Efficient Physically Based Rendering with Analytic and Neural Approximations

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Engineering

by

Aakash KT
2020801005
aakash.kt@research.iiit.ac.in

Advisor: Dr. P. J. Narayanan

International Institute of Information Technology Hyderabad
500 032, India

Jun 2025

International Institute of Information Technology Hyderabad
Hyderabad, India

CERTIFICATE

This is to certify that work presented in this thesis proposal titled ***Efficient Physically Based Rendering with Analytic and Neural Approximations*** by *Aakash KT* has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Advisor: Dr. P. J. Narayanan

Abstract

Path tracing is ubiquitous for photorealistic rendering of various real-world appearances. It follows the principles of light transport adapted from physics, which describe light propagation as a set of integral equations. These equations are stochastically evaluated by tracing light rays in virtual scenes. Such stochastic evaluations with ray-tracing form the bulk of the path tracing algorithm, which is widely used in the industry.

Stochastic evaluations in path tracing converge to the correct answer in time that is inversely proportional to the square root of the number of iterations. This coupled with the fact that the underlying integrals are often complex and high dimensional results in large compute complexity. Research efforts have thus largely focused on accelerating path tracing by improving the stochastic sampling processes. However, it is interesting to look at efficient analytic approximations by making reasonable assumptions on the nature of these light transport integrals. Such analytic methods have the potential to achieve zero variance at the outset. Practically, they are often used in conjunction with stochastic methods thereby achieving lower variance than the fully stochastic counterparts.

The primary focus of this thesis is to develop new (semi-)analytic methods and improve existing ones to accelerate direct lighting computations in path tracing. We base our research on the theory of Linearly Transformed Cosines (LTC) applied for direct lighting from area lights. The *LTC method* produces plausible renderings by building on the principles of light transport from the ground up and has proved useful for tasks other than real-time rendering We make the following three contributions that either build on LTCs or improve it.

We first explore fully-analytic direct lighting for arbitrarily shaped area lights, built on LTCs at the core. Due to assumptions of the LTC method, it can only handle polygonal area lights. Furthermore, rendering shadows with LTCs require stochastic evaluations - our contribution here relaxes these assumptions, enabling fully-analytic direct lighting with shadows from an arbitrary shaped area light. We show that our method achieves plausible and noise-free renderings compared to semi-analytic LTCs and ground truth ray-tracing, given equal compute budget.

Our second contribution improves the LTC formulation by relaxing an assumption that restricts their usage to isotropic GGX reflection distribution functions. Isotropic GGX restricts the kind of reflectance that can be modeled, ultimately restricting the visual fidelity of rendering with LTCs. We identify the core issues that prevent the use of anisotropic GGX, and propose solutions for each. Our contribution not only benefits (semi-)analytic rendering with LTCs but also other related methods that use LTCs at their core.

In our third contribution, we extend the theory of Linearly Transformed Spherical Distributions (LTSDs), which are a superset of and core to LTCs, to work with phase functions. This enables us to analytically compute in-scattered radiance, which we build on to semi-analytically render single scattering. Like the original LTC method, we ground our derivations and formulations on the Volume Rendering Equation (VRE) which paves the way for plausible photorealistic renderings despite the biased nature of our method. In effect, our work enables semi-analytic volumetric rendering with area lights.

Another focus of this thesis is to accelerate path tracing computations with neural approximations. Path tracing poses a challenge for (semi-)analytic methods as the underlying integrals become increasingly more complex and high-dimensional. In this setting, it is beneficial to use a neural network to approximate parts of these complex integrals for improved efficiency. Previous works have demonstrated the usage of neural networks in this manner, however their main focus was on scenes with relatively shorter light paths. We instead focus on scenes where longer paths are required for accurate rendering. More specifically, we tackle the problem of accelerating multiple scattering computations in hair using neural networks.

The contributions of this thesis range from exploring analytic and semi-analytic methods to neural approximations for physically based rendering. Moreover, our contributions advance the state of the art, in terms of efficiency, fidelity and capability. We believe our work can inspire further research in computer graphics and adjacent fields.

Acknowledgements

The work in this thesis is a continuation of my work in MS, with Prof. P. J. Narayanan yet again as my advisor. Needless to say, this work has been possible majorly because of his support. To this day, whenever I need to make a major decision, I implicitly think ”how would PJN have approached this?”.

The decision for a PhD in my alma mater from MS was a tough one, which was taken at the advent of COVID. A big part of the decision was my relationship with Prof. P. J. Narayanan - and the environment at IIIT. But before that, I was all set to go to GRAPHDECO, Inria for my PhD with George Drettakis and Adrien Bousseau. Due to COVID and uncertainty, they had to make a decision to stop international hires at the time. I really appreciate the way George and Adrien handled the situation. I am glad to still be in touch with them over the years, and to occasionally grab a coffee/beer at conferences.

I have been fortunate to have external collaborators and mentors in Eric Heitz and Jonathan Dupuy. I still have no clue why they decided to collaborate with me - a lowly second year PhD student - but I am sure glad they did! Both of them were incredibly tough on me, and I now see why. I have them to thank for my current approach towards problem solving and making directed progress on a problem.

I also have Giljoo Nam to thank, first and foremost for taking me on as an intern at Meta. Not only was this a huge confidence boost, but it gave me an exposure to real-world pipelines and how problems need to be clearly defined within them. Giljoo was kind enough to introduce me to yet another set of amazing researchers - Christophe Hery, Matt Chiang, Adrian Jarabo, Carlos Aliaga and Olivier Maury. Its hard to quantify how much fun I had working with them, all while learning a great ton. It’s amazing to have worked with someone, and then run into them at conferences - the instant connect persists!

I also want to specifically thank Chris Wyman, Markus Kettunen and Aaron Lefohn for their confidence in my abilities. They have supported me in ways not possible to enumerate. I have had very deep and insightful conversations with all of them and I look forward to a day when we work together!

A big thank you to Parikshit, my brother, friend, mentor. His role in my life is way beyond any description - but safe to say, I would probably not be a researcher in this field if it weren’t for him.

To Pulkit and Dhawal, the first students I mentored. I probably learned a lot more from them than they did from me! And my learnings were put to good use when Ishaan Shah joined our lab. All in all, it was super fun to collaborate with all of these individuals.

To my wife, Dipanwita Ray, who not only makes my work enjoyable, but my life fulfilling. Research and everything else is incomplete without her, especially during conference travels where we plan crazy trips!

To my mom, dad and brother for their unwavering support for everything under the sun. I can finally say that my mom isn't the only doctor in the family! (although, she is probably the only useful one).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"There is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order of things."
- Niccolò Machiavelli

Rendering is the process of generating synthetic images from precise virtual scene descriptions. A virtual scene is described as a collection of textured surfaces, participating media, light sources and a virtual camera. The task of a *renderer* is to then generate an image of this scene by estimating the light arriving at the virtual camera.

Physically Based Rendering (PBR) is a rendering process that uses principles of light propagation from physics to compute the light that falls on the virtual camera [1]. There is a good reason to follow physical principles: if done right, the resulting image is guaranteed to reflect reality. Physically based rendering generates images that are photorealistic and is the method of choice for visual effects in movies and more recently, computer games.

Derived and adapted from the theory of Radiative Transfer [2], the physical principles of light propagation used in computer graphics boil down to an integral equation, commonly referred to as the *Rendering Equation* (RE). This equation is simple in its form and completely describes the *light transport* in the scene. Although simple looking, this equation is infinite dimensional and admits no closed form solution. Furthermore, a few terms in this equation require knowledge of the virtual scene's geometry. To solve for the light transport, PBR methods resort to iterative numerical techniques. Specifically, Monte Carlo (MC) integration in conjunction with ray-casting is used to estimate the solution of the RE. MC is highly preferred in PBR since it's easy to implement, and imposes no restriction on the form of the integral that is to be solved.

The flexibility of Monte Carlo for solving arbitrary integrals like the RE comes at a cost: MC converges to the right answer at the rate of $O(\frac{1}{\sqrt{N}})$ [1], where N is the number of iterations (or samples). This means that for the Monte Carlo error to be halved, we need to evaluate four times as many samples. Added to this, each sample also requires ray-casting which is an expensive operation since it queries the scene's geometry. This makes physically based rendering with Monte Carlo compute intensive, and it is typical to expect per-frame rendering times of upto several hours, depending on the scene complex-

ity and hardware configuration. There has been extensive research to improve the convergence rate of Monte Carlo, which in their essence try to use better distributed MC samples.

Instead of exactly following the principles of physics, it is common to introduce reasonable approximations for the sake of efficiency. For example, RE already assumes geometric optics ignoring the wave nature of light. This greatly improves the efficiency of PBR at the cost of not being able to render wave phenomenon, which are uncommon in everyday settings anyway. Another example is the approximation of the reflectance from surfaces using a statistical distribution of micro-surface perturbations, rather than physically representing these micro-perturbations. Yet another commonly found simplification is assuming delta light sources, which reduces the solution of the RE to a simple closed form. Solving the RE in presence of delta light does not require the use of MC. Other approximations, restrictions or simplifications can also be applied on the RE that make it easier to solve or even allow closed form (analytic) integration.

The primary goal of this dissertation is to develop and improve fully-analytic and semi-analytic methods to efficiently solve a restricted but useful form of the rendering equation. We focus our attention on the *direct lighting* version of the RE which accounts for illumination from light sources only and not due to interreflections. Such direct lighting computations are at the core of physically based rendering and we first tackle the simpler case of surface only scenes. We then progressively build up a solution for scenes with participating media (fog, mist). For both scenes types, we further restrict our study to light sources having finite area, or *area lights*. Area lights are one of the most realistic types of light sources and mimic real-world characteristics well. We describe our three contributions in this setting in Sections 1.1, 1.2 & 1.3.

As a secondary goal, this thesis also explores using neural approximations to accelerate full path tracing computations, that is, considering illumination from light sources *and* from interreflections. Specifically, we focus on scenes where longer light paths carry significant energy and contribute heavily to the final rendered image. To this end, we demonstrate using an online trained neural network to achieve efficient, high-fidelity and controllable rendering of human hair. More details are given in Sect. 1.4.

Finally in Sect. 1.5, we discuss the structure of this thesis and the target audience. We end this chapter by listing the publications associated with this thesis.

1.1 Fully-analytic direct lighting from area lights

We formulate a fully analytic solution to direct lighting from area lights for surface-only scenes and describe a set of algorithms to compute it. Our solution is based on a well known semi-analytic method that is based on the theory of Linearly Transformed Cosines (LTCs), henceforth referred to as the *LTC method*.

The rendering equation for direct lighting from an area light integrates over the solid angle that the area light subtends. We refer to this solid angle as the *domain* of integration for the RE. The LTC

Figure 1.1 Brushed-metal appearance (Credits: Amazon product). Notice the elongated highlights on the Kettle's body.

method computes illumination from an area light by transforming this domain to a space where analytic integration can be done. The semi-analytic nature of LTCs stems from the fact that it ignores *visibility* in the scene, which is injected back into the solution with a secondary MC estimate.

We observe that this domain of integration can be modified to account for visibility, which can then transformed to analytically compute the full integral with visibility. Since the domain itself accounts for visibility, we can get rid the secondary MC estimate making our approach fully analytic. Computing a domain that accounts for visibility is non-trivial, and requires a set of carefully crafted algorithms. We describe our formulation and associated algorithms in detail in Chapter 3.

1.2 Analytic direct lighting of anisotropic appearance from area lights

One of the core assumptions of LTCs restricts the type of surface reflectance that can be rendered. This assumptions means that *anisotropic* appearance, which is typically found in brushed metal-like surfaces (see Fig. 1.1) cannot be rendered with LTCs. This is done primarily to reduce the complexity of and memory required to store the associated precomputations.

Figure 1.2 Real-world photograph (Credits: Teo Crawford, video thumbnail). Notice the glow around the street lamps, caused by scattering due to the presence of participating media.

The ability to render anisotropic appearance is important from a photorealism perspective: ultimately the more real-world effects that can be rendered, the better the photorealism. We analyse LTCs and point out the exact reasons for avoiding anisotropic appearances which stem for certain core limitations. We then exploit core properties of LTCs and devise a method that alleviates these limitations without having a significant memory or complexity overhead. Our method makes the rendering of anisotropic appearance with LTCs practical and improves the fidelity of semi-analytic rendering with LTCs. Furthermore, it can also be used as a drop-in replacement for fully-analytic rendering described in Sect. 1.1. We describe our analysis and solution in detail in Chapter 4.

1.3 Semi-analytic single scattering with area lights

Direct lighting in the presence of participating media is commonly referred to as *single scattering*. Chapter 5 explores efficient semi-analytic rendering of single scattering with area lights, building on the core theory of LTCs.

Like anisotropic appearance, rendering participating media in virtual scenes is also important from a photorealism perspective. Consider that almost any real-world scene has some form of scattering - this is typically seen in the form of *god rays* or a general glow especially around light sources (see Fig. 1.2). Enabling semi-analytic rendering of such effects is thus crucial to obtain increased photorealism of renderings from this class of methods.

Figure 1.3 Real-world photograph of the author's presentation at EGSR 2023 (Credits: Carlos Aliaga). Human hair has complex appearance, including it's color (blonde, brown), a soft look (note the hair of the audience and the author at the podium) all of which are a result of multiply scattered and long length light paths.

In Chapter 5, we describe our solution to achieve semi-analytic rendering of such effects that is competitive with widely popular Monte Carlo rendering methods. Our solution is built on extensive analysis of LTCs, and traces back to and builds on the core theory of LTCs - Linearly Transformed Spherical Distributions. We base our analysis and derivations on principles of light transport, ensuring that our approximation produces plausible renderings.

1.4 Neural approximation for higher order path tracing

In Chapter 6, we turn our attention to neural approximations of higher order path tracing. Our explorations are towards enabling real-time path tracing of human hair feasible.

Path tracing considers interreflections in addition to illumination from light sources - this implies that light that bounces around in the scene can also illuminate parts of the scene. Such interreflections are common in everyday scenes as well - consider a lit room with only one window and no lights. For

surface-only scenes, this computation is relatively efficient and can be further accelerated using a neural approximation in the form of a neural radiance cache [3].

We instead specifically consider scenes with human hair, which exhibit strong interreflections. Most of the color and the soft look of hair is due to such interreflections and contributions from very long light paths (bounces) within the hair volume. Approximating this kind of *multiple scattering* is non-trivial, and requires rethinking assumptions of neural approximations. Our analysis and the resulting method is described in detail in Chapter 6.

1.5 Thesis Structure and Target Audience

Chapter 2 reviews the necessary background and ends by listing the core assumptions of previous methods, both for semi-analytic area lighting and neural approximations of path tracing. This chapter not only lays the foundation to understand the contents of this thesis, but also gives context on why these problems exist and are important. The next four chapters discuss our four contributions as discussed above in detail. Finally, Chapter 7 discusses the future of this direction of research before concluding the thesis. The target audience of this thesis are rendering researchers or more generally, computer graphics researchers.

1.6 Publications

The work done in this thesis resulted in two primary peer-reviewed publications, listed below.

[P1] **Chapter 3**: **Aakash KT**, Parikshit Sakurikar and P. J. Narayanan, "Fast Analytic Soft Shadows from Area Lights", in proceedings of *Eurographics Symposium on Rendering (EGSR)*, 2021.

[P2] **Chapter 4**: **Aakash KT**, Eric Heitz, Jonathan Dupuy and P. J. Narayanan, "Bringing Linearly Transformed Cosines to Anisotropic GGX", in proceedings of *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D), 2022*, **Best Paper Award**
Collaboration: Unity 3D research, Grenoble
Journal: Proceedings of the ACM on Computer Graphics and Interactive Techniques (PACM-CGIT), Vol. 5, No. 1

[P3] **Chapter 5**: **Aakash KT**, Ishaan Shah and P. J. Narayanan, "Linearly Transformed Spherical Distributions for Interactive Single Scattering with Area Lights", in proceedings of *Eurographics (EG), 2025*
Journal: Computer Graphics Forum (CGF), Vol. 44, No. 2.

[P4] **Chapter 6**: **Aakash KT**, Adrian Jarabo, Carlos Aliaga, Matt Jen-Yuan Chiang, Olivier Maury, Christophe Hery, P. J. Narayanan and Giljoo Nam, "Accelerating Hair Rendering by Learning High-Order Scattered Radiance", in proceedings of *Eurographics Symposium on Rendering (EGSR), 2023*
Collaboration: Meta Reality Labs, Pittsburgh
Journal: Computer Graphics Forum (CGF), Vol. 42, No. 4.

The source code and/or project pages of each are given below:

- Fast Analytic Soft Shadows from Area Lights
- Bringing Linearly Transformed Cosines to Anisotropic GGX
- Linearly Transformed Spherical Distributions for Interactive Single Scattering with Area Lights
- Accelerating Hair Rendering by Learning High-Order Scattered Radiance

Finally, this thesis resulted in the following related publications, not only tackling problems related to the ones presented in this thesis but also general computer graphics research problems (* indicates equal contribution):

[P1] **Aakash KT**, Parikshit Sakurikar, Saurabh Saini and P. J. Narayanan, "A Flexible Neural Renderer for Material Visualization", in proceedings of *ACM SIGGRAPH Asia, Technical Briefs*, 2019.

[P2] Ishaan Shah*, **Aakash KT*** and P. J. Narayanan, "Combining Resampled Importance and Projected Solid Angle Samplings for Many Area Light Rendering", in proceedings of *ACM SIGGRAPH Asia, Technical Communications*, 2023.

[P3] Dhawal Sirikonda, **Aakash KT** and P. J. Narayanan, "Real-Time Rendering of Arbitrary Surface Geometries using Learnt Transfer", in proceedings of *ICVGIP*, 2022.

[P4] Dhawal Sirikonda, **Aakash KT** and P. J. Narayanan, "PRTT: Precomputed Radiance Transfer Textures", in proceedings of *Eurographics (EG)*, 2022.

[P5] Pulkit Gera, **Aakash KT** and P. J. Narayanan, "Neural View Synthesis with Appearance Editing from Unstructured Images", in proceedings of *ICVGIP*, 2021.

Chapter 2

Preliminaries

"The only way you can stay on top is to remember to touch bottom and get back to basics."
- Shane Black

This chapter introduces the preliminaries of light transport from the ground up. We start by describing the Volume Rendering Equation (VRE) which completely describes the light transport in a scene and is frequently used in graphics. We also look at a simplified form of this equation that ignores participating media and a related path formulation of this simplified form. We then describe the direct light transport equation, which is the primary focus of this thesis. This is followed by a discussion on various types of light sources, giving special attention to area lights. We then briefly discuss the GGX Bi-Directional Reflectance Distribution Functions (BRDF) which is one of the most commonly used BRDFs in graphics.

Next, we discuss Monte Carlo (MC) methods to solve for the light transport in a scene and some of their properties. Specifically, we describe Monte Carlo estimators of both direct light and path tracing. After this discussion which falls under stochastic methods, we turn our attention to analytic methods for light transport. We mainly focus on Linearly Transformed Cosines (LTCs) which is a semi-analytic method to compute the direct lighting from area lights. We end by discussing neural approximations of path tracing, which is another focus of this thesis.

Finally in Sect. 2.8, we list our contributions in the context of the preceding discussions. This chapter and the rest of the dissertation makes use of mathematical notation for the light transport equations, and the specific symbols used are denoted in Table 2.1.

2.1 Light Transport

Light Transport describes how light propagates in a virtual scene with precise equations, taking into account inter-reflections, scattering, absorption and different surface appearances. Mathematically, it is described by a differential equation, known as the *Radiative Transfer Equation* (RTE) [2].

Symbol	Description
$\mathbf{x}$	A 3D point in the scene.
$\bar{\mathbf{x}}$	Describes a light path in path tracing.
$n_{\mathbf{x}}$	Normal vector at a 3D point $\mathbf{x}$.
Ω	The solid angle of the upper hemisphere aligned with the z-axis.
$\langle I \rangle$	Monte Carlo estimation for the integral I.
Ω_k	Describes a space of light paths $\bar{\mathbf{x}} = \mathbf{x}_0 ... \mathbf{x}_k \in \Omega_k$.
ω_i	Incoming radiance direction.
ω_o	Outgoing radiance direction.
A	Denotes an area light.
$A(\mathbf{x})$	Solid angle subtended by the area light at a 3D point $\mathbf{x}$.
$f_r(\mathbf{x}, \omega_o, \omega_i)$	The Bi-Directional reflectance Distribution Function (BRDF) at point $\mathbf{x}$ for a pair of directions ω_o and ω_i. The BRDF describes the appearance of $\mathbf{x}$.
$\rho(\mathbf{x}, \omega_o, \omega_i)$	The phase function at a point in the participating media. The phase function describes the scattering behaviour (forward or backward) of the medium.
$L_e(\mathbf{x}, \omega)$	Radiance emitted from point $\mathbf{x}$ in the direction ω.
$L_i(\mathbf{x}, \omega)$	Incoming radiance at point $\mathbf{x}$ from the direction ω.
$L_o(\mathbf{x}, \omega)$	Outgoing radiance from point $\mathbf{x}$ in the direction ω.
$L_s(\mathbf{x}, \omega)$	In-scattered radiance at point $\mathbf{x}$ in the direction ω.
$T(\mathbf{x}, \mathbf{y})$	The transmittance or attenuation between points $\mathbf{x}$ and $\mathbf{y}$. $T(\mathbf{x}, \mathbf{y}) = 0$ when no participating media is present.
μ_a	Absorption Coefficient.
μ_s	Scattering Coefficient.
M	A 3×3 matrix.

Table 2.1 Symbols and their descriptions used in this dissertation.

2.1.1 Volumetric Light Transport & Volume Rendering Equation

The Volume Rendering Equation (VRE) is obtained by integrating the RTE and is frequently used in graphics. This equation gives the radiance L_o arriving at a point $\mathbf{y}$ form the direction ω_o in the presence of media as:

$$\begin{aligned} L_o(\mathbf{y}, \omega_o) = \int_0^x T(\mathbf{y}, \mathbf{z}) \left[\mu_a(\mathbf{z}) L_e(\mathbf{z}, \omega_o) + \mu_s(\mathbf{z}) L_s(\mathbf{z}, \omega_o) \right] dz \\ + T(\mathbf{y}, \mathbf{x}) \int_\Omega f_r(\mathbf{x}, \omega_o, \omega_i) L_i(\mathbf{x}, \omega_i) |n_{\mathbf{x}} \cdot \omega_i| d\omega_i, \end{aligned} \tag{2.1}$$

where $\mathbf{x} = \mathbf{y} - x\omega_o$ lies on a surface (Fig. 2.1) and $\mathbf{z} = \mathbf{y} - z\omega_o$. Terms in this equation can be recursively replaced with integrals of the similar form, implying that the VRE is infinite dimensional. The recursive substitution also means that this equation implicitly accounts for inter-reflections or *global illumination* in the scene.

In the VRE, the first integral calculates the increase in radiance due to medium emission and in-scattering along a ray that originates at $\mathbf{y}$ and terminates at a surface point $\mathbf{x}$. This integral also accounts for attenuation from the medium with the transmittance function T. The illumination at the surface point $\mathbf{x}$ is computed by yet another integral and is again attenuated by the transmittance $T(\mathbf{y}, \mathbf{x})$. The

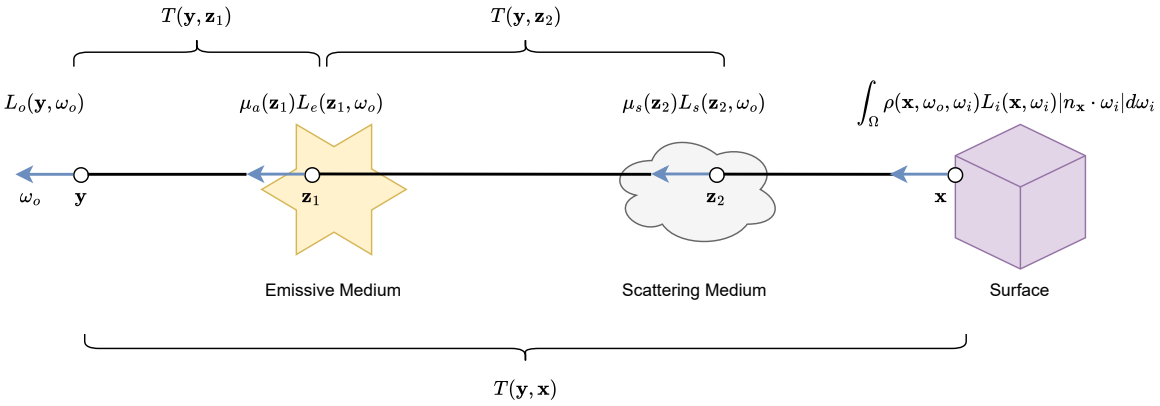


Figure 2.1 Overview of the Volume Rendering Equation (VRE): The radiance L_o at $\mathbf{y}$ in direction ω_o is computed by considering emissive events ($\mathbf{z}_1$), scattering events ($\mathbf{z}_2$), surface scattering ($\mathbf{x}$) and attenuation due to the medium ($T(\mathbf{y}, \mathbf{z}_1), T(\mathbf{y}, \mathbf{z}_2), T(\mathbf{y}, \mathbf{x})$).

summation of these two terms gives the radiance $L(\mathbf{y}, \omega_o)$ arriving at $\mathbf{y}$ from ω_o. All radiance terms in the VRE have *implicit* visibility: that is, it naturally accounts for objects that obstruct radiance towards $\mathbf{x}$ or $\mathbf{y}$. Refer to Fig. 2.1 for a pictorial depiction of the terms and intuition of Eq. 2.1.

Eq. 2.1 forms the basis of our derivations for semi-analytic rendering of participating media with area light in Chapter 5. We expand on various terms in this equation more in that chapter.

2.1.2 Light Transport Equation

In the absence of participating media, the transmittance is equal to one and the absorption and scattering is equal to zero: $T(\mathbf{y}, \mathbf{x}) = 1$ and $\mu_a = \mu_s = 0$. After simplification, we get the *Light Transport Equation* (LTE):

$$L_o(\mathbf{x}, \omega_o) = \int_\Omega f_r(\mathbf{x}, \omega_o, \omega_i) L_i(\mathbf{x}, \omega_i) |n_\mathbf{x} \cdot \omega_i| d\omega_i. \tag{2.2}$$

Like Eq. 2.1, L_i in the integrand can be recursively replaced with an integral of the similar form and thus this equation too naturally accounts for global illumination. Visibility in the LTE is also accounted for implicitly, similar to the VRE. This equation is a useful simplification for scenes with no participating media, and is essentially a subset of the VRE.

2.1.3 Path formulation of the Light Transport Equation

It is convenient to look at Eq. 2.2 as an integral equation over *paths* rather than directions: after all, it describes all paths a light ray can take in the scene. The *path integral formulation* described by Eric Veach [4] models the radiance L_o arriving at a pixel as:

$$L_o = \sum_{k=1}^{\infty} \int_{\Omega_k} f'(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}), \tag{2.3}$$

where Ω_k is the space of light paths $\bar{\mathbf{x}} = \mathbf{x}_0..\mathbf{x}_k \in \Omega_k$ of length k with $k+1$ path vertices. $\mathbf{x}_k$ and $\mathbf{x}_0$ are 3D points placed on a light source and the sensor respectively, and the differential measure $\mathrm{d}\mu(\bar{\mathbf{x}})$ models the area/volume integration for each vertex in the path. The path contribution $f'(\bar{\mathbf{x}})$ is defined as:

$$f'(\bar{\mathbf{x}}) = L_e(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1})\, T'(\bar{\mathbf{x}}), \tag{2.4}$$

where L_e is the emitted radiance of the light source at $\mathbf{x}_k$ towards $\mathbf{x}_{k-1}$, and $T'(\bar{\mathbf{x}})$ is the path throughput:

$$T'(\bar{\mathbf{x}}) = \prod_{i=1}^{k-1} S(\mathbf{x}_{i+1} \rightarrow \mathbf{x}_i \rightarrow \mathbf{x}_{i-1}) G(\mathbf{x}_{i+1} \leftrightarrow \mathbf{x}_i), \tag{2.5}$$

where S is the scattering kernel (e.g. the BRDF) and G the the geometric term between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$.

The path formulation is extensively used in the explorations for neural approximations of higher order path tracing in Chapter 6.

2.1.4 Direct Lighting

As mentioned in Sect. 2.1.2, the LTE (Eq. 2.2) naturally accounts from global illumination in the scene due to its recursive nature. It is however useful to decompose the LTE into *direct lighting* and *global illumination* components: the former computes light arriving at $\mathbf{x}$ only from light sources and the latter considers light arriving at $\mathbf{x}$ due to reflection from other geometry in the scene.

Define $L_{d(j)}$ as the incident radiance from the j^{th} light source and L'_i as the incident radiance that is *not* from a light source. Assuming k light sources in the scene, we can decompose the LTE as:

$$L_o(\mathbf{x}, \omega_o) = \underbrace{\sum_{j=1}^{k} \int_\Omega f_r(\mathbf{x}, \omega_o, \omega_i) L_{d(j)}(\mathbf{x}, \omega_i) |n_\mathbf{x} \cdot \omega_i| d\omega_i}_{\text{Direct Lighting}} + \underbrace{\int_\Omega f_r(\mathbf{x}, \omega_o, \omega_i) L'_i(\mathbf{x}, \omega_i) |n_\mathbf{x} \cdot \omega_i| d\omega_i}_{\text{Global Illumination}}, \tag{2.6}$$

where the summation over all lights is due to the common assumption of linearity of light transport in graphics.

Let's take a closer look at direct lighting, since its the primary focus area of this thesis . Like with the LTE and VRE, the visibility in the direct lighting equation is implicit. For ease of proofs and implementation, we make this dependence on visibility explicit:

$$L_o(\mathbf{x}, \omega_o) = \sum_{j=1}^{k} \int_\Omega f_r(\mathbf{x}, \omega_o, \omega_i) L_{d(j)}(\mathbf{x}, \omega_i) V(\mathbf{x}, \omega_i) |n_\mathbf{x} \cdot \omega_i| d\omega_i, \tag{2.7}$$

where V is a binary function. We set $V = 1$ if the current light source in consideration is visible (not blocked by other geometry) from $\mathbf{x}$ in the direction ω_i. Otherwise, we set $V = 0$.

We will frequently drop the summation over all lights and assume the scene consists of a single light source for brevity. All our analysis and derivations easily extend to many lights by summing up individual contributions.

2.2 Light Sources

We will now look at two types of light sources that can illuminate a scene: area lights (Sect. 2.2.1) and point lights (Sect. 2.2.2). Other types of lights like an environment light, directional light can be used, but we will not discuss them here.

2.2.1 Area Lights

Area lights are among the most realistic types of light sources, and result in realistic looking illumination with soft and hard shadows.

Let the solid angle subtended at $\mathbf{x}$ by an area light A be $A(\mathbf{x})$. Given this definition, we can rewrite the direct lighting equation 2.7 by changing the domain of integration as:

$$L_o(\mathbf{x}, \omega_o) = \int_{A(\mathbf{x})} f_r(\mathbf{x}, \omega_o, \omega_i) L_d(\mathbf{x}, \omega_i) V(\mathbf{x}, \omega_i) |n_{\mathbf{x}} \cdot \omega_i| d\omega_i. \tag{2.8}$$

In this equation, the only change we've made is to integrate over the subtended solid angle $A(\mathbf{x})$ instead of Ω which is the solid angle of the upper hemisphere (Table 2.1). This change is possible since L_d, the direct radiance from A, will be zero for a direction not in $A(\mathbf{x})$.

It is useful to restrict area light to be diffuse and spatially constant. In effect, L_d will only be a function of the 3D point $\mathbf{x}$, and not the direction ω, i.e. $L_d(\mathbf{x}, \omega) = L_d(\mathbf{x})$. This also implies that L_d can be taken out of the integral in Eq. 2.8.

2.2.2 Point Lights

Point lights are one of the most simplistic types of light sources. They are defined by a Dirac Delta on position: essentially, a point light is a infinitesimally small point in 3D space.

A point light subtends an infinitesimally small solid angle at $\mathbf{x}$. Similar to area lights, we can define the infinitesimally small solid angle subtended at $\mathbf{x}$ by a point light P as $dP(\mathbf{x})$. The direct lighting equation can similarly be modified by changing the integration over this differential solid angle as:

$$L_o(\mathbf{x}, \omega_o) = \int_{dP(\mathbf{x})} f_r(\mathbf{x}, \omega_o, \omega_i) L_d(\mathbf{x}, \omega_i) V(\mathbf{x}, \omega_i) |n_{\mathbf{x}} \cdot \omega_i| d\omega_i. \tag{2.9}$$

Point lights are not physically plausible: they are useful to approximate radiance arriving from relatively distant lights. The main benefit of using them is that the radiance contribution due to a point light admits a closed form solution, which we will discuss in Sect. 2.5.1.

2.3 The GGX Bi-Directional Reflectance Distribution Function

In this section, we review the mathematical background specifically related to GGX BRDF.

The GGX ("*Ground Glass Unknown*") microfacet BRDF was introduced by Walter et al. [5] and its anisotropic extension by Heitz [6]. The equations of this model are as follows.

Normalized directions. ω_o denotes the view direction and ω_i the light direction, with the following parameterizations:

$$\omega = (x, y, z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta). \tag{2.10}$$

Normal Distribution Function (NDF). The NDF represents the statistical distribution of specular microfacets that reflect the incident light. It is parameterized by two roughness parameters (α_x, α_y):

$$D_{\text{ggx}}(\omega) = \frac{1}{\pi\,\alpha_x\,\alpha_y\left(\frac{x^2}{\alpha_x^2} + \frac{y^2}{\alpha_y^2} + z^2\right)^2}. \tag{2.11}$$

Masking-shadowing. The masking-shadowing function G_2 computes the attenuation due to the microsurface's self-shadowing:

$$G_2(\omega_o, \omega_i) = \frac{1}{1 + \Lambda(\omega_o) + \Lambda(\omega_i)} \text{ with } \Lambda(\omega) = \frac{-1 + \sqrt{1 + \frac{\alpha_x^2\,x^2 + \alpha_y^2\,y^2}{z^2}}}{2}. \tag{2.12}$$

Bidirectional Reflectance Distribution Function (BRDF). The cosine-weighted GGX BRDF is defined as:

$$f_r(\omega_o, \omega_i)\,\cos\theta_i = \frac{F(\omega_o, \omega_h)\,D_{\text{ggx}}(\omega_h)\,G_2(\omega_o, \omega_i)}{4\,\cos\theta_o}, \tag{2.13}$$

where $\omega_h = \frac{\omega_o + \omega_i}{\|\omega_o + \omega_i\|}$ is the half vector and F is a Fresnel term. In the following, we do as Hill et al. [7] and always consider $F = 1$, since it can be reintroduced after the LTC approximation via a separate table. Eq. 2.13 is used in the fitting approach of Heitz et al. [8]. We use this formula to make reference comparisons, but not in the implementation of our method.

Sampling. Sampling the Visible Normals Distribution Function (VNDF) [9] produces an approximate sampling of the cosine-weighted BRDF. The remaining weight of the samples is $\frac{1+\Lambda(\omega_o)}{1+\Lambda(\omega_o)+\Lambda(\omega_i)}$. In Section 4.3, we use the rejection-sampling Alg. 1 to produce samples from the density that are perfectly proportional to the GGX cosine-weighted BRDF.

ALGORITHM 1: Sampling the cosine-weighted GGX BRDF.

while *true* **do**
  sample ω_h from the GGX VNDF `/* Heitz's procedure [9] */`
  $\omega_i = \text{reflect}\,(\omega_o, \omega_h)$
  $U \leftarrow \text{rand}()$ `/* Uniform random number in [0, 1) */`
  if $U < \frac{1+\Lambda(\omega_o)}{1+\Lambda(\omega_o)+\Lambda(\omega_i)}$ **then**
    return ω_i

2.4 Stochastic solutions for Light Transport

So far, we have described integral equations that compute the radiance at a point on the scene. These integrals admit no closed form solution and require knowledge of the scene. In this section, we will look at how to evaluate these integrals using stochastic numerical techniques.

Specifically, we will discuss Monte Carlo (MC) integration, which is a stochastic method that iteratively computes the solution to an integral. It does so by *sampling* the integrand in each iteration, and can be shown to compute the exact answer in the limit of infinite iterations (or samples).

Given an integral $I = \int f(x)dx$, a N-sample Monte Carlo (MC) estimator takes the following form:

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}, \tag{2.14}$$

where X_i is a random variable sampled from the probaility density function (PDF) $p(X_i)$ i.e. $X_i \sim p(X_i)$. MC estimators give the right answer *on average*, that is $E[\langle I \rangle] = \int f(x)dx$.

2.4.1 Monte Carlo Path Tracing

Applying MC estimation to the path integral in Eq. 2.3 gives us the following MC estimator:

$$L_o \approx \langle L_o \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\bar{\mathbf{x}}^i)}{p(\bar{\mathbf{x}}^i)}, \tag{2.15}$$

where $\bar{\mathbf{x}}^i$ is a randomly generated light path sampled from $\Omega_k \in \{\Omega_1, \Omega_2, ..., \Omega_\infty\}$ with probability $p(\bar{\mathbf{x}}^i)$. In practice, path tracing recursively builds this path by sampling new vertices starting from $\mathbf{x}_0$, until a light source is reached.

2.4.2 Monte Carlo Direct Lighting

Let's now apply MC estimation to the direct lighting integral in Eq. 2.7:

$$L_o(\mathbf{x}, \omega_o) \approx \langle L(x, \omega_o) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(\mathbf{x}, \omega_o, \omega_i) L_{d(j)}(\mathbf{x}, \omega_i) V(\mathbf{x}, \omega_i) |n_\mathbf{x} \cdot \omega_i|}{p_L(j) p_\omega(\omega_i)}. \tag{2.16}$$

For each term in the MC estimate, a single light source is sampled from p_L as $j \sim p_L(j)$ and a direction is sampled from p_ω as $\omega_i \sim p_\omega(\omega_i)$. Each sample also evaluates the visibility function V, which in turn requires a ray to be traced in the scene.

2.4.3 Unbiased and Biased Monte Carlo estimators

The bias of a MC estimator $\langle I \rangle$ is defined as the quantity:

$$B = E\left[\langle I \rangle\right] - I, \tag{2.17}$$

where $I = \int f(x)dx$. An estimator is unbiased if its expected value is equal to the integral it estimates, and is biased if not, where the bias is given by B.

It is not uncommon to strive for estimators that have zero bias, since they are guaranteed to render according to the laws of physics. However, introducing a small but known bias can often result in high performance gains while still retaining plausible photorealism. The analytic solutions that we work with and propose in Chapters 3, 4 and 5 are biased, but this bias is a known finite quantity. Furthermore, these solutions produce plausible renderings that closely match the ground truth.

In Chapter 6, we also explore *controllable* bias when using neural approximations for path tracing.

2.5 Semi and Fully Analytic solutions for Direct Lighting

In the above sections, we looked at stochastic techniques in detail. In this and the following sub-sections, we will look at analytic methods - both semi and fully analytic - to solve for the direct light transport (Equations 2.7, 2.8, 2.9). Simplistic light sources like point and directional admit exact closed form expressions and we will discuss the point light case in Sect. 2.5.1. For the more general area lights, an exact solution exists only for the irradiance which we discuss in Sect. 2.5.2. We will then discuss Linearly Transformed Cosines (LTCs) and how they achieve semi-analytic direct lighting for non-diffuse surfaces, followed by a discussion on the assumptions of LTCs. These assumptions form the problems that we tackle in Chapters 3, 4 and 5 of this thesis.

2.5.1 Analytic solution for Point Lights

A point light is an infinitesimally small point in space that subtends an infinitesimal solid angle at $\mathbf{x}$ (Fig. 2.2 (a)). Mathematically, a point light located at $\mathbf{p}_j$ is defined by a Dirac Delta term $\delta(\mathbf{p} - \mathbf{p}_j)$. This Dirac Delta term is over *differential area* whereas Eq. 2.9 is over differential solid angle.

The first task is to then convert Eq. 2.9 over differential area dP of a point light by applying change of variables with the corresponding Jacobian so we can apply the Dirac Delta:

$$L_o(\mathbf{x}, \omega_o) = \int_{dP} f_r(\mathbf{x}, \omega_o, \omega_i(\mathbf{p})) L_e(\mathbf{x}, L_i(\mathbf{p})) V(\mathbf{x}, \omega_i(\mathbf{p})) |n_\mathbf{x} \cdot \omega_i(\mathbf{p})| \delta(\mathbf{p} - \mathbf{p}_j) \frac{d\mathbf{p}}{r^2}, \tag{2.18}$$

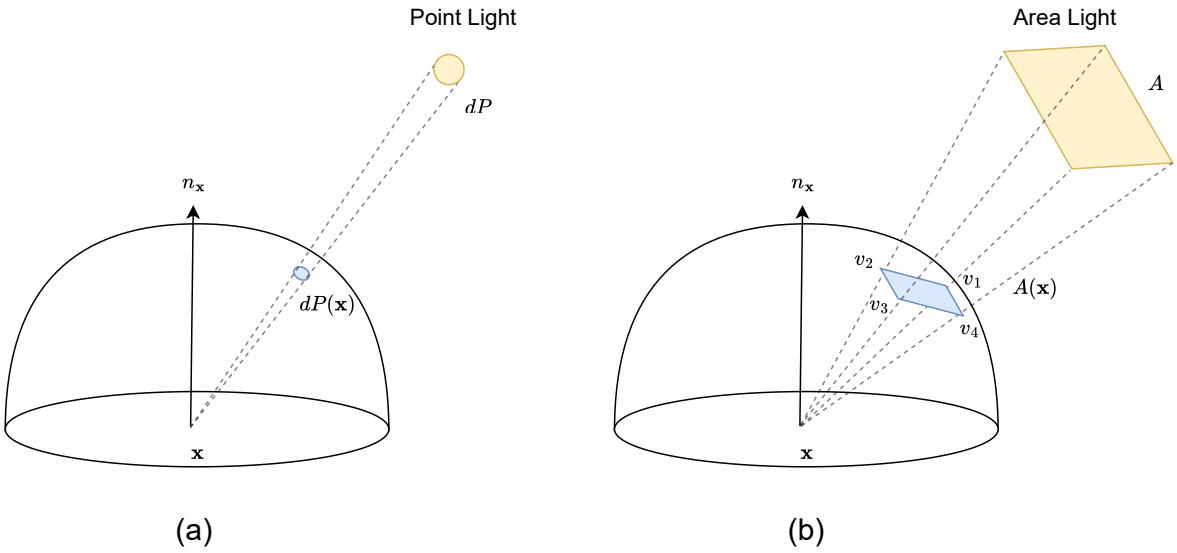


Figure 2.2 (a) Infinitesimally small solid angle $dP(\mathbf{x})$ subtended at $\mathbf{x}$ by a point light dP. (b) An area light A subtends a solid angle $A(x)$ at $\mathbf{x}$. $A(x)$ is described by a spherical polygon with vertices v_1, v_2, v_3, v_4.

where $\omega_i(\mathbf{p}) = \frac{\mathbf{p}-\mathbf{x}}{||\mathbf{p}-\mathbf{x}||}$ and $r = ||\mathbf{p}-\mathbf{x}||$. Using the property of Dirac Delta $\int f(x)\delta(x-x')dx = f(x')$:

$$L_o(\mathbf{x}, \omega_o) = \frac{f_r(\mathbf{x}, \omega_o, \omega_i(\mathbf{p}))L_e(\mathbf{x}, L_i(\mathbf{p}))V(\mathbf{x}, \omega_i(\mathbf{p}))|n_{\mathbf{x}} \cdot \omega_i(\mathbf{p})|}{r^2}. \tag{2.19}$$

In effect, the radiance of a point light is computed by evaluating the integrand for a single direction where the light is located, obtaining it's visibility by tracing a single ray and dividing by the squared distance to that light. The above derivation thus gives a fully analytic solution for computing the radiance from a point light.

2.5.2 Analytic solution for the Irradiance from an Area Light

Given an area light A that is diffuse and spatially constant with $A(\mathbf{x})$ being it's solid angle subtended at $\mathbf{x}$, the irradiance due to it is given by:

$$E(\mathbf{x}; A(\mathbf{x})) = L_d(\mathbf{x}) \int_{A(\mathbf{x})} |n_{\mathbf{x}} \cdot \omega_i| d\omega_i. \tag{2.20}$$

We can define $A(\mathbf{x})$ in terms of unit vectors $v_1, ..v_n$ that each represent the vertices of the spherical polygon describing the solid angle $A(\mathbf{x}) = \{v_1, v_2, ...v_n\}$ (Fig. 2.2 (b)). Given these, an analytic expression for the irradiance integral in Eq. 2.20 can be derived in terms of the solid angle vertices [10–12]:

$$E(\mathbf{x}; A(\mathbf{x})) = L_d(\mathbf{x}) \frac{1}{2\pi} \sum_{i=1}^{n} \arccos(v_i \cdot v_j) \left(\frac{v_i \times v_j}{||v_i \times v_j||} \cdot [0, 0, 1] \right), \tag{2.21}$$

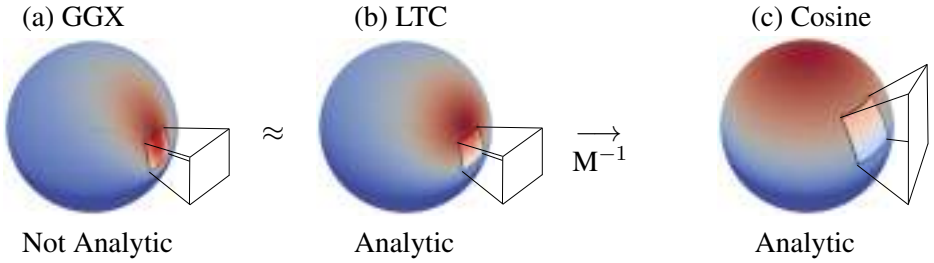


Figure 2.3 Visualization of the BRDF response and the projection of a polygonal area light on a unit sphere centered around the shading point $\mathbf{x}$. (a) A GGX lobe cannot be analytically integrated over the spherical domain covered by the area light. (b) An LTC represented by a matrix M provides a good approximation to the GGX lobe, and the integral equals (c) which is the analytic integral of a cosine lobe over the light transformed by M^{-1}.

where $j = (i+1)\%n$. This equation gives an exact solution to Eq. 2.20 - the bias is zero.

2.5.3 Linearly Transformed Cosines for Semi-Analytic Area Lighting

Consider the direct lighting Eq. 2.8 for a diffuse and spatially constant area light A, with $A(\mathbf{x})$ denoting it's subtended solid angle. As above, we will denote the solid angle with unit vectors as vertices: $A(\mathbf{x}) = \{v_1, v_2, ...v_n\}$.

Also ignoring the visibility function, we can rewrite Eq. 2.8 as:

$$L_o(\mathbf{x}, \omega_o) = L_d(\mathbf{x}) \int_{A(\mathbf{x})} f_r(\mathbf{x}, \omega_o, \omega_i) |n_\mathbf{x} \cdot \omega_i| d\omega_i \tag{2.22}$$

The above integral is of a cosine-weighted (since $n_\mathbf{x} \cdot \omega_i = \cos\theta$) BRDF over $A(\mathbf{x})$. Heitz et al. 2016 [8] showed that given a linear transformation matrix M that transforms the direction vectors of cosine to the direction vectors of the a cosine-weighted BRDF ρ, L_o in Eq. 2.22 can be analytically approximated as:

$$L_o(\mathbf{x}, \omega_o) \approx L_d(\mathbf{x}) \int_{A_o(\mathbf{x})} |n_\mathbf{x} \cdot \omega_t| d\omega_t = E(\mathbf{x}; A_o(\mathbf{x})), \tag{2.23}$$

where E is the analytic expression from Eq. 2.21 and $A_o(\mathbf{x}) = \mathrm{M}^{-1} A(\mathbf{x})$ is the transformed solid angle, which can be computed as:

$$A_o(\mathbf{x}) = \{\mathrm{M}^{-1} v_1, \mathrm{M}^{-1} v_2, ...\mathrm{M}^{-1} v_n\}. \tag{2.24}$$

We also define $\omega_t = \frac{\mathrm{M}^{-1}\omega_i}{||\mathrm{M}^{-1}\omega_i||}$ and refer to M as the LTC matrix. M depends only on the shape of the BRDF f_r and can thus be precomputed and stored in a table for a fixed number of hyperparameters of the BRDF that control it's shape. Heitz et al. 2016 [8] use the isotropic microfacet GGX BRDF, for which these hyperparameters are the viewing direction ω_o and the roughness α. Thus, Heitz et al. 2016

precompute a 2D table of LTC matrices M and fetch it during rendering, based on the hyperparameters at the shading point.

There are two points to note here: (1) LTCs only approximate the true integral (Eq. 2.22, Eq. 2.23) and (2) the LTC formulation does not consider visibility, hence no shadows are rendered. Rendering shadows requires knowledge of the scene due to the visibility function V, which is hard to compute analytically.

A followup work [13] injects visibility back in to LTCs with the following formulation:

$$L_o(\mathbf{x},\omega_o) \approx \underbrace{\int_{A(\mathbf{x})} f_r(\mathbf{x},\omega_o,\omega_i) L_d(\mathbf{x}) |n_\mathbf{x}\cdot\omega_i| d\omega_i}_{=E(\mathbf{x};A_o(\mathbf{x}))\ \text{(Analytic)}} \cdot \underbrace{\frac{\int_{A(\mathbf{x})} f_r(\mathbf{x},\omega_o,\omega_i) L_d(\mathbf{x}) V(\mathbf{x},\omega_i) |n_\mathbf{x}\cdot\omega_i| d\omega_i}{\int_{A(\mathbf{x})} f_r(\mathbf{x},\omega_o,\omega_i) L_d(\mathbf{x}) |n_\mathbf{x}\cdot\omega_i| d\omega_i}}_{\text{(Estimated with Monte Carlo)}}. \tag{2.25}$$

Both terms in the fraction on the right are estimated with Monte Carlo and we refer to them as a *secondary MC estimator* in the context of LTCs. Intuitively, if LTCs were exact, the term on the left and the denominater would exactly cancel and we would get back the original direct lighting equation.

Fig. 2.3 gives an overview of how the visibility free direct lighting equation for an area light is approximated with LTCs. The LTC matrix M approximates the true BRDF ρ, the advantage being that this approximation can be analytically integrated using Eq. 2.21 by transforming the spherical polygon $A(x)$ by M^{-1} to the cosine space.

Throughout this text, we will refer the formulation of injecting visibility in LTCs leading to Eq. 2.25 as semi-analytic LTCs. The semi-analytic nomenclature stems from the fact that the method makes use of both analytic and stochastic evaluations.

2.6 Linearly Transformed Cosines

We built up to LTCs and their formulation ending in Sect. 2.5.3. In this section, we will discuss their properties and end with a discussion on the assumptions of the LTC method.

2.6.1 Properties of Linearly Transformed Cosines

We now review properties of Linearly Transformed Cosines introduced by Heitz et al. [8] and illustrated in Fig. 2.3-(b, c):

Definition. An LTC is defined as a matrix M that maps a clamped cosine distribution D_o to a spherical distribution defined as

$$D(\omega) = D_o(\omega_o)\frac{\partial\omega_o}{\partial\omega} = D_o\left(\frac{\mathrm{M}^{-1}\omega}{||\mathrm{M}^{-1}\omega||}\right)\frac{|\mathrm{M}^{-1}|}{||\mathrm{M}^{-1}\omega||^3}. \tag{2.26}$$

This equation is used by Heitz et al. [8] in their fitting procedure to precompute the look-up table of Eq. 4.1. We use this formula to make a proof in Section 4.4, but not in the implementation of our method.

Area-light integration. The integral of an LTC D over the spherical domain $A(\mathbf{x})$ covered by an area light is the integral of the clamped cosine distribution D_o over the spherical domain $A_o(\mathbf{x})$ covered by the area light linearly transformed by M^{-1}:

$$\int_{A(\mathbf{x})} D(\omega)\,\mathrm{d}\omega = \int_{A_o(\mathbf{x})} D_o(\omega')\,\mathrm{d}\omega'. \tag{2.27}$$

Like Heitz et al., we use this property at run time in the fragment shader to evaluate the integral of an LTC over an area light. The integration procedure depends on the shape of the light [8, 14, 15]. Note that our contribution relates to how the matrix M is obtained, which is independent of how the integration is computed.

Sampling. An LTC can be sampled by generating samples ω' from the clamped cosine distribution and transforming them with the LTC matrix M, as shown in Alg. 2. We use this algorithm in our fitting procedure, which is introduced in Section 4.3.

ALGORITHM 2: Sampling an LTC.

1 sample ω' from a clamped cosine
2 $\omega = \frac{\mathrm{M}\omega'}{\|\mathrm{M}\omega'\|}$
3 **return** ω

2.6.2 Assumptions of the Linearly Transformed Cosines method

In the previous sections, we have looked at estimating direct lighting both stochastically with Monte Carlo and semi-analytically with the LTC method. We are now ready to list down the exact assumptions and limitations of the LTC method, keeping in mind that we are working with direct lighting from area lights.

- **Assumption 0.** This assumption restricts the area light to be spatially constant and diffuse (directionally constant). This means L_d can be taken out of the integral as done in Eq. 2.22.
- **Assumption 1.** This assumption restricts L_d to originate from a polygonal area light. This means that the light is restricted to be a 2D flat surface described by a polygon, which in turn subtends a solid angle which is described by a spherical polygon.
- **Assumption 2.** The next assumption is to ignore the visibility function V, as done in Eq. 2.22. In effect, we set $V = 1$ over the entire integration domain.
- **Assumption 3.** This assumption restricts the BRDF f_r to be isotropic. Specifically, as discussed in Sect 2.5.3, the BRDF is a isotropic microfacet GGX. Due to this assumption, the resulting

precomptation of the LTC matrix table becomes simpler and also allows the table to be two dimensional.

- **Assumption 4.** The final assumption assumes that the scene has no participating media, since the LTC method is based on the direct lighting equation (Eq. 2.7 and 2.22), which themselves are simplified forms of the volume rendering equation.

2.7 Neural approximations of path tracing

In the final chapter (Chapter 6), we consider full path tracing as described by equations 2.2 and 2.3. Deriving analytic solutions for these equations is considerably more challenging, primarily due to their recursive (high-dimensional) nature.

It is interesting to consider approximating a part or whole of this recursion with neural networks. This was first demonstrated by Neural Radiance Caching (NRC) [3], by using a small Multi-Layered Perceptron (MLP) to approximate L. One of the main contributions of NRC was a fully-fused implementation of this MLP, which allowed considerable ($5\times$ to $10\times$) faster training and inference. With such efficiency, they could embed training and inference of this MLP within the rendering loop. During rendering, only short paths are traced and the radiance at the last vertex of these short paths was approximated with the MLP. To achieve online training, they randomly choose a small percentage of the total paths and train on the radiance at each path vertex. This online training is extremely fast thanks to their use of fully-fused MLPs. NRC achieves faster convergence than path tracing, at the cost of some bias from the network.

2.7.1 Assumptions of Neural Radiance Caching

For Chapter 6 of this thesis, we are interested in accelerating the rendering human hair which exhibits very deep recursions due to long path lengths. We refer to approximating such deep recursions with neural networks as *neural approximations for higher order path tracing*. Directly applying NRC to such cases fails due to the following assumptions:

- **Assumption 1.** NRC assumes that learning on the radiance at each path vertex will suffice for all scene types. This strategy is useful for surface (NRC's target application) with energy quickly degrading deeper in the path, and results in more training data for the same number of paths. However, for very deep recursion, this strategy leads to averaging in the network's output, hence producing inaccurate renderings.

- **Assumption 2.** NRC assumes that the entire scene's radiance (Eq. 2.3) can be learnt by the small MLP. However, if this radiance signal has high frequency (which is the case in hair), the small MLP is unable to represent it well leading to artifacts in the final rendering.

2.8 Contributions of this thesis

We can now list the contributions of this thesis, which are essentially relaxing the assumptions listed in Sect. 2.6.2 and Sect. 2.7.1.

2.8.1 Relaxing assumptions of Linearly Transformed Cosines

2.8.1.1 Relaxing Assumption 1 and 2

We first extend the LTC method [8] to light sources of arbitrary 3D shapes. This is done with the observation that the solid angle of a convex 3D shape can be computed by considering its silhouette from $\mathbf{x}$. The vertices that make up the silhouette can be projected to the unit sphere to obtain a spherical polygon describing the light's solid angle. Non-convex lights can be handled by decomposing them into convex sub-parts. This strategy allows the light to be arbitrarily shaped within the LTC framework, relaxing assumption 1.

Next, we look at how to modify this spherical polygon to consider visibility. To this end, we compute silhouette edges and corresponding spherical polygons for all potential occluders. Taking a set difference between the occluder and light polygons gives the visible portion of the light. Finally, we apply the LTC matrix M to this polygon to analytically compute its contribution. Since this procedure directly computes a solid angle that represents the visible portion, it relaxes assumption 2 described above.

Chapter 3 describes our method in detail, along with associated algorithms and evaluations.

2.8.1.2 Relaxing Assumption 3

We discussed in Sect. 2.5.3 that Heitz et al. 2016 [8] use a fitting procedure to precompute LTC matrices for a given isotropic BRDF ρ and store them in a 2D table. The difficulty is that this fitting approach cannot be simply extended to the more general anisotropic BRDFs to get rid of assumption 3. In the isotropic case, the full dimensionality of LTCs is not used and this avoids several problems that arise in the anisotropic case.

Our analysis shows that using anisotropic GGX with LTCs requires *robust fitting*, *well-defined interpolation*, *valid symmetries* and *accurate storage*.

We achieve robust fitting by using the Sliced Wasserstein (SW) loss instead of a loss based on the magnitude of the BRDF as done by Heitz et al. 2016 [8]. The SW loss is based on sample distributions which is the reason for it's robustness. A drawback of such a loss is that interpolation is not guaranteed, which we fix by *aligning* the LTC matrices by removing unnecessary rotations and flips. Both of these allow to dramatically reduce the precomputed table resolution. We exploit symmetries in GGX to further reduce the resolution to $8 \times 8 \times 8 \times 8$. Our method makes careful design choices resulting from new insights into the mathematical properties of LTCs.

The final outcome of our method, described in more detail in Chapter 4, is a $8 \times 8 \times 8 \times 8$ 4D look-up table that yields a plausible and artifact-free LTC approximation to anisotropic GGX and is memory efficient. This table can be used for semi-analytic direct lighting of anisotropic surfaces from area lights.

2.8.1.3 Relaxing Assumption 4

To relax the fourth assumption of the LTC method, we start by extending the theory of Linearly Transformed Spherical Distributions (LTSDs) to work with phase functions. We show that this is non-trivial, and arrive at a solution with in-depth analysis. This enables us to analytically compute in-scattered radiance, which we build on to semi-analytically render unshadowed single scattering. We ground our derivations and formulations on the volume rendering equation 2.1 which paves the way for realistic renderings despite the biased nature of our method. We also formulate ratio estimators for the VRE to work in conjunction with our formulation, enabling the rendering of shadows.

Our contributions, described in Chapter 5, enable semi-analytic rendering of single scattering with area lights.

2.8.2 Relaxing assumptions of Neural Radiance Caching

Finally, we describe a method in Chapter 6 that relaxes the two assumptions of Neural Radiance Caching (NRC) [3] mentioned in Sect. 2.7.1, enabling efficient rendering of human hair.

Our method infers the higher order scattering in hair in constant time within the path tracing framework, while achieving better computational efficiency. We makes no assumptions about the scene and our method provides control over the renderer's bias & speedup. We achieve this by training a small multilayer perceptron (MLP) to learn the higher-order radiance online, while rendering progresses. This is similar to the NRC method, however, we identify and solve key issues that enable its application for accurate hair rendering.

Chapter 3

Fast Analytic Soft Shadows from Area Lights

"Somethings can only be seen in the shadows."
- Carlos Ruiz Zafron

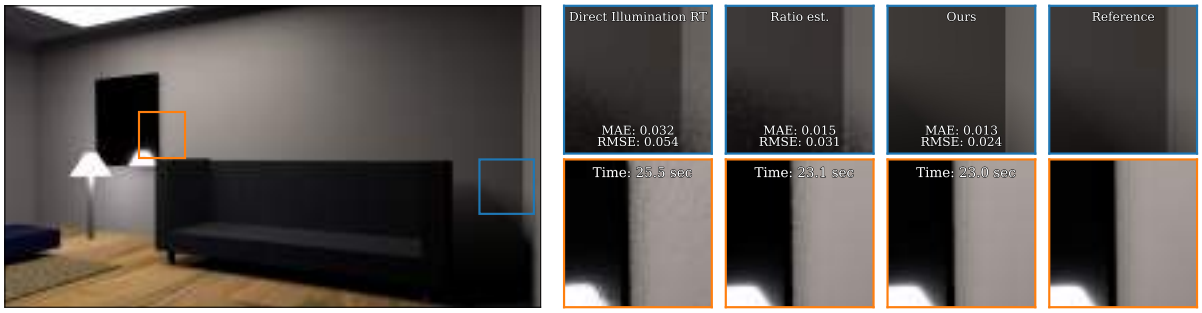


Figure 3.1 Our method computes analytic shading and soft shadows for physically based BRDFs from arbitrary area lights. The scene shown here has two area lights (Quad at the top, lamp at the left). Our result is completely noise free and has lesser MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) as compared to direct illumination ray tracing and semi-analytic LTCs. The reference is rendered for a large number of samples per pixel ($\sim$ 400 spp).

This chapter presents a method to analytically compute shading and soft shadows for physically based BRDFs from arbitrary area lights, using LTCs. The method presented here relaxes assumptions 1 and 2 presented in Chapter 2, Sect. 2.6.2.

To analytically compute shadows, we need to compute the visible solid angle for an area light. This is done by performing a set difference operation on spherical polygons of light sources and occluders. This polygon can be directly integrated over to compute the occluded irradiance using Linearly Transformed Cosines (LTCs) [8]. Since we always integrate over polygonal domains that are visible, our method can compute soft shadows. Finding the visible spherical polygon of an arbitrary light source is a non-trivial operation. We consider individual triangles of lights and occluders to find visible regions. Since this is computationally expensive, we further propose an optimization for the case of convex lights and occluders.

In summary, to relax assumptions 1 and 2 presented in Ch. 2, Sect. 2.6.2, we propose the following: (a) A structured approach to analytically compute soft shadows from spherical projections of lights and occluders with any 3D shape and (b) A method to efficiently obtain spherical polygons of arbitrary convex geometry.

3.1 Contributions in the context of previous work

Polygonal Area Light Shading. Baum et al. [11] derived an analytic formula for integrating the clamped-cosine distribution over a spherical polygon. Further extensions for generality and accuracy were done by [8, 16–18]. In this work, we extend the method of Heitz et al. 2016 [8] in two directions: (1) Analytically compute soft shadows from general 3D lights and occluders, and (2) Efficient computation for the case of convex meshes.

PRT with Polygonal Area lights. Precomputed Radiance Transfer (PRT) has been extended to incororate shading from area light sources [19–21]. PRT however assumes a static scene and requires expensive precomputation. Furtheremore, PRT does not work well for highly specular materials. Our method in contrast requires no precomputation and produces soft shadows on the fly and can handle glossy and specular materials correctly.

On the fly soft shadows. We compute high quality soft shadows on-the-fly & analytically, not stochastically or based on precomputation as done in previous approaches [13, 18]. Our work was done concurrently to the work of Zhou et al. [22]. They present a similar approach of using polygon operations for visibility computation. They demonstrate that such exact visibility computations along with LTC can not only be used to obtain analytic and noise free soft-shadows but also noise-free gradients for differentiable rendering. We note that their approach works on individual triangles (similar to our per-triangle approach, Sect. 3.3.4). In this work, in addition to the per-triangle approach, we also present an efficient algorithm for convex meshes and demonstrate a $2\times$ speedup in rendering.

3.2 Method

Recall the analytic approximation of direct lighting (ignoring visibility) used by LTCs (Eq. 2.23):

$$\begin{aligned} L_o(\mathbf{x}, \omega_o) &= L_d(\mathbf{x}) \int_{A(\mathbf{x})} f_r(\mathbf{x}, \omega_o, \omega_i) |n_{\mathbf{x}} \cdot \omega_i| d\omega_i \\ &\approx E(\mathbf{x}; A_o(\mathbf{x})), \end{aligned} \tag{3.1}$$

where E is an analytic expression, $A_o(\mathbf{x}) = \mathrm{M}^{-1} A(\mathbf{x})$ and $A(\mathbf{x})$ is the solid angle subtended at $\mathbf{x}$ by an area light A.

First, consider generalizing the light source to an arbitrary 3D shape given by a polygonal mesh. We observe that the radiance L_o at a point $\mathbf{x}$ due to such a mesh can be computed by replacing $A(\mathbf{x})$ with the spherical polygon $A'(\mathbf{x})$ which is the projection of the light source object on the unit sphere

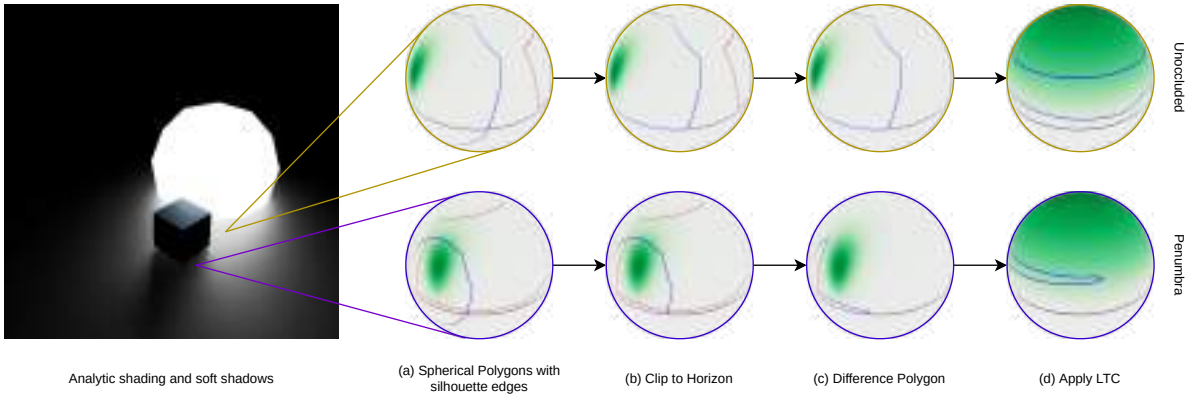


Figure 3.2 High-level steps of our algorithm to compute shading and soft shadows from area lights. Unit spheres at an unoccluded shading point (yellow) and at a shading point in the penumbra (purple) are visualized with the corresponding spherical polygons of the light source (icosphere, marked as blue polygon) and occluder (cube, marked as red polygon). (a) Obtain spherical polygons from silhouette edges, (b) Clip the spherical polygons to the horizon, (c) Compute difference of the light source and the occluder polygon, (d) Apply Linearly Transformed Cosines (LTC) [8], clip the result to the horizon and compute shading with analytic formula [11].

centered at $\mathbf{x}$. The problem thus reduces to finding the projected area $A'(\mathbf{x})$ represented as a spherical polygon for a given arbitrary light source. Our second observation concerns the impact on L_o of an arbitrary 3D mesh occluder O, which also has a corresponding projected spherical polygon $O'(\mathbf{x})$. The light source is occluded in regions where $O'(\mathbf{x})$ overlaps with the light's projection $A'(\mathbf{x})$. Thus, $\hat{A}(\mathbf{x}) = A'(\mathbf{x}) - O'(\mathbf{x})$ defines the projected area of the visible portion of the light source. Transforming $\hat{A}(\mathbf{x})$ with a LTC matrix M gets $\hat{A}_o(\mathbf{x})$ to use with the analytic formula E, giving the correct radiance at the shading point x due to the light source.

Directly obtaining the projected area for a non-convex polygonal mesh is non-trivial. We can instead treat individual triangles of the mesh as its own separate object and project them to the unit sphere. This can be done for both light sources and occluders one polygon at a time to obtain $\hat{A}(\mathbf{x})$. This produces correct results but is inefficient if larger objects are used (per-triangle method, Sect. 3.3.4). Estimating the projection of a shape onto the sphere can be efficiently done for convex shapes, achieving more than $2\times$ speedup from the per-triangle method. Therefore, for most of this paper, we assume both light sources and occluders to be convex 3D meshes. Simple non-convex meshes can still be represented by a combination of independent convex parts, for example the chair in Fig. 3.1 is made up of four rectangular cuboids. We discuss the per-triangle method in Sect. 3.3.4.

3.2.1 Overview

We now present an overview of our method for convex area light and convex occluder meshes. Note that the steps outlined below can also be used on non-convex meshes by iterating over individual triangles of the mesh (Sect. 3.3.4). Fig. 3.2 and Alg. 3 show high-level steps of our algorithm. The key observation

ALGORITHM 3: Overview of our algorithm.

Input: $\mathcal{L}$, $\mathcal{B}$, $\mathbf{x}$, $n_{\mathbf{x}}$, f_r: List of light sources $\mathcal{L}$, list of occluders $\mathcal{B}$, shading point $\mathbf{x}$, normal vector $n_{\mathbf{x}}$, BRDF f_r

Output: L_o: Outgoing radiance (Eq. 3.1)

1 $L_o \leftarrow \text{Rgb}(0.0)$ // Init. with black col.
2 $p \leftarrow \text{vec3}(0, 0, 1)$ // Arbitrary vector for plane projection
/* Consider each light source separately */
3 **for** l ***in*** $\mathcal{L}$ **do**
4 l' = SphPolySilhouette(l, $\mathbf{x}$, $n_{\mathbf{x}}$) // Light sph. poly
5 ${l_{clip}}'$ = ClipToHorizon(l')
6 ${l_{xy}}'$ = project(${l_{clip}}'$, p) // Project to plane
7 $\mathcal{B}_{bw}$ = GetBetween($\mathbf{x}$, l, $\mathcal{B}$) // Occluders b/w $\mathbf{x}$ & l
/* Set diff. b/w light and occluders */
8 **for** b ***in*** $\mathcal{B}_{bw}$ **do**
9 b' = SphPolySilhouette(b, $\mathbf{x}$, $n_{\mathbf{x}}$)
10 ${b_{clip}}'$ = ClipToHorizon(b')
11 ${b_{xy}}'$ = project(${b_{clip}}'$, p)
12 ${l_{xy}}'$ = SetDifference(${l_{xy}}'$, ${b_{xy}}'$)
/* Apply LTC and integrate */
13 l' = reproject(${l_{xy}}'$, p) // Project back to sphere
14 l_{ltc} = ApplyLTC(l', f_r)
15 l_{ltc} = ClipToHorizon(l_{ltc})
16 $L_o = L_o +$ AnalyticIntegrate(l_{ltc}, f_r)

is that for a convex 3D light source, the spherical polygon $A(\mathbf{x})$ in Eq. 3.1 can be obtained using its silhouette edges as viewed from $\mathbf{x}$. The silhouette edges can be efficiently computed using front and back facing polygon detection [23]. They are then projected to the unit sphere to obtain a spherical polygon of the silhouette (Fig. 3.2(a) *blue* polygon, Alg. 3 line 4). We then clip the spherical polygon to the horizon (Fig. 3.2(b), Alg. 3 line 5). Further, to obtain a polygon $A(\mathbf{x})$ that represents only the visible region of the light source, we first compute silhouette edges and corresponding spherical polygons for all potential occluders, clip them to the horizon (Fig. 3.2(a) *red* polygon). The clipped light and occluder polygons are then projected to a plane (Alg 3 line 6, lines 8-11). The set difference between the light occluder polygon represents the visible region of the light source from the point $\mathbf{x}$ (Fig. 3.2(c), Alg 3 line 12). We perform this set difference for each occluder polygon. Finally, we reproject the resultant polygon to the unit sphere, apply LTC and clip the result to the horizon, to obtain $A_o(\mathbf{x})$ for analytic evaluation in Eq. 3.1 (Fig. 3.2(d), Alg. 3 lines 13-16). These steps are repeated for each light source in the scene (Alg. 3 line 3).

We elaborate on the following functions in the next sections: *project* and *reproject* to project spherical polygons to a plane and repreoject planar polygons to the unit sphere (Sect. 3.2.2), *SphPolySilhouette* to obtain ordered spherical polygons (Sect. 3.2.3), *ClipToHorizon* to clip spherical polygons to the horizon (Sect. 3.2.4), *GetBetween* to prune objects that do not occlude the light source (Sect. 3.2.5), *SetDifference* to obtain a polygon representing the visible area of the light (Sect. 3.2.6).

3.2.2 Projecting spherical polygons to a plane

Ordering edges and performing set operations on spherical polygons have, to our knowledge, not been explored in the literature and are non-trivial. Our purpose, however, is served by performing the ordering and set operations on a plane to which we project spherical polygons. The modified polygon is then reprojected back to the unit sphere. In the *project* function, the projection is done using an appropriate look-at camera matrix $\mathcal{M}$. The look-at vector needs to be carefully chosen to avoid transformed vertices having negative z-coordinates. We then apply perspective division to obtain a correct planar projection and discard the z-coordinate. We visualize this process in Fig. 3.3(a), for an arbitrary look-at vector p. To reproject polygon vertices to the unit sphere, we first apply the inverse camera matrix $\mathcal{M}^{-1}$, and then normalize the vectors of each polygon vertex.

Projecting a sphere on to a plane has been previously explored, in the context of map projections or sphere mapping. The closest mapping operation to ours is the Gnomonic projection [24]. However, for simplicity and ease of implementation, we opt for the look-at transform with perspective camera projection.

ALGORITHM 4: SphPolySilhouette.

Input: $\mathcal{G}$, $\mathbf{x}$: Convex geometry $\mathcal{G}$, shading point $\mathbf{x}$
Output: $\mathcal{G}_{sil}$: Ordered silhouette edges

$\mathcal{G}'$ = ComputeSilhouette($\mathcal{G}$, $\mathbf{x}$)
$\mathcal{G}' = \mathcal{G}' - x$ `// Shift edges to x as origin`
$\mathcal{G}'$ = LocalShadingFrame($\mathcal{G}'$, $\mathbf{x}$) `// Normal vec aligned with z-axis`
$\mathcal{G}'$ = ToUnitSphere($\mathcal{G}'$) `// Spherical Poly`
c = centroid($\mathcal{G}'$)
$\mathcal{G}_{xy}$ = project($\mathcal{G}'$, c)
`/* Order edges clockwise or anti-clockwise */`
$\mathcal{G}_{ord} \leftarrow$ []
$\mathcal{G}_{ord}$.append($\mathcal{G}_{xy}[0]$)
for e ***in*** $\mathcal{G}_{ord}$ **do**
    $\mathcal{E}$ = [$\{Len(e.v2 - e'.v1), Len(e.v2 - e'.v2)\} \forall e' \in \mathcal{G}_{xy} - \mathcal{G}_{ord}$]
    $next$ = min($\mathcal{E}$)
    $nextEdge$ = argmin($\mathcal{E}$)
    if $next[0] < next[1]$ **then**
        $\mathcal{G}_{ord}$.append($nextEdge$)
    else if $next[0] > next[1]$ **then**
        $\mathcal{G}_{ord}$.append($nextEdge$.reverse)
$\mathcal{G}_{sil}$ = reproject($\mathcal{G}_{ord}$, c)

3.2.3 Spherical Polygons of Convex Shapes

We now describe *SphPolySilhouette* function from Alg. 3 which performs two tasks: (1) Compute the spherical polygon of the silhouette in the local shading frame and (2) order the spherical polygon (clock-

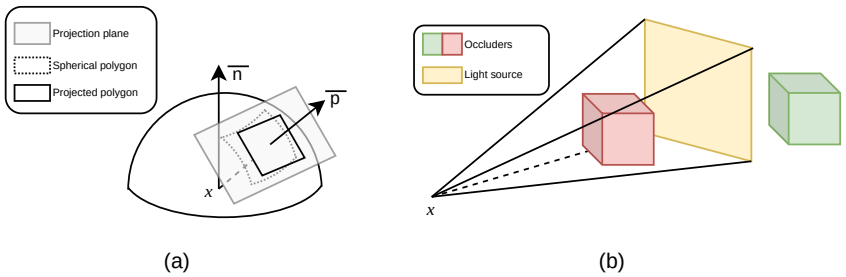


Figure 3.3 (a) Projection of a spherical polygon (dotted) to a camera plane defined by the look-at vector p. The camera origin is at the shading point $\mathbf{x}$. (b) Pruning objects that do not occlude the light source (green box) using a tetrahedral frustum defined by the shading point $\mathbf{x}$ and a plane at the light source (yellow).

wise or anti-clockwise). The second task is important for the correct functioning of the *ClipToHorizon* algorithm, *SetDifference* operation and analytic LTC integration.

Lines 1-4 of Alg. 4 compute and project the silhouette edges to the unit sphere to get a spherical polygon. They also shift the origin to the shading point $\mathbf{x}$ and transform them to the local shading frame. In lines 4-6, we use the centroid of the geometry G as the look-at vector for the *project* function. This projection results in X-Y co-ordinates for all edges, which we order by finding consecutive edges and adding them to a list (lines 7-16). For each edge e, we find another edge which has either of its two vertices closest to the second vertex of e (line 10). Note that since edges may not have vertices in the same direction, we reverse edges based on which vertex follows the second vertex of e (lines 13-16). Finally, we re-project the ordered edges to the unit sphere (line 17).

3.2.4 Clipping to Horizon

Heitz et al. 2016 [8] assumed a four sided spherical polygon, which on clipping results in either three or five edges. We need a more general clipping algorithm since silhouette edge projections could result in an N sided spherical polygon. We present a general algorithm to clip arbitrary spherical polygons to the horizon *ClipToHorizon* in Alg. 5. This algorithm takes as input a spherical polygon with ordered edges and outputs a horizon clipped spherical polygon while preserving the edge order. For each edge that crosses the horizon, we check whether it's vertices are above or below the horizon. If the edge is completely above the horizon (line 6), we add it to the final list (line 7) and if it is completely below (line 8), we ignore it (line 9). For an edge intersecting with the horizon (line 10), we find the intersection point using standard algorithms for intersection of spherical arcs (line 12). Note that we find intersection between the edge arc (defined by its two vertices) and the full horizon arc. We then replace one vertex of this edge with the intersection point. Which vertex gets replaced depends on which one of the two vertices are below the horizon (lines 13-18). In the final step on line 20, we close vertex pairs that

ALGORITHM 5: Clip to Horizon.

Input: $\mathcal{S}$: Spherical polygon $\mathcal{S}$
Output: $\mathcal{S}_{clip}$: Spherical polygon clipped to horizon

$\mathcal{S}_{clip} \leftarrow$ []
$n \leftarrow$ vec3$(0, 0, 1)$ // Normal vec (local shading frame)
/* Iterate over edges of sph. poly */
for *e* ***in*** $\mathcal{S}$ **do**
    dp_1 = Dot$(e.v1, n)$
    dp_2 = Dot$(e.v2, n)$
    if $dp_1 > 0.0$ ***and*** $dp_2 > 0.0$ **then**
        $\mathcal{S}_{clip}$.append(e)
    else if $dp_1 <= 0.0$ ***and*** $dp_2 <= 0.0$ **then**
        continue
    else
        $newEdge \leftarrow$ Edge()
        /* Intersect. of edge & horizon arc */
        i = IntersectHorizon(e)
        if $dp_1 < 0.0$ **then**
            $newEdge.v1$ = i
            $newEdge.v2$ = $e.v2$
        else
            $newEdge.v1 = e.v1$
            $newEdge.v2 = i$
        $\mathcal{S}_{clip}$.append($newEdge$)
$\mathcal{S}_{clip}$ = CloseVertices($\mathcal{S}_{clip}$) // Add edges on horizon

are consecutively on the horizon, which in effect adds missing edges on the horizon. The algorithm preserves the order of edges and results in a spherical polygon clipped to the horizon.

3.2.5 Pruning Non-Occluding Objects

In this section, we describe the *GetBetween* function of Alg 3 to prune out objects that cannot occlude the radiance from the light source. This avoids extra processing for objects that have no contribution to the occluded radiance. As shown in Fig. 3.3(b), we discard objects that are outside the frustum defined by the shading point and the light source. Note that in our implementation, we use frustum culling with spherical bounding boxes and a tetrahedral conic frustum, but any other suitable method could be used for this step.

3.2.6 Set operations on spherical polygons

We apply *SetDifference* operation after projecting spherical polygons to a plane, using the *projection* function defined in Sect. 3.2.2. Note that we use the z-axis as the look-at vector for this projection. The reason for this is that all spherical polygons are clipped to the horizon before projection, and using z-axis as the look-at vector ensures that no vertex has a negative z-coordinate after the look-at camera transform. Choosing any other look-at vector could potentially result in negative z-coordinates for either the light or occluder polygon. Negative z-coordinates result in wrong projections which distort the polygon shape, thus affecting the set operations. After projection, we take the set difference between the light polygon and each occluder polygon, progressively modifying the same light polygon. The result of these operations is a polygon, which represents the region of the light source which is visible.

3.3 Results, Evaluation and Comparisons

The entire approach presented in this paper is implemented as an integrator plugin in PBRT-v3 [1]. We use the publicly available pre-computed LTC matrices M from [8]. Our implementation requires that light sources and occluders are marked appropriately, which is done with a flag in the scene description file. This is necessary since it avoids wasteful computations, for example, the walls in a room scene are not occluders for light sources within the room. We use standard frustum culling to get occluders that lie between the shading point and the light source (Alg. 3 line 5). The frustum is a five sided tetrahedral cone, with the apex being the shading point and the bottom face at the light source (Fig. 3.3 (b)). To compute the set difference operation on polygons, we use the Greiner-Hormann polygon clipping algorithm [25]. The main advantage of this algorithm over other algorithms such as [26] is the ease of implementation, fast run-time and out-of-the-box support for set operations. Note that although we use Greiner-Hormann in our implementation, our method is independent of the choice of the polygon clipping algorithm. We plan to release the full source code and scenes used in this paper.

To study the run-time of our method, we first analyze its behaviour with varying number of light and occluder vertices. We then provide a comparison of our results with direct illumination ray tracing

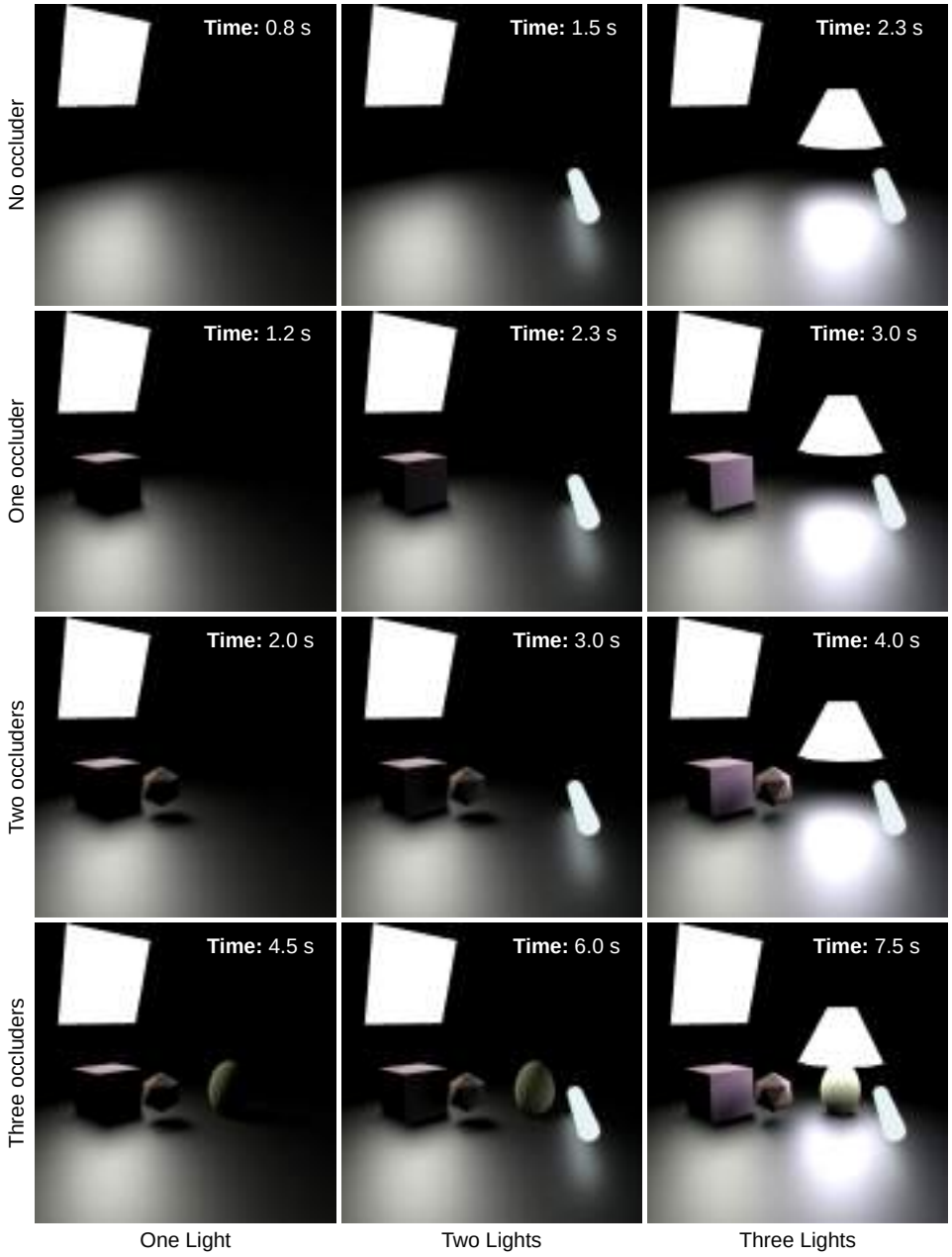
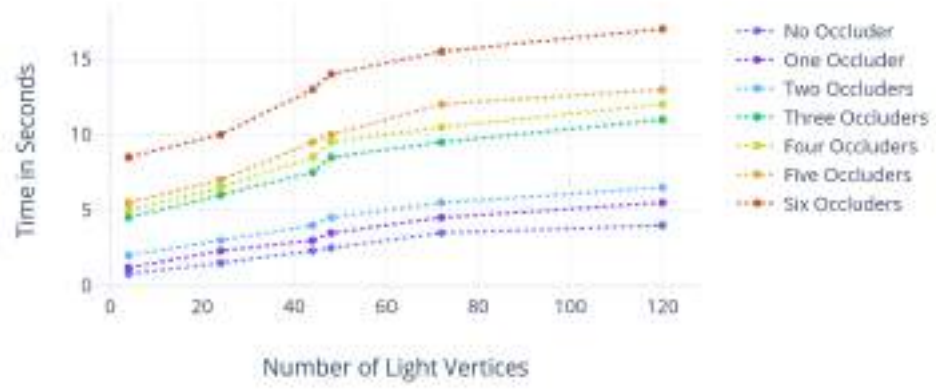
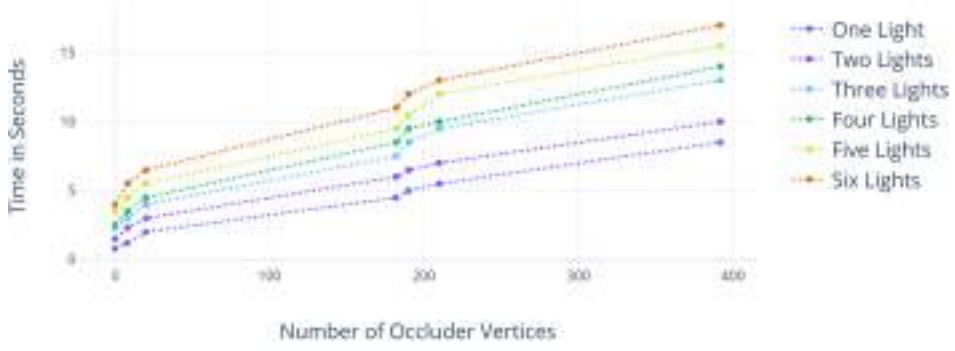


Figure 3.4 Test scenes with varying number and geometric complexity of light sources and occluders. Each image is rendered at a resolution of 1000×1000 using our method. Run-times are reported at the top right. We use three kinds of light sources: Plane (four vertices), Cylinder and Truncated cone (20 vertices each). We similarly use three kinds of occluders:Cube (eight vertices), Icosphere (16 vertices) and Ellipsoid (154 vertices)



(a) Run-time v/s No. of light vertices



(b) Run-time v/s No. of occluder vertices

Figure 3.5 Plot of #vertices vs. runtime for light sources (a) and occluders (b), for scenes like in Fig. 3.4. Our method is roughly linear in the #vertices of light sources and occluders.

and the control variate ratio estimator of Heitz et al. 2018 [13] on four realistic scenes having varying complexity. In addition, we compare our method to a naïve extension of Heitz et al. 2016 [8] for general 3D meshes to highlight the need for silhouette edge computation. Lastly, we compare to a variation of our method that can handle non-convex 3D meshes. All scenes are rendered on a workstation with a AMD Ryzen 5 CPU having eight cores. PBRT correspondingly utilizes all eight cores for rendering.

3.3.1 Run-time Analysis

We first demonstrate results and analyze our method's run-time on simple test scenes, as shown in Fig. 3.4. We vary the number and geometric complexity of light sources along the columns, starting from one to three (first to third column). We use three 3D meshes with varying complexity for light sources: Plane (four vertices), Cylinder and Truncated cone (20 vertices each). We follow the same variation for occluders along the rows, starting from zero to three (first to fourth row). We use three occluders: Cube (eight vertices), Icosphere (16 vertices) and Ellipsoid (154 vertices). Thus, the bottom right scene of Fig. 3.4 has the maximal complexity (three light sources and three occluders). We render 1000×1000 images for each variation. Note that the shadows become softer on addition of new light sources, and in the penumbra region on the ground and on occluders. Our method is able to produce plausible, realistic and accurate soft-shadows in presence of multiple area lights and occluders.

We further plot the run-times for each of the test scenes against the total number of vertices of the light sources and occluders. Fig. 3.5(a) shows the plot of number of light source vertices against the run-time and 3.5(b) shows the plot of number of occluder vertices against the run-time. To obtain more data points for the plot, we duplicate each light source and occluder and place them at a new location in the scene. Both plots show that our method is linear in the number of light source and occluder vertices.

3.3.2 Comparisons

We now show results of our method and compare with direct illumination ray tracing (RT) and the control variate ratio estimator (Ratio est.) of Heitz et al. 2018 [13]. We implement the Heitz et al. 2018 method as an integrator plugin in PBRT. Specifically, we use three integrators which separately compute S_N, U_N and U. We then denoise S_N and U_N separately with a bilateral filter, and combine all three terms. The run-time is a summation of the time taken to render S_N plus the time taken to render U and denoising. Note that we ignore the time taken to render U_N, since it could potentially use the same rays that are used for S_N. We request the reader to refer to their paper for the explanation of these terms.

We use four realistic scenes for this comparison: *Dining Room*, *Living Room*, *Outdoor Bench* and *Table*. We render each scene at a resolution of 1920×1080. Note that we set the number of samples for RT and Ratio est. such that their rendering time is roughly equal to ours. The results and comparisons are shown in Fig. 3.6. The *Table* scene has a circular planar light source with a circular occluder just in front, which our method is correctly able to handle. The quantitative metrics along with roughly equal run-times are shown in the insets. Since our method analytically computes soft shadows, our renderings have no noise or blurring artefacts caused due to denoising. Our method thus also achieves lower MAE

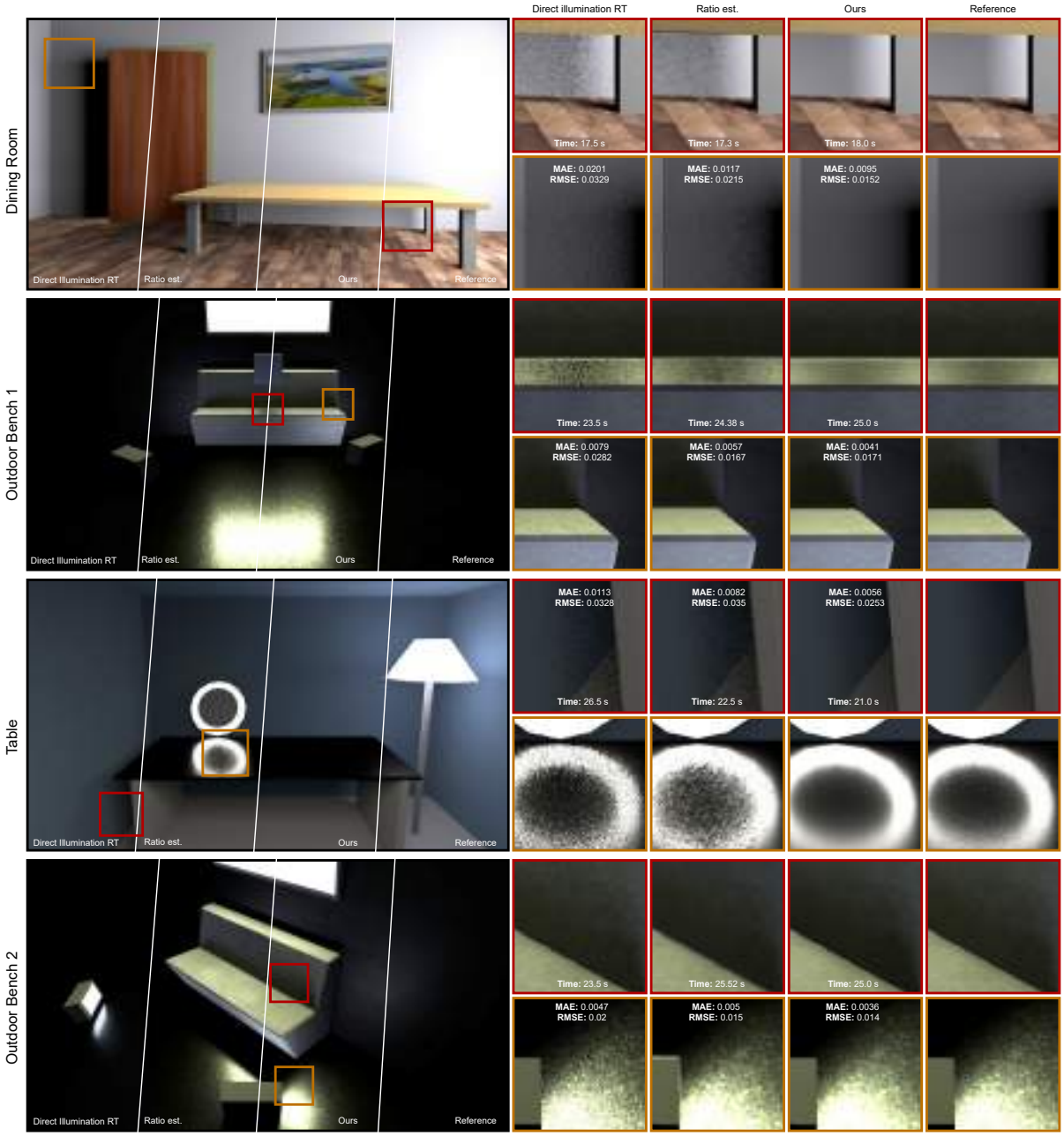


Figure 3.6 Roughly equal time comparison of our method with direct illumination Ray Tracing (RT) and the ratio estimator (Ratio est.) of Heitz et al. 2018 [13]. Run-times and quantitative values of MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) are shown in insets. Each scene is rendered at a resolution of 1920×1080. Our method outperforms RT and and Ratio est. in terms of both quality and quantitative metrics.

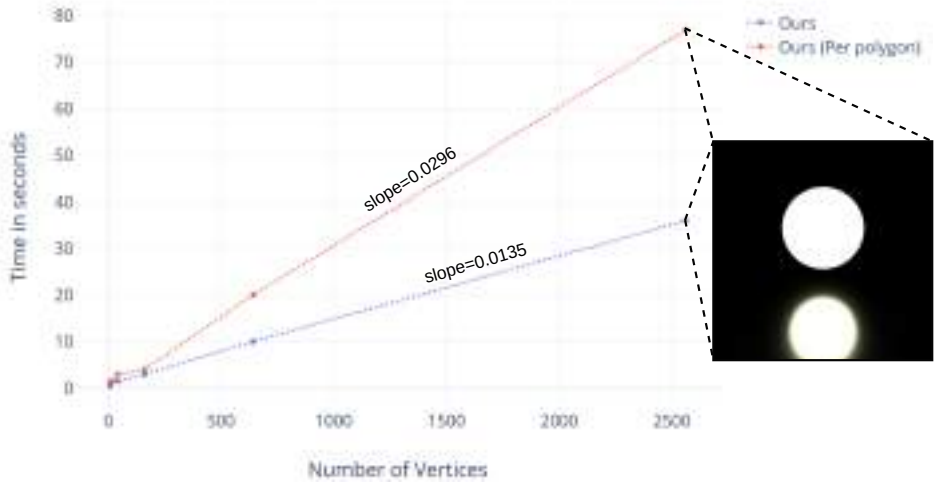


Figure 3.7 Run-time comparison of our method with a naïve extension of Heitz et al. 2016 [8], for a simple scene having only one icosahedral light source, a specular ground plane and no occluders. We progressively sub-divide the light source geometry to plot run-time.

(Mean absolute error) and RMSE (Root mean squared error) as compared to RT and Ratio est. The attached video shows more renderings of these scenes.

3.3.3 Naïve extension of LTCs for arbitrary area lights

In this section we highlight the necessity of computing silhouette edges for obtaining spherical polygons. The method proposed by Heitz et al. 2016 [8] can be naïvely extended to arbitrary emissive meshes, by iterating over individual faces of the geometry. Each face can be treated as an independent polygonal light source for which shading can be obtained with their method. Note that in this case, a per triangle check to determine whether its normal vector points towards the shading point is necessary. We also need to sort each polygon's edges either clock-wise or anti-clockwise for correctness of the LTC integration.

This strategy is however inefficient with a run-time linear in the number of faces of the light source. In comparison, the run-time of our method which obtains one spherical polygon for the light source using silhouette edges is linear in the number of silhouette edges of the light source, which are expected to be small (around $\sqrt{e}$, where e is the total number of edges [27]). Note that our method incurs an additional fixed cost of silhouette edge computation for each shading point. We demonstrate this in Fig. 3.7, with a simple scene having only one icosahedral light source, a specular ground plane and no occluders. We repeatedly subdivide the icosahedral light source to increase the number of vertices and plot the run-time against it. Both methods are linear in the number of vertices, however, the rate of increase of naïve method is more than twice to ours.

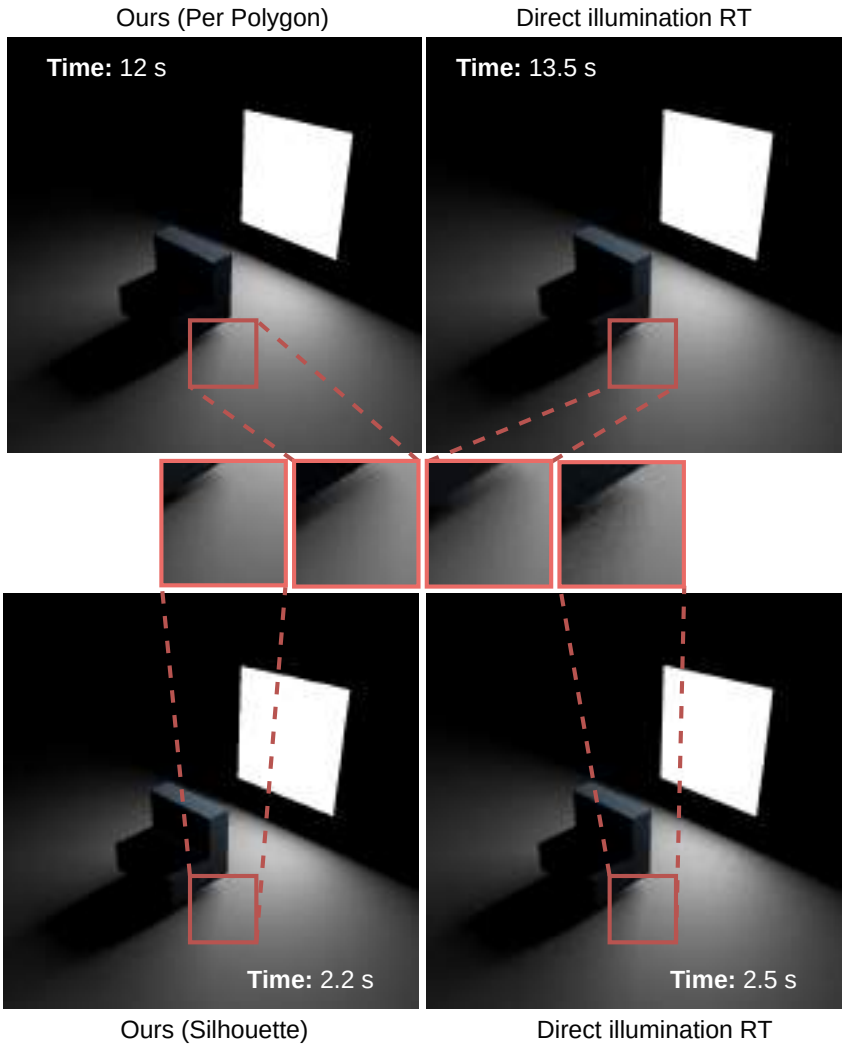


Figure 3.8 Comparison of the per-triangle method (Sect. 3.3.4) with direct illumination GT (top) and our method (silhouette edges) with direct illumination GT, run for equal time in both cases. Note that the per-triangle method exhibits instability due to degeneracies caused by vertices at the same location.

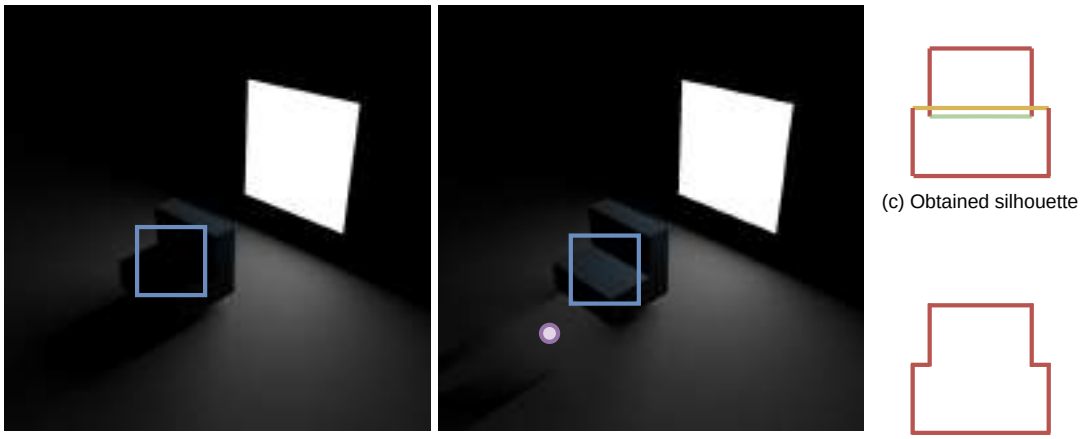


Figure 3.9 (a) Reference image rendered with direct illumination ray tracing. (b) Output of our method with silhouette edges. (c) shows the silhouette edges at a shading point (marked in purple) and (d) shows the ideal/expected silhouette. Due to the definition of silhouette edges, we get extra edges (marked in green and yellow), which results in incorrect outputs of our algorithm and incorrect LTC integration. Also note the missing shadow in the blue insets. Non-convex meshes result in self-occlusions, which need to be projected and handled separately, since direct silhouette edge projection will result in wrong shading.

3.3.4 Variation of our method for non-convex meshes

Fig. 3.9 shows the possible problems that occur when using silhouette edges for non-convex meshes. The silhouette edges are estimated wrongly, resulting in incorrect shading and shadows. To handle simple non-convex meshes, we can decompose them into a set of convex parts, as shown in Fig. 3.6 for the table in the Dining Room scene and bench in the Outdoor Bench scene. For direct handling of arbitrary non-convex meshes, we can modify our approach in Alg. 3, where the list $\mathcal{L}$ will instead be a list containing all triangles of all light sources and $\mathcal{B}$ will be a list containing all triangles of all occluders. Note that in this case, the function *SphPolySilhouette* will only shift the origin and transform the triangle to the local shading frame and exit. We refer to this approach as the *per-triangle* method.

We compare the per-triangle approach with ours and with equal time direct illumination ray tracing. The result is shown in Fig. 3.8, for a scene having a chair like object and a four sided polygonal light. Note that for the bottom left rendering of our method, the object is decomposed into two convex shapes (the bottom portion and the top portion). The per-triangle method takes much more time than our method, for which equal time direct illumination (shown in the left column) is almost noise free. Furthermore, we would like to note that the per-triangle approach in unstable, mainly due to degeneracies arising from vertices at the same location (for example two triangles sharing a common edge). This ultimately leads to a failure case for the *setDifference* and other geometric operations. Thus, there is a trade-off between using our per-triangle approach with arbitrary meshes, at the cost of high run-time versus our approach for convex meshes at a lower run-time.

3.4 Discussion and Conclusions

We presented a method to analytically compute shading and soft shadows from light sources with general 3D shapes. We futher demonstrated several examples of efficiently rendering high-quality scenes with realistic soft-shadows when light source and occluders are convex meshes. One limitation of our method is its inability to directly handle non-convex shapes, though a simple variation can handle them at much reduced speeds. Another limitation is that our approach has a memory footprint that is not predictable, especially for the *setDifference* and the *clipToHorizon* functions, which poses a challenge for efficient GPU implementation. Finally, since we use LTCs which are itself an approximation to the true BRDF, our result may not exactly match the ground truth rendered with ray-tracing.

An interesting future direction of our work is to obtain proper silhouettes even for non-convex shapes. This would lead to increase in generality of our algorithm at acceptable run-time. Another area of future work is to improve the speed of computation of each step of our algorithm. It is also possible to obtain and exploit reasonable upper bounds on the memory required for each step of our algorithm for an efficient GPU implementation. We would further like to investigate and exploit the continuity of spherical projections between adjacent shading points to improve efficiency. Lastly, we would like to investigate the integration of analytic solutions like ours with methods like [28], which re-use spatial and temporal information to drastically reduce variance.

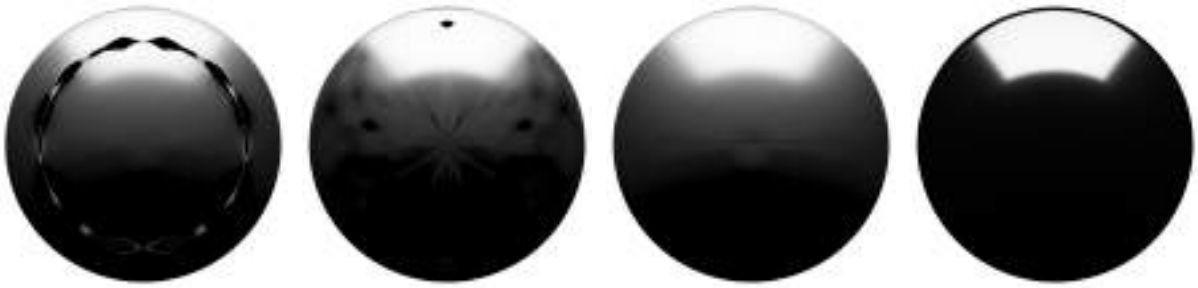
Chapter 4

Bringing Linearly Transformed Cosines to Anisotropic GGX

"We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances."
- Isaac Newton

Figure 4.1 Real-time area lighting with Linearly Transformed Cosines (LTCs) in a commercial game engine. Our LTC approximation for anisotropic materials takes 0.7 ms at 1080p resolution on an NVIDIA RTX 2080 GPU.

Linearly Transformed Cosines (LTCs) are a family of distributions that are used for real-time area-light shading thanks to their analytic integration properties, as we saw in Sect. 2.5.3. Modern game engines use an LTC approximation of the ubiquitous GGX model, but currently this approximation only exists for isotropic GGX and thus anisotropic GGX is not supported. While the higher dimensionality presents a challenge in itself, we show that several additional problems arise when fitting, post-processing, storing, and interpolating LTCs in the anisotropic case. Each of these operations must be done carefully to avoid rendering artifacts. We find robust solutions for each operation by introducing and exploiting invariance properties of LTCs. As a result, we obtain a small 8^4 look-up table that provides a plausible and artifact-free LTC approximation to anisotropic GGX. This makes it possible to analytically render anisotropic appearances with area lights.



(a) Broken fitted entry (Sect. 4.3) (b) Incorrect interpolation (Sect. 4.4) (c) Broken symmetry (Sect. 4.5) (d) Early inversion (Sect. 4.6)

Figure 4.2 Illustration of the problems to overcome for using LTCs with anisotropic GGX.

4.1 Introduction

Today's physically based shading models are largely based on the GGX *Bidirectional Reflectance Distribution Function* (BRDF) [5,29]. In real-time engines, computing direct illumination requires integrating the BRDF-light product. Dedicated techniques have been developed to integrate the GGX BRDF against different kinds of lights (probes, area lights, volumes, etc.). In this chapter, we focus on *Linearly Transformed Cosines* (LTCs), which have been widely adopted as a means to integrate the GGX BRDF against area lights of various shapes [8, 14, 15, 18]. For instance, LTCs are used in the Unity and Unreal engines for this purpose [30, 31]. However, their support is currently limited to the isotropic GGX BRDF, and thus anisotropic materials such as brushed metals cannot be shaded under area lighting with LTCs (see Fig. 4.1). The objective of this work is to alleviate this limitation and bring real-time area lighting to the anisotropic GGX BRDF via LTCs.

More specifically, LTCs are spherical distributions with analytic integration properties over specific spherical domains. Thanks to the LTC approximation of GGX, the integral of the BRDF over the spherical domain covered by an area light can be computed analytically in real time (Fig. 2.3). LTCs are represented by 3×3 matrices M fitted to isotropic GGX lobes and stored in a small 2D look-up table [8]. Computing a similar look-up table for anisotropic GGX raises new challenges, which is the focus of this work.

4.1.1 Objective

The crux of the problem is to obtain the 3×3 matrix M of the LTC that best approximates a given GGX lobe. Previously, Heitz et al. [8] proposed a fitting approach to compute a 2D look-up table

$$M = T_{\text{isoGGX}}(\theta, \alpha) \tag{4.1}$$

that approximates GGX lobes defined by the incidence angle θ and a roughness coefficient α. This is sufficient to cover the full isotropic GGX BRDF. Our objective is to compute a similar 4D look-up table

$$M = T_{\text{anisoGGX}}(\theta, \phi, \alpha_x, \alpha_y) \tag{4.2}$$

that takes an additional azimuthal angle ϕ and anisotropic roughness coefficients (α_x, α_y). Note that once the matrix M is obtained, all of the existing applications of LTCs (area-light integration with various shapes, importance sampling, etc.) can be used without further modification. Thus, the only problem to solve is the precomputation of the 4D look-up table T_{anisoGGX}.

4.2 Contributions in the context of previous work

Real-time stochastic techniques. Recent graphics hardware makes it possible to use purely stochastic techniques such as reservoir sampling [28]. With stochastic approaches, integrating area lighting with arbitrary materials is simple but leads to noisy results. In the context of this chapter, we instead aim to provide an *analytic shading* technique that produces a clean (noise-free) image.

Real-time analytic shading techniques. Even though stochastic techniques are appealing for the future, analytic methods remain important for today's real-time graphics. The first analytic solution to area lighting dates back to Lambert, who derived the irradiance from a polygonal light [10]. This early formula was brought to graphics for radiosity by Baum et al. [11] and was later extended by Arvo [16], who derived the integral over spherical polygons of cosines of arbitrary integer exponents (i.e., *Phong distributions*). Despite the detailed implementation of this technique provided by Snyder [32], it had limited practical impact due to the algorithmic complexity of the integration. In practice, real-time methods involved cheap approximations, mainly based on punctual evaluations [33–35]. As GPUs became more powerful, more accurate techniques arose, such as the approach of Lecocq et al., which was the first method to provide an accurate approximation for physically based materials while still being fast enough for real-time rendering [17]. This method was later outperformed by *Linearly Transformed Cosines* (LTCs) [8], which remain today's leading approach for real-time area-light shading. We build on the state-of-the-art LTC method by adding support of anisotropic materials.

Applications of Linearly Transformed Cosines (LTCs). While LTCs were initially proposed for real-time polygonal-light shading, they have since found uses in many applications that will benefit from our contribution. The LTC analytic integration has been extended to other types of light, such as line lights [14] and sphere/disk lights [15]. Another important addition to LTC integration is shadowing. Though LTCs do not include visibility in the analytic shading integral, a low-variance ratio estimator has been proposed to incorporate shadows on top of the analytic shading integral [13]. An alternative approach consists of incorporating visibility by removing the edges of occluders in the LTC integral [22, 36]. The analytic integration property of LTCs has also proven useful in offline rendering, in the context of path guiding [37]. Besides integration, LTCs can also be importance sampled to provide noise-free ray tracing with very low samples per pixel [38]. Furthermore, LTCs have also found

uses in differentiable rendering. Specifically, they have been used to select points on edges for efficient differentiable rendering [39] and to analytically compute gradients of the rendering equation [22]. Note that all of the aforementioned applications leverage properties of LTC distributions and work independently of how these distributions were fitted to a given material, and most of them use the isotropic GGX look-up table originally provided by Heitz et al. [8]. We provide a look-up table for anisotropic GGX.

BRDF fitting. There is a significant amount of work on the problem of fitting parametric models to BRDFs [40, 41] but the problem we address is different. BRDF fitting means fitting a 4D function with a simpler one. In our case, we fit the 2D outgoing-radiance lobe of the BRDF in each view-roughness configuration separately.

4.3 Fitting

In this section, we address the problem of fitting an LTC represented by a matrix M to a GGX lobe, as shown in Fig. 4.3.

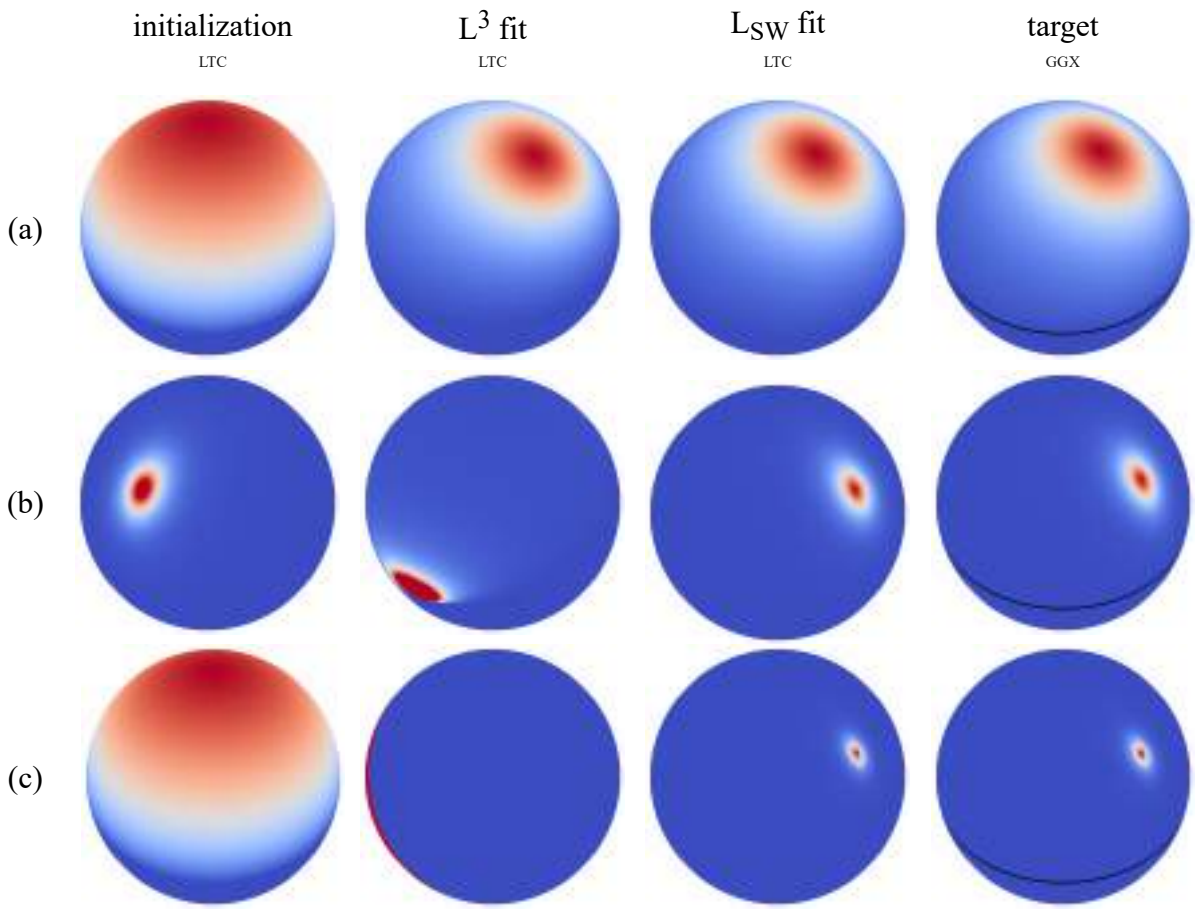


Figure 4.3 Fitting an LTC to a GGX lobe. We compare the L^3 fit of Heitz et al. to the L_{SW} fit we propose.

4.3.1 Experimenting with the Previous Approach

Heitz et al. minimize the point-wise $\mathcal{L}_3$ error between the cosine-weighted GGX BRDF (Eq. 2.13) and the LTC (Eq. 2.26). Their approach works in simple cases such as in Fig. 4.3-(a), but we observed two main issues that make it unstable in more challenging configurations, leading to broken fits in our look-up table (Fig. 4.2-(a)).

Problem 1: null gradients. In Figure 4.3-(b), the LTC provided as a starting point does not overlap with the target distribution. The gradients of the $\mathcal{L}_3$ error metric are thus close to 0 and the optimizer diverges.

Problem 2: high values. To avoid problem 1, in Figure 4.3-(c) we use a diffuse LTC (represented by an identity matrix M) as the starting point, such that there is significant overlap with the target GGX distribution. However, the target is sharp and evaluates to high values at the center of its lobe. These high values produce extremely high L^3 error gradients, which cause the optimizer to overshoot and stay trapped in a divergent configuration with null gradients (back to problem 1).

Discussion. Despite these issues, the $\mathcal{L}_3$ optimization of Heitz et al. is successful because of the *accuracy of their starting points*. They use a diffuse LTC (an identity matrix M) for high roughnesses and initialize the matrix parameters with the already-optimized neighboring entries of the look-up table as the roughness decreases. The resolution of their table (64×64) ensures neighboring entries are close enough to provide accurate starting points. However, we need to aggressively reduce the resolution to store a 4D table (Sec. 4.7), so neighboring entries do not always overlap, especially with sharp distributions (low roughness). This is why we need an optimization process that is robust even with poor initialization.

4.3.2 Our Approach

Our objective is to find an optimization metric that is not subject to vanishing gradients or numerical instabilities with sharp distributions and works regardless of the accuracy of the initialization.

The Sliced Wasserstein loss. We use the *Sliced Wasserstein (SW)* loss [42,43] between the direction samples of the target GGX lobe and the samples of the LTC distribution. This sample-wise loss approximates the optimal transport between two distributions and has shown several benefits in the machine learning community. The advantage over point-wise losses such as $\mathcal{L}_3$ used by Heitz et al. is that it always provides smooth and stable gradients [44].

Consider two Probability Density Functions (PDFs) f and g and their respective marginals f_ω and g_ω over a random direction $\omega \in \Omega$. The SW distance between f and g is the expected difference between their respective Inverse Cumulative Distribution Functions (iCDF) F_ω^{-1} and G_ω^{-1} over all random directions:

$$L_{SW}(f,g) = \mathop{\mathbb{E}}_{\omega\in\Omega}\left[\int_0^1 \left|F_\omega^{-1}(u) - G_\omega^{-1}(u)\right| \,\mathrm{d}u\right]. \tag{4.3}$$

Despite this formulation appearing complicated at first sight, its implementation is extremely simple, as we shall see in Alg. 6.

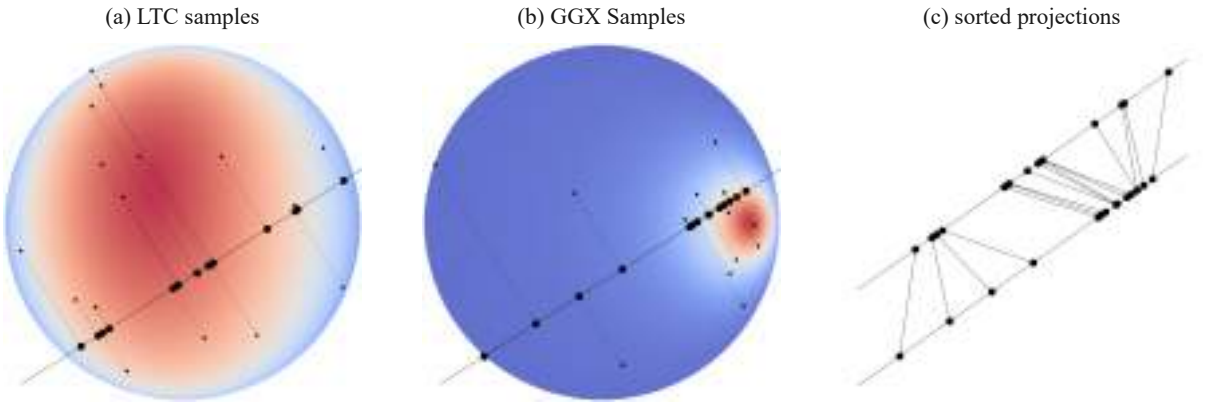


Figure 4.4 Illustration of Alg. 6. We project random samples from the LTC (a) and the GGX lobe (b) onto a random direction and average the absolute differences between the sorted projections (c).

Discretization. An inverse CDF can be approximated by a list of sorted samples from the corresponding density. Hence, if we consider two sets of random samples $(f_1, ..., f_n)$ and $(g_1, ..., g_n)$ from respectively f and g and their *sorted* projections $(f_{1,\omega}, ..., f_{n,\omega})$ and $(g_{1,\omega}, ..., g_{n,\omega})$ onto direction ω, Eq. 4.3 can be written as:

$$L_{SW}(f, g) = \lim_{n\to\infty} \mathbb{E}_{\omega\in\Omega} \left[\frac{1}{n} \sum_{i=1}^{n} |f_{i,\omega} - g_{i,\omega}| \right], \tag{4.4}$$

i.e., the average of the differences between the sorted projections over the set of projection directions.

Optimization. In Alg. 6, we compute a stochastic estimator of Eq. 4.4 by sampling n random samples from the densities, projecting the samples onto a random direction ω, sorting the projections and averaging the absolute differences. Finally, we propagate the gradient of the loss back to the matrix M and do a gradient descent step. The variables highlighted in blue are the ones through which the gradients are propagated from L_{SW} back to M. Fig. 4.4 illustrates the calculations. It is because this algorithm computes a mapping between samples rather than a difference between the densities that it is numerically stable even with sharp or non-overlapping densities, as shown in Fig. 4.3. With this algorithm, we successfully fit all the entries of our look-up table T_{anisoGGX}.

Implementation. We implement this stochastic estimator in a differentiable calculus library, PyTorch [45], which provides automatic gradient backpropagation, and use the *Stochastic Gradient Descent (SGD)* [46] optimizer for M. We compute 10000 gradient descent steps and for each step we use $n = 2048$ random samples and average the estimator over 64 random directions.

4.4 Interpolation

Even with the robust fitting approach described in the previous section, we noticed that our rendered results still suffered from jiggling artifacts shown in Fig. 4.2-(b) when interpolating the entries of our

ALGORITHM 6: Optimization step over the LTC matrix M with the Sliced-Wasserstein distance (blue variables depend on M).

Input: LTC matrix M
Input: GGX lobe view direction ω_o and roughnesses (α_x, α_y)
generate n random samples $(g_1, ..., g_n)$ from the GGX lobe `/* Alg. 1 */`
generate n random samples $(f_1, ..., f_n)$ from the LTC distribution `/* Alg. 2 */`
generate a random direction ω
$(f_{1,\omega}, ..., f_{n,\omega}) = \text{sort}(f_1 \cdot \omega, ..., f_n \cdot \omega)$
$(g_{1,\omega}, ..., g_{n,\omega}) = \text{sort}(g_1 \cdot \omega, ..., g_n \cdot \omega)$
$L_{SW} = \frac{1}{n} \sum_{i=1}^{n} |f_{i,\omega} - g_{i,\omega}|$
backpropagate gradient from L_{SW} to M
update $\mathrm{M} = \mathrm{M} - \epsilon \frac{\nabla L_{SW}}{\nabla \mathrm{M}}$

fitted look-up table T_{anisoGGX}. In this section, we explain that the *non-uniqueness* of LTCs (4.4.1) makes interpolation ill-defined, and we propose a solution to this problem (4.4.2).

4.4.1 Non-Uniqueness of LTCs

The essential point of this section is that different matrices M can produce the same LTC distribution.

Intuitive explanation. An LTC is a cosine distribution that has undergone a linear transformation M (and a normalization). But the cosine distribution can also be *invariant* under some linear transformations: rotations R_z aligned with the z-axis and flipping matrices F_{xy} that flip the x- and/or y-axis. Hence, these linear operations can be appended to the matrix M without changing the resulting LTC distribution, as shown in Fig. 4.5.

Property. For any rotation and flipping matrices:

$$\mathrm{R}_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{F}_{xy} = \begin{bmatrix} \pm 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.5}$$

the LTC distributions associated with matrices M and $\mathrm{M}\,\mathrm{R}_z\,\mathrm{F}_{xy}$ are the same.

Proof. We show that the evaluation of Eq. 2.26 remains the same if we replace M by $\mathrm{M}\,\mathrm{R}_z\,\mathrm{F}_{xy}$. First, the value of the Jacobian $\frac{\partial \omega_o}{\partial \omega}$ is unchanged because rotations or flips are area-preserving transformations (i.e., their Jacobians are 1). Second, the variable ω_o is mapped to another location where D_o (the cosine distribution) evaluates to the same value, since rotations around z, or x, y-axis flipping do not change the value of the cosine distribution.

4.4.2 Well-Defined Interpolation with Alignment

The obvious way to interpolate LTCs consists of interpolating their matrices M. The non-uniqueness of the matrix M of a given LTC therefore has a direct impact on the interpolation behavior.

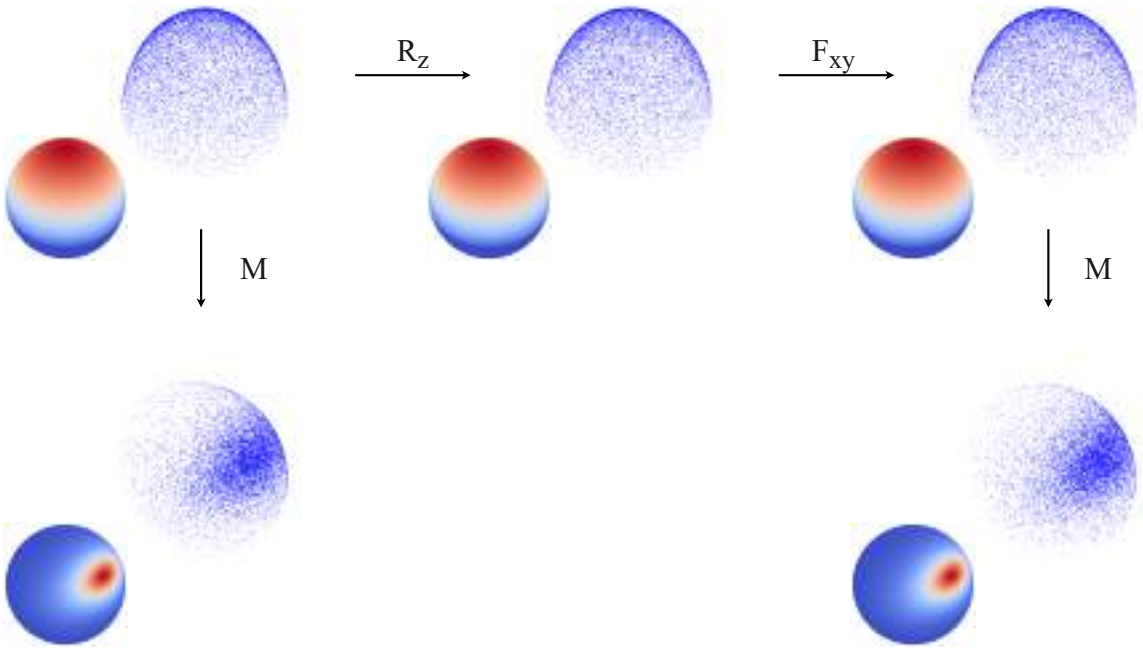


Figure 4.5 Non-uniqueness of LTCs. Cosine distributions remain unchanged under z-axis rotation R_z and xy flipping F_{xy}. Linearly transformed cosines inherit this invariance.

Naive interpolation. In Fig. 4.6-(a1), we show that two different matrices that represent the same LTC distribution yield another unexpected distribution under interpolation. In the general case of Fig. 4.6-(b1), where M_1 and M_2 represent different LTC distributions, interpolating the matrices does not produce a smooth transition between their respective distributions. Intuitively, this is because the matrices are *misaligned* due to the degrees of freedom introduced by their arbitrary rotation or flipping. This is what causes the interpolation artifacts shown in Fig. 4.2-(b).

Aligned LTCs. Our idea is to *align* the LTC matrices M to cancel out rotation and flipping of the transformed cosine samples by aligning them with the original cosine samples. To do that, for a given M, we find the LTC matrix $\mathrm{M}_{\text{aligned}} = \mathrm{M}\,\mathrm{R}_z\,\mathrm{F}_{xy}$ that minimizes the average squared distance between the original cosine samples and their transformed counterparts, i.e. we compute

$$\min_{\mathrm{F}_{xy}} \min_{\mathrm{R}_z} \quad \mathop{\mathbb{E}}_{\omega_o \sim D_o} \left[\left\| \frac{\mathrm{M}\,\mathrm{R}_z\,\mathrm{F}_{xy} \cdot \omega_o}{\|\mathrm{M}\,\mathrm{R}_z\,\mathrm{F}_{xy} \cdot \omega_o\|} - \omega_o \right\|^2 \right]. \tag{4.6}$$

Intuitively, this optimization minimizes the average squared lengths of the lines in Figure 4.7. We implement it as a linear search over the rotation angle α and the four flipping cases.

Robust interpolation. By aligning the LTC matrices before interpolating them, we obtain robust interpolation behavior. As expected, interpolating between the same distribution leaves it unchanged (Fig. 4.6-(a2)) and interpolating between different distributions produces smooth transitions (Fig. 4.6-

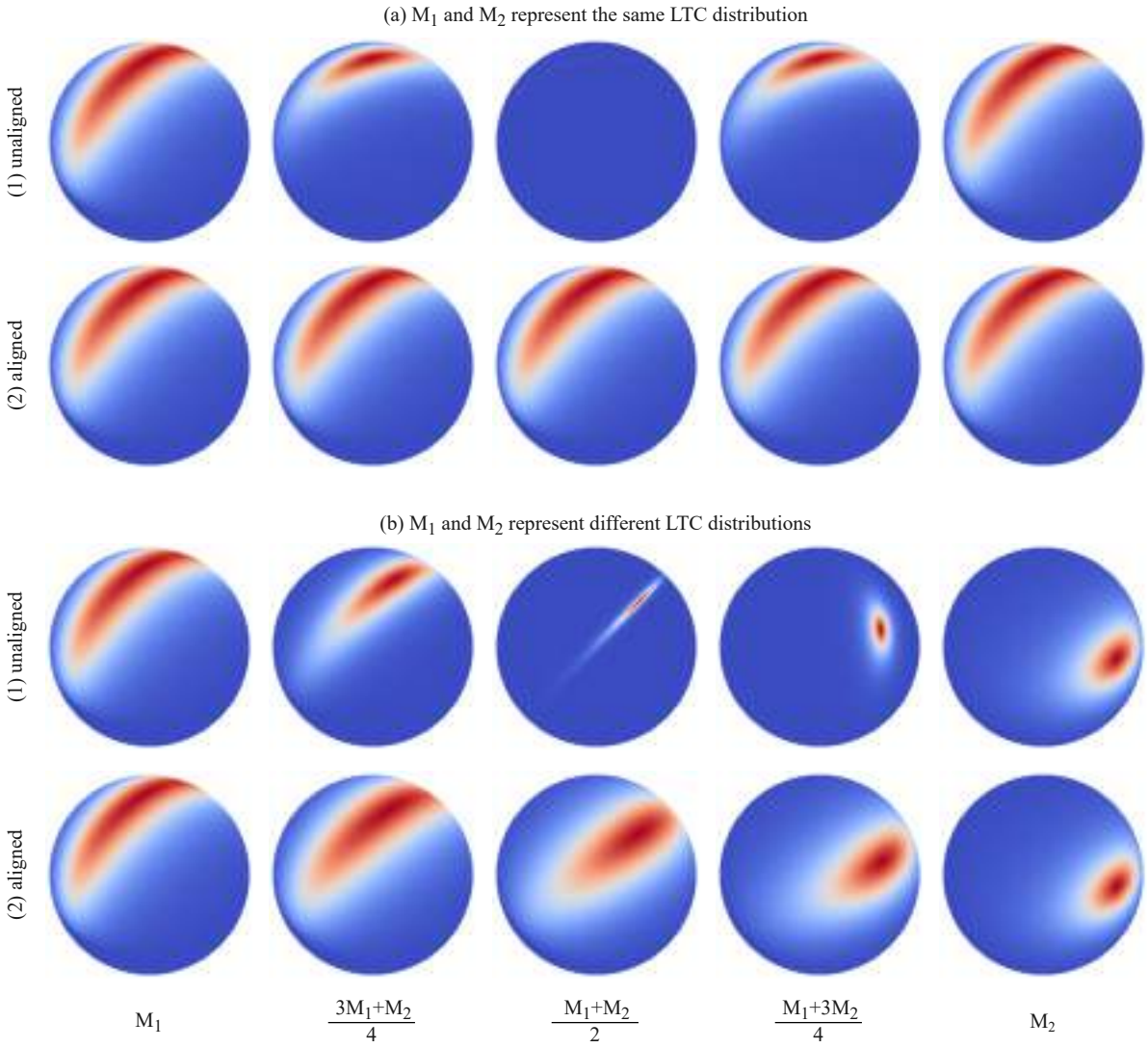


Figure 4.6 Interpolating LTC matrices with and without alignment.

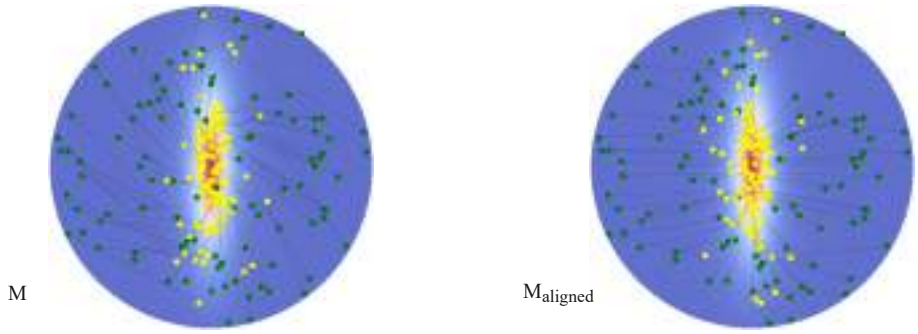


Figure 4.7 LTC alignment. We minimize the average squared distance between the cosine (yellow) and LTC (green) samples.

(b2)). Aligning the entries of our fitted look-up table T_{anisoGGX} removes the interpolation artifacts of Fig. 4.2-(b).

Discussion. Heitz et al. [8] do not report interpolation problems despite using a naive interpolation without alignment. This is because they only need four parameters of the LTC matrix to fit isotropic GGX: $\mathrm{M} = \begin{bmatrix} a & 0 & b \\ 0 & c & 0 \\ d & 0 & 1 \end{bmatrix}$. A side effect of this special case is that the null entries of the matrix force its alignment and thus make the interpolation well-defined. The interpolation problem that we solve arises in the general case where all nine parameters of the LTC matrix are used, which is necessary for fitting anisotropic GGX. This is why we are, to our knowledge, the first to face the problem of LTC interpolation misbehaving, and to investigate the non-uniqueness property of LTCs.

4.5 Symmetries

In this section, we leverage symmetries of the anisotropic GGX BRDF to remove numerical errors and reduce the storage induced by the dimensionality of our 4D fitted look-up table $T_{\text{anisoGGX}}(\theta, \phi, \alpha_x, \alpha_y)$.

4.5.1 Parameterization of the Look-Up Table

Azimuthal symmetry. The GGX BRDF has axial symmetries over the x and y axes with respect to the view vector, as shown in Fig. 4.8. We leverage this property to fit our look-up table over $\phi \in [0, \frac{\pi}{2}]$ rather than $\phi \in [0, 2\pi]$. We recover the full range $[0, 2\pi]$ in the following manner:

$$\mathrm{M} = \begin{cases} \begin{bmatrix} +1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\phi) & \text{if } 0 \leq \phi < \frac{\pi}{2}, \\ \begin{bmatrix} -1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\pi - \phi) & \text{if } \frac{\pi}{2} \leq \phi < \pi, \\ \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\phi - \pi) & \text{if } \pi \leq \phi < \frac{3\pi}{2}, \\ \begin{bmatrix} +1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(2\pi - \phi) & \text{if } \frac{3\pi}{2} \leq \phi < 2\pi. \end{cases} \tag{4.7}$$

This reduces the size of the look-up table by a factor of four for the same angular resolution.

Roughness symmetry. The GGX BRDF has roughness symmetries shown in Fig. 4.9 that can be written in the following manner:

$$T_{\text{anisoGGX}}(\theta, \phi, \alpha_x, \alpha_y) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\theta, \frac{\pi}{2} - \phi, \alpha_y, \alpha_x). \tag{4.8}$$

Storing the data directly with the (α_x, α_y) parameterization virtually duplicates the data, so instead we use an alternative parameterization $T_{\text{anisoGGX}}(\theta, \phi, \alpha, \lambda)$, where $\alpha \in [0, 1]$ is the largest roughness and $\lambda \in [0, 1]$ is the roughness ratio:

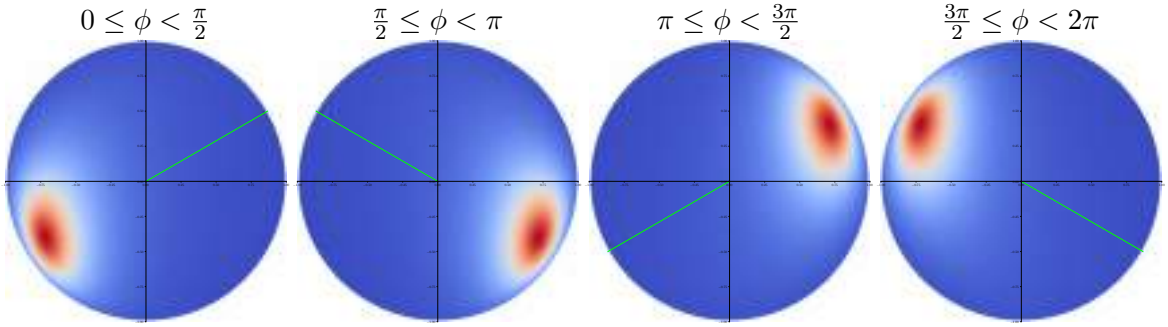


Figure 4.8 Azimuthal symmetries of the GGX BRDF. When the view vector (green) is symmetrized over the x and y axes, the GGX lobe undergoes the same symmetry.

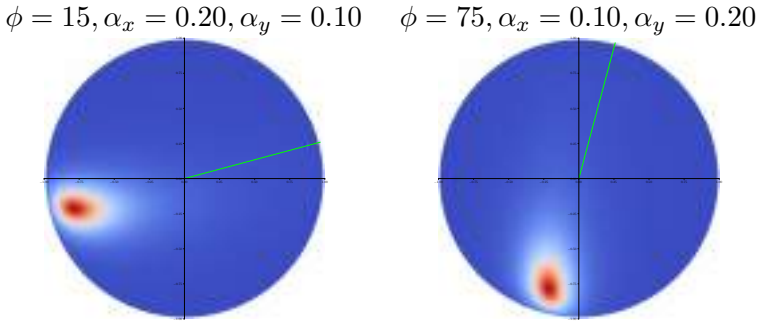


Figure 4.9 Roughness symmetries of the GGX BRDF. Permuting the x and y coordinates of the view vector and the roughnesses α_x and α_y produces the same permutation in the lobe's shape.

$$\mathrm{M} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\theta, \phi, \alpha_x, \frac{\alpha_y}{\alpha_x}) & \text{if } \alpha_x \geq \alpha_y, \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\theta, \frac{\pi}{2} - \phi, \alpha_y, \frac{\alpha_x}{\alpha_y}) & \text{otherwise.} \end{cases} \tag{4.9}$$

This increases the roughness resolution by a factor of two for the same storage.

4.5.2 Fixing Residual Errors in the Look-Up Table

Other symmetries of the GGX BRDF imply that certain entries of our fitted look-up table should be null. This would be the case if the fitting and alignment optimizations were perfect, but even small residual errors are sufficient to break these symmetries and produce the discontinuity and singularity artifacts present in Fig. 4.2-(c). We can eliminate these artifacts by enforcing the expected symmetries in the entries of the look-up table:

Axial symmetries. The GGX lobe has axial symmetry over the x and y axes when, respectively, $\phi = 0$ and $\phi = \frac{\pi}{2}$ (Fig. 4.10-(a, b)).

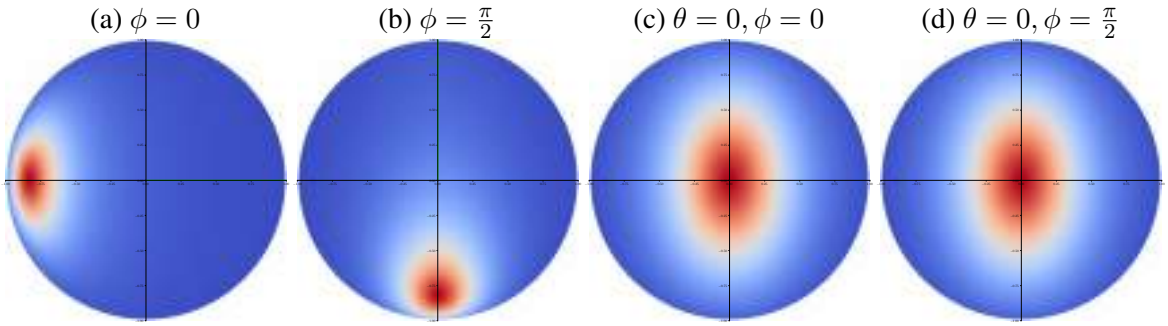


Figure 4.10 Symmetries of the GGX lobes with axis-aligned view directions.

Rotational symmetry. In upward views, where $\theta = 0$, the azimuthal angle ϕ does not contribute to the shape of the GGX lobe. Furthermore, the GGX lobe is centered on 0 and symmetric over the axes x and y (Fig. 4.10-(c, d)).

Look-up table post-processing. We post-process our look-up table to set the following entries (blue) to zero, and ensure that when $\theta = 0$ all of the entries match for the different values of ϕ:

$$T_{\text{anisoGGX}}(\theta, \phi) = \begin{cases} \begin{bmatrix} m_{00} & 0 & m_{02} \\ 0 & m_{11} & 0 \\ m_{20} & 0 & m_{22} \end{bmatrix} & \text{for } \phi = 0, \\ \begin{bmatrix} m_{00} & 0 & 0 \\ 0 & m_{11} & m_{12} \\ 0 & m_{21} & m_{22} \end{bmatrix} & \text{for } \phi = \frac{\pi}{2}, \\ \begin{bmatrix} m_{00} & 0 & 0 \\ 0 & m_{11} & 0 \\ 0 & 0 & m_{22} \end{bmatrix} & \text{for } \theta = 0 \text{ and } \phi = 0, \\ T_{\text{anisoGGX}}(\theta, 0), & \text{for } \theta = 0. \end{cases} \tag{4.10}$$

This post-processing step fixes the artifacts present in Fig. 4.2-(c).

4.6 Matrix Inversion

The LTC integration shown in Figure 2.3 actually uses the inverse matrix M^{-1}. Heitz et al. spare the inversion by storing and interpolating the inverse in their look-up table. However, interpolating the inverse creates severe distortions with our coarse resolution. This is because the diagonal coefficients of M that control the aperture of the LTC behave linearly with respect to α_x and α_y at low roughness while the diagonal coefficients of M^{-1} correlate with $\frac{1}{\alpha_x}$ and $\frac{1}{\alpha_y}$. We can therefore use a more aggressive discretization by storing and interpolating M instead of M^{-1}, and performing the inversion at run time in the fragment shader. To bring all of the entries of the look-up table into the same precision range, we divide M by the length of its third column, i.e., we constrain $\mathrm{M} \cdot [0, 0, 1]$ to be a unit-length vector.

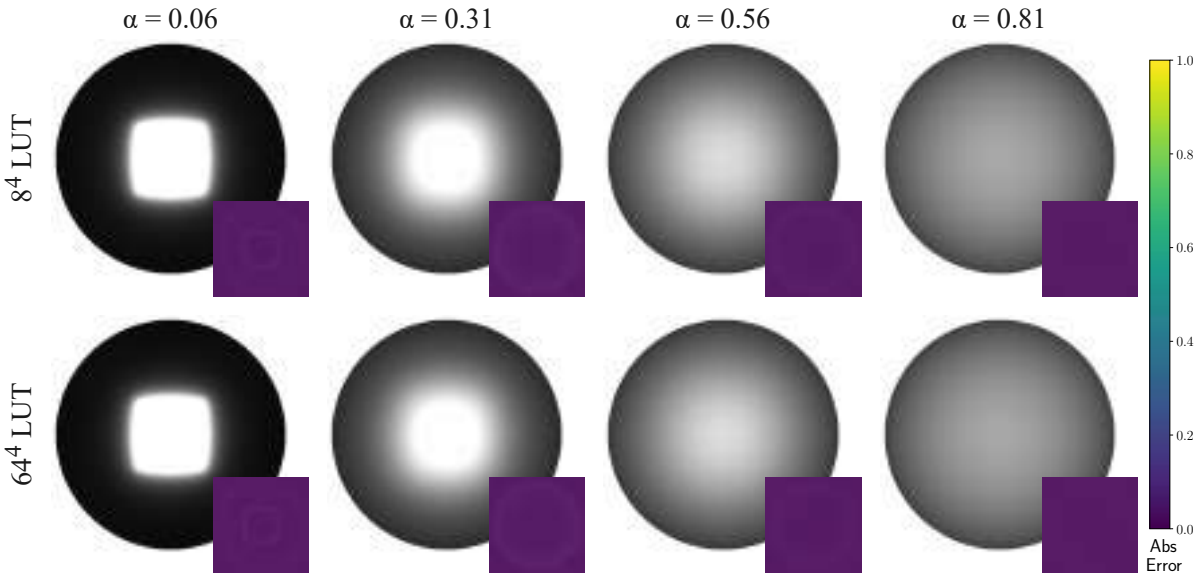


Figure 4.11 Renderings using 8^4 and 64^4 look-up tables. Inset difference images are with respect to the GGX reference.

4.7 Discretization

Heitz et al. [8] use a 64^2 resolution for their 2D look-up table $T_{\text{isoGGX}}(\theta, \alpha)$ with five floats per entry (they only use five parameters of the matrix M), which represents about 80 KB. If we were to use the same angular and roughness resolution, i.e., a 64^4 4D look-up table with nine floats per entry (we use the full matrix M), the memory requirement would be 576 MB. Thanks to the design choices introduced in the previous sections, we are able to reduce a resolution to 8^4. In Figure 4.11, we compare a resolution of 8^4 and 64^4 with respect to the GGX reference. The results show that the additional discretization error at 8^4 is negligible compared to the LTC approximation. As such, there is little to be gained by using a larger resolution.

4.8 Implementation of our Method

In this section, we explain the implementation of our method, and summarize its requirements in Table 4.1.

Precomputations. We compute an 8^4 table $T_{\text{anisoGGX}}(\theta, \phi, \alpha, \lambda)$ with a uniform discretization over $\theta \in [0, \frac{\pi}{2}]$, $\phi \in [0, \frac{\pi}{2}]$, $\alpha \in [0, 1]$ and $\lambda \in [0, 1]$. We compute each entry in the following way:

- We fit M to the corresponding anisotropic GGX lobe of roughnesses $\alpha_x = \alpha$ and $\alpha_y = \lambda\,\alpha$ using Alg. 6 from Sect. 4.3.
- We align M by minimizing Eq. 4.6 from Sect. 4.4, to ensure that interpolation is well-defined.
- We fix residual errors in M by applying Eq. 4.10 from Sect. 4.5.

	Heitz et al. [7, 8]	ours
param.	$\mathrm{M}^{-1} = T_{\text{isoGGX}}(\theta, \alpha)$	$\mathrm{M} = T_{\text{anisoGGX}}(\theta, \phi, \alpha, \lambda)$
resolution	64×64	$8 \times 8 \times 8 \times 8$
containers	2D textures (64×64)	3D textures ($64 \times 8 \times 8$)
channels	5 (for M) + 2 (for Fresnel)	9 (for M) + 2 (for Fresnel)
memory	112 KB	176 KB
interpolation	HW 2D	HW 3D + SW 1D
inversion	-	fragment shader
total timing	0.160 ms	0.610 ms
LTC integration	0.110 ms/light	

Table 4.1 Requirements of our method.

- We make M well-conditioned by dividing it by the length of its last column, as explained in Sect. 4.6.

The whole precomputation procedure is implemented in PyTorch and takes around two hours on an NVIDIA GeForce 2080 RTX GPU.

Storage. We store the parameters in 3D textures of resolution $64 \times 8 \times 8$ to benefit from trilinear hardware interpolation.

Run time. In the fragment shader, we proceed as follows:

- We get θ, ϕ, α_x, and α_y.
- We map ϕ to the first quarter $[0, \frac{\pi}{2}]$ using Eq. 4.7 and compute α and λ using Eq. 4.9.
- We fetch the 3D textures accordingly, manually interpolate over the last dimension, and apply the flipping and/or rotation matrices following Equations (4.7) and (4.9). This provides us with the matrix M.
- We invert M to obtain M^{-1}.

The cost of these operations is 0.610 ms for a full-screen quad at 1080p resolution with an NVIDIA GeForce RTX 2080 GPU, which is about four times the cost of the isotropic version.

LTC integration. Once M^{-1} is obtained, we use the existing LTC integration algorithms as in previous work [8, 14, 15]. Note that the overhead of our method only impacts the obtainment of M^{-1}, which can be amortized over the integration of multiple area lights.

Fresnel. We inject the Fresnel term of the GGX BRDF in the same way as Hill et al. [7]. We preintegrate the first and second Fresnel terms for each $(\theta, \phi, \alpha, \lambda)$ entry and store them as two additional channels in the look-up table that we interpolate at run time.

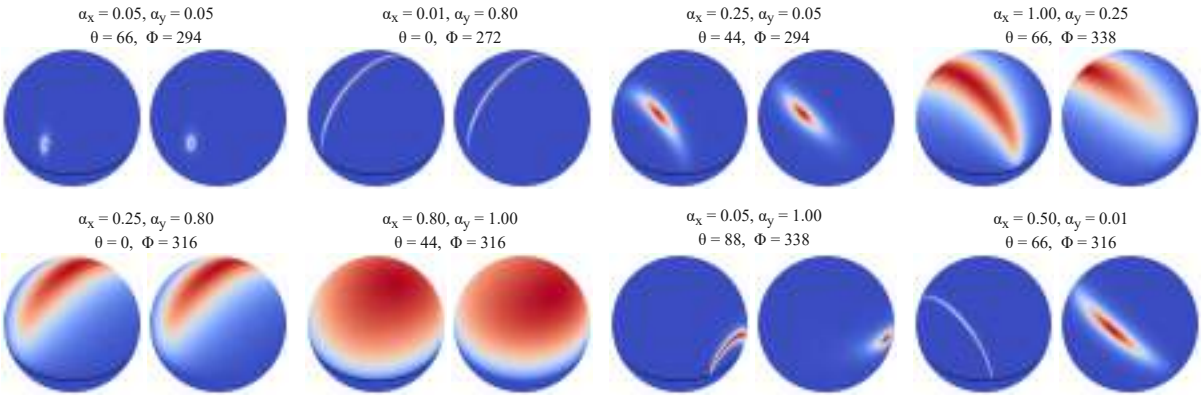


Figure 4.12 Plot results. We show the GGX reference (left) and our LTC approximation (right).

4.9 Results

In this section, we discuss the results produced by our method. Note that Chapter 8, Sect. 8.2 material covers a dense set of plot and rendering configurations.

Plots. Fig. 4.12 shows the fitted GGX lobes and our LTC approximation. We found out that the main limiting factor of our approximation is not our fitting technique but rather the representation power of LTCs. When the shape of the GGX lobe can be closely approximated by an LTC, our fitting technique is always successful (top rows in Fig. 4.12). However, the GGX lobe can exhibit *lune shapes* in certain configurations (high anisotropy, grazing view angle) and these shapes cannot be represented by LTCs (bottom rows in Fig. 4.12). Indeed, an LTC is a diffuse distribution transformed by a linear transformation. This allows for various first-order transformations such as changing the isotropic span of the lobe, elliptic anisotropy or skewness, but excludes lune-shaped lobes. In other words, it is not possible to accurately approximate these GGX configurations with LTCs, regardless of the fitting technique.

Renderings. Fig. 4.13 shows the GGX reference and our LTC approximation. As expected from the plots, the approximation might have large errors compared to the reference but it remains plausible, and we did not find configurations where the result is visually unacceptable. Therefore, we believe that our approximation is good enough to be considered for non-predictive real-time rendering, but would discourage its use for more demanding applications.

4.10 Conclusion

We have proposed a method to bring Linearly Transformed Cosines to anisotropic GGX. It is the product of the experience we gained from many failed attempts. New insights into the mathematical properties of LTCs, careful design choices, and attention to detail were all crucial elements in ensuring a clean and artifact-free approximation. We believe that the proposed method provides a plausible approximation that passes the quality bar for game engines. It has low memory overhead compared to the isotropic version already used by practitioners. The 4D texture fetch is more expensive than the isotropic version

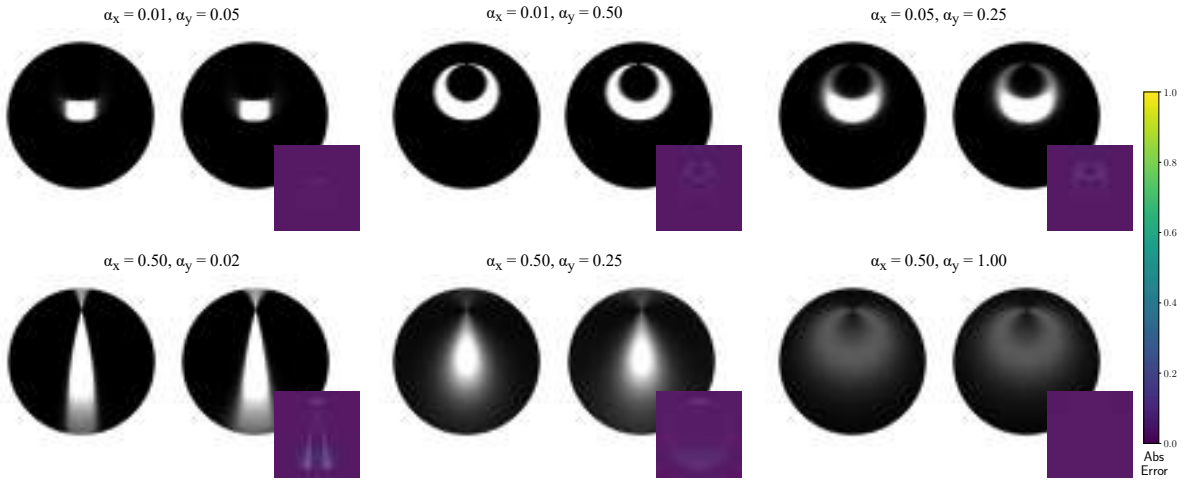


Figure 4.13 Rendered spheres with anisotropic materials and rectangular lights. We show the GGX reference (left), our LTC approximation (right), and the difference image.

but remains competitive for a real-time technique and can be amortized over multiple area lights. Thus, we don't see any barrier to using our anisotropic extension for video games. Furthermore, the same methodology could be applicable to other anisotropic materials.

The main limitation is that our approximation is not accurate enough for all applications. For instance, we do not advise using it for predictive rendering or as an importance sampling technique for anisotropic GGX.

A notable finding is that the limiting factor of our approximation is the representation power of LTCs, which cannot produce all of the possible GGX shapes, such as lunes. Therefore, we believe our method reaches the limit of what is possible with LTC approximations of GGX BRDFs. Further improvements should thus be sought with a fundamentally different approach.

Chapter 5

Linearly Transformed Spherical Distributions for Interactive Single Scattering with Area Lights

"Science is a perception of the world around us. Science is a place where what you find in nature pleases you."

- Subrahmanyan Chandrasekhar

Figure 5.1 We present a semi-analytic method for interactive single-scattering in homogeneous media with polygonal area lights. Our method achieves biased but noise-free renderings and better performance compared to Volume-RIS [47]. This figure shows equal-time renderings of two scenes, Sponza (left) with four area lights and a spherical light shaft (right) with two area lights inside the sphere. Our core unshadowed method (left) is completely noise-free, while Volume-RIS exhibits noise especially near the area light. Our method with the ratio estimator formulation for shadows (right) can handle volumetric shadows due to light shafts, rendering plausible light beams. Note that the we denoise the entire image for Volume-RIS and only shadows of our method for the figure on the right.

5.1 Introduction

This chapter relaxes assumption 4 of LTCs, as described in Sect. 2.6.2. We develop a new formulation based on Linearly Transformed Spherical Distributions (LTSD) which are a superset of LTCs, to analytically render unshadowed single scattering in infinite homogeneous participating media with polygonal area lights. At the core of single scattering is a spherical in-scattering integral and a spherical surface integral. These integrals have a form that is similar to the surface rendering equation where LTCs are applicable. However, they differ in two aspects: (1) the presence of an additional transmittance term and (2) usage of phase functions instead of Bi-directional Reflectance Distribution Functions (BRDF) in the in-scattering integral, making the application of LTSDs non-trivial. We present a simple and practical approximation of the transmittance term that can be analytically computed at render time. While this approximation already enables analytic evaluation of the spherical surface integral, the presence of a phase function in the in-scattering integral makes such a direct application non-trivial. We analyze LTSDs and determine the exact reasons that prevent their application to phase functions and propose a new and efficient LTSD precomputation approach.

An analytic solution to the in-scattering integral is however not sufficient. This integral is nested in a line integral which forms the Volume Rendering Equation (VRE). Analytic evaluation of the VRE is not straightforward due to this nesting and the fact that the in-scattering integral can only be evaluated at individual points on the media. We show that the line integral can instead be evaluated using quadrature rules. Although not analytic, they are well behaved across pixels thanks to their deterministic nature and produce noise-free renderings. Finally, we also formulate ratio estimators for the line and surface integral to render shadows in a separate denoised pass. Since only the shadow pass is denoised, renderings preserve high-frequency details, similar to the combination of LTCs with ratio estimators for surface rendering.

We comprehensively validate our method against Monte Carlo single-scattering references and show that it produces high-quality plausible single scattering renderings. We also compare our method with Volume-RIS [47] for unshadowed and shadowed configurations of both methods. Figure 5.1 shows two scenes rendered using our method compared to renderings of Volume-RIS. The left section of this figure shows unshadowed renderings where our method is completely noise-free. The right section shows shadowed renderings using denoised ratio estimators on a light shaft scene, demonstrating that our approach is capable of rendering volumetric shadows.

In summary, the following are the contributions of this chapter:

- A new LTSD fitting approach for phase functions which paves the way for analytic evaluation of the in-scattering integral.
- A noise-free semi-analytic formulation of the unshadowed Volume Rendering Equation (VRE).
- Ratio-estimator formulation of the VRE to handle shadows.

5.2 Contributions in the context of previous work

In this section we discuss related works which are grouped by the approach used to solve the VRE.

Stochastic Methods: Monte-Carlo integration can naively be applied to solve the VRE. However, better importance sampling strategies are needed to ensure better convergence. Sampling proportional to the transmittance [1] is a good strategy, however it disregards the contribution of the light term which increases variance. Kulla et al. [48] propose a method of sampling the initial distance which considers the incoming light from a point light source. Villeneuve et al. [49] further propose a method to sample proportional to the product of transmittance, the phase function and the orientation of a point light source. Both of the above methods can be applied to estimate the illumination due to a polygonal light source by randomly sampling a point on the light source at the cost of additional variance. Instead of sampling proportional to the product of phase function and light one can use Resampled Importance Sampling (RIS) [50]. RIS operates by generating a set of random initial candidates and chooses one of them by weighing them according to the target PDF. In the case of single scattering, one can sample the distance with a simple sampling strategy (for eg. proportional to the transmittance) and use RIS to obtain a sample which is distributed according to the VRE. RIS is at the core of Volume-ReSTIR [47] which is the state of the art method to render volumes.

Analytic Methods: Previous work in this area has mainly focused on obtaining closed-form solutions to single scattering in a homogeneous medium under the influence of punctual lights. Sun et al. [51] propose a combination of analytic evaluation and precomputation to achieve real-time rendering of single scattering under isotropic point lights. They also extend their formulation to render single scattering from distant complex illumination such as environment and achieve real-time frame rates. However, the interpolation of the precomputed data introduces artifacts in the rendering. Pegoraro et al. [52] propose an analytical solution for isotropic phase functions which does not rely on any precomputation and storage and produces more accurate results. However it is not suited for real-time applications. Further efforts to optimize the method [53, 54] and relax the condition of isotropic phase functions and lighting [55, 56] were also made. It is important to note that all of the analytic methods do not support shadows and additional methods need to be employed to add shadow information.

Our method works in a similar fashion, in that the core method itself does not support shadows and we show how ratio estimators can be formulated and combined for shadow support. We would like to point out that any other method that can render volumetric shadows [57–60] could potentially be used as well. Like the works of [61, 62], our method is based on the principles of volumetric scattering (VRE) and thus produces plausible renderings. Finally, similar to the usage of LTCs for better unbiased importance sampling of area lights for surface based scenes [38, 63], our LTSD formulation could be used for better sampling in participating media instead.

5.3 Preliminaries Recap

In this section, we revisit the Volume Rendering Equation (VRE) Eq. 2.1 to put it in context of semi-analytic single scattering from area lights.

The outgoing radiance L at a point $\mathbf{y}$ in the direction ω_o in the presence of homogeneous participating media is given by the Volume Rendering Equation (VRE) [64] as:

$$L(\mathbf{y}, \omega_o) = \mu_s \int_0^x T(\mathbf{y}, \mathbf{z}) L_s(\mathbf{z}, \omega_o) dz + T(\mathbf{y}, \mathbf{x}) L_c(\mathbf{x}, \omega_o), \tag{5.1}$$

where μ_s is a spatially constant scattering coefficient. We refer to the first term in the above equation as the air-light integral, inspired by Sun et al. [51]. In the following, we will assume that the scene contains one area light A with $A(\mathbf{x})$ denoting the solid angle it subtends on the unit sphere around $\mathbf{x}$. Our method trivially extends to multiple area lights by summing up individual contributions due to the linearity of light transport.

The point $\mathbf{z} = \mathbf{y} - z\omega_o$ is a point in the medium and L_s is its in-scattered radiance. With single scattering, we need to explicitly account for the visibility and the transmittance in L_s with the integration domain over the solid angle of the area light:

$$L_s(\mathbf{z}, \omega_o) = L_i \int_{A(\mathbf{z})} \rho(\mathbf{z}, \omega_o, \omega_i) T_v(\mathbf{z}, \omega_i) d\omega_i, \tag{5.2}$$

where ρ is the media's phase function which describes forward or back scattering. Similarly, the point $\mathbf{x} = \mathbf{y} - x\omega_o$ lies on a surface and its radiance L_c towards ω_o is given by:

$$L_c(\mathbf{x}, \omega_o) = L_i \int_{A(\mathbf{x})} f(\mathbf{x}, \omega_o, \omega_i) T_v(\mathbf{x}, \omega_i) |n_\mathbf{x} \cdot \omega_i| d\omega_i. \tag{5.3}$$

where f is the Bi-Directional Reflectance Distribution Function (BRDF) describing the surface's reflectance and $n_\mathbf{x}$ is the normal vector at $\mathbf{x}$. In both equations, L_i denotes the spatially constant incoming radiance from the area light A and T_v is defined as:

$$T_v(\mathbf{x}, \omega_i) = T(\mathbf{x}, \mathbf{t}(\mathbf{x}, \omega_i)) V(\mathbf{x}, \omega_i), \tag{5.4}$$

where $\mathbf{t}(\mathbf{x}, \omega_i)$ is the ray-casting function that returns the intersection point of a ray towards ω_i from $\mathbf{x}$. The visibility $V = 1$ if the area light A is visible from $\mathbf{x}$ in direction ω_i or $V = 0$ otherwise.

The function T in Eq. 5.1 and Eq. 5.4 gives the transmittance (or attenuation) between two points in a homogeneous medium:

$$T(\mathbf{y}, \mathbf{x}) = e^{-\mu_t ||\mathbf{y} - \mathbf{x}||_2}. \tag{5.5}$$

5.3.1 Linearly Transformed Spherical Distributions

Linearly Transformed Cosines or LTCs are a class of Linearly Transformed Spherical Distributions (LTSD) as defined by Heitz et al. [8]. An LTSD is defined by a matrix M that maps a source distribution D_o to a target distribution D as:

$$D(\omega) = D_o(\omega')\frac{\partial \omega'}{\partial \omega} = D_o\left(\frac{M^{-1}\omega}{||M^{-1}\omega||}\right)\frac{M^{-1}\omega}{||M^{-1}\omega||^3} \tag{5.6}$$

Heitz et al. [8] also establish the following equivalence for integral of a LTSD over an arbitrary solid angle P:

$$\int_P D(\omega)d\omega = \int_{P_o} D_o(\omega')d\omega', \tag{5.7}$$

where $P_o = M^{-1}P$ is the transformation of the original solid angle by the LTSD matrix M.

LTCs use the clamped cosine distribution for $D_o(\omega') = \frac{1}{\pi}\text{max}(n \cdot \omega', 0)$ and approximate $D(\omega) \approx f(\omega_o, \omega)|n \cdot \omega|$ i.e. the BRDF times cosine target. In this setting, the unshadowed surface radiance can be analytically approximated as:

$$\begin{aligned} L_c(\mathbf{x}, \omega_o) &= L_i \int_{A(\mathbf{x})} f(\mathbf{x}, \omega_o, \omega_i)|n_{\mathbf{x}} \cdot \omega_i| d\omega_i \\ &\approx L_i \int_{A(\mathbf{x})} D(\omega_i) d\omega_i \\ &\approx L_i \int_{A_o(\mathbf{x})} D_o(\omega_o) d\omega_o = L_i E(\mathbf{x}, A_o(\mathbf{x})), \end{aligned} \tag{5.8}$$

where $A_o(\mathbf{x}) = M^{-1}A(\mathbf{x})$ and E represents the cosine irradiance expression [11].

LTC matrices are precomputed in a table for different parameter combinations of the target distribution and fetched during rendering. For an isotropic BRDF, a two-dimensional table parameterized by (α, θ) is precomputed, where α is the BRDF's roughness and θ is the elevation of ω_o. For an anisotropic BRDF, this table is four-dimensional parameterized by (α_x, α_y, θ, ϕ) [65], where similarly α_x, α_y and BRDF roughness in x and y and θ, ϕ are the spherical angles are of ω_o.

5.4 Semi-Analytic Single Scattering

Our goal is to develop a semi-analytic method to render unshadowed single-scattering in homogeneous media with polygonal area lights. Achieving this goal involves deriving expressions for (1) unshadowed surface radiance L_c, (2) unshadowed in-scattered radiance L_s and (3) the air-light integral.

To this end, we first derive a practical closed-form approximation of the transmittance at a point towards an area light (Sect. 5.4.1). This expression is useful to derive analytic expressions for L_c (Sect. 5.4.2) and L_s (Sect. 5.4.3) using appropriate LTSDs. Finally, in Sect. 5.4.4, we describe the usage of quadrature rules that use analytic expressions of L_c and L_s to evaluate the air-light integral.

It is straightforward to apply LTSDs for analytic surface radiance as we show in Sect. 5.4.2, which is not the case for analytic in-scattered radiance due to the usage of phase functions. In Sect. 5.5, we discuss the challenges in fitting LTSDs to phase functions, identify the core problem and propose our solution that is used in Sect. 5.4.3.

We ignore the visibility V in our derivations of analytic L_c and L_s. In Sect. 5.6, we formulate ratio estimators to take visibility into account.

5.4.1 Aggregated transmittance towards an area light

The aggregated transmittance $\overline{T}$ from a point $\mathbf{x}$ towards an area light A can be written as an integral of the point trasmittance (Eq. 5.5) over the area light's solid angle $A(x)$:

$$\overline{T}(\mathbf{x}) = \int_{A(\mathbf{x})} e^{-\mu_t ||\mathbf{t}(\mathbf{x},\omega_i)-\mathbf{x}||_2} d\omega_i. \tag{5.9}$$

To find an analytic approximation of the above equation, we take the exponent out of the integral as follows:

$$\overline{T}(\mathbf{x}) = e^{-\mu_t d} \int_{A(\mathbf{x})} d\omega_i = e^{-\mu_t d} A(\mathbf{x}), \tag{5.10}$$

where d is a statistical quantity related to A. We use $d = \min(A)$. This expression is a reasonable closed-form approximation, and requires only the minimum distance to the light and the solid angle subtended by it. As such, Eq. 5.10 can be computed on the fly at rendering time.

5.4.2 Analytic surface radiance

Applying LTCs to obtain an analytic expression for L_c seems intuitive at first glance, due to similar forms of Eq. 5.3 and Eq. 5.8. However, LTCs can only be applied in the absence of the transmittance term. We thus split the transmittance out as a separate integral, similar to the split sum strategy used in image-based lighting [66]:

$$L_c(\mathbf{x}, \omega_o) = L_i \int\limits_{A(\mathbf{x})} T(\mathbf{x}, \omega_i) d\omega_i \int\limits_{A(\mathbf{x})} f(\mathbf{x}, \omega_o, \omega_i) |n_\mathbf{x} \cdot \omega_i| d\omega_i. \tag{5.11}$$

In the above equation, the integral on the right is now in a form where LTCs can be applied. Using also the result from Eq. 5.10, we get the expression for analytic surface radiance $\overline{L_c}$:

$$\overline{L_c}(\mathbf{x}, \omega_o) = L_i \overline{T}(\mathbf{x}) E(\mathbf{x}, A_o(\mathbf{x})), \tag{5.12}$$

where E is the cosine irradiance.

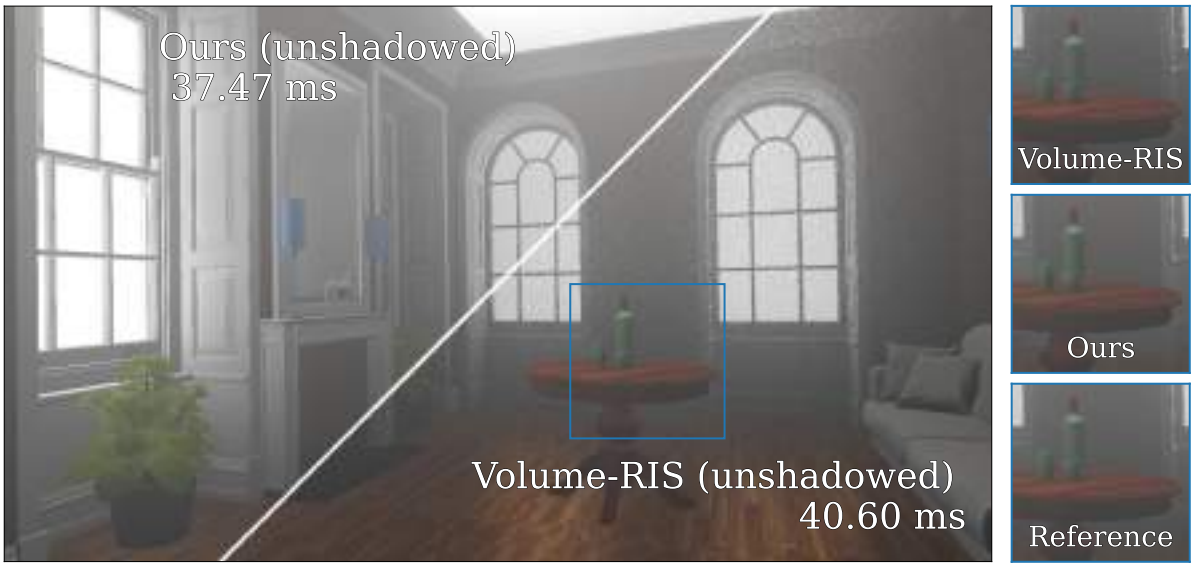


Figure 5.2 Equal-time ($\sim$ 40 ms) comparison of our semi-analytic unshadowed single scattering with Volume-RIS. Thanks to our use of analytic in-scattered and surface radiance (Eqs. 5.4.3, 5.4.2) and deterministic quadrature for the air-light integral (Eq. 5.15), we obtain noise-free renderings.

5.4.3 Analytic in-scattered radiance

To derive an analytic expression for the in-scattered radiance, we similarly split transmittance out of Eq. 5.2 and use the result from Eq. 5.10:

$$L_s(\mathbf{z}, \omega_o) = L_i \overline{T}(\mathbf{z}) \underbrace{\int_{A(\mathbf{z})} \rho(\mathbf{z}, \omega_o, \omega_i) d\omega_i}_{\text{Phase Function Integral}} . \tag{5.13}$$

Define S to be the analytic expression of the phase function integral, we get the final expression for analytic in-scattered radiance $\overline{L_s}$:

$$\overline{L_s}(\mathbf{z}, \omega_o) = L_i \overline{T}(\mathbf{z}) S(\mathbf{z}; A(\mathbf{z})). \tag{5.14}$$

S approximates the phase function integral by fitting LTSD matrices to the phase function. We show in Sect. 5.5 that the choice of source distribution D_o to compute these matrices is non-trivial: we cannot simply use a clamped cosine distribution, like done in LTCs.

5.4.4 Air-light integral

We now turn our attention to the air-light integral in Eq. 5.1. We observe that in the presence of the analytic in-scattered radiance $\overline{L_s}$, the air-light integral is a simple one-dimensional integral. In such a low dimensional setting, it is beneficial to use quadrature rules for better convergence.

We thus approximate the outgoing radiance in Eq. 5.1 using the Riemann sum, a formulation used also in ray-marching. With N partitions of the air-light integral, noise-free unshadowed single scattering $\overline{L}$ is computed as:

$$\overline{L}(\mathbf{y}, \omega_o) = \underbrace{\mu_s \sum_{i=1}^{N} T(\mathbf{y}, \mathbf{z}_i^*) \overline{L_s}(\mathbf{z}_i^*, \omega_o) \Delta z}_{\overline{\mathcal{A}}} + T(\mathbf{y}, \mathbf{x}) \overline{L_c}(\mathbf{x}, \omega_o), \tag{5.15}$$

where $\overline{\mathcal{A}}$ is the semi-analytic solution to the air-light integral, $\Delta z = z_i - z_{i-1}$ and $z_i^* \in [z_{i-1}, z_i]$. An added benefit of using quadrature rules is that they do not produce noise in the final renderings, thanks to their deterministic nature. Fig. 5.2 shows an equal-time comparison with Volume-RIS, demonstrating the noise-free nature of our method.

5.5 Fitting Linearly Transformed Spherical Distributions to Phase Functions

In this section, we describe our approach to fit LTSDs to phase functions. The resultant LTSD matrices are used to compute S for the analytic in-scattered radiance (Eq. 5.14).

The equation for L_s has a similar form to Eq. 5.8 albeit with a major difference: the BRDF is replaced by a phase function. In our work, we use the Henyey-Greenstein (HG) [67] phase function which has the following form:

$$\rho(\mathbf{x}, \omega_o, \omega_i) = \rho(\cos\theta) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}}, \tag{5.16}$$

where $\cos\theta = (\omega_o \cdot \omega_i)$ and $g \in [-1, 1]$ is the asymmetry parameter controlling the scattering (forward or back scattering). Unlike BRDFs, phase functions are defined on the entire sphere.

We work in a co-ordinate frame where ω_o is aligned with the z-axis. In this setting, the only parameter that controls the phase function's shape is g. We thus need to compute a one-dimensional table of matrices parameterized by g and approximate $D(\omega) \approx \rho(\omega_o, \omega)$ for the target. In the next subsections, we discuss different choices of the source distribution D_o to compute these matrices and their failure cases.

5.5.1 Clamped-cosine distribution for D_o

Fig. 5.3 visualizes the usage of clamped cosine distribution on the upper hemisphere for D_o. The top row of this figure shows the distribution D with a solid angle defined by a black boundary. The bottom row shows this solid angle and the distribution transformed with the corresponding LTSD matrix. While a clamped-cosine source distribution works well for highly directional lobes of Henyey-Greenstein, it fails as its lobes approach isotropic. The failure case is demonstrated at $g = -0.15$ when the solid

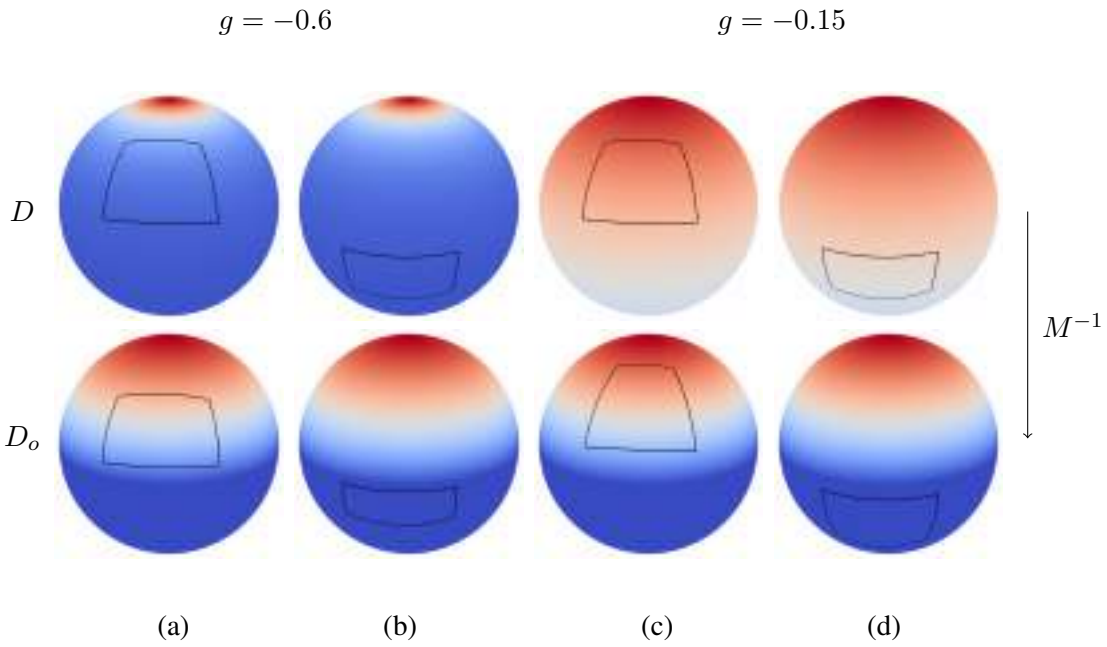


Figure 5.3 Approximating HG target ($D(\omega) = \rho(\omega_o, \omega)$, top row) with a clamped-cosine source ($D_o(\omega') = \max(n \cdot \omega', 0)$, bottom row) with a LTSD matrix M. (d) A solid angle with non-zero integral in D is transformed to have zero integral in D_o, demonstrating that fitting to clamped-cosine fails for near isotropic values of g.

angle lies in the lower hemisphere of D with a non-zero integral (Fig. 5.3(d)). After transformation, this solid angle still remains lower and will get clipped, giving a value of zero. This gives incorrect analytic integration for near isotropic values of g. Fig. 5.5 (b), top row shows the failure case of using LTSD matrices fitted to clamped-cosine for rendering, demonstrating the inability to render isotropic scattering.

5.5.2 Uniform spherical distribution for D_o

Next, consider using the uniform spherical distribution for D_o as shown in Fig. 5.4, which follows the layout of Fig. 5.3. This choice fails in the opposite case: at highly directional lobes of Henyey-Greenstein, and works well for near isotropic lobes. For example, at $g = -0.6$ with the solid angle in the lower hemisphere (Fig. 5.4(b)). The true integral has a value of zero, but after transformation, it evaluates to a non-zero value. In effect, this choice fails to capture forward and back scattering of phase functions. The result of rendering with matrices fitted to uniform spherical source is shown in Fig. 5.5(a), (c) top row, demonstrating that forward and back scatter effects are not reproduced.

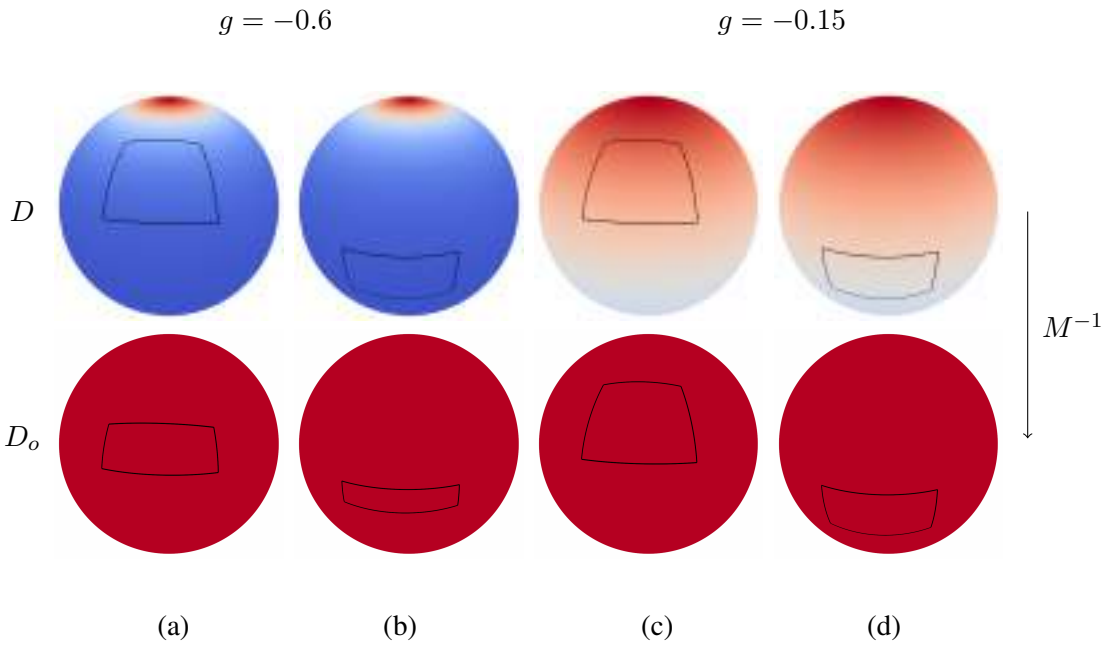


Figure 5.4 Approximating HG target ($D(\omega) = \rho(\omega_o, \omega)$, top row) with a uniform sphere source ($D_o(\omega') = 1/4\pi$, bottom row) with a LTSD matrix M. (b) A solid angle with a zero integral in D is transformed to have non-zero integral in D_o, demonstrating that fitting to uniform sphere fails for highly directional values of g.

5.5.3 Symmetric nature of Linearly Transformed Spherical Distributions

The core issue is that LTSDs are *symmetric* across the horizon plane. More generally, these transformations are point-symmetric across the origin; however in our case, the rotational symmetry of Henyey-Greenstein results in plane symmetry. The symmetric nature is illustrated in Fig. 5.6 by fitting different source distributions D_o to a HG target D with $g = -0.8$. The figure also visualizes directions S sampled uniformly on the sphere (Fig. 5.6 (b)). Fig. 5.6 (c) shows S transformed by a LTSD matrix fitted to uniform sphere source and Fig. 5.6 (d) similarly shows S transformed by a LTSD matrix fitted to clamped cosine source. Note that symmetry exists between the upper and lower hemisphere, and the transformed distribution in (c) and (d) do not match the target. This is precisely the challenge in fitting LTSDs to target distributions defined on the entire sphere, as they cannot capture asymmetric distributions on it.

5.5.4 Fitting to symmetric distributions on the unit sphere

Drawing from the analysis above, we propose the following. For a given value of g, we fit two LTSD matrices M_u and M_l. M_u transforms samples of HG in the upper hemisphere while M_l transforms samples of HG in the lower hemisphere. Effectively, we split the phase function integral in Eq. 5.13 as:

$$\int_{A^u(\mathbf{z})} \rho(\mathbf{z}, \omega_o, \omega_i) d\omega_i + \int_{A^l(\mathbf{z})} \rho(\mathbf{z}, \omega_o, \omega_i) d\omega_i, \tag{5.17}$$

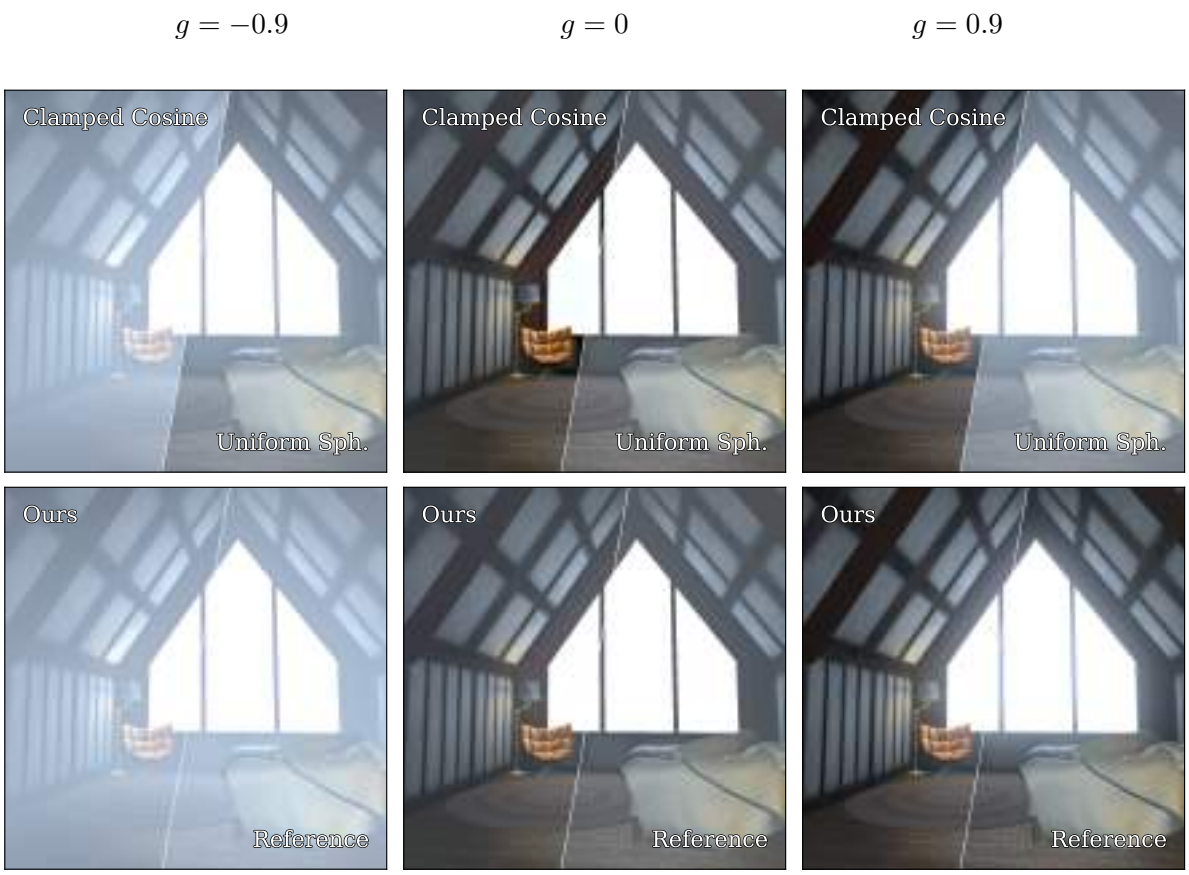


Figure 5.5 Renderings using our approach with LTSD matrices fitted to clamped-cosine source and uniform spherical source (top row). The bottom row shows renderings with our method and the proposed fitting approach in Sect. 5.5. Fitting to clamped cosine source is unable to render isotropic scattering ((b), top row) while fitting to uniform spherical source is unable to render directional scattering ((a), (c), top row). Our fitting approach renders all effects accurately, in accordance to reference ray-tracing (bottom row).

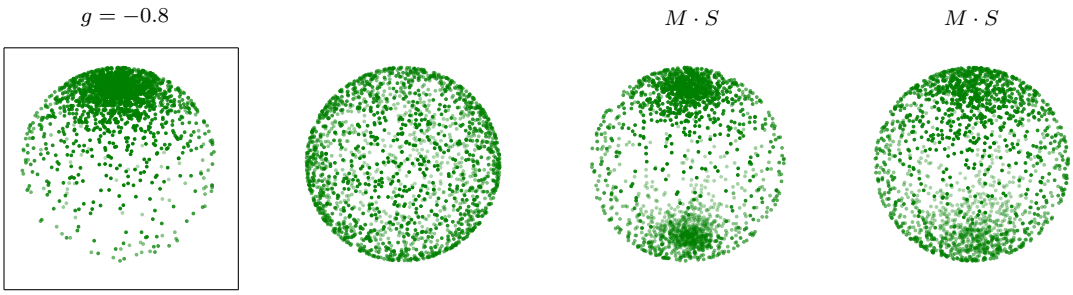


Figure 5.6 Fitting LTSDs to the Henyey-Greenstein phase function at $g = -0.8$ with uniform spherical and clamped-cosine source distributions. We randomly sample and visualize directions from (a) Henyey-Greenstein distribution, (b) uniform spherical distribution S. In (c), we visualize samples S transformed by a LTSD matrix fitted to uniform spherical source and in (d), we visualize S transformed by a LTSD matrix fitted to clamped cosine source. LTSD transforms are symmetric across the horizon and cannot capture a asymmetric distribution over the sphere.

where $A^u(\mathbf{z})$ is the solid angle in the upper hemisphere and $A^l(\mathbf{z})$ is the solid angle in the lower hemisphere. Given this, we can compute S in Eq. 5.13 using M_u and M_l as:

$$S(\mathbf{z}, A(\mathbf{z})) = \int_{A_o^u(\mathbf{z})} D_o^u(\omega')d\omega' + \int_{A_o^l(\mathbf{z})} D_o^l(\omega')d\omega', \tag{5.18}$$

where $A_o^u(\mathbf{z}) = M_u A^u(\mathbf{z})$ and $A_o^l(\mathbf{z}) = M_l A^l(\mathbf{z})$.

The source distribution to which these matrices are fitted depends on the value of g:

$$\begin{aligned} \text{for } M_u: \quad D_o^u(\omega') &= \begin{cases} \max(\omega' \cdot n, 0), & \text{if } g < 0 \\ 1 - \max(\omega' \cdot n, 0), & \text{if } g \geq 0 \end{cases}, \\ \text{for } M_l: \quad D_o^l(\omega') &= \begin{cases} 1 - \max(-\omega' \cdot n, 0), & \text{if } g < 0 \\ \max(-\omega' \cdot n, 0), & \text{if } g \geq 0 \end{cases}, \end{aligned} \tag{5.19}$$

where $n = (0, 0, 1)$. In effect, we fit to a clamped cosine source if the HG peak is in the upper hemisphere for M_u and if not, we fit to a inverse clamped cosine source. For M_l we fit the reverse. We found that renderings for forward and back scattering are better when using inverse clamped cosine to fit the hemisphere where the HG peak is not present.

The integral over an arbitrary solid angle of the clamped cosine is well known and used above in Eq. 5.8, Eq. 5.12 (cosine irradiance). Similarly, the integral over an arbitrary solid angle A_o of the inverse

clamped cosine distribution is:

$$\begin{aligned}\int_{A_o}(1-\max(n\cdot\omega',0))d\omega' &= \int_{A_o}d\omega' - \int_{A_o}\max(n\cdot\omega',0)d\omega' \\ &= A_o - E(A_o),\end{aligned} \tag{5.20}$$

that is, the cosine irradiance subtracted from the solid angle.

Combining Eq. 5.18, Eq. 5.19 and Eq. 5.20 we obtain an analytic expression for S:

$$S(\mathbf{z},A(\mathbf{z})) = \begin{cases} a_u\cdot E(\mathbf{z},A_o^u(\mathbf{z})) + & \text{if} \quad g<0 \\ \quad a_l\cdot[A_o^l(\mathbf{z}) - E(\mathbf{z},A_o^l(\mathbf{z}))] & \\ & \\ a_u\cdot[A_o^u(\mathbf{z}) - E(\mathbf{z},A_o^u(\mathbf{z}))] + & \text{if} \quad g\geq 0 \\ \quad a_l\cdot E(\mathbf{z},A_o^l(\mathbf{z})) & \end{cases} \tag{5.21}$$

where E is the cosine irradiance [11] as before, a_u is the amplitude of HG the upper hemisphere and a_l is its amplitude in the lower hemisphere. These amplitudes are defined as an integral over a given solid angle and are trivially precomputed, by integrating Henyey-Greenstein over the upper hemisphere for a_u and the lower hemisphere for a_l.

5.6 Ratio Estimators for Visibility

With our approach, using the formulation from Sect. 5.4 and LTSD matrices from Sect. 5.5, we can render biased but noise-free unshadowed single scattering. We now formulate ratio estimators to render shadows, following the work of Heitz et al. [13].

Since the final radiance L is given by a sum of two terms (Eq. 5.1, Eq. 5.15), we need to formulate two ratio estimators: one for the air-light integral and one for the surface radiance. A ratio estimator for the air-light integral can be formulated as:

$$R_{\bar{\mathcal{A}}} = \frac{\cancel{\mu_s}\int_0^x T(\mathbf{y},\mathbf{z})\left[L_i\int_{A(\mathbf{z})}\rho(\mathbf{z},\omega_o,\omega_i)T_v(\mathbf{z},\omega_i)d\omega_i\right]dz}{\cancel{\mu_s}\int_0^x T(\mathbf{y},\mathbf{z})\left[L_i\int_{A(\mathbf{z})}\rho(\mathbf{z},\omega_o,\omega_i)T(\mathbf{z},\omega_i)d\omega_i\right]dz}, \tag{5.22}$$

where the numerator uses T_v (transmittance with visibility) and the denominator uses T (only transmittance). We can similarly formulate a ratio estimator $R_{\bar{\mathcal{C}}}$ for the surface integral. The semi-analytic radiance $\overline{L}$ in Eq. 5.15 can be modified with these ratio estimators to render shadows:

$$\overline{L}(\mathbf{y},\omega_o) = \overline{\mathcal{A}}\cdot\langle R_{\bar{\mathcal{A}}}\rangle + T(\mathbf{y},\mathbf{x})\overline{L_c}(\mathbf{x},\omega_o)\cdot\langle R_{\bar{\mathcal{C}}}\rangle, \tag{5.23}$$

where $\langle . \rangle$ denotes a MC estimator. A benefit of this formulation is that the noise due to ratio estimators can be denoised independently. This denoising does not over-blur the final render thanks to its combi-

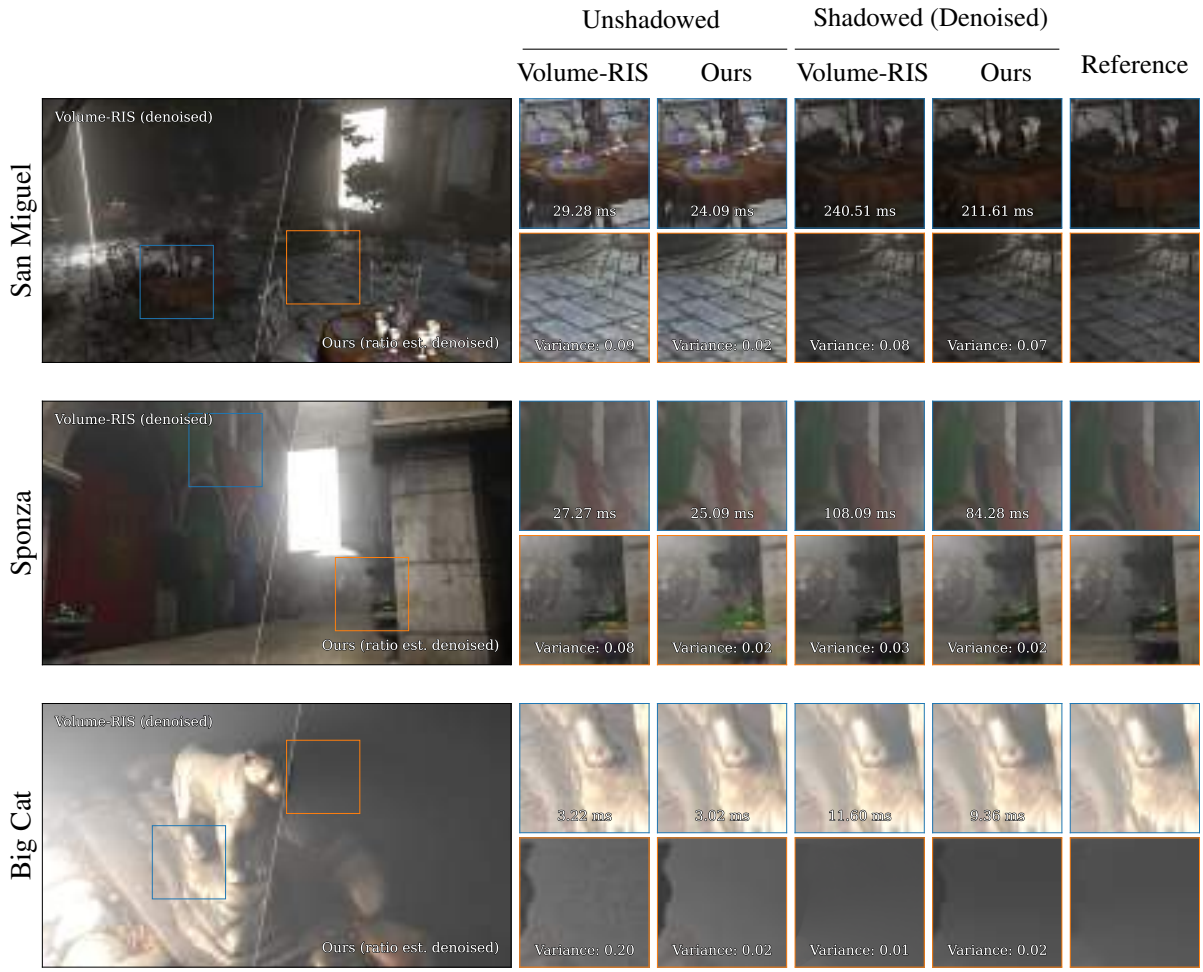


Figure 5.7 We perform equal-time comparisons on three scenes of our method with Volume-RIS, for both unshadowed and shadowed configurations. We used denoised Volume-RIS and denoised ratio estimators of our method in the shadowed comparison. We also show the variance, which is error between the rendered image and it's fully converged counterpart. Our method is preserves all visual details, thanks to the semi-analytic unshadowed part while also achieving lower variance in most cases. All scenes are rendered with $N = 8$ quadrature samples.

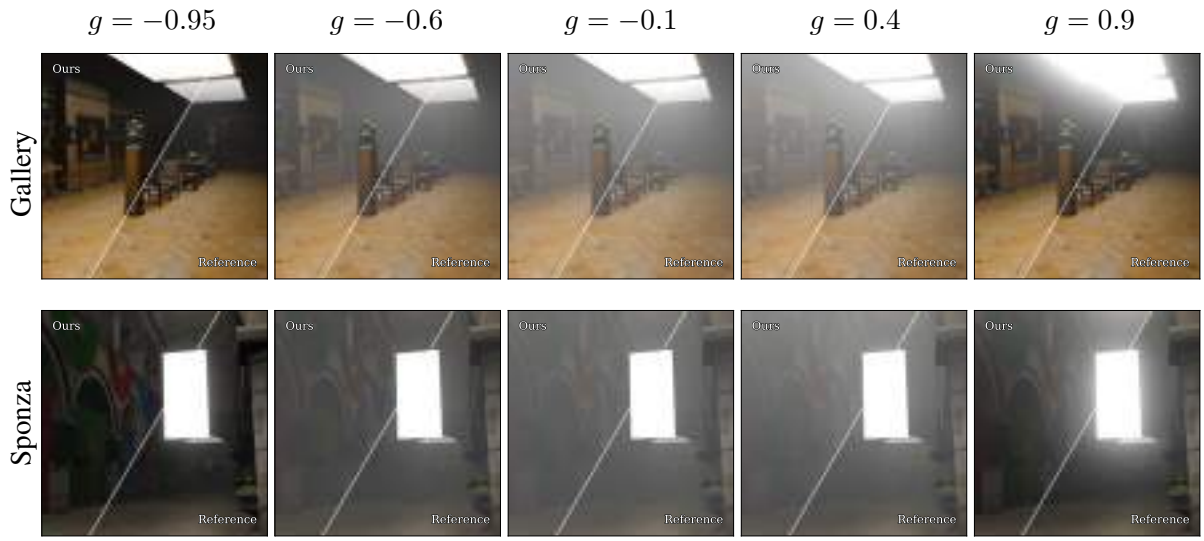


Figure 5.8 Varying values of g rendered using our method and a 2K spp reference. Our method plausibly renders forward and back scattering effects.

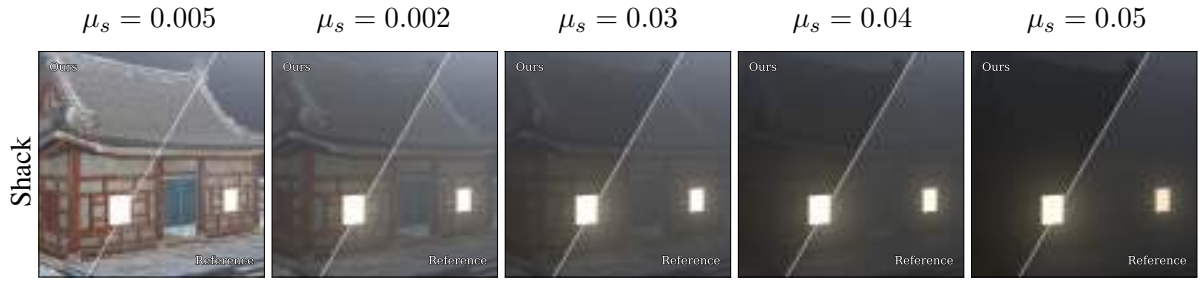


Figure 5.9 Varying values of the scattering coefficient μ_s rendered using our method and a 2K spp reference. Our method plausibly renders density variations.

nation with noise-free semi-analytic radiance $\overline{L}$. More specifically, we separately denoise the numerator and denominator of $R_{\bar{A}}$ and $R_{\bar{C}}$ before evaluating Eq. 5.23

Fig. 5.7 shows an equal-time comparison of our method with denoised ratio estimators with denoised Volume-RIS. Denosing the output of Volume-RIS leads to blurring of details, most visible in the red curtain in Sponza and the floor in San Miguel. Our method preserves these details thanks to the semi-analytic unshadowed contribution.

5.7 Implementation

We implement our semi-analytic single scattering approach with CUDA and OptiX [68] for hardware accelerated ray-tracing. We refer the reader to Chapter 8, Section 8.3 for the pseudo-code of our rendering algorithm. We also implement Volume-RIS, which is the method of Lin et al. [47] *without* spatial

and temporal reuse. This choice was made for easier evaluation, and as such any form of reuse will also benefit our method (by applying reuse to ratio-estimators), proportionally scaling the render quality and run-times.

Our implementation for fitting LTSDs to phase functions builds on the fitting approach and the code of KT et al. [65]. We modify the code to sample phase functions instead of a BRDF. We also need sampling routines for the two source distributions: clamped-cosine and inverse clamped cosine (Eq. 5.19). The sampling procedure for the former is well-known, and we derive a sampling procedure for the latter in Chapter 8, Section 8.3.

Since Henyey-Greenstein is symmetric across $g = 0$, we make the following optimization to store the LTSD matrices. For a value of $g \in [-1, 0]$, we fit matrices using the clamped cosine source. This is done for 128 equally spaced values of g and stored in a table $\mathcal{T}_1$. For a value of $g \in [0, 1]$, we fit matrices using the inverse clamped cosine source, and similarly store in a 128 dimensional table $\mathcal{T}_2$. For both cases, we discard samples in the lower hemisphere of HG during fitting. The matrices M_u and M_l are fetched as follows:

$$
\begin{aligned}
M_u &= \begin{cases} \mathcal{T}_1(g) & \text{if} \quad g < 0 \\ \mathcal{T}_2(g) & \text{if} \quad g \geq 0 \end{cases} \\
M_l &= \begin{cases} \text{diagonal}(1, 1, -1) \cdot \mathcal{T}_2(|g|) & \text{if} \quad g < 0 \\ \text{diagonal}(1, 1, -1) \cdot \mathcal{T}_1(-g) & \text{if} \quad g \geq 0 \end{cases}
\end{aligned} \tag{5.24}
$$

This avoids repeated storage, requiring only two one-dimensional tables of size 128 to be precomputed and stored. The fitting precomputation takes about six hours on a NVIDIA RTX 3090 GPU.

5.8 Results & Comparisons

In this section, we validate the renderings of our method, discuss its characteristics and compare with Volume-RIS [47]. Since Volume-RIS is unbiased, we use it for large number of samples per pixel (spp) reference comparisons as well.

Our rendering results are shown on various scenes with varying number of area lights, geometric and texture detail. We show both unshadowed and shadowed renderings and our scenes also help evaluate volumetric shadows with light shafts (Fig. 5.1 right, Fig. 5.7 Big Cat). All images are rendered at a resolution of 1920×1080 or 1080×1080 on a workstation with NVIDIA RTX 3090 GPU, unless otherwise specified.

5.8.1 Validation

We begin by validating our method with ground truth ray traced 2K spp reference. We evaluate renderings at different parameter values of the medium and setting $N = 500$ (no. of quadrature samples). Fig.

Figure 5.10 Equal-time renderings of Volume-RIS compared with our renderings for various values of N (no. of quadrature samples), on two scenes. For all values of N, our method achieves a noise-free result, with bias being more for lower values. N thus serves as a quality-speed tradeoff slider of our method.

5.8 shows renderings of our method for different values of $g = \{-0.95, -0.6, -0.1, 0.4, 0.9\}$ on two different scenes, compared with the reference. This validation serves to show that our method plausibly renders backward (negative g) and forward scattering g (positive g), thanks to our fitting approach in Sect. 5.5. Fig. 5.9 similarly evaluates renderings of our method for different values of μ_s against the reference. Note that μ_s is a spectrally varying quantity that is supported by our method as well. However, for validation purposes, it suffices to use a constant values across the spectrum. Our renderings faithfully captures the density variation in the medium.

5.8.2 Comparisons

In Fig. 5.7 we perform equal-time comparisons of our renderings with Volume-RIS on three scenes, in both unshadowed and shadowed configurations. Our unshadowed method visually performs better than Volume-RIS thanks to no noise. In the shadowed configuration, we denoise Volume-RIS and the ratio estimator in our method. Note that for Volume-RIS, we perform denoising separately on the medium and the surface for a fair comparison. The renderings of Volume-RIS exhibits blurring from the denoiser, especially of high-frequency details (floor in San Miguel, curtain in Sponza). This is not the case for our method, since we get exact details from our semi-analytic unshadowed part. The bottom row of this figure and the teaser figure show the renderings on a shadow shaft scene, demonstrating that our ratio estimators work well for volumetric shadows too.

We also show the variance for each configuration in Fig. 5.7. The variance of a method is computed as the error between the current render and its converged counterpart. This error estimates ignores the bias and allows us to measure variance. In most cases, we achieve lower variance than Volume-RIS.

Figure 5.11 We show bias of our method with false color images, computed by taking a difference with a 2K spp reference.

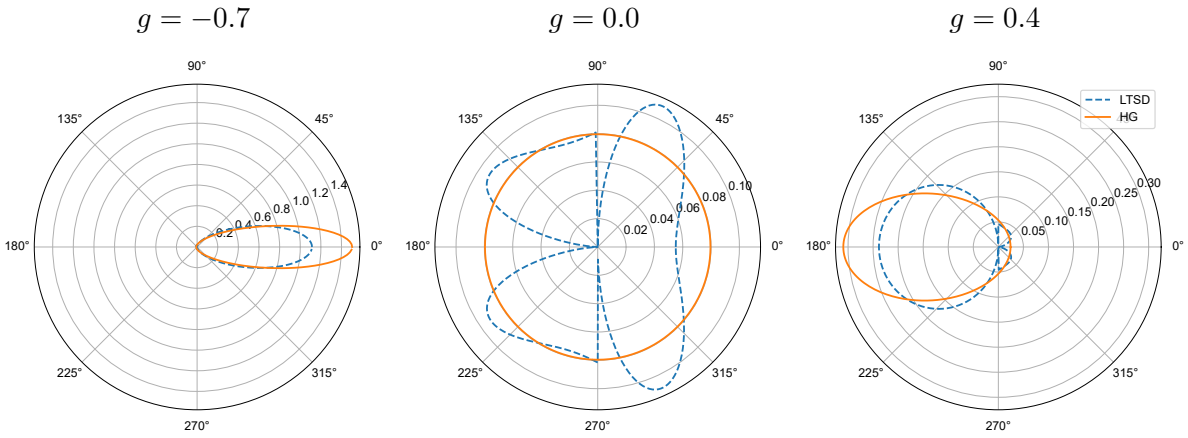


Figure 5.12 Polar plots of the Henyey-Greenstein phase function (orange solid line) and our LTSD approximation from Sect. 5.5.4 (blue dotted line). Our approximation follows the overall shape of HG, but is discontinuous across the horizon and doesn't match HG values for individual angles (similar to previous LTC fits not exactly matching the GGX BRDF). We thus do not recommend to use it for applications such as importance sampling (a similar observation was drawn by KT et al. [65]). However, our approximation results in plausible rendering, owing to it being integrated over the solid angle of the area light.

5.8.3 Characteristics

Number of Quadrature samples. We evaluate equal-time renderings of our method compared to Volume-RIS for different values of N in Fig. 5.10. For all values of N, our method renders a noise-free result, thanks to the use of analytic evaluations of Sect. 5.4.2, Sect. 5.4.3 and the use of quadrature rule in Sect. 5.4.4. The bias increases for lower values of N, but the time taken to render at these values approaches real-time values. Thus, N can serve as a quality-speed tradeoff slider of our method, which is useful for gaming and lookdev applications. Equal-time renderings of Volume-RIS have significant noise at lower N which quickly startes to disappear for higher N.

Bias. Fig. 5.11 shows the renderings of our method for $N = 500$ and a false-color difference with ray-traced 2K spp reference. This figure shows the bias of our method, due to the approximation of LTSDs and aggregated transmittance approximation (Eq. 5.10). The bias of our method is acceptable for many applications and the renderings are plausible since our method is derived directly from the VRE.

Temporal Consistency. Our method is temporally consistent, in-spite of no temporal information sharing, as shown in our supplementary video. This is because our renderings have no noise and our approximations smoothly vary across frames. This is also true with shadows, where the ratio estimator is denoised. Since denoising is done only on the ratio part and combined with our analytic result, it hides most temporal inconsistencies [13].

5.9 Conclusion, Limitations & Future work

We presented a method for semi-analytic interactive and unshadowed single scattering in homogeneous media with area lights. We formulated our method in a way that permitted the application of LTSDs for analytic evaluation. We showed that applying LTSDs to the in-scattered radiance is not straightforward due to the presence of a phase function that is defined on the entire sphere. This required analysis on the failure cases of LTSDs and identification of the core problem that prevents their application. We then proposed a solution that accurately fits phase functions and plausibly renders forward and back scattering effects. We also provided details to efficiently fit LTSDs to phase function using our approach, taking advantage of symmetries. Finally, we thoroughly validated our method's renderings and discussed its various characteristics. We also showed equivalent to better quality results compared to equal-time renderings of Volume-RIS.

Limitations. The biggest limitation of our method is that it is not physically correct, even though we base our formulation on the VRE. Although the resulting renderings are plausible, they may contain subtle artefacts: Fig. 5.8, rightmost shows an example over-brightening due to the transmittance approximation. Evaluation with quadrature of the air-light could also cause ringing artefacts, for example, if samples for two adjacent rays cross a light boundary, and should be carefully handled.

For a scene with a large number of area lights, the LTSD evaluation starts to get expensive and may not hold benefit over pure Monte Carlo evaluation, especially with a good importance sampling strategy

like Volume-RIS. A minor drawback of our method is also that we now have two hyper-parameters controlling the render quality: (1) the number of quadrature samples and (2) the number of ratio estimator samples. This can be problematic for artists as these parameters, like the number of samples in traditional MC, are not intuitive.

In our current formulation, we have not considered emissive media and our method assumes homogeneous infinite media - thus it cannot handle bounded or heterogeneous media.

Finally in Fig. 5.12 we show polar plots of the HG phase function in orange solid lines and our fits in blue dotted lines. Our fits match the overall shape of the Henyey-Greenstein phase function, it may not be exact for most individual angles. We thus do not recommend to use our LTSD fits for sampling applications. Furthermore, our fit is not continuous across the horizon for near isotropic values of g. This is akin to previous fits shown in Heitz et al. [8] and KT et al. [65], where the LTC fits do not exactly match the GGX BRDF. We note that a similar recommendation was made by KT et al. [65].

Future Work. In the future, we would like to minimize the discontinuities and the deviations of our fits from the Henyey-Greenstein phase function. We would also like to explore the use of other phase functions like the SGGX phase function [69]. This should be possible with the fitting approach presented, but needs to be carefully explored. We would further like to derive a better approximation of the aggregated transmittance than the one in Eq. 5.10. It may be possible to use Stokes theorem for this. The extension to bounded and heterogeneous media is an interesting future direction, along with exploring the extension to non-exponential media.

Chapter 6

Accelerating Hair Rendering by Learning High-Order Scattered Radiance

"I think we all change each other's paths. I don't know which law idea that is in physics, but I don't think any of us can live without affecting one another."
- Frank Ocean

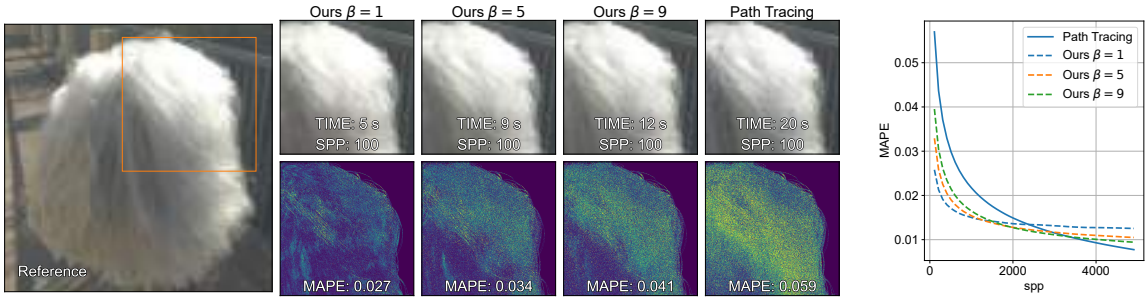
6.1 Introduction

Accurately rendering hair & fur is a challenging open problem. The appearance of hair is the result of light-matter interactions at different scales. On the one hand, local scattering at each individual fiber is responsible for glints and anisotropic highlights. On the other hand, multiple scattering of light between the potentially large number of hair strands generates a colored low-frequency shading, especially in light colored hair.

It is precisely this combination of high-frequency local scattering and potentially large number of global scattering events that make hair rendering computationally expensive: light transport in hair involves tracing rays for a near caustic light path and the final color is obtained only when the radiance from many such paths is averaged. Even with hardware accelerated ray tracing, the large number of scattering events result in slow convergence. Thus, accelerating computation and convergence of path tracing in hair is an active research area.

Several works have been proposed that accelerate multiple scattering by caching the illumination [70, 71], approximating it using diffusion [72], or by means of asymptotic approximations under certain assumptions such as *dual scattering* [73]. These methods are either too complex while gaining little computational advantage, or make assumptions about the scene & lighting or impose strong approximations lacking the visual fidelity of hair.

Our work starts from the observation that light transport in hair presents two different regimes: (1) A low order scattering term responsible for highlights, glints and a directional glow, (2) A high order scattering term which further contributes to this glow along with a diffuse-like scattering component. These regimes are not exclusive to hair and manifest in other discrete media as well. We thus follow a



(a) (b)

Figure 6.1 We propose to use a small multi-layer perceptron (MLP) to approximate higher order scattering in hair without any pretraining, by online learning of the error between biased and unbiased light paths. Our method converges faster compared to path tracing and provides control over it's bias & speedup via the max. path depth parameter β. This figure shows the characteristics of our method on messy white hair, with the MAPE metric against a 10k spp reference: (a) Our method with $\beta = 1$ converges to about one-half the error in one-fourth of the time while tracing the same amount of samples per pixel (100 spp). By increasing β, the error and time required to render approaches the time taken by path tracing. (b) Equal spp convergence graphs show that with lower β values, our method converges to a lower error faster (since same spp takes less time) but to a biased result at high spp (MAPE of path tracing eventually crosses over to a lower value). With higher β, our method's bias reduces eventually approaching path tracing.

similar approach to methods proposed for granular media [74, 75] and decompose the light transport of hair into these regimes.

Similar to these methods for granular media, our aim is to avoid precomputations and assumptions on the scene structure. However, unlike such methods and other hair rendering techniques, we aim for: (1) constant time to render the high-order scattering component, and (2) controllable bias and speedup of the renderer with an accurate estimate of render time. To this end, we propose to use a small multi-layer perceptron (MLP) to approximate the high order scattering component on-the-fly.

Specifically, in our approach, rendering a frame consists of three stages: In the first stage, we train an MLP, on a small fixed-size dynamic set of light path pairs within the hair volume. Each pair consists of a *short path* which can be chosen according to a given render time budget, and a *long path* which is an extension of the short path until termination. The MLP is tasked to learn the error between these paths. In the second stage, we trace one short path per pixel from the camera and accumulate its radiance. In the final stage, the error of these short paths is inferred from the MLP and corrected for, resulting in a final radiance accounting for low and high order scattering. This process is repeated every frame which progressively renders the scene. Since the MLP is always trained and inferred on fixed size data, the time required for the first & third stages is constant. This coupled with the fact that short paths can be adjusted for a specific time budget gives an upper bound on the render time.

We derive the quantity learnt by the MLP and show that it is bounded and converges. We then describe a methodology to efficiently train and use the MLP within the rendering loop. We demonstrate that our method renders the true color of the hair due to multiple scattering in a short time as opposed

to path tracing which gradually recovers it. We further show that in the best case, our method achieves a $40\% - 70\%$ speedup in run-time at the cost of minimal bias. We analyse our method extensively and show that it naturally provides a control over its bias and speedup by controlling the maximum path depth β of these short paths. In effect, our method can trade off less bias for more speedup & vice versa.

Fig. 6.1 demonstrates two characteristics of our method on messy white hair: (a) smaller error & time required to trace equal number of samples per pixel as path tracing, (b) controllable bias & speedup.

6.2 Contributions in the context of previous work

Physically-based hair scattering. The seminal work of Marschner and colleagues [76] modeled the scattering of hair as a bidirectional curve scattering function (BCSDF) [77], approximating each fiber as a dielectric filament with circular cross section. Inspired by Marschner's model, several works have been proposed improving its accuracy [78–80], generalizing to elliptical cross sections [81, 82], or making it more practical for production scenarios [83–85]. While advanced hair scattering models can improve the efficiency of rendering, our work is independent of specific a scattering model used and provides a general rendering acceleration solution, with a focus on accelerating computations of multiple scattering between hair strands.

Accelerating multiple scattering. Multiple scattering is responsible of hair coloration, specially in light colored hair, but extremely expensive due to the long light paths and forward scattering. Moon and colleagues proposed to use photon mapping to cache the radiance at the hair volume [70], which was later improved by encoding radiance in spherical harmonics [71]. Another approach for accelerating multiple scattering is to separate light transport into different regimes (ballistic, directionally diffuse, diffuse): Dual scattering [73] approximated multiple scattering by deriving closed-form approximations of the near and far scattering components. Meng et al. [74] and Müller et al. [75] used a similar light regime decomposition in the context of particulate media. Dual scattering however fails to reproduce the soft look and saturation visible in light colored hair. Yan et al. [72] extended dual scattering to fur, by adding a diffusion term. For human hair, their method falls back to dual scattering.

Hery and colleagues [86, Chap.7] proposed to reduce the multiple scattering albedo of hair (i.e. its overall color) by some factor and compensate for the energy loss by multiplying the illumination by the same factor. This effectively reduced the path weight, increasing the possibility of early termination which led to increased efficiency. They demonstrated that by using a factor of two, render times decreased by 45% while achieving a similar look. However, increasing the factor will result in shorter paths with larger deviation from the desired look.

Zhu et al. [87] accelerated multiple scattering in fur by reducing the fur density, and increasing the fiber thickness using learned aggregate scattering behaviour; this approach is similar to the similarity theory [88, 89] where the optical parameters of general media are altered for reducing the density of the medium and therefore reducing the amount of scattering events. However, their approach is not directly applicable to hair with long unordered strands, unlike fur. This makes the aggregation of hair difficult.

Our method takes a different approach, avoiding precomputations and accelerating hair rendering by inferring high-order scattered radiance in hair using an online-trained & small multi-layer perceptron.

Deep Learning in Rendering. A number of works have focused on accelerating global illumination using neural networks. As mentioned above, Zhu et al. [87] aggregate hair strands and learn the aggregated scattering with an MLP. In the context of optically thin, high albedo volumes, Kallweit et al. [90] render atmospheric clouds by training a network to learn the spatial and directional distribution of radiant flux from cloud exemplars. For translucent, optically thick materials, Vicini et al. [91] proposed to learn the Bidirectional Subsurface Scattering Distribution Function (BSSRDF) of a target object with any shape and material properties. Also for translucent objects, Che et al. [92] infer material properties from images by training an auto-encoder with a differentiable Monte Carlo volume renderer as a decoder. All of these methods pre-train neural networks for lower-dimensional sub-problems (the scattering functions). Our approach on the other hand, works directly in the high-dimensional path space, and learns a mathematically well-defined function of the total radiance at a given ray.

With a more generic goal, Müller et al. [93] presented neural importance sampling, an approach for online learning of the directional distributions and path sampling and guiding for simulation of light transport, as well as a neural version of control variates [94]. Closer to our work, Müller and colleagues [3] presented a neural radiance caching technique that approximates a 5D radiance field of a scene. Differently, we tackle the specific case of hair volumes, which exhibit high frequencies both spatially (i.e. there are around 200k hair strands in a human head) and angularly (highly directional, anisotropic scattering). Such properties, which are particularly enhanced for light colored hair where light paths undergo many bounces inside the volume, make the previous techniques either impractical or lack the visual fidelity. Instead of learning the complex 5D radiance field in the hair volume, we train a similar small MLP to learn and infer the error between biased low-order scattered radiance and full unbiased scattered radiance (effectively, high-order scattered radiance).

6.3 Preliminaries

In this section, we revisit the path formulation of the Light Transport Equation (Eq. 2.3) to put it in context of multiple scattering in hair. We also briefly describe Russian roulette, a technique to achieve early termination of low energy paths. Our method is described in Sect. 6.4.

6.3.1 Path Tracing

Recall the path formulation in Eq. 2.3 which models the radiance L arriving at a pixel as:

$$L = \sum_{k=1}^{\infty} \int_{\Omega_k} f'(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}), \tag{6.1}$$

and its corresponding unbiased Monte Carlo estimator:

$$L \approx \langle L \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\bar{\mathbf{x}}^i)}{p(\bar{\mathbf{x}}^i)}, \tag{6.2}$$

where $\bar{\mathbf{x}}^i$ is a randomly generated light path. In case of scenes with hair and especially light coloured hair, the radiance estimate $\langle L \rangle$ is dominated by multiple scattering, which means most of the sampled paths $\bar{\mathbf{x}}^i$ need to be long and should also cover a wide range of path lengths. However the energy contribution of deeper path vertices drops quickly for a few of these lights paths, thus lowering the overall efficiency.

6.3.2 Path Termination with Russian Roulette

Russian Roulette (RR) is used to improve efficiency of $\langle L \rangle$ by probabilistically terminating long paths with low energy. During recursive sampling of path vertices, at each vertex or scattering event $\mathbf{x}_j$ on $\bar{\mathbf{x}}$, the path is terminated by a probability $p_{\text{RR}}(\mathbf{x}_j)$. The integral at the terminated vertex is not evaluated and is zeroed. We note that an estimate that better represents this integral can be used instead [95] which may help to further reduce variance, and our method as is can potentially provide this estimate. There are multiple ways for computing $p_{\text{RR}}(\mathbf{x}_j)$, based on the single scattering albedo at $\mathbf{x}_j$ [96], the expected incident radiance [97,98], or the accumulated subpath weight $w(\bar{\mathbf{x}}_j) = T(\bar{\mathbf{x}}_j)/p(\bar{\mathbf{x}}_j)$ [1], with $\bar{\mathbf{x}}_j = \mathbf{x}_0..\mathbf{x}_j$.

We use RR based on accumulated subpath weight. Typically, RR termination is applied only after the first n path vertices (for example, PBRT [1] uses $n = 3$). However, we apply RR from $n = 1$ for our method and path tracing. We note that our method and formulation is independent of the choice of n.

6.4 Method

In this section, we describe our method to approximate higher order radiance in hair using an MLP. We begin by setting a maximum depth $\beta << \infty$ in Eq. 6.1, giving the radiance L' arriving at the pixel:

$$L' = \sum_{k=1}^{\beta} \int_{\Omega_k} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}). \tag{6.3}$$

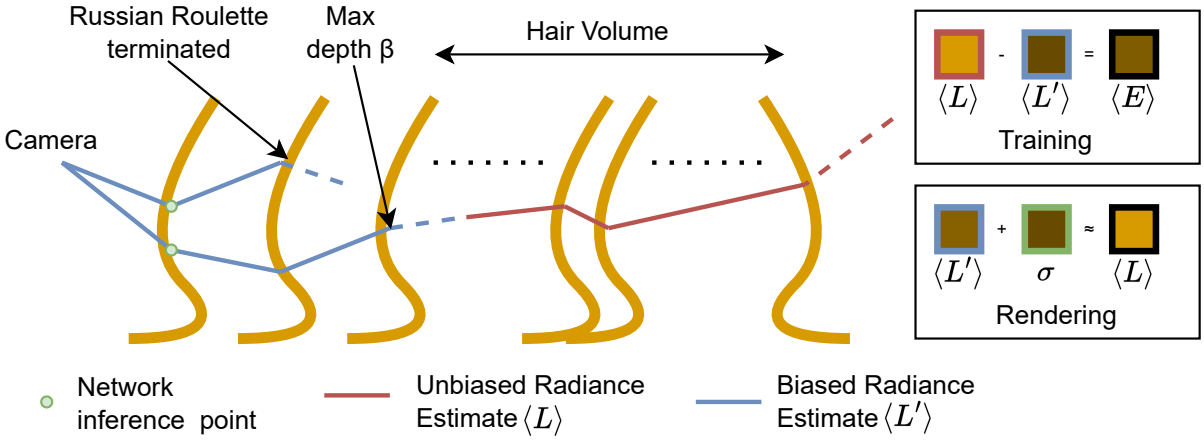


Figure 6.2 We learn the error $\langle E \rangle$ (Eq. 6.4) between radiance estimates of long unbiased paths $\langle L \rangle$ (red, Eq. 6.1) and short biased paths $\langle L' \rangle$ (blue, Eq. 6.3) using an MLP. The short paths are traced until termination by Russian Roulette (RR) or upto a small max depth β and the long paths are traced until termination by RR. The network σ is trained to reproduce $\langle E \rangle = \langle L \rangle - \langle L' \rangle$. During rendering, the radiance at the camera is computed as the radiance of the short path plus the error inferred by the network at the primary path vertex (green). The thick yellow lines depict hair strands.

We estimate $\langle L' \rangle$ inplace of $\langle L \rangle$ during rendering. Modifying Eq. 6.1 in this way effectively sets a maximum depth for path termination, which introduces bias/error in $\langle L' \rangle$, given by:

$$\begin{aligned} E &= L - L' \\ &= \sum_{k=1}^{\infty} \int_{\Omega_k} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}) - \sum_{k=1}^{\beta} \int_{\Omega_k} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}) \\ &= \sum_{k=\beta+1}^{\infty} \int_{\Omega_k} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}), \end{aligned} \tag{6.4}$$

which we can estimate using the Monte Carlo estimate $\langle E \rangle$ in the same fashion as Eq. 6.2. This error estimate $\langle E \rangle$ is bounded and converges in expectation, and captures the missing higher-order scattering component in $\langle L \rangle$. In effect, $\langle E \rangle$ represents the error between the expected value of unbiased path contributions & the contribution from early terminated paths. Furthermore, $\langle E \rangle$ is itself unbiased, since its form is similar to Eq. 6.2, except that the paths $\bar{\mathbf{x}}$ are instead sampled from $\Omega \in \{\Omega_{\beta+1}, \Omega_{\beta+2}, ..., \Omega_{\infty}\}$.

We task an MLP σ to learn $\langle E \rangle$ at the primary path vertex $\mathbf{x}_1$, given the view direction $\omega = \frac{\mathbf{x}_0 - \mathbf{x}_1}{||\mathbf{x}_0 - \mathbf{x}_1||}$ and the hair tangent $\mathbf{t}_1$ at $\mathbf{x}_1$:

$$\sigma(\mathbf{x}_1, \omega, \mathbf{t}_1) \approx \langle E \rangle. \tag{6.5}$$

Fig. 6.2 shows an overview of our method, which computes $\langle E \rangle$ to train the network σ and uses the network's inferred output at the primary path vertex to correct for the bias in $\langle L' \rangle$.

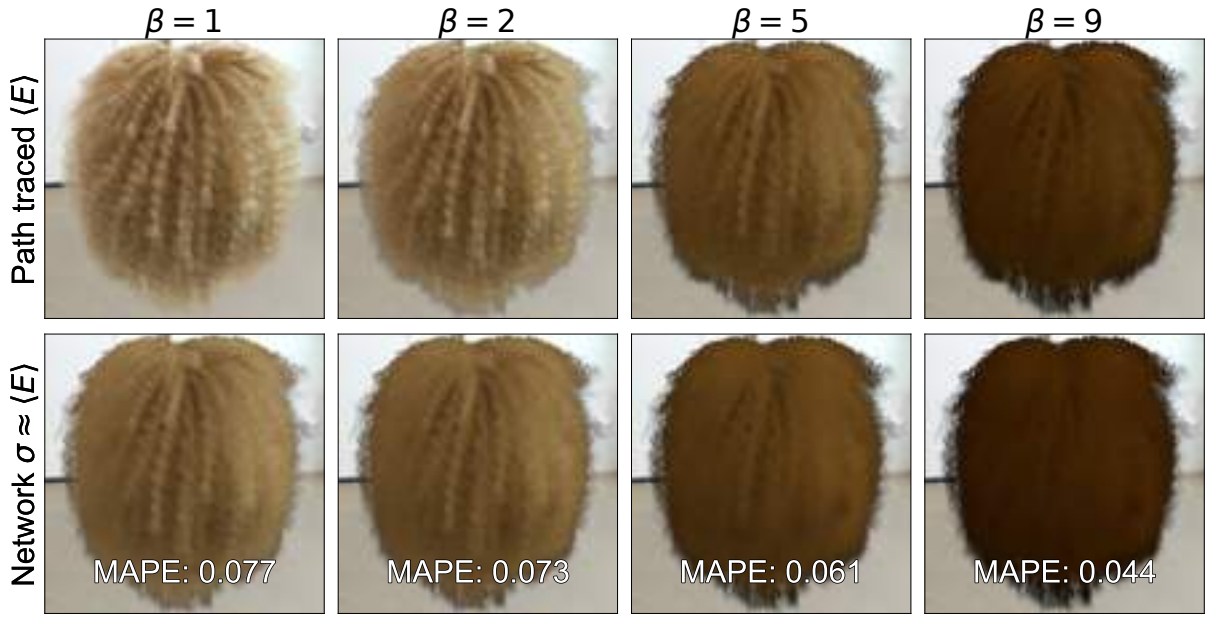


Figure 6.3 Visualization of the target function $\langle E \rangle$ and its learned approximation for $\beta = 1, 2, 5$ and 9. For lower values of β, the target function has higher frequencies which the network is unable to reproduce. At higher β's, the target function is lower frequency and the network can better represent it, which is also confirmed by the MAPE values. The full rendering can be found in Fig. 6.4

Discussion. We aim to avoid precomputations, which precludes training the MLP σ offline. Therefore, limited by the time constraints of online training and inference, we use a small MLP. Furthermore, the computation of $\langle E \rangle$ to train the network is done on a small percentage ($\sim 1\%$) of the total paths traced per frame. In Sec. 6.5, we describe our implementation to efficiently train σ during rendering.

Learning an estimate of the error $\langle E \rangle$ instead of directly learning an estimate of the full radiance field has two major advantages:

- The frequency of $\langle E \rangle$ over paths $\bar{\mathbf{x}}$ starting at $\mathbf{x}_0$ and with a shared common first hair vertex $\mathbf{x}_1$ is low: The magnitude of L' closely follows the magnitude of L for their single sample MC estimates thanks to RR termination, given that these estimates share the same path vertices. This makes $\langle E \rangle$ smooth over paths and of lower magnitude.

- The estimate $\langle E \rangle$ is a high-order radiance estimate, and thus has lower spatial frequency compared to $\langle L \rangle$, which has high frequencies as a result of complex primary visibility within hair and high dynamic range BCSDF lobes. This also follows from the observation that high order scattering in hair is diffuse-like [70, 76, 78].

We demonstrate this by visualizing the target signal $\langle E \rangle$ and its learned approximation with σ in Fig. 6.3, for $\beta = 1, 2, 5$ & 9. For lower β's, $\langle E \rangle$ has higher frequencies (highlights) due to the specular & directional nature of the hair BCSDF. Although the network is able to sufficiently capture spatial variation, it does not reproduce these highlights. For higher β's, the frequency is attenuated and the

network is better able to reproduce the target signal. This is also confirmed by the mean absolute percentage error (MAPE) [99] values computed with respect to $\langle E \rangle$, which are lower for higher β's. Thus, choosing $\langle E \rangle$ as the target for the MLP makes it robust and better represent the signal, more so since we are limited to a small MLP with limited learning capacity.

Our second goal of controllable bias and efficiency is also naturally achieved by means of the parameter β. For large β, $\langle E \rangle$ has even lower frequency and amplitude (since the error vanishes) and thus the network's output closely matches the ground truth. In the limit for very large β, $\langle E \rangle$ approaches zero, and our method is equivalent to unbiased path tracing. This comes at the cost of efficiency, since the rendering cost with $\langle L' \rangle$ approaches that of $\langle L \rangle$, both consisting of long paths. On the other hand, for a very small β, $\langle E \rangle$ increases in frequency, starting to reach the network's learning capacity. This not only results in an overall increase in bias, but the efficiency also increases since rendering with $\langle L' \rangle$ now requires shorter paths to be traced.

In summary, learning the *error* instead of the full radiance field makes the network learning robust and better match the target signal. Furthermore, our formulation for learning this error naturally allows for control over the bias and speedup. In the next section, we describe our implementation for efficiently training σ while rendering & give details on the network structure.

6.5 Implementation

We implement our approach using CUDA and OptiX [68] for hardware accelerated ray tracing and use tiny-cuda-nn [100] for efficient network evaluation and training. We use path tracing as a baseline within the same framework. For better convergence, we use Quasi Monte Carlo with Sobol random number generation for our method as well as path tracing. Furthermore, all direct lighting computations use multiple importance sampling [4] with BCSDF and environment map importance sampling. Alg. 7 shows the pseudo code for rendering a frame using the inferred error from the network, while also training it per frame.

Training. The training procedure is given in lines 1-10. We train the network on paths originating from randomly sampled points that are visible to the camera. Given a sampled point x on hair, we trace two paths: A short path with maximum depth β, and a long path which extends this short path until termination (lines 2-6). This is depicted by the use of the same random number sequence for tracing these paths (line 4). In practice, the computations of the short path are reused. Note that both short and long paths are terminated by RR, although short paths are strictly terminated at bounce β. The sampled short and long paths are used to estimate Eq. 6.3 and Eq. 6.1 respectively. We then compute the error (Eq. 6.4) in line 7, and update the network weights using the relative L_2 luminance loss [3, 101] (lines 8-10). We use the same training setup (optimizer, learning rate) as [3].

Rendering. Rendering a frame consists of two stages, which are described in lines 17-28 in Alg. 7. In the first stage, the network is trained for one second (lines 19-21), ensuring that large errors from an untrained network are not propagated into the final render. A crucial point to note is that this training is

ALGORITHM 7: Rendering a frame while training the MLP

```
1  Def trainNetwork(β, x):
2      t = getHairTangent(x)
3      ω_o = norm(cam.origin - x)
4      ξ = randomSequence()                                   // Seq. of rand. num
5      L = pathTrace(x, ω_o, k ∈ {1, ∞}, ξ)                   // Eq. 6.1
6      L' = pathTrace(x, ω_o, k ∈ {1, β}, ξ)                  // Eq. 6.3
7      E = L - L'                                             // Eq. 6.4
8      E' = σ[hashGrid(x), ob(ω_o), ob(t)]                    // MLP σ forward
9      loss = L_2^rel(E, E')                                  // Rel. L2 luminance
10     loss.backward()                                        // Update weights of σ
11 Def renderPixel(pix, β):
12     /* Trace short path & get G-Buffer                                      */
13     L', G_buf = pathTraceCam(pix, k ∈ {1, β})              // Eq. 6.3
       /* Evaluate MLP σ on G-Buffer                                           */
14     E' = σ[hashGrid(G_buf.x), ob(G_buf.ω_o), ob(G_buf.t)]
       /* Add inferred error                                                   */
15     L = L' + E'
16     return L
17 Def renderFrame(β: Max. depth, SPP: spp, N: # Train Samples):
18     /* Init. training for 1 sec                                             */
19     while time ≤ 1sec do
20         x = sampleVisible()
21         trainNetwork(β, x)
       /* Render Loop                                                          */
22     for sample in SPP do
23         for n in N do
24             x = sampleVisible()
25             trainNetwork(β, x)
26         for pix in PIXELS do
27             L = renderPixel(pix, β)
28             accumulate(pix, L)
```

frame independent, which allows for more initial training iterations and a better converged network. The next stage is responsible for both training the network and rendering the frame. After the initial training and before a sample is traced, the network is trained on a fixed size (size N), small & dynamically sampled set of path pairs (lines 23-25), starting from a visible point. Using a small set is crucial since training is repeated for every sample that is traced. In our implementation, we set the number of training samples $N = 16,384$, which is 1% of our render resolution of 1024×1024. After training, for each pixel we trace short paths with maximum depth β and estimate Eq. 6.3 (line 26, 27, 13). We also record a G-buffer at the primary intersection of the camera rays. We then evaluate the network to get inferred error estimate E' and add it back to the biased short path estimate L' (lines 14-16).

Network structure. We use a 64-neurons wide and 2-layers deep MLP σ with ReLU activations (except the last layer). We use a multi-resolution hash grid (hashGrid) [102] for encoding the 3D point

x, and one-blob (ob) encoding [93] for the view direction ω_o and the hair tangent t (Alg. 7, line 14). Multi-resolution hash grids can better represent the target signal than frequency encoding with a minimal performance penalty, and have also been shown to work well for radiance caching [102].

Discussion. Since we use OptiX and CUDA in our implementation, the loops in line 19, 23, 26 of Alg. 7 are parallelized across GPU cores. Thus, the function *sampleVisible* returns a buffer of visible points on hair. Consequently, the function *trainNetwork* operates on this buffer in one single pass. Similarly, the function *renderPixel* operates on a buffer of all image pixels.

6.6 Results, Analysis & Comparison

In this section, we show the rendering results of our method and analyse its characteristics. We also compare against Neural Radiance Caching [3], which uses a similar MLP for inferring the full radiance field, and Dual Scattering [73] which is a real-time method for rendering multiple scattering in hair. We demonstrate better qualitative & quantitative results than both these approaches.

To evaluate our method on a spectrum of challenging scenes, we use three different hair styles: Curly, Messy & Straight, and two different HDR lighting setups: Indoor & Outdoor. Table 6.1 shows the statistics of different hair styles. Rendering is done with Disney's hair BCSDF model [85] for four absorption coefficients resulting in different hair colors: White $\sigma_a = (0.01, 0.01, 0.01)$, Blond $\sigma_a = (0.06, 0.1, 0.2)$, Brown $\sigma_a = (0.2, 0.3, 0.5)$ and Black $\sigma_a = (3.35, 5.58, 10.96)$. We note that significant multiple scattering occurs with small σ_a values and our goal is to accelerate its computation. Thus, we majorly show results on Blond & White hair. Results on other σ_a values are shown for completeness. We also note that renderings of our method are generated by initially training the network for the one second (Sect. 6.5). For a fair comparison, we start path tracing and all other methods without the initial one second gap (i.e. start rendering from 0 seconds), unless otherwise stated. Furthermore, the network continues training for each sample that is traced, as mentioned in Sect. 6.5.

6.6.1 Rendering Results

High sample count, equal spp. Fig. 6.4 shows our results at various values of β at 5k samples per pixel (spp) compared to path tracing for the same sample count. We also show false color difference images and report the MAPE metric with respect to a path traced reference rendered at 10k spp. We also show render times (in seconds) for both methods. This comparison illustrates the behaviour of the bias introduced by our method for a high spp and also demonstrates the bias-speedup control. At $\beta = 1$, our method achieves the maximum speedup of $\sim 60\%$ with respect to path tracing. However, the MAP error is also larger due to higher bias from the network, as also shown by the difference images. In general, for low values of β, the error is larger at deeper points in the hair volume or where there is significant multiple scattering. This error is largely due to the network reaching its learning capacity and averaging its output. At higher values of β, our method produces results that closely match path tracing, with

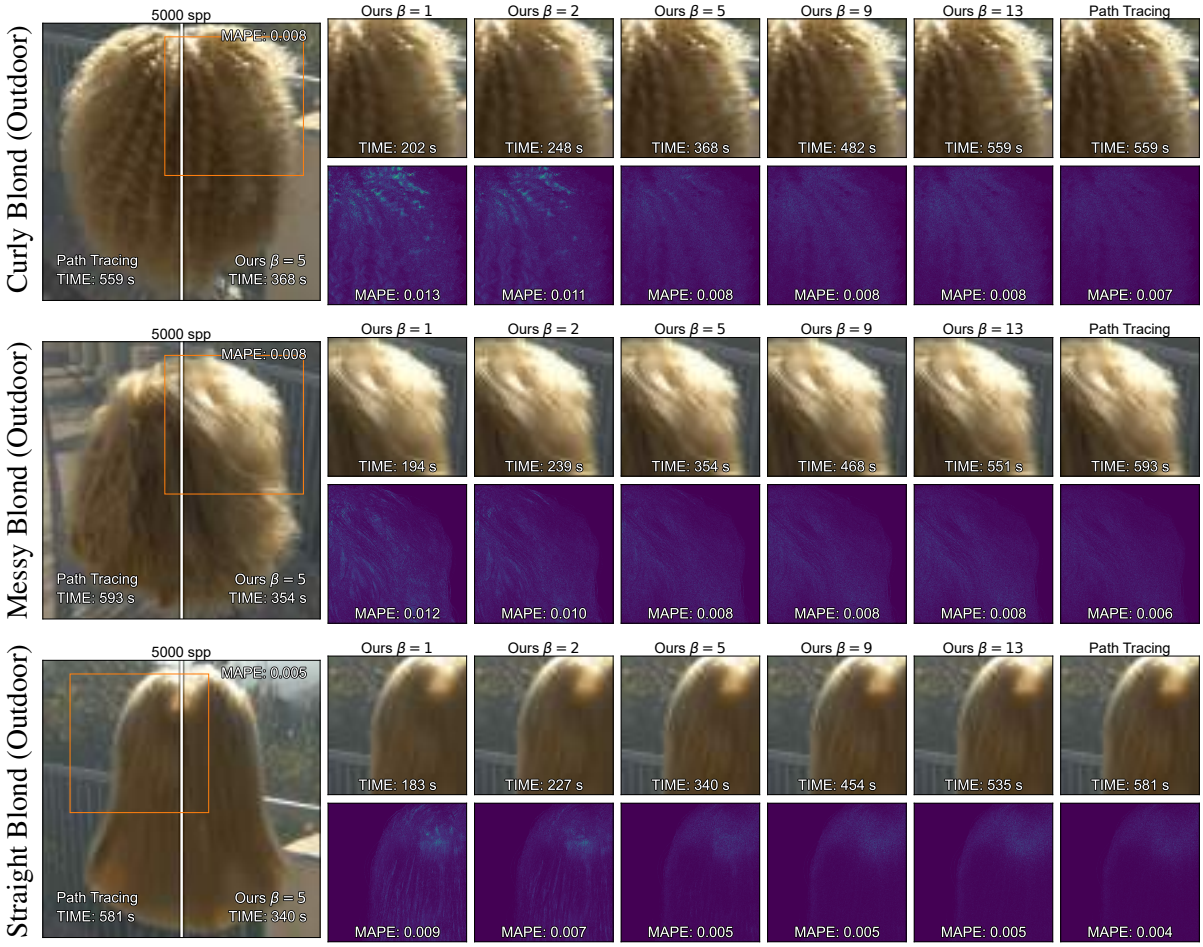


Figure 6.4 We show results of our method for $\beta = 1, 2, 5, 9$ & 13 and compare to path tracing at 5k spp, along with false colour difference images with respect to path traced reference (10k spp). We also report time (in seconds) and the MAP error for all variants. Our method converges in error to unbiased path tracing for large β, while exhibiting larger bias at lower values. On the other hand, at lower β, the time taken for 5k spp is halved to that of path tracing, increasing for larger values. This demonstrates the control that our method provides over it's bias & speedup.

similar run-time. In the end, after a certain threshold of β (e.g. 13), our method gains little benefit and could lead to increased run-time from the additional overhead of network training and inference steps.

Low sample count, equal spp. We show equal spp (100 spp) renderings of our method on different scenes in Fig. 6.5. This figures serves to illustrate the benefit of our method at low spp. For white hair which has the most multiple scattering, our method with $\beta = 1$ achieves about one-third the error in about one-fourth the time. These values approach that of path tracing with increasing β. With blond hair and in general darker hair, the gain in run-time and error decreases, albeit still being significant. On black hair our method matches the performance and quality of path tracing for all β values. Visually, our method has lesser noise and takes lesser time to trace 100 spp compared to path tracing, especially for

Figure 6.5 Equal spp (100 spp) renders of our method for $\beta = 1, 2, 5, 9, 13$ compared to path tracing. For white hair, our method with $\beta = 1$ achieves one-third the error in one-fourth the time. These metrics approach path tracing with increasing β. For darker hair, the gain is less pronounced but still significant. Our method behaves equivalently to path tracing on black hair.

light coloured hair. Table 6.1 consolidates the run-time and MAPE values for blond hair with different hair styles rendered for 100 spp with indoor HDR lighting.

Equal variance. We analyze the convergence characteristics of our method by comparing equal levels of variance for different β against path tracing (Fig. 6.6). For each β, the variance is computed as

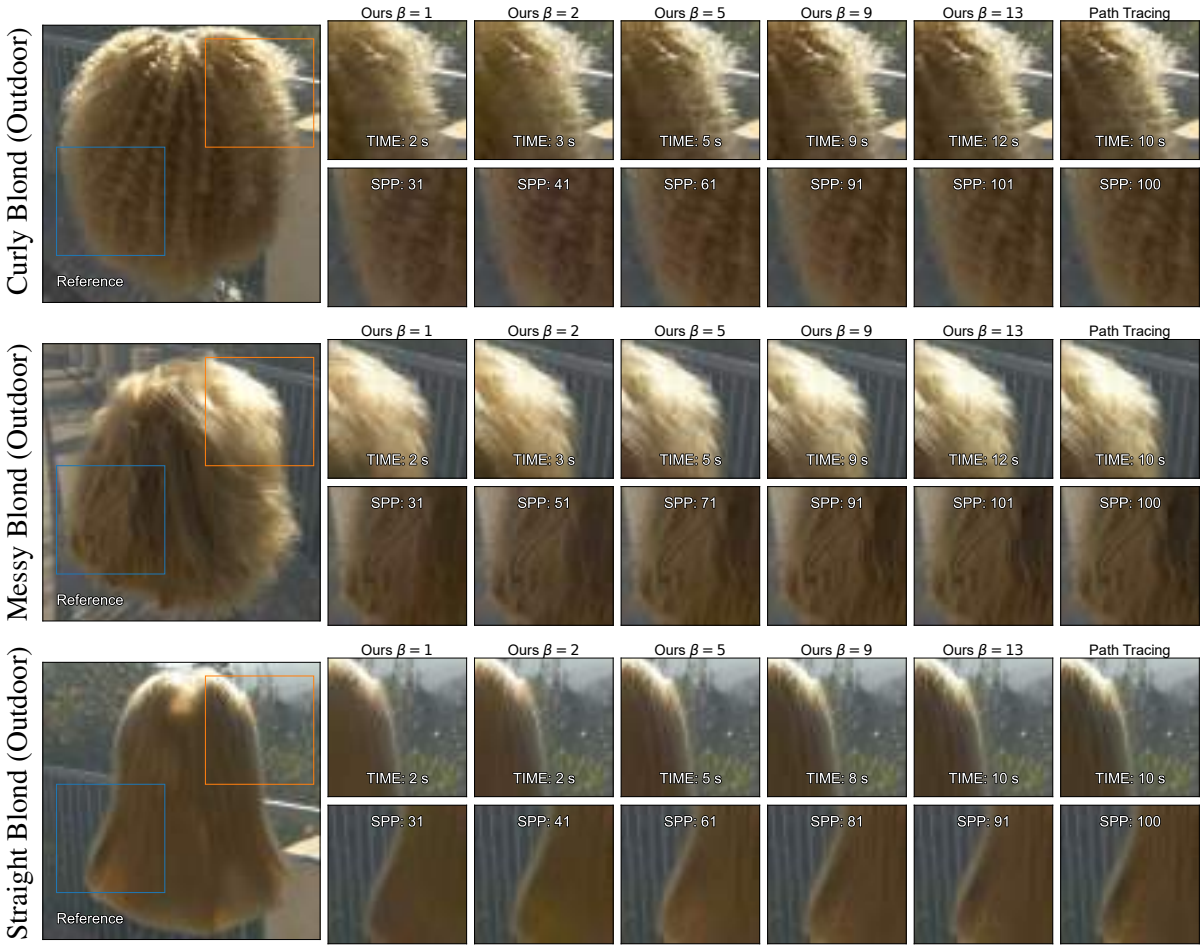


Figure 6.6 We calculate variance of path tracing at 100 spp, and show results, timings & spp required for our method with $\beta = 1, 2, 5, 9$ & 13 to reach the same variance. The variance of each method is computed as the mean-squared error (MSE) with it's own fully converged counterpart. This figure in combination with Fig. 6.4, illustrates the effect of β to trade bias in favor of performance, and shows that our method achieves better performance than path tracing for mid-range β values.

the render's mean-squared error (MSE) with respect to it's own fully converged counterpart (10k spp). This figure in combination with Fig. 6.4, illustrates the effect of β to trade bias in favor of performance. For lower β, our method can achieve the same levels of variance as the path tracing with much fewer spp and lower time. This is at the cost of progressively introducing more bias as β decreases, that nevertheless remains reasonable at its peak $\beta = 1$, depending on the target application. Our method achieves better performance for both low and high β until up to very high values, where obviously the learning overhead does not compensate for the very little energy left, in which case the performance of path tracing is better.

Convergence. To further analyze the convergence characteristics, we plot the MAP error against time for our method and path tracing in Fig. 6.7. We also similarly plot the MAP error against spp in

Table 6.1 Number of strands of different hair styles, along with run-times & MAPE of our method with $\beta = 1$ and Path Tracing (PT) for 100 spp. Metrics are calculated with blond hair $\sigma_a = (0.06, 0.1, 0.2)$ and indoor HDR lighting. Our method converges to around half the error in half the time compared to path tracing.

		Blond (Indoor), 100 spp			
		Ours ($\beta = 1$)		PT	
Hair Geom.	# Strands	Sec.	MAPE	Sec.	MAPE
Curly	60k	**4**	**0.028**	10	0.064
Messy	100k	**4**	**0.026**	10	0.056
Straight	100k	**4**	**0.020**	10	0.046

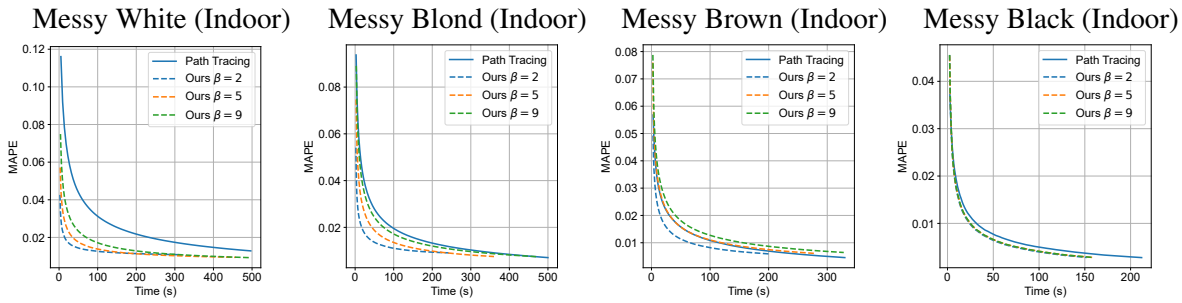


Figure 6.7 Equal time convergence graphs with the MAP error metric. Our method showcases the most benefit for light hair, where the convergence is significantly faster than path tracing for lower β values. For higher values, the convergence starts to approach path tracing. For darker hair, the convergence is closer to that of path tracing, for all β's.

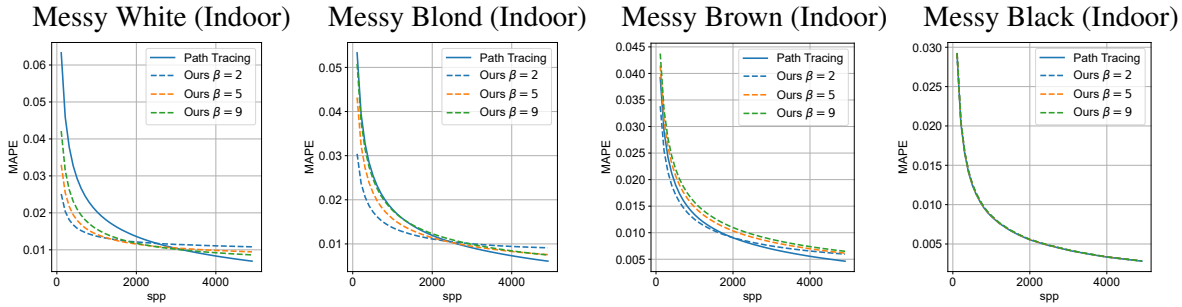


Figure 6.8 Equal spp convergence graphs with the MAP error metric. Our method converges to a lower error for the same spp, especially for light hair. These graphs also show that at high spp, our method converges to a biased result, and this bias can be reduced by using higher β values.

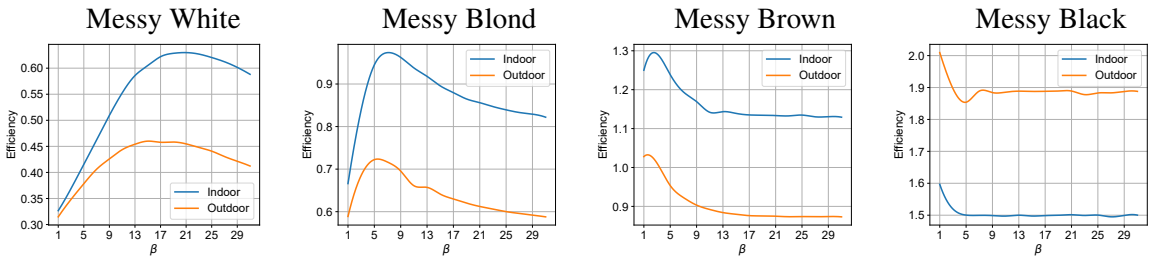


Figure 6.9 We plot β vs. efficiency graph to analyse the characteristics of our method. Efficiency is computed as $\frac{1}{\text{time}\cdot\text{error}}$ with time in seconds & the MAP error. For white hair, maximum efficiency is achieved at larger $\beta \in \{13, 25\}$. This peak shifts as the hair gets darker. Within a hair style, the maximum efficiency depends on the lighting condition, however the peak is roughly in a similar β range.

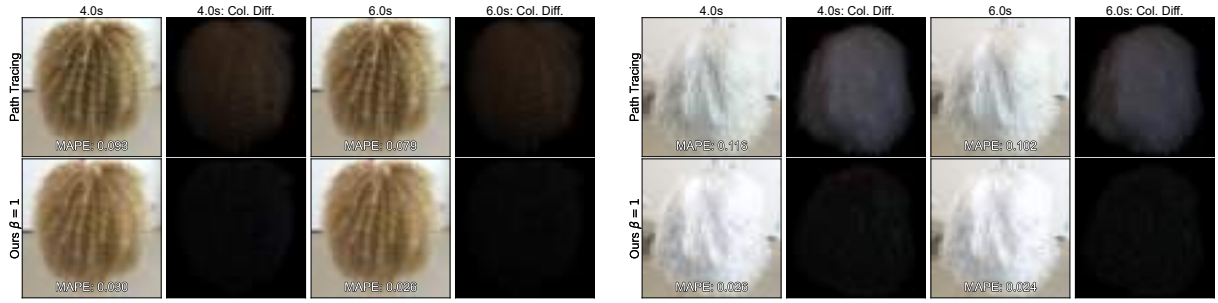


Figure 6.10 Our method is beneficial at low spp & small render times to get an accurate estimate of the true color of hair, as shown in the color difference images which have a larger color component for path tracing.

Fig. 6.8. These graphs are plotted for messy white, blond, brown & black hair styles for $\beta = 2, 5, 9$. Note that, as mentioned before, we train our network for one second before rendering starts, while the rendering is started immediately for path tracing. These plots show that in general, our method converges to a lower error faster, especially for light hair, even with the additional training delay at the beginning. Note however that for larger values of β, our method's convergence approaches that of path tracing, as at these values the paths are longer and there is less bias from the network. Since path tracing converges to an unbiased solution, unlike our method, there always exists a time at which the error of path tracing will crossover to a lower value than that of our method. This is shown in Fig. 6.8, where for very large spp, the error of path tracing is lower. For darker hair, both extremes of β result in similar convergence closely following that of path tracing. To that end, our method is less beneficial for darker hair.

Efficiency. To further study our method's characteristics with respect to β, we analyze it's efficiency for 5k spp renders with respect to a 10k spp reference. We compute efficiency as $\frac{1}{\text{time}\cdot\text{error}}$, with time in seconds and the MAP error. Note that efficiency is typically computed using *variance* instead of *error*; however, we use the latter to account for both bias and variance, which helps us evaluating which value of β achieves the maximum speedup with the least bias. The plots are shown in Fig. 6.9 for messy white,

Table 6.2 Decomposition of timings of our method and path tracing (PT). For our method, we show timings for tracing rays upto a max path depth β along with training and inference timings. All timings are in milliseconds (ms) and averaged over ten runs.

	Ours time (ms) for 1spp					
Hair Geom.	$\beta = 2$	$\beta = 5$	$\beta = 9$	Train	Infer	PT
Curly	46	69	96	3	0.05	112
Messy	44	69	89	2	0.06	117
Straight	45	64	97	2	0.06	116

blond, brown & black hair styles, each with two different lighting setups (indoor & outdoor). For white hair, maximum efficiency is achieved at larger $\beta \in \{13, 25\}$. This peak shifts as the hair gets darker, ultimately being the most efficient at $\beta = 1$ for black hair. Within a hair style, the maximum efficiency depends on the lighting condition (max. value is different for indoor & outdoor lighting). However, the peak is roughly in the same β range for both lighting types.

Progressive, equal time. We demonstrate another benefit of our method at low spp & small render times: to get an accurate visual estimate of the true color of hair. Fig. 6.10 shows equal-time renders of our method with $\beta = 1$ compared to path tracing, sampled at different time budgets, for curly blond and messy white hair. We also show a color difference image with respect to a 10k spp path traced reference. Path tracing gradually recovers the color with time as more samples are traced. This can be seen in the difference images where they have a larger color component for path tracing. On the other hand, the difference images of our method show that the difference is majorly in the intensity. Indeed, as compared to path tracing, our method produces the multi-scattered color the hair from the start. This is useful in situations where the parameters of the hair need to be constantly adjusted to achieve a desired look. In such situations, it is beneficial to have an accurate estimate of how the hair looks from the start, allowing quicker iteration.

Upper bound on run-time. By fixing β, we get an estimate of the maximum number of rays that need to be traced for each pixel, giving an upper bound on render time. This coupled with the fact that the network training and inference take constant time (since $N = 16,384$ for training, and $N = 1024 \times 1024$ for inference, Sect. 6.5, Alg. 7), we get an accurate estimate of the total time required to render a 1spp frame. Table 6.2 shows timings to render one spp for the three hair styles, averaged over ten runs. Note that the training & inference timings of our method are the same irrespective of β, as mentioned above.

6.6.2 Comparison with previous work

Fig. 6.11 shows 5k spp renderings of our method for two values of $\beta = 1, 5$ in comparison to 5k spp renders of Neural Radiance Caching (NRC) [3], dual scattering [73] & a 10k spp path traced reference. We also compare to a modified version of NRC, referred to as *NRC++*.

Dual Scattering. Dual scattering is a real-time hair rendering method that efficiently approximates multiple scattering. We use it in the offline context with ray-shooting, as described in their paper. In

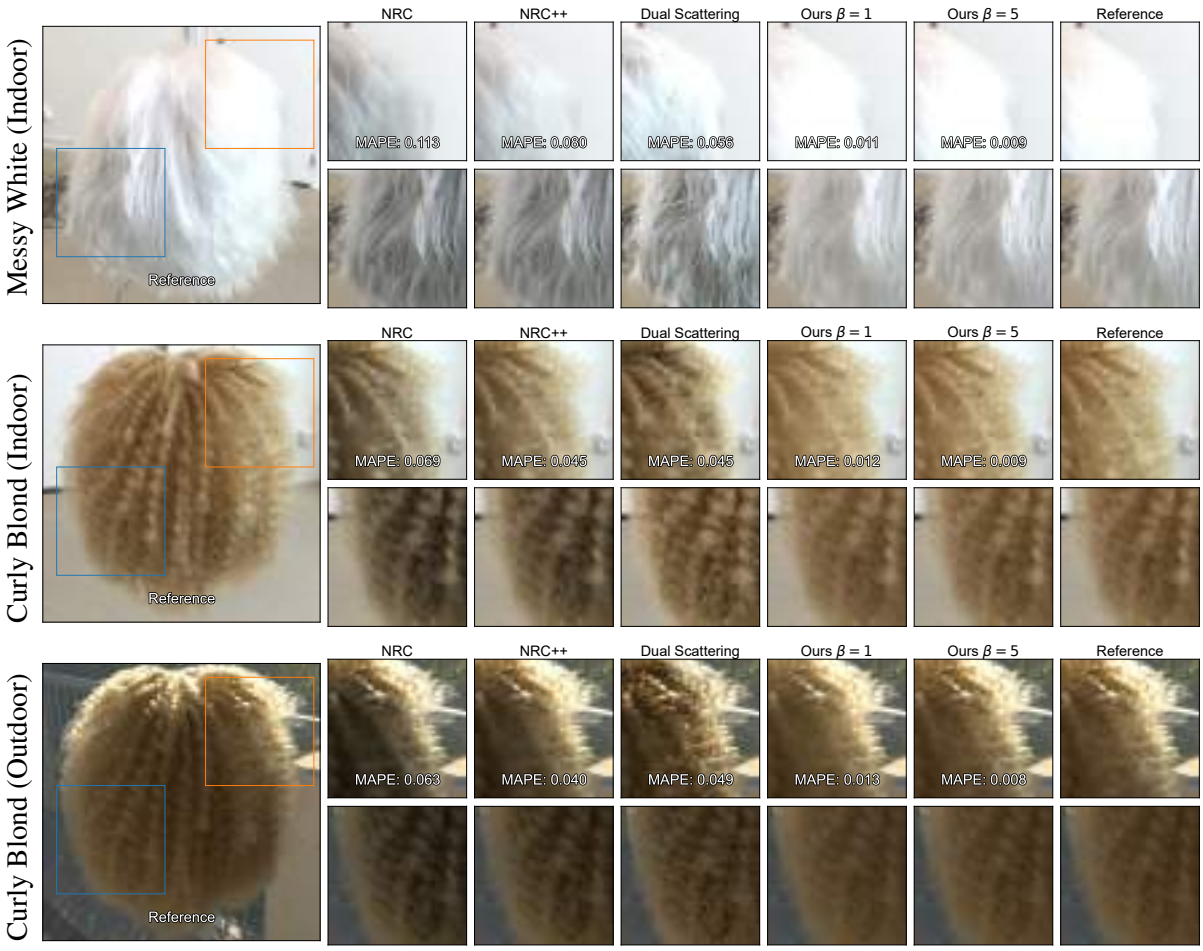


Figure 6.11 We compare 5k spp renderings of our method for $\beta = 1, 5$ to 5k spp renderings of Neural Radiance Caching (NRC) [3], dual scattering [73] & a 10k spp path traced reference. We also compare to a version of NRC where all the training paths are unbiased (*NRC++*). All these methods fail to reproduce the soft look and saturation in hair. Our method with $\beta = 1$, which is the most efficient but also the most biased, not only achieves a lower MAP error in most cases, but also reproduces the saturation and soft look.

essence, at the primary path vertex, we shoot multiple rays towards the light source and apply dual scattering for each ray and average their radiance to obtain the final color. Renders using dual scattering are unable to reproduce the soft look and miss a significant component of multiple scattering, especially for light hair (Fig. 6.11, top row). For darker hair, the renders are slightly better, but still have considerable bias, as depicted by the MAPE values.

NRC. NRC shares a similar approach to ours: They also use a small MLP [100] to efficiently compute global illumination. However, our approach differs in the following:

- We learn on the *final* accumulated radiance, instead of radiance at each path vertex. The latter is useful in surfaces (NRC's target application) with energy quickly degrading deeper in the path, and results in more training data for the same number of paths. However, in hair, this leads to averaging in the network due to excessively long path lengths.
- We train on higher order radiance (E in Eq. 6.4), which has much less frequency than the full radiance field. The small MLP is thus able to represent the target signal better (Fig. 6.3, Sect. 6.4).
- Our method & formulation provide explicit control over the render's bias & speedup. The bias in NRC however cannot be directly controlled.

As shown in the figure, NRC renders do not faithfully reproduce the color saturation from multiple scattering.

NRC++. NRC takes inspiration from Q-learning and trains the neural network on it's own output. Only a small percentage of their training paths are truly unbiased. We found that using this approach leads to extremely short paths altogether, which are unable to capture higher-order scattering in hair. We thus also compare to a version of NRC where *all* the training paths are traced in an unbiased fashion. We refer to this version as *NRC++*. As shown in Fig. 6.11, NRC++ is better able to capture the higher-order energy, as compared to NRC which is darker, thanks to better training data. On the other hand, renders of dual scattering result in similar MAPE values as NRC++.

All of the above methods (NRC, NRC++, dual scattering) fail to reproduce the soft look, saturation & multiple scattered component in hair. Our method with $\beta = 1$, which is the most efficient but also the most biased, not only achieves a lower MAP error in all cases, but also reproduces the saturation and soft look.

6.6.3 Summary

Our method is most beneficial in light hair where it achieves a significant speed & error reduction for small spp. Furthermore, the amount of bias induced by our method can be controlled with β, where larger values result in similar performance & quality as unbiased path tracing. We show that there exists a value of β that achieves the maximum efficiency for a given hair style. The parameter β also gives an upper bound on our method's run-time. This achieves both goals stated in Sect. 6.1. For darker hair, our

method behaves very similarly to unbiased path tracing and has little benefit, suggesting that one could automatically adjust β based on the luminance of the albedo or even turn our method off with dark hairs and $\beta > 9$. Finally, the most biased variant of our method with $\beta = 1$ has a consistently lower error compared to NRC, NRC++ and dual scattering. Visually, our method reproduces the saturation & soft look better than previous approaches (Fig. 6.11).

6.7 Conclusions, Limitation & Future Work

We presented an approach to efficiently compute the multi-scattered radiance in hair by learning error between biased short paths and unbiased long paths using an MLP. We described an implementation that efficiently and robustly trains this MLP on the fly, while rendering. We demonstrate the ability of our approach to provide a control over the bias & speedup, specifically allowing to trade unbiasedness for gain run-time and vice versa. We thoroughly analyzed our method and demonstrated that it achieves a speedup of $40\% - 70\%$ with respect to path tracing at the cost of a little bias. We also demonstrated that our method achieves the true color of hair from the get-go, which is useful in look-dev situations. Finally, we compared against an existing general technique that incorporated small MLPs to estimate global illumination (NRC) and with an approximate hair rendering technique for recovering multiple scattering (dual scattering), and showed that our method qualitatively & quantitatively outperforms both.

A limitation of our approach stems from the initial training: training the network from scratch for each frame in an animation setting may lead to temporal artifacts. Although this could be alleviated by choosing a proper β, it needs further investigation. Another limitation is that since the network output is used directly, it may exhibit patterns or colour inaccuracies (Fig. 6.6 first & last row, Fig. 6.5 first row). This is less visible when the render converges, but nonetheless needs to be carefully handled. Typical production settings have multiple characters and thus multiple hair grooms in a single frame. For such cases, using a single network with our approach may not be feasible due to network size & learning capacity limitations. One can imagine the use multiple networks for each hair groom and index the corresponding network for different pixels, depending on which groom is visible. However, this remains to be investigated & efficiently engineered.

For future work, we would like to gracefully handle temporal inconsistencies and efficiently render multiple hair-grooms to make our approach truly suitable for production. We would also like to push this technique towards the real-time realm, while maintaining the visual fidelity. Finally, as mentioned in Sect. 6.3.2, the integral at the last vertex of a path terminated by Russian Roulette can be better estimated, potentially using our network's output, to further reduce variance. This is an interesting direction that can be explored for path tracing of general scenes.

Chapter 7

Conclusion

This thesis presents contributions on analytic methods for direct light transport in virtual scenes. We sidestepped the noise and convergence issues of stochastic techniques by restricting light transport integrals to a form where semi-analytic solutions are possible.

Building on Linearly Transformed Cosines (LTCs), which is a semi-analytic method to compute direct lighting, we first presented a fully-analytic method for this computation. This involved carefully computing the visible region of the light source though a series of algorithms that we proposed. Our method also works on arbitrary area lights, as opposed to polygonal area lights by using silhouette edges. We showed that for a certain category of scenes, our method has better efficiency than Monte Carlo path tracing and exhibits no noise.

Our second contribution focused on a core assumption of LTCs that limited its use on surfaces having isotropic GGX as it's material model. We looked at the core LTC method and pointed out the exact reasons why this assumption was made. We proposed fixes and exploited core properties of LTCs that allow their efficient usage for (semi-)analytically rendering anisotropic appearances. Our contribution improves the generality of LTCs and is applicable to all methods that use LTCs at their core.

The third contribution in this thesis exploited the core theory of Linearly Transformed Spherical Distributions (LTSDs), specifically for enabling semi-analytic single scattering with area lights. Our motivation to work with LTSDs was due to the fact that they are the underlying theory behind LTCs. We analysed the difficulties of fitting LTSDs to phase functions and presented simple solutions for those. We also formulated ratio estimators for volumetric scattering which, when combined with our analytic in-scattering and semi-analytic air-light evaluations, resulted in an interactive & competitive semi-analytic rendering approach for single-scattering with area lights.

Finally, we explored the interesting yet challenging application of neural networks to accelerate path tracing. The specific usecase we focused on was the rendering of human hair. We analysed the shortcomings of a Neural Radiance Caching (NRC) whose direct application to hair rendering was not feasible. Specifically, we showed that NRC for hair rendering results in missing energy and soft looks that is typically expected from lighter-coloued hair. Our analysis led to effective and simple modifications to the core of NRC, resulting in photorealistic hair rendering. Additionally, we added a control knob to the resulting renderer, allowing control over it's bias and speedup gained.

7.1 Future Directions

Semi-analytic methods for light transport benefit from no noise and better compute efficiency as compared to Monte Carlo approaches, given reasonable restrictions and assumptions. Semi-analytic methods proposed thus far, including those proposed in this thesis ignore more spatially varying participating media, subsurface scattering and multi-lobed BRDFs to name a few. Developing semi-analytic methods to handle these more general cases is an interesting future direction to explore.

In the neural space, we believe its important to engineer solutions that can render more complex scenes than just a single hair volume. Generalising and increasing the network capacity to learn that kind of light transport, without adding too much computational overhead is an interesting future avenue to explore.

Finally, the author believes that the best acceleration can be achieved when analytic, neural and stochastic approaches are combined for rendering.

Chapter 8

Supplementary

8.1 Greiner-Hormann Algorithm

We provide the implementation details of the *setDifference* function used in 3. The implementation uses the Greiner-Hormann polygon clipping algorithm. This algorithm relies on a doubly linked list of vertices to represent polygons. Consecutive nodes in the list point to the next vertex, defining an implicit order, which is used in the algorithm. It considers two polygons, which are denoted as the *subject* and the *clip* polygon (the subject polygon is clipped) and proceeds in three phases. Detailed steps and the vertex datastructure are given in Fig. 8.1. The first phase is responsible for the determination and storage of edge intersection points of the subject and the clip polygon. The algorithm terminates if no-intersection points are found. Note that, the subject polygon could also be completely inside the clip polygon and vice-versa, which can be easily determined. The second phase is responsible for marking *entry* and *exit* points of intersection vertices of each polygon, which is done by looping over each vertex of a polygon and testing whether it *enters* or *leaves*. Note that this assumes contiguous ordering of polygon vertices, which we perform in the *SphPolySilhouette* function (Alg. 4). In the final phase, the previous computations are used to filter out the clipped polygon. This is done by jumping to the other polygon's vertex at each *entry* or *exit* vertex, starting from a random vertex of the clip polygon, effectively tracing the clipped polygon edges (Fig. 8.1, right). For more details, please refer to the original paper [25].

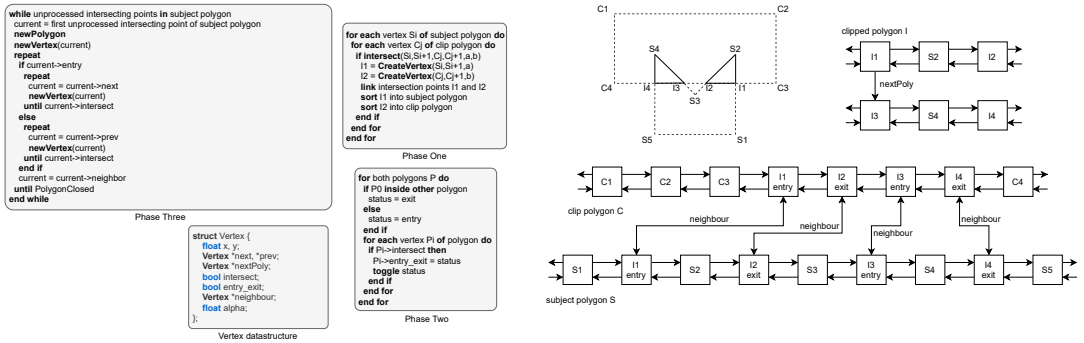
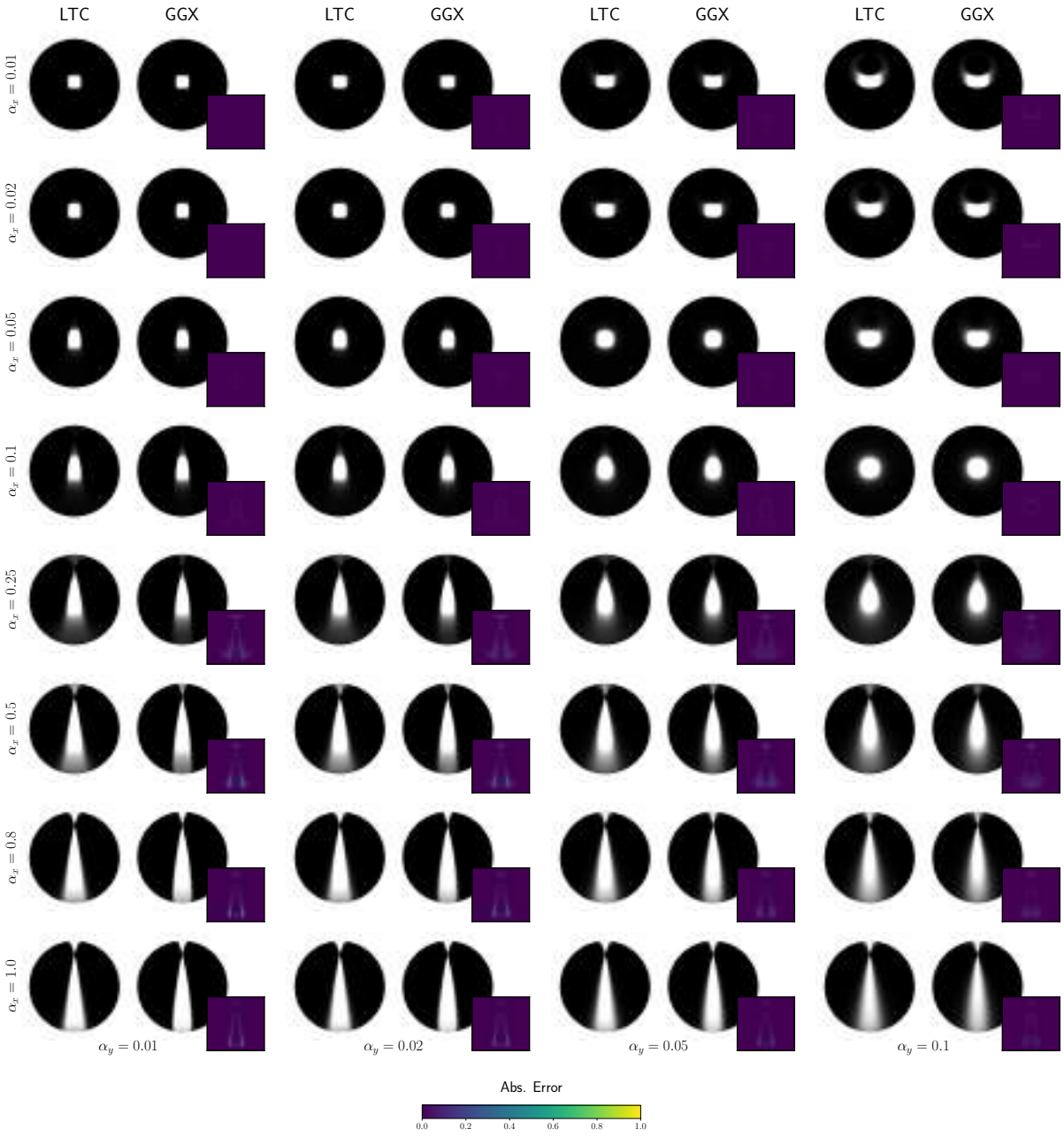
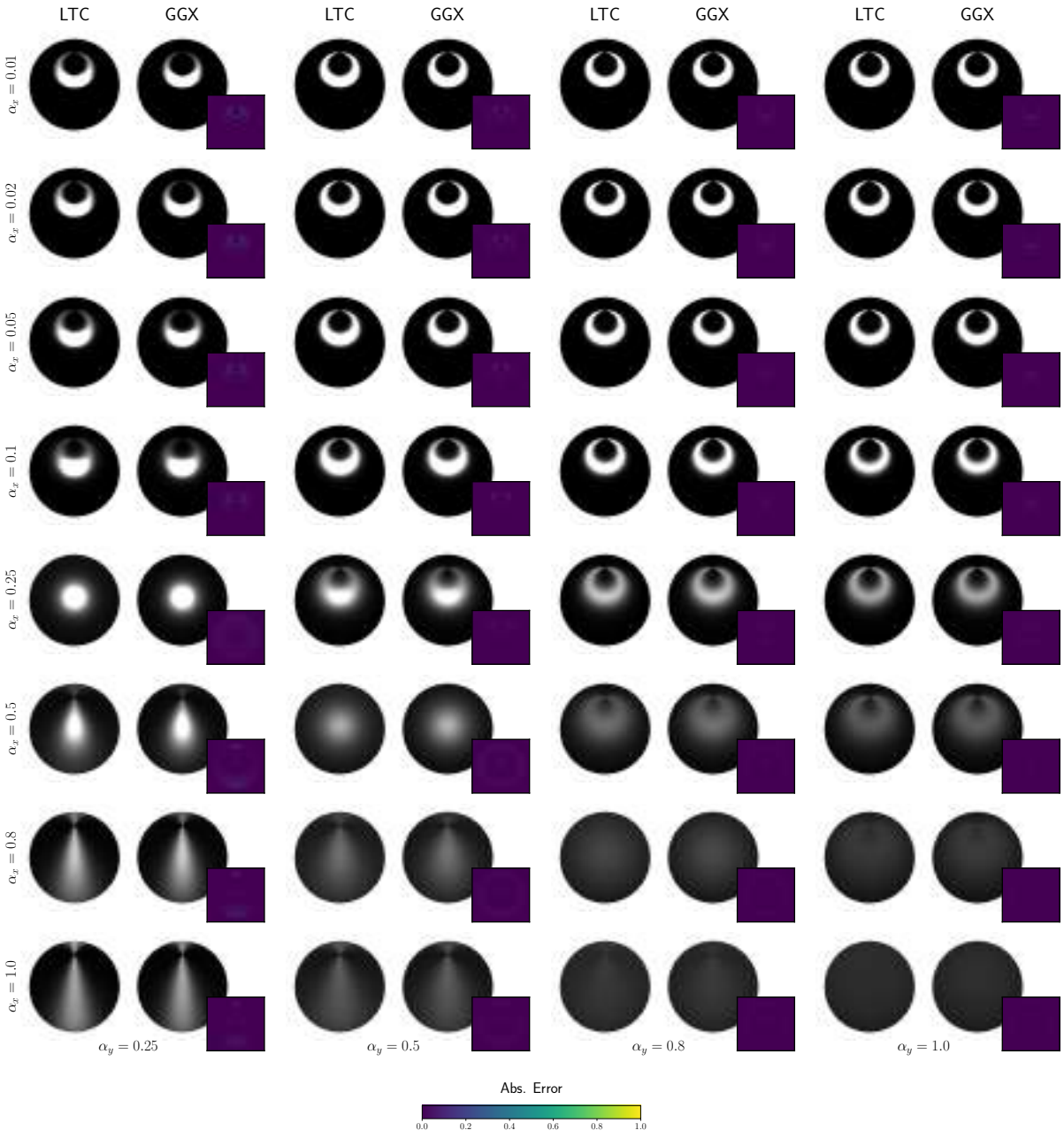


Figure 8.1 Left: Pseudocode for the three phases of Greiner-Hormann polygon clipping algorithm along with the datastructure used to represent vertices. **Right:** Example of a doubly linked list of the Vertex datastructure, representing the clip and subject polygons. The intersection points (I1, I2..) are generated by phase one, entry and exit is marked by phase two and the final clipped polygon (top right) is generated by phase three. The algorithms and the example are directly adapted from [25].

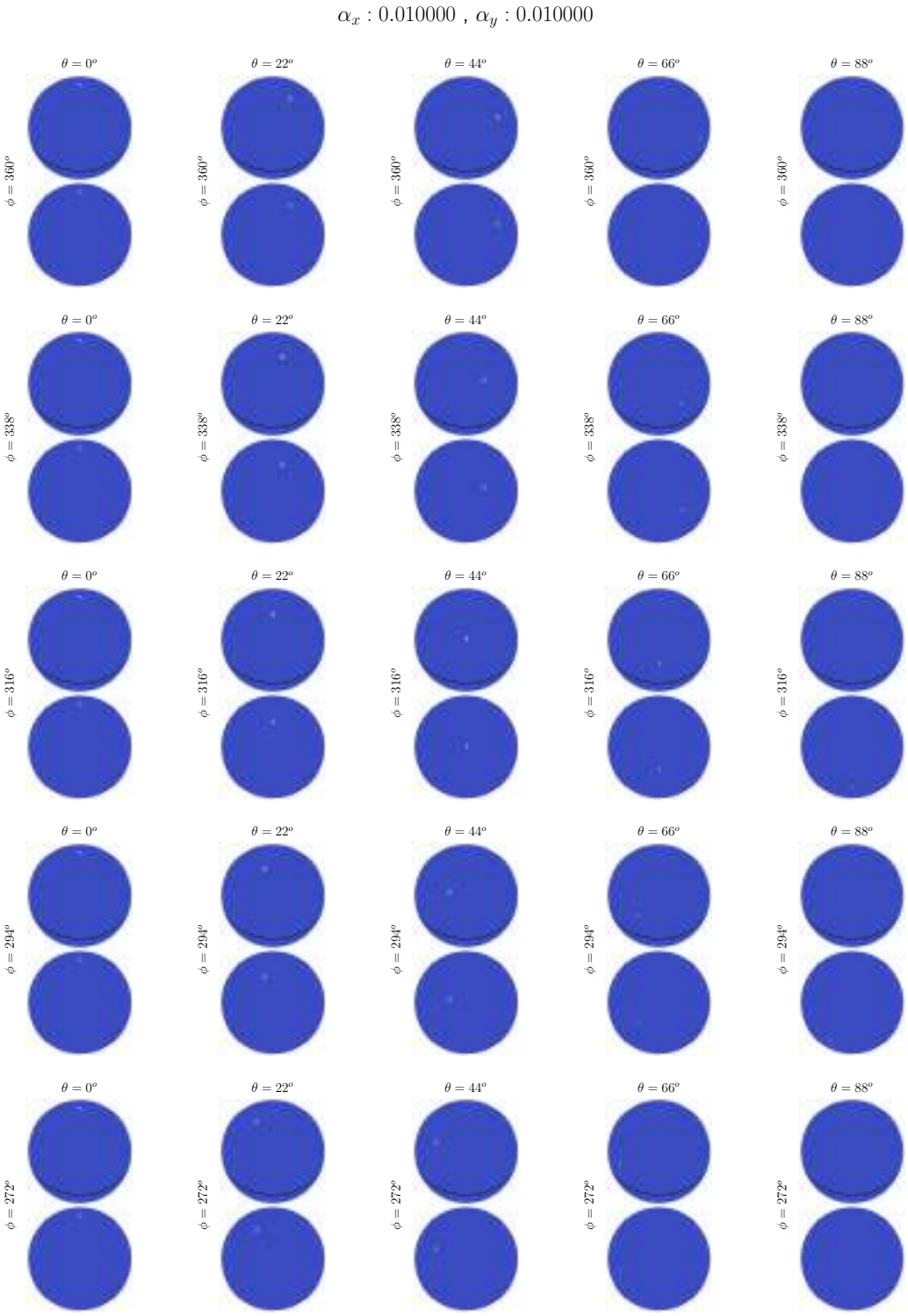
8.2 Additional Results for Chapter 4

8.2.1 Area-light shading

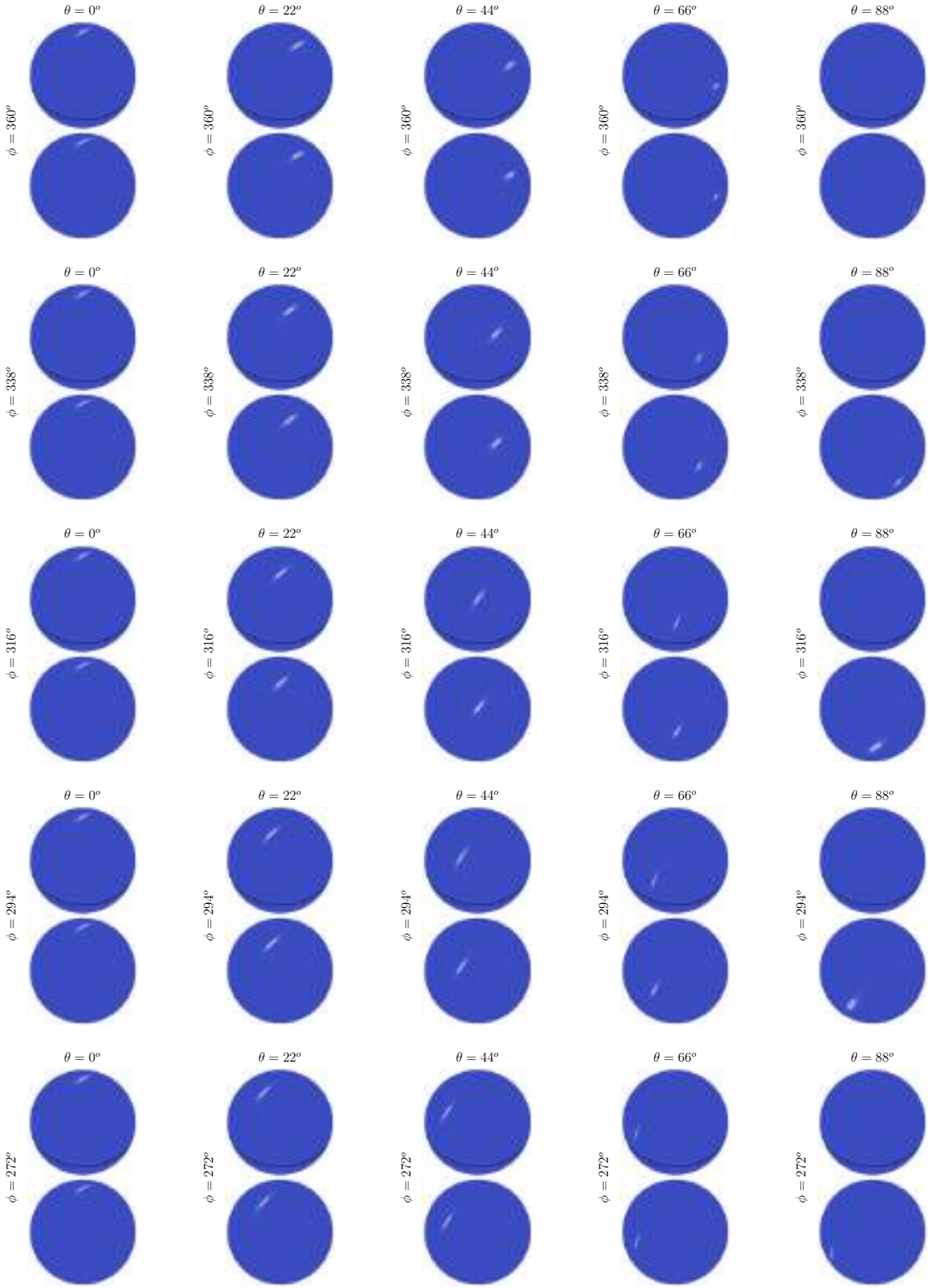



LTC
GGX
LTC
GGX
LTC
GGX
LTC
GGX
$\alpha_x = 0.01$
$\alpha_x = 0.02$
$\alpha_x = 0.05$
$\alpha_x = 0.1$
$\alpha_x = 0.25$
$\alpha_x = 0.5$
$\alpha_x = 0.8$
$\alpha_x = 1.0$
$\alpha_y = 0.25$
$\alpha_y = 0.5$
$\alpha_y = 0.8$
$\alpha_y = 1.0$
Abs. Error
0.0
0.2
0.4
0.6
0.8
1.0

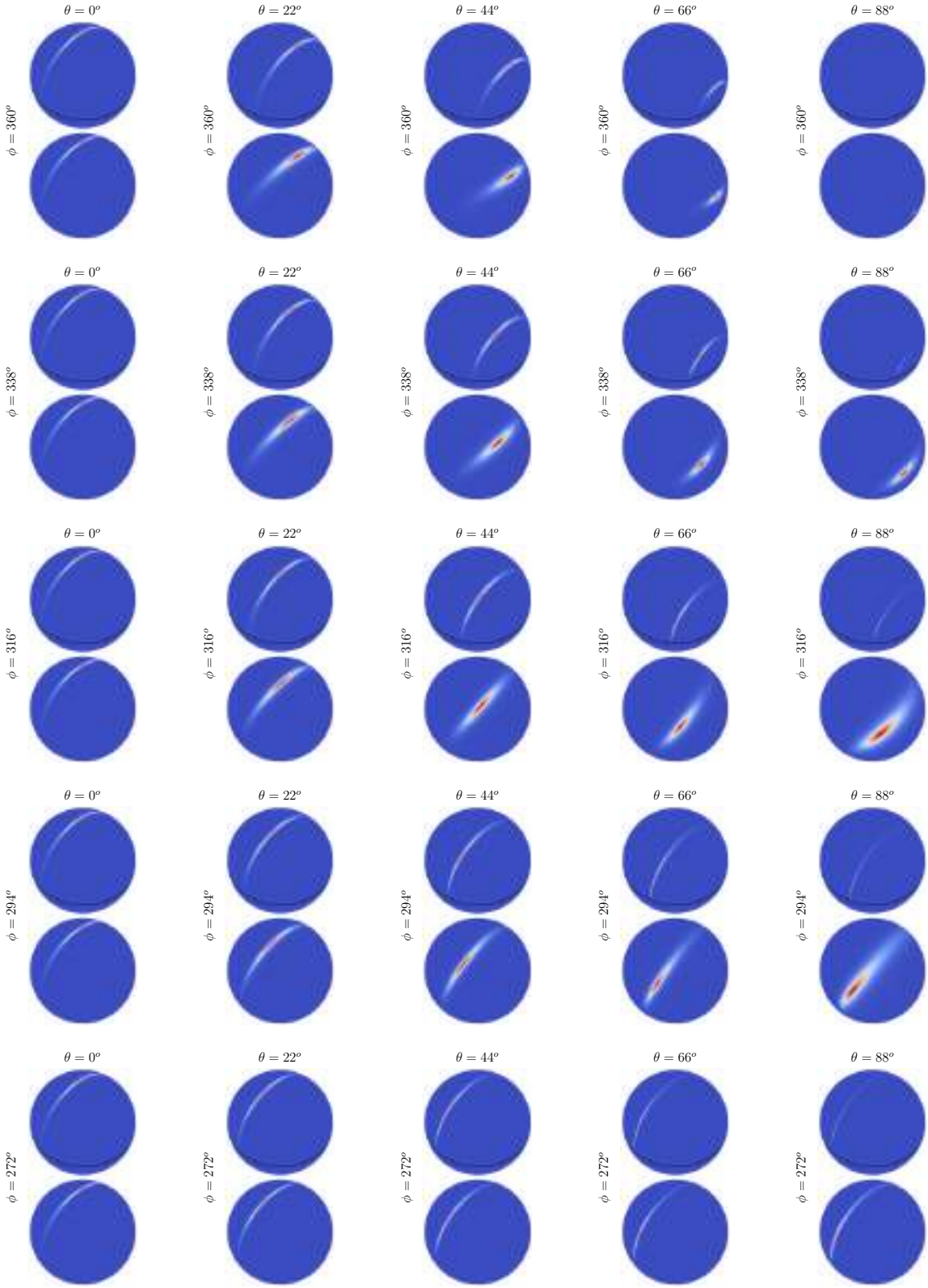
8.2.2 Fitting Results



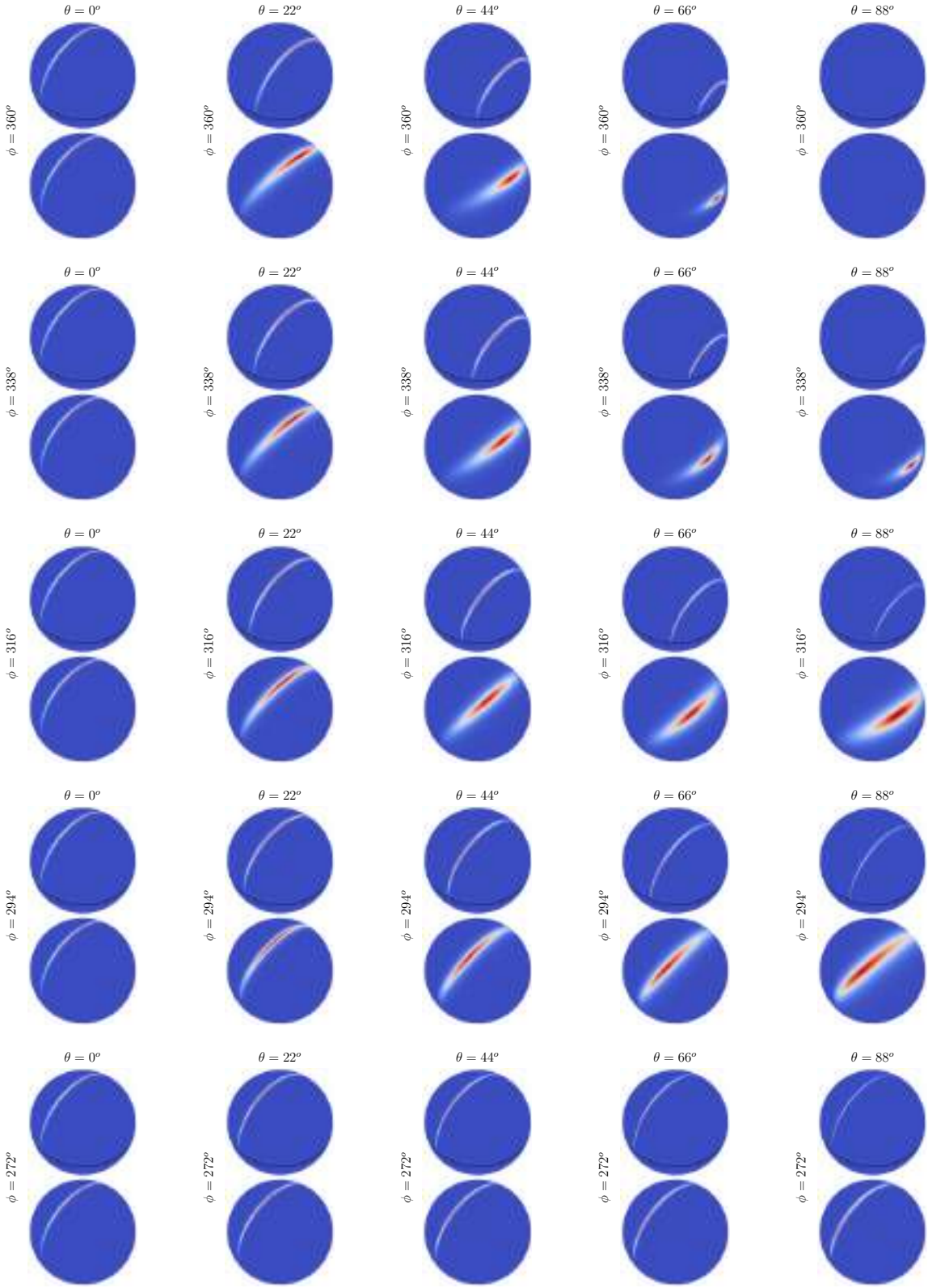
$\alpha_x : 0.010000$, $\alpha_y : 0.050000$



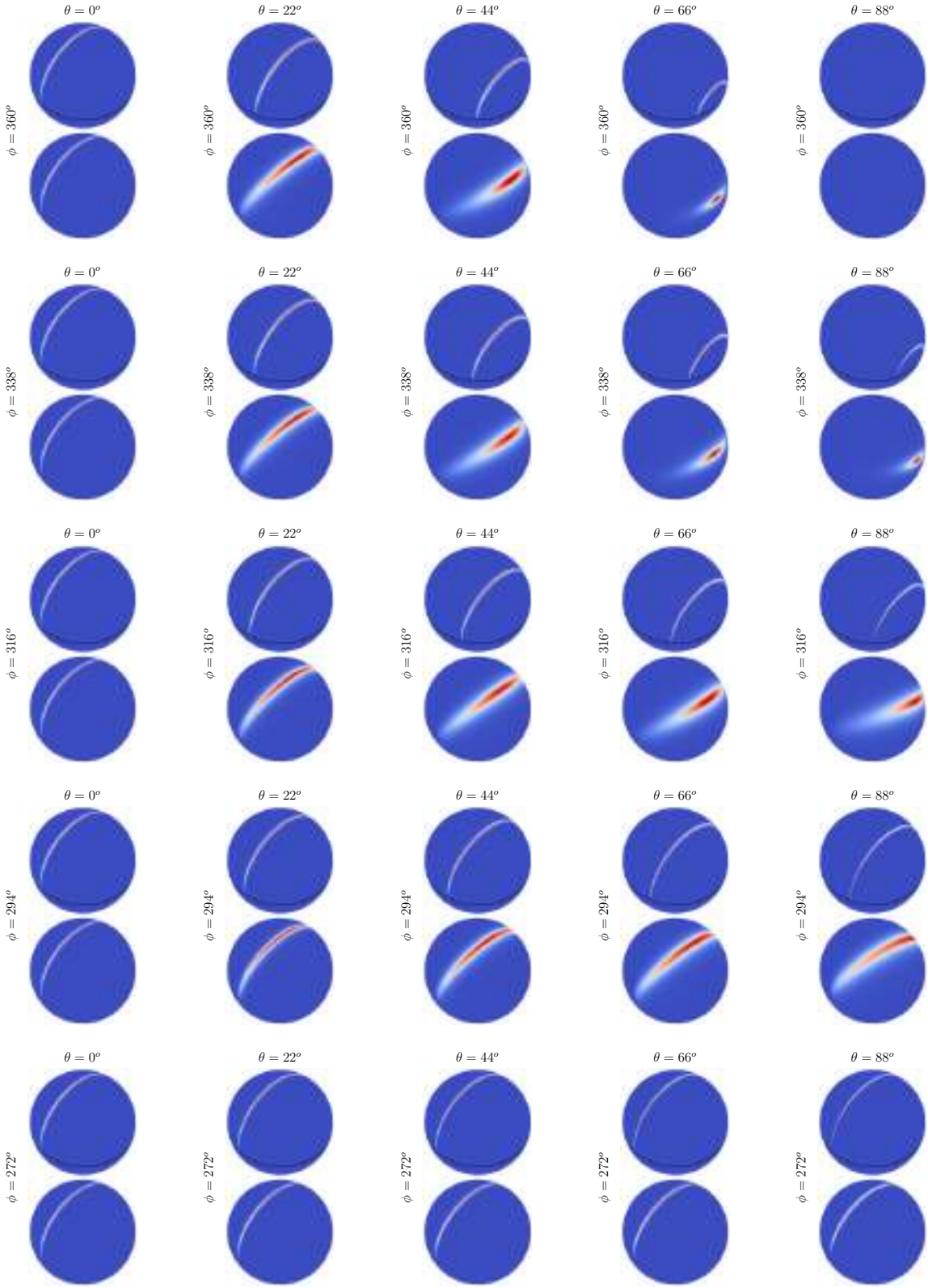
$\alpha_x : 0.010000$, $\alpha_y : 0.250000$



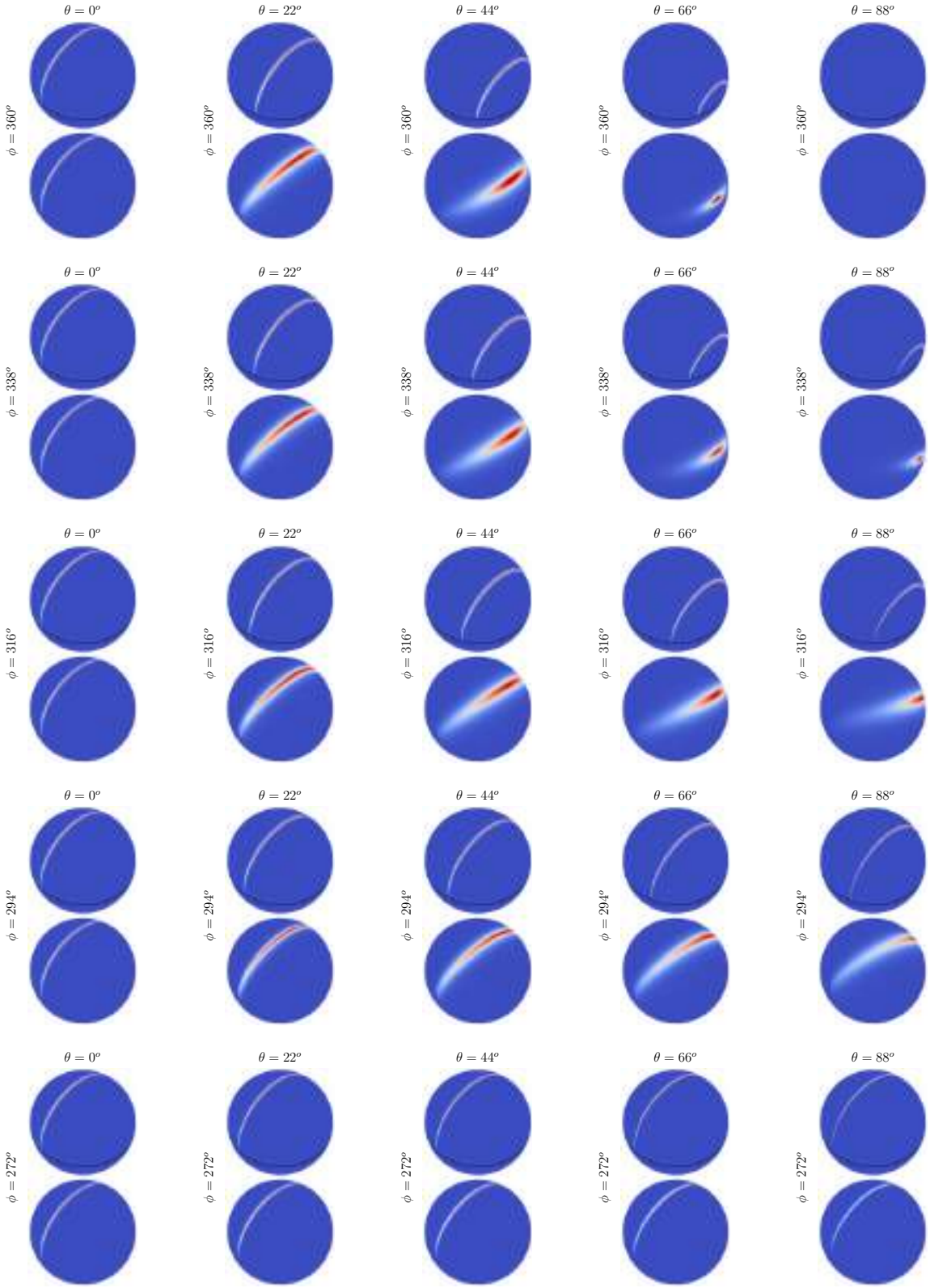
$\alpha_x : 0.010000$, $\alpha_y : 0.500000$



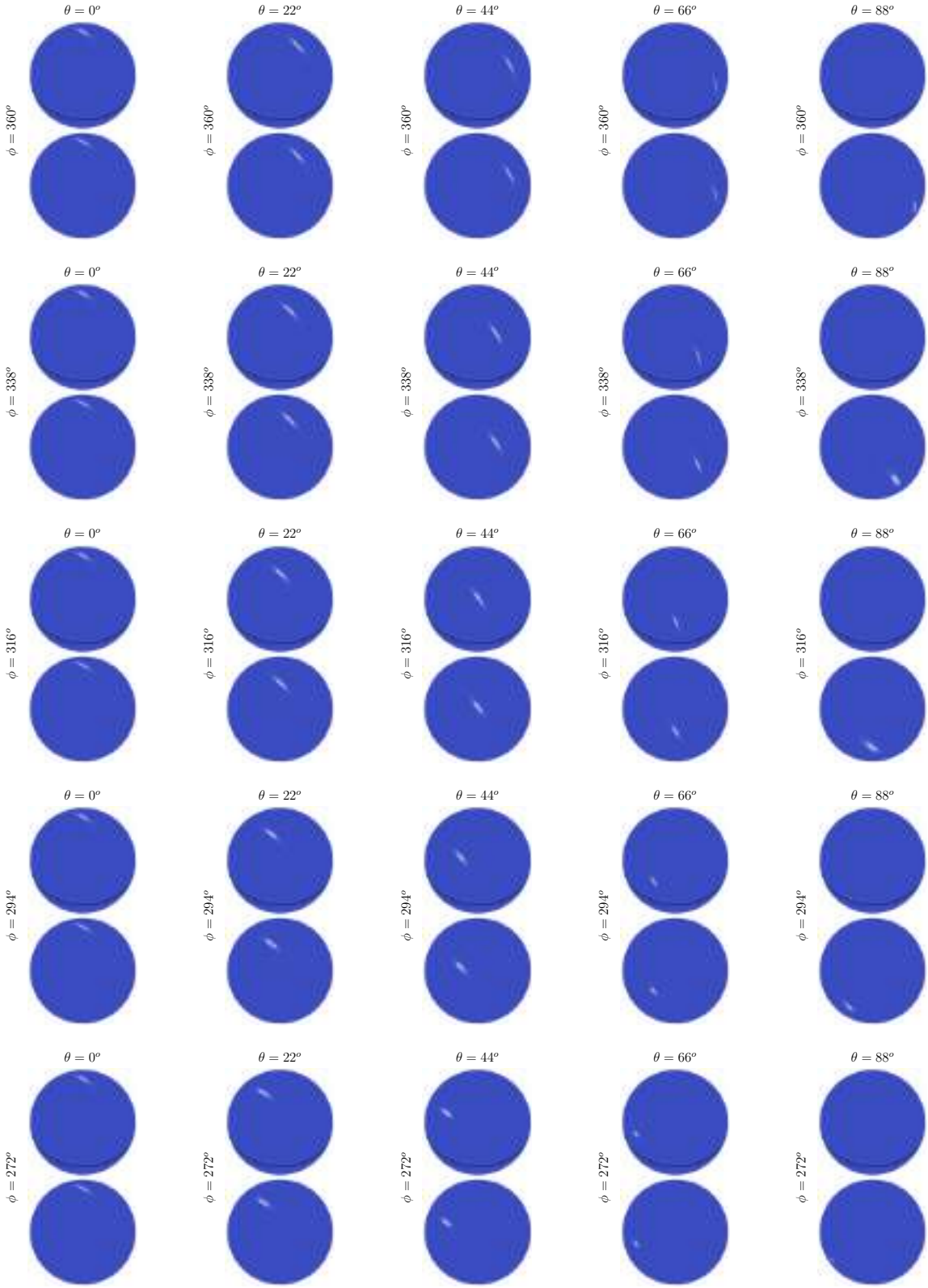
$\alpha_x : 0.010000$, $\alpha_y : 0.800000$



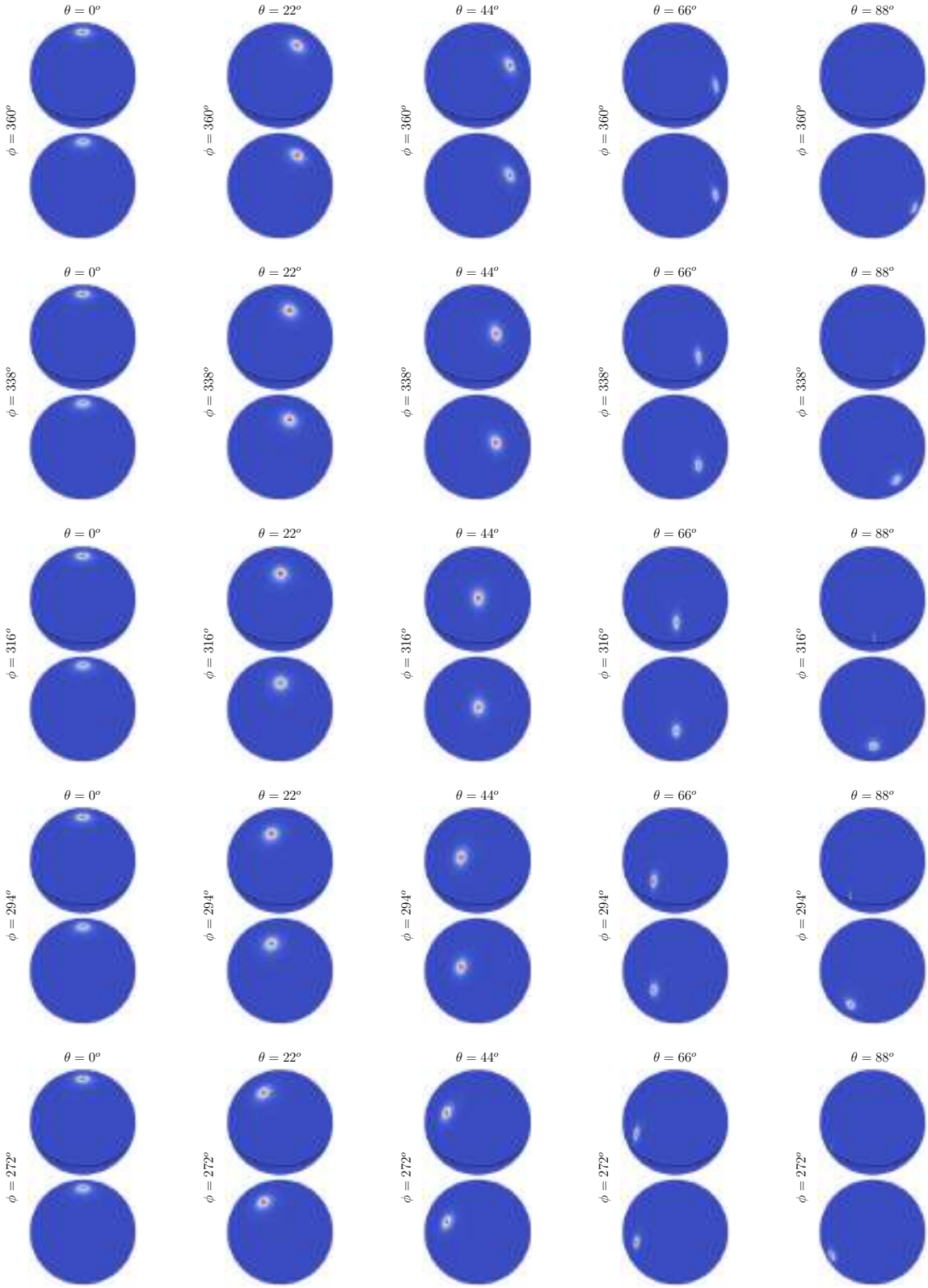
$\alpha_x : 0.010000$, $\alpha_y : 1.000000$



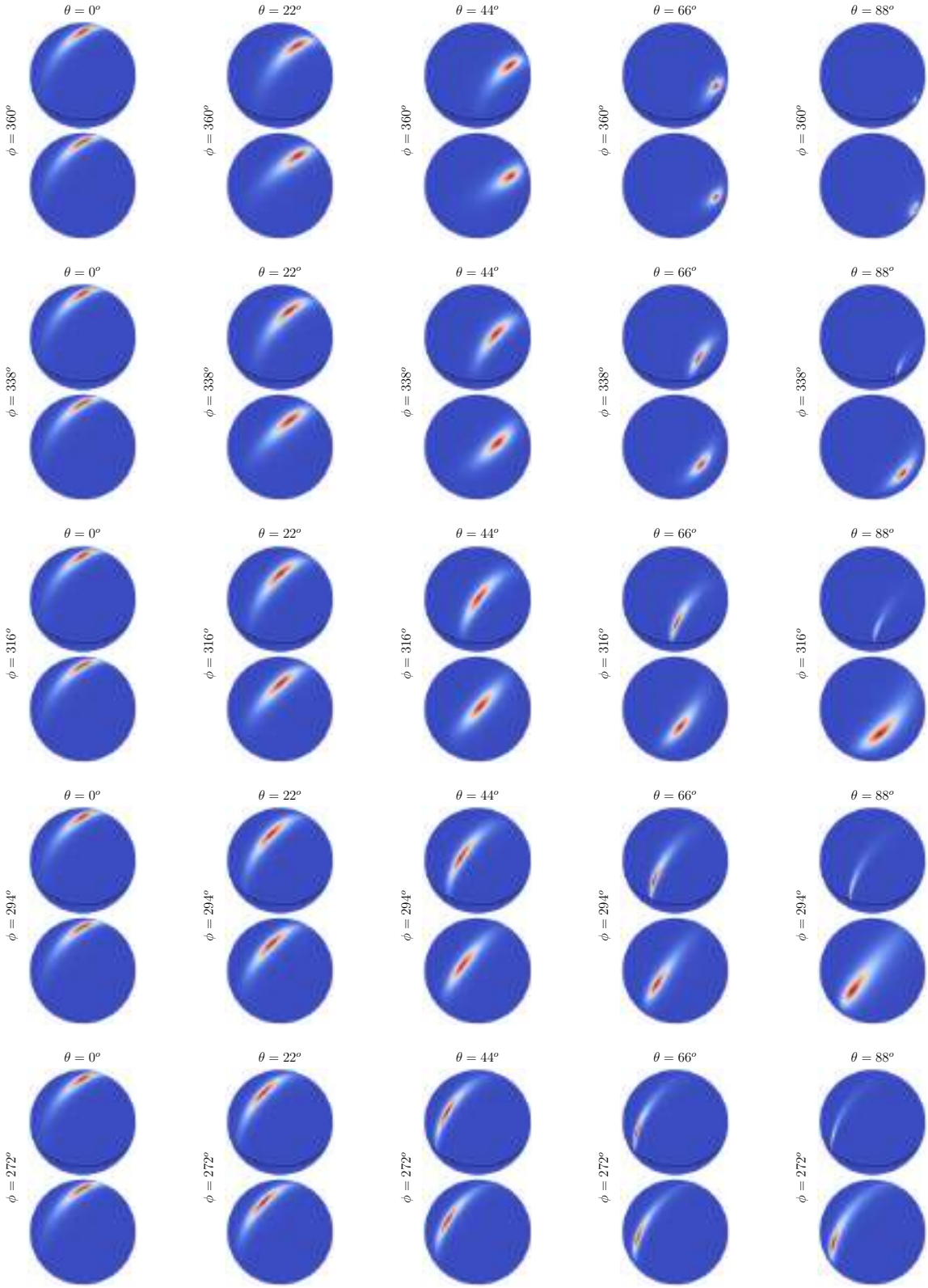
$\alpha_x : 0.050000$, $\alpha_y : 0.010000$



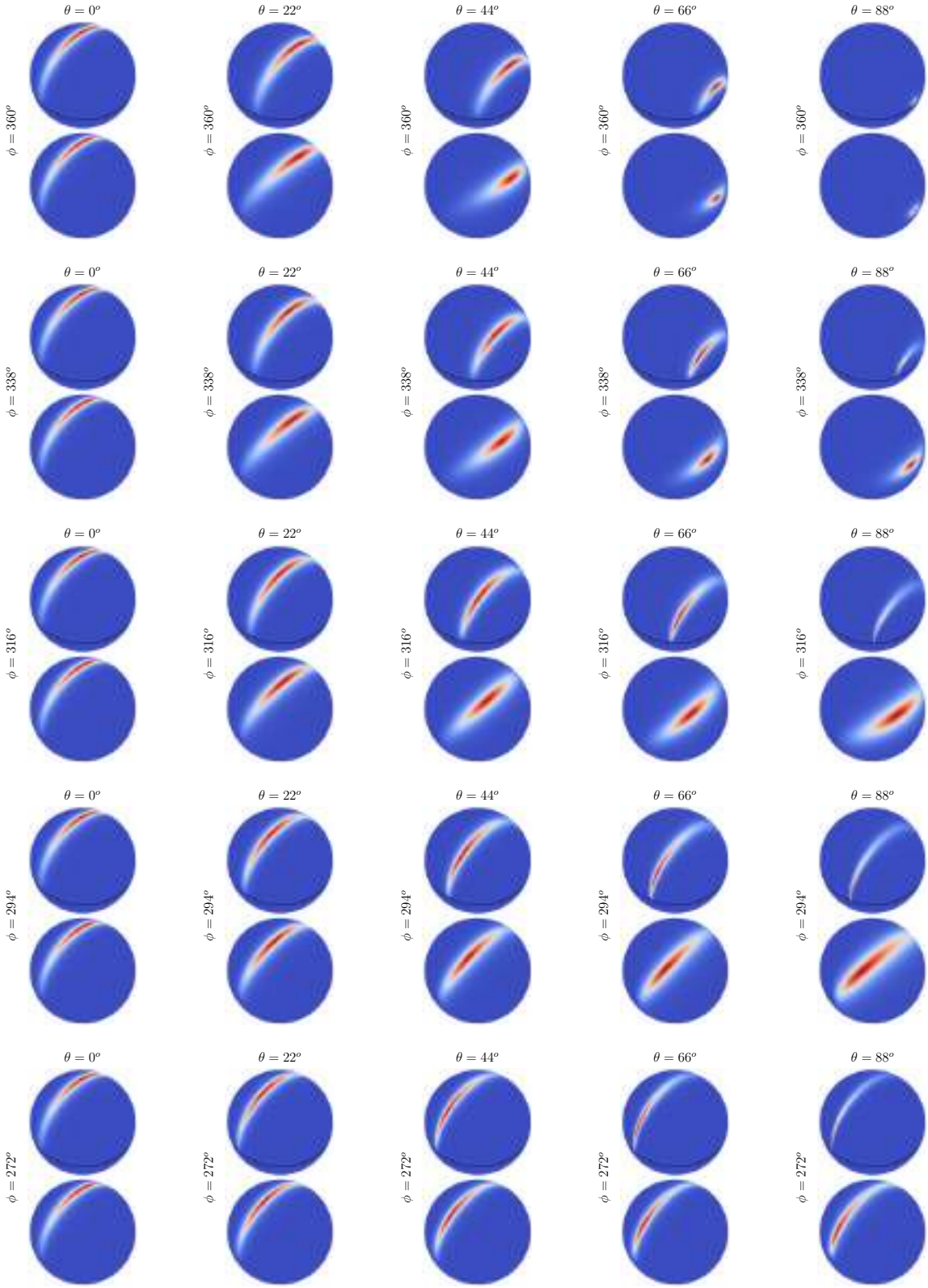
$\alpha_x : 0.050000$, $\alpha_y : 0.050000$



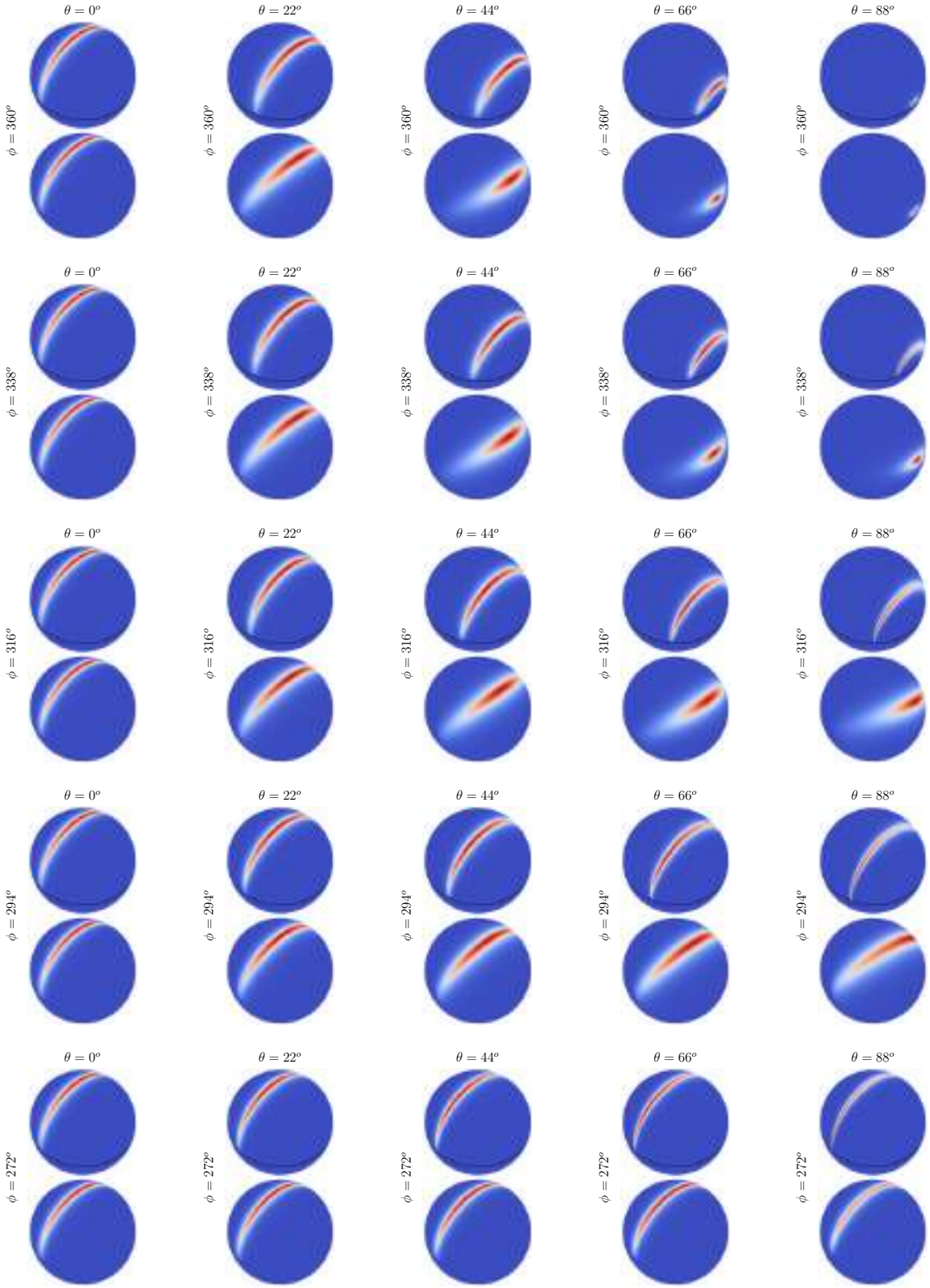
$\alpha_x : 0.050000$, $\alpha_y : 0.250000$



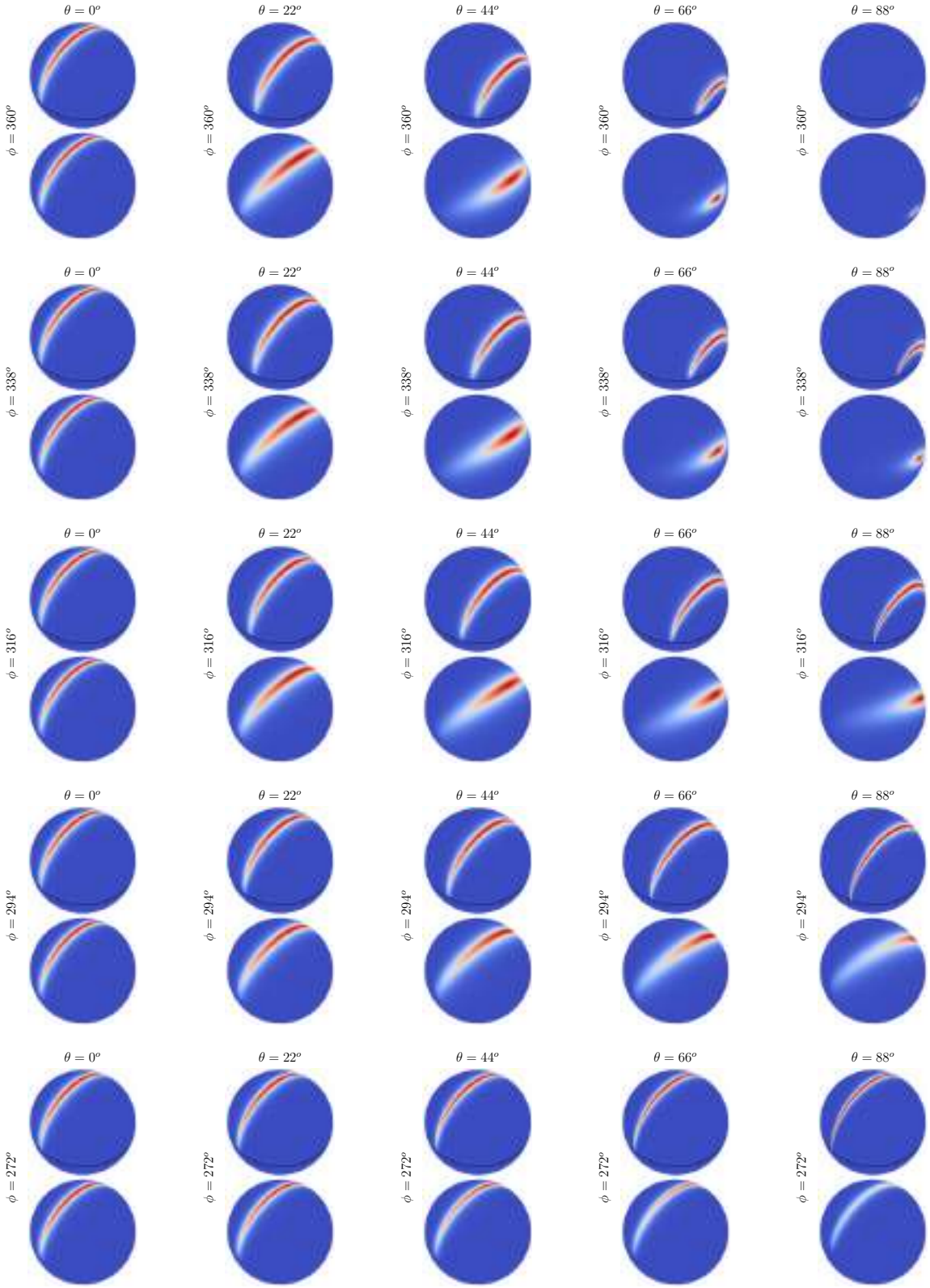
$\alpha_x : 0.050000$, $\alpha_y : 0.500000$



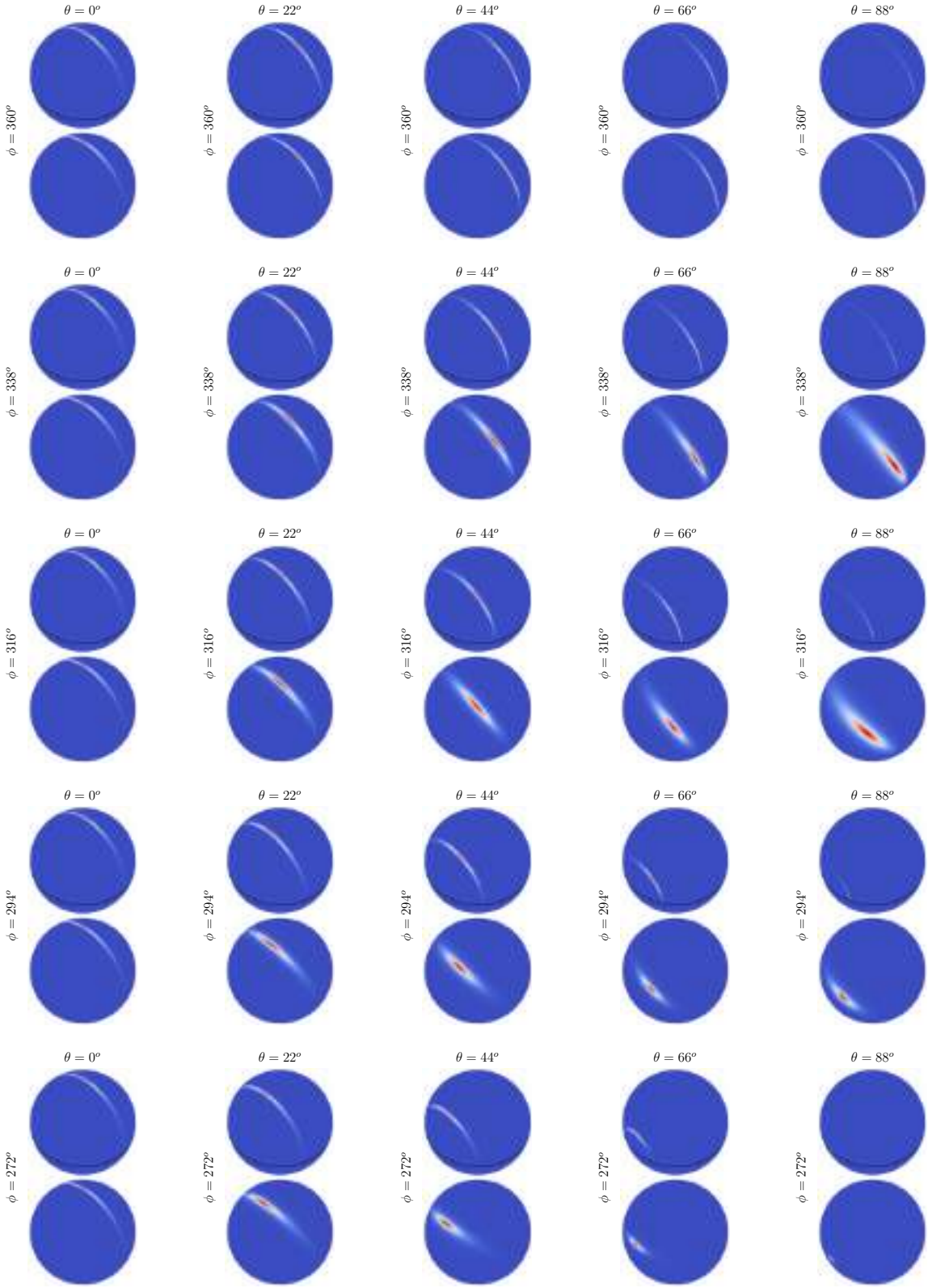
$\alpha_x : 0.050000$, $\alpha_y : 0.800000$



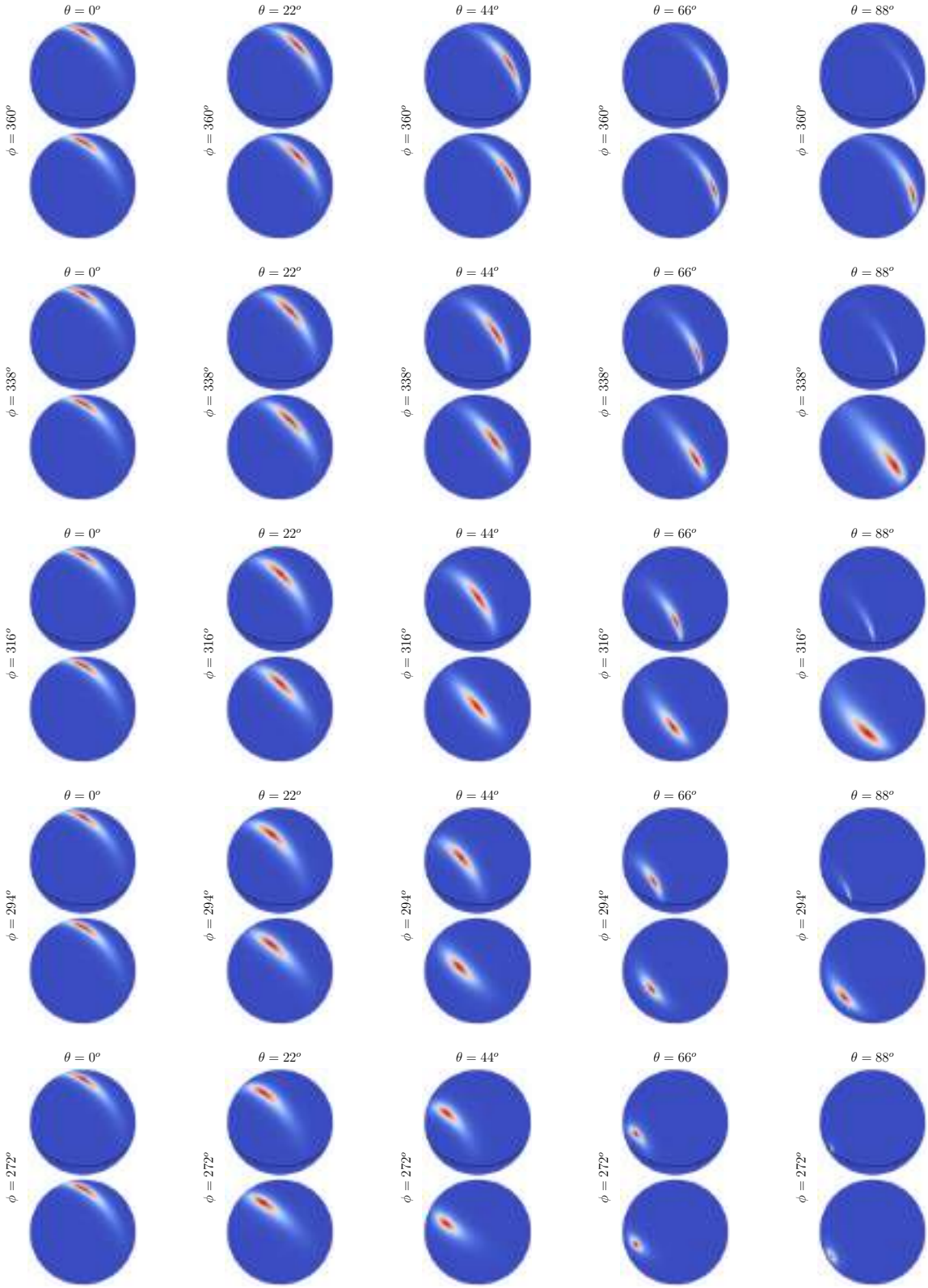
$\alpha_x : 0.050000$, $\alpha_y : 1.000000$



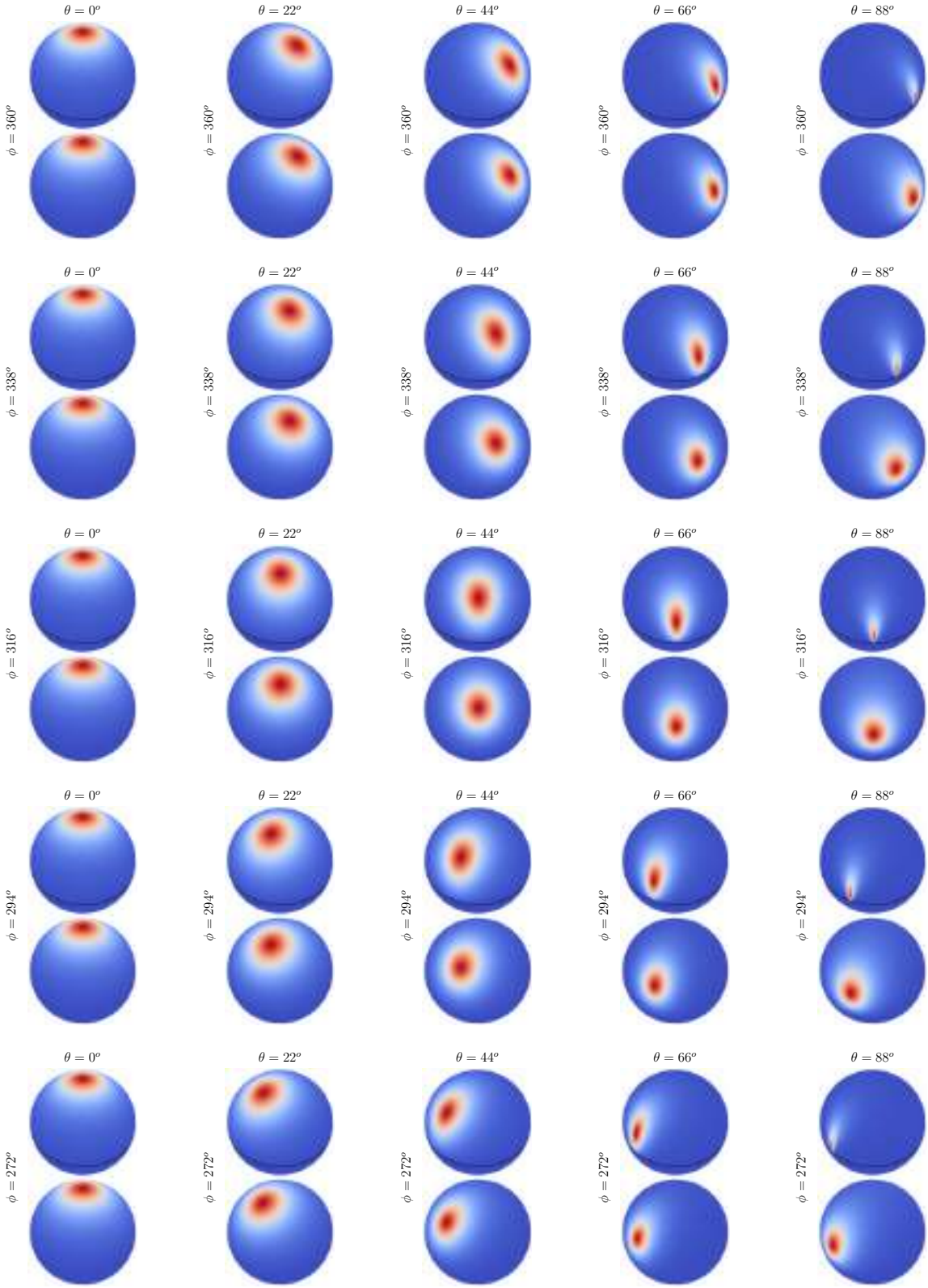
$\alpha_x : 0.250000$, $\alpha_y : 0.010000$



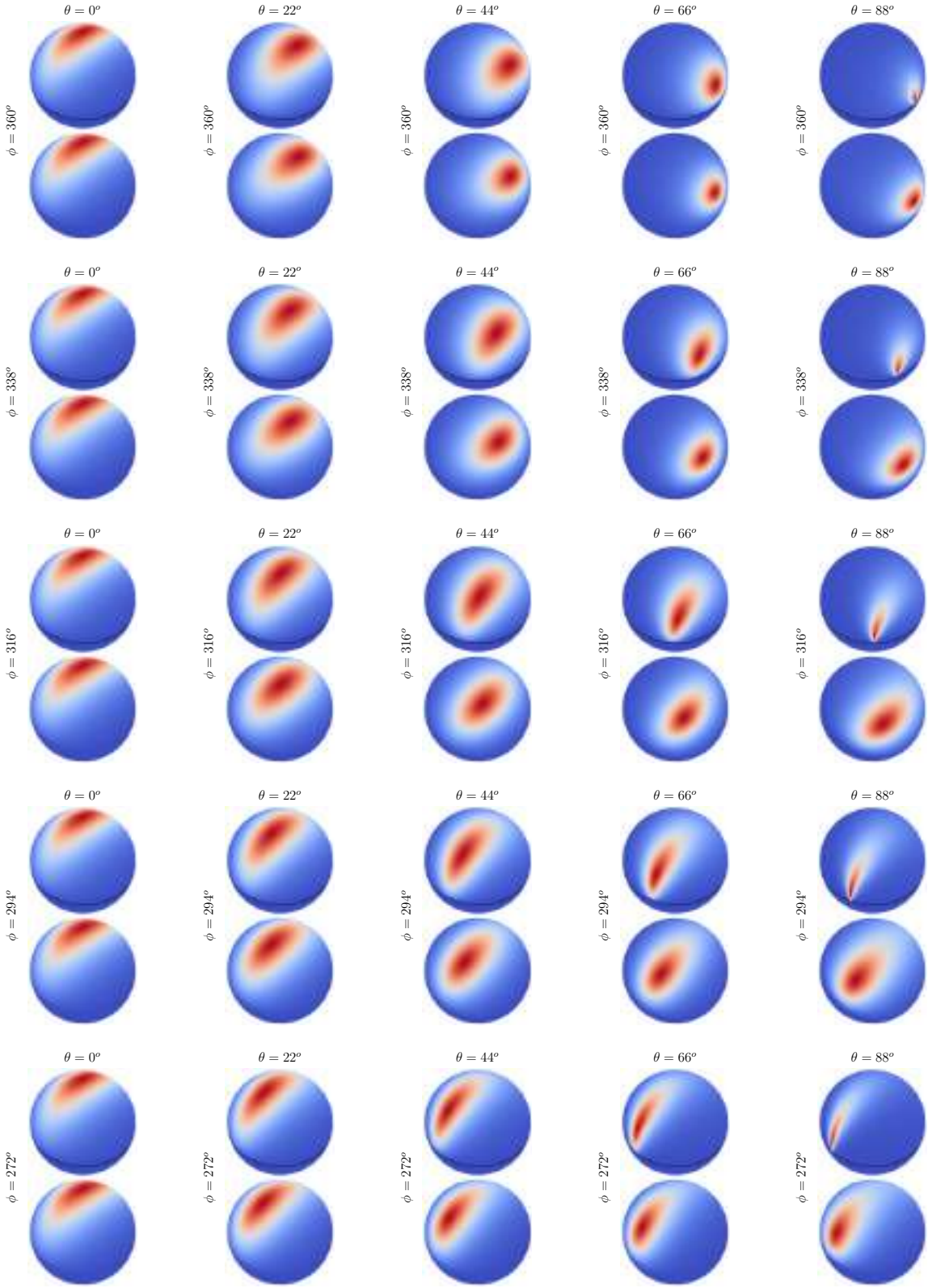
$\alpha_x : 0.250000$, $\alpha_y : 0.050000$



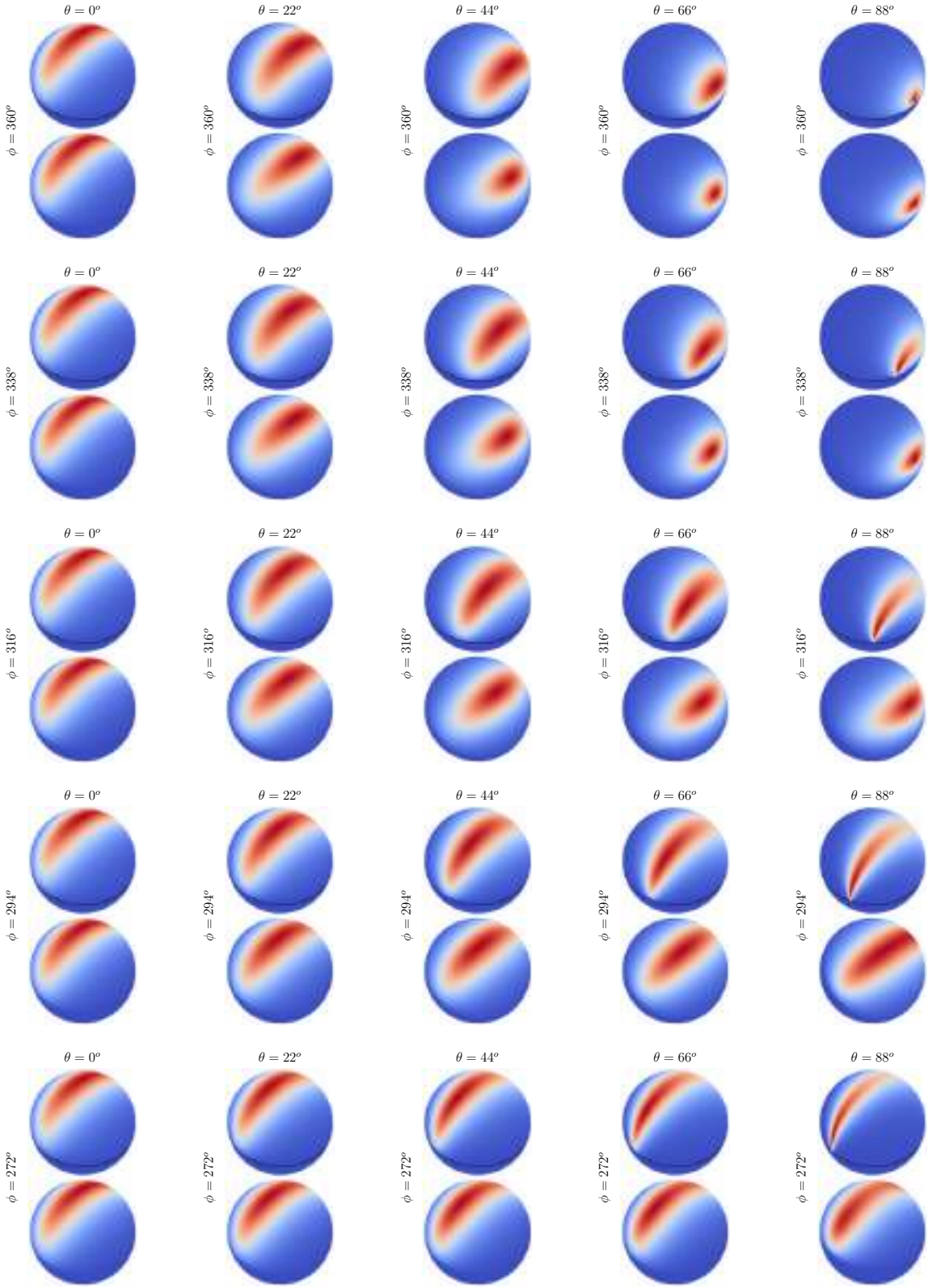
$\alpha_x : 0.250000$, $\alpha_y : 0.250000$



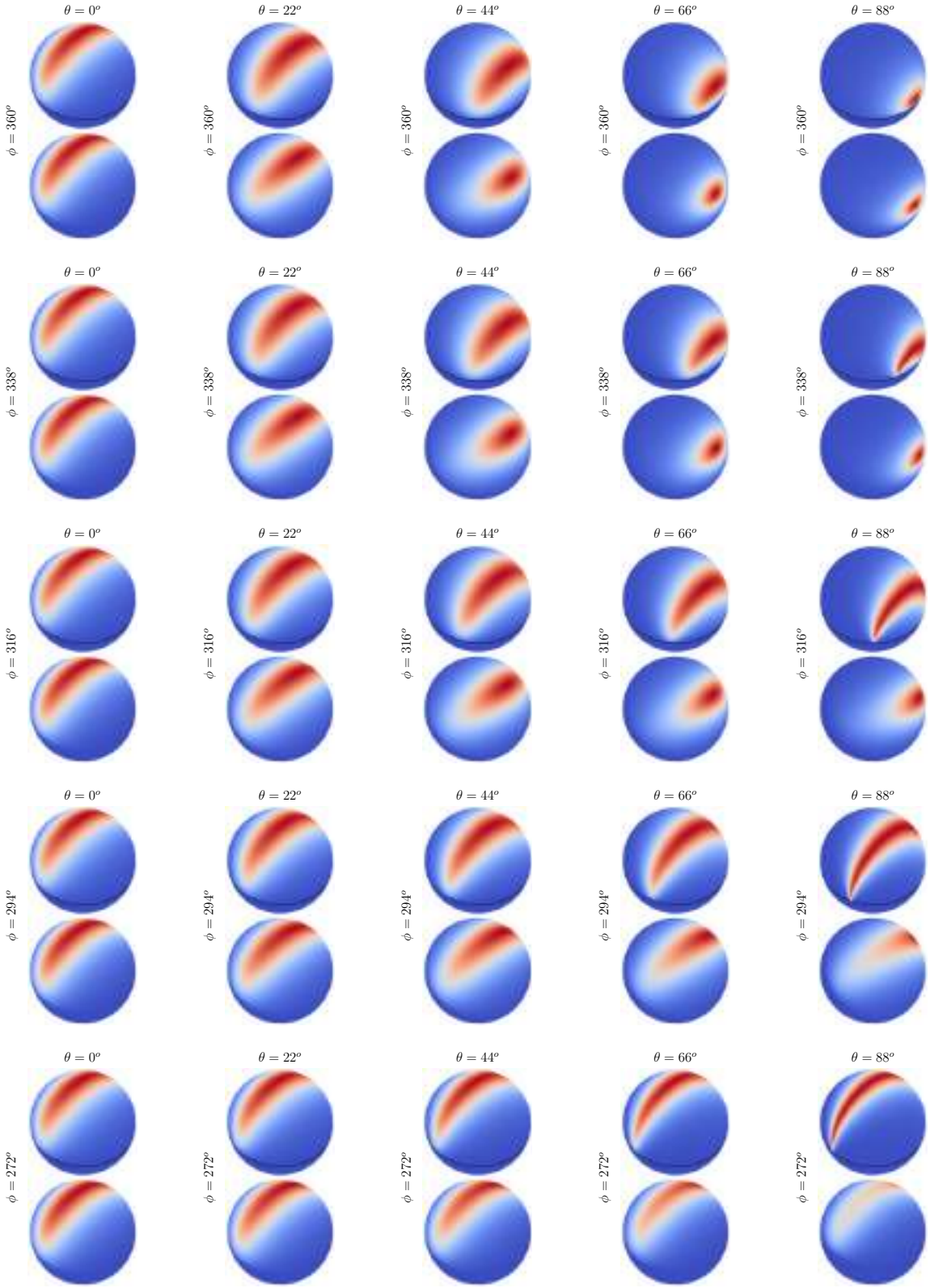
$\alpha_x : 0.250000$, $\alpha_y : 0.500000$



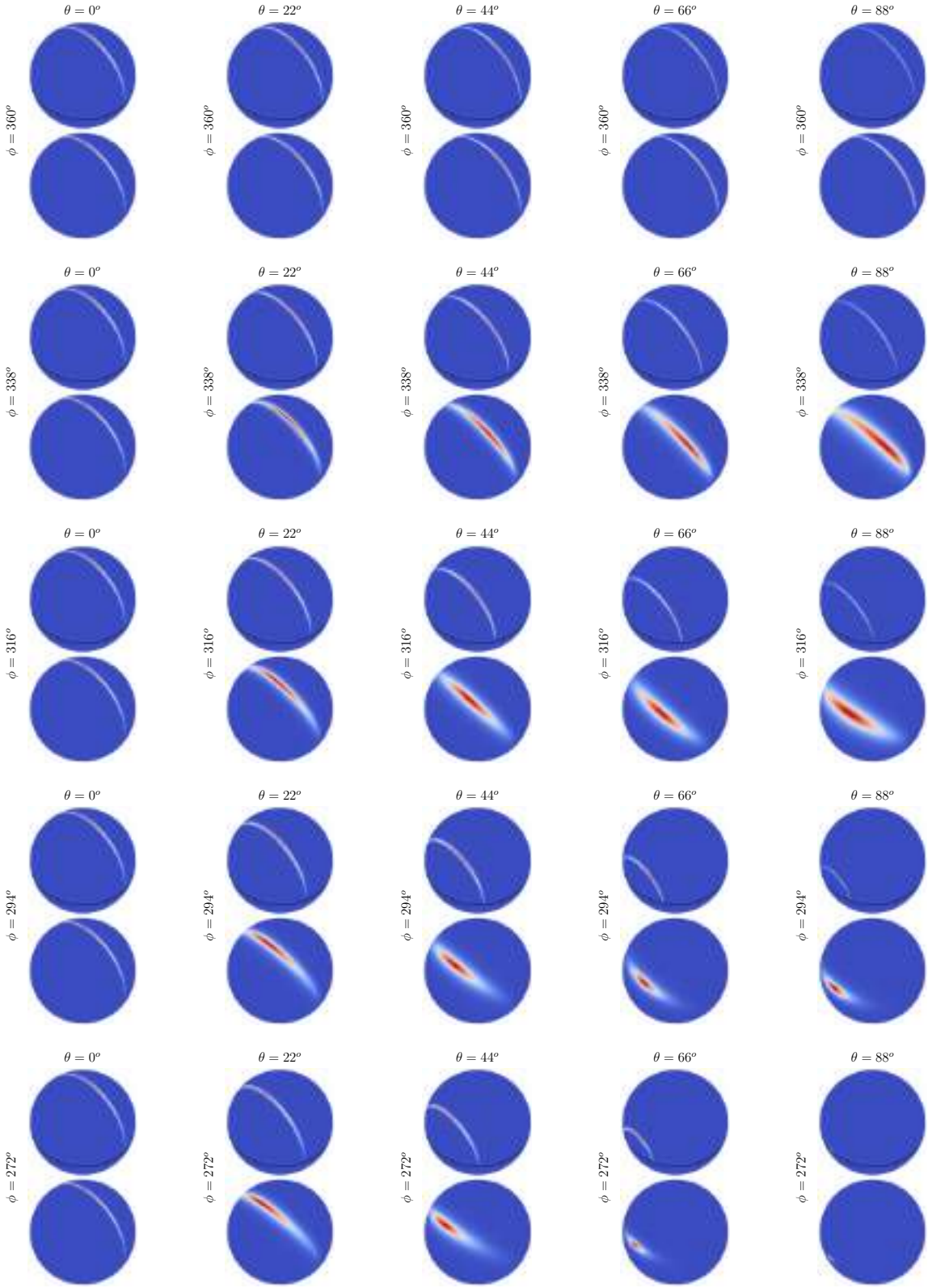
$\alpha_x : 0.250000$, $\alpha_y : 0.800000$



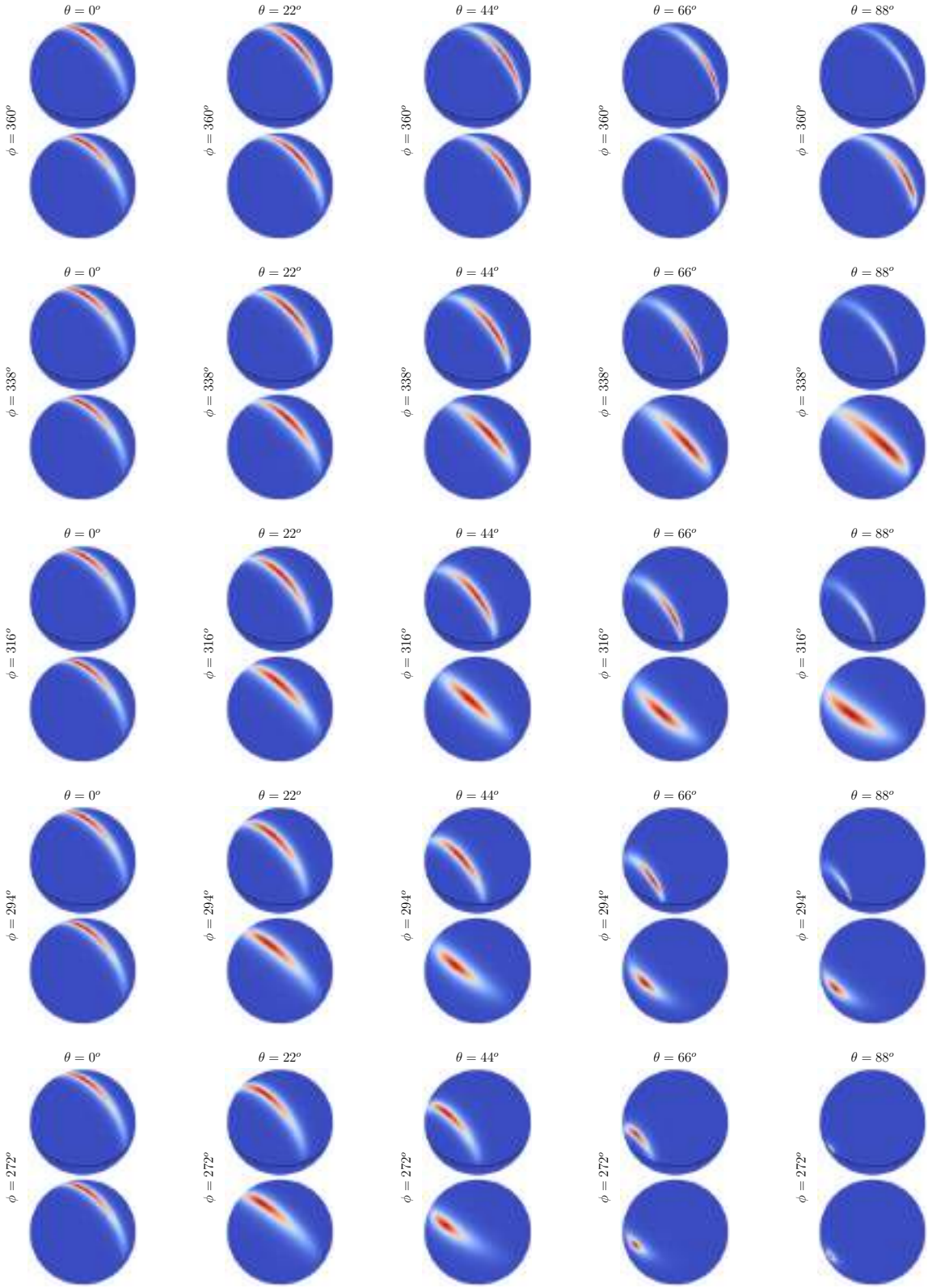
$\alpha_x : 0.250000$, $\alpha_y : 1.000000$



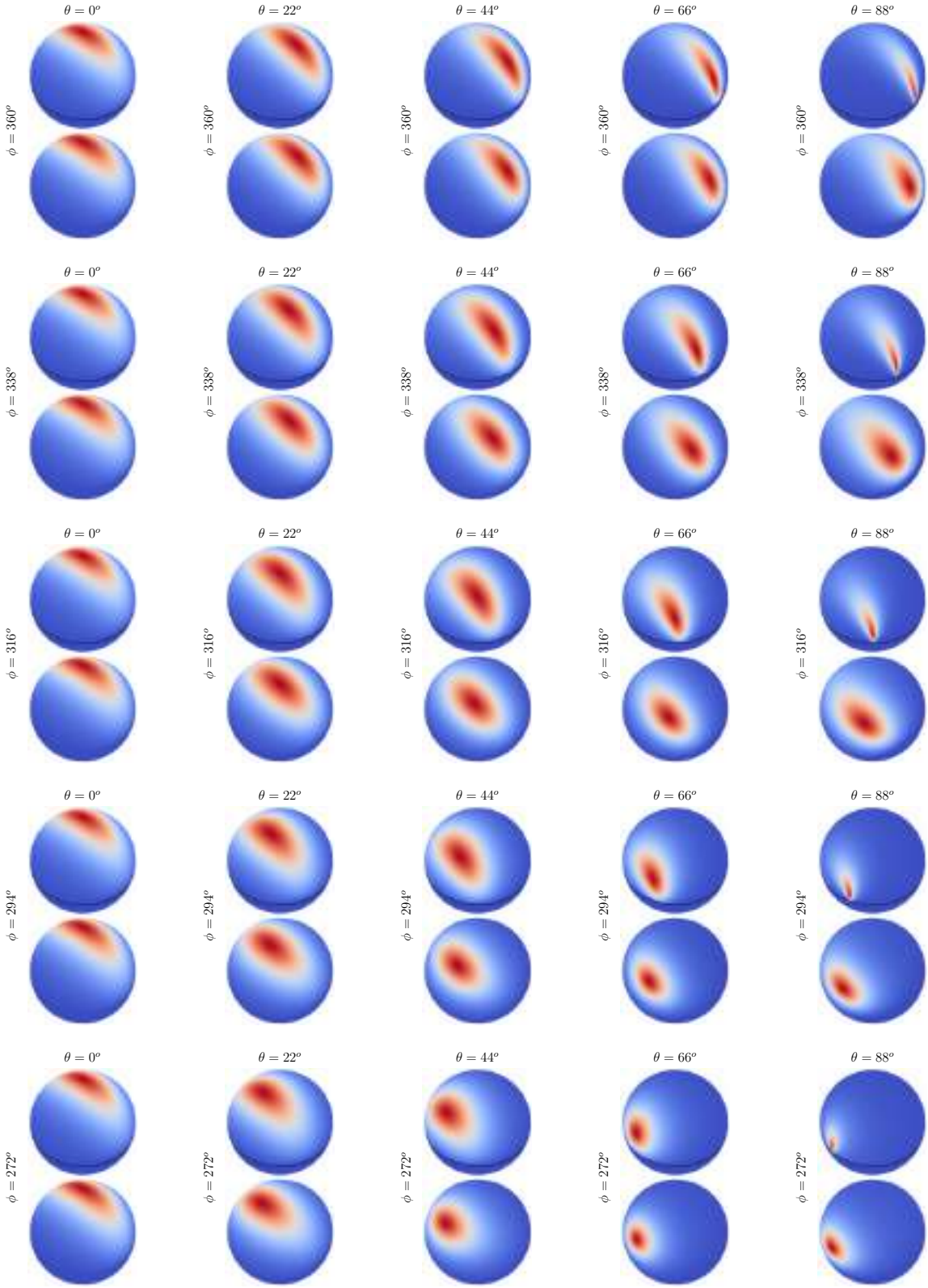
$\alpha_x : 0.500000$, $\alpha_y : 0.010000$



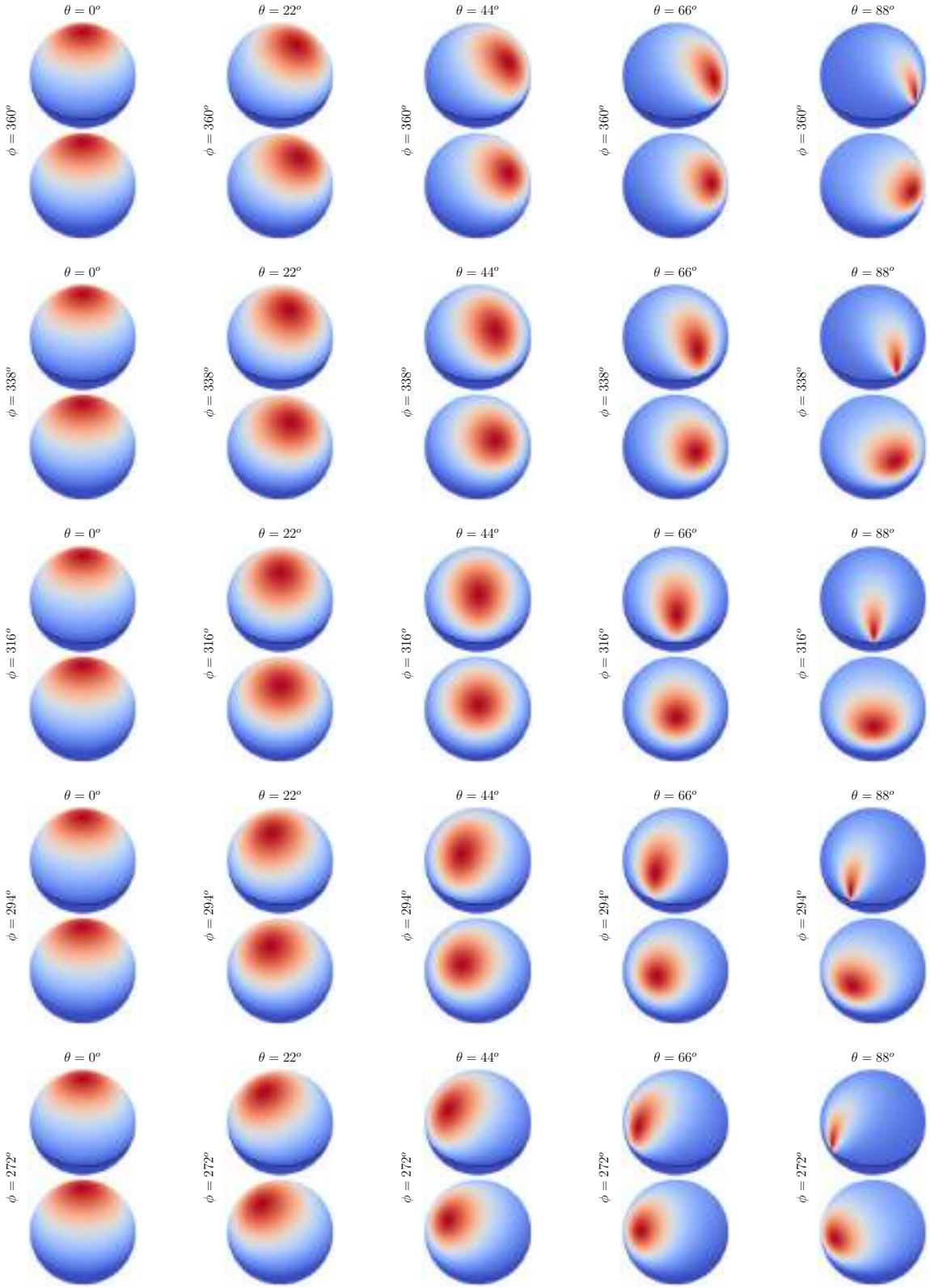
$\alpha_x : 0.500000$, $\alpha_y : 0.050000$



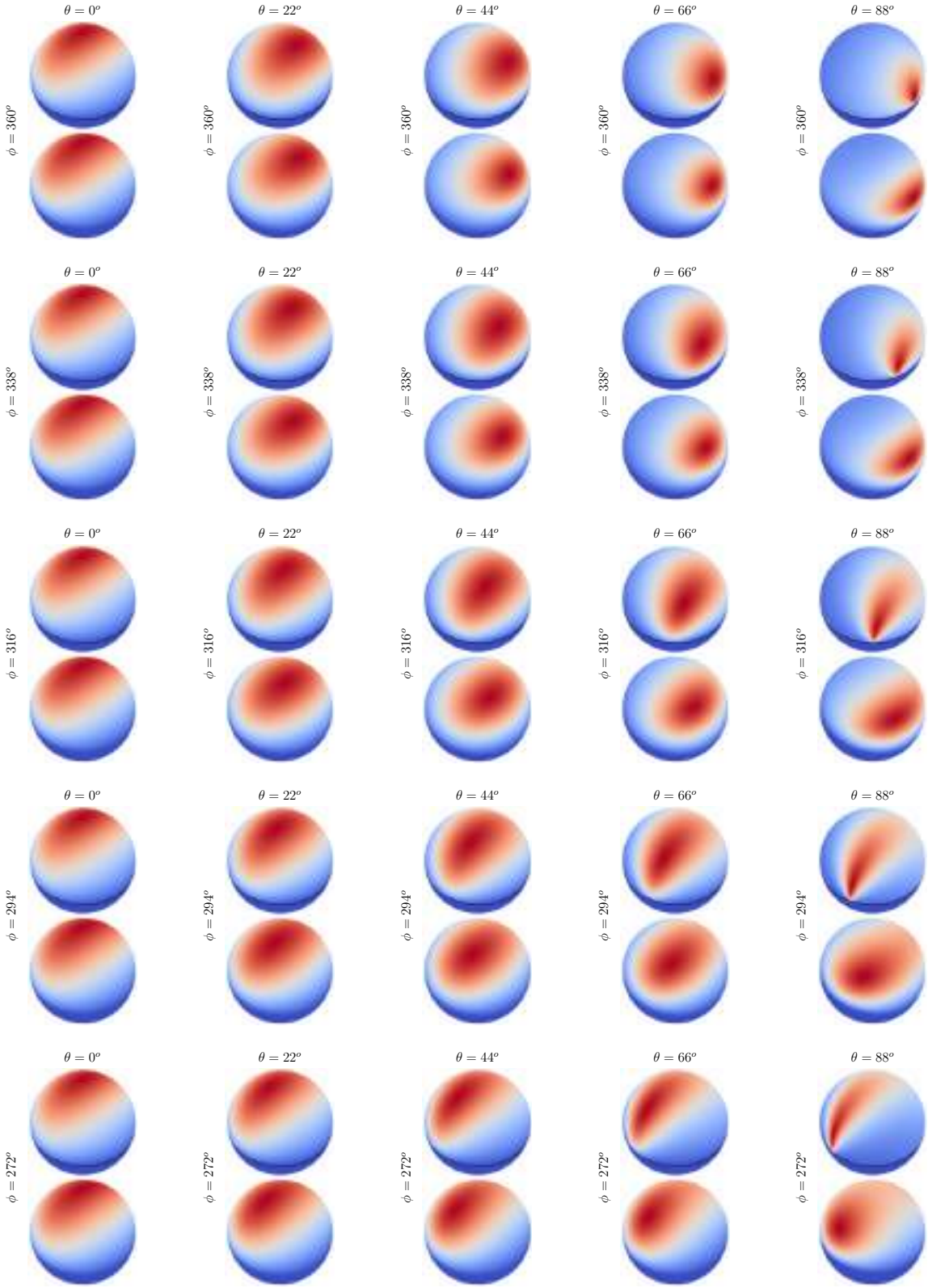
$\alpha_x : 0.500000$, $\alpha_y : 0.250000$



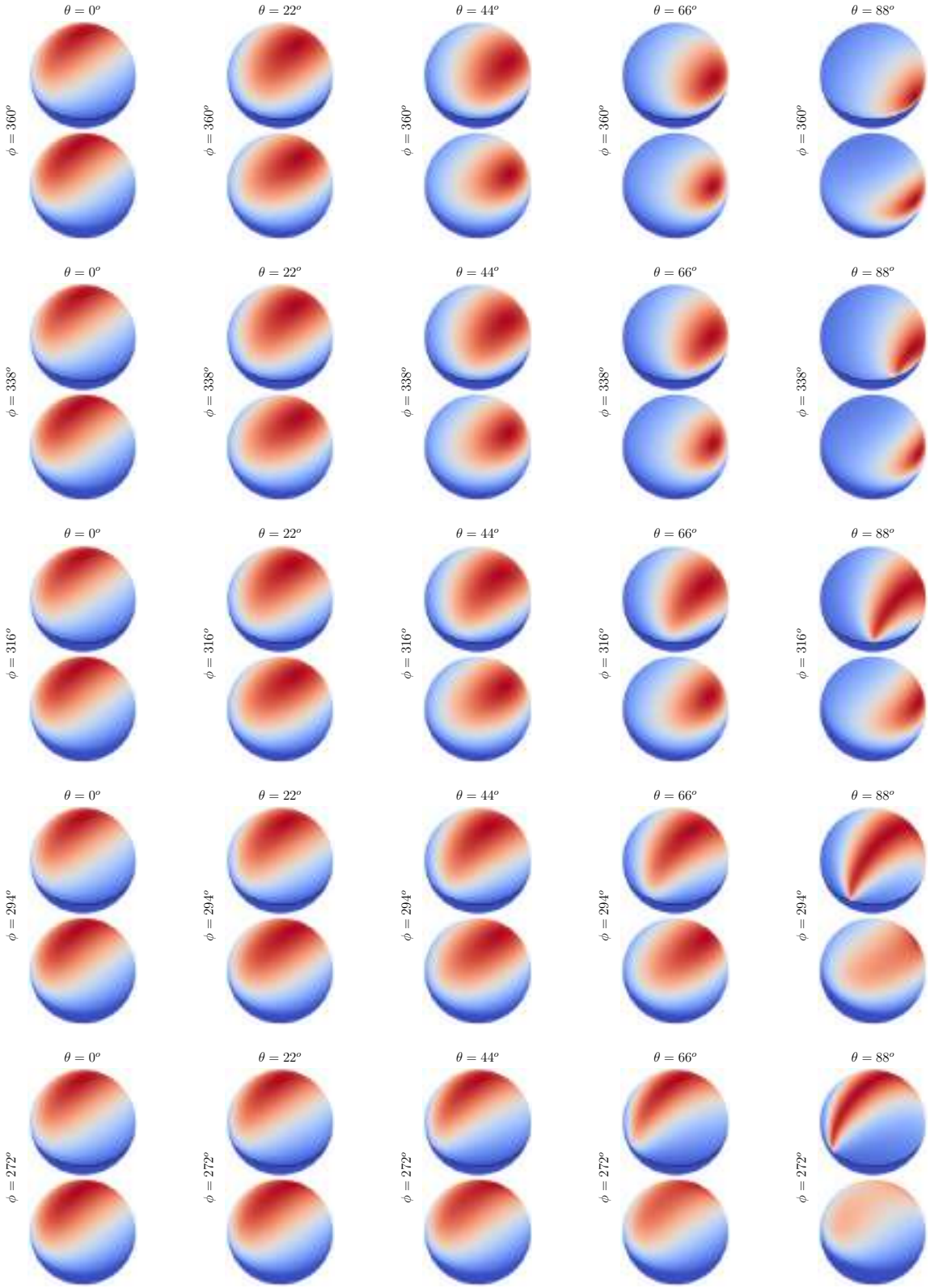
$\alpha_x : 0.500000$, $\alpha_y : 0.500000$



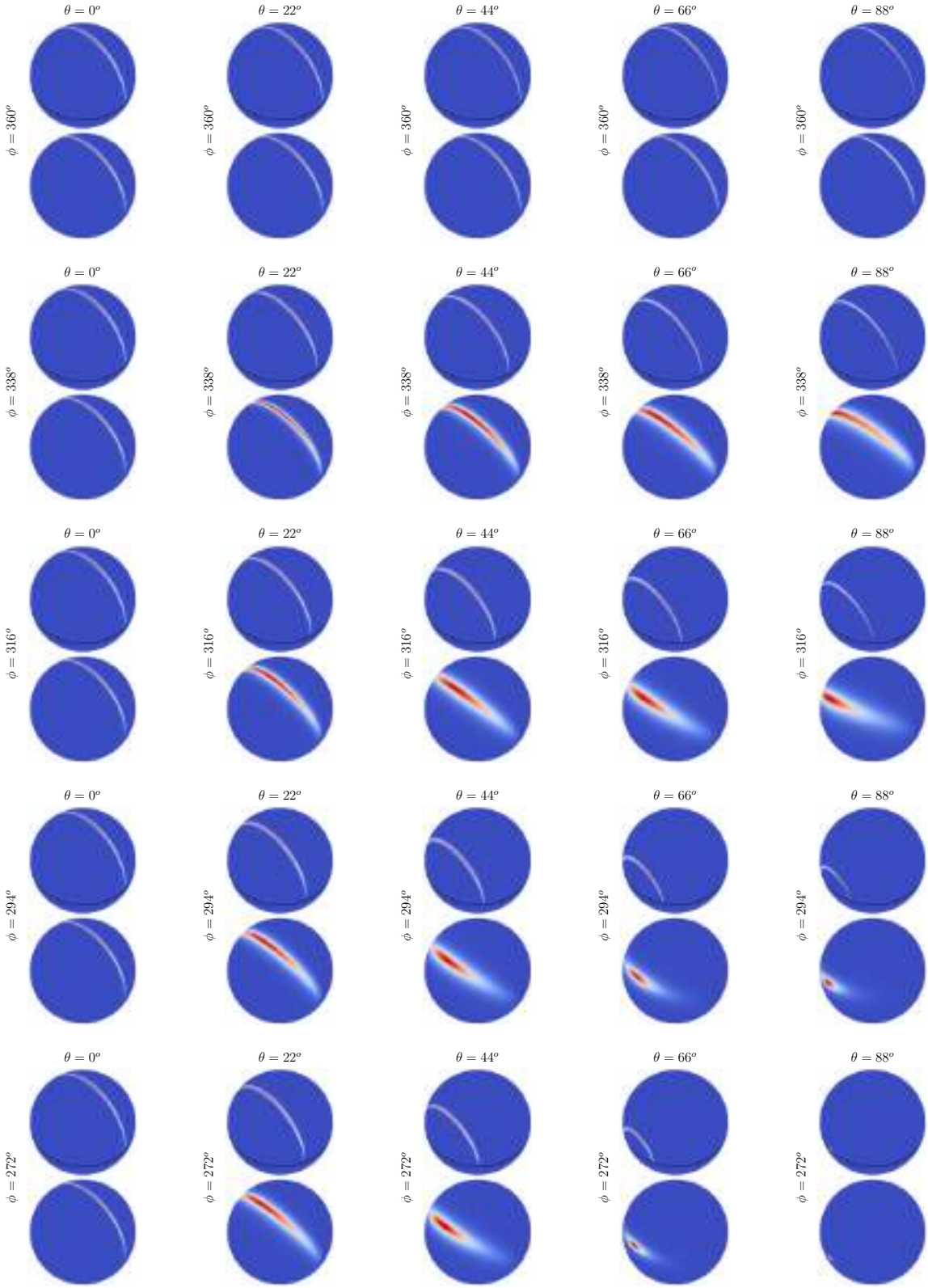
$\alpha_x : 0.500000$, $\alpha_y : 0.800000$



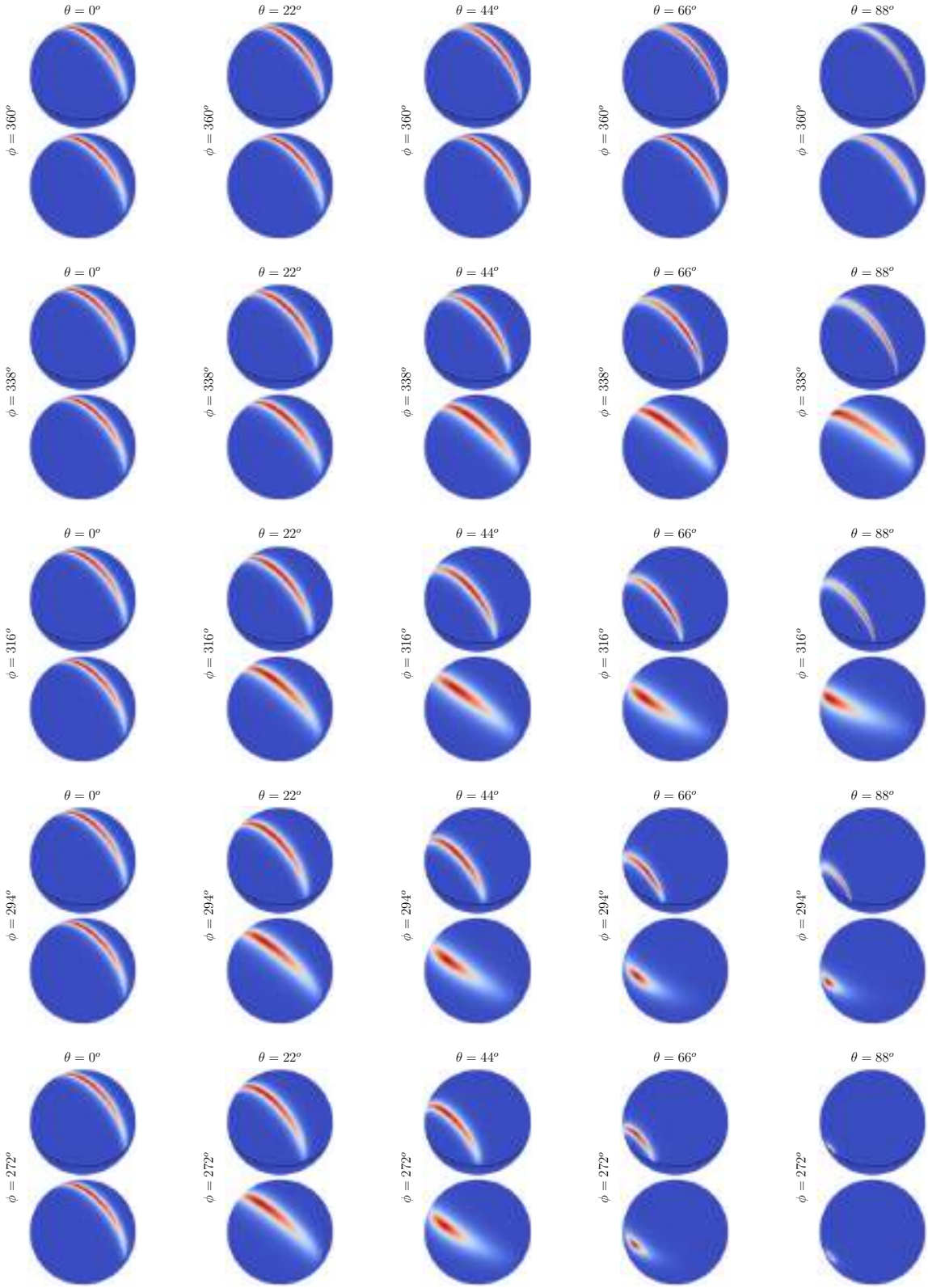
$\alpha_x : 0.500000$, $\alpha_y : 1.000000$



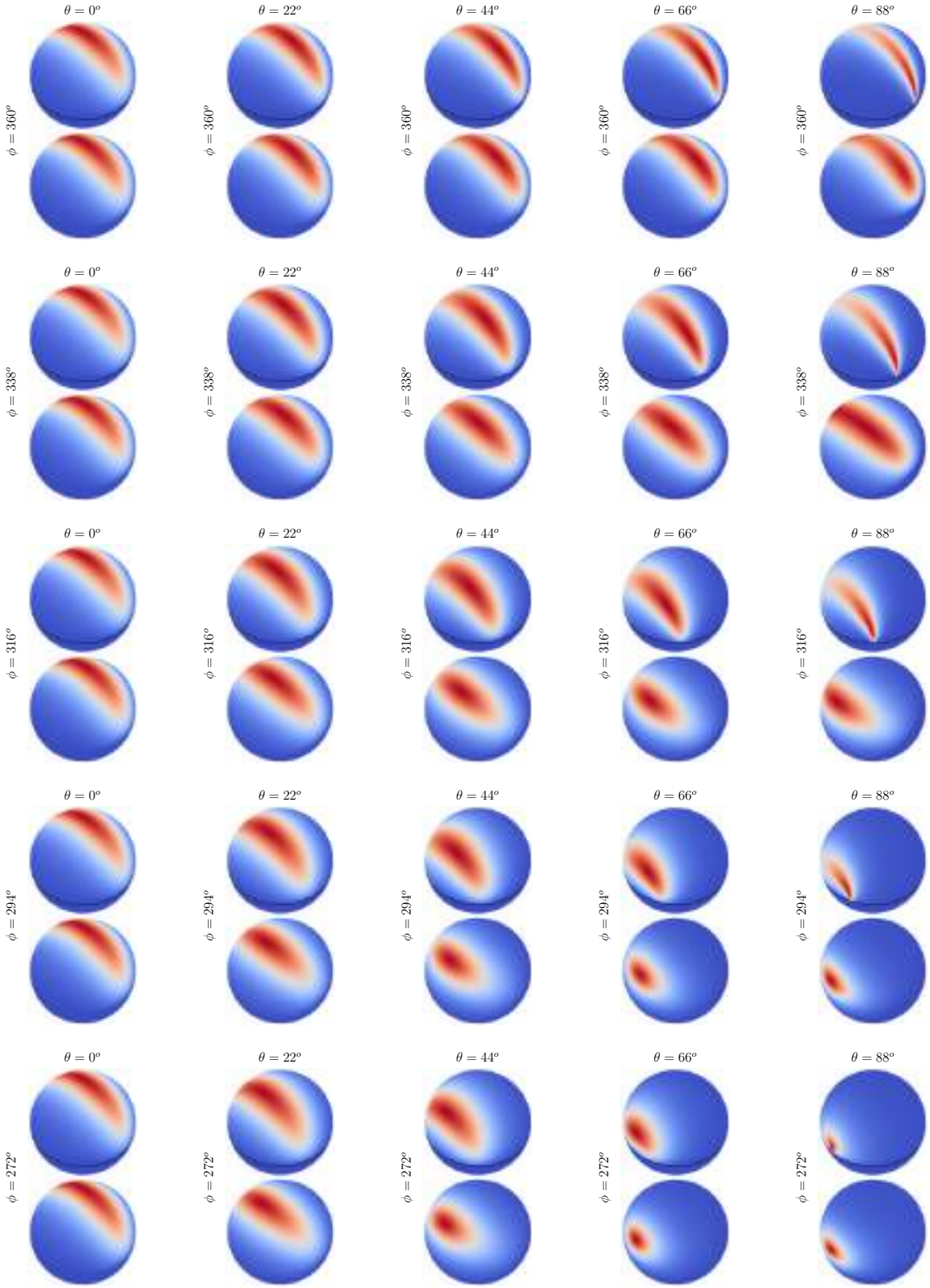
$\alpha_x : 0.800000$, $\alpha_y : 0.010000$



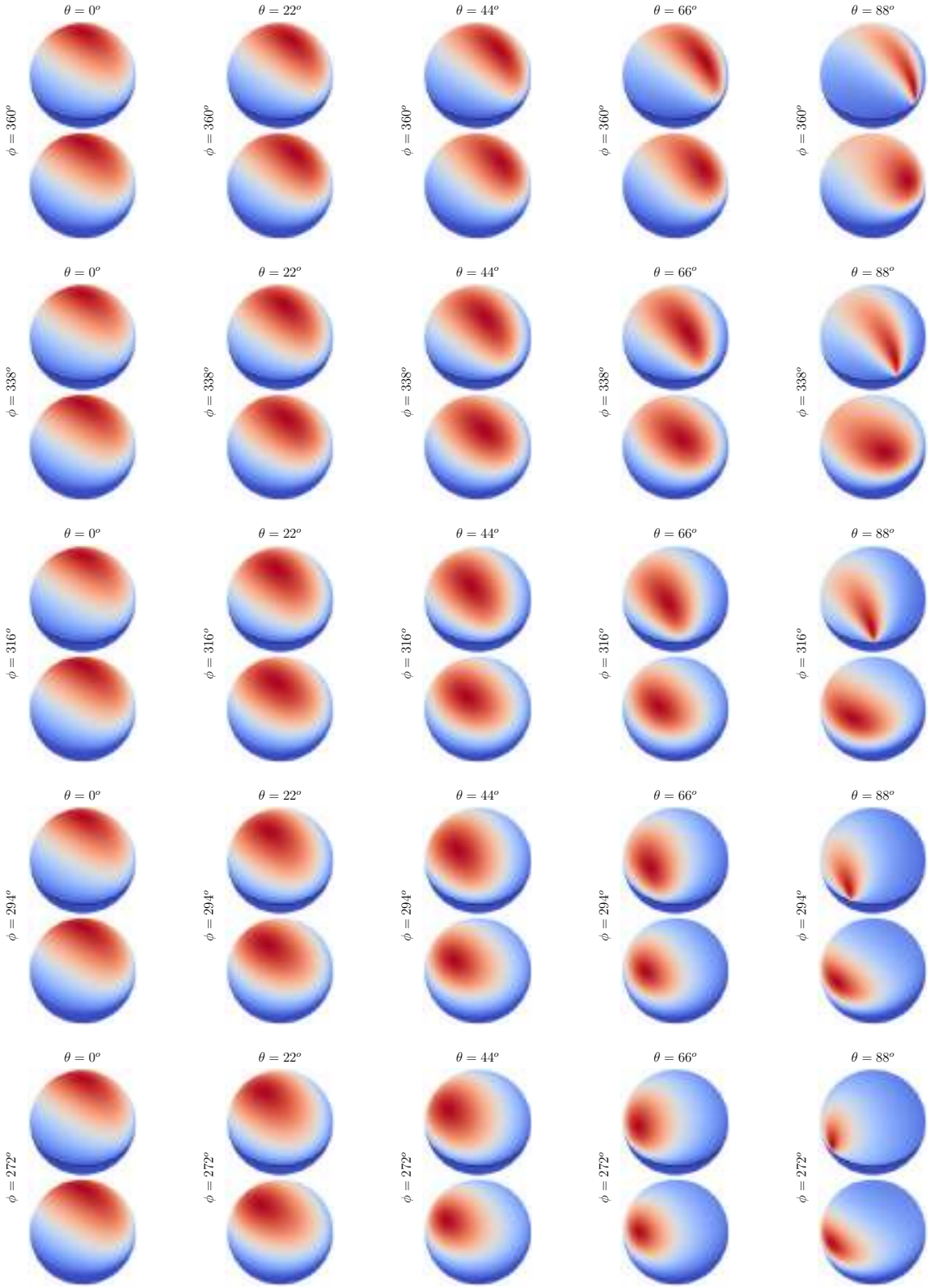
$\alpha_x : 0.800000$, $\alpha_y : 0.050000$



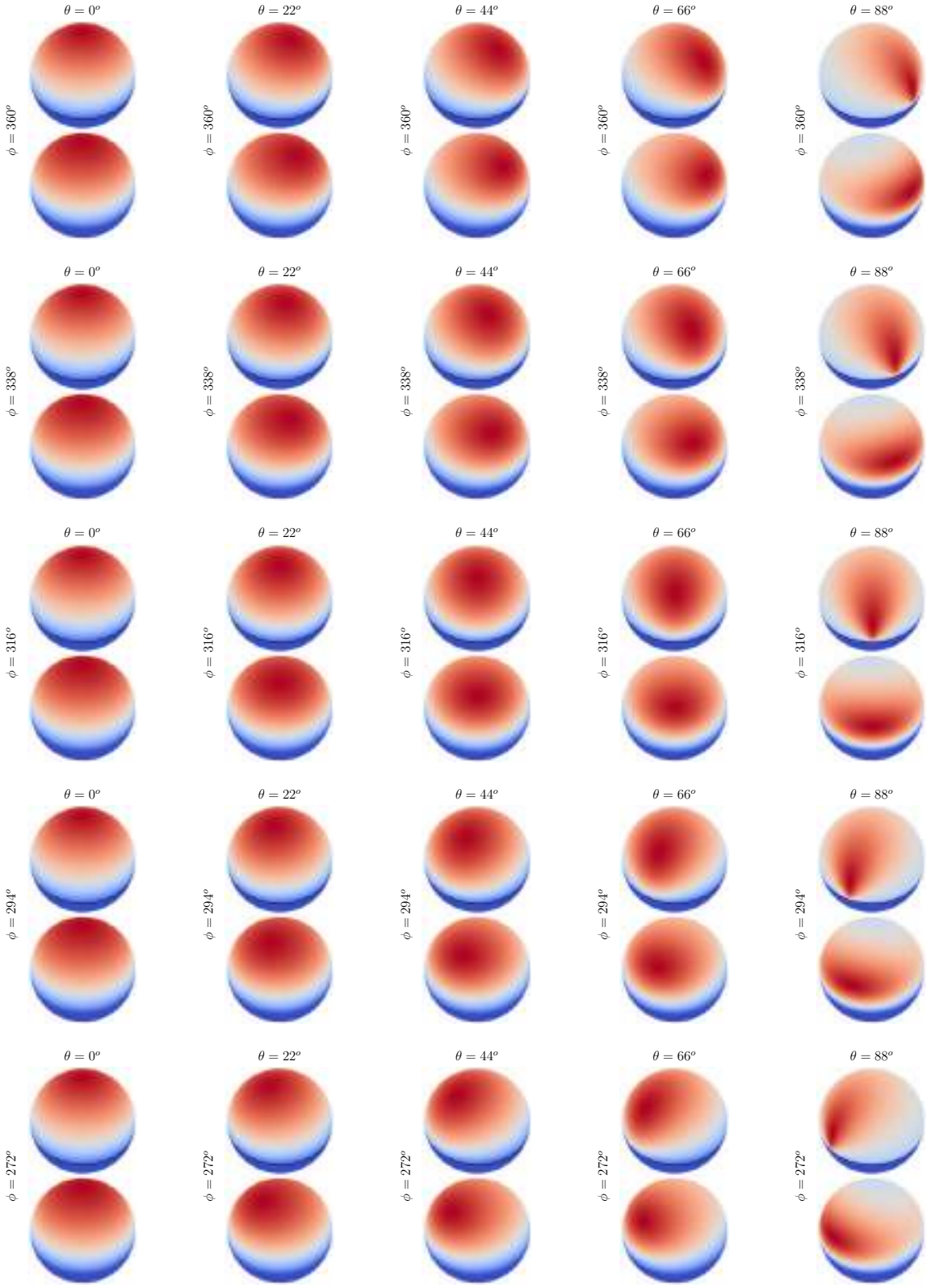
$\alpha_x : 0.800000$, $\alpha_y : 0.250000$



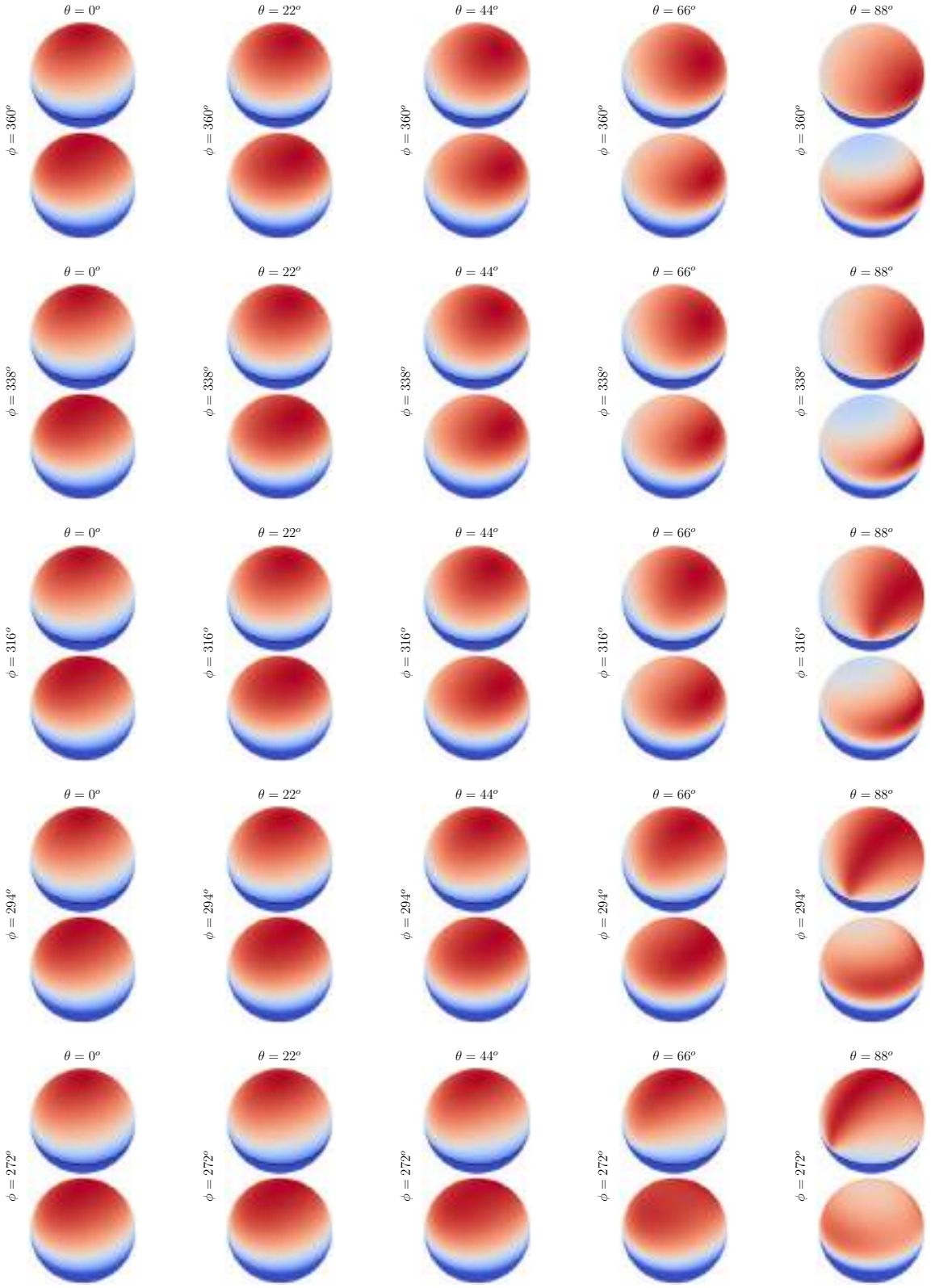
$\alpha_x : 0.800000$, $\alpha_y : 0.500000$



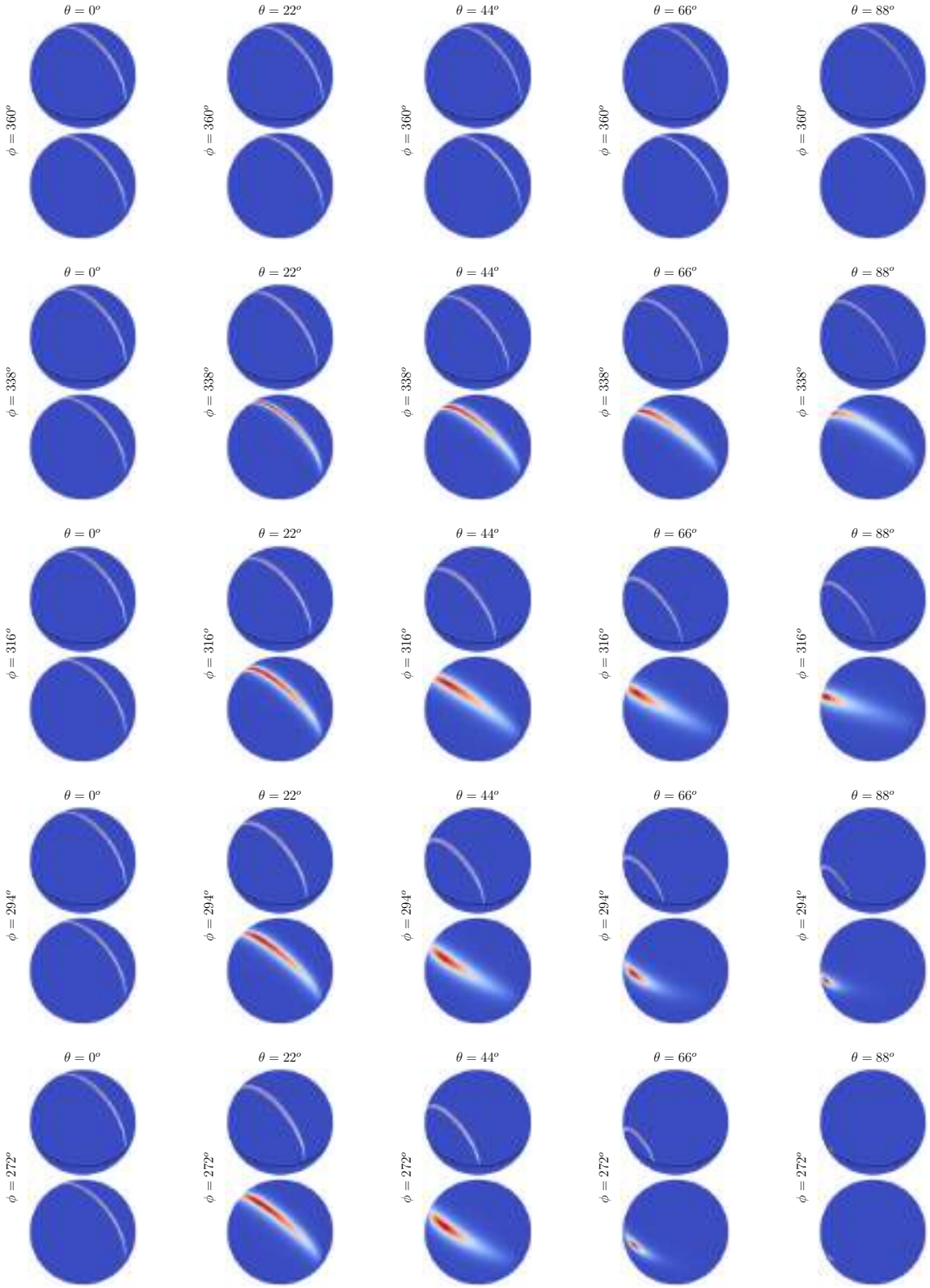
$\alpha_x : 0.800000$, $\alpha_y : 0.800000$



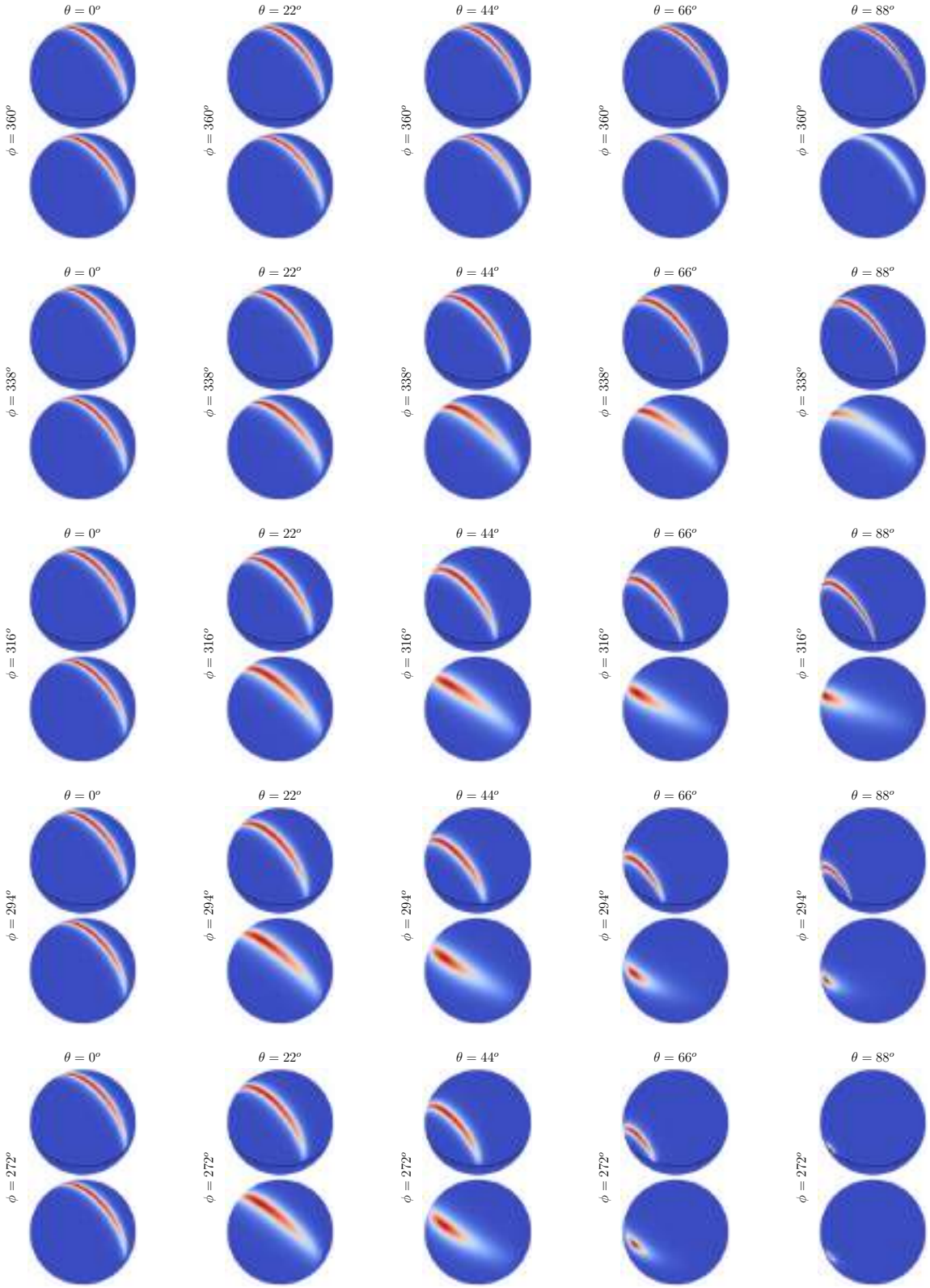
$\alpha_x : 0.800000$, $\alpha_y : 1.000000$



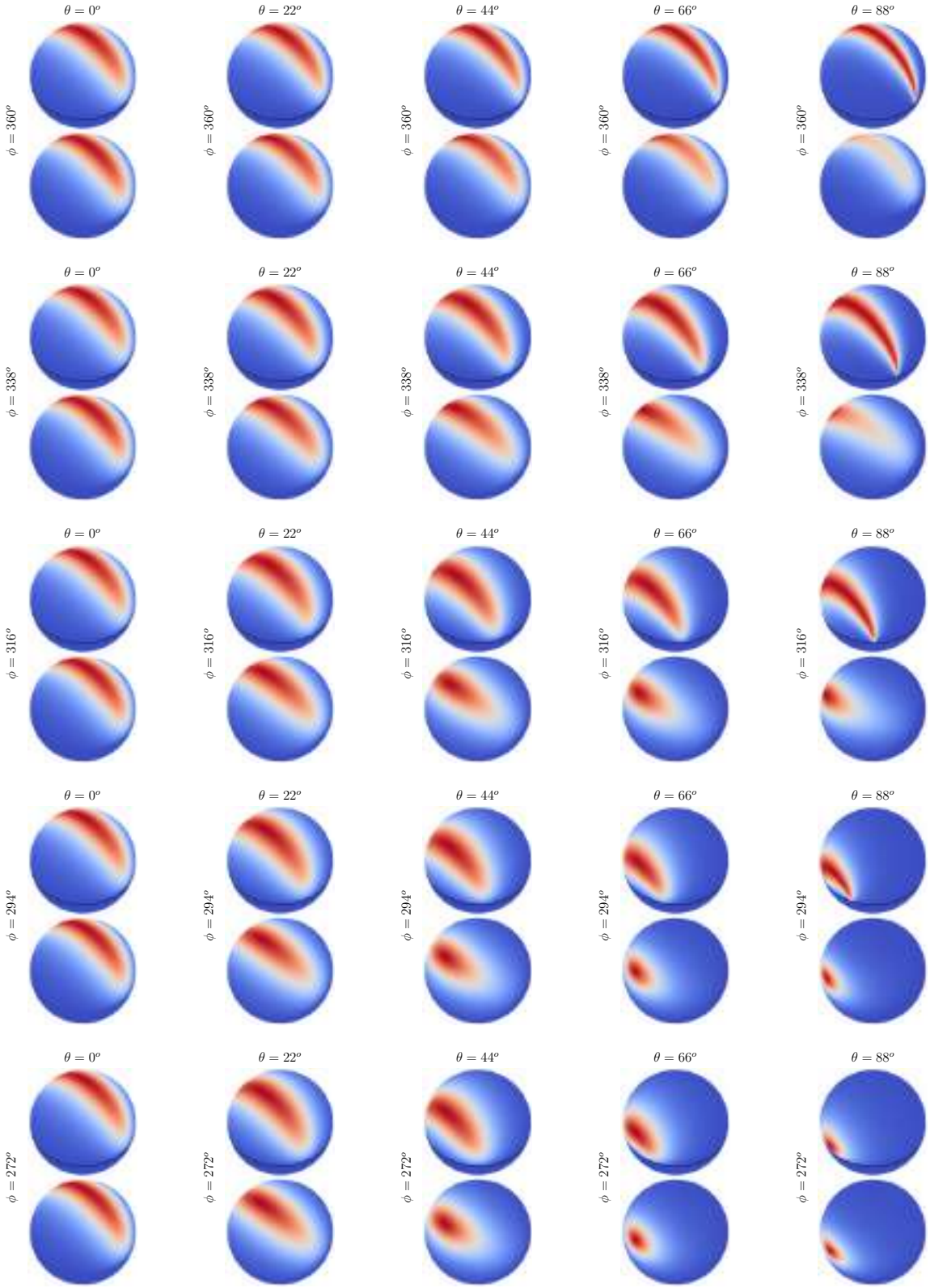
$\alpha_x : 1.000000$, $\alpha_y : 0.010000$



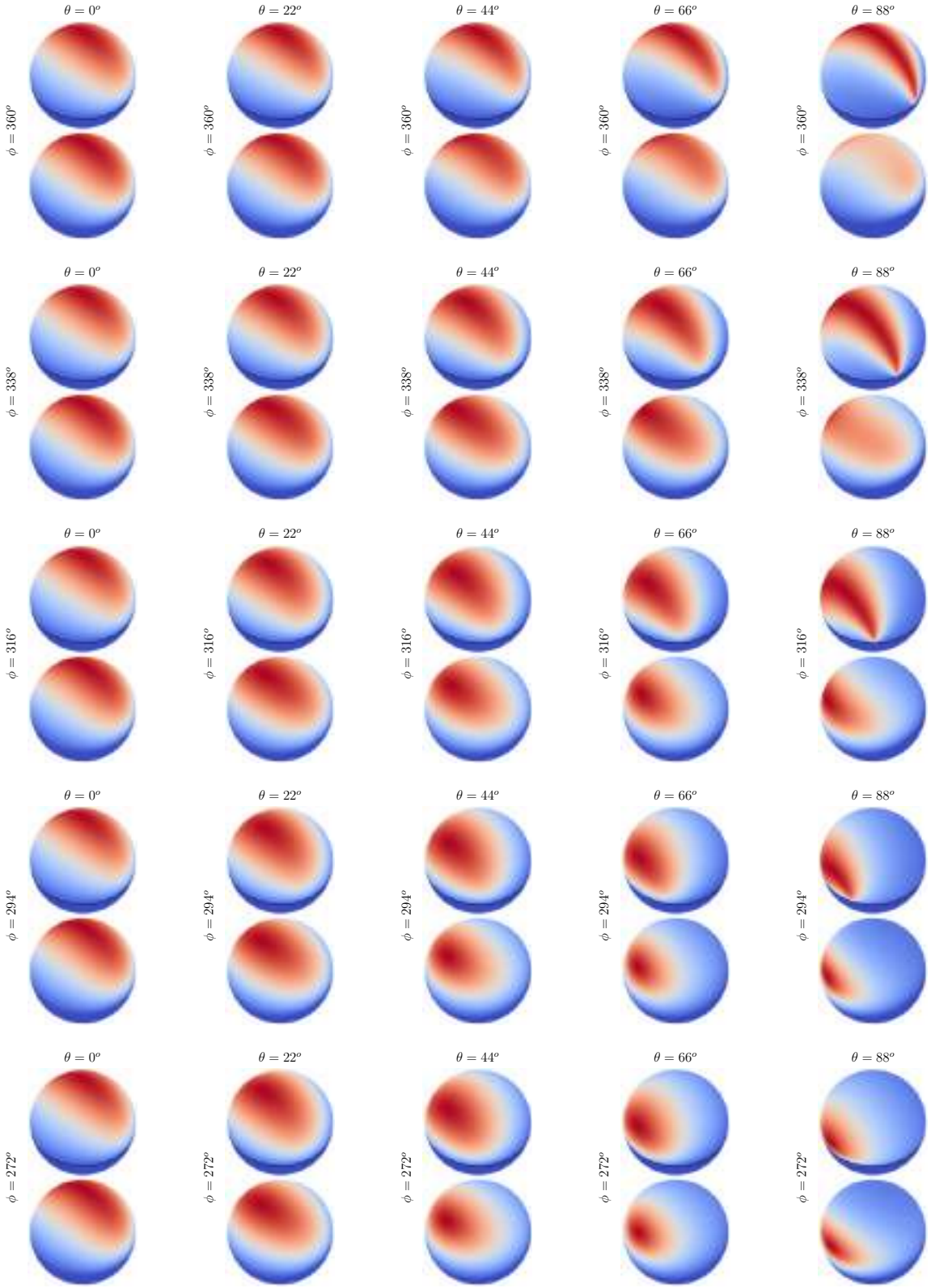
$\alpha_x : 1.000000$, $\alpha_y : 0.050000$



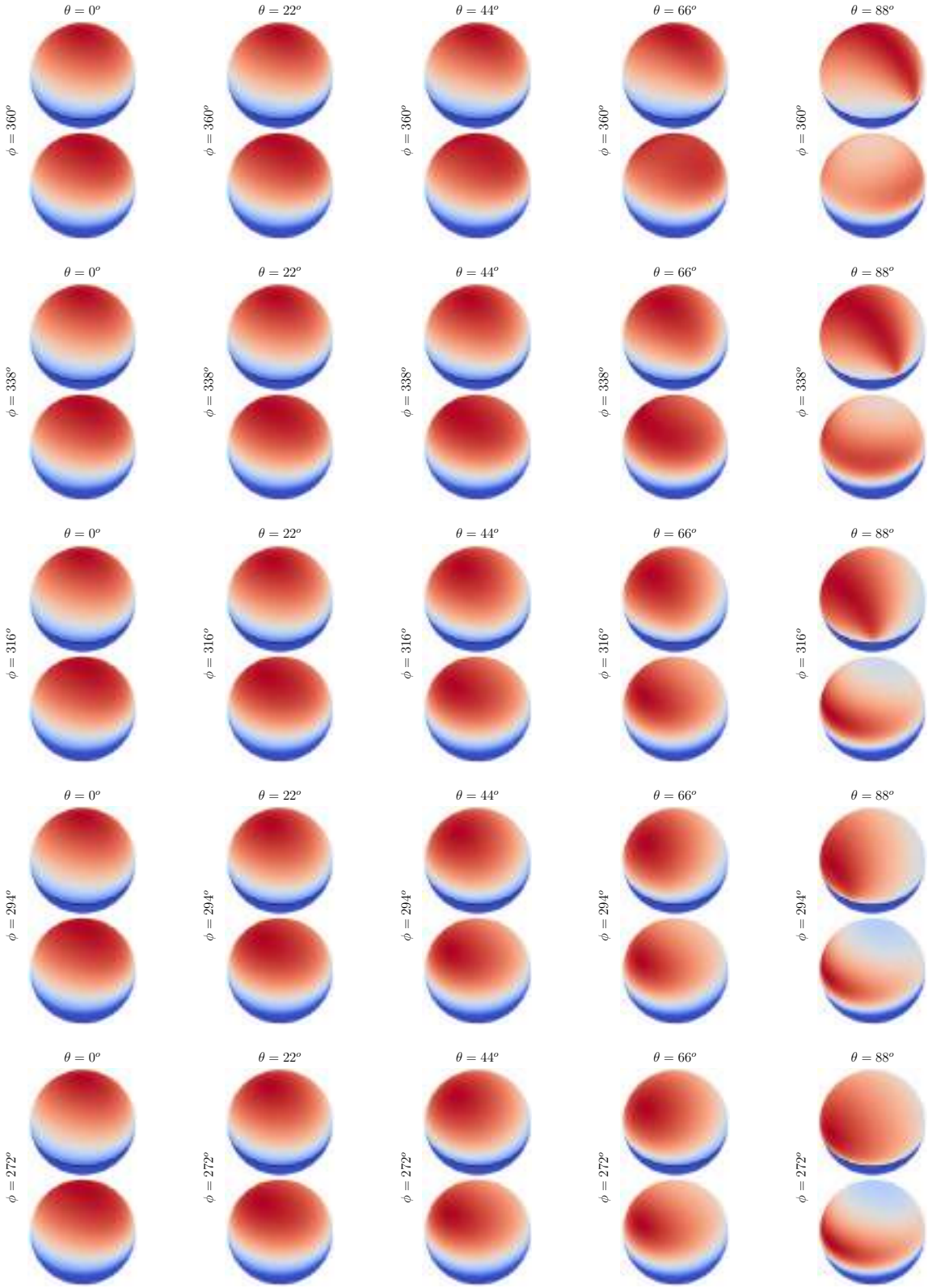
$\alpha_x : 1.000000$, $\alpha_y : 0.250000$



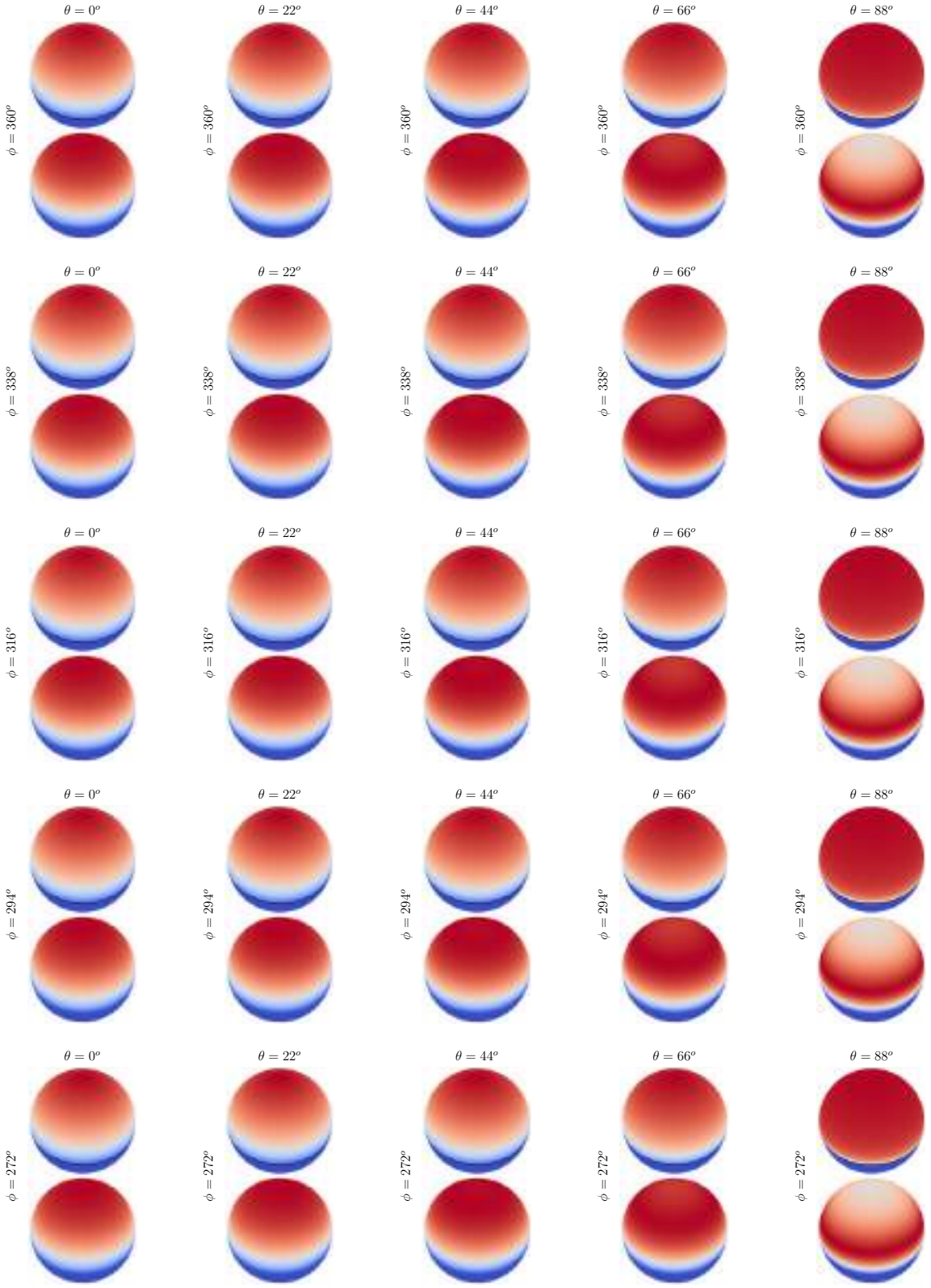
$\alpha_x : 1.000000$, $\alpha_y : 0.500000$



$\alpha_x : 1.000000$, $\alpha_y : 0.800000$



$\alpha_x : 1.000000$, $\alpha_y : 1.000000$



8.3 Supplementary material for Chapter 5

8.3.1 Sampling Inverse Cosine

In this section we derive the sampling routine for sampling for an inverse cosine distribution i.e $1 - \cos\theta$. We use the CDF inversion technique [1] to do this.
Since the PDF is not dependent on ϕ it can be sampled uniformly in range $[0, 2\pi]$. The PDF $p(\theta, \phi)$ is given by:

$$p(\theta, \phi) = \frac{1}{\pi}(1 - \cos\theta)\sin\theta.$$

The marginal PDF of $p(\theta)$ can is:

$$\begin{aligned} p(\theta) &= \int_0^{2\pi} p(\theta, \phi) d\phi \\ &= \int_0^{2\pi} \frac{1}{\pi}(1 - \cos\theta)\sin\theta d\phi \\ &= 2\sin\theta(1 - \cos\theta). \end{aligned}$$

The CDF $P(\theta)$ is given as:

$$\begin{aligned} P(\theta) &= \int_0^{\theta} 2\sin\theta(1 - \cos\theta) d\theta \\ &= 2\left[\int_0^{\theta} \sin\theta d\theta - \int_0^{\theta} \sin\theta\cos\theta d\theta\right] \\ &= 2\left[-\cos\theta\Big|_0^{\theta} + \frac{1}{2}\cos^2\theta\Big|_0^{\theta}\right] \\ &= 2\left[1 - \cos\theta + \frac{1}{2}\cos^2\theta - \frac{1}{2}\right] \\ &= 1 - 2\cos\theta + \cos^2\theta. \end{aligned}$$

Given a random number ξ sampled uniformly from $[0, 1]$, the θ can be found as:

$$\theta = P^{-1}(\xi),$$

where $P^{-1}(\xi)$ is the inverse of the CDF $P(\theta)$ which can be found as follows:

$$\begin{aligned}
\xi &= 1 - 2\cos\theta + \cos^2\theta \\
\xi &= (1 - \cos\theta)^2 \\
\pm\sqrt{\xi} &= (1 - \cos\theta) \\
1 \pm \sqrt{\xi} &= \cos\theta \\
\cos^{-1}(1 \pm \sqrt{\xi}) &= \theta.
\end{aligned}$$

Since, the argument of $\cos^{-1}$ should be in the range $0 - 1$, $P^{-1}(\xi) = \cos^{-1}(1 - \sqrt{\xi})$. Hence, given two uniformly distributed random numbers ξ_1, ξ_2 in $[0, 1]$, we can sample a θ, ϕ as follows:

$$\begin{aligned}
\phi &= 2\pi\xi_1 \\
\theta &= \cos^{-1}(1 - \sqrt{\xi_2}).
\end{aligned}$$

8.3.2 Rendering Algorithm

The pseudo-code of our unshadowed semi-analytic single scattering is given in Alg. 8. The entry point is the function `renderPixel` on line 27, which accepts the pixel's ray and the number of quadrature samples N as input. We get the surface intersection for this ray in line 28. Next, the air-light integral is estimated using quadrature (line 29) and surface color is computed and added to it (lines 30-32).

The function `airLight` estimates the air-light integral by first equally dividing the ray's extent into N parts (line 16), resulting in N point samples. For each sample, we get it's medium interaction and compute $\bar{L_s}$ (lines 18-21). This is then multiplied by the transmittance between the camera and that sample (line 22), before performing the quadrature sum. The obtained quantity after evaluating all samples is multiplied by μ_s, according to Eq. 5.15.

Next, the function `analyticInScatter` analytically evaluates $\overline{L_s}$ at a single point in the medium. We set the local-coordinate frame such that ω_o aligns with the z-axis (line 2). We then split the solid angle of the area light at the horizon (lines 4-5). The LTSD matrices are fetched using the value of g, along with their amplitudes (lines 6-7). We then analytically evaluate the in-scattered radiance using M_s and M_l according to Eq. 5.21. Finally, we multiply the comptued value by the analytic transmittance towards the area light (Eq. 5.10) and return (lines 12-13).

ALGORITHM 8: Semi-analytic unshadowed single scattering

Def *analyticInScatter(mei)*:
    setLocalCoord(mei.wo) // ω_o aligned with z-axis
    $\overline{L_s} = 0$
    A_1, A_2 = splitAtHorizon(A) // Split area light A
    S_1, S_2 = solidAngle(A_1), solidAngle(A_2)
    /* Fetch LTSD matrices with amplitudes */
    M_s, a_u = fetchLTSDUpper(mei.g)
    M_l, a_l = fetchLTSDLower(mei.g)
    /* Evaluate Eq. 5.21 */
    if *mei.g ¡ 0* **then**
        $\overline{L_s} = a_u \cdot E(M_s^{-1} A_1) + a_l \cdot [S_2 - E(M_l^{-1} A_2)]$
    else if *mei.g* $\geq$ *0* **then**
        $\overline{L_s} = a_u \cdot [S_1 - E(M_s^{-1} A_1)] + a_l \cdot E(M_l^{-1} A_2)$
    $\overline{L_s} = T(A) \cdot \overline{L_s}$ // Mul. by analytic transmittance (Eq. 5.10)
    return $\overline{L_s}$
Def *airLight(ray, si, N)*:
    $\overline{\mathcal{A}} = 0$
    Δz = si.tmax / N // Equally divide ray extent
    $\overline{Q_1}, \overline{Q_2} = 0$
    for *s in 0...N* **do**
        t = ray.o + s * ray.d
        mei = getMediumInteraction(t, ray)
        $\overline{L_s}$ = analyticInScatter(mei)
        $Q_2 = T(t) \cdot \overline{L_s}$ // Transmittance from Eq. 5.5
        if $s \neq 0$ **then**
            $\overline{\mathcal{A}}$ += $\frac{Q_1+Q_2}{2}\Delta z$ // Eq.5.15
        $Q_1 = Q_2$
    return $\mu_s \cdot \overline{\mathcal{A}}$
Def *renderPixel(ray, N)*:
    si = rayCast(ray)
    /* Air-light with N quadrature samples (Eq. 5.15) */
    $\overline{L}$ = airLight(ray, si, N)
    if *si.hit* **then**
        /* LTC & mult. by analytic transmittance (Eq. 5.10) */
        $\overline{L_c}$ = analyticSurface(si) $\cdot T(A)$ // A is the area light
        $\overline{L}$ += T(si.t) $\cdot \overline{L_c}$
    return $\overline{L}$

Bibliography

[1] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[2] S. Chandrasekhar, *Radiative transfer*. Courier Corporation, 2013.

[3] T. Müller, F. Rousselle, J. Novák, and A. Keller, "Real-time neural radiance caching for path tracing," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–16, 2021.

[4] E. Veach, *Robust Monte Carlo methods for light transport simulation*. Ph.D Thesis, Stanford University, 1998.

[5] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance, "Microfacet models for refraction through rough surfaces," in *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, ser. EGSR'07. Goslar, DEU: Eurographics Association, 2007, p. 195–206.

[6] E. Heitz, "Understanding the masking-shadowing function in microfacet-based brdfs," *Journal of Computer Graphics Techniques (JCGT)*, vol. 3, no. 2, pp. 48–107, 2014.

[7] S. Hill and E. Heitz, "Real-time area lighting: a journey from research to production," in *ACM SIGGRAPH Courses 2016*, 2016.

[8] E. Heitz, J. Dupuy, S. Hill, and D. Neubelt, "Real-time polygonal-light shading with linearly transformed cosines," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, Jul. 2016. [Online]. Available: https://doi.org/10.1145/2897824.2925895

[9] E. Heitz, "Sampling the ggx distribution of visible normals," *Journal of Computer Graphics Techniques (JCGT)*, vol. 7, no. 4, pp. 1–13, 2018.

[10] I. H. Lambert, "Photometria sive de mensura et gradibus luminis, colorum et umbrae," 1760.

[11] D. R. Baum, H. E. Rushmeier, and J. M. Winget, "Improving radiosity solutions through the use of analytically determined form-factors," in *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '89. New York, NY, USA: Association for Computing Machinery, 1989, p. 325–334. [Online]. Available: https://doi.org/10.1145/74333.74367

[12] E. Heitz, "Geometric Derivation of the Irradiance of Polygonal Lights," Unity Technologies, Research Report, Jan. 2017. [Online]. Available: https://hal.science/hal-01458129

[13] E. Heitz, S. Hill, and M. McGuire, "Combining analytic direct illumination and stochastic shadows," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ser. I3D '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3190834.3190852

[14] E. Heitz and S. Hill, "Linear-light shading with linearly transformed cosines," in *GPU Zen*, 2017.

[15] E. Heitz and S. Hill, "Real-time line and disk-light shading with linearly transformed cosines." ACM SIGGRAPH Courses 2017.

[16] J. 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*, ser. SIGGRAPH '95. New York, NY, USA: Association for Computing Machinery, 1995, p. 335–342. [Online]. Available: https://doi.org/10.1145/218380.218467

[17] P. Lecocq, G. Sourimant, and J.-E. Marvie, "Accurate analytic approximations for real-time specular area lighting," in *ACM SIGGRAPH 2015 Talks*, 2015, pp. 68:1–68:1.

[18] J. Dupuy, E. Heitz, and L. Belcour, "A spherical cap preserving parameterization for spherical distributions," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, Jul. 2017. [Online]. Available: https://doi.org/10.1145/3072959.3073694

[19] J. Wang and R. Ramamoorthi, "Analytic spherical harmonic coefficients for polygonal area lights," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, Jul. 2018. [Online]. Available: https://doi.org/10.1145/3197517.3201291

[20] L. Wu, G. Cai, S. Zhao, and R. Ramamoorthi, "Analytic spherical harmonic gradients for real-time rendering with many polygonal area lights," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, Jul. 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392373

[21] L. Belcour, G. Xie, C. Hery, M. Meyer, W. Jarosz, and D. Nowrouzezahrai, "Integrating clipped spherical harmonics expansions," *ACM Transactions on Graphics (Presented at SIGGRAPH)*, vol. 37, no. 2, Mar. 2018.

[22] Y. Zhou, L. Wu, R. Ramamoorthi, and L.-Q. Yan, "Vectorization for fast, analytic, and differentiable visibility," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 3, pp. 27:1–27:21, 2021.

[23] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte, "A developer's guide to silhouette algorithms for polygonal models," *IEEE Comput. Graph. Appl.*, vol. 23, no. 4, p. 28–37, Jul. 2003. [Online]. Available: https://doi.org/10.1109/MCG.2003.1210862

[24] J. P. Snyder, *Map projections–A working manual*. US Government Printing Office, 1987, vol. 1395.

[25] G. Greiner and K. Hormann, "Efficient clipping of arbitrary polygons," *ACM Transactions on Graphics (TOG)*, vol. 17, no. 2, pp. 71–83, 1998.

[26] B. R. Vatti, "A generic solution to polygon clipping," *Commun. ACM*, vol. 35, no. 7, p. 56–63, Jul. 1992. [Online]. Available: https://doi.org/10.1145/129902.129906

[27] M. Olson and H. Zhang, "Silhouette extraction in hough space," *Comput. Graph. Forum*, vol. 25, pp. 273–282, 2006.

[28] B. Bitterli, C. Wyman, M. Pharr, P. Shirley, A. Lefohn, and W. Jarosz, "Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting," *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, vol. 39, no. 4, Jul. 2020.

[29] S. Hill, S. McAuley, L. Belcour, W. Earl, N. Harrysson, S. Hillaire, N. Hoffman, L. Kerley, J. Patry, R. Pieké, I. Skliar, J. Stone, P. Barla, M. Bati, and I. Georgiev, "Physically based shading in theory and practice," in *ACM SIGGRAPH 2020 Courses*, 2020.

[30] A. Benyoub, "Leveraging ray tracing hardware acceleration in unity," in *ACM SIGGRAPH Courses 2019*, 2019.

[31] M. Wassmer, J. Platteaux, A. Schober, and I. Llamas, "Cinematic lighting in unreal engine," in *Game Developers Conference 2018*, 2018.

[32] J. M. Snyder, "Area light sources for real-time graphics," Microsoft Research, Tech. Rep. MSR-TR-96-11, 1996.

[33] L. Wang, Z. Lin, W. Wang, and K. Fu, "One-shot approximate local shading," Tech. Rep., 2008.

[34] M. Drobot, "Physically based area lights," in *GPU Pro 5*, 2014, pp. 67–100.

[35] S. Lagarde and C. de Rousiers, "Physically based shading in theory and practice: Moving Frostbite to PBR," in *ACM SIGGRAPH Courses 2014*, 2014.

[36] A. KT, P. Sakurikar, and P. J. Narayanan, "Fast Analytic Soft Shadows from Area Lights," in *Eurographics Symposium on Rendering - DL-only Track*, A. Bousseau and M. McGuire, Eds. The Eurographics Association, 2021.

[37] S. Diolatzis, A. Gruson, W. Jakob, D. Nowrouzezahrai, and G. Drettakis, "Practical Product Path Guiding Using Linearly Transformed Cosines," *Computer Graphics Forum*, 2020.

[38] C. Peters, "Brdf importance sampling for polygonal lights," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, 2021.

[39] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable monte carlo ray tracing through edge sampling," *ACM Transactions on Graphics (TOG) (Proc. SIGGRAPH Asia)*, vol. 37, no. 6, 2018.

[40] A. Ngan, F. Durand, and W. Matusik, "Experimental analysis of brdf models," in *Proceedings of the Eurographics Symposium on Rendering*, 2005, pp. 117–226.

[41] J. Dupuy and W. Jakob, "An adaptive parameterization for efficient material acquisition and rendering," *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, vol. 37, no. 6, pp. 274:1–274:18, 2018.

[42] J. Rabin, G. Peyré, J. Delon, and M. Bernot, "Wasserstein barycenter and its application to texture mixing," in *Scale Space and Variational Methods in Computer Vision*, 2012, pp. 435–446.

[43] N. Bonneel, J. Rabin, G. Peyré, and H. Pfister, "Sliced and radon wasserstein barycenters of measures," *J. Math. Imaging Vis.*, vol. 51, no. 1, p. 22–45, 2015.

[44] S. Kolouri, G. K. Rohde, and H. Hoffmann, "Sliced wasserstein distance for learning gaussian mixture models," in *Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 2018.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[46] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[47] D. Lin, C. Wyman, and C. Yuksel, "Fast volume rendering with spatiotemporal reservoir resampling," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, dec 2021.

[48] C. Kulla and M. Fajardo, "Importance sampling techniques for path tracing in participating media," *Computer Graphics Forum*, vol. 31, no. 4, pp. 1519–1528, 2012.

[49] K. Villeneuve, A. Gruson, I. Georgiev, and D. Nowrouzezahrai, "Practical Product Sampling for Single Scattering in Media," in *Eurographics Symposium on Rendering - DL-only Track*, A. Bousseau and M. McGuire, Eds., 2021.

[50] J. F. Talbot, D. Cline, and P. Egbert, "Importance resampling for global illumination," in *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, ser. EGSR '05, Goslar, DEU, 2005, p. 139–146.

[51] B. Sun, R. Ramamoorthi, S. G. Narasimhan, and S. K. Nayar, "A practical analytic single scattering model for real time rendering," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, p. 1040–1049, jul 2005.

[52] V. Pegoraro and S. G. Parker, "An analytical solution to single scattering in homogeneous participating media," *Computer Graphics Forum*, vol. 28, no. 2, pp. 329–335, 2009.

[53] V. Pegoraro, M. Schott, and S. G. Parker, "Reduced dual-formulation for analytical anisotropic single scattering," in *HPG Posters*, 2009.

[54] V. Pegoraro, M. Schott, and P. Slusallek, "A mathematical framework for efficient closed-form single scattering," in *Proceedings of Graphics Interface 2011*, ser. GI '11, Waterloo, CAN, 2011, p. 151–158.

[55] V. Pegoraro, M. Schott, and S. G. Parker, "An analytical approach to single scattering for anisotropic media and light distributions," in *Proceedings of Graphics Interface 2009*, ser. GI '09, CAN, 2009, p. 71–77.

[56] ——, "A closed-form solution to single scattering for general phase functions and light distributions," *Computer Graphics Forum*, vol. 29, no. 4, pp. 1365–1374, 2010.

[57] C. Wyman and S. Ramsey, "Interactive volumetric shadows in participating media with single-scattering," in *2008 IEEE Symposium on Interactive Ray Tracing*, 2008, pp. 87–92.

[58] C. Wyman, "Voxelized shadow volumes," in *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, ser. HPG '11, New York, NY, USA, 2011, p. 33–40.

[59] M. Billeter, E. Sintorn, and U. Assarsson, "Real time volumetric shadows using polygonal light volumes," in *Proceedings of the Conference on High Performance Graphics*, ser. HPG '10, Goslar, DEU, 2010, p. 39–45.

[60] J. Chen, I. Baran, F. Durand, and W. Jarosz, "Real-time volumetric shadows using 1d min-max mipmaps," in *Symposium on Interactive 3D Graphics and Games*, ser. I3D '11, New York, NY, USA, 2011, p. 39–46.

[61] T. Engelhardt and C. Dachsbacher, "Epipolar sampling for shadows and crepuscular rays in participating media with single scattering," in *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ser. I3D '10, New York, NY, USA, 2010, p. 119–125.

[62] O. Klehm, H.-P. Seidel, and E. Eisemann, "Prefiltered single scattering," in *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ser. I3D '14, New York, NY, USA, 2014, p. 71–78.

[63] I. N. Shah, A. KT, and P. J. Narayanan, "Combining resampled importance and projected solid angle samplings for many area light rendering," in *SIGGRAPH Asia 2023 Technical Communications*, ser. SA '23, New York, NY, USA, 2023.

[64] J. Novák, I. Georgiev, J. Hanika, and W. Jarosz, "Monte Carlo methods for volumetric light transport simulation," *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)*, vol. 37, no. 2, May 2018.

[65] A. KT, E. Heitz, J. Dupuy, and P. J. Narayanan, "Bringing linearly transformed cosines to anisotropic ggx," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 5, no. 1, May 2022. [Online]. Available: https://doi.org/10.1145/3522612

[66] P. Debevec, "Image-based lighting," in *ACM SIGGRAPH 2005 Courses*, ser. SIGGRAPH '05, New York, NY, USA, 2005, p. 3–es.

[67] L. G. Henyey and J. L. Greenstein, "Diffuse radiation in the Galaxy." vol. 93, pp. 70–83, Jan. 1941.

[68] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich, "Optix: A general purpose ray tracing engine," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, Jul. 2010. [Online]. Available: https://doi.org/10.1145/1778765.1778803

[69] E. Heitz, J. Dupuy, C. Crassin, and C. Dachsbacher, "The sggx microflake distribution," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, jul 2015.

[70] J. T. Moon and S. R. Marschner, "Simulating multiple scattering in hair using a photon mapping approach," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 1067–1074, 2006.

[71] J. T. Moon, B. Walter, and S. Marschner, "Efficient multiple scattering in hair using spherical harmonics," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, pp. 1–7, 2008.

[72] L.-Q. Yan, W. Sun, H. W. Jensen, and R. Ramamoorthi, "A bssrdf model for efficient rendering of fur with global illumination," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, 2017.

[73] A. Zinke, C. Yuksel, A. Weber, and J. Keyser, "Dual scattering approximation for fast multiple scattering in hair," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, pp. 1–10, 2008.

[74] J. Meng, M. Papas, R. Habel, C. Dachsbacher, S. Marschner, M. Gross, and W. Jarosz, "Multi-scale modeling and rendering of granular materials," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–13, 2015.

[75] T. Müller, M. Papas, M. Gross, W. Jarosz, and J. Novák, "Efficient rendering of heterogeneous polydisperse granular media," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, 2016.

[76] S. R. Marschner, H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan, "Light scattering from human hair fibers," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 780–791, 2003.

[77] A. Zinke and A. Weber, "Light scattering from filaments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 342–356, 2007.

[78] E. d'Eon, G. Francois, M. Hill, J. Letteri, and J.-M. Aubry, "An energy-conserving hair reflectance model," in *Computer Graphics Forum*, vol. 30, no. 4. Wiley Online Library, 2011, pp. 1181–1187.

[79] W. Huang, M. B. Hullin, and J. Hanika, "A microfacet-based hair scattering model," in *Computer Graphics Forum*, vol. 41, no. 4. Wiley Online Library, 2022, pp. 79–91.

[80] M. Xia, B. Walter, E. Michielssen, D. Bindel, and S. Marschner, "A wave optics based fiber scattering model," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–16, 2020.

[81] P. Khungurn and S. Marschner, "Azimuthal scattering from elliptical hair fibers," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 2, pp. 1–23, 2017.

[82] A. Benamira and S. Pattanaik, "A combined scattering and diffraction model for elliptical hair rendering," in *Computer Graphics Forum*, vol. 40, no. 4. Wiley Online Library, 2021, pp. 163–175.

[83] L. Pekelis, C. Hery, R. Villemin, and J. Ling, "A data-driven light scattering model for hair," *Pixar Technical Memo*, vol. 2, 2015.

[84] I. Sadeghi, H. Pritchett, H. W. Jensen, and R. Tamstorf, "An artist friendly hair shading system," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, pp. 1–10, 2010.

[85] M. J.-Y. Chiang, B. Bitterli, C. Tappan, and B. Burley, "A practical and controllable hair and fur model for production path tracing," in *Computer Graphics Forum*, vol. 2, no. 35, 2016, pp. 275–283.

[86] L. Fascione, J. Hanika, R. Pieké, R. Villemin, C. Hery, M. Gamito, L. Emrose, and A. Mazzone, "Path tracing in production," in *ACM SIGGRAPH 2018 Courses*, 2018, pp. 1–79.

[87] J. Zhu, S. Zhao, L. Wang, Y. Xu, and L.-Q. Yan, "Practical level-of-detail aggregation of fur appearance," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–17, 2022.

[88] D. R. Wyman, M. S. Patterson, and B. C. Wilson, "Similarity relations for the interaction parameters in radiation transport," *Applied optics*, vol. 28, no. 24, pp. 5243–5249, 1989.

[89] S. Zhao, R. Ramamoorthi, and K. Bala, "High-order similarity relations in radiative transfer," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–12, 2014.

[90] S. Kallweit, T. Müller, B. Mcwilliams, M. Gross, and J. Novák, "Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–11, 2017.

[91] D. Vicini, V. Koltun, and W. Jakob, "A learned shape-adaptive subsurface scattering model," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–15, 2019.

[92] C. Che, F. Luan, S. Zhao, K. Bala, and I. Gkioulekas, "Towards learning-based inverse subsurface scattering," in *2020 IEEE International Conference on Computational Photography (ICCP)*. IEEE, 2020, pp. 1–12.

[93] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, "Neural importance sampling," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, pp. 1–19, 2019.

[94] T. Müller, F. Rousselle, A. Keller, and J. Novák, "Neural control variates," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–19, 2020.

[95] L. Szécsi, L. Szirmay-Kalos, and C. Kelemen, "Variance reduction for russian-roulette," 2003.

[96] J. Arvo and D. Kirk, "Particle transport and image synthesis," in *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, 1990, pp. 63–66.

[97] J. Vorba and J. Křivánek, "Adjoint-driven russian roulette and splitting in light transport simulation," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–11, 2016.

[98] A. Rath, P. Grittmann, S. Herholz, P. Weier, and P. Slusallek, "Ears: efficiency-aware russian roulette and splitting," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–14, 2022.

[99] D. Lin, M. Kettunen, B. Bitterli, J. Pantaleoni, C. Yuksel, and C. Wyman, "Generalized resampled importance sampling: foundations of restir," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–23, 2022.

[100] T. Müller, "tiny-cuda-nn," 4 2021. [Online]. Available: https://github.com/NVlabs/tiny-cuda-nn

[101] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," in *Proceedings of the 35th International Conference on Machine Learning, PMLR*, vol. 80, 2018.

[102] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127