j)\cos(m'\phi_j) = \begin{cases} N_\phi/2 & m = m' \ne 0 \\ N_\phi & m = m' = 0 \\ 0 & m \ne m' \end{cases} \tag{59}$$

$$\sum_{j=0}^{N_\phi-1} \cos(m\phi_j)\sin(m'\phi_j) = 0. \tag{60}$$

Then the discrete sums of the product $R_m(\phi)R_{m'}(\phi)$ can be simplified as

$$\sum_{j=0}^{N_\phi-1} R_m(\phi_j)R_{m'}(\phi_j) = \begin{cases} N_\phi, & m = m', \\ 0, & m \neq m'. \end{cases} \tag{61}$$

Similarly, only the terms with $m = m'$ contribute. Substituting Eq. (61) into Eq. (56) yields

$$\sum_{j=0}^{N_\phi-1} F(\theta,\phi_j)R_m(\phi_j) = N_\phi \sum_{l'=|m|}^{n-1} f_{l'}^m C_{l'}^m(\theta). \tag{62}$$

Conclusion. Comparing Eq. (55) and Eq. (62), we conclude that Eq. (47) holds when $N_\phi \geq n + |m|$, completing the proof.

C Proof of Eq. 30

Given an order-n band-limited function F, we aim to show that for any integers l and m satisfying $0 \leq l < n$ and $-l \leq m \leq l$, the following identity holds when $2N_\theta \geq n + l$:

$$\int_0^\pi C_l^m(\theta)\, s_m(\theta)\, \sin\theta \,\mathrm{d}\theta = \sum_{i=0}^{N_\theta-1} \omega_i\, C_l^m(\theta_i)\, s_m(\theta_i), \tag{63}$$

where

$$s_m(\theta) = \int_0^{2\pi} F(\theta,\phi)\, R_m(\phi)\, \mathrm{d}\phi, \tag{64}$$

$\{\omega_i\}$ are the Gauss-Legendre weights, and $\{\cos\theta_i\}$ are the Gauss-Legendre nodes.

To establish Eq. 63, we first recall the Gauss-Legendre quadrature rule [Press 2007], which states that for any polynomial $f(x)$ of degree at most $2n-1$, the following equality holds:

$$\int_{-1}^{1} f(x)\,\mathrm{d}x = \sum_{i=0}^{n-1} \omega_i\, f(x_i), \tag{65}$$

where $\{x_i\}$ are also the Gauss-Legendre nodes. By introducing the change of variables $x = \cos\theta$, Eq. 65 can be equivalently written as

$$\int_0^\pi f(\cos\theta)\, \sin\theta \,\mathrm{d}\theta = \sum_{i=0}^{n-1} \omega_i\, f(\cos\theta_i). \tag{66}$$

From Eq. 66, it immediately follows that Eq. 63 holds if the integrand $C_l^m(\theta)\, s_m(\theta)$ can be expressed as a polynomial in $\cos\theta$ of degree at most $2N_\theta - 1$.

Recall that F is an order-n band-limited function with SH coefficients $\mathbf{f}$. From Eq. 55, we have

$$s_m(\theta) = \int_0^{2\pi} F(\theta,\phi)R_m(\phi)\,\mathrm{d}\phi = 2\pi \sum_{l'=|m|}^{n-1} f_{l'}^m C_{l'}^m(\theta) \tag{67}$$

Since $C_l^m(\theta) = N_l^{|m|} P_l^{|m|}(\cos\theta)$ (Eq. 21), we can expand

$$C_l^m(\theta)\, s_m(\theta) = 2\pi\, N_l^{|m|} P_l^{|m|}(\cos\theta) \sum_{l'=|m|}^{n-1} f_{l'}^m\, N_{l'}^{|m|} P_{l'}^{|m|}(\cos\theta). \tag{68}$$

Therefore, it suffices to show that for any non-negative integers $l, l' < n$,

$$P_l^{|m|}(\cos\theta)\, P_{l'}^{|m|}(\cos\theta) \tag{69}$$

is a polynomial in $\cos\theta$ of degree at most $2N_\theta - 1$.

From Eq. 3, the associated Legendre polynomial admits the following decomposition:

$$P_l^m(x) = (1-x^2)^{\frac{m}{2}}\, A_l^m(x), \tag{70}$$

where $A_l^m(x)$ is a polynomial of degree at most $l - m$. Applying Eq. 70 to Eq. 69, we obtain

$$P_l^{|m|}(\cos\theta)\, P_{l'}^{|m|}(\cos\theta) = (1-\cos^2\theta)^{|m|} A_l^{|m|}(\cos\theta)\, A_{l'}^{|m|}(\cos\theta), \tag{71}$$

which is clearly a polynomial in $\cos\theta$ with degree at most

$$2|m| + (l - |m|) + (l' - |m|) = l + l'. \tag{72}$$

Since $0 \leq l' < n$ and $2N_\theta \geq n + l$, we have

$$l + l' \leq n + l - 1 \leq 2N_\theta - 1. \tag{73}$$

Then the Gauss-Legendre quadrature rule applies exactly, then the Eq. 63 is therefore proved.