Karlsruhe Institute of Technology

Linearly Transformed Spherical Harmonic Expansions

Bachelor-Thesis von

Jan Allmenröder

An der Fakultät für Informatik
Institut für Visualisierung und Datenanalyse,
Lehrstuhl für Computergrafik

25. März 2020

Erstgutachter:	Prof. Dr.-Ing. Carsten Dachsbacher
Zweitgutachter:	Prof. Dr. Hartmut Prautzsch
Betreuender Mitarbeiter:	Dr. Christoph Peters

Bearbeitungszeitraum:	07. November 2019 - 06. April 2020

Table of Contents

1 Abstract . . . 1
2 Zusammenfassung . . . 2
3 Introduction . . . 3
4 Related Work . . . 5
5 Basics . . . 7
 5.1 Rendering Equation . . . 7
 5.2 BRDF . . . 8
 5.3 Spherical Functions . . . 11
 5.4 Linearly Transformed Spherical Functions . . . 12
6 Spherical Harmonic Fitting . . . 15
 6.1 Optimizing the SH coefficients . . . 15
 6.2 Optimizing Linear Transformations . . . 16
7 Polygonal Light Shading with LTSH Expansions . . . 18
 7.1 Lighting Space . . . 18
 7.2 Integration of Spherical Harmonic Expansions . . . 19
 7.3 Solving the Rendering Equation . . . 20
8 Results . . . 22
 8.1 Implementation . . . 22
 8.2 Quality Comparison . . . 24
 8.3 Run Times . . . 34
9 Conclusion and Future Work . . . 36
10 Acknowledgments . . . 38

Bibliography **39**

1 Abstract

In this thesis, we use spherical harmonic (SH) expansions with linear transformations to approximate common BRDFs. An existing technique uses linearly transformed cosines to approximate BRDFs and performs shading with them. We swap out the rigid cosine function with more versatile SH expansions to improve the quality of the BRDF approximation. SH expansions possess an analytic integral over polygonal domains and we use them for shading with Lambertian, planar, polygonal area lights. We show, how good approximations of BRDFs are obtained using a linear optimization for the SH coefficients inside a non-linear optimization for the parameters of the linear transformation. We explain, how the approximations are integrated over a polygonal domain and describe our implementation for the use in shading. Our technique produces realistic renderings and offers improvement in image quality compared to linearly transformed cosines, but it performs significantly slower.

2 Zusammenfassung

In dieser Bachelorarbeit benutzen wir Kugelflächenfunktionen zusammen mit linearen Transformationen um gängige BRDFs zu approximieren. Ein bereits existierender Ansatz benutzt linear transformierte Kosinusfunktionen, um die BRDF zu approximieren und verwendet diese im Shading. Wir ersetzen die unflexiblen Kosinusfunktionen durch Kugelflächenfunktionen, um die Qualität der BRDF Approximation zu verbessern. Kugelflächenfunktionen besitzen ein analytisches Integral für polygonale Integrationsdomänen und wir benutzen sie für Shading mit Lambertschen, planaren, polygonalen Flächenlichtquellen. Wir beschreiben, wie man gute Approximationen der BRDFs finden kann, indem wir lineare Optimierung für die Koeffizienten der Kugelflächenfunktion innerhalb einer nicht-linearen Optimierung für die Parameter der linearen Transformation benutzen. Wir erklären weiterhin, wie diese Approximationen über ein Polygon integriert werden und erklären, wie unsere Shader Implementierung funktioniert. Unsere Technik erzeugt realistische Bilder und bietet verbesserte Qualität gegenüber linear transformierten Kosinusfunktionen, aber sie ist deutlich langsamer.

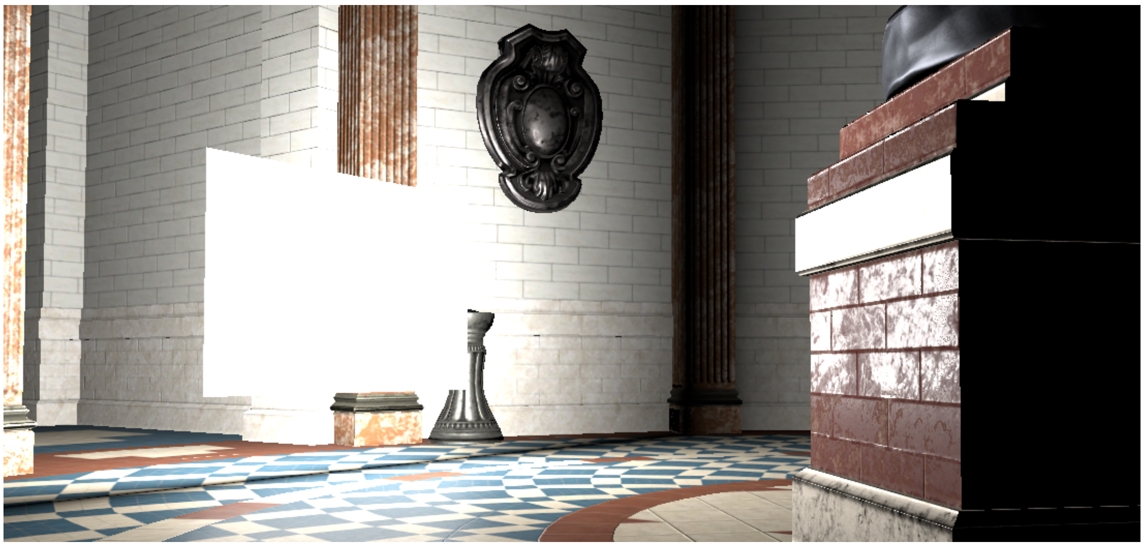


Figure 1: Rendering with a rectangular area light source using linearly transformed spherical harmonics.

3 Introduction

In real-time rendering directional lights and point lights remain the most common light sources. Point lights assume, that light is sent out from an infinitely small volume of space (hence “point” light) and thus lack an equivalent in the real world. Light bulbs, for example, are often represented in real-time graphics as a point light although a spherical light source would be a better match.
A similar problem exists with planar area light sources. Sampling the light source remains too costly for most real-time settings and existing approximations often introduce visible artifacts. In reaction, artists often alter material parameters in order to resemble area lighting [AMHH^{+}18, p. 377ff] or avoid it altogether. This prohibits the use of common light types of the real world (e.g. an LED panel) or makes the scene creation prone to errors, if material parameters correlate to the light setting.
Using polygonal area lights for physically based real-time rendering applications is a challenging problem, since it involves the integration of bidirectional reflectance distribution functions (BRDFs) over polygonal domains. Analytic integrals for modern physically based distributions have not been found yet and thus the approximation of either the light source or the distribution function is a common approach in tackling this problem.
Linearly transformed cosines (LTC) make the use of area lights in real-time settings more convenient and are directly integrated into the Unity engine [HDHN16]. However, LTC still introduce artifacts, especially in the specular reflection of light sources close to a rough surface (which happens often e.g. with light sources attached to a wall) or at grazing angles.
Our approach is to refine the existing technique of linearly transformed spherical distributions [HDHN16]. To achieve this, we replace the inflexible clamped cosine function with more versatile spherical harmonic expansions, since their integration over polygonal domains possesses a closed form expression [WR18, BXH^{+}18].
First, we give an overview of important scientific advances in real-time area lighting as well as in integrating spherical functions over polygonal domains in section 4.
We then introduce common concepts in computer graphics such as the rendering equation and BRDFs and introduce spherical functions and their linear transformations (section 5). Meanwhile, we fix notation and conventions used throughout our work.
In a preprocess, our technique finds linearly transformed spherical harmonic (LTSH) ex-

pansions (section 6). Here, our approximation needs to fit the properties of modern BRDFs as closely as possible. To this end, we find an optimal LTSH expansion given a linear transformation and rate its quality to then use this information to explore the space of possible linear transformations and to find a good fit. The following integration of our approximated BRDF is exact, which limits the quality of our technique only to the quality of our BRDF approximation with LTSH.
Furthermore, we explain the integration of LTSH expansions over polygonal domains and outline the process of transforming a polygon from world space to the domain of an untransformed spherical function (section 7). We then combine both steps to solve the rendering equation for local area lighting.
Following that, we give an idea of how practical our novel technique is, in a brief description of how we implement it both for fitting BRDFs and real-time rendering of scene geometry. We compare the quality of our approximated BRDF and rendered images against linearly transformed cosines [HDHN16] and ground truth. Besides image quality, we also compare run times and memory usage of LTC and LTSH (section 8).
Lastly, in section 9 we give a conclusion to our discoveries, point out advantages and disadvantages compared to other common techniques and elaborate on aspects worth further research.

4 Related Work

Using area lights for real time local illumination introduces some complexity compared to shading with point lights. If computed correctly or approximated closely, it can introduce substantial improvement in image quality and thus is a large area of research in computer graphics.
Arvo [Arv95] derived an analytic expression to solve local illumination with polygonal lights. Using irradiance tensors to calculate exact lighting evaluations he enabled polygonal lights with the Phong model. Those irradiance tensors are reduced to boundary integrals by the Stoke's theorem, which are then evaluated analytically for polygons. Since the tensors represent different orders n, they are used to calculate the integral of $\cos^n$, which closely relates to the Phong model, over polygonal domains.
A similar approach [LDSM16], leverages irradiance tensors and introduces an edge splitting technique to approximate more realistic physically based microfacet BRDFs. This way, it speeds up the integration to $\mathcal{O}(v)$ from $\mathcal{O}(nv)$ where v is the number of polygon vertices and n is the Phong exponent, by approximating the integral with analytically integrable peak shape functions.
Another frequently used technique is the most representative point (MRP) approach. Here the lighting gets reduced to evaluating a single light point, which is obtained by different approximation methods. Picott [Pic92] uses the closest point to the reflected view vector. Wang et al. [WLWF08] also take into account the distance of the light source to blur remote lights and enable arbitrary intensity distributions. Those methods remain approximate and contain visible artifacts in various configurations.
The recent advances in hardware, allowing for more and more use of ray tracing in real-time rendering pushes sampling strategies in the focus of research. To solve the illumination integral, it is a common approach to sample the integral and thus approximate it. A good sampling strategy is needed to reduce noise. Simple strategies involve sampling of the incident light or the BRDF. For example, the introduction of the GGX BRDF [WMLT07] came alongside a sampling strategy for it and common light sources have well known sampling strategies [PJH16, ch. 14.2].
Far better results are obtained by taking both the incident light and the BRDF into account. This is done by combining samples from multiple distributions and correctly weighting them (often called multiple importance sampling) [VG95], or by directly sampling the product of both the incident lighting function and the BRDF (product sampling) [CJAMJ05].
A further advantage of sampling based approaches is that they narrow down the integration domain to a limited set of samples and do not seek an analytic expression for it. This leaves the space open for additional integral terms like visibility functions or contributions from participating media. They enable effects like shadows and foggy scenes among others and improve realism significantly. Analytic approaches like LTC [HDHN16] and our own fail to produce these effects and have to be combined with other techniques to achieve realistic lighting.
Another frequently used type of area lights are spherical lights. Recent advances [DHB17, PD19] make spherical lights more suitable for real-time rendering in dealing with the complexity of sampling spherical caps. This is done for specular rendering by approximating the BRDF with a transformed spherical function suitable for closed form evaluation [DHB17]. For diffuse rendering, spherical caps are decomposed or approximated by cut disks which can be efficiently sampled [PD19].
The inspiration for our technique comes from LTC [HDHN16]. They approximate common microfacet BRDFs with LTC which are then used to analytically solve the shading integral. Here, the shading integral reduces to an integral of the cosine function over a polygonal domain. It thus possesses a closed form expression. A more detailed explanation

follows in section 5.4. Although they provide a fast and accurate method in most cases, at grazing angles and with high intensities it produces noticeable differences compared to ground truth.
We approach the problem in a similar way using LTSH to approximate the BRDF. The novel spherical harmonic (SH) integration techniques [BXH$^+$18, WR18] enable us to solve the emerging integration of SH expansions over polygonal domains. The concurrent work in both papers is built upon the zonal decomposition of SH expansions [NSF12] with shared lobe directions and describes a closed form expression of the integral. Through the shared lobes combined with an iterative approach, the integral is computed in time $\mathcal{O}(n^2)$, where $n = N + 1$ and N is the order of the SH expansion. A more detailed introduction to SH integration will follow in section 7.2. Combining the techniques [HDHN16, BXH$^+$18, WR18] we are able to derive our LTSH technique to leverage the more versatile SH expansions in place of simple cosine functions.
As a followup to LTCs, the technique is extended to support line and disk lights [HH17]. To achieve this, they change the domain of the cosine integral to a spherical segment or a spherical ellipse respectively. This is noteworthy for us, since those extensions could work with our technique in a similar manner. Integrating SH over spherical segments is probably done straight forwardly, whereas integration over spherical ellipses may not be trivial. Further elaboration on this topic is an interesting field for future work.

5 Basics

In this section we introduce some fundamental concepts which are used throughout this thesis. In section 5.1 we discuss the rendering equation and ways to solve it. Section 5.2 introduces the concept of BRDFs alongside some popular BRDF models. Then we give a brief introduction to spherical functions in section 5.3 before discussing their linear transformation in section 5.4.

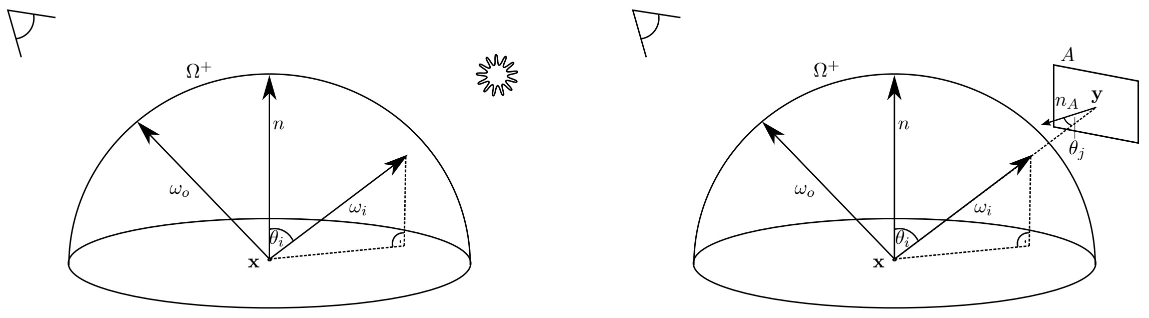
5.1 Rendering Equation

The rendering equation (RE)[Kaj86] is a generalization of earlier rendering algorithms. It is stated as:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega^+} f_r(\omega_i, \mathbf{x}, \omega_o) L_i(\mathbf{x}, \omega_i) \cos\theta_i d\omega_i \tag{1}$$

Where $L_o(\mathbf{x}, \omega_o)$ denotes the outgoing radiance at the shading point $\mathbf{x}$ from the view direction ω_o. Furthermore, $L_e(\mathbf{x}, \omega_o)$ denotes the emitted radiance at shading point $\mathbf{x}$ in direction ω_o. Ω^+ is the set of all directions on the positive hemisphere along the surface normal n at shading point $\mathbf{x}$ and since it is our integration domain, it is where the incident direction ω_i is taken from. The BRDF $f_r(\omega_i, \mathbf{x}, \omega_o)$ states how much incoming radiance from the incident direction ω_i is reflected to the outgoing direction ω_o at shading point $\mathbf{x}$. $L_i(\mathbf{x}, \omega_i)$ then denotes the incoming radiance from the hemispherical direction ω_i at shading point $\mathbf{x}$. θ_i denotes the angle between the incident direction ω_i and the surface normal n at shading point $\mathbf{x}$.
Since our technique only solves local area lighting, we simplify the equation. The RE can be reformulated over surface radiance and the surface of the area light is thus the only surface of importance for us. We obtain the RE for area lights [SM09, p. 526f] (the visibility term is omitted for simplicity):

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\mathbf{y} \in A} f_r(\omega_i, \mathbf{x}, \omega_o) L_e(\mathbf{y}, -\omega_i) \frac{\cos\theta_i \cos\theta_j}{\|\mathbf{y} - \mathbf{x}\|^2} d\mathbf{y} \tag{2}$$



(a) Point light (b) Area light

Figure 2: Visualization of the rendering equation.

Here, A is the polygonal area of the planar area light, clipped to the horizon and $\mathbf{y} \in A$ is a point on the area light. This leads to $\omega_i = \frac{\mathbf{y}-\mathbf{x}}{\|\mathbf{y}-\mathbf{x}\|}$ being the unit vector pointing from shading point $\mathbf{x}$ to light point $\mathbf{y}$. θ_j denotes the angle between the incident direction ω_i and the surface normal of the area light plane n_A. This narrows the integration domain down to the polygonal area of the light.
A common way to solve this equation is by Monte-Carlo sampling. It provides a probabilistic approach to numerically solve integrals. By replacing the light area A with a discrete

set of points $\tilde{A}$ of size $N = |\tilde{A}|$ and $\tilde{A} \subset A$ the domain of integration becomes discrete and thus solvable. The simplest way of sampling those points is uniformly inside A. Those uniform samples $\tilde{y} \in \tilde{A}$ then also have a uniform probability density $p(\tilde{y}) = \frac{1}{Area(A)}$. Given the Monte-Carlo formula for an arbitrary function $f(x)$:

$$\int f(x)dx \approx \frac{1}{N} \sum_{j=1}^{N} \frac{f(x_j)}{p(x_j)} \tag{3}$$

which in the case of the area light resolves into

$$L(\mathbf{x}, \omega_o) \approx \frac{Area(A)\ L}{N} \sum_{j=1}^{N} f_r(\omega_{i,j}, \mathbf{x}, \omega_o) \frac{\cos\theta_i \cos\theta_j}{\|\tilde{\mathbf{y}}_\mathbf{j} - \mathbf{x}\|^2} \tag{4}$$

with $\omega_{i,j} = \frac{\tilde{\mathbf{y}}_\mathbf{j} - \mathbf{x}}{\|\tilde{\mathbf{y}}_\mathbf{j} - \mathbf{x}\|^2}$ and L the spatially constant intensity of the light source. We obtain an approximate closed form expression of the equation. More sophisticated sampling strategies can be used but in general sampling is compromised by increased computational cost and thus is often unsuitable for real-time rendering. We seek to derive a closed form expression for the integral to compute an exact result with acceptable effort.

5.2 BRDF

To simulate realistic light transport, materials are often described as a bidirectional reflectance distribution function (BRDF). This function describes how much incoming light from a direction ω_i is reflected to a specific view direction ω_o (see f_r in section 5.1).
Note that we focus on specular BRDFs in this work. In theory, diffuse BRDFs can be handled in a similar manner by our technique. However, the untransformed clamped cosine (also known as Lambertian diffuse) provides a suitable diffuse model for our needs.
A simple model for specular surfaces is the Phong model [SM09, p. 236ff]. It is described as follows:

$$f_{Phong}(\omega_i, \mathbf{x}, \omega_o) = k_{specular} \max(0, \omega_o \cdot r)^p \tag{5}$$

Here $k_{specular}$ is an RGB material constant defining the specular color, $r = -\omega_i + 2(\omega_i \cdot n)n$ is the reflected incident direction ω_i at surface normal n and p is the Phong exponent, defining the sharpness of the specular highlight. However, it lacks a physical basis and is seldomly used in modern graphics.
To enable realistic light transport, there is a variety of physically based microfacet BRDFs. We focus on the GGX model [WMLT07], which introduces the following formula:

$$f_r(\omega_i, \mathbf{x}, \omega_o) = \frac{F(\omega_i, h)\ G(\omega_i, \omega_o, h)\ D_{GGX}(h)}{4\ |\omega_i \cdot n|\ |\omega_o \cdot n|} \tag{6}$$

Here $h = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}$ denotes the half-vector between ω_i and ω_o and thus represents the surface normal, at which light is reflected from ω_i to ω_o. F is the Fresnel-term, G the shadowing masking function and D_{GGX} the microfacet distribution function, on which we elaborate now.
The function D_{GGX} is a distribution of surface normals on the micro surface level. The scene geometry defines surface normals on a coarse level without enough detail for realistic shading. D_{GGX} states the proportion of surface normals aligned with the incident and outgoing directions to reflect light from one to another. As a visualization, we imagine the surface as a set of differently aligned infinitesimally small mirrors. The D_{GGX} function then states, how many of the mirror normals are aligned in a way, that light from the incident direction is reflected to the outgoing direction. In other words, it states how many mirror normals align with the half-vector h, as we show in figure 3. This enables realistic

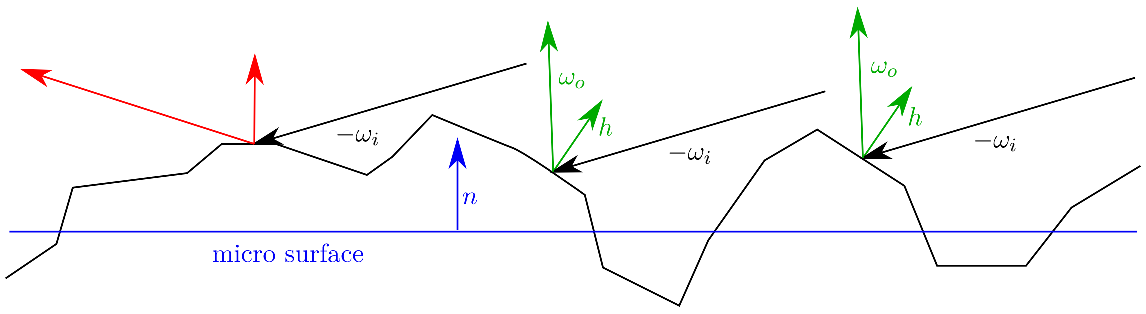


Figure 3: Visualization of microfacettes. Here, two of the microfacettes are aligned with h and reflect light from the incident direction ω_i to the outgoing direction ω_o.

shading of rough surfaces, where the micro surface normals contain a lot of variation. We use the GGX distribution:

$$D_{GGX}(h) = \frac{\alpha^2 \chi^+(h \cdot n)}{\pi \cos^4 \theta_h \left(\alpha^2 + \tan^2 \theta_h\right)^2}. \tag{7}$$

$$\chi^+(a) = \begin{cases} 1 & a > 0 \\ 0 & a \leq 0 \end{cases} \tag{8}$$

Here, α is the surface roughness defined by the material, n is the macro surface normal and θ_h is the angle between the half-vector h and the normal n.
In general, more light gets reflected, if θ_h is small. For rough surfaces ($\alpha \approx 1$), the difference in reflected light for different values of θ_h is comparatively small, meaning that light gets scattered in every direction, whereas for glossy surfaces ($\alpha \approx 0$) almost no light gets reflected for $\theta_h \neq 0$ and all light gets reflected for $\theta_h = 0$. This makes glossy surfaces more dependent on θ_h and the peak of $D_{GGX}(h)$ is sharper (figure 4).

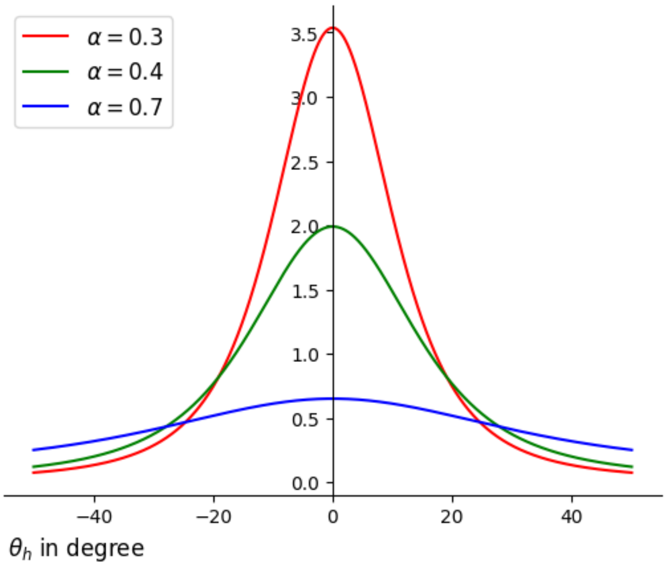


Figure 4: $D_{GGX}(\theta_h)$ plotted for different values of α.

The function G, depending on the distribution D_{GGX}, accounts for shadowing and masking of light at the micro surface level. Shadowing means, that incoming light is blocked by the geometry of the micro surface itself, before reaching its destined surface point. Masking is the blocking of reflected light by the micro surface. See a visualization of both effects in

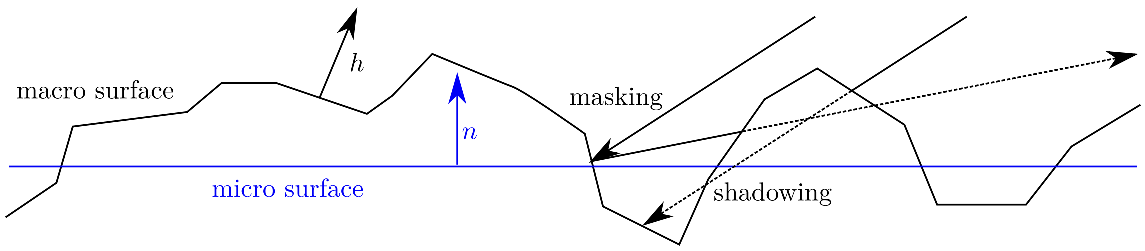


Figure 5: Visualization of shadowing and masking

figure 5. Again, the scene geometry is not sufficiently detailed to account for this shadowing and masking. We make use of the Smith masking function [WMLT07], which is proven to be correct [Hei14].

$$G(\omega_i, \omega_o, h) = G_1(\omega_i, h)G_1(\omega_o, h) \tag{9}$$

$$G_1(\omega, h) = \chi^+\left(\frac{\omega \cdot h}{\omega \cdot n}\right)\frac{2}{1+\sqrt{1+\alpha^2\tan^2(\theta_\omega)}} \tag{10}$$

θ_ω is the angle between the direction ω and the macro surface normal n.
Both the D_{GGX} and the G term depend heavily on the material at the shading point $\mathbf{x}$, more specifically the roughness of the material. Since the roughness is defined per shading point and does not vary for different view or light directions, it is not specified as a parameter of the BRDF in terms of the RE. However, it is noteworthy, that different roughness values are crucial for realistic light transport and we need to account for them later on.
The Fresnel-term F indicates the ratio between incoming light being reflected or transmitted by the material. It models the physical Fresnel effect where more light is reflected at grazing angles compared to steep incident directions. Although exact equations are available [CT82], we make use of the common Schlick-Fresnel approximation [Sch94] (see equation 11) to avoid drawbacks [Hof19]. The Fresnel-term depends on the incoming and outgoing directions as well as an index of refraction. This index is defined by the material at the shading point $\mathbf{x}$.

$$F(\omega_i, h) = f_0 + (1 - f_0)(1 - h \cdot \omega_i)^5 \tag{11}$$

Here, f_0 is the Fresnel value defined by the physical properties of the surface material.
There are two classes of BRDFs commonly used: isotropic and anisotropic. Isotropic BRDFs are independent of the azimuthal orientation of ω_o and ω_i in tangent space, as long as their azimuthal difference remains unchanged, whereas anisotropic BRDFs depend on the azimuthal orientation of ω_o and ω_i in tangent space. In other words, if both directions ω_o and ω_i are rotated around the z-axis, the value of an isotopic BRDF does not change, whereas for an anisotropic BRDF the value might change. Most materials can be described sufficiently by isotropic BRDFs, which is why we only focus on those in the following work. Extending our model to anisotropic BRDFs would be possible, but it would introduce another dimension to our fitting and thus vastly increase the time needed to find BRDF approximations and video memory needed in shading. The shading performance should be mainly unaffected by it.

BRDFs should always satisfy the following conditions to match their physical model in optics [SM09, p. 639ff]:

$$f_r(\omega_i, \mathbf{x}, \omega_o) = f_r(\omega_o, \mathbf{x}, \omega_i) \quad \text{reciprocity} \tag{12}$$

$$\int_\Omega f_r(\omega_i, \mathbf{x}, \omega_o) \, \cos\theta_i d\omega_i \leq 1 \quad \text{energy conservation} \tag{13}$$

The GGX BRDF [WMLT07] fulfills those requirements and leaves us with a much more realistic appearance of surfaces compared to the simpler Phong model. However, the complexity of the equation does not allow for analytic integration whereas integration of the Phong model over polygonal domains is a solved problem [Arv95]. To make use of the realistic properties of the GGX BRDF and other physically based specular BRDFs, we thus approximate it with functions suitable to integration.

5.3 Spherical Functions

Real spherical functions are restricted to the unit sphere $\mathbb{S}^2 \subset \mathbb{R}^3$ and map unit vectors to real scalars.

$$\text{spherical functions} \quad f : \mathbb{S}^2 \to \mathbb{R} \tag{14}$$

Since BRDFs are also defined on the spherical domain, spherical functions offer a good entry point to look for approximate functions. The Cartesian coordinates (x, y, z) of $\mathbb{R}^3$ can be expressed in spherical coordinates (r, θ, φ) as in equation 15. Intuitively, r is the length of the vector, θ is the angle between the z-axis and the vector and φ is the angle in the xy-plane as shown in figure 6. These coordinates are reduced to the unit sphere $\mathbb{S}^2$ by setting $r = 1$, leaving us with equation 16. The incoming and outgoing directions ω_i and ω_o are unit vectors and can thus be represented with (θ, φ).

$$\begin{aligned} x &= r\sin\theta\cos\varphi & r &\geq 0 \\ y &= r\sin\theta\sin\varphi & \theta &\in [0, \pi] \\ z &= r\cos\theta & \varphi &\in [0, 2\pi] \end{aligned} \tag{15}$$

$$(x, y, z) = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta) \tag{16}$$

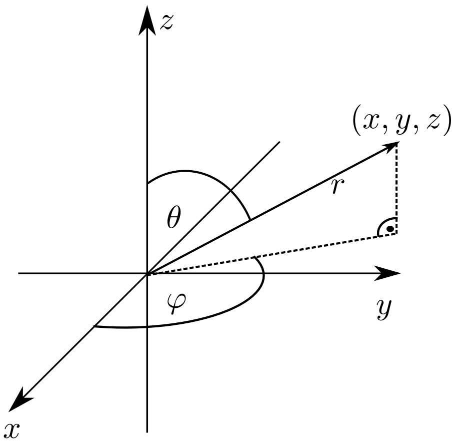


Figure 6: Spherical coordinates

A simple example of a spherical function is the cosine function clamped against 0: $f(\theta, \varphi) = \max(0, \cos\theta)$. This function is 0 in the lower hemisphere and can fulfill energy conservation introduced in equation 13 through a normalization constant $\frac{1}{\pi}$. Thus it matches basic properties of BRDFs by definition and is a suitable candidate for BRDF approximation.

Another family of spherical functions frequently used in computer graphics are spherical harmonics (SH) [Slo08]. As a harmonic function they are the solution to Laplace's equation [Mac67, p. 69]. They can be thought of as Fourier expansions on the spherical domain. Every spherical function can be expressed as an SH expansion with possibly an infinite number of frequencies. The SH basis functions form an orthonormal basis [Slo08] and are given in equation 17. Here l denotes the band of the coefficient, consisting of $2l + 1$ band functions indexed from $m = -l$ to $m = l$. K_l^m is a normalization constant as shown in equation 18 and P_l^m is the associated Legendre polynomial.

$$Y_l^m(\theta, \varphi) = K_l^m e^{im\varphi} P_l^{|m|}(\cos\theta), \; l \in \mathbb{N}, -l \leq m \leq l \tag{17}$$

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}} \tag{18}$$

This definition of SH with complex values can be defined as a real basis for our sake as shown in equation 19.

$$y_l^m(\theta, \varphi) = \begin{cases} \sqrt{2} K_l^m \cos(m\varphi) P_l^m(\cos\theta) & m > 0 \\ \sqrt{2} K_l^m \sin(|m|\varphi) P_l^{|m|}(\cos\theta) & m < 0 \\ K_l^0 P_l^0(\cos\theta) & m = 0 \end{cases} \tag{19}$$

A function f can be expressed as an SH expansion of degree N with coefficients c_l^m like this:

$$f(\theta, \varphi) = \sum_{l=0}^{(N+1)^2} \sum_{m=-l}^{l} c_l^m y_l^m \tag{20}$$

An important subset of the spherical harmonics are the zonal harmonics (ZH). The ZH functions are the SH basis functions where $m = 0$, y_l^0. They are radially symmetric around the z-axis, making them independent of φ and only correspond to the z-coordinate in Cartesian coordinates [WR18]. They can be directly expressed as:

$$y_l^0(\theta, \varphi) = \sqrt{\frac{2l+1}{4\pi}} P_l^0(\cos\theta) \tag{21}$$

While every spherical function can be expressed as an SH expansion, it is often necessary to limit the bands of the expansion. The SH basis makes it easy to determine the optimal coefficients for approximating a function [BXH+18]. This results in an approximation of the spherical function with a fixed number of coefficients. Since every band introduces $2l + 1$ coefficients respectively to their band functions, an SH expansion of order or degree N (consisting of $N + 1$ bands since the first band is indexed with 0) consists of $(N + 1)^2$ coefficients. This means SH expansions become heavily compute intensive for higher orders.

5.4 Linearly Transformed Spherical Functions

Linearly transforming spherical functions possessing analytic integral forms allows for integration of functions that closely match the properties of modern BRDFs [DHB17, HDHN16]. Here, a function with simple properties but with an analytic integral form, like the clamped cosine function introduced in section 5.3, are transformed linearly to resemble BRDFs as closely as possible. For computing lighting over polygonal [HDHN16] or spherical [DHB17] domains, the original geometry in world space needs to be inversely transformed. Then, the integral over the inversely transformed geometry on the original spherical function can be computed exactly. Figure 7 shows a clamped cosine fitted to the GGX BRDF.

Our main contribution is to swap out the rigid cosine function used by Heitz et al. [HDHN16] with more flexible SH expansions to match the properties of BRDFs more closely and thus achieve a more realistic approximation of polygonal area lighting.
Our SH expansions is transformed in the same way as clamped cosine functions or any

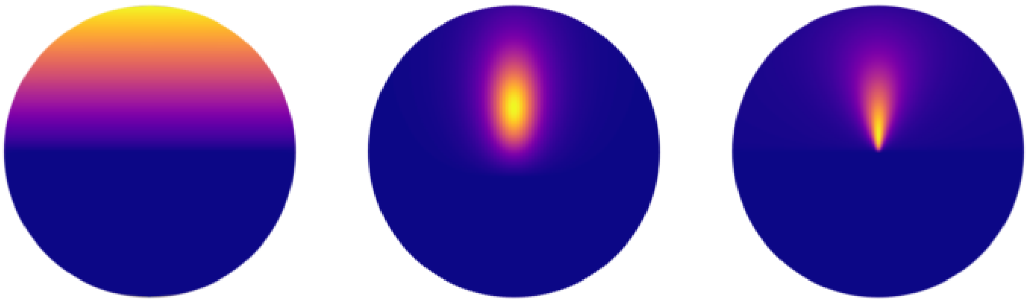


Figure 7: From left to right: clamped cosine, linearly transformed clamped cosine, GGX BRDF. $\omega_o = 88,6°$and $\alpha = 0.4$.

other spherical distribution. The following mathematical derivations originate from Heitz et al. [HDHN16] but are repeated here for convenience, since they are crucial to our work. We now assume an original spherical function D_{org}, that can be evaluated on the unit sphere with a unit vector $\omega_{org} \in \mathbb{S}^2$. Further assuming a linear transformation described by a 3×3 matrix M, a linearly transformed spherical function D is obtained as shown in equation 22 and likewise inversely as shown in equation 23. Here ω is the linearly transformed unit vector $\omega = \frac{M\omega_{org}}{\|M\omega_{org}\|}$ and $|M|$ denotes the magnitude of the matrix determinant of M with $|M| = |\det(M)|$. The term $\frac{\partial \omega_{org}}{\partial \omega} = \frac{|M^{-1}|}{\|M^{-1}\omega\|^3}$ will be called the Jacobian from now on and a derivation of it can be found in the appendix of Heitz et al. [HDHN16].

$$D_{org}(\omega_{org}) = D(\omega)\frac{\partial \omega}{\partial \omega_{org}} = D\left(\frac{M\omega_{org}}{\|M\omega_{org}\|}\right)\frac{|M|}{\|M\omega_{org}\|^3} \tag{22}$$

$$D(\omega) = D_{org}(\omega_{org})\frac{\partial \omega_{org}}{\partial \omega} = D_{org}\left(\frac{M^{-1}\omega}{\|M^{-1}\omega\|}\right)\frac{|M^{-1}|}{\|M^{-1}\omega\|^3} \tag{23}$$

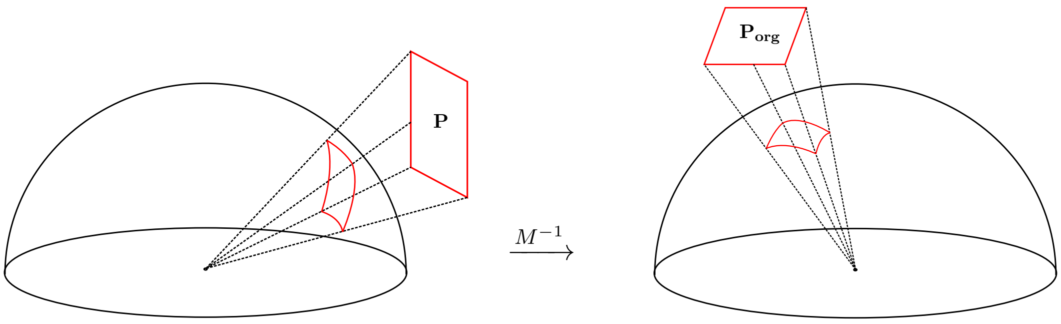


Figure 8: Inverse linear transformation of a polygon on the hemisphere.

With this, we can transform a spherical function in a way that resembles the GGX BRDF as closely as possible. For example, we transform most of our spherical points of the original distribution into a small area for specular highlights. The more glossy the surface becomes, the closer together we transform the points. This way we achieve delicate features in our transformed function, although the original spherical function is much more coarse.
We then use the inverse linear transformation to bring a polygon into the domain of our original function and calculate the integral of this function over the polygon.
The introduced integration only concerns the original function, since the Jacobian term is canceled out as a part of the transformation (see equation 24). We thus only account for the Jacobian in our fitting preprocess and drop it entirely for our shading.
We then reduce the sphere integral to the polygonal domain of our light source, because it

is the only contribution to incident lighting in the sphere for direct lighting. With this, we transform the shading integration domain to the domain of the original spherical function. This is done as shown in equation 25, where $\mathbf{P_{org}}$ simply denotes the inversely transformed polygon $M^{-1}\mathbf{P}$ (see figure 8). This reduces the shading integral to an integral of the chosen untransformed function over a polygonal domain (or other integration domains for other types of light as mentioned in section 3). The spherical basis function and the integration domain are not fixed and swapped out easily, making it a versatile approach for the shading of different light types with flexible quality and performance. The complexity of the basis function is adjusted to the needed quality of the BRDF approximation and to the time budget in rendering.

$$\int_{\Omega} D(\omega)d\omega = \int_{\Omega} D_{org}(\omega_{org})\frac{\partial\omega_{org}}{\partial\omega}d\omega = \int_{\Omega} D_{org}(\omega_{org})d\omega_{org} \tag{24}$$

$$\int_{\mathbf{P}} D(\omega)d\omega = \int_{\mathbf{P_{org}}} D_{org}(\omega_{org})d\omega_{org} \tag{25}$$

For LTCs, the inversely transformed polygon $\mathbf{P_{org}}$ is clipped against the horizon, for the closed form expression of the polygonal integral is only defined in the upper hemisphere. It is stated as

$$\int_{\mathbf{P'}} \frac{1}{\pi}\cos\theta_{\omega}d\omega = \frac{1}{2\pi}\sum_{i=0}^{n-1}\arccos\left(\mathbf{p_i}\cdot\mathbf{p_j}\right)\left(\frac{\mathbf{p_i}\times\mathbf{p_j}}{\|\mathbf{p_i}\times\mathbf{p_j}\|}\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}\right), \tag{26}$$

where $\mathbf{P'} = \{\mathbf{p_0}, ..., \mathbf{p_{n-1}}\}$ is an arbitrary polygon in the upper hemisphere, $n = |\mathbf{P'}|$ the number of polygon vertices $\mathbf{p_i}$ and $j = i + 1 \mod n$. The integral is suitable for efficient shader implementation and scales only in $\mathcal{O}(n)$ [HDHN16].
To make LTCs suitable for a renderer, Heitz et al. [HDHN16] create a two-dimensional lookup table with 64×64 entries. Because of the isotropy of the BRDF, only one dimension corresponds to the view vectors θ component, and we set the φ component of ω_o to 0. In shading, we can just rotate each pair of ω_o and ω_i so that the φ component of ω_o is 0, because it does not change the value of an isotropic BRDF.
The other dimension corresponds to the surface roughness α of the material. They hence require a preprocess, where $64 \times 64 = 4096$ BRDF configurations are fitted. Alongside the matrix parameters, they also save a normalization value for each table entry to match the BRDFs albedo.
Since the parameters for LTCs are continuous for neighbored fits, the discrete values are interpolated in the shader implementation.
In reality, we do not directly fit for pairs of (ω_o, α) bit instead for pairs of (ω_o, a) with $\alpha = a^2$. Therefore we have an increasing density in our BRDF sampling for increased specularity of the BRDF, since small changes in α yield significant changes in the resulting BRDF for α values close to 0.

We seek approximations of a target function by optimizing our transformation matrix parameters in a non linear manner. For a given matrix, the optimal SH expansion is found with a linear least squares solver. The parameter space for our transformation matrix is large and the non-linearity of the optimization makes it difficult to find the global minimum in-between the local minima.
The next section describes, how good approximations of the GGX BRDF are found.

6 Spherical Harmonic Fitting

To achieve realistic approximations of the GGX BRDF, we need to fit our LTSH expansion as closely as possible for different combinations of view direction ω_o and roughness $a = \sqrt{\alpha}$ values. Those LTSHs for a pair (ω_o, a) then enable shading of a pixel, where (ω_o, a) is constant but ω_i needs to be evaluated over the whole surface area of the polygonal light. For this purpose, we need to find approximations of the BRDF for a sufficient number of (ω_o, a) pairs.
In section 6.1 we explain how to find an optimal SH expansion given a linear transformation M, which matches the target function as close as possible after the linear transformation is applied.
Then, in section 6.2 we use the previously gained insights to find the linear transformation, which when applied to its optimal SH expansion, yields the smallest error compared to the target function.
Using both techniques in combination allows for close approximations of the GGX BRDF, yielding plausible results in real-time rendering.

6.1 Optimizing the SH coefficients

In this section, we assume that we have a transformation given by its 3×3 matrix M, a target function $f_{GGX}(\omega_i, \mathbf{x}, \omega_o)$, which we want to approximate as close as possible and a pair of (ω_o, a) which leave ω_i as our only unfixed variable. The variables $\mathbf{x}$ and ω_o are defined by (ω_o, a) and the dependency on them will be dropped for the sake of brevity. Note, that we directly fit the cosine weighted GGX BRDF to get rid of the cosine later in the integral and thus $f_{GGX}(\omega_i, \mathbf{x}, \omega_o) = f_r(\omega_i, \mathbf{x}, \omega_o) \cos \omega_i$. In our case, f_r is the GGX BRDF introduced in 5.2.
We can project a function on an SH basis of order N with coefficients f_l^m in solving equation 27 for a sufficient set of directions ω.

$$\sum_{l=0}^{N-1} \sum_{m=-l}^{l} f_l^m y_l^m(\omega) \approx f_{GGX}(\omega) \tag{27}$$

Since we do not want to approximate the BRDF directly with an SH function but instead with an LTSH expansion, we need to modify equation 27. We seek coefficients for our original SH function but approximate with our LTSH expansion to harness the advantages of our linear transformation. Using the properties introduced in section 5.4 we thus receive:

$$\sum_{l=0}^{N-1} \sum_{m=-l}^{l} f_l^m y_l^m \left(\frac{M^{-1}\omega}{\|M^{-1}\omega\|} \right) \frac{|M^{-1}|}{\|M^{-1}\omega\|^3} \approx f_{GGX}(\omega) \tag{28}$$

We can restrict our sampled directions to the upper hemisphere since BRDFs are 0 in the lower hemisphere and in practice we clip our polygon to the horizon leaving us with no contribution from the lower hemisphere.
To achieve sufficient sampling of the incident direction ω, we evaluate the upper hemisphere for different value pairs of (θ, φ). We sample $\theta \in [0, \frac{\pi}{2}]$ and $\varphi \in [0, 2\pi]$ uniformly. We take n samples for θ and $4n$ samples for φ leaving us with $4n^2$ sample pairs and $\triangle\theta = \triangle\varphi$. Although the samples are equidistant in the angular domain, they do not map to a uniform sampling on the sphere. For small values of θ (intuitively points near the pole) we have the same number of φ samples as for the equator although the perimeter of the small circle near the pole is much smaller. We thus have to introduce a weighting factor of $\sin\theta$, the Jacobian of spherical coordinates with $r = 1$, to compensate for the oversampling when we move closer to the pole.
We can now formulate equation 28 as a system of linear equations to find our optimal

coefficient vector:

$$A = (a_{j,k}) \quad j \in \{1, ..., 4n^2\}, k \in \{1, ..., (N+1)^2\} \tag{29}$$

$$a_{j,k} = y_k \left(\frac{M^{-1}\omega_j}{\|M^{-1}\omega_j\|} \right) \frac{|M^{-1}|}{\|M^{-1}\omega_j\|^3} \sin\theta_j \tag{30}$$

$$x = \begin{pmatrix} f_0^0 \\ f_1^{-1} \\ \vdots \\ f_{N-1}^{N-1} \end{pmatrix} \tag{31}$$

$$b = \begin{pmatrix} f_{GGX}(\omega_1) \\ f_{GGX}(\omega_2) \\ \vdots \\ f_{GGX}(\omega_{4n^2}) \end{pmatrix} \tag{32}$$

Here y_k corresponds to the SH basis functions ($y_0 = y_0^0, y_1 = y_1^{-1}, ..., y_{(N+1)^2} = y_N^N$), ω_j is one of the $4n^2$ sampled direction pairs (θ, φ) and θ_j is the corresponding θ to ω_j.
In practice, we need to inversely transform each direction ω_j and calculate its Jacobian. Then the SH basis of order N is evaluated at the inverse direction and weighted with $\sin\theta_j$ and the Jacobian. Lastly, to acquire the target vector, we need to evaluate our target function f_{GGX} at every direction ω_j.
As we want to minimize the squared residual $r = \|b - Ax\|^2$ this system can be interpreted as a linear least squares problem. Algorithms to efficiently solve it therefore exist [TB97, p. 77ff]. The dimension of the matrix A is $4n^2 \times N^2$ which is an over determined system of equations (given sufficient direction sampling with $4n^2 \gg N^2$). It yields the N^2 optimal coefficients f_l^m for an SH expansion so that its corresponding LTSH expansion matches the target function f_{GGX} as closely as possible. Note, that we do not need a normalization constant as LTCs do, since scaling of the LTSH expansion is done with the coefficients of the SH expansion directly and thus already included in our least squares problem. The quality of our fitted SH expansion responds directly to the squared residual r and is important for our next step.

6.2 Optimizing Linear Transformations

After solving our inner optimization problem, finding SH coefficients for a linear transformation, we can now take on the challenge to find good linear transformations.
As with LTCs, the planar symmetry of isotropic BRDFs and the scale invariance of LTSH expansion leaves us with a linear transformation that can be represented by four parameters [HDHN16] as

$$M = \begin{pmatrix} a & 0 & b \\ 0 & c & 0 \\ d & 0 & 1 \end{pmatrix}. \tag{33}$$

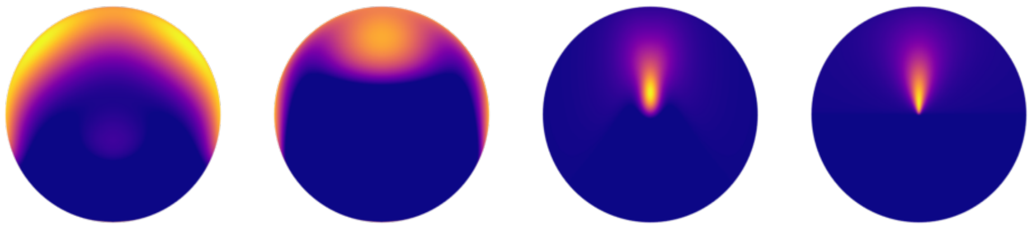


Figure 9: From left to right: SH expansion *(back side),* SH expansion *(front side)*, LTSH and GGX. We used an SH expansion of degree 4 with 25 coefficients to fit the GGX BRDF. $\omega_o = 88,6°$and $\alpha = 0.4$.

Our approach to find such transformations is solving it as a non linear curve fit problem.

The function f_{fit} we want to fit has the four matrix parameters a, b, c, d as inputs and for our sampled incident directions and an outgoing direction ω_j it finds the optimal SH expansion for M as described in section 6.1. Then it evaluates the LTSH expansion at all directions ω_j and compares them to the GGX BRDF evaluation. Our goal is to tweak our parameters in a way to minimize the sum over the squared residual r for all our sampled directions ω. Algorithms for least squares estimation of nonlinear parameters exist [Mar63]. In our Python implementation we used the `curve_fit()` function provided in the `scipy.optimize` package. It internally uses the Levenberg-Marquardt algorithm for unconstrained problems and the closely related trust region reflective algorithm if bounds are provided [sci20] and finds a local minimum four our LTSH expansion. We try to come at least near the global minimum by a good parameter initialization.
We also speed up the optimization by providing a good initial guess for our fit. To do so, we were inspired by the approach of Heitz et al. [HDHN16]. In their fitting code, they brute force the parameters for a first fit. This means, they try a lot of combinations in their parameter space and take the parameters that yield the smallest error. They then use the previously found fits as a guess for neighboring (ω_o, a) pairs.
We divide our fitting by the a value and fit all associated ω_o values for one a. We start in the middle of the ω_o values ($\theta \approx 45°$) and use the LTC transformation (provided by Heitz et al. [HDHN16]) as a first guess, because a brute force search in our extended parameter space is to expensive. We then iterate over the descending $\theta : 45° \to 0°$ and always use the previous parameters as the next guess. After that, we again start in the middle and traverse in the other direction ($\theta : 45° \to 90°$) in the same manner. We do this to avoid singularities which offset our parameters close to the minimal and maximal values of θ. In Figure 9, we show an example of an LTSH expansion fitted to the GGX BRDF.
Another motivation for this approach is to decrease variation of parameters between different fits. Our spherical function domain is vastly increased compared to LTCs which makes our parameters much more prone to discontinuities. For example, a similar change of features of the LTSH expansion can either be achieved by altering the linear transformation or by altering the SH coefficients.
We empirically observe that our approach improves the smoothness of parameters compared to just using the LTC parameters as a guess. However, a significant amount of variation persists in our parameters which we need to address in our implementation in section 8. Other ways of decreasing variation in the parameters could not be assessed in this thesis and are left as future work.
In theory, our LTSH BRDF fits cannot be worse than the LTC BRDF fits, since the cosine function is a subset of the SH expansions with degree $N \geq 1$.
With our found fits we now transition to shading with LTSH.

7 Polygonal Light Shading with LTSH Expansions

We now have our BRDF expressed as an LTSH expansion and address how to use it in shading. We therefore need to solve the polygonal integral and with it the RE (equation 2). We decide to use SH expansions as our basis because recent work [WR18, BXH$^+$18] made integration of SH expansions over polygonal domains possible.
The integration only works for SH expansions and not our LTSH BRDF approximation. We thus have to transform the light polygon from world space to our SH basis domain. The process is described in section 7.1.
An overview over the integration process itself is given in section 7.2.
In section 7.3 we explain how to solve the RE using the SH expansion and the polygon in the SH domain.

7.1 Lighting Space

Before we solve the RE for direct, local lighting of planar polygonal area lights, we transform our world space polygon into the domain of our untransformed SH expansion. Here we need clipping to avoid false contributions to our shading integral.
First of all, we need to rotate our polygon into the tangent space of the shading point, where the surface normal n corresponds to the Cartesian z-axis. A rotation matrix R is obtained by finding two tangent vectors t_1 and t_2.
It is also important, that our view direction ω_o rotated into the tangent space is aligned with t_1 when projected onto the $t_1 t_2$ plane, or in other words that its φ component becomes 0 if expressed relative to the tangent space. As said earlier, because of the isotropy of the BRDF we only fit BRDFs where the φ component of ω_o is 0 and we now need to match this convention for our shading. We thus use the Gram-Schmidt process as shown in equations 34 and 35. It basically projects the view vector ω_o onto the plane defined by the normal n and normalizes this projection vector and thus perfectly fits our needs.
With normal n, tangent t_1 and bitangent t_2 we create the basis rotation matrix R (equation 36) [SM09, p. 126ff].

$$t_1 = \frac{\omega_o - n(\omega_o \cdot n)}{\|\omega_o - n(\omega_o \cdot n)\|} \tag{34}$$

$$R = (t_1, t_2, n)^T \tag{36}$$

$$t_2 = n \times t_1 \tag{35}$$

Here we clip the polygon to the upper hemisphere, introducing at maximum one ad-

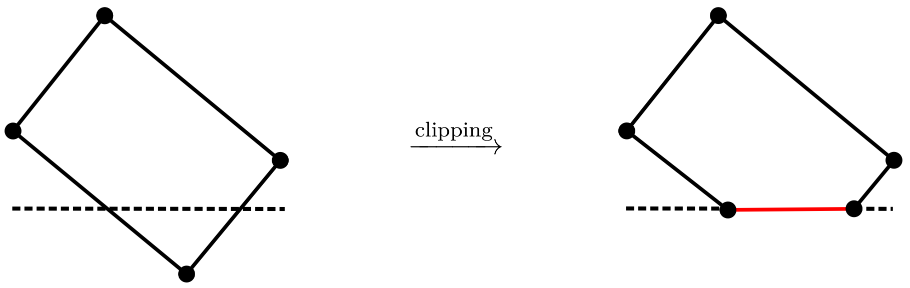


Figure 10: Polygon clipping, introducing at most one additional vertex.

ditional polygon vertex (see figure 10). As explained in section 6.1, we only fit our LTSH expansion to the upper hemisphere of the BRDF. This makes it necessary to clip our polygon since our LTSH expansion might be non-zero in the lower hemisphere. Clipping in tangent space is done straight forwardly, the horizon corresponds to the xy-plane and we eliminate all negative values in the z component. After that, we transform our polygon.

Using the relations introduced in section 5.4, we compute the integral of an LTSH expansion D over a Polygon $\mathbf{P}$ by inversely transforming the polygon vertices $\mathbf{P_{org}} = M^{-1}\mathbf{P}$ and then computing the integral of the original SH expansion D_{org} over the projected polygon $\mathbf{P_{org}}$ as shown in equation 25 (revisit figure 8). Actually, we cannot calculate the integral directly over the inverse linearly transformed polygon but instead over its projection to the unit sphere centered at the origin of the coordinate system. For a point $\mathbf{p_i}$ of the polygon and a shading point $\mathbf{x}$, this projection is easily obtained by $\mathbf{p_i'} = \frac{\mathbf{p_i} - \mathbf{x}}{\|\mathbf{p_i} - \mathbf{x}\|}$.
From now on, we treat $\mathbf{P_{org}}$ as the translated and projected polygon. Note that we are not restricted to the upper hemisphere anymore and that in reality, crucial parts of the lower hemisphere often are transformed to the upper hemisphere to match the BRDF. Thus, by inversely transforming the polygon, it is plausible to have vertices in the lower hemisphere, which contribute correctly to our integral. This is why we clip after the tangent space rotation.
We now combine all previously mentioned parts to solve the RE.

7.2 Integration of Spherical Harmonic Expansions

With our polygon in the domain of our untransformed spherical function, we now solve the spherical integral. The integration of SH expansions over polygons was solved concurrently and in a similar manner [BXH+18, WR18]. Both techniques rely on the zonal harmonic (ZH) factorization introduced by Nowrouzezahrai et al. [NSF12]. For our implementation, we reused a lot of the code provided by Wang and Ramamoorthi [WR18] and thus, we lay out their approach to solve the integral for clarification.
The underlying problem, which they provided a solution for, is stated as

$$L_l^m = \int_{\mathbf{P_{org}}} y_l^m(\omega) d\omega \tag{37}$$

where L_l^m can be understood as the contribution of the SH basis function y_l^m over the projected polygon $\mathbf{P_{org}}$. The following integration is defined for a polygonal domain projected onto the unit sphere. From now on we assume, that $\mathbf{P_{org}}$ is the projected polygon.
A key to the integration is ZH factorization. It is stated as

$$y_l^m(\omega) = \sum_{k=-l}^{l} \alpha_{l,k}^m y_l^0(\omega_{l,k} \cdot \omega) \tag{38}$$

and intuitively means, every SH function of band l is expressed as a weighted sum of rotated ZH basis functions. Here, $\omega_{l,k}$ is the direction of the rotated ZH lobe and $\alpha_{l,k}^m$ is the according unique weight. The index k reaches from $-l$ to l and thus possibly introduces $2l + 1$ summands.
The lobe directions are shared across one SH band, meaning that for example the basis functions of band l, namely y_l^m with m ranging from $-l$ to l, are all reconstructed by the same lobe directions $\omega_{l,k}$. This is why the lobe direction $\omega_{l,k}$ has no dependency on m. Furthermore, the lobe directions are optimized to be shared across bands. This leaves us with only $2N + 1$ unique lobe directions for a ZH factorization of an SH expansion of degree N.
We do not optimize solely for shared lobe directions, but also seek a high sparsity in the unique weights $\alpha_{l,k}^m$ of our ZH factorization. Nowrouzezahrai et al. [NSF12] explore the candidates for lobe directions with sparse weights and provide pre-calculated directions and weights for ZH factorizations of SH expansions up to order $N \leq 8$.
We use these lobe directions and weights to compute an efficient ZH factorization of our SH expansion. Next, we use this factorization to compute the polygonal integral for a ZH factorization from which we then reconstruct our SH integral. We thus reformulate

equation 37 for ZH functions rotated to the central axis ω_r:

$$L_l = \int_{\mathbf{P_{org}}} y_l^0(\omega_r \cdot \omega) d\omega \tag{39}$$

And using equation 38, we get:

$$L_l^m = \sum_{k=-l}^{l} \alpha_{l,k}^m L_l \tag{40}$$

We solve our integral now by finding a solution to the polygonal integral of the rotated associated Legendre polynomials. For an in-depth explanation of how this is done, see the work of Wang and Ramamoorthi [WR18].
Because of the shared lobe directions, we iteratively compute the ZH integral for as many SH bands as we need, making the resulting code quickly adaptable for SH integration of various degrees.
The so found integral of our ZH factorization is then rotated with pre-calculated lobe weights into our SH basis. We now have the influence of each SH basis function L_l^m in the polygonal domain. Calculating the integral of our specific SH expansion reduces to a simple dot product of our transposed SH coefficient vector C^T and our integral coefficient vector L:

$$C^T \cdot L = \left[c_0^0, c_1^{-1}, ..., c_N^N\right] \cdot \begin{bmatrix} L_0^0 \\ L_1^{-1} \\ \vdots \\ L_N^N \end{bmatrix} \tag{41}$$

Wang and Ramamoorthi [WR18] provide an exemplary implementation for SH expansions up to degree 8, that we use.
As mentioned, the provided code from the authors [WR18] is easily truncated to compute lower order SH integrals. Here it is noteworthy, that the lobe directions and shared lobes are optimized for the integration of SH expansions of degree 8. Although the result remains exact for lower degrees, the weighted rotation of the ZH integral is not as sparse as it could be. Recalculating those weights for the optimal lobe directions provided by Nowrouzezahrai et al. [NSF12] is however not a trivial task and is not addressed in this thesis. We leave this optimization open for future work.
It is now time to get our light polygon from world space to a spherical projection in the space of our SH expansion.

7.3 Solving the Rendering Equation

We are now able to express our BRDF as an LTSH expansion, inversely linearly transform our world space polygon and calculate the integral of an SH expansion over a polygonal domain. With these tools at hand, we find a solution to the RE.
As a reminder, the RE for area lights is stated as:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\mathbf{y} \in A} f_r(\omega_i, \mathbf{x}, \omega_o) L_i(\mathbf{x}, \omega_i) \frac{\cos\theta_i \cos\theta_j}{\|\mathbf{y} - \mathbf{x}\|^2} d\mathbf{y} \tag{42}$$

We can now express it as:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\mathbf{P_{org}}} D_{org}(\omega_{org}) L_i(\mathbf{x}, \omega_{org}) d\omega_{org} \tag{43}$$

Here, $\mathbf{P_{org}}$ denotes the inversely transformed, translated, projected and clipped polygon of our light source, D_{org} is the SH expansion to our LTSH BRDF fit and ω_{org} is the inversely linearly transformed incoming light direction. Here, the incident light $L_i(\mathbf{x}, \omega_{org})$ is constant, since we only handle spatially constant polygonal light emitters. We can pull it outside of our integral as a constant light intensity factor of our light source L_i:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + L_i \int_{\mathbf{P_{org}}} D_{org}(\omega_{org}) d\omega_{org} \tag{44}$$

Also note, how our dependencies to $\frac{\cos\theta_i \cos\theta_j}{\|\mathbf{y}-\mathbf{x}\|^2}$ dropped. The $\cos\theta_i$ term is already included in our fitting process and thus contained in D_{org}. The $\frac{\cos\theta_j}{\|\mathbf{y}-\mathbf{x}\|^2}$ term is handled in our projection of the polygon to the unit sphere, since we transition from an area measure to a solid angle measure. Intuitively, this term means, that light from far away or from a polygon facing away from the shading point is less intense. The same is achieved by our projection, since a polygon far away or facing in another direction has a smaller projection. We now get rid of the integral:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + L_i(C^T \cdot L) \tag{45}$$

with the SH coefficients of our LTSH BRDF fit C and our integral coefficient vector L. We thus achieved a feasible solution for the RE with spatially constant emitting, planar, polygonal area lights. The only approximation in our process remains the fitting of our LTSH expansion to the BRDF and is the only limiting factor to our quality.
The quality can be increased by increasing the degree of our SH expansion and we thus have derived an adaptable technique for direct area light shading. However, our technique runs in $\mathcal{O}(n^2v)$ where n is the degree of the SH expansion and v is the number of polygon vertices. Thus, increasing the degree of the SH expansion significantly slows down our technique.
On the other hand, improvements in graphics hardware constantly make room for more and more compute intensive tasks and increasing the SH degree instead of entirely switching to novel techniques is a quick and easy alternative for production use, once LTSH is integrated.
It is now time to compare the results of our technique to related work.

8 Results

We are now interested in the quality of our technique in comparison to LTC [HDHN16] and a ground truth implementation. Although we introduced several real-time techniques for area light shading in section 4, we will only compare our technique to LTC. This is due to the close relation of those techniques and the impact LTC had in the graphics community.
First of all, we give an overview of our implementation. This includes both fitting and rendering implementations, our memory consumption in shading and what problems arose during the implementation (section 8.1).
Then, we compare the quality of our technique. Here again we divide into LTSH fitting and LTSH shading and compare results both visually and numerically (section 8.2).
Lastly, we elaborate on run times of fitting and shading and compare shading run times of different SH degrees and related work, before we give ideas on how to further improve our technique.

8.1 Implementation

We now sketch our implementation to give an intuition of how our results are achieved. It is noteworthy, that we do not optimize every single line of our pre-process and shading applications but rather give an idea of what our technique is capable of. The following results should therefore be taken as estimates rather than fine tuned peak performances of our technique.
Our fitting is implemented in Python. As described in section 6, we take samples equidistantly in the angular domain all over the upper hemisphere and formulate a curve fit problem with them to find suitable parameters for our linear transformation.
For a good coverage of our pairs of ω_o and a values, we fitted the GGX BRDF for 64 samples of either parameter, resulting in a grid of $64 \times 64 = 4096$ different fitted functions. To avoid singularities in our fitting process, we introduce epsilon values to our ω_o and a. This prevents angles of $0°$ and $90°$ in ω_o and roughness values of 0 in $\alpha = a^2$.
For LTCs, we solve the curve fitting without constraints on the matrix parameters. For our LTSH fits however, we encountered problems with extreme values for the matrix parameters and the SH coefficients. The resulting fits yield plausible results in our Python evaluation and the error compared to the GGX BRDF is acceptable, but in shading they yield obvious artifacts. We hold the truncation of double precision to half precision floats between Python and shader code responsible for these artifacts. To avoid these numerical issues, we introduce bounds to our transformation matrix parameters. We observed, that fixing the range of possible transformation parameters to $[-10000, 10000]$ solved the numerical issues and still left the parameter space large enough to encounter good transformations.
In contrast to Heitz et al. [HDHN16], we do not minimize the L^3 error for our LTC fitting, but instead seek to minimize the L^2 error, as we view our problem as a least squares problem. Also, we do not compute the norm of our LTC fit after we found a good transformation and then make it match the norm of the GGX BRDF through applying a normalization constant. We instead seek this constant instantly while finding our linear transformation as a linear least squares problem similarly to the finding of our SH coefficients in LTSH fitting. This yields fairly similar results compared to the normalization and matches the existing program structure from our LTSH fitting.

For our rendering, we take the *Falcor* framework [BYC$^+$19] provided by NVIDIA as our basis and use the DirectX 12 build. We implement our technique as a deferred renderer and only change the shading in the lighting pass. To test our lighting in a complex scene, we use the Sun Temple scene [Gam17] provided by NVIDIA's ORCA library.

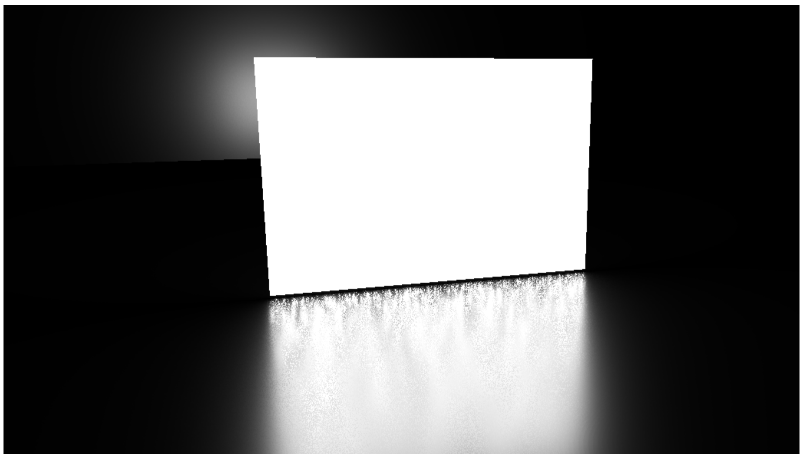


Figure 11: Ground truth implementation: area light near a specular surface, artifacts visible even with 4096 samples per pixel (spp).

Our ground truth implementation is a rather simple one. For each shading point, we take a number of light samples distributed uniformly in the area of the light source (refer to Monte-Carlo sampling explained in section 5.1). We compute a fixed set of samples once for our light source and then randomly take a subset of these samples for each shading point. This introduces noise but yields good results for most configurations, given we take a sufficient number of samples. However, if a light source is close to scene geometry or if we have highly specular surfaces, the single light samples become apparent and our ground truth looks like many point lights close together (figure 11).
The LTC shading is imported from the code provided by Heitz et al. [HDHN16] and we mainly ported it to be used in an HLSL/SLANG environment but did not change its logic. Similarly to Heitz et al. [HDHN16], we write our coefficients and matrix parameters to textures and sample them for our shading. This inherently adds interpolation between different fits for our shading and works well, because the LTC parameters are continuous, as discussed earlier.
Since we only need the inverse transformation matrix for our shading, we invert our fitted matrices as a part of the preprocess. Inverting a matrix of form

$$\begin{pmatrix} a & 0 & b \\ 0 & c & 0 \\ d & 0 & 1 \end{pmatrix} \tag{46}$$

possibly yields a matrix of form

$$\begin{pmatrix} e & 0 & f \\ 0 & g & 0 \\ h & 0 & i \end{pmatrix} \tag{47}$$

and thus, we have 5 parameters. But we also normalize each vector after its transformation, rendering the scale of the transformation irrelevant. If we divide the matrix by its first parameter e, we receive a matrix with four parameters and can write it to a single RGBA texture in our shader. We do this for both LTC and LTSH.
Using 16 bit floats for each parameter, we need 64-bits for each of the 4096 matrices. Additionally, we need a 16-bit normalization constant for LTC for each of the 4096 fits, resulting in an overall memory consumption of 40,960 bytes in our shader.
For LTSH, we need to save $(N+1)^2$ coefficients of our SH expansion additionally to our matrices. Here, N denotes the degree of our SH expansion. We use multiple RGBA textures, which increases the actual number of coefficients in textures to the next multiple of 4. For an SH expansion of degree 2 we thus have 12 16-bit coefficients per fit and for degree 4

we get 28 coefficients per fit. The overall memory consumption for our LTSH shading with degree 2 and degree 4 thus is 131,072 bytes and 262,144 bytes, respectively. This memory consumption is insignificant to modern GPUs and since our data does not change, it only needs to be set once and does not consume cache or main memory bandwidth.
For the LTSH shading, we port the code provided by Wang and Ramamoorthi [WR18] from GLSL to HLSL. With this integration code we compute the shading integral and then multiply in the lights intensity as described in equation 45.
Because of the high variation in our LTSH parameters between different fits, we cannot use texture interpolation for our LTSH shading. We therefore use dithering to smooth out our shading result. For dithering, we draw two random numbers for our shading point and then use them to decide, which of the two discrete values for ω_o or a should be used, considering where in between the discrete values our continuous value lies. This introduces noise to our image, but it hides the otherwise obvious transitions from one discrete value to another.

While implementing our LTSH shading, we encountered several issues. After importing the code base from Wang and Ramamoorthi [WR18], we received shading results, that did not resemble the shading of our scene in any way. We did not see any obvious mistake or difference in our code and debugging shader code is cumbersome. To improve debugging capabilities, we decided to port our shader code to Python and debug it there, which caused a delay in the implementation of our technique.
In Python, we could quickly rule out the ZH factorization, as evaluations of ZH decompositions matched direct SH evaluations. We then implemented a Monte-Carlo sampled SH integration, to get a reference for our integral. This enabled us to rule out different parts of the shader code step by step. After a series of different tests, we were able to identify, that the normalization constant for the ZH base functions (revisit equation 21) were missing in the shader code. We suspect, that they were already included in the pre-calculated SH coefficients by Wang and Ramamoorthi [WR18]. However, this fact was not obvious to us and took some time to identify. After that, only differences in signs were left and could be fixed quickly.

8.2 Quality Comparison

In this section, we give an understanding of how closely we are able to approximate the GGX BRDF using LTSH. To do so, we compare the calculated error of LTC and LTSH of degree 2 and 4. We also compare the fits visually to the GGX BRDF and inspect images rendered with the different techniques.
Our error E of our function f_{fit} to the target function f_{GGX} was calculated as the square root of the mean squared error for N samples:

$$E = \sqrt{\sum_{i=1}^{N} \frac{(f_{GGX}(x_i) - f_{fit}(x_i))^2 \sin\theta}{N}} \tag{48}$$

Again, we do not sample uniformly in the hemisphere but equidistantly in the angle domain and thus weight our error samples by $\sin\theta$. This was the error our optimizer tried to minimize during the fitting process and we thus use it to assess our error.
We show the error for different pairs of view direction ω_o (only its θ component is of interest for us due to isotropy) and alpha $a = \sqrt{\alpha}$ (note the non-linear connection to the roughness) in tables 1, 2, 3. Figure 12 visualizes the error tables. Note, that we only show 8×8 values in our tables for brevity, but that the corresponding figure visualizes the whole resolution of 64×64 for our (ω_o, a) pairs. We also cap the maximum of the color-map to a value significantly lower than the true maximum value. We do this to better visualize the overall tendencies of our fits instead of some maximum values. The magnitude of those values is shown in the tables and thus not needed in the figures.

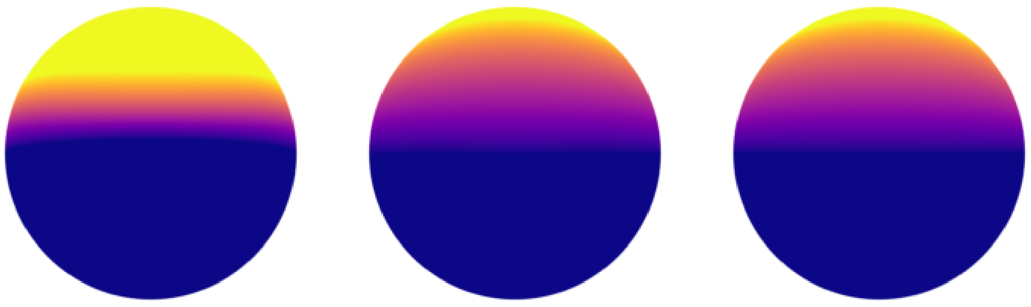


Figure 13: “Backside” (opposite of the highlight) of LTC *(left)*, LTSH of degree 4 *(middle)*, and GGX *(right)*. Color map is clamped to $[0, 0.05]$. $\theta = 35.7°$ and $\alpha = 0.630$.

The majority of values shows our predicted quality improvement from LTC to LTSH and in between LTSH with increasing degree. However, some irregularities strike the eye. We get unusually high error values for some combinations with low roughness and θ values close to either $0°$ or $90°$. Also, those values do not correspond to the trend of quality improvement between the techniques. We thus classify them as outliers that arise from insufficient sampling in our BRDF fitting. For highly specular BRDFs, the reflection peak becomes very small and is missed easily, if no importance sampling is used.
Note, that for the error calculation we evaluate our hemisphere with a higher resolution (512 steps compared to 128 steps for θ) than we do in our fitting. The error in the same resolution as our fit was significantly lower. This explains, why our optimizer did not further improve the error and leads us to the conclusion, that important parts of the BRDF are missed in the fitting process of highly specular BRDFs. A fitting resolution of 512 steps for θ however was not feasible with our fitting technique.
These broken fits, where the BRDF highlight is not covered by our sampling, happen with really small roughness values ($\alpha < 0.01$) and do not appear in most of our rendering configurations. They fall into the field of improving the quality of our BRDF fits and hence are left to future work.
Another tendency we can read from the tables is, that we achieve significant improvements even with an LTSH expansion of degree 2, especially in highly rough surfaces. To visualize the problem, we compare LTC and our LTSH to the GGX BRDF and cap the color map to $[0, 0.05]$ (figure 13). This enables us to see differences in values close to zero. Here we see, that LTC differs significantly from the GGX BRDF, while our LTSH fit yields a closer approximation. The inflexible cosine basis function is not able to match the heterogeneous properties of the GGX BRDF over the whole hemisphere. For light sources with a high intensity, this difference becomes apparent. The same effect appears in renderings (figure 14).
We also see, that our error increases for flat or grazing angles. Here, the shape of the GGX function becomes more and more complex and hence difficult to reproduce with linearly transformed functions.

Table 1: Error of LTC for different pairs of (θ, a).

$\theta \setminus a$	0.063	0.190	0.317	0.444	0.571	0.698	0.825	0.952
5.7°	1.83E-5	8.96E+3	6.62E+0	8.05E-3	1.57E+1	4.78E-3	4.50E-2	1.53E-2
17.1°	5.26E+0	1.72E-4	3.60E-3	7.84E-3	1.40E-2	2.49E-2	4.61E-2	6.42E-2
28.6°	9.10E+0	1.01E-2	9.28E-4	6.31E-3	1.33E-2	2.37E-2	4.00E-2	5.35E-2
40.0°	4.00E-1	4.30E-2	1.22E-2	3.05E-3	9.54E-3	1.74E-2	2.61E-2	3.38E-2
51.4°	6.42E+0	1.24E-1	4.22E-2	1.60E-2	5.46E-3	9.62E-3	1.69E-2	2.50E-2
62.9°	8.06E+0	3.20E-1	1.10E-1	4.36E-2	1.63E-2	5.90E-3	1.06E-2	2.17E-2
74.3°	6.54E+0	8.56E-1	2.71E-1	9.48E-2	3.78E-2	1.66E-2	5.63E-3	1.51E-2
85.7°	2.83E+0	3.08E+0	6.43E-1	1.94E-1	7.76E-2	3.95E-2	2.02E-2	8.40E-3

Table 2: Error of LTSH of degree 2 for different pairs of (θ, a).

$\theta \setminus a$	0.063	0.190	0.317	0.444	0.571	0.698	0.825	0.952
5.7°	6.03E-5	6.36E-5	1.73E-4	3.63E-4	4.34E-4	4.52E-4	3.53E-4	3.22E-4
17.1°	2.01E-3	5.93E-4	3.05E-4	5.44E-4	5.55E-4	3.92E-4	2.72E-4	3.07E-4
28.6°	1.05E-2	9.60E-4	9.61E-4	1.03E-3	7.51E-4	4.30E-4	2.63E-4	3.21E-4
40.0°	9.86E-3	2.20E-3	2.54E-3	2.03E-3	1.32E-3	7.27E-4	3.63E-4	3.73E-4
51.4°	3.57E-2	4.59E-3	4.85E-3	4.08E-3	2.84E-3	1.68E-3	7.55E-4	5.02E-4
62.9°	7.56E-1	1.06E-2	1.07E-2	8.47E-3	6.08E-3	3.69E-3	1.66E-3	8.11E-4
74.3°	7.72E+0	3.54E-2	2.77E-2	1.92E-2	1.31E-2	7.85E-3	3.65E-3	1.61E-3
85.7°	3.38E+0	1.61E+0	1.36E-1	6.09E-2	3.25E-2	1.86E-2	9.06E-3	4.16E-3

Table 3: Error of LTSH of degree 4 for different pairs of (θ, a).

$\theta \setminus a$	0.063	0.190	0.317	0.444	0.571	0.698	0.825	0.952
5.7°	6.74E-4	5.93E-5	1.16E-4	3.32E-4	2.01E-4	3.36E-5	9.21E-6	1.25E-5
17.1°	2.74E+6	6.22E-5	1.61E-4	3.33E-4	1.77E-4	3.36E-5	1.05E-5	1.24E-5
28.6°	2.61E-4	9.43E-5	2.20E-4	4.15E-4	2.11E-4	8.41E-5	3.69E-5	1.69E-5
40.0°	7.41E-4	1.65E-4	3.81E-4	5.95E-4	4.05E-4	2.57E-4	1.11E-4	3.39E-5
51.4°	6.54E-3	3.63E-4	8.13E-4	1.04E-3	8.64E-4	5.85E-4	2.59E-4	8.18E-5
62.9°	2.27E+5	1.13E-3	2.20E-3	2.55E-3	2.02E-3	1.13E-3	4.83E-4	1.78E-4
74.3°	9.74E-1	2.52E-2	7.94E-3	7.76E-3	5.12E-3	2.64E-3	1.11E-3	4.67E-4
85.7°	3.39E+0	5.04E+1	7.38E-2	3.08E-2	1.65E-2	8.15E-3	3.55E-3	1.83E-3

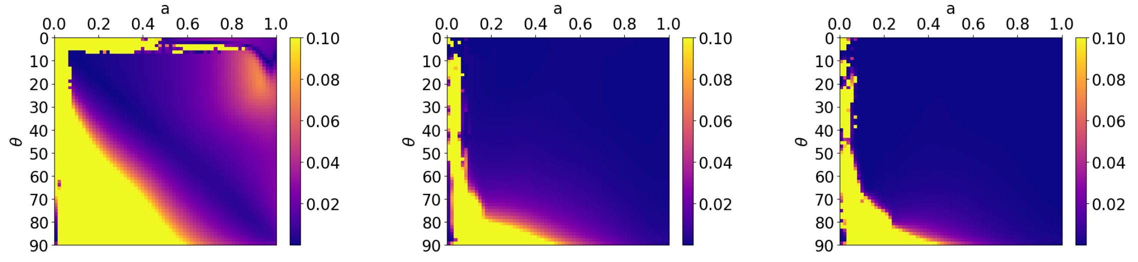


Figure 12: Visualization of the error of different GGX BRDF fits for LTC (left), LTSH of degree 2 (middle) and LTSH of degree 4 (right).

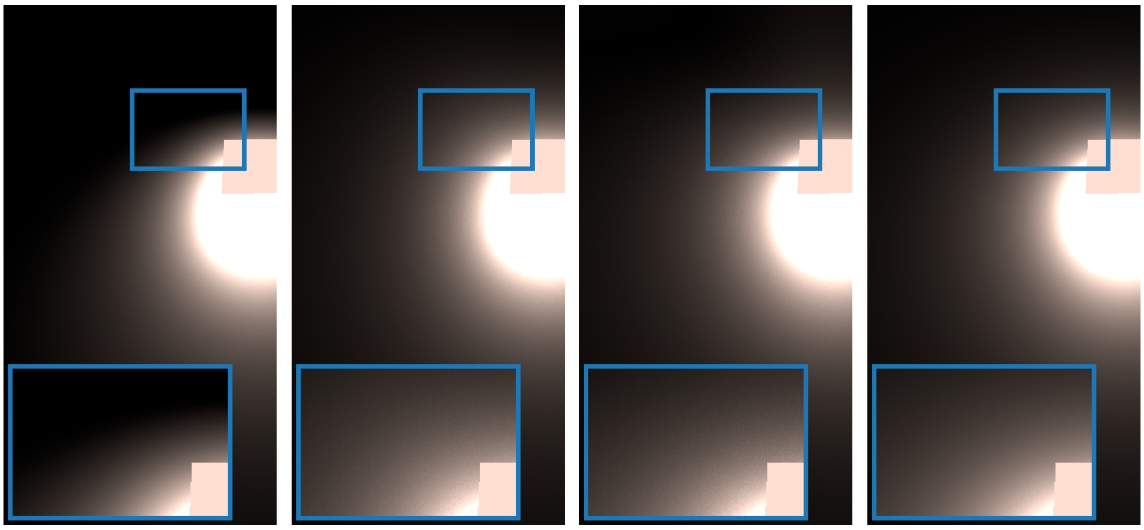


Figure 14: High intensity area light over surface with roughness $\alpha = 0.3$. Comparison of LTC, LTSH of degree 2, LTSH of degree 4 and ground truth *(left to right)*. Our technique shows significant improvement over LTC, even with a degree of 2.

We now compare the BRDF fits visualized on the unit sphere, to give a better understanding of where and why the techniques differ from the target function.
Figure 15 compares LTC and LTSH of degree 2 and 4 to the GGX BRDF. The figure shows the approximated BRDF fits and the GGX BRDF that is used as the fitting objective. The surface roughness $\alpha = 0.242$ is neither highly rough nor highly specular. These settings are already handled convincingly by LTC and in the upper two rows not much difference is apparent. At grazing angle however, visible improvements happen between the fits. Here, the improved versatility of the basis functions accounts for a better fit to the GGX BRDF. Also note, that we view the unit sphere from a $45°$ angle.
Figure 16 compares the BRDF fits for a rougher GGX BRDF. The quality improvements between the techniques become clearly visible. Even though our LTSH technique still contains visible differences to the GGX BRDF at grazing angles, the improvements compared to the LTC approximation are significant. The line of the horizon is defined much clearer for our LTSH approximations and especially the LTSH fit of degree 4 appears less rounded than the other two.
The error on the backside of our BRDF fit becomes visible in figure 17. The big artifact on the lower hemisphere is to be ignored, since it does not contribute to our polygonal integral due to clipping. We magnified our color-map, to make the error visible and compared to the magnitude of the specular highlight, the error is small. It still results in visible artifacts in rendering (blue rectangle in figure 18). This kind of error occurs in both LTC and our technique especially at grazing angles.

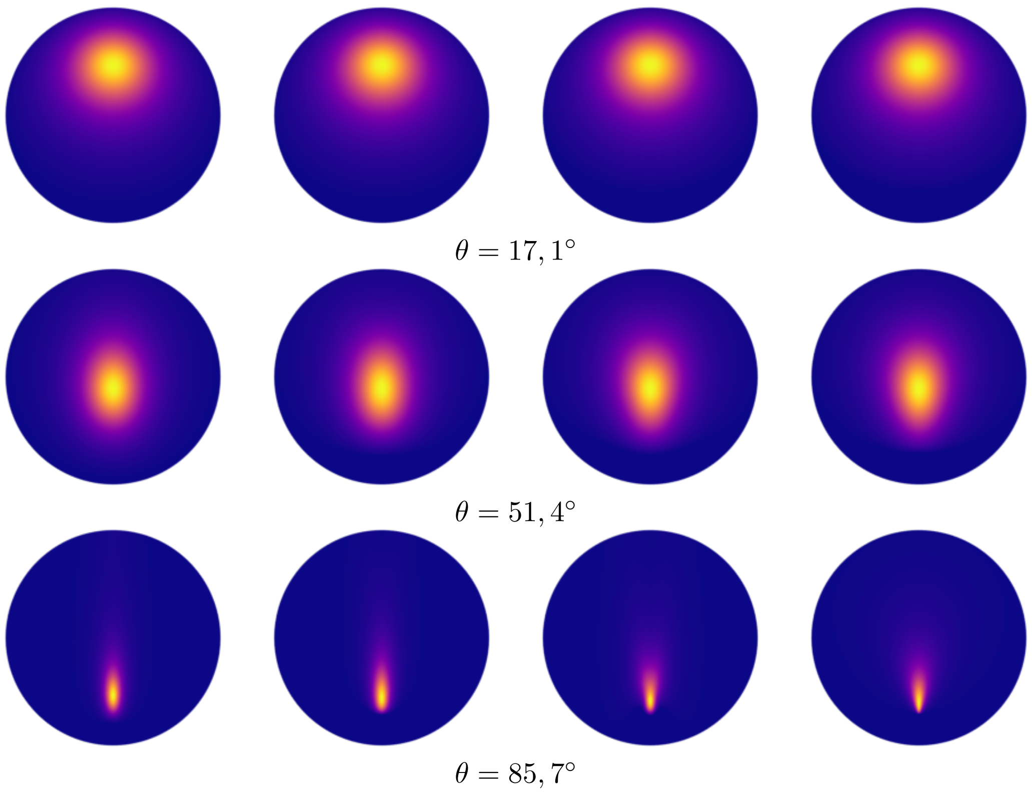


Figure 15: LTC *(left)*, LTSH of degree 2 *(middle-left)*, LTSH of degree 4 *(middle-right)* and GGX *(right)*. $\alpha = 0.242$.

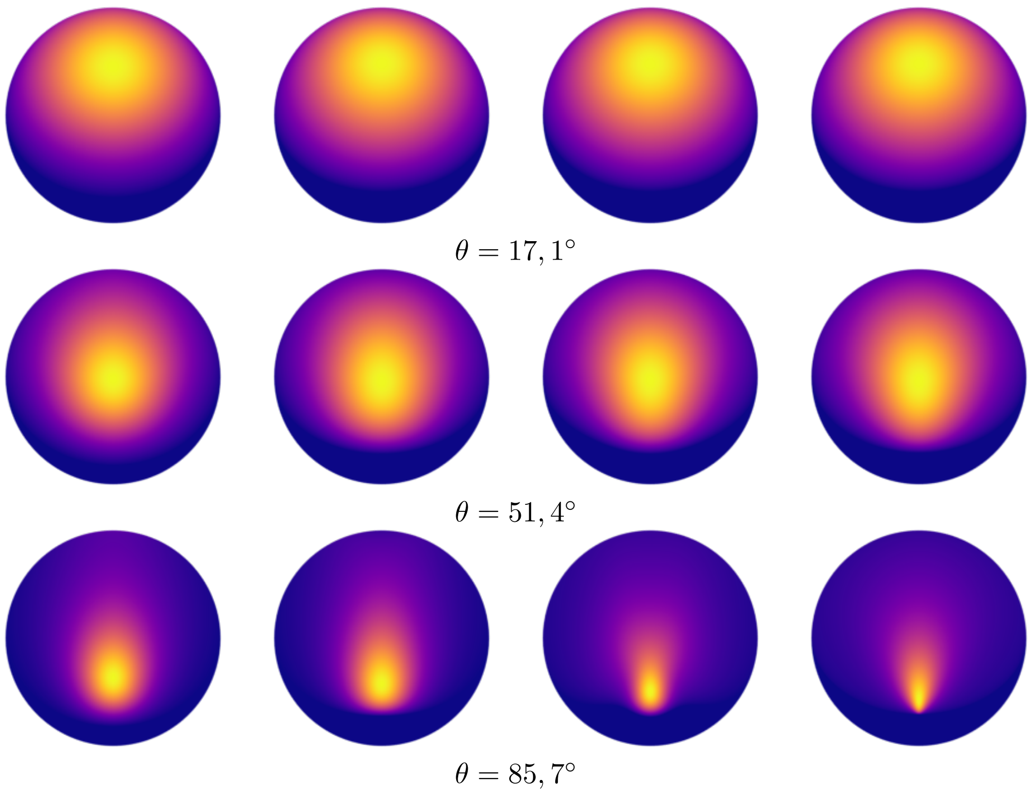


Figure 16: LTC *(left)*, LTSH of degree 2 *(middle-left)*, LTSH of degree 4 *(middle-right)* and GGX *(right)*. $\alpha = 0.510$.

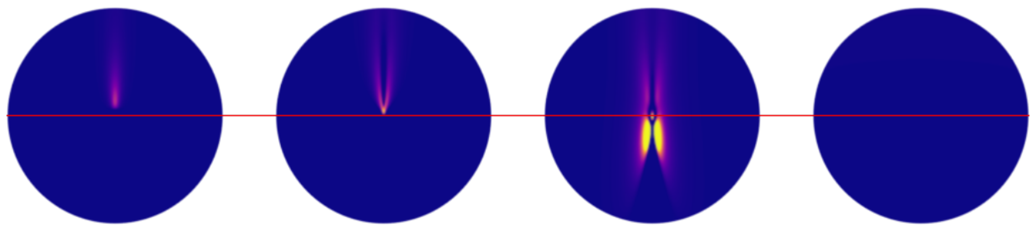
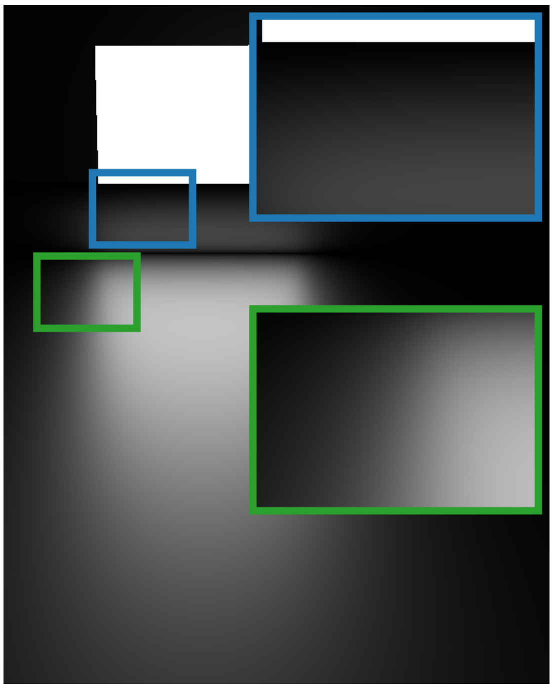
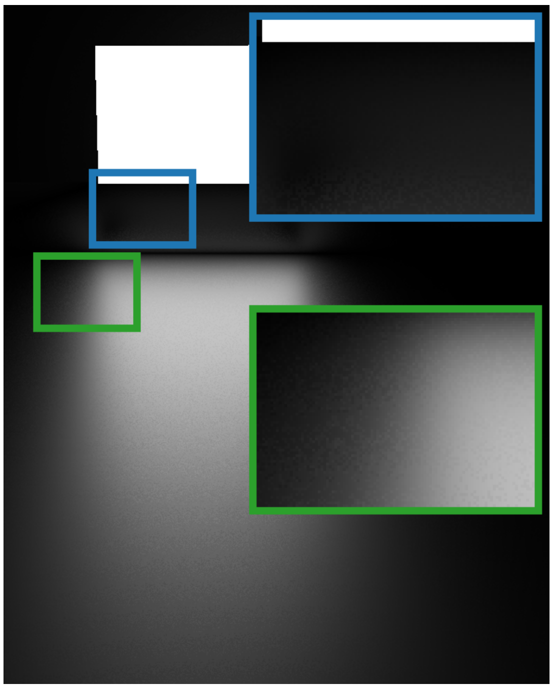
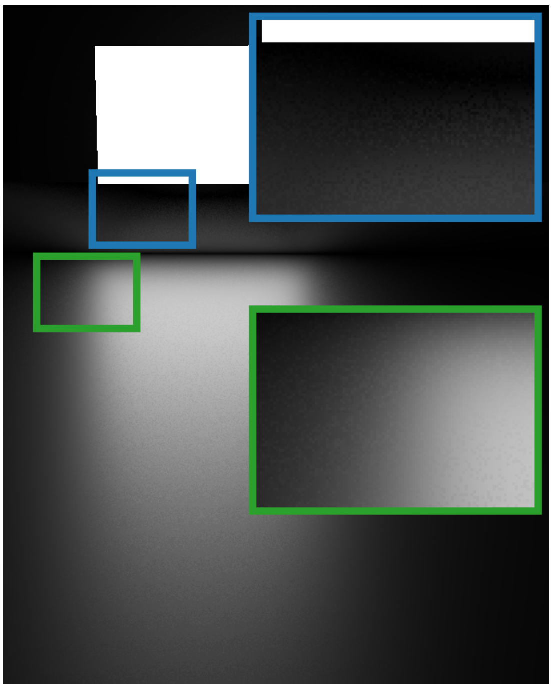


Figure 17: "Backside" of LTC *(left)*, LTSH of degree 2 *(middle-left)*, LTSH of degree 4 *(middle-right)* and GGX *(right)*. $\theta = 85.7°$ and $\alpha = 0.1$. The red line is the horizon.


(a) LTC


(b) LTSH of degree 2


(c) LTSH of degree 4

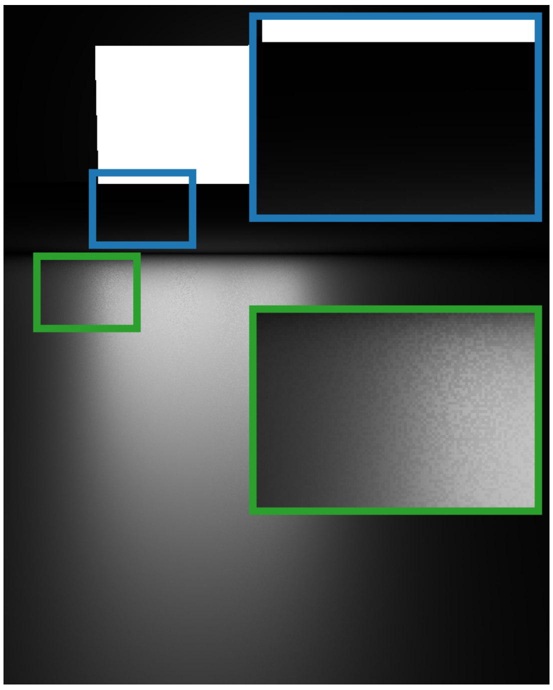

(d) Ground truth

Figure 18: Area light over a flat plane.

After giving an impression of what to expect from our technique, we now focus on actual renderings. Figure 18 shows a rectangular area light over a flat plane. The shape of the highlight grows closer to the ground truth from LTC to LTSH with degree 4. On the other hand, our technique introduces ringing in the reflection facing away from the camera. The noise from dithering and sampling in LTSH and the ground truth becomes apparent. Also, the artifacts for surfaces behind the area light become visible.
Our fits for view directions ω_o with a θ value close to zero are unstable because we have to weight samples with $\sin\theta$ which also grows close to zero. Since we use the same weighting for our error calculation, the error is not apparent in our error tables. This issue results in a small black dot as shown in figure 19. It occurs in both LTC and LTSH and can likely be fixed by importance sampling.
While experimenting with the light parameters, we observed numerical instabilities (figure 20), if the quotient $\frac{intensity}{area}$ grows too large (very intense light and/or a small area light). However, for normal sized area lights the intensities needed to reach numerical instability are high and unlikely to be reached in a real world setting.
For tiny area lights, point lights still remain a good approximation and LTSH shading is not needed. Implementing line lights with thin and stretched out polygonal area lights however is not recommended and will most likely result in numerically unstable shading. Here, a direct solution for line lights and LTSH should be sought as discussed in section 4.
A visible improvement of LTSH over LTC occurs with lights placed near rough surfaces. The LTC approximation is 0 where the GGX BRDF still reflects a minor portion of light. The difference is noticeable (figure 14). This kind of configuration happens a lot in reality, e.g. for an area light mounted to a wall. Even a LTSH approximation of degree 2 significantly improves quality here.
In figure 21 we shade a wall gizmo of the sun temple scene [Gam17]. The overall exposure increases from left to right and resembles the ground truth more closely. However, some parts of the image become overexposed with our technique, shown in the red magnification. Overall, the differences for such detailed geometry are rather subtle and only apparent in direct comparison.
We now look at specular renderings of a complex scene with normal mapped surfaces in figure 22. Here we see, that most of the discussed artifacts are less noticeable with complex geometry and normal maps.
Generally, our technique is able to produce appealing and realistic images with complex scenes. We present this in figure 23, where we use both diffuse and specular rendering in the sun temple scene.

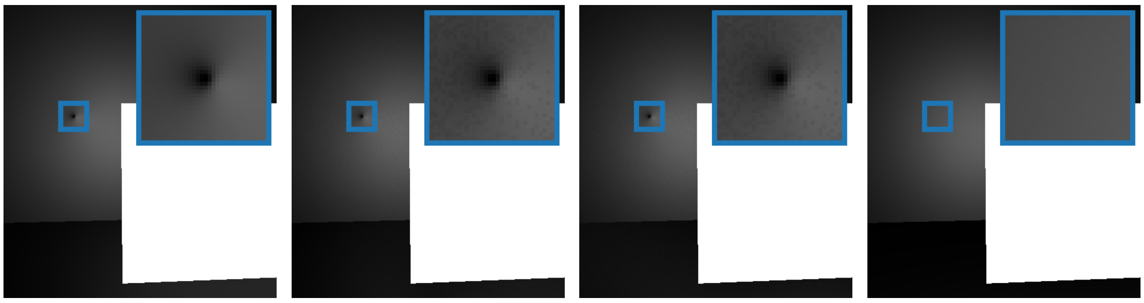


Figure 19: Artifact due to our weighted equidistant sampling. Shading done with LTC, LTSH of degree 2, LTSH of degree 4 and ground truth *(left to right).*

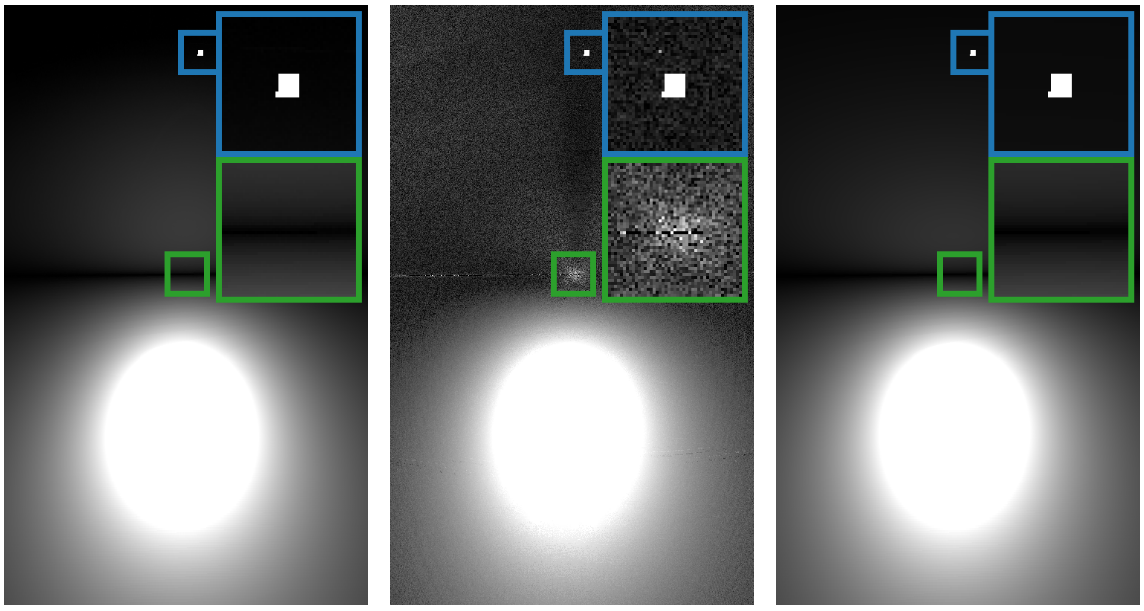


Figure 20: Area light with high intensity and small area using LTC, LTSH of degree 4 and ground truth implementation *(left to right)*. Our technique shows strong numerical issues.

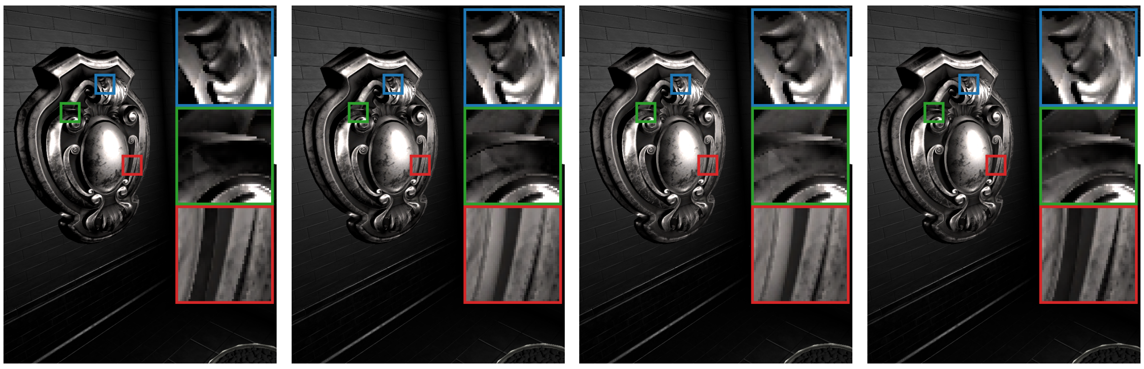


Figure 21: Specular rendering of a wall gizmo in the sun temple scene. Comparison of LTC, LTSH of degree 2, LTSH of degree 4 and ground truth *(left to right)*.

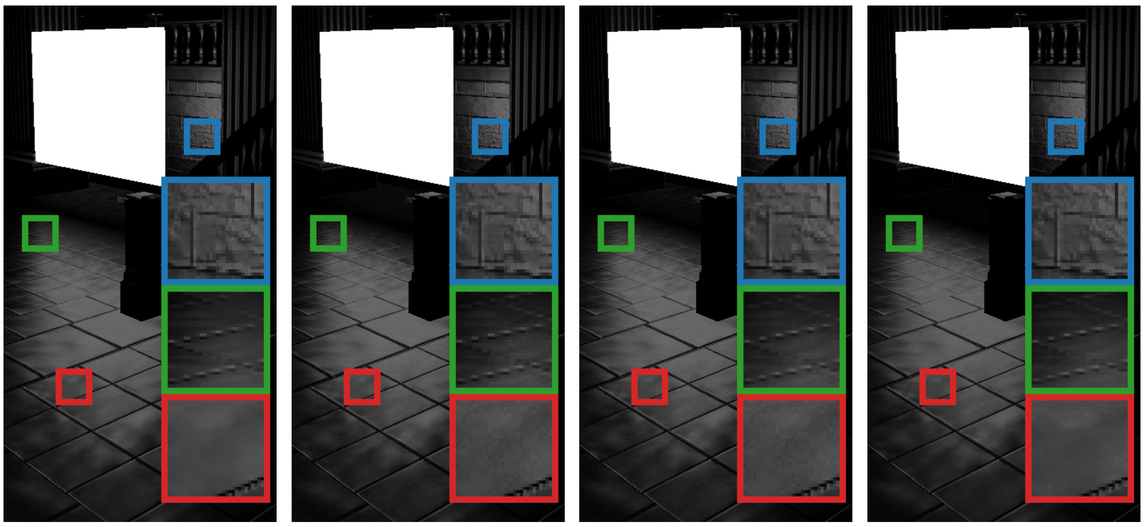


Figure 22: Specular rendering of normal mapped surfaces in the sun temple scene. Comparison of LTC, LTSH of degree 2, LTSH of degree 4 and ground truth *(left to right)*.

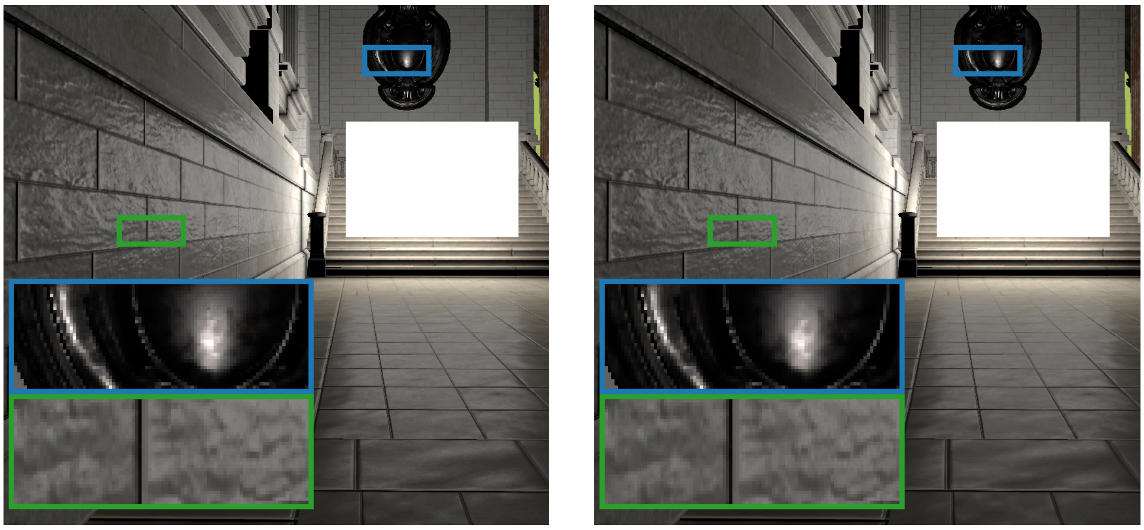


Figure 23: Rendering (specular and diffuse) of the sun temple scene. LTSH of degree 4 *(left)* and ground truth *(right)*.

8.3 Run Times

We now lay out, how long it takes us to compute the results presented above. First we will elaborate on the fitting process and then compare LTC and LTSH in a shading application.
All our BRDF fitting was done on an *AMD Ryzen 7 2700X* 16 core processor with
3.7 GHz per core and 64 GB of RAM. We use Python and equidistant sampling in the angular domain, so significant speedup is most likely possible by a C++ implementation and importance sampling. It was however out of scope for this thesis and is left to future work.
That being said, for 64×64 BRDF fits and 128×512 hemisphere samples, our LTC fitting took 1 hour and 10 minutes. LTSH of degree 2 took around 2 hours and LTSH of degree 4 around 7 hours for the same configuration.
For rendering our images, we use the same CPU and RAM as above in combination with an *NVIDIA GeForce RTX 2070 SUPER* with 8 GB of VRAM in our DirectX 12 implementation using *Falcor* [BYC+19] (see section 8.1).
The profiling is done with a resolution of 1920×1137 pixels. We only profile the lighting pass of our deferred shader, since our technique only impacts this shaders performance. We thus have no dependence on the geometry of the scene we rendered, as long as no pixel is discarded.
Running our lighting pass without doing actual lighting calculations and only fetching our g-buffer takes 0.14 ms. The following timings are the actual time our lighting pass took for each technique subtracted by these 0.14 ms.
For LTC we have 0.30 ms per frame, LTSH of degree 2 have 0.81 ms and LTSH of degree 4 have 1.64 ms. We used one area light with 4 vertices for every technique.
In table 4 we show the run times and plot them against the 0.14 ms baseline in figure 24.
We could not asses the impact of the number of the light polygons vertices to our shading performance, but our LTSH code is based on the technique of Wang and Ramamoorthi, which scales linearly in the number of polygon edges [WR18]. This is similar to the performance scaling of LTC [HDHN16].
Moving LTSH from degree 2 to degree 4 (which means we use 5 SH bands instead of 3) roughly doubles the time needed in our lighting pass. We do not asses further degrees for LTSH in our work, but again, Wang and Ramamoorthi empirically showed linear scaling in the number of SH coefficients for the zonal harmonic integral and even worse than linear scaling for the whole SH integral [WR18]. This is because the rotation becomes less sparse for higher SH orders. Since the number of coefficients scales quadratically in relation to the SH degree, our shading code is supposed to scale roughly quadratically in the number of the SH degree.
Our novel technique thus is slower compared to LTC but also improves the visual quality of rendering in most cases as shown in section 8.2.
With this as our basis we now draw our conclusion.

Table 4: Run time comparison

	LTC	LTSH of degree 2 *(ours)*	LTSH of degree 4 *(ours)*
Lighting pass	0.44 ms	0.95 ms	1.78 ms
Area lighting only	0.30 ms	0.81 ms	1.64 ms

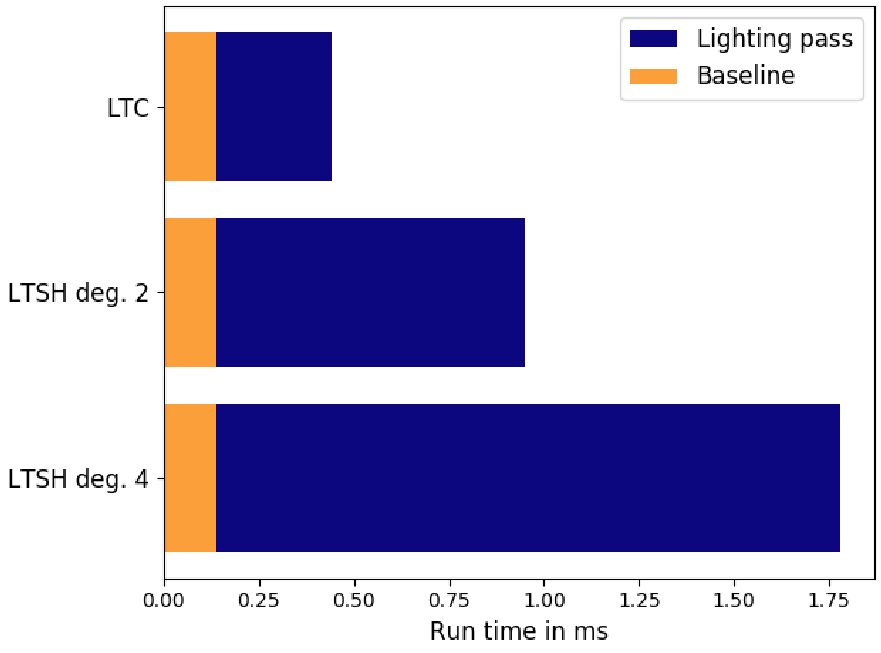


Figure 24: Run time comparison

9 Conclusion and Future Work

With the results of our technique at hand, we now conclude the findings of this thesis. We briefly discuss, if linearly transformed spherical harmonic expansions should be used, its future potential and what the possible next steps are.
First of all, it is important to notice, that this thesis is a proof of concept and not a refined technique. We therefore do not recommend the use of LTSH in production rendering as of now. The performance drawback compared to quality improvement does not make it appealing for real applications in relation to linearly transformed cosines [HDHN16]. Also, ringing artifacts, not present in LTC, occasionally happen.
That being said, LTSH introduces an interesting improvement to LTC which is likely to become viable with further research.
The most obvious point to improve our technique lies in refining our BRDF fitting process. A C++ implementation with importance sampling would be a large step towards better BRDF fits. While lowering the computation time to find fits in the preprocess may not drastically increase the value of our technique for production rendering, it makes experimenting with parameters for the fitting process less cumbersome. Trying different error measures (for example L^3 instead of L^2 as proposed by Heitz et al. [HDHN16]) or bounds for the matrix parameters could be investigated.
Another use of a more efficient fitting process lies in increasing the samples. A higher density in samples enables a closer approximation with the fitting algorithm. With importance sampling and a higher sample density, the highlight of a highly specular BRDF is unlikely to be missed, as opposed to our current sampling.
With more versatile fitting, continuous fits are likely to be found, rendering the noisy dithering unnecessary. Instead, interpolation between the fits, as in LTC, is then possible.
We do not support different indices of refraction (IORs) at the moment. Adding them as a dimension to our pre-computed BRDF fit table is possible. However, it has serious implications on the computation time of the pre-process, depending on the resolution for the IOR values.
A further possible extension for our technique is the support of anisotropic BRDFS. They would again introduce an additional dimension, for the φ component of the view direction ω_o, to our pre-computed table. Supporting both, different IORs and anisotropic BRDFs, is thus unlikely to be feasible.
Investigation the textured polygonal lights with LTSH is also another field for future work. We do not asses, if the process described by Heitz et al. [HDHN16] is suitable for our technique. Since they rely on the cosine distribution for texture fetching, some modification is likely to be needed. Simple texture-like lighting is achieved by breaking down the polygonal light into smaller polygonal lights of different color. This however has implications on the performance, as the shading time scales linearly in the number of polygon vertices.
Another large area to improve our technique is the optimization of the shading. The shading mainly consists of the integral of the SH expansion over the polygon. Further improving this integration holds the largest potential for a speedup of our technique. A first step is the use of the optimal lobe directions for the zonal harmonic factorization. The weights for the rotation need to be recalculated but the resulting code uses a more sparse and thus faster rotation (revisit section 7.2 for more information).
Another approach to optimize the shading would be to use a different integration method altogether. We had the idea to only use even or odd bands for our SH expansion. The resulting integral may be easier to solve and these limited SH bases still offer improved versatility compared to LTC. We do not asses this approach here due to time constraints.
Our technique could be suitable for line and disk lights as discussed in section 4. Heitz and Hill [HH17] extended the LTC model to support line and disk lights and using their approach with our technique is an interesting area for further research.

There are many entry points to further improve our technique. We think the improvements offered by the versatility of spherical harmonics compared to the clamped cosine make our technique an interesting field of further research. Our implementation already shows improvements in various configurations and could become an attractive solution for polygonal area lighting in production rendering.

10 Acknowledgments

I would like to thank NVIDIA for providing the *Falcor* framework [BYC$^+$19] and Epic Games for the Sun Temple scene available in the ORCA library [Gam17]. I used both for the rendered images and profiling.
I would also like to thank my advisor Christoph Peters for the countless hours of review, feedback and advisory over the whole process of this bachelors thesis. His expertise and time investments made this whole thesis possible. Furthermore, he provided me with code to evaluate SH expansions in C++ and Python, a Python script to create images with magnified areas (as seen in section 8.2) and a blender script to project spherical textures onto the unit sphere (used for all spherical visualizations).
Last but not least I want to thank my parents, siblings and grandparents for their support during my whole life and especially my recent university years.

Bibliography

[AMHH$^+$18] Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michal Iwanicki und Sébastien Hillaire: *Real-time rendering, fourth edition.* AK Peters/CRC Press, 2018, ISBN 978-1-1386-2700-0.

[Arv95] James Arvo: *Applications of Irradiance Tensors to the Simulation of non-Lambertian Phenomena.* In: *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, Seiten 335–342, New York, NY, USA, 1995. ACM, ISBN 0-89791-701-4. `http://doi.acm.org/10.1145/218380.218467`.

[BXH$^+$18] Laurent Belcour, Guofu Xie, Christophe Hery, Mark Meyer, Wojciech Jarosz und Derek Nowrouzezahrai: *Integrating Clipped Spherical Harmonics Expansions.* ACM Trans. Graph., 37(2), März 2018, ISSN 0730-0301. `https://doi.org/10.1145/3015459`.

[BYC$^+$19] Nir Benty, Kai Hwa Yao, Lucy Chen, Tim Foley, Matthew Oakes, Conor Lavelle und Chris Wyman: *The Falcor Rendering Framework*, Oktober 2019. `https://github.com/NVIDIAGameWorks/Falcor`, `https://github.com/NVIDIAGameWorks/Falcor`.

[CJAMJ05] Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller und Henrik Wann Jensen: *Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions.* In: *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, Seite 1166–1175, New York, NY, USA, 2005. Association for Computing Machinery, ISBN 9781450378253. `https://doi.org/10.1145/1186822.1073328`.

[CT82] Robert L Cook und Kenneth E. Torrance: *A reflectance model for computer graphics.* ACM Transactions on Graphics (TOG), 1(1):7–24, 1982.

[DHB17] Jonathan Dupuy, Eric Heitz und Laurent Belcour: *A Spherical Cap Preserving Parameterization for Spherical Distributions.* ACM Trans. Graph., 36(4), Juli 2017, ISSN 0730-0301. `https://doi.org/10.1145/3072959.3073694`.

[Gam17] Epic Games: *Unreal Engine Sun Temple, Open Research Content Archive (ORCA)*, October 2017. `http://developer.nvidia.com/orca/epic-games-sun-temple`, `http://developer.nvidia.com/orca/epic-games-sun-temple`.

[HDHN16] Eric Heitz, Jonathan Dupuy, Stephen Hill und David Neubelt: *Real-time Polygonal-light Shading with Linearly Transformed Cosines.* ACM Trans. Graph., 35(4):41:1–41:8, Juli 2016, ISSN 0730-0301. `http://doi.acm.org/10.1145/2897824.2925895`.

[Hei14] Eric Heitz: *Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs.* Journal of Computer Graphics Techniques (JCGT), 3(2):48–107, June 2014, ISSN 2331-7418. `http://jcgt.org/published/0003/02/03/`.

[HH17] Eric Heitz und Stephen Hill: *Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines.* `https://eheitzresearch.wordpress.com/757-2/`, ACM SIGGRAPH Courses 2017, 2017. Accessed: 2020-03-05.

[Hof19] Naty Hoffman: *Fresnel Equations Considered Harmful.* In: Reinhard Klein und Holly Rushmeier (Herausgeber): *Workshop on Material Appearance Modeling.* The Eurographics Association, 2019, ISBN 978-3-03868-080-2.

[Kaj86] James T. Kajiya: *The Rendering Equation.* In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, Seite 143–150, New York, NY, USA, 1986. Association for Computing Machinery, ISBN 0897911962. `https://doi.org/10.1145/15922.15902`.

[LDSM16] Pascal Lecocq, Arthur Dufay, Gaël Sourimant und Jean Eudes Marvie: *Accurate Analytic Approximations for Real-Time Specular Area Lighting.* In: *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '16, Seite 113–120, New York, NY, USA, 2016. Association for Computing Machinery, ISBN 9781450340434. `https://doi.org/10.1145/2856400.2856403`.

[Mac67] T. M. MacRobert: *Spherical Harmonics, An Elementary Treatise on Harmonic Functions with Applications, Third edition.* Pergamon Press, 1967.

[Mar63] Donald W Marquardt: *An algorithm for least-squares estimation of nonlinear parameters.* Journal of the society for Industrial and Applied Mathematics, 11(2):431–441, 1963.

[NSF12] Derek Nowrouzezahrai, Patricio Simari und Eugene Fiume: *Sparse Zonal Harmonic Factorization for Efficient SH Rotation.* ACM Trans. Graph., 31(3), Juni 2012, ISSN 0730-0301. `https://doi.org/10.1145/2167076.2167081`.

[PD19] Christoph Peters und Carsten Dachsbacher: *Sampling Projected Spherical Caps in Real Time.* Proc. ACM Comput. Graph. Interact. Tech., 2(1), Juni 2019. `https://doi.org/10.1145/3320282`.

[Pic92] K. P. Picott: *Extensions of the Linear and Area Lighting Models.* IEEE Computer Graphics and Applications, 12(02):31–38, mar 1992, ISSN 1558-1756.

[PJH16] Matt Pharr, Wenzel Jakob und Greg Humphreys: *Physically based rendering: From theory to implementation.* Morgan Kaufmann, 2016. available here: `http://www.pbr-book.org/`, accessed 2020-03-03.

[Sch94] Christophe Schlick: *An Inexpensive BRDF Model for Physically-based Rendering.* Computer Graphics Forum, 13(3):233–246, 1994. `https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.1330233`.

[sci20] *Documentation of scipy.optimize.curve_fit.* `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html`, 2020. Accessed: 2020-03-02.

[Slo08] Peter Pike Sloan: *Stupid spherical harmonics (sh) tricks.* In: *Game developers conference*, Band 9, Seite 42, 2008.

[SM09] Peter Shirley und Steve Marchner: *Fundamentals of Computer Graphics. Third edition. AK Peters.* CRC Press, 2009.

[TB97] Lloyd N. Trefethen und Steve Bau: *Numerical Linear Algebra.* SIAM, 1997, ISBN 978-0-898713-61-9.

[VG95] Eric Veach und Leonidas J. Guibas: *Optimally Combining Sampling Techniques for Monte Carlo Rendering.* In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, Seite 419–428, New York, NY, USA, 1995. Association for Computing Machinery, ISBN 0897917014. `https://doi.org/10.1145/218380.218498`.

[WLWF08] Lifeng Wang, Zhouchen Lin, Wenle Wang und Kai Fu: *One-Shot Approximate Local Shading.* Autodesk Technical Report, 2008.

[WMLT07] Bruce Walter, Stephen R. Marschner, Hongsong Li und Kenneth E. Torrance: *Microfacet Models for Refraction Through Rough Surfaces.* In: *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, Seiten 195–206, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association, ISBN 978-3-905673-52-4. `http://dx.doi.org/10.2312/EGWR/EGSR07/195-206`.

[WR18] Jingwen Wang und Ravi Ramamoorthi: *Analytic Spherical Harmonic Coefficients for Polygonal Area Lights.* ACM Trans. Graph., 37(4), Juli 2018, ISSN 0730-0301. `https://doi.org/10.1145/3197517.3201291`.

Erklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe. Die Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt und von dieser als Teil einer Prüfungsleistung angenommen.

Karlsruhe, den 25. März 2020

(Jan Allmenröder)