FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Echtzeit Approximation von Photometrischen Flächenlichtquellen für Interaktives Lichtdesign

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Visual Computing**

eingereicht von

**Lukas Tobias Prost, BSc**
Matrikelnummer 01225511

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Dipl.-Ing. Christian Luksch

Wien, 7. März 2019

Lukas Tobias Prost Michael Wimmer

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Real-Time Rendering of Photometric Area Lights for Interactive Lighting Design

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Visual Computing**

by

**Lukas Tobias Prost, BSc**
Registration Number 01225511

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Dipl.-Ing. Christian Luksch

Vienna, 7th March, 2019

| Lukas Tobias Prost | Michael Wimmer |
|---|---|

# Erklärung zur Verfassung der Arbeit

Lukas Tobias Prost, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 7. März 2019

Lukas Tobias Prost

# Danksagung

Ich möchte mich bei allen bedanken, die bei dieser Arbeit mitgewirkt haben. Zum einen bei meinem Betreuer Michael Wimmer für die großartige Unterstützung und das wertvolle Feedback. Weiters bei Christian Luksch, weil er immer Zeit für lange Diskussionen hatte und er mit seinem Wissen und seiner Erfahrung viel zur Diplomarbeit beigetragen hat.

Ich möchte mich auch bei Harald Steinlechner, Attila Szabo und Georg Haaser für die Unterstützung und Hilfe bei der Implementierung bedanken, wie auch bei Andreas Walch, Michael Schwärzler und Stefan Maierhofer für die allgemeine Unterstützung.

Abschließend möchte ich mich noch bei meiner Familie und meinen Freunden dafür bedanken, dass sie mich während der Arbeit unterstützt und angetrieben haben.

# Kurzfassung

Photometrische Flächenlichtquellen sind Lichtquellen mit einer Abstrahlcharakteristik, die denen von Leuchten aus der echten Welt gleicht. Sie haben eine wichtige Rolle im Lichtdesign und der Lichtplanung, wo mit ihrer Hilfe Beleuchtung durch Echtweltleuchten simuliert wird. Um akkurate Ergebnisse zu erhalten, können offline Algorithmen wie Path Tracing verwendet werden, um die globale Beleuchtung durch diese Lichter fehlerfrei auszuwerten. Um jedoch Lichtdesignern eine interaktive Arbeitsweise zu ermöglichen, ist es notwendig, photometrische Lichter auch in Echtzeit auswerten zu können. Aktuelle Echtzeit-Lösungen sind jedoch fehlerbehaftet, wenn eine Lichtquelle zu nah an beleuchteten Objekten ist.

In dieser Diplomarbeit präsentieren wir eine neue Technik zur Auswertung von photometrischen Flächenlichtquellen in Echtzeit. Die Technik basiert auf der Kombination von zwei Sampling Methoden, die derzeit in modernen Game Engines zur Approximation von diffusen Flächenlichtquellen verwendet werden. Flächenlichtquellen werden zuerst mit der kombinierten Sampling Methode gesampelt, danach wird die Beleuchtung mit einer Kubatur Technik basierend auf der Delaunay Triangulation berechnet. Um diese Berechnung in Echtzeit durchzuführen, haben wir unsere Technik auf der GPU implementiert und dazu eine kompakte Dreieck-Datenstruktur entwickelt, welche es ermöglicht, eine Delaunay Triangulation effizient auf der GPU zu generieren.

Das Ergebnis dieser Diplomarbeit ist eine neue Technik, welche photometrische Flächenlichtquellen visuell plausibel in Echtzeit annähern kann, wenn die Lichtquelle nah bei beleuchteten Objekten ist.

# Abstract

Photometric area light sources are modeled after real-world luminaires and are used in lighting design to accurately simulate lighting. An accurate evaluation of their illumination can be computed with offline global-illumination algorithms. However, such algorithms take time to compute a solution. Therefore, real-time approximations are required to enable an interactive lighting-design workflow. However, currently used techniques are prone to errors when the light source is close to illuminated objects.

In this thesis, we present a new technique to approximate photometric area lights in real time. This new technique is based on combining two sampling strategies that are currently used in game engines to approximate the illumination from diffuse area lights. Our technique samples the photometric area light with this combined sampling strategy and then computes the illumination with a cubature technique based the Delaunay triangulation. To do this in real time, we implemented our method on the GPU and developed a compact triangle data structure that enables an efficient generation of a Delaunay triangulation.

The result of this thesis is a new technique for photometric area lights that creates visually plausible approximations in real time when the light source is close to illuminated objects.

# Contents

CHAPTER 1

# Introduction

## 1.1 Motivation

Lighting Design is a part in the planning process of buildings and entertainment shows, and poses a challenge both in technical and artistic terms. It is the process of placing luminaires in a building such that lighting requirements are met, while keeping energy consumption and costs low. Modern lighting design software tools like HILITE, DIALux and Relux are used to create lighting solutions with light sources that have the emission characteristics of real-world luminaires. Such light sources are referred to as *Photometric Light Sources*, and the emission characteristics are provided by a *Photometric Report*, which describes how radiance is emitted by a luminaire. An example for a photometric light source with its radiance emission visualized is shown in Figure 1.1.

The lighting design tools discussed above render images with physically accurate global illumination, such that a designer can evaluate whether the illumination meets given requirements in the real world. For this task, offline rendering techniques like radiosity or photon mapping are used to simulate light transport. Having a correct and accurate light-transport simulation technique guarantees the correct evaluation of complex light interactions.

However, these offline techniques take time to compute a solution which slows down the working process of lighting designers. Even slight position changes of a light source require a costly reevaluation of the lighting in a scene. Providing a solution that allows accurate previews in real time improves the workflow of the designers, as it allows them to receive feedback instantly, which facilitates rapid prototyping.

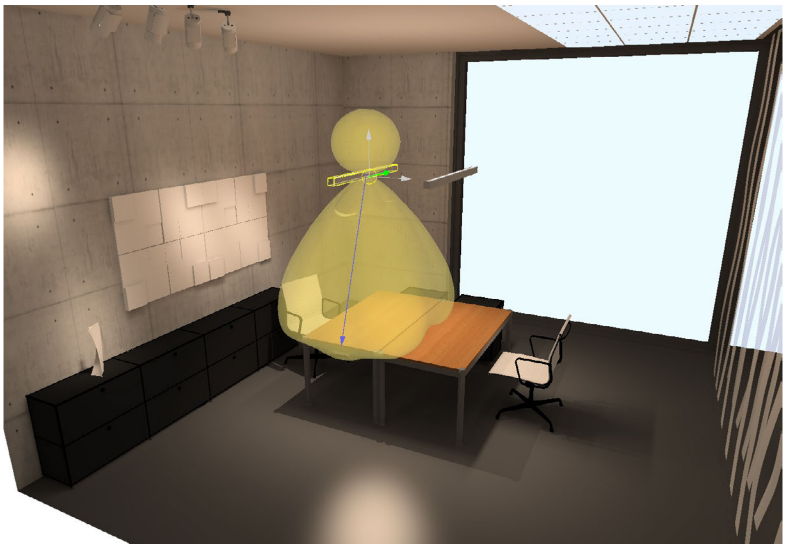


Figure 1.1: Scene rendered in Hilite, where the emission profile of a photometric light source is visualized as a yellow volume.

## 1.2 Problem Statement

The problem with evaluating photometric light sources in real time is rooted in them being based on real-world luminaires. Real-world luminaires are complex light sources, which means that light is generally emitted not uniformly and that the emission is not centered at a point but distributed over an area or a volume. Luminaires may define a light-emitting object (LEO), which describe the geometric object from which the light is emitted. While evaluating photometric point light sources is possible in real time, non-zero dimensional photometric light sources currently prove to be a problem.

Techniques exist to evaluate the irradiance by planar, polygonal light sources (area lights). These techniques are accurate and efficient enough to be used in applications with real-time constraints. The existing methods, however, are limited to area lights with a diffuse or Phong-like emission profile. Photometric area lights, where the emission can be arbitrary, are not taken into account by these techniques.

To still provide an interactive workflow for lighting designers, HILITE, DIALux and Relux use various approximations to provide real-time previews of scenes illuminated by photometric light sources. The currently used approximations, however, suffer either from inaccuracies when the light source is near an object it illuminates, or are more accurate but evaluable in real time and just interactive. An example for the approximation used by Hilite and DIALux is shown in Figure 1.2. To provide a real-time preview, the photometric light is replaced with a single photometric point light, which is placed at the center of the photometric light's LEO. While this approximation is visually plausible for large distances between the light source and the illuminated surfaces, errors become noticeable as soon as the distance decreases.

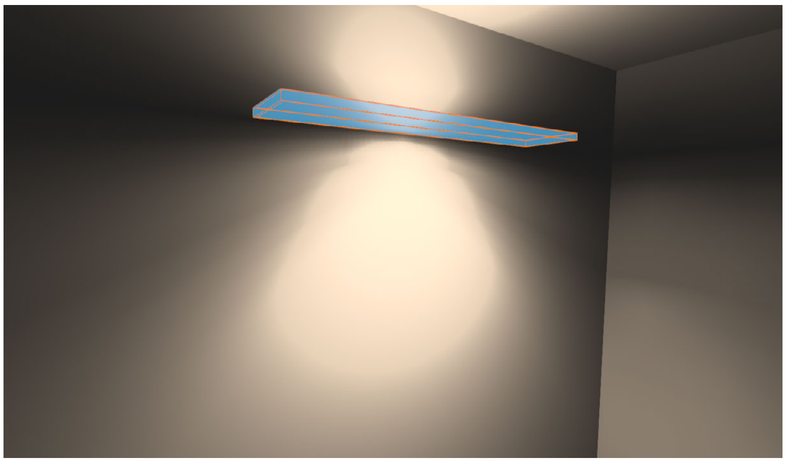


Figure 1.2: DIALux approximates a luminaire in real time by placing a single photometric area light in the center of the light's LEO.

Hilite addresses these problems by using more photometric point lights at smaller distances. Patterns, however, remain visible as the number of point lights that can be used is limited in real-time applications. Relux generally uses many photometric point lights to avoid these artifacts as long as possible. Yet, the high number of used photometric point lights does not allow them to evaluate the illumination in real time. After changing a photometric light in the scene, the software requires a short time to reevaluate the lighting setup.

As current real-time area-light evaluation techniques are designed for area lights with diffuse or Phong-like radiance emission, and approximations used by lighting design software have either artifacts when they are close to an illuminated surface or do not provide real-time performance, there is currently no technique which returns accurate approximations for photometric light sources close to illuminated surfaces.

## 1.3 Contributions

The main contribution of this diploma thesis is a novel real-time approximation for photometric area lights that can be used in interactive lighting design, but also other real-time systems like games. This new approximation emulates the diffuse light transport and handles rotationally symmetric and asymmetric photometric lights with only a small number of samples. Approximations rendered with this new technique are visually more plausible than solutions presented by modern lighting design software.

Given the option to have plausible illumination from photometric area lights during the design process, designers can develop new lighting systems even faster and more intuitively. The possible new lighting design process will be inspired by the established interactive workflows provided by modern game-engine editors, and thus reduce the development time for new lighting systems and prototypes.

Visibility is out of scope of this thesis, meaning that the focus is on unblocked direct lighting, which describes the light transport from the light source to the camera with exactly one light bounce in the scene with a light source which is not occluded. Occlusions of the light source, and therefore shadows, are not addressed. This thesis also only considers the illumination of diffuse surfaces. The illumination of glossy or specular surfaces will be left for future work.

The presented approximation addresses only photometric lights with polygonal, planar LEOs, to which we refer to as photometric area lights. While LEOs can by definition be volumetric objects like boxes or cylinders, volumetric LEOs are not common. Lighting design software like Hilite approximates volumetric LEOs with planar LEOs, which makes providing a solution for planar LEOs sufficient.

## 1.4 Structure of the Work

Starting in Chapter 2, general information regarding lighting design, photometry and direct lighting is given. The chapter explains the quantities used in lighting design, what information photometric reports contain and what information is required to use photometric lights in a scene. This section also provides a mathematical formulation for direct lighting, as this is the form of light transport our work focuses on. Besides the mathematical formulation, it is also explained how it can be computed efficiently with Monte Carlo integration. Following in Chapter 3 the related work is described. It provides a concise overview of global-illumination algorithms, which handle area lights implicitly, and then explains past and current work regarding the evaluation of area lights. Chapter 4 goes into more detail regarding real-time direct lighting by area lights and explains two sampling techniques currently used in game engines for diffuse area lights, on which our presented technique is based. Chapter 5 presents our new approach, which we named *scattered sampling*. The results are presented in Chapter 6, where we analyze scattered sampling with a variety of luminaires. This thesis concludes in Chapter 7.

CHAPTER 2

# Background

This chapter provides an introduction to two core topics of this thesis: *Photometry* and *direct lighting.* Starting with photometry, Section 2.1 gives an overview on the common terms, units and quantities. Following, Section 2.2 concisely explains photometry and its role in the context of luminaires. Section 2.3 explains how photometric data is made available for rendering and light evaluation and Section 2.4 describes how photometric lights are used in lighting design software.

Then, before this chapter goes on to direct lighting, it provides an introduction to Monte Carlo integration in Section 2.5. Monte Carlo integration provides the mathematical basis for sampling-based rendering techniques and has an important role for the sampling-based direct-lighting algorithms which are discussed in this thesis. Then, direct lighting is explained in Section 2.6. The section provides a mathematical formulation for direct lighting, and explains how it can be computed accurately with Monte Carlo integration.

## 2.1 Quantities and Units in Lighting Design

In terms of physics, light sources emit *Radiant Energy* $Q_e$, which is measured in *Joule*. *Radiant Flux* $\Phi_e$ is used to measure the rate of flow of radiant energy. Its definition is shown in Equation 2.1. The unit of radiant flux is *Watt* (Joule per second). [DHMS11]

$$\Phi_e = \frac{dQ_e}{dt} \tag{2.1}$$

While these quantities can be used to describe radiant power transfer, they are not suitable to describe radiant power as it is perceived by the human eye. For one, the photoreceptors in our eyes (rods and cones) only react to wavelengths in the interval of approximately $[380, 780]$ $nm$. Moreover, the magnitude of the photochemical reaction varies in this interval, meaning that the same spectral power at different wavelengths can be perceived unequally. How the human photoreceptors react to certain wavelengths is defined by an *Action Spectrum*. This action spectrum provides information about the photochemical effect of optical radiation at a certain wavelength. The action spectrum for humans is defined by a standard observer response curve, which was developed by the Commission Internationale de l'Eclairage (CIE) [DHMS11]. This response curve is given as the *Luminous Efficiency Function* $V(\lambda)$. The luminous efficiency function for daylight vision is plotted in Figure 2.1. More information about the function is provided by [dl90].

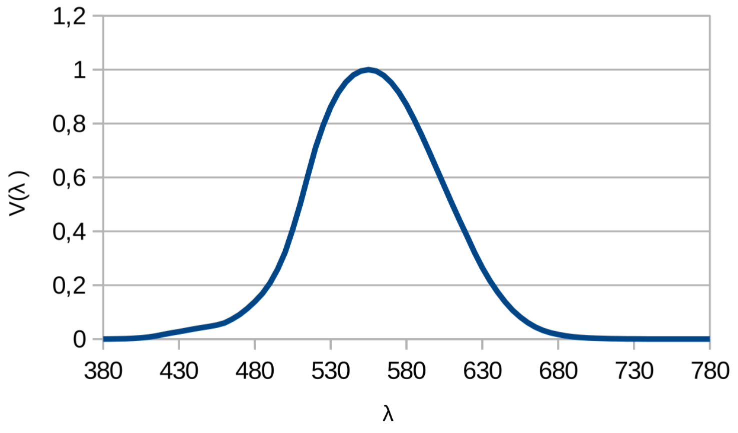


Figure 2.1: The luminous efficiency function for daylight for the standard observer. Plot generated with the data provided by [web18a].

Radiant power that was evaluated by the human visual system is referred to as *Light*. Since the physical radiation quantities described above are not defined for light, other quantities are used to describe and measure it [DHMS11]. Following, the photometric units used to describe and measure light are listed with their radiometric counterparts given in the parentheses:

**Luminous Flux $\Phi$ (Radiant Flux)** is the base unit to measure light in lighting design. It is measured in *Photopic Lumen* (lm) [DHMS11]. Luminous flux is derived by weighting the spectral power distribution with the luminous efficiency function as shown in Equation 2.2 [iec18a]. $K_m$ is the luminous efficacy of radiation for daylight vision. It is a constant with the value 683 $lm \cdot W^{-1}$. [iec18b]

$$\Phi = K_m \int_0^\infty \frac{d\Phi_e(\lambda)}{d\lambda} V(\lambda) d\lambda \tag{2.2}$$

**Illuminance $E$ (Irradiance)** measures the incident luminous flux on a surface point from all directions and is defined as shown in Equation 2.3. It provides no information regarding the direction of the incoming luminous flux (example shown in Figure 2.2). The unit of illuminance is *Lux* (lumens per square meter). [DHMS11]

$$E = \frac{d\Phi}{dA} \tag{2.3}$$

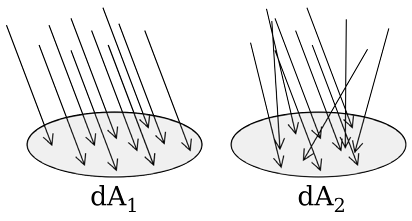


Figure 2.2: Illustration of illuminance. Both areas, $dA_1$ and $dA_2$, have the same illuminance, despite receiving the luminous flux from different directions.

**Average Illuminance** $\overline{E}$ defines the incident luminous flux over an area $A$ (see Equation 2.4)

$$\overline{E} = \frac{\Phi}{A} = \frac{1}{A}\int_A E dA \tag{2.4}$$

**Exitance** $M$ **and Average Exitance** $\overline{M}$ are defined equivalently to illuminance and average illuminance for exitant luminous flux. [DHMS11]

**Solid angle** $\omega$ is the extension of radians into the 3D space. It provides a measure for a spatial extend (the perceived size of an object observed from a point) and is measured in *Steradians* (sr).

The mathematical definition is given in Equation 2.5, where $A$ is the surface and $dA$ a surface point, $D$ is the distance from this surface point to the point of regard and $\theta$ is the angle between the normal vector at $dA$ and the vector from $dA$ to the point of regard. The corresponding illustration is shown in Figure 2.3. The solid angle is used to formulate flow of luminous flux in space. [DHMS11]

$$d\omega = \frac{dA\ \cos\theta}{D^2}\ ;\ \omega = \int_A \frac{\cos\theta}{D^2} dA \tag{2.5}$$

**Luminous Intensity** $I$ **(Radiant Intensity)** is defined as the amount of luminous flux in a certain direction in space. It is measured in *Candela* (cd), and the direction is specified by spherical coordinates. The mathematical definition is given in Equation 2.6. [DHMS11]

$$I(\theta, \psi) = \frac{d\Phi(\theta, \psi)}{d\omega} \tag{2.6}$$

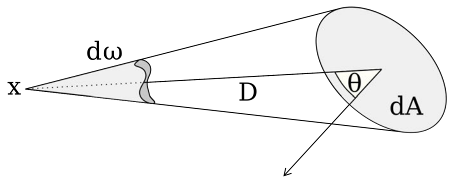


Figure 2.3: Illustration of a solid angle with $x$ as point of regard.

**Luminance $L$ (Radiance)** is defined as the luminous intensity per unit area as shown in Equation 2.7. Its unit is candela per meter-square (nit). Luminance is used in lighting design to describe the luminous flux emitted by a surface. The mathematical definition for the *Average Luminance* $\overline{L}$ is shown in Equation 2.8. [DHMS11]

$$L(\theta, \psi) = \frac{dI(\theta, \psi)}{dA \ \cos\theta} = \frac{d^2\Phi}{d\omega \ dA \ \cos\theta} \tag{2.7}$$

$$\overline{L} = \int_A \frac{dI(\theta, \psi)}{dA \ \cos\theta} \tag{2.8}$$

## 2.2 Photometry

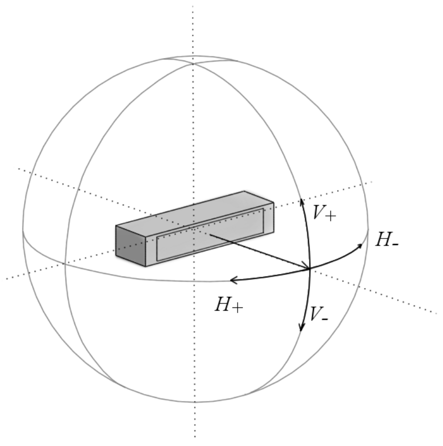
*Photometry* is the measurement of light. It is a branch of radiometry that is concerned with the measurement and observation of optical radiation. Photometry is used to evaluate, describe and characterize the photometric performance of luminaires. This is crucial information for lighting design and illumination engineering. [DHMS11]

The performance of a luminaire is described in a *Photometric Report.* In general, the following information is provided [DHMS11]:

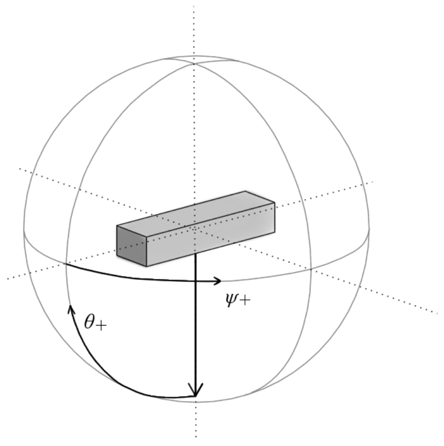
**Luminous Intensity Distribution** describes the light distribution characteristics of a luminaire. Luminous intensity is listed for various directions, which are specified in spherical coordinates: $\theta$ (altitude) and $\psi$ (azimuth). There exist three methods to measure and describe a luminous intensity distribution of which two are described in more detail because of their relevance for lighting design:

- **Type B Goniometry**

  Type B is generally used to describe outdoor lights. The spherical coordinates are referred to as $V$ and $H$. Both have their origin in the main axis of the luminaire. The coordinate system is orientated such that the main axis lies in the horizontal



(a) Type B Goniometry



(b) Type C Goniometry

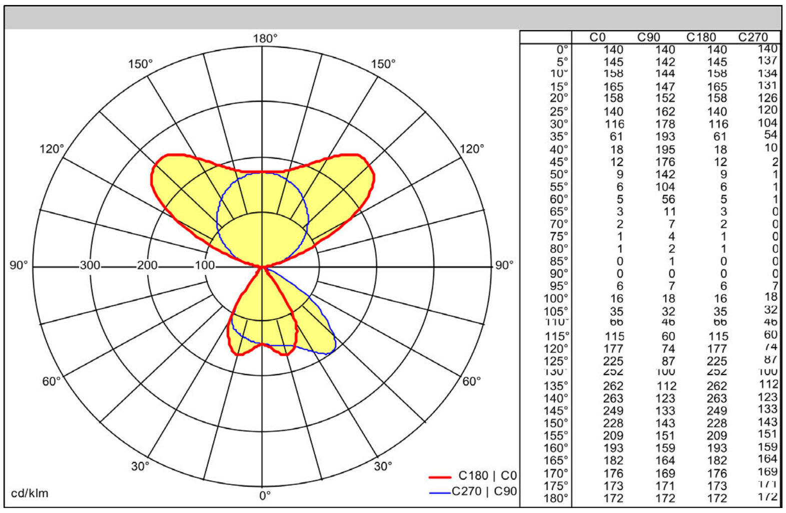
Figure 2.4: Type B goniometry (a): The main axis varies per luminaire. It is aligned in the coordinate system such that the main axis lies in the horizontal plane. Type C goniometry (b): The main orientation is downwards, which is also the origin of $\theta$. The origin of $\psi$ depends on the specific luminaire.

plane. Angle ranges and increments are variable and depend on the luminaire. An illustration is shown in Figure 2.4a.

- **Type C Goniometry**

  Is usually used for indoor lights. The origin of $\theta$ is downwards and $\psi$ is orientated along an axis of the luminaire. The ranges of the angle $\theta$ depends on the luminaire. Whether it emits light downwards ($0° \leq \theta \leq 90°$), upwards ($90° \leq \theta \leq 180°$) or both ($0° \leq \theta \leq 180°$). The step size is by default 5° or 10°, but it can be more granular. If the luminous intensity is rotationally symmetric, all data is given for $\psi = 0°$. Otherwise, depending on the symmetry, various ranges of $\psi$ can be listed. In latter case, the usual step size of $\psi$ is 22.5°. An illustration is provided in Figure 2.4b.

The measurements of the luminous intensity distribution are taken with a *Goniophotometer*, a combination of a photometer and a goniometer. Luminous intensity is measured from different positions around a light source. The measurement distance is large relative to the measured light source, such that the luminaire can be seen as a point light source and its geometry can be neglected. As a result, the goniophotometric measurements are always considered for point lights. Besides providing the raw values, photometric profiles also present the data in a polar plot. An example is shown in Figure 2.5. [DHMS11]



| | C0 | C90 | C180 | C270 |
|---|---|---|---|---|
| 0° | 140 | 140 | 140 | 140 |
| 5° | 145 | 142 | 145 | 137 |
| 10° | 158 | 144 | 158 | 134 |
| 15° | 165 | 147 | 165 | 131 |
| 20° | 158 | 152 | 158 | 126 |
| 25° | 140 | 162 | 140 | 120 |
| 30° | 116 | 178 | 116 | 104 |
| 35° | 61 | 193 | 61 | 54 |
| 40° | 18 | 195 | 18 | 10 |
| 45° | 12 | 176 | 12 | 2 |
| 50° | 9 | 142 | 9 | 1 |
| 55° | 6 | 104 | 6 | 1 |
| 60° | 5 | 56 | 5 | 1 |
| 65° | 3 | 11 | 3 | 0 |
| 70° | 2 | 7 | 2 | 0 |
| 75° | 1 | 4 | 1 | 0 |
| 80° | 1 | 2 | 1 | 0 |
| 85° | 0 | 1 | 0 | 0 |
| 90° | 0 | 0 | 0 | 0 |
| 95° | 6 | 7 | 6 | 7 |
| 100° | 16 | 18 | 16 | 18 |
| 105° | 35 | 32 | 35 | 32 |
| 110° | 66 | 46 | 66 | 46 |
| 115° | 115 | 60 | 115 | 60 |
| 120° | 177 | 74 | 177 | 74 |
| 125° | 225 | 87 | 225 | 87 |
| 130° | 252 | 100 | 252 | 100 |
| 135° | 262 | 112 | 262 | 112 |
| 140° | 263 | 123 | 263 | 123 |
| 145° | 249 | 133 | 249 | 133 |
| 150° | 228 | 143 | 228 | 143 |
| 155° | 209 | 151 | 209 | 151 |
| 160° | 193 | 159 | 193 | 159 |
| 165° | 182 | 164 | 182 | 164 |
| 170° | 176 | 169 | 176 | 169 |
| 175° | 173 | 171 | 173 | 171 |
| 180° | 172 | 172 | 172 | 172 |

Figure 2.5: Luminous intensity distribution as shown in a photometric report. The left side shows the luminous intensity distribution in a polar plot. The disk encodes the altitude angle $\theta$ and the colored lines encode ranges of the azimuth angle $\psi$. The exact candela measurements for the various directions are listed in the table to the right. Photometry of LINETIK (see Section 6.3). Image taken from [Zum18].

**Average Luminance** is sometimes provided for indoor luminaires. It gives a general idea of the light's characteristics, but is only useful for rotationally symmetric lights. Moreover, if the luminous intensity of a light varies strongly, the average luminance gives a false impression.

**Zonal Lumens** show the amount of lumens per zone, which is the result of discretizing and dividing of the solid angle around a luminaire.

**Efficiency** is the ratio of the lumens emitted by the luminaire to the lumens emitted by the lamp in the luminaire. It provides a measure for how effectively the system of the luminaire works (how the reflectors and the geometry influence the performance of the lamp inside the luminaire).

**Other Components** Some reports provide *Coefficients of Utilization*, which describe how effectively lumens are distributed on a horizontal work plane. A *Spacing Criterion* states the distance between luminaires to provide a uniform horizontal illuminance. There are some more specialized components like *Glare Assessment* and *Illumination Pattern*. A complete and more detailed list is provided by [DHMS11].

## 2.3 Photometric Data for Rendering

To render a scene illuminated by a photometric light source, the luminous intensity distribution is required. It is used to retrieve the luminous intensity of a light source when the brightness of a surface point is calculated. The luminous intensity distribution measurements are categorized into two classes [Ash95]: *Near-field photometry* and *far-field photometry*.

Far-field photometry, which is described in Section 2.2, is the current industry standard. A luminaire is measured from a distance large enough such that it can be seen as a point light source. When using this data for computations, exact results are only given when the distance of a surface in a computation to the light source matches the measuring distance. It was observed, however, that when the distance to the light source is five times larger than the largest dimension of the luminaire, the error in the calculated illumination can be neglected. This is referred to as the *Five-Times Rule*. If the distance is lower, the computation is most likely inaccurate. [DHMS11]

Currently, there are two standards for storing far-field photometric profiles [Zal12]. The first standard, *IESNA LM-63 (IES)*, was defined and is maintained by the Illuminating Engineering Society of North America (IESNA). The second one, *EULUMDAT (LDT)*, is the European standard format. It has no official documentation and was not updated since its introduction in 1990. Both standards store the photometric profiles in *ASCII* format.

Both standards provide in general the same information, although there are some minor differences regarding the provided data. Both start with meta information about the luminaire: Name, product number, the manufacturer, type of the luminaire and so on. Then, some technical details are provided like the type and number of lamps used in the luminaire. More importantly, the dimensions (length, width, height) of the luminous area [1] can be provided. The luminous area is the part of the luminaire where light is exiting. It can either be rectangular or circular. Finally, both standards contain the luminous intensity distribution of the luminaire.

Near-field photometry provides data that allows accurate computations for short distances. One solution to derive it is to measure the light at distances that occur in the computation. Near-field photometry data, which is generated with this process, often provides data for several distances. [DHMS11]

Another approach is *Luminance-Field photometry* introduced by Ashdown [Ash93]. This solutions measures the luminance distribution around a luminaire at fixed points that sit on a convex hull around the luminaire. The luminance is measured with a camera, which takes images of the luminaire at the measurement positions. The resulting images describe a field of light that describes the luminance for every point in space [Ash95]. While the measurements are taken at a constant distance, the resulting data can be used for any distance as the field of light accurately describes the leaving flux of the luminaire

[1] Also referred to as luminous opening [Lag].

for any direction. The cost of this solution, however, is the size and complexity of the resulting data.

## 2.4 Luminaires in Lighting Design Software

Modern lighting design software like *Hilite* [web18b], *Relux* [web18d] and *DIALux* [Gmb18] are a combination of CAD software and a toolset to render and evaluate lighting. Using such software, a lighting designer can import CAD plans of buildings (or create them inside the software) and place light sources in the scene. Then, he can evaluate the quality of the lighting and whether certain criteria are met (e.g., if an area is uniformly lit).

There are several possibilities to use a measured luminaire in lighting design software. All of the prior mentioned products can import IES and LDT files. The stored lighting profiles are used to generate a photometric light.

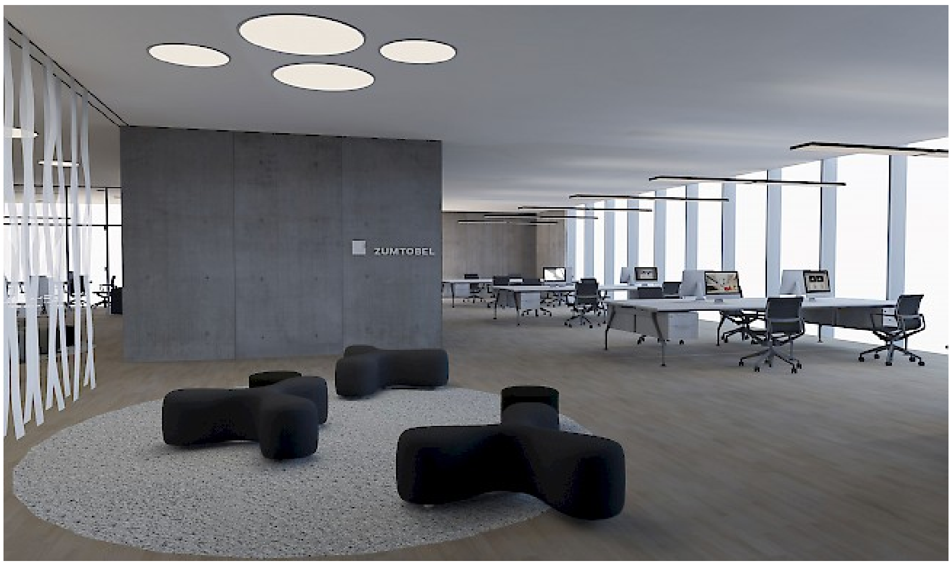


Figure 2.6: Scene with photometric lights rendered in Hilite. Image taken from [web18b]

Alternatively, Hilite, Relux and DIALux each support custom file formats that provide additional information about the luminaire. These custom file formats store, inter alia, a CAD model of the luminaire. As a result, the imported photometric lights look like their real world counterparts. The CAD model, however, has no influence on how the light is evaluated. It is only used to represent the photometric light.

The evaluation during the rendering process is done with an auxiliary geometry, which is generated based on the luminous area. This auxiliary geometry is referred to as *Light Emitting Object* (LEO). It is not rendered, but used to define the area/volume from which light is emitted.

Each of these software products uses a different rendering technique to create realistic, high-quality images to show the designed lighting system. DIALux uses photon shooting [Wit], Relux uses radiance [web18c] and Hilite uses a many-light global-illumination solution [LTH+13], where virtual polygon lights describe the illumination in a scene. An example for a scene rendered in Hilite with a complex lighting setup is shown in Figure 2.6.

Because creating high-quality renderings takes time, Hilite, Dialux and Relux also support interactive previews to provide a faster workflow for the lighting designer during the planning process:

Relux renders preview images of the scene if the camera was not moved for some time. The photometric light sources are approximated as a collection of photometric point lights that are distributed inside the LEO. This approach creates accurate preview images, but does not work in real time. An example of the approximation is shown in Figure 2.7.

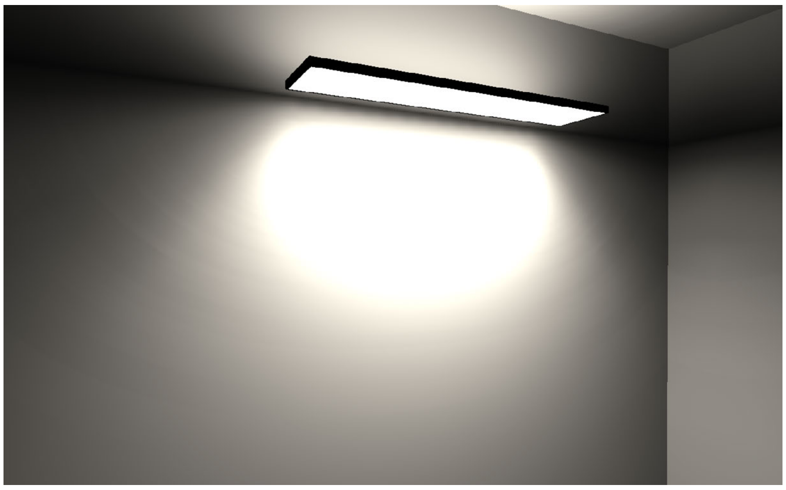


Figure 2.7: Relux approximates photometric light sources by using a collection of point lights. While this enables accurate previews, the interaction is not real time and requires the user to wait for a short time after something in the scene has changed.

Dialux supports real-time previews for a single photometric light. It is approximated with a single photometric point light in the center of the LEO. Hilite uses a combination of these techniques, where depending on the distance, one or more point lights are used. While this technique is fast, the violation of the *Five-Times Rule* becomes apparent for lights with larger LEOs. A light approximated with this technique in DIALux is shown in Figure 1.2 in the introduction.

## 2.5 Monte Carlo Integration

*Monte Carlo* integration is a numerical method used to solve integrals which are complex and cannot be solved analytically. It provides the mathematical foundation for sampling-based rendering techniques as it describes how integrals, e.g., the rendering equation, can be evaluated accurately with sampling.

As some of the techniques discussed in this thesis and our reference solution utilize sampling to evaluate direct lighting as presented in Section 2.6, we provide a concise introduction to Monte Carlo integration with a short overview of the used statistic tools in this section. If not otherwise stated, the following explanations are based on [KW08].

### 2.5.1 Estimator

$X$ is a continuous random variable which takes on selection a possible value $x \in \mathbb{R}$. The probability $P\{X \leq x\}$ that a random selection of $X$ gives a value less than $x$ is given by the *Cumulative Distribution Function* (CDF) $F(x)$, which is a nondecreasing function of its argument.

Assuming that $F(x)$ is differentiable, the *Probability Density Function* (PDF) is given by

$$f(x) \equiv \frac{dF(x)}{dx} \geq 0. \tag{2.9}$$

The PDF is always positive and has a normalization property:

$$\int_{-\infty}^{\infty} f(x)dx = 1. \tag{2.10}$$

Using a PDF, the probability $P\{x_1 \leq X \leq x_2\}$ with $x_1 < x_2$, can be computed:

$$P\{x_1 \leq X \leq x_2\} = \int_{x_1}^{x_2} f(x)dx = F(x_2) - F(x_1). \tag{2.11}$$

The PDF of a random variable can be used to derive some properties of the random variable, such as the *Expected Value* and the *Variance*.

The expected value $E[X]$ is the stochastic mean of a random variable $X$ with a PDF $f(x)$. Given is a random variable $Y = g(X)$, which is defined as a function on $X$. The expected value $E(Y)$ is defined by

$$E[Y] = E[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)dx. \tag{2.12}$$

Using the expected value, the variance $V[Y]$ can be defined:

$$V[Y] = E[(Y - E[Y])^2] = E[Y^2] - E[Y]^2. \tag{2.13}$$

The following rules apply for the variance and the expected value:

$$E[aY] = aE[Y], \tag{2.14}$$

$$V[aY] = a^2 V[Y], \tag{2.15}$$

$$E\left[\sum_{i=1}^{N} Y_i\right] = \sum_{i=1}^{N} E[Y_i]. \tag{2.16}$$

For a set of independent random variables $Y_i$, the following rule also applies for the variance:

$$V\left[\sum_{i=1}^{N} Y_i\right] = \sum_{i=1}^{N} V[Y_i]. \tag{2.17}$$

Finally, the *Estimator* is defined. A function $G_N$ of a set of $N$ random variables $X_1, X_2, \ldots, X_N$ is an estimator of a quantity $Q$ (e.g., the result of an integral), if the mean $E[G_N]$ is a useful approximation for that quantity. An estimator is random. A concrete value derived from $G_N$ is called an *Estimate*. The quality of an estimator depends on several criteria:

**Bias** An estimator $G_N$ is unbiased if Equation 2.18 is valid. An unbiased estimator guarantees that the expected error will be 0. If an estimator is biased, it is not excluded that the expected error will also be 0, but it is also not guaranteed.

$$E[G_N] = Q \text{ for all } N > 0 \tag{2.18}$$

**Consistency** An estimator $G_N$ is consistent, if the estimator converges to $Q$ with a probability 1 as $N \to \infty$, as shown in Equation 2.19. It does not state when exactly the estimator converges to the right solution. An unbiased estimator can be inconsistent and a biased estimator can be consistent. A biased estimator is consistent as the bias converges to 0 with more samples.

$$P\{\lim_{N\to\infty} G_N = Q\} = 1 \tag{2.19}$$

**Mean Squared Error** Another criteria for an estimator is that it minimizes the mean squared error shown in Equation 2.20. The first term of the decompositioned expected value is the variance of the estimator, meaning that a small variance is a good quality measure of an estimator since it minimizes the error. The second term is minimized when the estimator is unbiased.

$$MSE = E[(G_N - Q)^2] = E[(G_N - E[G_N])^2] + (E[G_N] - Q)^2 \tag{2.20}$$

### 2.5.2 Monte Carlo Estimator

The basic idea of Monte Carlo integration is to approximate an integral with an estimator. Given is the integral $I$ shown in Equation 2.21, where $\Omega$ is a subset of $R^m$, and set of $N$ independent, identically distributed ($i.i.d.$) random variables $X_1, X_2, \ldots, X_N \in \Omega$ drawn from a uniform probability function:

$$I = E[g(x)] = \int_\Omega g(x)dx. \tag{2.21}$$

A naive Monte Carlo estimator for that integral is given by

$$I \approx G_N = \Phi \frac{1}{N} \sum_{i=1}^{N} g(X_i) \tag{2.22}$$

where $\Phi$ is the volume of $\Omega$:

$$\Phi = \int_\Omega dx. \tag{2.23}$$

The estimator is consistent and unbiased. Since the estimator is unbiased, the behavior of the expected error depends solely on the variance of the estimator. The variance is shown in Equation 2.24. It can be seen that the variance decreases as the number of samples increases. Hence, the expected error becomes smaller as the number of samples grows. The estimator converges towards the solution.

$$V[G_N] = V\left[\Phi \frac{1}{N} \sum_{i=1}^{N} g(X_i)\right] = \Phi^2 \frac{1}{N} V[g(X)] \tag{2.24}$$

### 2.5.3 Importance Sampling

While increasing the number of samples decreases the variance, the magnitude of the variance cannot be estimated generally but depends on the function $g(x)$ in Equation 2.21. This means that, depending on the function $g(x)$, the Monte Carlo estimator could have a larger or smaller variance, which again has influence on the convergence speed. An example for a function with a large variance would be a function that is zero in many regions.

Generally, a uniform PDF as assumed in the prior section is not guaranteed to be the best PDF for the estimator. Another PDF $\tilde{f}(x)$ can be used to draw random samples, which may results in a smaller variance and therefore in a faster shrinkage of the error. The PDF can be added to the integral:

$$I = E\left[\frac{g(x)}{\tilde{f}(x)}\right] = \int_\Omega \frac{g(x)}{\tilde{f}(x)}\tilde{f}(x)dx. \tag{2.25}$$

The Monte Carlo estimator has to be adopted as shown in Equation 2.26, where $X_1, X_2, \ldots, X_N$ are drawn from $\tilde{f}(x)$. This is the general form of the Monte Carlo estimator. The form shown in Equation 2.22 is a special form, which is derived by using a uniform PDF.

$$G_N = \frac{1}{N}\sum_{i=1}^{N}\frac{g(X_i)}{\tilde{f}(X_i)} \tag{2.26}$$

When $\tilde{f}(x)$ is chosen such that $\tilde{f}(x) \propto |g(x)|$, the variance of the estimator will be minimized. This can be easily shown for a function $g(x)$ where $g(x) > 0 \ \forall x \in \Omega$. Assuming that $g(x)$ is known (which is usually not the case when Monte Carlo integration is required), a PDF $\tilde{f}(x) \propto |g(x)|$ can be chosen such that $\alpha\tilde{f}(x) = g(x)$ where $\alpha$ is constant. This means that $g(x)/\tilde{f}(x) = \alpha$ as is the expected value $E[g(X)/\tilde{f}(X)]$. As a result, the variance is 0. (Proof taken from [And99]).

Because $g(x)$ is normally not known or integrateable, a PDF $\tilde{f}(x)$ is chosen that is approximately proportional to $g(x)$. This already reduces the variance. Using such a PDF for Monte Carlo integration is called importance sampling.

### 2.5.4 Random Variable Generation

An important issue of Monte Carlo integration is the generation of random variables [Nie92]. The issue arises with computers, which cannot generate true random variables and use deterministic algorithms instead. Random variables generated this way are referred to as *Pseudorandom Variables* (PRVs). The success of a Monte Carlo integration depends on the quality of such PRVs, where the quality is measured by how well the variables approximate the behavior of real random variables.

Current random sample generators can create high-quality uniform PRVs. The PDFs used for importance sampling, however, are often non-uniform. A common approach to generate non-uniform PRVs is to transform uniform PRVs.

One way to achieve such a transformation is the *inversion method* [Nie92], which is defined as follows [Dev86]: Given is a CDF $F$, with an inverse $F^{-1}(u) = \inf\{x : F(x) = u, 0 < u < 1\}$. If $U$ is a uniform random variable, then $F^{-1}(U)$ has the CDF $F$. This means that if $F^{-1}(u)$ is known, random variables with the CDF $F$ can be generated using uniform random variables.

This method can also be used for two-dimensional random variables [SWZ96], e.g., directions. For a two-dimensional random variable $(X, Y)$, the CDF for each dimension is separately inverted and used. The PDF $p(x, y) : [x_1, x_2] \times [y_1, y_2] \to \mathbb{R}^+$ of $(X, Y)$ first has to be integrated over the $y$ domain to receive the marginal density function $p_X(x)$:

$$p_X(x) = \int_{y_1}^{y_2} p(x, y')dy'. \tag{2.27}$$

Then, given a fixed value $x$ for $X$, the PDF for $Y$, $p_{Y|X}(y|x)$ can be computed:

$$p_{Y|X}(y|x) = \frac{p(x, y)}{p_X(x)}. \tag{2.28}$$

The CDFs for both PDFs are then derived by using Equation 2.9. Then, the inversion method can be applied. More information regarding the inversion method and actual inversion techniques can be found in [Dev86].

A simple example for the application of the inversion method in the context of rendering is importance sampling (see Section 2.5.3) the cosine falloff, which occurs in the rendering equation (see Section 2.6), by transforming uniform random samples such that they have the *cosine distribution*.

Given is a integral of a function $f(\omega)$ defined over the unit hemisphere multiplied with the cosine of the polar angle $\theta$ (i.e., cosine falloff):

$$\int_\Omega f(\omega) \cos\theta \; d\omega. \tag{2.29}$$

This function integral can be importance sampled by using samples which have the cosine distribution

$$p(\hat{\omega}_i) = p(\theta, \phi) = \frac{\cos\theta \sin\theta}{\pi} > 0 \; \forall \theta \in [0, \frac{\pi}{2}]. \tag{2.30}$$

The term $\sin\theta$ occurs because the solid angle $d\omega$ is expressed in spherical coordinates, which is given by $d\omega = sin\theta \; d\theta \; d\phi$:

$$\int_\Omega g(\omega) \cos\theta \; d\omega = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} g(\theta, \phi) \cos\theta \sin\theta \; d\theta \; d\phi. \tag{2.31}$$

The division by $\pi$ is required such that the PDF satisfies the normalization property:

$$p(\theta, \phi) = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \frac{\cos\theta \sin\theta}{\pi} \; d\theta \; d\phi = 1. \tag{2.32}$$

Instead of drawing samples from the cosine distribution directly, the inversion method is used to transform uniformly drawn samples. The first step is to compute the individual PDFs for the angles. The marginal distribution for $\theta$ is computed by integrating the PDF $p(\theta, \phi)$ over the domain of $\phi$

$$p_\Theta(\theta) = \int_0^{2\pi} \frac{\cos\theta \sin\theta}{\pi} \, d\phi = \sin(2\theta). \tag{2.33}$$

Using $p_\Theta(\theta)$, the probability $p_{\Phi|\Theta}(\phi|\theta)$ can be calculated:

$$p_{\Phi|\Theta}(\phi|\theta) = \frac{p(\theta,\phi)}{p_\Theta(\theta)} = \frac{\frac{\cos\theta\sin\theta}{\pi}}{\sin(2\theta)} = \frac{\sin\theta\cos\theta\csc(2\theta)}{\pi}. \tag{2.34}$$

Given the corresponding PDFs, the CDF for $\theta$ and the CDF for $\phi$ can be derived:

$$P_\Theta(\theta) = \int_0^\theta p_\Theta(\theta') \, d\theta' = \int_0^\theta \sin(2\theta') \, d\theta' = \sin^2\theta, \tag{2.35}$$

$$P_{\Phi|\Theta}(\phi|\theta) = \int_0^\phi p_{\Phi|\Theta}(\phi'|\theta) \, d\phi' = \int_0^\phi \frac{\sin\theta\cos\theta\csc(2\theta)}{\pi} \, d\phi' = \frac{\phi}{2\pi}. \tag{2.36}$$

These CDFs have to be reversed. One approach is to pick two variables $\xi_1$ and $\xi_2$, which represent uniform random variables, and inverse the CDF through rearrangement:

$$P_\Theta(\xi_1)^{-1} : \ \xi_1 = \sin\theta \to \theta = \arcsin\sqrt{\xi_1}, \tag{2.37}$$

$$P_{\Phi|\Theta}(\xi_2)^{-1} : \ \xi_2 = \frac{\phi}{2\pi} \to \phi = 2\pi\xi_2. \tag{2.38}$$

After transforming the inverse CDFs to the Cartesian space, random samples with the cosine distribution can be generated by transforming two uniform random variables $\xi_1$ and $\xi_2$:

$$\begin{aligned} x &= \sqrt{\xi_1}\cos(2\pi\xi_2), \\ y &= \sqrt{\xi_1}\sin(2\pi\xi_2), \\ z &= \sqrt{1-\xi_1}. \end{aligned} \tag{2.39}$$

### 2.5.5 Low-Discrepancy Sampling

Discrepancy is a measure for how uniformly distributed a sequence is. A high discrepancy means that a sequence has several clusters and empty spaces. A discrepancy of 0 means that a sequence is perfectly equidistributed, which also means that there is no randomness. Samples drawn from a uniform PDF usually have a high discrepancy. An example is shown in Figure 2.8.

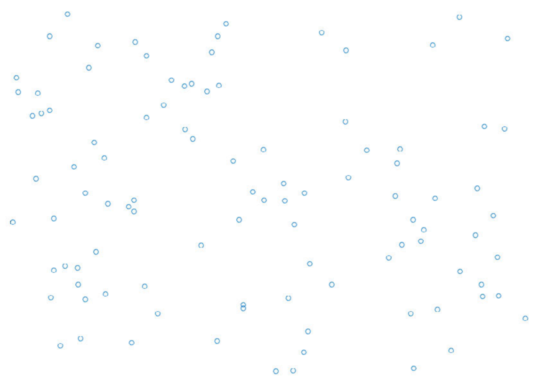


Figure 2.8: Random samples drawn from a uniform PDF. There are several clusters and empty spaces, which is a result of the high discrepancy.

Instead of creating uniformly distributed random variables, low-discrepancy methods are concerned with creating sequences and variable sets with a uniform distribution over the sample domain. Such samples are not random, but *quasi-random*, and integrating with such samples is referred to as *Quasi-Monte Carlo integration* [Nie92].

Niederreiter [Nie92] demonstrates that using samples with a low discrepancy reduces the error of Monte Carlo integration. One often used low-discrepancy sequence is the *Halton Sequence* [Hal64]. Given a set of co-prime bases $b_1, \ldots, b_s$, the Halton sequence $x_1, x_2, \ldots$ is defined as

$$x_n = (\phi_{b_1}(n),\ \ldots\ , \phi_{b_s}(n)) \in [0,1]^s \quad \forall n > 0 \tag{2.40}$$

where $\phi_b$ is the radical-inverse function:

$$\phi_b(n) = \sum_{j=0}^{\infty} a_j(n) b^{-j-1} \quad \forall n \geq 0. \tag{2.41}$$

Here, $a_j$ is in $Z_b = \{0, 1, \ldots, b-1\}$ for $j \geq 0$ and $a_j = 0$ if $j > b-1$. The input $n$ is given by its digit expansion

$$n = \sum_{j=0}^{\infty} a_j(n) b^j. \tag{2.42}$$

Figure 2.9 shows 100 samples generated from a Halton sequence generated from base 2 and 3. The samples are more uniformly distributed over the domain than the random samples in Figure 2.8.

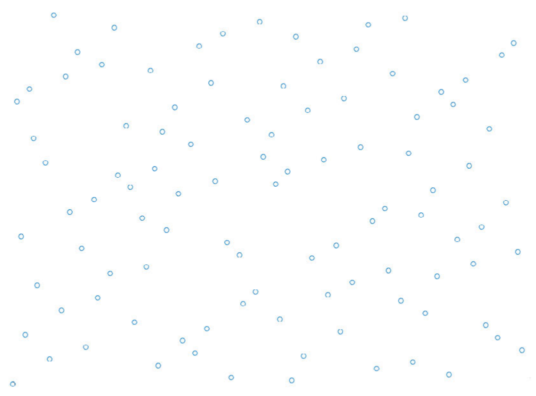


Figure 2.9: 2D Halton sequence for the bases 2 and 3. The samples are relatively uniformly distributed over the domain and appear random, although they are deterministic.

## 2.6 Direct Lighting

The focus of this thesis is the direct light transport, which is described by direct lighting. This section presents the mathematical formulation for direct lighting, which describes the light a surface reflects directly to the camera that it received directly from light sources, i.e., there is exactly one light reflection between a light source and the camera [SWZ96]. The section starts with explaining the rendering equation, which is the fundamental framework for rendering, and then goes on to derive the direct lighting formulation as demonstrated by Shirley, Wang and Zimmerman [SWZ96]. This section concludes by showing how direct lighting can be computed with Monte Carlo integration (see Section 2.5) and how importance sampling can be utilized in that context.

The rendering equation was first presented by Kajiya [Kaj86] as a base framework for global-illumination algorithms:

$$L_s(x, \hat{\omega}_o) = L_e(x, \hat{\omega}_o) + \int_{\Omega} f_s(x, \hat{\omega}_o, \hat{\omega}_i) L_s(x_{\hat{\omega}_i}, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x) d\hat{\omega}_i. \tag{2.43}$$

It describes the shading of every surface point in a scene. The outgoing radiance $L_s$ from a point $x$ in direction $\hat{\omega}_o$ is given by a sum of two terms: An emitting term $L_e(x, \hat{\omega}_o)$, which describes the emitted radiance of the surface point $x$ in direction $\hat{\omega}_o$ (e.g., $x$ is on a light source), and reflecting term, which is given by the integral. It integrates the incoming, reflected radiance of all directions $\hat{\omega}_i$ over the unit sphere $\Omega$ centered at $x$. It is the union of two hemispheres: An upper hemisphere $\Omega_+$ aligned to the normal of $x$ $\hat{n}_x$, and a lower hemisphere $\Omega_-$ aligned to $-\hat{n}_x$. The hemispheres have the relation $-\Omega_+ = \Omega_-$. All directions $\hat{\omega}_\bullet$ occurring in the equation are, if not otherwise stated, given as infinitesimal solid angles. The corresponding illustration is shown in Figure 2.10.

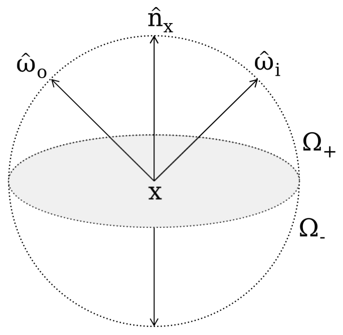


Figure 2.10: Illustration of the vectors used in the rendering equation.

The term $f_s(x, \hat{\omega}_o, \hat{\omega}_i)$ is the *Bidirectional Scattering Distribution Function* (BSDF), which describes how light from direction $\hat{\omega}_i \in \Omega$ is scattered at a surface point $x$ into the direction $\hat{\omega}_o \in \Omega$. It is the union of two functions, where one handles the reflectance and the other one handles the transmittance.

The *Bidirectional Reflectance Distribution Function* (BRDF) $f_r$ describes how light is reflected at surface point. It is derived by restricting $\hat{\omega}_i$ and $\hat{\omega}_o$ to the upper hemisphere $\Omega_+$. It has some important properties. The first one is the Hemlholtz reciprocity shown in Equation 2.44, which states that the function is symmetric. The second constraint is energy conservation, meaning that not more energy can leave a surface point than energy comes in. [Vea97]

$$f_r(x, \hat{\omega}_o, \hat{\omega}_i) = f_r(x, \hat{\omega}_i, \hat{\omega}_o) \; \forall \; \hat{\omega}_i, \hat{\omega}_o \tag{2.44}$$

The *Bidirectional Transmittance Distribution Function* (BTDF) $f_t$ describes how light at the point $x$ is transmitted. It is derived by restricting $\hat{\omega}_i$ to $\Omega_A$ and $\hat{\omega}_o$ to $\Omega_B$, where $\Omega_A = -\Omega_B$ and $\Omega_A = \Omega_+$ or $\Omega_A = \Omega_-$. The properties of the BRDF do not always apply for the BTDF. [Vea97]

$L_s(x_{\hat{\omega}_i}, -\hat{\omega}_i)$ describes the radiance emitted from the point $x_{\hat{\omega}_i}$ into the direction $-\hat{\omega}_i$. $x_{\hat{\omega}_i}$ is another surface point in the environment that is derived by casting a ray starting at $x$ into the direction $\hat{\omega}_i$. The function $L_s$ is solved by the rendering equation, which means the rendering equation is a recursive function and the contained integral is infinitely dimensional.

An alternative form to write the rendering equation, which was also first used by Kajiya, is to define the integral over all surfaces in the scene:

$$L_s(x, \hat{\omega}_o) = L_e(x, \hat{\omega}_o) + \int_{x' \in S} g(x, x') f_r(x, \hat{\omega}_o, \hat{\omega}_i) L_s(x', -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x) \frac{(-\hat{\omega}_i \cdot \hat{n}_{x'})}{\|x' - x\|^2} dA(x'). \tag{2.45}$$

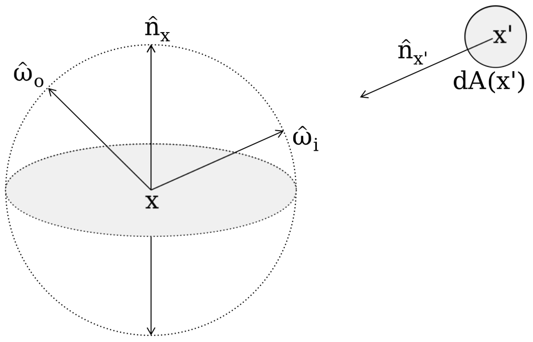


Figure 2.11: Illustration of the vectors used in the rendering equation defined over the surfaces of the scenes.

The integral domain changes to $S$, the collection all surface points in the scene where a single scene surface point is denoted as $x'$ with a normal $\hat{n}_{x'}$ and an infinitesimal area $A(x')$. The term $g(x, x')$ describes the visibility between $x$ and $x'$, which is 1 if there is no occluder between the two points and 0 otherwise. Additionally, the weight $(-\hat{\omega}_i \cdot \hat{n}_{x'})/\|x' - x\|^2$ is added which is the subtended solid angle of $x'$. The corresponding illustration is shown in Figure 2.11.

Direct lighting $L_d$ can be formulated based on the rendering equation defined over directions (Equation 2.43) by discarding the emitting term $L_e(x, \hat{\omega}_o)$ and ignoring the reflected radiance at $x_{\hat{\omega}_i}$ [SWZ96]:

$$L_d(x, \hat{\omega}_o) = \int_\Omega f_s(x, \hat{\omega}_o, \hat{\omega}_i) L_e(x_{\hat{\omega}_i}, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x) d\hat{\omega}_i. \tag{2.46}$$

This integral can be computed with Monte Carlo integration by using the estimator given in Equation 2.47. [SWZ96]

$$L_{d,N}(x, \hat{\omega}_o) = \frac{1}{N} \sum_{i=1}^{N} \frac{f_s(x, \hat{\omega}_o, \hat{\omega}_i) L_e(x_{\hat{\omega}_i}, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x)}{p(\hat{\omega}_i)}. \tag{2.47}$$

Direct lighting can be derived equivalently from the rendering equation defined over surfaces [SWZ96]:

$$L_d(x, \hat{\omega}_o) = \int_{x' \in S} g(x, x') f_r(x, \hat{\omega}_o, \hat{\omega}_i) L_e(x, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x) \frac{(-\hat{\omega}_i \cdot \hat{n}_{x'})}{\|x' - x\|^2} dA(x'). \tag{2.48}$$

The corresponding estimator is given by Equation 2.49. [SWZ96]

$$L_{d,N}(x, \hat{\omega}_o) = \frac{1}{N} \sum_{i=1}^{N} g(x, x_i) f_r(x, \hat{\omega}_o, \hat{\omega}_i) L_e(x_i, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x) \frac{(-\hat{\omega}_i \cdot \hat{n}_{x_i})}{p(x_i)\|x_i - x\|^2} \tag{2.49}$$

Given these estimators, the direct lighting for a surface point in a scene can be computed. The PDF $p$, which governs how the samples are created, is important for the convergence speed of the estimators (see Section 2.5.3). It is reasonable to use a PDF which creates samples with a higher probability of hitting light sources when computing direct lighting, as these are the only samples which return a non-zero value for $L_e$. Otherwise, many samples return zero for $L_e$, which results in a high variance and a slow convergence speed. Using a PDF that favors samples on the light source, or on the subtended solid angle of the light source, is called *light-source importance sampling.*

The perfect PDF would be

$$p(\hat{\omega}_i) = C f_s(x, \hat{\omega}_o, \hat{\omega}_i) L_e(x_{\hat{\omega}_i}, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x) \tag{2.50}$$

or

$$p(x') = C g(x, x') f_r(x, \hat{\omega}_o, \hat{\omega}_i) L_e(x', -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x) \frac{(-\hat{\omega}_i \cdot \hat{n}_{x'})}{\|x' - x\|^2} \tag{2.51}$$

where $C$ is a normalization constant, as it is zero everywhere except on the light source. However, the value of the integral must be known to compute $C$, which means that using such a PDF is not feasible [SWZ96]. Instead, it is more reasonable to use a PDF which is proportional to only some terms in the integral.

To derive a fitting PDF, Shirley, Wang and Zimmerman [SWZ96] propose to classify the terms in the integrand either as *low variation* (LO), *high variation* (HI) or *unknown* (UN), depending on how much the term varies in the integration domain, and try to find a PDF which is proportional to the high-variation terms. They classify the terms in Equation 2.48 as follows, if no information regarding the light source is given:

$$\underbrace{g(x, x')}_{UN} \underbrace{f_r(x, \hat{\omega}_o, \hat{\omega}_i) L_e(x', -\hat{\omega}_i)}_{LO} \underbrace{(\hat{\omega}_i \cdot \hat{n}_x) \frac{(-\hat{\omega}_i \cdot \hat{n}_{x'})}{\|x' - x\|^2}}_{HI}. \tag{2.52}$$

This is the general classification, which can vary depending on the scene and the light source. Depending on the type of the light source (spherical, planar, disc), its dimension and location, terms defined as highly varying can be low-variation terms. For example, if the light is far away, the term $\|x' - x\|^2$ will not vary much. Or if the light source is planar, the term $(-\hat{\omega}_i \cdot \hat{n}_{x'})$ will not change much. For some lights, e.g., small planar lights, all terms except the geometry term may be classified LO.[SWZ96] If all terms are classified as low-variation terms, Shirley et al. [SWZ96] suggest to use a constant PDF. Although a PDF proportional to the low-variation terms can be designed, they deem the computational cost of evaluating such a PDF not worth the slightly lower variance. Moreover, the geometry term could have a high variance, making every improvement by the low-variation PDF irrelevant.

Given a light source and a classification of the terms in the integral, a PDF proportional to the high-variation terms can be designed. Wang [Wan94] demonstrates this for various light sources (e.g., spherical, rectangular, polygonal, etc.). Samples can then be created for the used PDF by transforming uniform samples with the inversion method (see Section 2.5.4), assuming that it is possible for the light source. If it is not possible, an approximation might be developed. For a more detailed explanation see Wang [Wan94] and Shirley et al. [SWZ96].

For planar polygonal lights, Shirley et al. [SWZ96] suggest to create the samples on the light source and use the constant PDF $p(x') = 1/S_L$, where $S_L$ is the area of the light source $L$. They generate the samples on the bounding rectangle and reject them if they are not inside the light's polygon. While a PDF

$$p(x') \propto (\hat{\omega}_i \cdot \hat{n}_x) \frac{(-\hat{\omega}_i \cdot \hat{n}_{x'})}{\|x' - x\|^2} \tag{2.53}$$

can be used for polygonal lights (see Wang [Wan94]), they deem the evaluation too complex and expensive.

An improvement to $p(x') = 1/S_L$ would be to use Equation 2.47 with

$$p(\omega_i) = \frac{1}{\Omega_L} \propto \frac{(-\hat{\omega}_i \cdot \hat{n}_{x'})}{\|x' - x\|^2}, \tag{2.54}$$

as areas on the light with a larger subtended solid angle are more likely to be sampled, which noticeably reduces the variance if the light source is close. [Wan94]

This can be done efficiently by uniformly sampling the subtended solid angle of a light source. Arvo [Arv95b] presents a mapping that allows generating uniformly distributed random samples on a spherical triangle $T$. The mapping function $M : [0, 1] \times [0, 1] \to T$ takes two uniformly distributed random variables as input and outputs a uniformly distributed random sample inside of a spherical triangle. Ureña, Fajardo and King [UnFK13] extended this method such that it can also be used for spherical rectangles.

To create the reference solution for this thesis, we implemented the approach proposed by Shirley et al. [SWZ96] for our reference renderer (see Section 6.1), because it does not require the solid angle for the evaluation. Being independent of the solid angle is important for artifact-free approximations when evaluating photometric area lights, as the solid angle and the photometry may be conflicting. Some luminaires emit light inside the light plane where the solid angle is by definition zero, which means that there cannot be any illumination. Using a technique which utilizes the solid angle leads to divisions by very small numbers when evaluating the illumination for points near or inside the light plane, which results in visible numerical artifacts. Our reference renderer avoids this contradiction and creates reference images without numerical artifacts by using the approach proposed by Shirley et al. [SWZ96].

CHAPTER 3

# Related Work

This chapter starts with an overview on important publications regarding *Global Illumination* algorithms in Section 3.1. Global Illumination algorithms are concerned with solving the rendering equation (see Section 2.6) correctly. They handle photometric lights implicitly. Section 3.2 focuses specifically on research regarding the illumination by area lights. The presented techniques provide different approaches of which some are currently used in real-time applications. The final Section 3.3 presents work concerned with approximating lights with near-field characteristics of real-world luminaires.

## 3.1 Global Illumination Algorithms

In 1984, Goral et al. [GTGB84] introduced *Radiosity*, the first rendering technique that approximates global illumination in a scene consisting only of diffuse surfaces. The algorithm is based on a discretized representation of the scene, where each surface is subdivided into a set of patches. The light transport in the scene is then derived by computing the light transport between all patch pairs, which is given by the form-factor between them. The radiosity in a scene can be precomputed, since only the diffuse light transport is calculated. This, however, assumes that the scene is static. The accuracy of the method depends on the size of the patches. The smaller the patches, the more accurate the solution.

Another approach which can compute the light transport in scenes with diffuse, glossy and specular surfaces was introduced by Kajiya [Kaj86] in 1986: *Path Tracing.* Path tracing is based on *Monte Carlo integration*, where the integral of a function is computed by averaging over a set of samples taken from the function. This converges to the right solution as the number of samples grows. In the context of path tracing, Monte Carlo integration is implemented by shooting rays originating from the viewer into the scene. When a ray hits a surface point, it is reflected in a random direction and continues. This creates a path. The path ends as soon as a light source is hit. It is then propagated backwards, accumulating the energy depending on the materials of the path's surface points. Instead of reflecting the ray in a random direction, the new direction can also be derived by *Importance Sampling*, where rays are shot into directions with a higher contribution to the integral. Doing this, the algorithm converges faster towards the solution.

In 1993, Lafortune and Willems [LW93] improved the technique and developed *Bi-Directional Path Tracing*. Additional to the path originating from the viewer, a second path is traced starting from a light source. After both paths have reached a certain depth, all hit points from each path are connected, which results in many paths between the camera and the light source. This technique converges faster than simple path tracing.

*Metropolis Light Transport* was presented by Veach and Guibas [VG97] in 1997 as a further improvement on path tracing. Instead of generating new paths at random (or with importance sampling), they are derived by mutating existing paths slightly. The paths to mutate are selected by the magnitude of their illuminance contribution. This results in a stronger sampling of brighter paths, while darker paths with less contribution are left aside. The algorithm can handle complex lighting situations (e.g., light coming only through a small hole) better than prior mentioned techniques.

Jensen [Jen96] first described *Photon Maps* in 1996. Before the rendering process, photons are emitted from light sources and distributed throughout the scene based on path tracing. At each hit point, a photon is stored in a map that represents all surfaces of the scene: the photon map. After the photon map is generated, the scene is rendered with a ray tracer. The illuminance at a surface point is derived by estimating the photon density at that point. Photon mapping can handle diffuse, glossy and specular surfaces. Since many photons are required for highly glossy and specular surfaces, Jensen proposes to use Monte Carlo ray tracing for such surfaces additionally to photon mapping to reduce the memory consumption. Using this approach, complex lighting situation like caustics can be rendered efficiently. For caustics, a second photon map is used storing only indirect illumination.

Hachisuka, Ogaki and Jensen [HOJ08] further improved photon mapping in 2008 with *Progressive Photon Mapping*. Instead of computing a single, dense photon map only once, several, less dense photon maps are generated throughout the rendering process. The first pass is a ray tracing pass where rays are casted into the scene and reflected until a non-specular surface is hit. From the second pass onward, photon maps are generated, which are evaluated using the endpoints of the first pass as it is done with photon mapping. At each photon mapping pass, the photon map from the previous pass is discarded. The gather radius is reduced during the rendering process. Using this technique, the memory consumption of photon mapping is reduced significantly. Moreover, the quality of the resulting image is more controllable since the rendering process becomes iterative. The technique was further improved with *Stochastic Progressive Photon Mapping* by Hachisuka and Jensen [HJ09] in 2009, which allows to render effects like depth-of-field or motion blur using photon maps.

Georgiev et al. [GKDS12] introduced a novel approach in 2012 called *Vertex Connection and Merging*, where they combined bi-directional path tracing and photon mapping using *Multiple Importance Sampling* [VG95]. This is achieved by reformulating photon mapping as a path sampling technique, where various light paths starting at a light source are sampled throughout the scene. The resulting photons in the photon map represent the endpoints of these light-paths. As soon as the scene is rendered, a path is traced from the camera until a certain depth is reached. Following, photons within a certain merging radius of the camera-path's endpoint are searched. Each of these photons is merged with the camera-path endpoint, creating a direct path from the light to the camera, which is then used to evaluate the illuminance. This approach creates paths that would not exist otherwise, resulting in a bias of the algorithm. But by reducing the merging radius during the rendering process, the bias disappears. Using this technique, complex light paths can be evaluated efficiently.

Keller [Kel97] presents *Instant Radiosity*, a Quasi-Monte Carlo technique for global illumination, in 1997. Light sources emit particles, which are reflected when they collide with a surface in the scene. This creates a path, which is continued until a certain number of reflections occurred. The starting points as well as each hit point of the path are then used as *Virtual Point Lights* (VPLs). When the scene is rendered, it is illuminated with every VPL. Indirect light is handled implicitly. The quality of the result depends on the number VPLs.

Instant Radiosity was the initial publication of a new type of global-illumination rendering: *Many-Light Global Illumination* (MLGI). The research in this field is concerned with the generation of VPLs, rendering and scalability. One approach to generate VPLs more efficiently is presented by Georgiev and Slusallek [GS10]. Their approach is to discard VPLs that do not contribute much to the final image. This is achieved by generating candidate VPLs, where for each candidate the contribution to the image is estimated. The ratio of this contribution to a desired average distribution (derived e.g., by some initial VPLs) results in a probability that decides whether the light will be rejected or not. The algorithm converges regardless of the used average distribution. Another approach for efficient VPL generation is given by Segovia, Iehl and Péroche [SIP07], who present a technique based on Metropolis Light Transport: *Metropolis Instant Radiosity.* New VPLs are generated by slightly mutating the light paths of existing ones, as it is done in the Metropolis Light Transport technique.

The research regarding lighting in MLGI algorithms is mainly focused on handling singularity, which can occur as bright light spots when a surface point and a VPL are very close. The singularity is caused by the geometry term of the Instant Radiosity equation (see [Kel97]), where a division by the distance between the shaded point and the VPL happens. A common solution to handle this is to provide an upper bound for the geometry term. This, however, creates a bias. One approach to handle this bias is proposed by Kollig and Keller [KK06], who handle it by adding a compensation term. This term equals the Instant Radisotiy equation, but has a different geometry term that accounts for the lost energy. Another approach is to avoid singularities in the first place.

An example for such a technique is given by Hašan et al. [HKWB09], who introduce *Virtual Spherical Lights* (VSL). The usage of a VSL replaces the point to point evaluation between a shaded point and a VPL with an evaluation over the solid angle of a VSL. By doing this, singularities are avoided.

Various techniques have been developed to improve scalability such that a high number of VPLs can be handled efficiently. One such technique is given by Walter et al. [WFA+05], called *Lightcuts*, where the illumination of many point lights is approximated by clustering them using a *Light Tree*. Latter one is a binary tree, where the leaves are VPL and the interior nodes represent clusters. The final clustering during the rendering process is defined per surface point by a lightcut through the tree that is given by a set of nodes where every path from the root to a leave of the tree has to pass exactly one of these nodes. The lightcut depends on a user-defined threshold that defines an upper limit for an allowed error of a cluster. Another clustering approach is *Matrix Row-Column Sampling* presented by Hašan, Pellacini and Bala [HPB07]. The technique is based on a matrix representation of the scene, where the rows represent surface points and the columns represent VPLs. Each entry stores the contribution of a light to a surface point. This matrix is then subsampled by selecting a set of rows. The resulting subsampled matrix equals a matrix representing a smaller version of the scene. Clustering is then performed on the columns of the subsampled matrix, where each final clusters holds VPLs with similar contribution to the image. The final clustering is then used on the full scene matrix. For the rendering process, a representative column is used per cluster, which is scaled such that it accounts for the energy of the whole cluster.

## 3.2 Calculating Light Transport from Area Lights

An early approximation for photometric area lights in computer graphics is given by Verbeck and Greenberg [VG84]. They numerically integrate a photometric area light by using a collection of photometric point light sources. The point lights are evaluated with diffuse Phong shading, specular reflections are not support. Verbeck's and Greenberg's method is computationally expensive and not suitable for real-time applications because many point lights are required per photometric area light. If too few point lights are used, artifacts appear because the single point lights become noticeable.

Drobot [Dro14] does not use multiple point lights, but presents instead an importance sampling approach with a single sample to approximate the illumination from an area light. This single sample is taken from the *most representative point* (MRP) of an area light, which is the sample with the highest contribution to the illumination integral. The computation cost is independent of the complexity of the area light in both approaches.

The *Frostbite Engine* (Lagarde and De Rousiers [LR14]) provides two approaches to evaluate diffuse area lights in real time. One approach is the MRP solution of Drobot for Lambertian rectangular area lights. The other technique is *structured sampling*, where they use few, well chosen samples to compute the irradiance by a diffuse area light. Latter approach can lead to banding artifacts and depends, in contrast to the MRP approach, on the form of the area light. The *Unreal Engine* (Karis [KG13]) uses an MRP approach for sphere lights that also addresses energy conservation for glossy reflections.

Another method is given by Wang et al. [WLWF08]. They approximate area lights with pruned disk lights, which are generated by intersecting the approximated area lights with disk lights which are derived per shading point. The disk lights are positioned in a location of the approximated area light that contributes more to the illumination of the shaded point than the rest of the light. The radius is computed such that the illumination from the disk light matches the illumination from the area light. This method is efficient enough for real-time applications because the exitance of the disk lights is precomputed for various radii.

The first analytic solution to evaluate the illumination from a diffuse area light is given by Baum et al. [BRW89]. They present an analytic formula to compute the form-factor of a surface. It allows to determine the irradiance for a Lambertian (diffuse) polygonal light in $O(k)$, where $k$ is the number of vertices of the light. This solution returns accurate results and is nowadays usable in real-time applications.

Arvo [Arv95a] introduces *Irradiance Tensors*. Using them, he presents a closed-form expressions for area lights with a Phong-like emission (cosine with an exponent that controls the directionality). The expression can be evaluated in $O(n * k)$, where $k$ is the number of the edges of the polygon and $n$ is the directionality of the luminaire. Through further development by Chen and Arvo [CA00], Irradiance Tensors can be used to evaluate area lights with linearly-varying exitance with a complexity of $O(n^2 * k)$.

Lecocq et al. [LDSM17] further improve Arvo's solution. Approximating the complex integral of Arvo's method that is responsible for the dependency of the time complexity on the directionality with an accurate analytic function, they are able to reduce the computation time from $O(n * k)$ to $O(k)$. Moreover, they extend the technique such that the illumination of surfaces with microfacet Bidirectional Reflectance Distribution Functions (BRDFs) by such lights can be approximated.

Heitz et al. [HDHN16] introduce *Linearly Transformed Cosines* (LTC) which, using the form-factor computation introduced by Baum et al., allow to analytically evaluate an arbitrarily formed area light in $O(k)$ for surfaces with microfacet BRDFs. Before the illumination of a light is computed, it is linearly transformed according to a prior fitting process, where a light transformation is found that minimizes the illumination error for a given BRDF. This solution returns accurate results and is simpler to implement, but requires precomputation steps in contrast to the solution presented by Lecocq et al. [LDSM17].

## 3.3 Illumination by Complex Light Sources

The work presented in this section is concerned with complex light sources, which are light sources that have the near-field characteristics of real-world luminaires. The following techniques are not focusing on being usable for real-time applications, but much rather improving the computation time and memory consumption of offline global-illumination algorithms for which the reflective interior of luminaires can be problematic due to the high path depth.

Ashdown and Rykowski [AR98] developed a compression for the images of a near-field photometric dataset (see Section 2.3) such that they can require less memory. Their method reaches a file compression ratio of up to 37.9, while still being efficiently decodeable in a few milliseconds.

Mas et al. [MMP08] also developed a compression for near-field measurements. They compress the rayset representation of the data into a set of point lights. A rayset is a set of particles on a bounding surface with a position and direction. Mas et al. cluster the rays depending on the particle positions and orientations. The degree of the clustering is variable and depends on the particle density. For each cluster, they create an anisotropic point light on a virtual bounding surface. Depending on the number of used clusters, they can reduce the required memory by a compression ratio of up to ~54. Besides this high compression ratio, they present an importance sampling technique for their compressed data structure.

Heidrich et al. [HKSS98] presents a technique to efficiently render virtual complex light sources by precomputing their field of light of which a discretized version is stored. With this field of light, the light source acts as a black box during the rendering process, as the lamp's geometry is abstracted away which for example reduces the required path depth when path tracing.

Velázquez-Armendáriz et al. [VDWG15] presents another approach to render complex luminaires efficiently with anisotropic virtual point lights (APLs) and a radiance volume. APLs are used to compute the illumination of the scene by the light source, whereas the radiance volume is used for the illumination of the light source itself. Both, the APLs and the radiance volume, are precomputed for a specific light source with particle tracing starting from the lamp in the luminaire, similar to Instant Radiosity as presented by Keller [Kel97], until the particles either leave the structure of the luminaire or the traced path reaches a certain depth. All exiting particles are then clustered into APLs, which have a radiant intensity distribution matching the radiance by the particles in the cluster. For the radiance volume, the radiance by all particles inside the luminaire is considered and stored in voxels as spherical harmonics. The precomputation step can take long (up to 3 hours per luminaire for 1 billion particles), but the resulting data is scene independent and can therefore be easily reused. Their implementation requires up 70MB per luminaire (with a radiance volume resolution of $32^3$ and 512 APLs).

CHAPTER 4

# Real-Time Direct Lighting by Photometric Area Lights

The main topic of this thesis is the real-time evaluation of direct lighting in the context of photometric area lights.

This chapter discuss the real-time evaluation of direct lighting in more detail as this thesis is concerned with this in the context of photometric area lights. While the mathematical foundation for direct lighting and its computation is presented in Section 2.6, this chapter goes into further detail regarding direct lighting in real time and with photometric area lights.

Direct lighting can be derived using one of the direct lighting integrals presented in Section 2.6, Equation 2.46 and Equation 2.48. Accurate results can be computed with Monte Carlo integration by using the corresponding estimators. However, Monte Carlo integration is not feasible for real-time applications even with the presented importance sampling approaches, as a large number of samples is needed.

To address this problem, various approximations were developed that create visually plausible results in real time for area lights with a diffuse or Phong-like emission characteristic (see Section 3.2). Techniques as presented by Drobot [Dro14] and Lagarde and De Rousiers [LR14] use a small number of well-chosen samples to approximate the direct lighting by area lights. Other techniques analytically compute the direct lighting by area lights, e.g., irradiance tensors as developed by Arvo [Arv95a] and further improved by Lecocq et al. [LDSM17], which allow approximating direct lighting by area lights with a Phong-like emission characteristic in real-time.

These approximations, however, are not developed for area lights whose emission characteristics are defined by far-field photometric profiles, i.e., photometric area lights. The additional complexity when handling them, compared to area lights with a diffuse or Phong-like emission characteristic, comes from the fact that they are derived from real-world luminaires. Thus, there is a wide variety of emission characteristics, which are not limited in their complexity. Because of the lack of fast, high-quality approximations for photometric area lights, modern lighting-design software resorts to simpler approximations, which may show noticeable artifacts, or more expensive approximations, which allow interactive, but not real-time frame rates (see Section 2.4). For example, DIALux [Gmb18] uses a single photometric point light to approximate luminaires in real time (see Figure 1.2). While this approach is fast, the errors are noticeable at short distances to illuminated surfaces. Relux [web18d], on the other hand, uses a large collection of point lights, which results in a visually plausible approximation, however, it can not be used in real time.

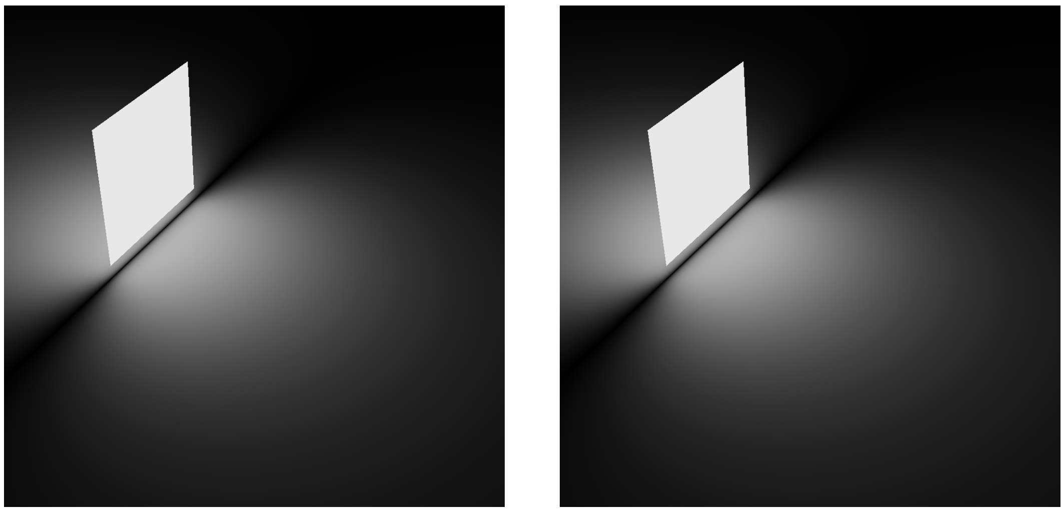
We believe that the issues regarding quality or performance of the interactive sampling-based approaches used by the mentioned lighting-design software is caused by either too few poorly chosen samples (e.g., only the centroid) or by using too many samples although a smaller number would suffice. In this thesis, we show that good approximations can be generated with a small number of samples such that rendering in real time is possible, if the samples are positioned such that the illuminance can be sampled sufficiently as it is done in structured sampling as presented by Lagarde and De Rousiers [LR14] and with the most representative point as presented by Drobot [Dro14].

We deem analytical approaches not suitable, as the existing ones are currently restricted to diffuse area lights or area lights with a Phong-like emission (see Section 3.2). While sampling can be used with photometry without further adjustments by just using the luminance at the sampled position instead of a constant value, analytic approaches require a mathematical model that allows describing the arbitrary emission characteristics of luminaires. Besides the complexity of defining such a model, developing an algorithm which allows evaluating direct lighting with this model in real time is also a complex problem.

In the context of real-time sampling-based techniques, structured sampling and the most representative point are interesting as they are currently used in modern game engines and return visually plausible results. As game engines have a hard real-time constraint, it can be assumed that these techniques are computationally cheap and stable enough for production, which means that they provide a reasonable starting point to develop a fast and stable real-time approximation for photometric area lights. Therefore, we explain these two approaches in more detail in the following two sections and analyze them in the context of photometric area lights. Section 4.1 presents and analyzes structured sampling. Section 4.2 presents and analyzes the most representative point in a generalized way, to which we refer to as *dynamic samples.*

## 4.1 Structured Sampling

*Structured sampling* is a technique presented by Lagarde and De Rousiers [LR14] to approximate direct lighting by diffuse area lights. The technique is an approximation



(a) Reference (b) Structured sampling

Figure 4.1: Structured sampling as presented by Lagarde and De Rousiers [LR14] returns good approximations for diffuse area lights. The reference solution is rendered with 20,000 samples. The images are tone mapped as explained in Section 6.2.

of Monte Carlo integration, where direct lighting is computed with the estimator given in Equation 2.47 and a constant set of well-selected samples. The samples are selected a priori such that only a small number of samples is needed to compute a plausible approximation. An approximation rendered with scattered sampling is shown in Figure 4.1.

Given is the estimator for direct lighting in Equation 2.47. As this thesis is only concerned with diffuse light transport, we will use a diffuse BRDF $f_r(x, \hat{\omega}_o, \hat{\omega}_i) = \rho/\pi$, which results in the simplified estimator:

$$L_{d,N} = L_e(L)\frac{\rho}{\pi N}\sum_{i=1}^{N}\frac{(\hat{\omega}_i \cdot \hat{n}_x)}{p(\hat{\omega}_i)}. \tag{4.1}$$

Instead of using many randomly chosen samples as is done in Monte Carlo integration, Lagarde and De Rousiers use a small number of well-selected samples, which cover the integration domain, i.e., the area of the light source, sufficiently. These are the four corner points and the barycenter for a rectangular light source. Lagarde and De Rousier use them as if they were drawn randomly form a uniform distribution $p(\hat{\omega}_i) = 1/\Omega_L$, because these samples are placed uniformly on the light source. The final scattered sampling estimator with the uniform PDF for a diffuse area light is given by

$$L_{d,N} = \Omega_L L_e(L)\frac{\rho}{\pi N}\sum_{i=1}^{N}(\hat{\omega}_i \cdot \hat{n}_x). \tag{4.2}$$

Using this estimator with the 5 well-selected samples results in a visually plausible approximation without artifacts, which hardly differs from the reference. It is efficient to evaluate due to the small number of samples, which allows it to be used in real-time applications. Selecting the samples a priori, however, means that the form of the light must also be known before. This method is therefore not flexible, since it can only handle known light forms.

Structured sampling can also be used with photometric area light sources. To achieve this, the estimator has to be adopted and suitable samples need to be selected. To use this estimator with photometric data, the incoming light $L_e(x_{\hat{\omega}_i}, -\hat{\omega}_i)$ has to be placed back into the sum, since it no longer emits a constant luminous intensity:

$$L_{d,N} = \Omega_L \frac{\rho}{\pi N} \sum_{i=1}^{N} L_e(x_{\hat{\omega}_i}, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x). \tag{4.3}$$

The selection of the samples is crucial for the accuracy and visual quality of the approximation. Lagarde and De Rousiers use the corners and the barycenter of an area light, which yields good results for diffuse emitters. But as soon as the emission profile of a light becomes less diffuse, sample patterns become visible. Figure 4.2 shows the technique with ARCOS (see Section 6.3), a luminaire with a high directionality. It can be seen that the samples can be distinguished based on the illumination near the light source. This shows that sampling the corners and the barycenter is not a reasonable choice for arbitrarily emitting area lights.

One approach to solve this is to use a low-discrepancy sample set with more samples. The *Poisson Disk Distribution* is often used to generate low-discrepancy samples, because it is not prone to aliasing and artifacts [Coo86]. Poisson sampling is based on uniform sampling [DW85]. It differs in that there is a minimum distance constraint on the samples.

Figure 4.3 shows the result of using a fixed Poisson sample set with various numbers of samples. It can be seen that for a small number of samples, similar sample patterns appear as when the samples proposed by Lagarde and De Rousiers are used. While increasing the number of samples reduces the artifacts, the still remain visible.

## 4.2 Dynamic Samples

Instead of using several well-chosen samples, Drobot [Dro14] approximates area lights with a single sample. This is a form of importance sampling where only the most important sample is used to approximate the whole function. The position of this sample on the light source is referred to as the *most representative point* (MRP) [LR14]. Its position depends on the spatial relation between the illuminated point and the light source. To generalize this approach, we refer to samples positioned at dynamic points like the MRP as dynamic samples.

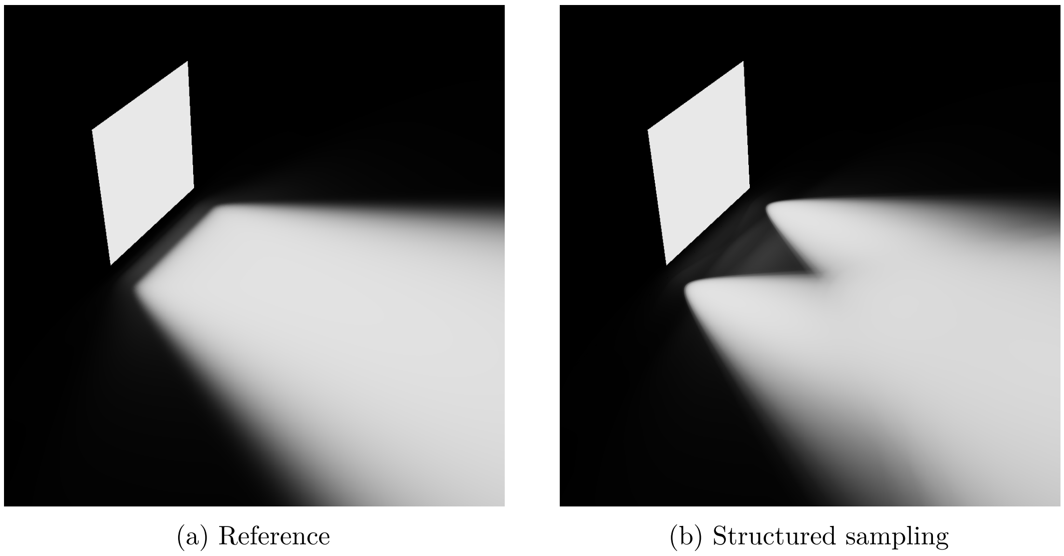

(a) Reference (b) Structured sampling

Figure 4.2: Structured sampling as presented by Lagarde and De Rousiers [LR14] with ARCOS (see Section 6.3), a luminaire with a directional emission. It can be seen that the sample positions are visible. The reference solution is rendered with 20,000 samples. The images are tone mapped as explained in Section 6.2.

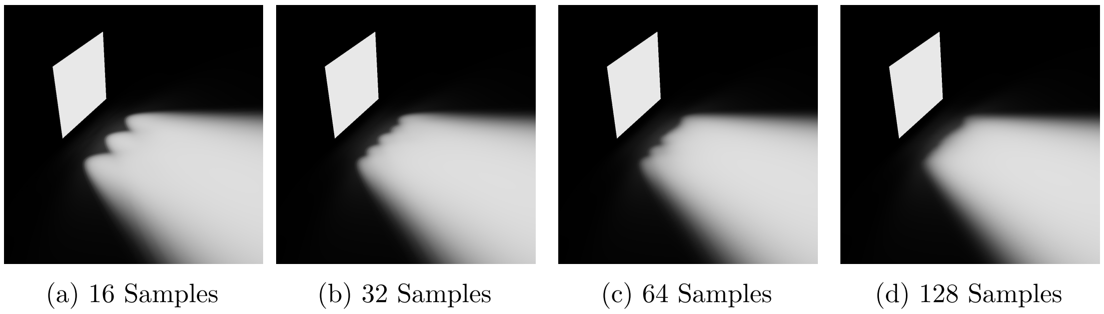

(a) 16 Samples (b) 32 Samples (c) 64 Samples (d) 128 Samples

Figure 4.3: Structured sampling using a fixed Poisson sample set of various sizes with ARCOS (see Section 6.3), a luminaire with a directional emission. Sample patterns remain visible, even with 128 samples. The images are tone mapped as explained in Section 6.2.

The MRP is derived from spatial heuristics. Given is a point $x$ with a diffuse BRDF that is illuminated by a diffuse area light $L$ that is totally visible from $x$. The illuminance of $p$ by $L$ can be computed with a simplified form of Equation **??**:

$$L_d(x, L) = L_e(L) \int_{x' \in S_L} \frac{(\hat{\omega}_i \cdot \hat{n}_x)(-\hat{\omega}_i \cdot \hat{n}_L)}{\|x' - x\|^2} dA(x'). \tag{4.4}$$

The BRDF term is omitted for simplicity and the geometry term can be removed because the light source is assumed to be fully visible. Because the light source is diffuse, the luminance is constant, which means that $L_e$ can be moved out of the integral.

To find a single sample that best approximates the integral in Equation 4.4, the position on the light source has to be found which maximizes the term inside the integral on the domain of the surface $S_L$ of the area light. Drobot deemed the exact calculation too expensive, which is why he uses heuristics instead to approximate the maximum.

Firstly, the term can be simplified by assuming that the term $(-\hat{\omega}_i \cdot \hat{n}_L)$ is roughly constant. By moving it out of the integral, the term which needs to be maximized becomes

$$\frac{(\hat{\omega}_i \cdot \hat{n}_x)}{\|x' - x\|^2}. \tag{4.5}$$

The maximum of this term is approximated by finding the maximum of the numerator and the minimum of the denominator respectively and combining the results. We refer to the position of the sample which maximizes the numerator as *normal point* $p_N$ and the position of the point which minimizes the denominator as *Closest Point* $p_C$. The position which results by the combination of these two is referred to as the MRP $p_M$. The corresponding illustration is given in Figure 4.4.

**Normal point (NP) $p_N$** The numerator $(\hat{\omega}_i \cdot \hat{n}_x)$ is maximized when $\hat{\omega}_i = \hat{n}_x$. Geometrically, this means that a sample maximizes the numerator when its positioned at the intersection point of the normal $\hat{n}_x$ and the light plane. It is, however, not guaranteed that the sample lies inside the area $S_L$. If the sample is positioned outside, its position has to be clamped to the closest point on $S_L$. Moreover, it is possible that $\hat{n}_x \cdot \hat{n}_L \geq 0$, which means that the light plane of $L$ is either parallel to or turned towards $\hat{n}_x$. In the first case, there would not be an intersection. In the second case, the intersection would be below $x$, which would minimize the function. To handle this issue, it is necessary to skew $\hat{n}_x$ towards the light plane if $\hat{n}_x \cdot \hat{n}_L \geq 0$ is given.

**Closest point (CP) $p_C$** The denominator can be minimized by projecting $x$ onto the light plane. As with the NP, it can occur that the projected point does not lie inside $S_L$. So it may also need to be clamped to the closest point on the area of the light source. After the clamping, it can occur that the resulting point is below the horizon of $x$, which

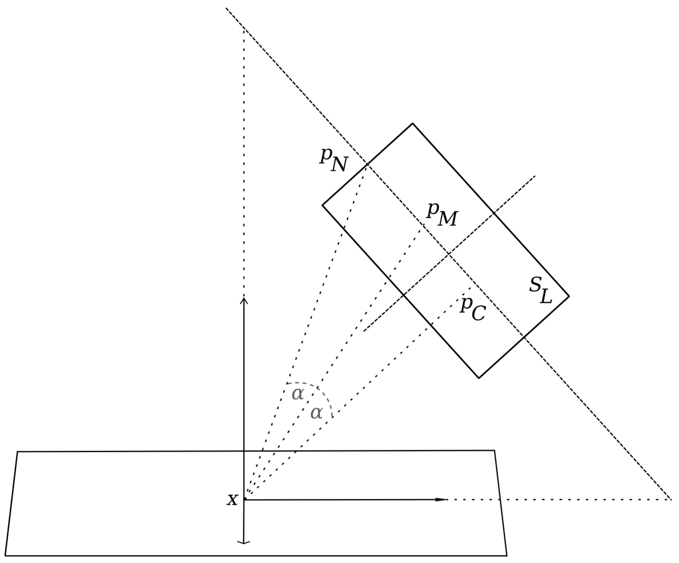


Figure 4.4: The 3 dynamic points: normal point $p_N$, closest point $p_C$ and the most representative point $p_M$. The position of the points on the area light $L$ depends on the spatial relation to $x$.

means that it would not be visible. In this case, the clamped point needs to be projected on the intersection line of the light plane and the horizon plane of $x$.

**Most representative point (MRP) $p_M$** Drobot states that computing the MRP as the intersection point of the light plane and the half-vector of $\overrightarrow{xp}_C$ and $\overrightarrow{xp}_N$ returns generally good results with a small error. This statement is based on observations and empirical tests. He also proposes a faster method to compute the MRP, which does not clamp the CP and the NP to the area light but the resulting half-vector. This, however, results in a larger error.

The MRP is designed such that a single sample at this position provides a plausible approximation for a diffuse area light. As can be seen in Figure 4.5, a single sample position at the MRP returns a good approximation. The image is rendered using the same Monte Carlo estimator as used by structured sampling (Equation 4.3), but with just one sample.

This, however, is not the case for photometric area lights, as shown in Figure 4.6b. The illumination starts noticeably further away from the area light source as in the reference image. Still, the MRP as well as its construction points, the closest point and the normal point, appear to be interesting choices, since each of these points maximizes a term in the direct lighting equation given in Equation **??**. Of these 3 dynamic points, the closest point appears to be especially interesting in the context of photometric area lights. As shown in Figure 4.6c, sampling its position approximates the illumination near the area light well. The worst approximation is given by using the normal point (Figure 4.6d).

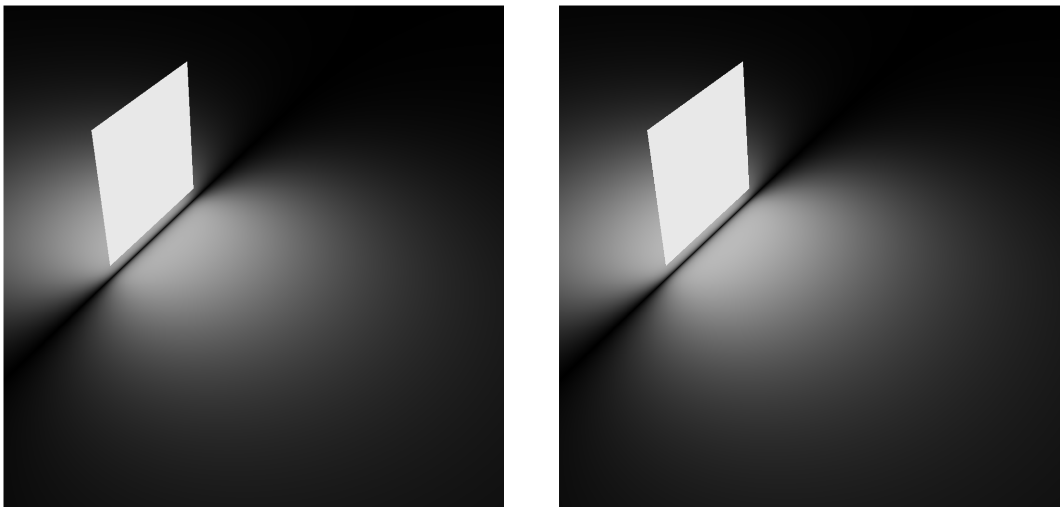

(a) Reference (b) MRP

Figure 4.5: Sampling at the MRP as presented by Drobot [Dro14] returns good approximations for diffuse area lights. The reference solution is rendered with 20,000 samples. The images are tone mapped as explained in Section 6.2.

This, however, does not mean that the normal point is not an interesting dynamic sample position. It just means that it is not an interesting choice if only one dynamic sample is used.

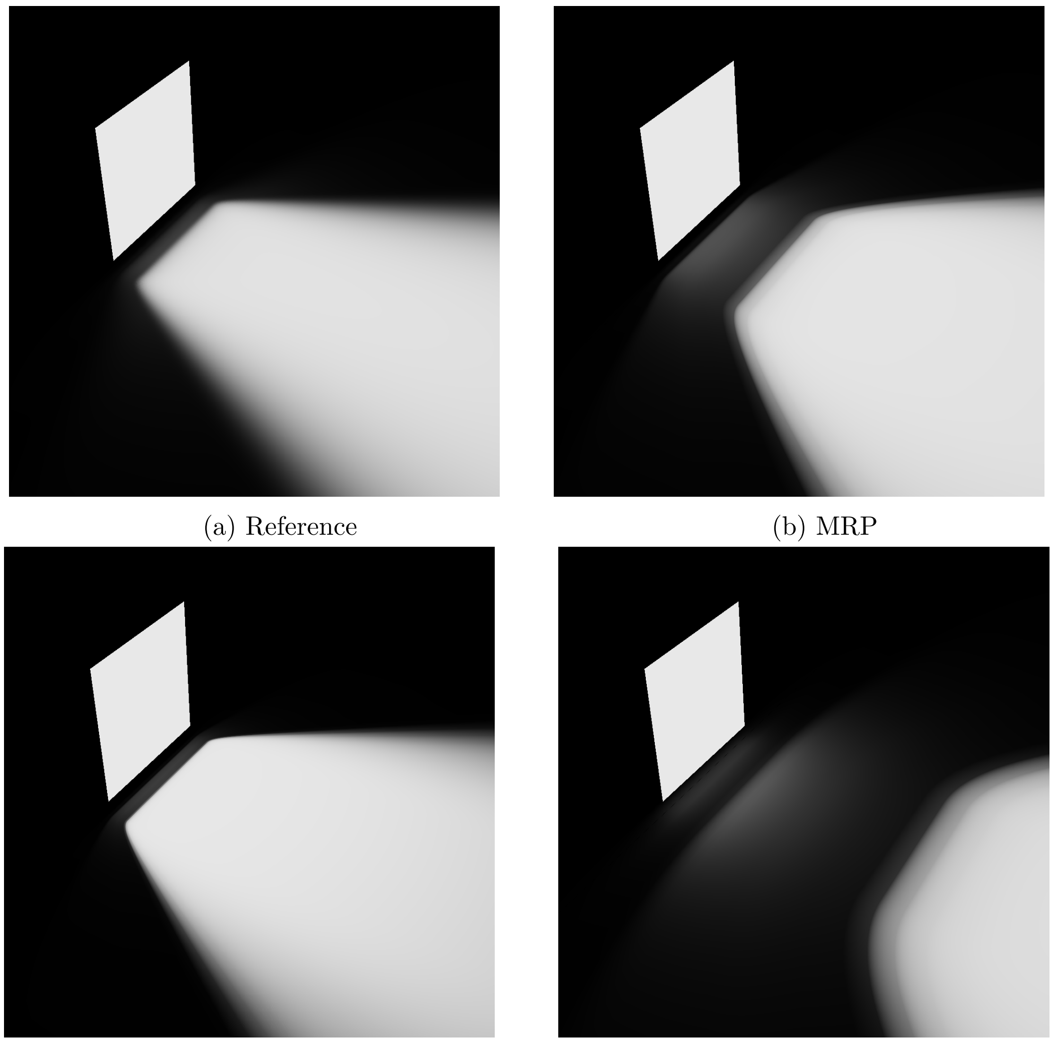

(a) Reference (b) MRP (c) Closest point (d) Normal point

Figure 4.6: Sampling at the MRP, closest point and nomal point with ARCOS (see Section 6.3), a luminaire with a directional emission. The reference solution is rendered with 20,000 samples. The images are tone mapped as explained in Section 6.2.

CHAPTER 5

# Scattered Sampling

This chapter presents *scattered sampling*, our new approach to approximate the direct lighting by a photometric area light in real time with a small number of samples.

Scattered sampling is the result of combining structured sampling (see Section 4.1), a technique presented by Lagarde and De Rousiers [LR14], with sample points like the *most representative point* (see Section 4.2), presented by Drobot [Dro14], which is a point whose position depends on the spatial relation between the shaded surface point and the area light source. We propose to combine these techniques, as they appear to complement each other. While a dynamic sample, e.g., positioned at the most representative point, allows sampling pattern-free approximations of photometric area lights, it does not cover the integration domain sufficiently, which, in contrast, is done by structured samples.

We combine these techniques by using both types of samples together to approximate the direct lighting. We refer to sample sets consisting of both types of samples as *scattered samples*. We borrow the term from Freeden at al. [FNS15], who refer to scattered samples as irregularly distributed samples. Since the resulting sample sets are neither random nor structured, we deem the term scattered to be fitting.

Scattered sampling is designed such that it can be implemented in a fragment shader. The algorithm consists of two steps:

1. **Generate a scattered sample set** Each frame, a dynamic sample is computed and added to the structured sample set, resulting in a scattered sample set.
2. **Approximate direct lighting with the sample set** Given the scattered samples from Step 1, the incoming irradiance by a photometric area light is computed.

For the scattered sampling set we use the corners of the area light as structured samples and the closest point as dynamic sample. We use the closest point because it already generates plausible approximations on its own (see Section 4.2) and add the corner samples because they cover the light source well.

To compute an approximation for the incoming irradiance with a scattered sample set in the second step, the Monte Carlo estimator in Equation 4.3 cannot be used as it is done by structured sampling. This is because by adding a dynamic sample to a structured sample set, the uniformness of the sample set, which is required to justify the use of the Monte Carlo estimator with a uniform PDF, is lost. As the dynamic sample's position can be anywhere on the light source, it can come close to structured samples or even overlap them. Therefore, another integration technique is required.

Instead of Monte Carlo integration, we use a cubature technique currently used in Geomathematics [FNS15] based on the *Delaunay triangulation* to integrate with scattered samples. Given the scattered samples as normalized directions, we create a Delaunay triangulation based on them on the unit sphere and use the areas of the resulting triangles to weigh the samples in the cubature technique. This process is explained in more detail in the following sections.

Section 5.1 explains how scattered samples given as directions can be integrated on the sphere and why we decided to use a cubature technique based on the Delaunay triangulation. The following Section 5.2 goes into more detail regarding the implementation of the cubature technique on the GPU and how a Delaunay triangulation can be derived in real time.

Throughout this chapter, we will assume that the area light is rectangular for simplicity. Our technique, however, can also be used with arbitrary convex polygons. Moreover, we will provide exemplary images for the presented techniques as well as our approach, which we will evaluate briefly based on the visual appearance. A detailed analysis of the performance of the technique with regards to errors is provided in Chapter 6.

## 5.1 Numerical Integration with Scattered Samples

Given are scattered samples, which are a collection of structured samples and dynamic samples. Each set of scattered samples is deterministic and unique for each surface point in a scene. Scattered samples are similar to structured samples, however, by also using dynamic samples, the integration technique used in structured sampling is no longer applicable.

Structured sampling is an approximation of Monte Carlo integration and uses an estimator to compute the illuminance by an area light. The approximation results from using the Monte Carlo estimator and a uniform PDF with the deterministic structured samples. This approach is justified by assuming that the manually distributed samples are randomly drawn from a uniform PDF. This assumption, however, does not hold for scattered samples, as the generated sample sets can vary much and are not guaranteed to be approximately uniformly distributed. While there can be sets with a roughly uniform distribution, there can also be sets where dynamic samples come close to other samples, or even overlay them.

As the Monte Carlo estimator is not suitable, another cubature technique is required that works on the sphere, as the sample positions are defined as normalized directions. One cubature technique which works without random samples and distribution assumptions is cubature based on partitioning the sphere [FNS15, p. 1195]. This technique is used in Geomathematics to integrate functions on the unit sphere with arbitrarily distributed samples that are not structured (e.g., in a grid).

### Cubature Based on Voronoi Tesselation

One way to partition the sphere is Voronoi tessellation [FNS15, p. 1205]. Voronoi tessellation refers to partitioning a space based on the cells of a Voronoi diagram, which is first defined. The definition by Aurenhammer [Aur91] is used adapted for spherical geometry. Given is the unit sphere $\mathbb{S}^2 := \{x \in \mathbb{R}^3 : \|x\| = 1\}$ and a set of points $X_N$ on this sphere. The *Dominance* of a point $p \in \mathbb{S}^2$ over a point $q \in \mathbb{S}^2$ is given by

$$dom(p, q) = \{x \in \mathbb{S}^2 | dist(x, p) \leq dist(x, q)\} \tag{5.1}$$

where *dist* is the geodesic distance function:

$$dist(p, q) = \arccos(p \cdot q) \ \in \ [0, \pi]. \tag{5.2}$$

The dominance separates all points closer to $p$ from the ones closer to $q$. The *Region* of $p$ consists of all elements of the sphere over which $p$ has dominance with regard to the other points:

$$reg(p) = \bigcap_{q \in S \setminus \{p\}} dom(p, q). \tag{5.3}$$

All points inside the region of $p$ are closer to $p$ than to any other point $q \in \mathbb{S}^2$. The points lying exactly on the line between two regions are equidistant to both the associated points of the regions. This partition of a sphere is called a Voronoi diagram $V$ and the regions are referred to as Voronoi cells $T(p)$.

Given is a function $f(x)$ defined on the unit sphere, a set of samples $X_N \in \mathbb{S}^2$ and the Voronoi cells $T(x_i) = T_i \ \forall x_i \in X_N$. An estimate for the integral of $f(x)$ is given by [FNS15, p. 1205]

$$Q_N = \sum_{i=1}^{N} |T_i| f(x_i) \tag{5.4}$$

where $|T_i|$ is the surface area of $T_i$.

This cubature technique can be used to estimate the illuminance by a photometric area light $L_d(x, \hat{\omega}_o)$ for a surface point $x$ because the function is already defined on $\mathbb{S}^2$. Since $L_d(x, \hat{\omega}_o)$ returns only values in $\Omega_L \in \mathbb{S}^2$ and 0 elsewhere, it is enough to apply this cubature only the $\Omega_L$.

While the cubature is basically a sum of weighted samples, the complexity of this technique comes from the Voronoi diagram. It has to be derived for each set of scattered samples, and the area of each resulting Voronoi cell has to be computed. This has to be done every frame due to the real-time constraint of this thesis. Moreover, the algorithm to generate the Voronoi diagram has to be implemented in a fragment shader, which further reduces the allowed complexity and available memory. Assuming that a scene is rendered with a resolution of $1920 \times 1080$, $2,073,600$ Voronoi diagrams would need to be computed per area light (occlusion ignored) in less than 16 milliseconds (60 FPS)[1].

There exist several solutions to utilize the GPU to efficiently compute Voronoi diagrams ([RT06][RT07][YRGW11][RLW$^+$11]). But besides not being designed for spherical Voronoi diagrams, the main issue with these techniques is that they utilize the whole GPU to compute a single Voronoi diagram in real time, not several. So even newer solutions like the approach presented by Scheider et al. [SKW09], which only needs $2.5ms$ to derive a crude approximation of a single Voronoi diagram, are too slow and memory consuming for our use case.

Since computing a Voronoi diagram per fragment in real time is not feasible, another approach would be to precompute $|T_i|$, save the values in a texture and read the corresponding values during the rendering process. We implemented this approach with a set of scattered samples for a rectangular area light containing 5 structured samples (corners and barycenter) and a dynamic sample, the closest point. An exemplary illustration is shown in Figure 5.1.

The areas of the Voronoi cells per sample $|T_i|$ can be precomputed for various positions of the dynamic sample with any Voronoi algorithm. We store the results in two textures, where each color channel holds $|T_i|$ for a sample. Our textures are shown in Figure 5.2.

During rendering, the position of the dynamic sample on the area light is then used as a UV coordinate to lookup the Voronoi area for each sample. To store the Voronoi areas, an HDR texture is used. The accuracy of the technique during rendering depends on the resolution of the texture. The technique is shown in Figure 5.3. It can be seen that although only 6 samples are used, no sample patterns are visible near the area light.

While this technique is computationally efficient, it has several drawbacks. Firstly, an error is introduced by precomputing the surface area of the Voronoi cells on the area light, visible in Figure 5.3b as irregularities at the borders of the illuminated surface. The surface areas $|T_i|$ are derived from the Voronoi tessellation on the rectangular area light,

[1]Actually less than 16 milliseconds, assuming that other rendering techniques and effects may also be used, which also require computation time.

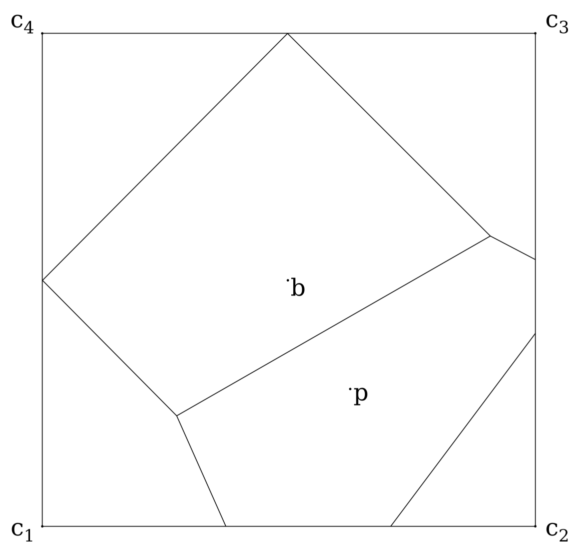


Figure 5.1: Voronoi tessellation of a rectangular area light with a scattered sample set: The 4 corners $c_1 \ldots c_4$, the barycenter $b$ and a dynamic point $p$. The tessellation depends on the position of $p$.

not from the tessellation of the projected area light. Secondly, this approach is not usable for practical applications. For example, if the area light comes from below the horizon of an illuminated point, it needs to be clipped at the horizon. This clipping process results in another convex area light, which has, depending on its position and orientation, only 3 or up to 5 corners. The precomputated values for the rectangular area light can no longer be applied. Additional precomputations would be required for arbitrary clippings of the area light. One approach to achieve this would be to use the Hesse normal form to parameterize cuts in the 2D-space of area light. This introduces 2 more dimensions (the normal vector of the clipping line and its distance from a prior chosen origin in the area light plane), which leads to a total 4 dimensions. However, creating a 4D-lookup texture that stores up to 7 values per entry (3 - 5 corners, barycenter and dynamic point) is a complex problem.

### Cubature Based on the Delaunay Triangulation

Since Equation 5.4 is not practical to use due to the issues arising by deriving the Voronoi cells $T_i$, we instead propose to use another cubature technique based on the *Delaunay* triangulation. It is a relatively uniform triangulation which means that it avoids thin triangles if possible. This property makes the Delaunay triangulation a suitable choice when interpolating data from samples. [Ren97]

A triangulation is referred to as Delaunay when the *empty circumcircle interior property* is valid for all triangles in a triangulation [Ren97]. This property states that there must not be a vertex inside the circumcircle of a triangle. An example is shown in Figure 5.4.

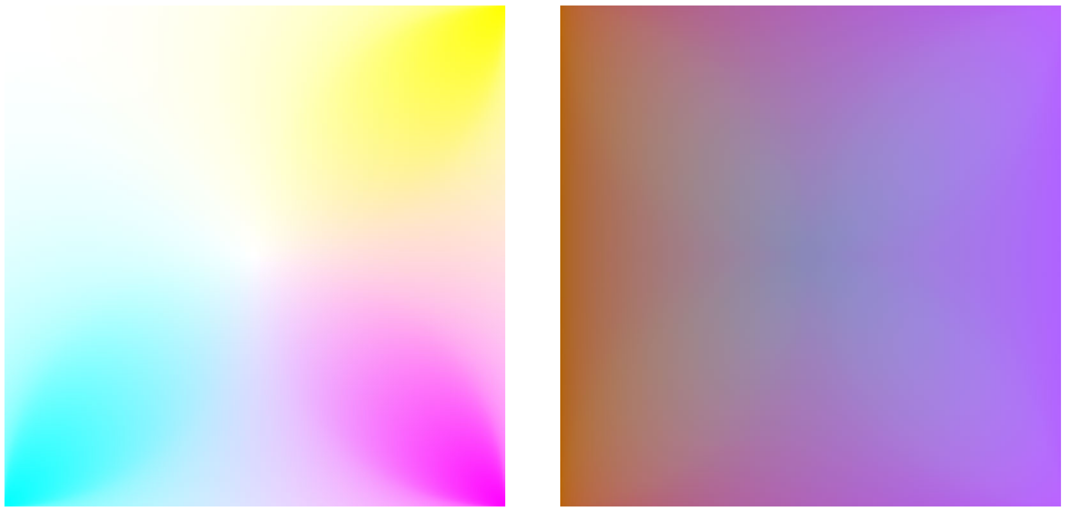


Figure 5.2: Textures storing the areas of the Voronoi cells $|T_i|$. The left texture stores the area of a corner cell per channel. The right stores the area of the barycenter cell and the dynamic cell in the first two channels. The color variance in the second texture occurs because we also store the UV-coordinates in the blue and the alpha channel for debugging reasons. The UV-coordinate of the dynamic point $p$ is used to look up the Voronoi cell areas for each sample.

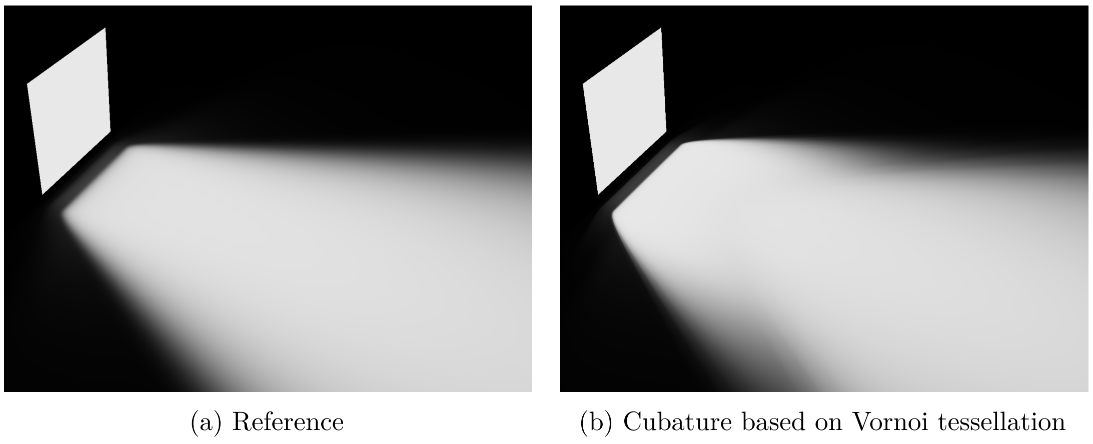


Figure 5.3: Approximating the illuminance by ARCOS (see Section 6.3) using the cubature based on Voronoi tesselation with a scattered sample set consisting of the corners, the barycenter and the closest point. The reference solution is rendered with 20,000 samples. The images are tone mapped as explained in Section 6.2.

The circumcenter for the spherical triangle $\triangle_{ijk}$ is given by Equation 5.5. [Ren97]

$$v_{ijk} = \frac{(x_k - x_i) \times (x_j - x_i)}{\|(x_k - x_i) \times (x_j - x_i)\|} \tag{5.5}$$

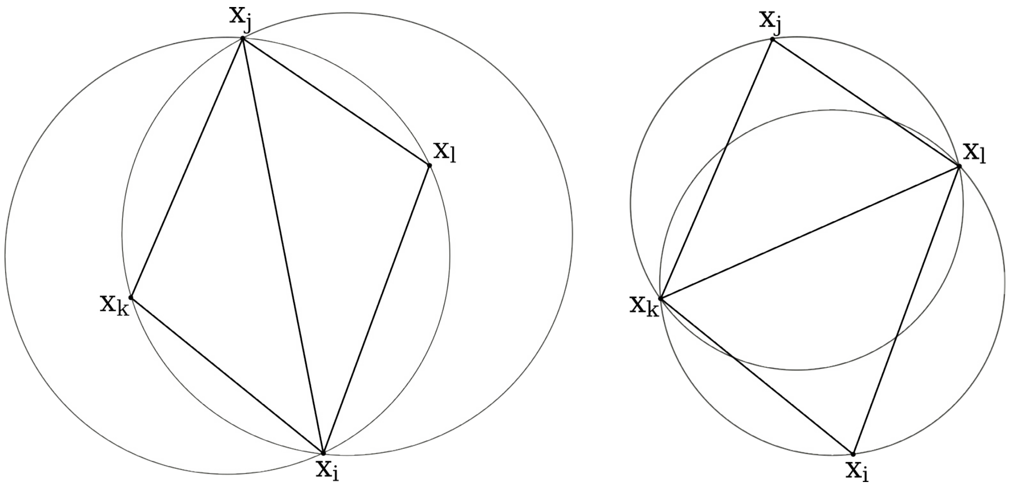


Figure 5.4: The left triangulation is not Delaunay due to the violation of the *empty circumcircle interior property.* The triangulation on the right is Delaunay because both circumcircles do not contain a vertex.

The Delaunay triangulation is the dual to Voronoi diagrams. This means that one can be created from the other. Given a point in a Delaunay triangulation, the corresponding Voronoi cell is given by circumcenters of the triangles containing the vertex. On the contrary, if a Voronoi diagram is given, the corresponding Delaunay triangulation can be derived by connecting the points of neighbouring cells that share more than one vertex. [Ren97]

The cubature technique utilizing the Delaunay triangulation is given by

$$Q = \sum_{\triangle_{ijk} \in D} \frac{f(x_i) + f(x_j) + f(x_k)}{3} | \triangle_{ijk} | \tag{5.6}$$

where $D$ is the Delaunay triangulation of a sample set consisting of the spherical triangles $\triangle_{ijk}$ $\forall i, j, k$. [FNS15, p. 1206] The samples $x_i$, $x_j$ and $x_k$ do not lie on a common line and are in counterclockwise order looked at from the center of the sphere (see Figure 5.5), i.e., $\det(x_i, x_j, x_k) \leq 0$. $| \triangle_{ijk} |$ is the spherical excess of the triangle:

$$| \triangle_{ijk} | = \alpha_i + \alpha_j + \alpha_k - \pi \tag{5.7}$$

where $\alpha_i$, $\alpha_j$ and $\alpha_k$ are the angles of $\triangle_{ijk}$.

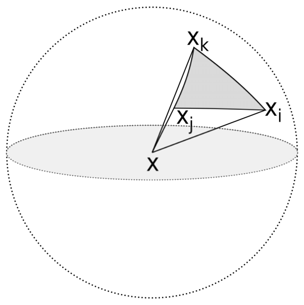


Figure 5.5: Illustration of a spherical triangle as used by the spherical cubature based on the Delaunay triangulation. The vertices are ordered in counterclockwise order relative to the center of the sphere.

Compared to the cubature based on Voronoi tessellation, this cubature uses each sample several times. The exact number is given by the number of triangles which contain the corresponding vertex [FNS15, p. 1206]. Moreover, the result of the cubature techniques differ. While the cubature based on Voronoi tessellation uses the area of the corresponding Voronoi cell, this cubature technique computes an average value and weighs it with the area of the corresponding triangle.

Using the cubature technique based on a Delaunay triangulation solves several problems encountered by using the cubature based on Voronoi tessellation. First of all, a Delaunay triangulation can be generated fast and accurately with the *Flip-Algorithm* [BE70], which transforms an arbitrary triangulation into an Delaunay triangulation in $O(N^2)$. The algorithm can be easily implemented on the GPU due to its simplicity and allows some optimizations based on precomputations. Secondly, since the Delaunay triangulation can be derived correctly per fragment, no additional error is introduced since no approximations are required. Thirdly, computing the excess of a spherical triangle is cheaper (see Equation 5.7) than computing the surface area of an arbitrary Voronoi cell. The Flip-Algorithm is explained in Section 5.2 together with our GPU implementation of the algorithm and the cubature technique.

## 5.2 Real-Time Delaunay Triangulation

This section explains how we derive a Delaunay triangulation of the projected area light based on scattered samples such that we can use the cubature technique shown in Equation 5.6 to compute the illuminance of a surface point by the respective photometric area light. The reasons for using this approach as well as the definition of the Delaunay triangulation are given in Section 5.1.

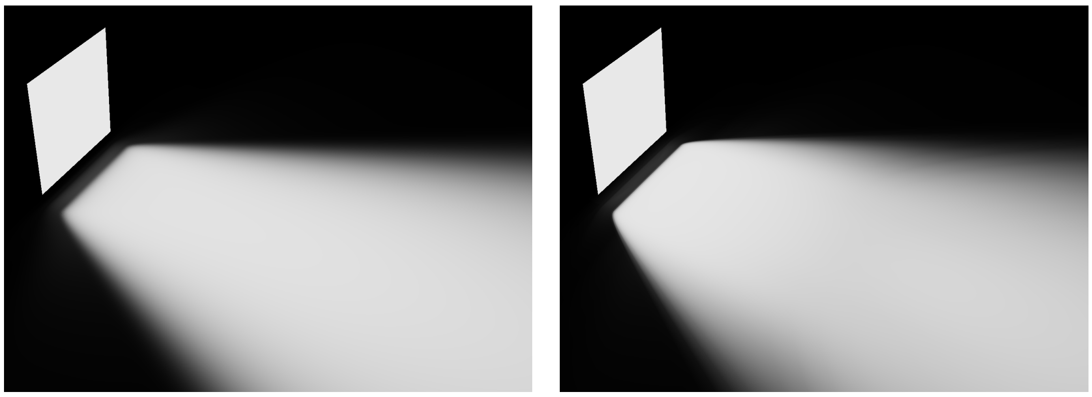
We use the *Flip-Algorithm* [BE70] to generate a Delaunay triangulation per fragment per frame. The algorithm starts with an arbitrary triangulation. For every edge $e = (x_i, x_j)$ that is not part of the convex hull we check whether the *empty circumcircle interior property* is valid for the triangles $\triangle_{ijk}$ and $\triangle_{ilj}$ on each side of $e$. If the property is not fulfilled, $e$ is removed and replaced by $e' = (x_k, x_l)$, which creates the new triangles $\triangle_{klj}$ and $\triangle_{kil}$. Replacing $e$ by $e'$ is referred to as flipping the edge $e$. Hence the name Flip-Algorithm. If edge $e$ is flipped, the other edges of $\triangle_{klj}$ and $\triangle_{kil}$ have to be checked again if they already have been checked and are not part of the convex hull. This algorithm creates a Delaunay triangulation with a complexity of $O(N^2)$, where $N$ is the number of samples.

Practically, this algorithm can be implemented using a stack as shown in Algorithm 5.1. The algorithm has several properties which make it suitable for a real-time application. First of all, it works with the topology of the triangulation rather than the form of the area light. It works the same whether the light has 3, 4 or 5 corners and is independent of any distortions of the light caused by the projection onto the unit sphere. Another advantageous property is that the *empty circumcircle interior property* can be checked with a single formula based on Equation 5.5 for the quadrilateral formed by the two triangles $\triangle_{ijk}$ and $\triangle_{ilj}$ [Ren97]:

$$\det(x_i - x_j, x_l - x_j, x_k - x_j) = (x_i - x_j) \cdot ((x_l - x_j) \times (x_k - x_j)) < 0. \tag{5.8}$$

**Algorithm 5.1:** Flip-Algorithm

```
Data: A sample set $X_N$ and any triangulation of $X_N$
Result: Delaunay triangulation of $X_N$
1  Initialize empty edge stack $S$
2  forall $e$ in edges $\notin$ convex hull do
3      Push $e$ on $S$
4      Mark $e$ as unchecked
5  end
6  while $S$ is not empty do
7      Pop $e = (x_i, x_j)$ from $S$
8      if Empty circumcircle interior property not valid for $\triangle_{klj}$ or $\triangle_{kil}$ then
9          Flip $e$ and mark $e$ as checked
10         forall $e'$ in $\triangle_{klj} \cup \triangle_{kil} \setminus e$ do
11             if $e'$ is checked then
12                 Mark $e'$ as unchecked
13                 Push $e'$ on $S$
14             end
15         end
16     end
17 end
```



(a) Reference (b) Cubature based on Delaunay triangulation

Figure 5.6: Approximating the illuminance by ARCOS (see Section 6.3) using the cubature based on Delaunay triangulation with a scattered sample set consisting of the corners and the closest point. The reference solution is rendered with 20,000 samples. The images are tone mapped as explained in Section 6.2.

The last property which makes this algorithm and therefore the cubature based on Delaunay triangulation so practical is that the initial triangulation can be chosen arbitrarily. This means that initial triangulations can be predefined for various cases (unclipped light or clipped light), which will most likely require no flip at all, which results in a simple process where the stack is emptied without any manipulation of the topology.

An image rendered with our implementation of this technique is shown in Figure 5.6. The used scattered sample set consists of the four corners and the closest point. The figure shows that this technique returns a stable approximation with a small number of samples. Artifacts are less visible compared to our implementation of the cubature based on the Voronoi tessellation shown in Figure 5.3, although the Delaunay cubature uses one sample less.

We explain our implementation of the algorithm in the following two sections. We start with explaining the used data structure in Section 5.2.1. Then, in Section 5.2.2, we go into more detail regarding the implementation of the algorithm.

### 5.2.1 Data Structure

This section describes the data structure we use for the cubature theoretically and explains how we implemented it. We go into more detail on where the data structure is created and filled in the following section. The data structure used for the cubature based on Delaunay triangulation (see Section 5.1) and Flip-Algorithm should meet the following requirements:

1. Direct access to the quadrilateral given by the triangles of an edge. This is required so that the *empty circumcircle interior property* can be checked efficiently. Moreover, the direct access allows a fast local transformation in the case of a flip.
2. Direct access to the vertices of the final triangles for the cubature. By granting this access, Equation 5.6 can be implemented efficiently as a summation over a list where all needed values are instantly accessible.

While there exist several data structures which are commonly used to handle triangle meshes ([Bau72], [Hop98], [BSBK02], [TM06]), we decided to implement a custom data structure. This decision was based on the real-time constraint of this thesis and the requirement that the algorithm should be implemented in a fragment shader and run efficiently on the GPU. Although existing data structures can also be used, they do not perfectly fit the requirements listed above, which are rather specific for our use case. And since performance is crucial, using a tailored data structure seemed more sensible.

The data structure we are using focuses on the topology of the triangulation. Edges are described implicitly by their quadrilateral, which helps meeting requirement 1. Each edge has access to all the vertices and edges of the quadrilateral. Faces are described by their vertices and their edges. This means that by iterating over the faces, the vertices (samples) are directly available, which fulfills requirement 2. Vertices themselves only store their position on the unit sphere and the function value of the corresponding sample. They have no outgoing relation to edges or faces. This is not required since vertices are never transformed and are read-only constants.

The data is stored in 6 arrays, where the index of an entry is mapped to the index of the object it holds the data for:

- **Vertices (V)** Each vertex $v$ stores a sample $x$, which has a position and luminance value derived from sampling the sample. This data is required by the Flip-Algorithm and the cubature.
- **Edge Vertices (EV)** Each edge stores the indices of its two end-vertices and the third vertex of each of its triangles, to which we refer to as *opposite vertices* ($o$). This vertices are stored in counterclockwise order in the form: $\{v_0, o_0, v_1, o_1\}$.
- **Edge Edges (EE)** Additionally to the vertices, each edge stores indices of the edges of its quadrilateral in counter clockwise order, starting with the edge between $v_0$ and $o_0$: $\{e_0, e_1, e_2, e_3\}$ The neighbor edges are needed in case of a flip, where their respective quadrilaterals have to be updated.
- **Edge Meta Data (EM)** This data is required by the flip algorithm. Each edge stores whether it is checked or unchecked, and whether it is an inside edge or not.
- **Face Vertices (FV)** The indices of the vertices defining a face in counterclockwise order: $\{v_0, v_1, v_2\}$. Edges reference faces by their vertices.

- **Face Edges (FE)** The indices of the edges defining a face in counterclockwise order: $\{e_0, e_1, e_2\}$. The first edge is the edge between $v_0$ and $v_1$.

An example is illustrated in Figure 5.7.

| | | | | | | |
|---|---|---|---|---|---|---|
| EE | 1,2,3,4 | _,_,2,0 | 0,1,_,_ | _,_,4,0 | _,_,0,3 | |
| EV | 0,1,2,3 | 0,_,1,2 | 2,0,1,_ | 2,_,3,0 | 3,_,0,2 | |
| EM | inside checked | inside checked | inside checked | inside checked | inside checked | |

| | | | | | | |
|---|---|---|---|---|---|---|
| V | $x_0$ | $x_1$ | $x_2$ | $x_3$ | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| FV | 0,1,2 | 0,2,3 | | | | |
| FE | 1,2,0 | 0,3,4 | | | | |

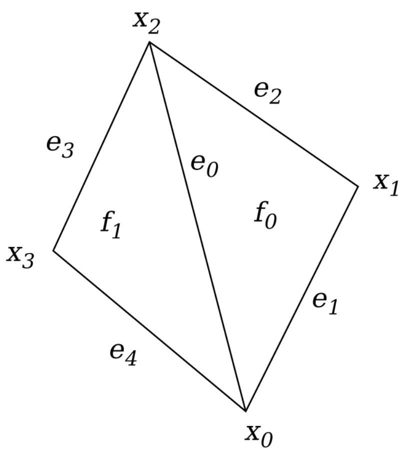


Figure 5.7: Illustration of the representation of the triangle to the right in the data structure, shown as set of arrays to the left. The vertices array stores the samples $x$, which have a position and a luminance value.

The edges are self-contained with direct access to the indices of the vertices and edges of the corresponding quadrilateral. Faces have direct links to their vertices and edges. To reference a face from an edge, the face needs to be searched that has the same vertices as the corresponding triangle of the edge. This means that the relation from face to edge can be resolved in $O(1)$ while edge to face relations are resolved in $O(n_f)$, where $n_f$ is the number of faces. We accept the additional cost by the second relation because the number of faces is usually relatively low. Using this kind of relation reduces the complexity of the data structure.

There exist several possibilities to implement this data structure, with the most trivial being a set of 6 arrays holding vectors with various dimensions. For example, EV and EE can be stored in arrays consisting of 4D integer vectors, where each entry in a vector holds an index. Yet, as the memory for the data structure is allocated in the fragment shader, it should be implemented as compactly as possible. A compact implementation results in less memory usage and better performance since more data can be held in the cache.

Instead of using an integer per index, we write several indices into one integer. This is possible as we use a relatively small number of samples, which results in a small number of triangles and edges. Consider the example of 5 samples: the 4 corners of an area light and a point in the middle of the light. By connecting these samples with 8 edges, the area light would be triangulated with 4 triangles. The biggest index would be given by an edge with 7, which means that all edges can be encoded with only 3 bits.

Instead of using a 32 bit integer to store a single index, we use 1 byte, which allows us to store up to 4 indices in a single integer. The EM array only holds two booleans per edge, which can be encoded with a single bit. Therefore, we use single integers to encode the EM array, where each integer can store information for up to 16 edges. Given the compact representation, the topology of the example can be stored with only 100 bytes. The vertex array V stores the positions as 3D vectors and values of samples as floats, which, in the case of 5 samples, results in a size of 100 bytes.

### 5.2.2 Algorithm

We implemented the Flip-Algorithm (Algorithm 5.1) and the data structure in a fragment shader. Our implementation of the Flip-Algorithm consists of two parts. The first part is generating the initial triangulation (Lines 1-5). The second part is given by the flipping process (Lines 6-18). We use a scattered sample set consisting of the corners of an area light and a dynamic point, in this case the closest point, to explain our implementation, since this is the sample set we use in our reference implementation. At the end of this section, we explain why we chose this sample set and how more complex sample sets could be used.

Since the algorithm as well as our data structure is mainly concerned with the topology, we statically define a set of initial triangulations and stacks in the shader. During the rendering process, we evaluate the relative positions of the samples in the fragment shader and choose the fitting triangulation. The fitting triangulation is then copied from the constant memory and used for the flipping algorithm, with exception of the V array, which is not copied but created dynamically for each fragment.

Disregarding clipping without loss of generality[2], there are 3 cases that can occur due to the relation of the dynamic sample to the structured samples:

1. The dynamic sample is inside the area light source.
2. The dynamic sample is on an edge of the area light source.
3. The dynamic sample lies in a corner of the area light source.

Figure 5.8 shows our initial triangulations for the 3 cases mentioned above. To ensure generality, we order the sample set such that the dynamic sample is always the first sample, followed by the structured samples in counterclockwise order. This order has to be respected when the scattered sample set is generated, otherwise the initial triangulation cannot be used.

The initial stack contains all edges which are not part of the convex hull (inside edges).

There are two steps required to determine which initial triangulation should be used. The first step is to check the number of corners of the area light after clipping it with the

[2]An area light clipped at the horizon only differs in the number of corner points.

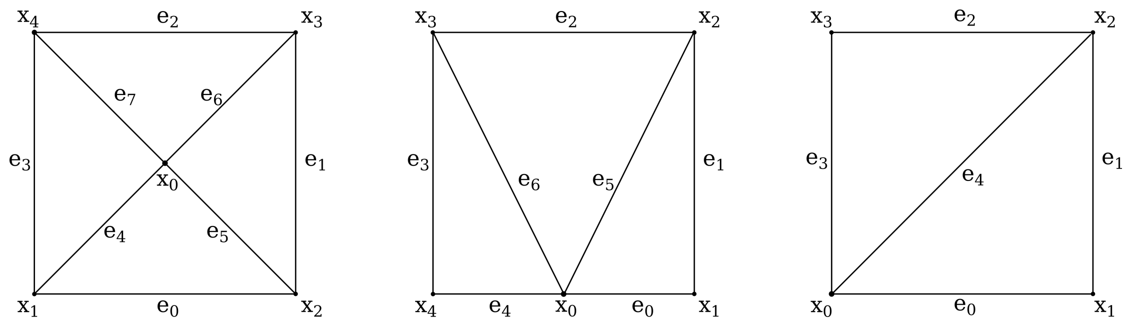


Figure 5.8: Initial triangulations based on the topological relation of the dynamic sample $x_0$ to the structured samples.

horizon. In the case of a rectangular area light, the result can be 3, 4 or 5 corners. The second step is to derive the relative position of the dynamic sample to the corner points.

To do this accurately, we use a clamping function that not only returns the clamped point, but also information regarding the clamping process. It returns an integer that encodes whether the point was not clamped (it was already inside the area light) or the point was clamped to an edge or a corner. In the latter cases, it also returns which edge (defined by two corners in counterclockwise order) or which corner the point was clamped to.

Before using the initial triangulation, it has to be ensured that the structured samples are in the correct order. Depending on the edge or corner the dynamic sample was clamped to, this can mean that a reordering of the structured samples is required.

After the initial triangulation is set up, we start with the flipping process (Algorithm 5.1 Lines 6-18). Since the data structure stores all vertices of the quadrilateral of an edge in the EV array, the empty circumcircle interior property can efficiently checked with Equation 5.8. In case a flip is required, we flip an edge in the data structure with Algorithm A.1 in Appendix A.

After the triangulation is Delaunay, the cubature in Equation 5.6 can be evaluated by iterating over the FV array.

We use the scattered sample set consisting of the corners and one dynamic sample because adding another sample, structured or dynamic, which is inside the area light raises the complexity significantly and introduces numerical issues we were not able to handle. The new complexity arises due to the several possible relations that can occur. For example, the second point can either be in an inside triangle, which means a further triangulation of that triangle, or it can lie on an inside edge, which requires splitting that edge (vertices must not lie on one line as explained in Section 5.1). Moreover, the second inside sample can overlay not only the structured corner samples, but also the first inside sample.

Issues arise due to the determination of the relations, which can yield numerical problems. For example, determining whether a dynamic sample overlaps with a corner sample can

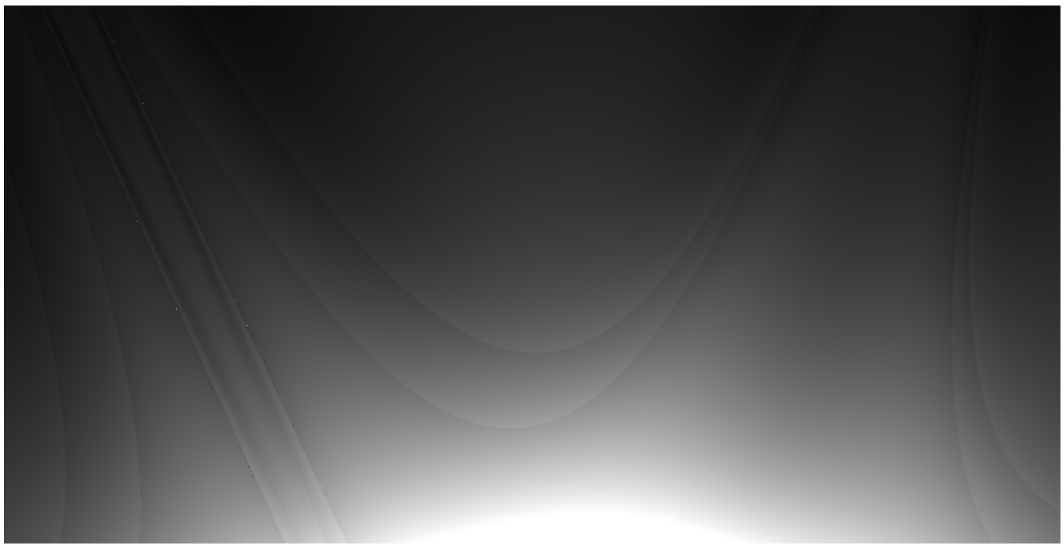


Figure 5.9: Patterns become visible when evaluating the cubature at large distances due to numerical issues.

be determined accurately with our clamping function. But deciding whether two points overlap inside the area light is prone to numerical inaccuracies. This is also the case when deciding whether the second inside sample lies on an inside edge or not.

We tried to use more complex sample sets by using the initial triangulations from above and then dynamically adding additional samples while updating the triangulation (the required algorithms to further triangulate a triangle and split an edge are given in Appendix A by Algorithm A.4 and A.5). Besides the additional computation costs, which reflected negatively in the frame rate, we were not able to achieve numerically stable results without artifacts. For this reason, we use the scattered sample set mentioned above.

### 5.2.3 Handling Numerical Issues

When the subtended solid angle of the light source becomes small, our implementation for computing the spherical excess is prone to numerical inaccuracies. The same goes for our clamping function at large distances, which still has numerical problems when the closest point is close to an edge of the light source. These issues are visible as line patterns as shown in Figure 5.9.

We address these problems by using the form factor, which can be analytically computed as presented by Baum et al. [BRW89]. Ignoring occlusion, the form factor for the energy the surface $S_j$ receives by the surface $S_i$ is given by

$$F_{S_i \to S_j} = \frac{1}{S_i} \int_{x_i \in S_i} \int_{x_j \in S_j} \frac{\cos\theta_i \cos\theta_j}{\pi \|x_i - x_j\|^2} dS_j(x_j) dS_i(x_i) \tag{5.9}$$

Figure 5.10: No patterns are visible when direct lighting is evaluated by using the analytical evaluation of the form factor as presented by Baum et al. [BRW89].

and can be approximated with

$$F_{S_i \to S_j} = \int_{x_i \in S_i} \frac{\cos\theta_i \cos\theta_j}{\pi \|x_i - x_j\|^2} dS_i(x_i) = \frac{1}{\pi} \int_{\Omega_i} (\hat{\omega}_i \cdot \hat{n}_{S_j}) d\hat{\omega}_i \tag{5.10}$$

if the distance is large compared to the area of the surface $S_j$ [BRW89]. With the method presented by Baum et al. [BRW89], Equation 5.10 can be solved analytically, which allows computing the form factor $F_{S_L \to x}$ of a light $L$ for a point $x$ efficiently and in a numerically stable way.

This technique can be utilized by transforming Equation 2.46 as follows:

$$\begin{aligned} L_d(x, \hat{\omega}_o) &= \frac{\int_{\Omega_L} f_s(x, \hat{\omega}_o, \hat{\omega}_i) L_s(x_{\hat{\omega}_i}, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x) d\hat{\omega}_i}{\frac{1}{\pi}\int_{\Omega_L} (\hat{\omega}_i \cdot \hat{n}_x) d\hat{\omega}_i} \frac{1}{\pi} \int_{\Omega_L} (\hat{\omega}_i \cdot \hat{n}_x) d\hat{\omega}_i \\ &= \pi \overline{L_{s,\Omega_L}}(x) F_{S_L \to x}, \end{aligned} \tag{5.11}$$

which results in the direct lighting being the product of the cosine weighted mean of $f_s(x, \hat{\omega}_o, \hat{\omega}_i) L_s(x_{\hat{\omega}_i}, -\hat{\omega}_i)$ and the form factor of the light source. By computing the mean with the cubature based on Delaunay triangulation and the form factor as presented by Baum et al. [BRW89], we remove the patterns visible in Figure 5.9 as shown in Figure 5.10.

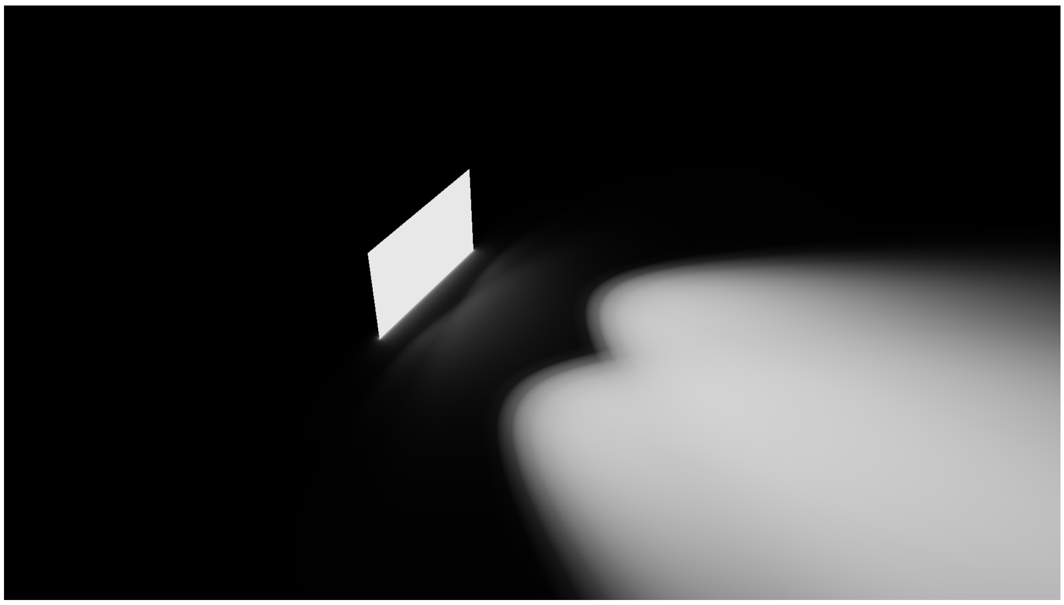


Figure 5.11: Approximating the illuminance by ARCOS (see Section 6.3), which is clipped at the horizontal plane and is normal to it. As some samples lie in the horizontal plane after the clipping process, they cannot contribute any illumination, which results in noticeable errors. The image is tone mapped as explained in Section 6.2.

### 5.2.4 Handling Clipping

To ensure that all samples are placed on the area light above the horizon, we first clip the area light at the horizontal plane[3], before the positions of the samples are computed. The positions of the structured samples are implicitly given by the corners of the clipped area light, and the closest point is ensured to be on the upper hemisphere as it is clamped to the clipped area light.

However, when the light is clipped, 2 corner samples may lie inside the horizontal plane. Depending on the orientation of the light, the closest point can also lie inside the horizontal plane. Due to the term $\hat{\omega}_i \cdot \hat{n}_x$ in the rendering equation (see Equation 2.43), these samples cannot contribute any illumination, which means that the number of contributing samples is reduced significantly. The result can be seen in Figure 5.11.

We address this issue by tilting the horizontal plane slightly for the clipping process, such that the clipped area light does not intersect with the horizontal plane. We generate a new, tilted horizontal plane by elevating the horizontal plane slightly by a value $\epsilon$ (in the case of our implementation $\epsilon = 0.1$), intersecting it with the light plane and then choosing two distinctive points $p_0$ and $p_1$ on the intersection line. The new clipping plane is then defined by these two points and the illuminated surface point. By doing the tilting as described, the introduced error has a negative correlation to the distance

[3]Assuming that the calculations are done in tangent space.

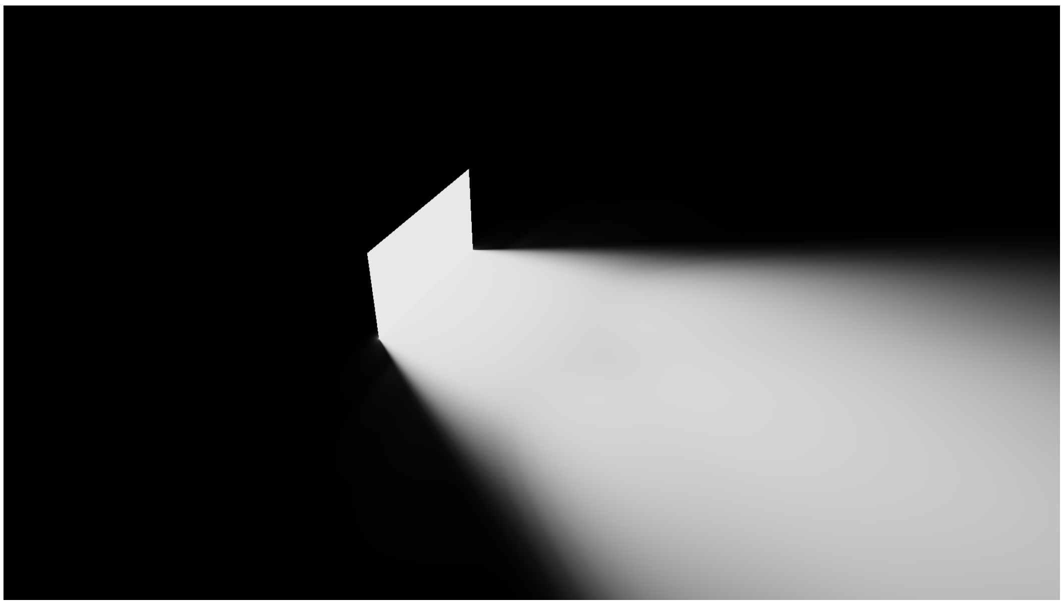


Figure 5.12: Approximating the illuminance by ARCOS (see Section 6.3), which is clipped at the slightly tilted horizontal plane. The image is tone mapped as explained in Section 6.2.

from the area light because the tilting decreases with the distance. The improved result can be seen in Figure 5.12.

CHAPTER 6

# Results and Evaluation

In this chapter we evaluate the performance of scattered sampling (see Chapter 5) with our reference implementation, which uses the corner points as structured samples and the closest point as dynamic sample. To provide a reference frame for the results, we will compare the results with Monte Carlo integration, where we use a fixed set of Poisson samples each frame to evaluate the direct lighting by a light source.

Starting in Section 6.1, we present the renderer we use to create the reference solutions. It evaluates direct lighting with Monte Carlo integration and light-source importance sampling as explained in Section 2.6. Section 6.2 explains our normalized evaluation setup used to evaluate the quality of an approximation in relation to our reference solutions. It is normalized such that we ignore the corresponding light-emitting object for each photometry, which defines the area or volume from which a luminaire emits light, and instead use a rectangular area light with the dimensions of $1 \times 1$, which illuminates a ground plane from a fixed set of predefined positions. By doing this, the quality of the approximations can be better compared for various photometric profiles with the focus on the luminous intensity distribution. In Section 6.3, the luminaires we use for the evaluation are presented. Section 6.4 shows the results of scattered sampling, which are then compared to the results of Monte Carlo integration with Poisson samples in 6.5. This chapter is concluded in Section 6.6, where the results are discussed.

## 6.1 Reference Rendering

To generate the reference solutions to which we compare our approach to, we implemented a renderer which accurately renders scenes that are illuminated by photometric area lights. The renderer only handles the diffuse light transport and unblocked direct lighting, which means that it ignores any occlusions between a surface point and a light source.

The reference images are rendered by computing the direct lighting given by a photometric area light for each surface point in a scene with Quasi-Monte Carlo integration (see Section 2.5.5). Because this renderer was used during the research process, we implemented it iteratively. Instead of computing the illumination once with a large number of samples, we use a small number of samples each frame and accumulate the results in a buffer. This enables to see rough approximations early, which speeds up the research process. Over time, the image in the accumulation buffer converges to the correct solution. When the view matrix is modified, said accumulation buffer is cleared and the process starts anew.

The renderer executes two passes each frame. The first pass, explained in Section 6.1.1, is the sampling pass, where the direct lighting is evaluated with a small number of samples. The second pass adds the result of the first pass to an accumulation buffer, which equals a rendered image where all previously used samples are taken into account. Section 6.1.2 explains the second pass in further detail.

### 6.1.1 1st Pass: Sampling

This pass is the sampling pass, where the direct lighting by a photometric area light is evaluated with a small number of samples (e.g., $N = 8$) per area light. To reduce the error and increase the convergence speed, Quasi-Monte Carlo integration is used (see Section 2.5.5), where the Halton sequence is used to create low-discrepancy samples. The renderer utilizes light-source importance sampling by generating samples only on the light source with the PDF $p(x') = 1/S_L$.

The first step is to create $2N$ new Halton samples $\xi_1$ and $\xi_2$, which are loaded onto the GPU with a uniform buffer. Because all fragments use the same random numbers, patterns would become visible. To counter this effect, each fragment modifies the random variables with the jitter function

$$J(\xi, \tau) = (\xi + \tau) - \lfloor \xi + \tau \rfloor \tag{6.1}$$

where $\tau$ is a unique jitter value per fragment per random number.

Our renderer uses a hash function for floating point numbers presented by Sharpe [Bri11] to compute a unique $\tau$ per fragment.

The jittered random variables $J(\xi_1, \tau_1)$ and $J(\xi_2, \tau_2)$ are then used to generate a sampling direction $\hat{\omega}_i$. The directions towards the subtended solid angle of the light source are the only directions which return radiance. Therefore, we use light-source importance sampling to generate the sampling directions. Using uniform sampling or BRDF importance sampling [1] would create many samples that would miss the light and therefore return 0 radiance.

[1]Importance sampling the BRDF would be importance sampling the cosine falloff as explained in Section **??**, because this thesis is only concerned with the diffuse light transport.

We implemented light-source importance sampling as suggested by Shirley et al. [SWZ96], where samples are generated directly on the light source with the PDF $p(x') = 1/S_L$. and evaluated using the estimator shown in Equation 2.49 without the geometry term, as this thesis is only concerned with the unblocked light transport without occlusion. More sophisticated solutions like sampling the subtended solid angle as proposed by Ureña et al. [UnFK13] would result in a slightly faster convergence speed, but would also lead to problems as some light sources emit light in the plane of the area light, while the solid angle is 0 for all points in the plane. If directions should be sampled for a point lying in the plane of a planar area light, the subtended solid angle of the light would be 0. Using the approach by Ureña et al. would require to sample the nonexistent subtended solid angle, which again would result in visible artifacts due to numerical problems.

We therefore decided to generate the samples uniformly on the bounding rectangle of the light, which is defined by a corner point $x_0$ and two vectors $v_1$ and $v_2$:

$$x' = x_0 + \xi_1 * v_1 + \xi_2 * v_2. \tag{6.2}$$

If the generated sample lies inside the light source, it is used to sample the light. Otherwise the sample returns the constant value 0. After all samples are evaluated, the estimate can be computed with

$$L_{d,N} = \frac{S_L}{N} \sum_{i=1}^{N} \frac{f_s(x, \hat{\omega}_o, \hat{\omega}_i) L_e(x_{\hat{\omega}_i}, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x)(-\hat{\omega}_i \cdot \hat{n}_{x'})}{\|x_{\hat{\omega}_i} - x\|^2}. \tag{6.3}$$

### 6.1.2 2nd Pass: Blending

Since each iterative rendering uses a small number of samples, all the iterative renders have to be combined into one image by averaging them such that the rendering converges. This is achieved by using a second framebuffer where the iterative renderings are accumulated with blending. Assume that the current accumulated estimate is given by $L_{d,k*N}$, where k is the number of accumulated renderings:

$$L_{d,k*N} = \frac{A}{kN} \sum_{i=1}^{kN} \frac{f_s(x, \hat{\omega}_o, \hat{\omega}_i) L_e(x_{\hat{\omega}_i}, -\hat{\omega}_i)(\hat{\omega}_i \cdot \hat{n}_x)(-\hat{\omega}_i \cdot \hat{n}_{x'})}{\|x_{\hat{\omega}_i} - x\|^2} = \frac{1}{kN} \sum_{i=1}^{kN} L_i. \tag{6.4}$$

This estimate can be updated with $L_{d,N}$ (estimate of a sampling pass, see Equation 6.3) with the following procedure:

$$
\begin{aligned}
L_{d,(k+1)*N} &= \frac{k}{k+1} L_{d,k*N} + \frac{1}{k+1} L_{d,N} \\
&= \frac{k}{k+1} \left( \frac{1}{kN} \sum_{i=1}^{kN} L_i \right) + \frac{1}{k+1} \left( \frac{1}{N} \sum_{i=kN+1}^{(k+1)N} L_i \right) \\
&= \frac{1}{(k+1)N} \sum_{i=1}^{(k+1)N} L_i .
\end{aligned}
\tag{6.5}
$$

To achieve this with blending, the estimator $L_{d,N}$ is multiplied with $1/(k+1)$ before it is written into the iterative buffer. Moreover, the term $k/(k+1)$ is written into the alpha channel of the iterative buffer. The blending is then achieved with

$$C = 1.0 * S + S.A * D \tag{6.6}$$

where $S$ is the source color taken from the iterative buffer with the alpha value $S.A$ and $D$ is the destination color taken from the accumulation buffer.

Example results are shown in Figure 6.1 and Figure 6.2. Both images were rendered with approximately 20.000 samples.

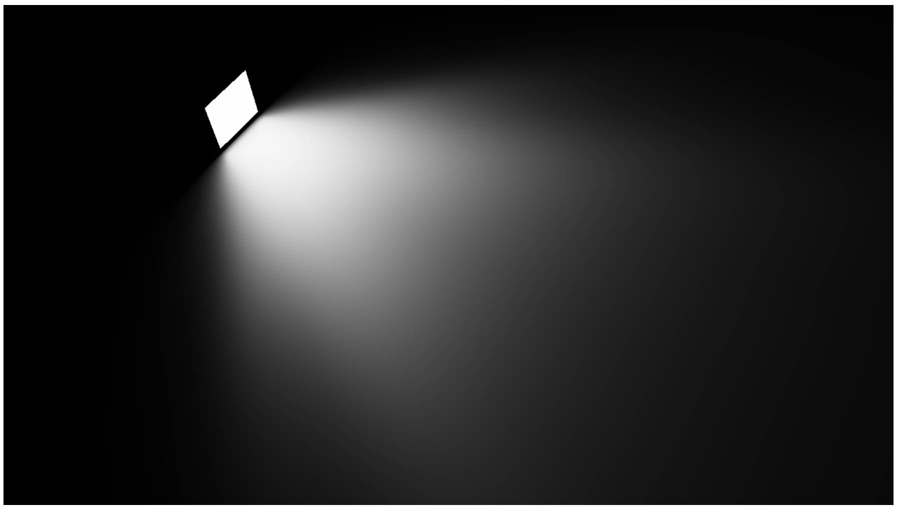


Figure 6.1: Plane illuminated by MIREL (see Section 6.3). Image generated with approximately 20.000. The image is tone mapped as explained in Section 6.2.

## 6.2 Evaluation Setup

To evaluate the performance of our technique, we implemented an evaluation process that tests the technique with various photometric profiles under normalized conditions.

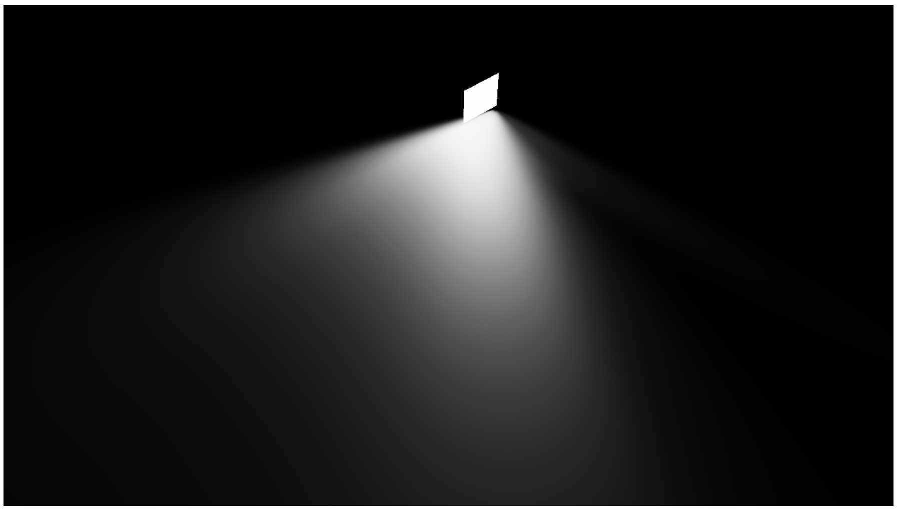


Figure 6.2: Plane illuminated by INTRO (see Section 6.3). Image generated with approximately 20.000. The image is tone mapped as explained in Section 6.2.

Normalized means in this context that we ignore the physical dimensions given by the photometric profile for the luminaire and instead use the same light-emitting object for all photometric profiles. While this means that the resulting setup may not reflect the real world (e.g., the emission profile of a spot light on a large, rectangular area light), it makes the results better comparable, as the focus shifts to the luminous intensity distribution.

The scene consists of a ground plane $g = \{p \mid n_g \cdot p = 0, n_g = (0,0,1)\}$ and a planar, rectangular area light with a dimension of $1 \times 1$ units and the normal $\hat{n}_L$. The light illuminates the ground plane from above. An illustration is shown in Figure 6.3. Starting with $n_L = (0,1,0)$, the light is rotated downwards during the evaluation process until $n_L = (0,0,-1)$. At fixed rotation steps $(\cos^{-1}(n_L \cdot n_g) = \pi/2, 5\pi/8, 3\pi/4, 7\pi/8, \pi)$, an orthographic camera takes an image of the ground plane from above. An example is shown in Figure 6.4. The light source is not rendered to avoid occlusions of the illuminated surface. These images are then used to evaluate the quality of the technique. The rotation process enables us to observe the illumination for various spatial relations between illuminated surface points and the area light.

The evaluation is executed for vertical light translations of 0.1 units, 1.1 units, 3.1 units and 5.1 units. We do not evaluate the light directly on the ground, as this is a case which hardly occurs in the real world.

The evaluation process consists of two parts, where the first part is the rendering part, which we implemented with the Aardvark Platform developed by the VRVis. Images rendered with our implementation are HDR images with a resolution of $2048 \times 2048$ pixels. The second part, which evaluates these images, is implemented in Matlab. Our scripts return the following output per rendering technique per photometry:

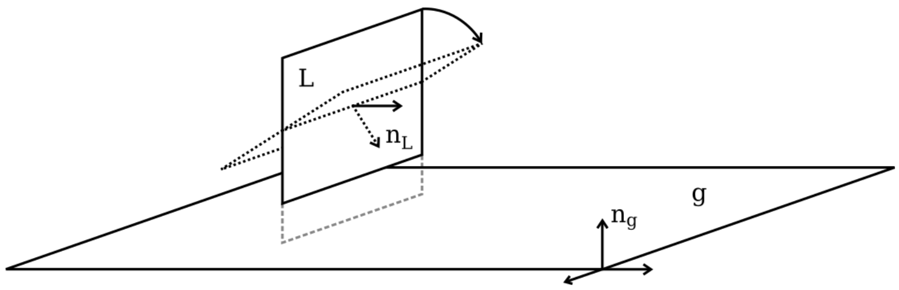


Figure 6.3: The test scene consists of a rectangular, planar area light which is rotated towards the ground plane with an infinite width.

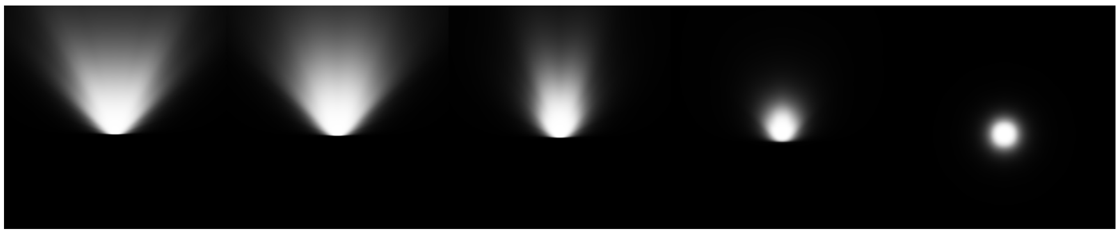


Figure 6.4: The ground plane illuminated by MIREL (see Section 6.3), which is rotated downwards with 5 iteration steps. This is a combined image of 5 images, where each is taken with an orthographic camera from above looking downwards.

- Normalized Mean Squared Error (NMSE)
- Upper Error Bound (UEB)
- Lower Error Bound (LEB)
- Tone-mapped Images
- Error Images

The normalized mean squared error is computed as shown in Equation 6.7, where $N$ is the numbers of pixels, $x_R$ and $x_A$ are corresponding pixels in the reference rendering and approximation rendering and $\bar{R}$ and $\bar{A}$ are the means of reference and approximation renderings.

$$NMSE = \frac{1}{N}\sum_{i}^{N}\frac{(x_{R,i} - x_{A,i})^2}{\bar{R}\bar{A}} \tag{6.7}$$

We use Equation 6.8 to tonemap the images.

$$L_d(x,y) = \frac{L(x,y)}{1 + L(x,y)} \tag{6.8}$$

$L_d(x, y)$ is the tone-mapped value of the pixel at position $(x, y)$ and $L(x, y) = (a/\bar{L}_w) * L_w(x, y)$, where $L_w(x, y)$ is the value of pixel $(x, y)$, $a$ is a constant and $\bar{L}_w$ is the log average luminance of the picture [RSSF02]. Since the average illuminance on the ground plane varies for the same photometry due to approximation errors, which means a variance in the average luminance $\bar{L}_w$, we decided to replace the scale value $(a/\bar{L}_w)$ with a fixed value (= 10.0) for all images so that the different approximations are better comparable.

The errors are computed by comparing the results with our reference solution (see Section 6.1). In addition to the error measures, error images are generated that visualize the spatial deviations of the approximations relative to the reference. An example is shown in Figure 6.5. Orange areas are too bright, blue areas are too dark. The saturation encodes the magnitude of the error, where white means no error. The magnitude of the error for fully saturated areas is given by the respective upper and lower error bounds.

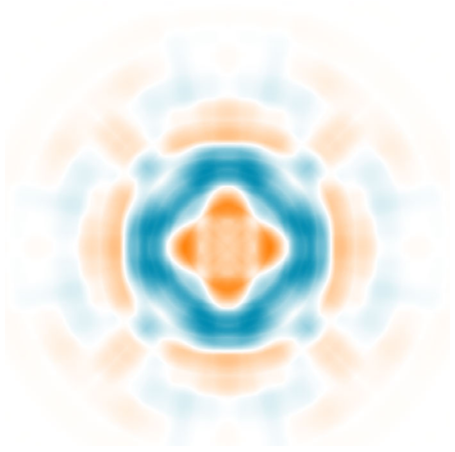


Figure 6.5: Example error image. Orange areas are too bright, blue areas are too dark, where the magnitude of the error is given by the saturation. The magnitudes of fully saturated areas are given by the respective upper and lower error bounds.

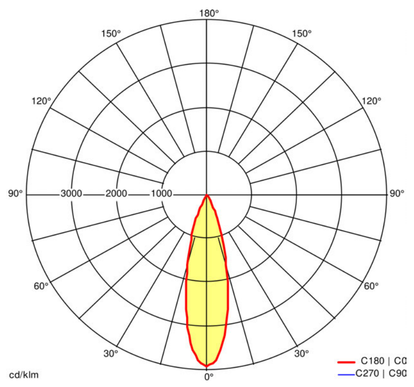
## 6.3 Light Selection

To evaluate the performance with different photometries, we selected a representative set of luminaires from the luminaire catalogue of Zumtobel [Zum18] with a high variance regarding the emission characteristics. On the following pages, each luminaire is presented with the following information:

- **Name**: A shortened name which we use to refer to the luminaire in this thesis.
- **Product Name**: The full name of the luminaire with which it can be found in the Zumtobel catalogue [Zum18].
- **Luminous Intensity Distribution**: An illustration of the luminous intensity distribution (see Section 2.2) taken from [Zum18], which shows the emission characteristic in a polar plot.
- **Reference Table**: A table of references images for the luminaire generated with our evaluation setup (see Section 6.2), which shows the illumination profile of the luminaire for various relations to the ground plane.

**ARCOS**

ARC3 1/28W LED927-65 LDO 3CD SP WHM



Height — $\cos^{-1}(n_L \cdot n_g)$

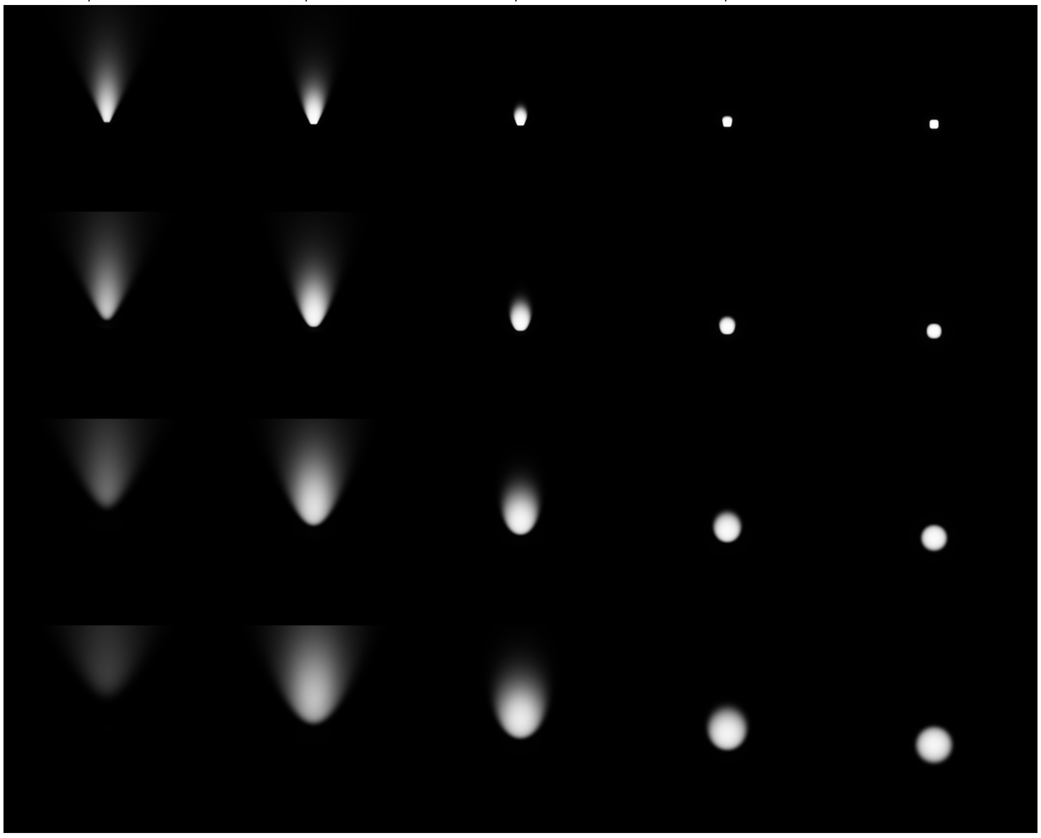
$\pi/2$ | $5\pi/8$ | $3\pi/4$ | $7\pi/8$ | $\pi$

0.1

1.1

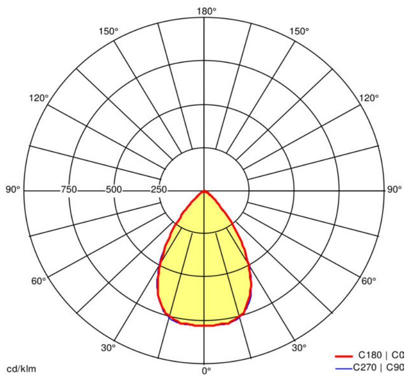
3.1

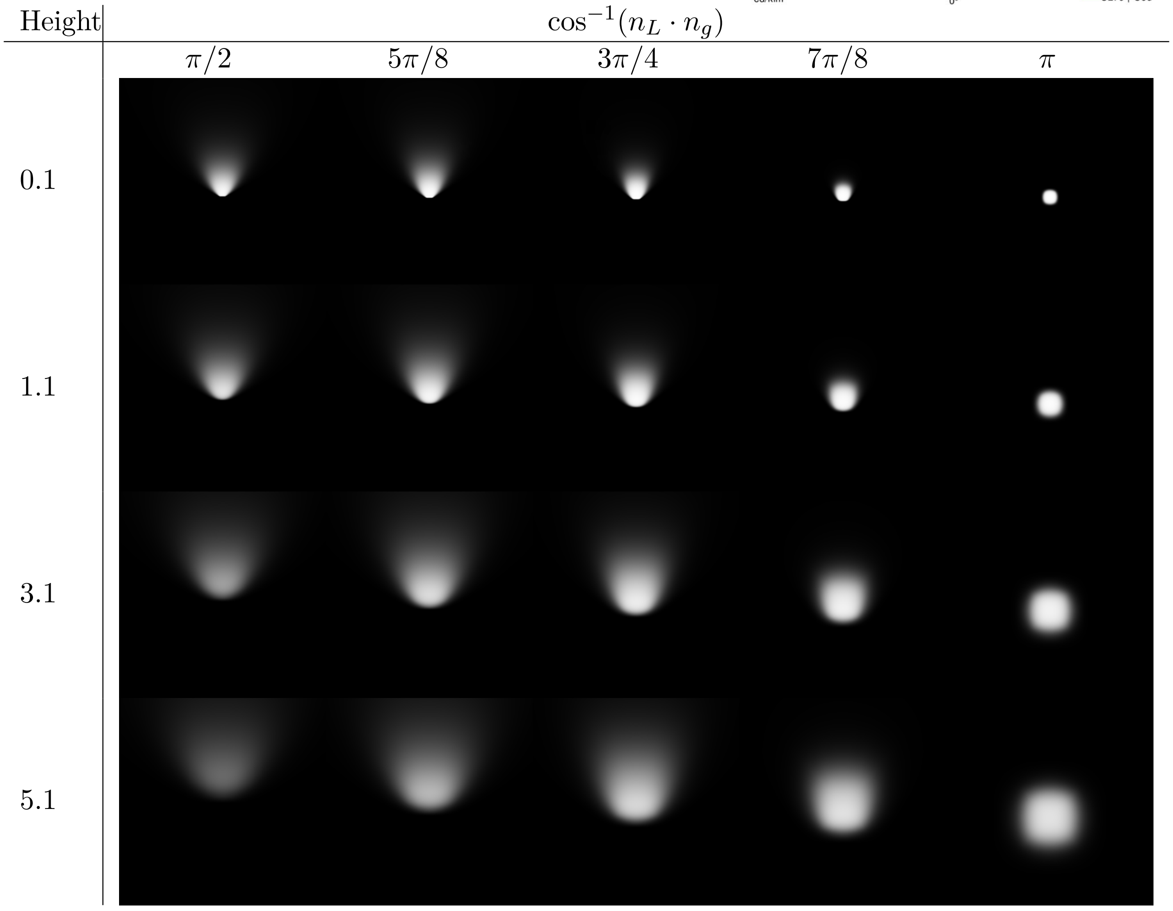
5.1



**PANOS**

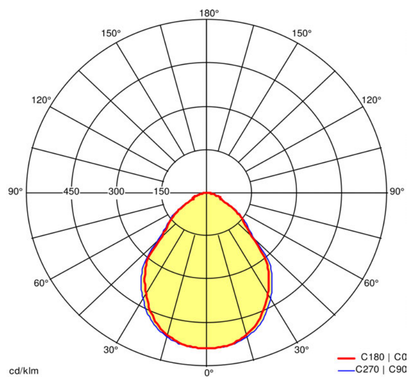
PANOS INF Q140HF 22W LED927-65 LDE WH

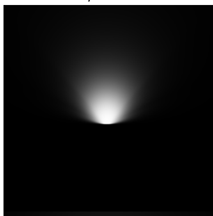
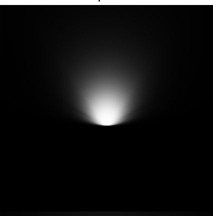
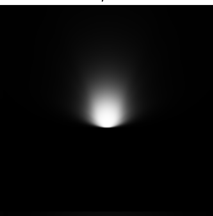
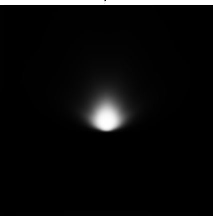
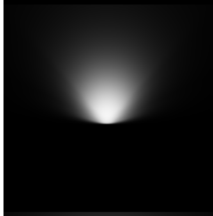
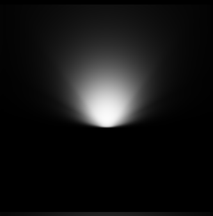
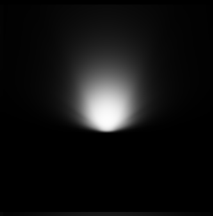
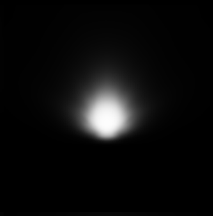
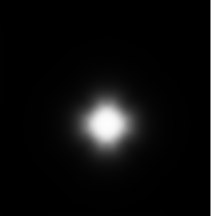
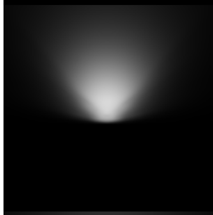
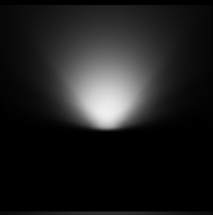
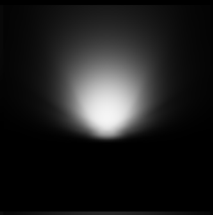
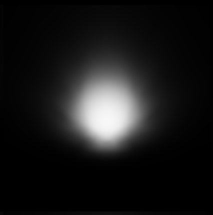
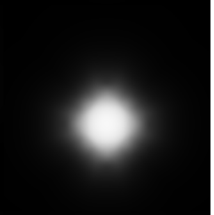
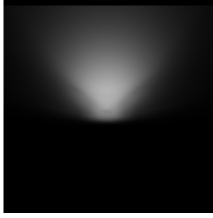
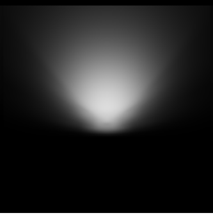
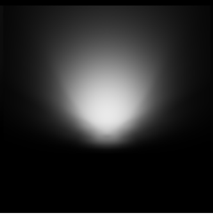
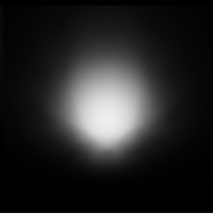
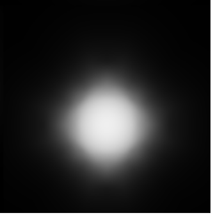




**SLOTLIGHT**

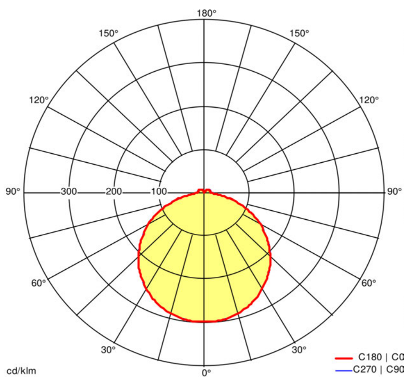
SLOIN A SL IP54 LED2800-840 L2040
PCO

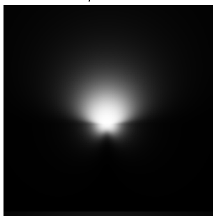
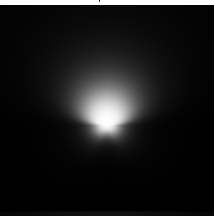
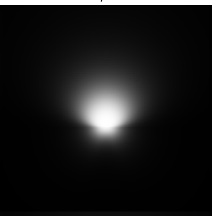
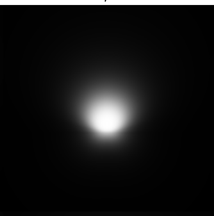
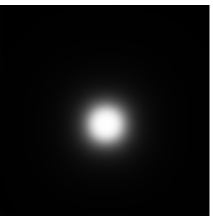
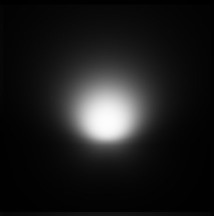
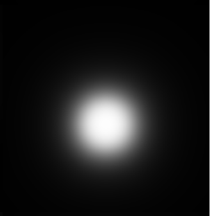
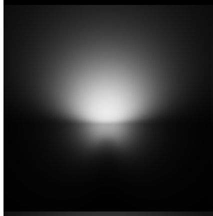
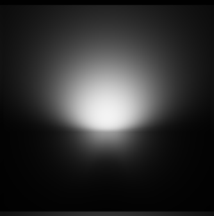
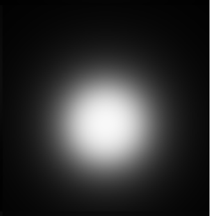
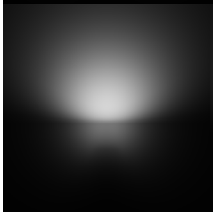
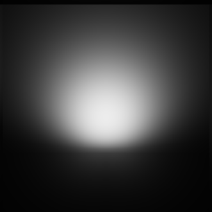
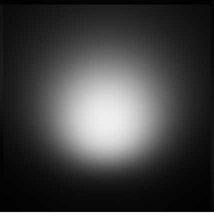
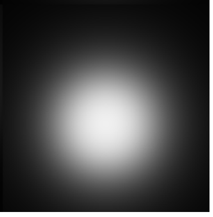


| Height | $\cos^{-1}(n_L \cdot n_g)$ | | | | |
|---|---|---|---|---|---|
| | $\pi/2$ | $5\pi/8$ | $3\pi/4$ | $7\pi/8$ | $\pi$ |
| 0.1 |  |  |  |  | |
| 1.1 |  |  |  |  |  |
| 3.1 |  |  |  |  |  |
| 5.1 |  |  |  |  |  |

**PERLUCE**

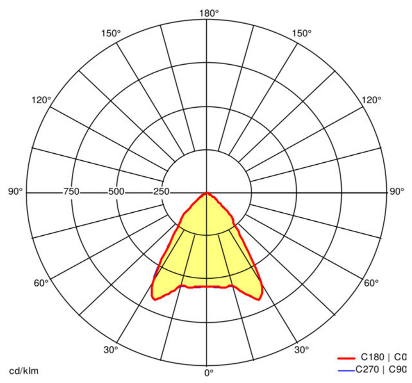
PERLUCE O LED5200-840 Q620 LDE
IP50 WH

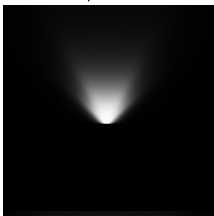
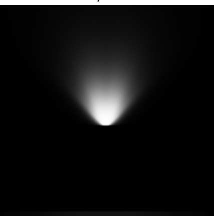
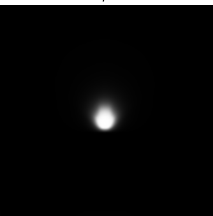
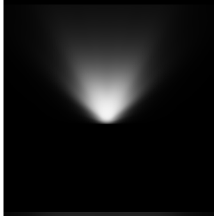
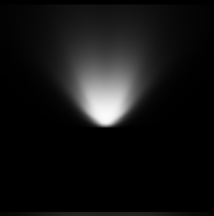
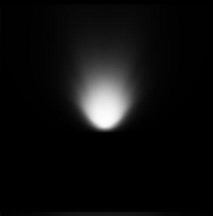
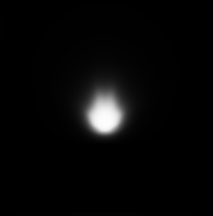
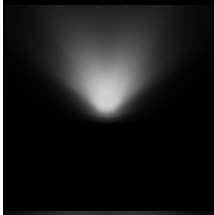
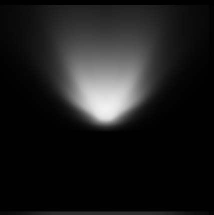
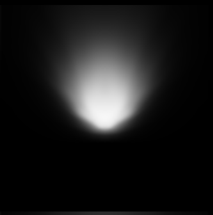
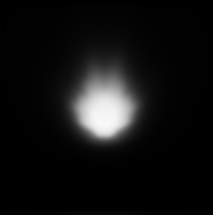
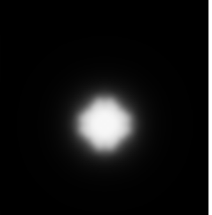
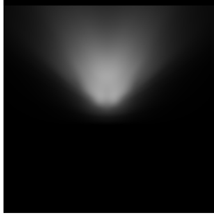
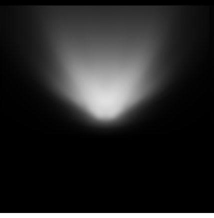
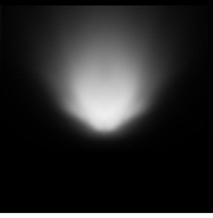
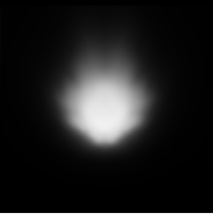
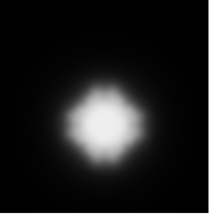


| Height | $\cos^{-1}(n_L \cdot n_g)$ | | | | |
|---|---|---|---|---|---|
| | $\pi/2$ | $5\pi/8$ | $3\pi/4$ | $7\pi/8$ | $\pi$ |
| 0.1 |  |  |  |  |  |
| 1.1 | | | |  |  |
| 3.1 |  |  | | |  |
| 5.1 |  | |  |  |  |

**MIREL**

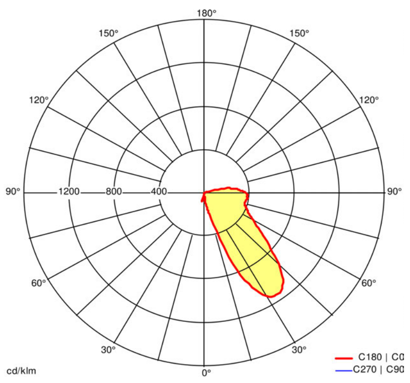
MIRL NIV LED2800-830 M625Q LDO



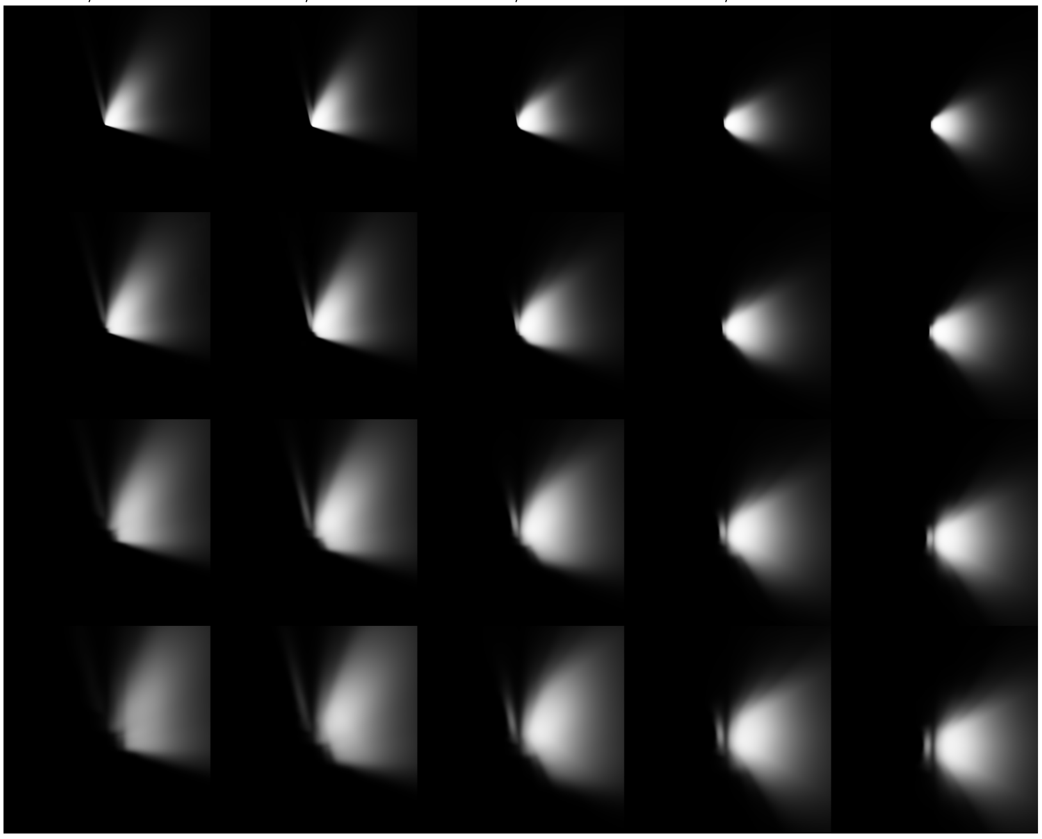
| Height | $\cos^{-1}(n_L \cdot n_g)$ | | | | |
|---|---|---|---|---|---|
| | $\pi/2$ | $5\pi/8$ | $3\pi/4$ | $7\pi/8$ | $\pi$ |
| 0.1 |  |  | |  | |
| 1.1 |  |  |  |  | |
| 3.1 |  |  |  |  |  |
| 5.1 |  |  |  |  |  |

**INTRO**

INT LED2800-940 LC 3CV BK

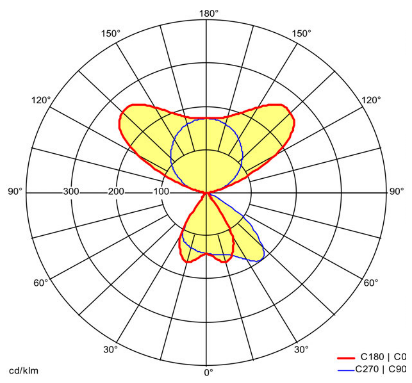


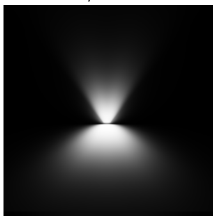
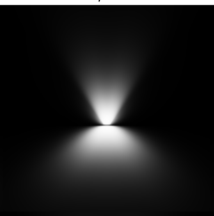
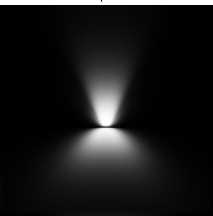
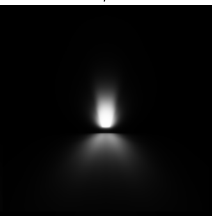
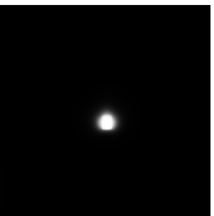
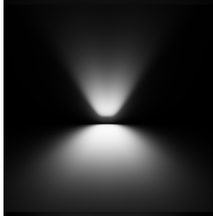
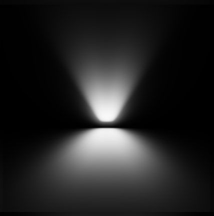
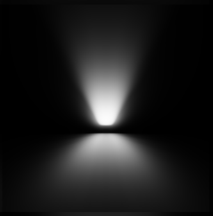
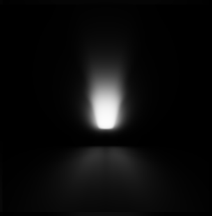
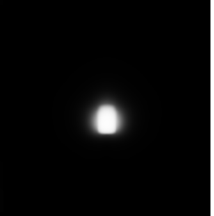
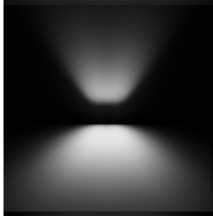
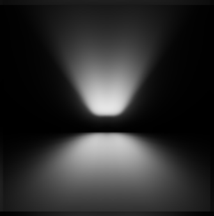
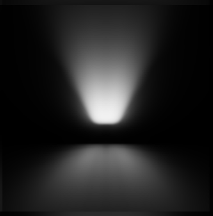
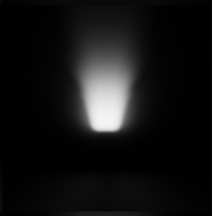
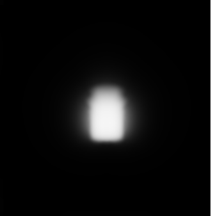
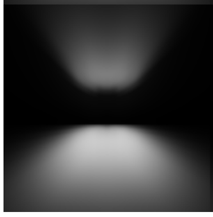
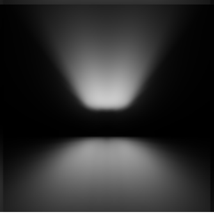
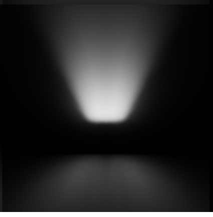
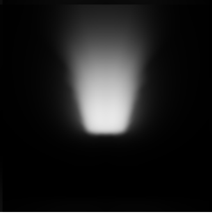
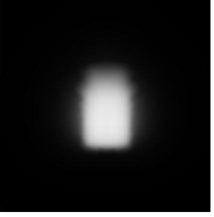
| Height | $\cos^{-1}(n_L \cdot n_g)$ | | | | |
|---|---|---|---|---|---|
| | $\pi/2$ | $5\pi/8$ | $3\pi/4$ | $7\pi/8$ | $\pi$ |
| 0.1 | | | | | |
| 1.1 | | | | | |
| 3.1 | | | | | |
| 5.1 | | | | | |



**LINETIK**

LINETIK-S D/I LED8000-830 SC WH SR2 IL



| Height | $\cos^{-1}(n_L \cdot n_g)$ | | | | |
|---|---|---|---|---|---|
| | $\pi/2$ | $5\pi/8$ | $3\pi/4$ | $7\pi/8$ | $\pi$ |
| 0.1 |  |  |  |  |  |
| 1.1 |  |  |  |  |  |
| 3.1 |  |  |  |  |  |
| 5.1 |  |  |  |  |  |

## 6.4 Results

In this section we discuss the performance and quality of scattered sampling. To evaluate the performance, we used a Nvidia GeForce GTX 1080 and rendered with a resolution of $1920 \times 1080$. The measured performance values are presented in Table 6.1. The table shows the FPS for scattered sampling with flipping and without flipping. We implemented and measured the latter because we did not observe flipping in our setup (rectangular area light with closest point as dynamic sample). Discarding flipping allowed further optimizations.

| | Inside | Edge | Corner | Average |
|---|---|---|---|---|
| SCT w/ flipping | 240 | 260 | 310 | 300 |
| SCT w/o flipping | 370 | 390 | 410 | 400 |

Table 6.1: Measured FPS for scattered sampling (SCT) with and without flipping. *Inside*, *edge* and *corner* describe the relation of the clamped closest point to the light source, i.e., whether it is clamped to a corner, an edge or not clamped at all (inside). The values were measured such that the relation of the clamped closest point to the light source was the same for all fragments. The average value is derived by placing the camera above the illuminated plane directed downwards, as in our evaluation setup, such that all cases occur.

The computational cost of the technique for a fragment depends on the relation of the clamped closest point (CCP) to the polygon of the light source, as this defines the number of edges which have to be checked for flipping and the number of triangles which have to be evaluated for the cubature. The CCP can either be inside the polygon, sit on an edge or be in a corner (see Section 5.2.2).

The performance when navigating the scene naturally, i.e., all cases occur on the screen, depends on several factors: the size of the light source, its position and orientation in the scene and the position of the camera. These factors define the number of fragments for each case and hence the resulting performance. To measure an average value, we positioned the camera like the camera in the evaluation setup (see Section 6.2) and rotated the light directly downward such that all cases occur.

The error measurements for scattered sampling for all luminaires presented in Section 6.3 are given in Appendix B. An example table is given by Table 6.2. Each table shows tone-mapped renderings for various rotation degrees ($\cos^{-1}(n_L \cdot n_g) = \{\pi/2, 5\pi/8, 3\pi/4, 7\pi/8, \pi\}$) with corresponding reference and error images (see Section 6.2). Additionally to the tables in Appendix B, the error images are shown enlarged in Appendix C.

In the following we analyze the quality results for each luminaire. To describe the results, we define areas on the illuminated ground plane based on the position of the unclamped closest point relative to the area $S_L$ on the light plane of the area light $L$:

- The *Outside Area* contains all points on the ground plane where the corresponding closest point lies outside of $S_L$.
- The *Inside Area* contains all points where the closest point lies inside $S_L$.
- The *Border Area* contains all points where the closest point lies on the border of $S_L$. We define this area in a fuzzy way, meaning that it also contains points of the outside and inside area, where the closest point is close to the border.
- *Corner Areas* are subsets of the border area, and contain all points where the closest point is near a corner of $S_L$.

The error in the outside area is expected to become smaller with increasing distance to the area light, because the error introduced when using a far-field photometric profile at a different distance than the profile was generated from becomes smaller with increasing distance to the light source (see Section 2.3). Therefore, the following analysis will mainly focus on the border area and the inside area:

**ARCOS** (Tables B.1, B.2, B.3 and B.4) is a directional luminaire with a maximal luminous intensity of $3921 cd$, which is relatively bright compared to the other selected luminaires. Visually, the approximation is without artifacts and hardly distinguishable from the reference solution. The main difference can be seen at the heights 0.1, 1.1 and 3.1, where the border area is more rectangular than the reference solution. The reason can be seen in the error images and plots, where it shows that the border area is generally too bright.

The high illuminance is caused by using too few samples in combination with a strong, directional luminaire. At the border area, where the dynamic sample comes close to the static corner samples, the illumination of the border area is computed with 2 to 3 samples[2] with a large luminance and and 2 to 3 samples with no luminance. Since the 0 luminance samples can hardly outweigh the high luminance samples, a high illuminance is computed. As the light is rotated downwards at height 0.1, the error images show that the under-illumination on the inside turns to an over-illumination, as the closest point for each surface point in the inside area starts to return a high illuminance.

**PANOS** (Tables B.5, B.6, B.7 and B.8) is, in contrast to ARCOS, generally too dark in the inside area. This is because of the more diffuse emission pattern, where a larger area of $S_L$ contributes to the illumination. Because the number of samples is too small, the illumination is too low. There are not enough samples around the closest point that can contribute luminance, as it is the case in the border and corner areas.

The evaluation data shows that the NMSE is smaller compared to ARCOS. The strong difference, however, is mainly due to the smaller maximal luminous intensity. However,

[2] 2 if in the border area but not a corner area, 3 otherwise.

comparing the visual quality, the approximation is also hardly distinguishable from the reference solution.

**SLOTLIGHT** (Tables B.9, B.10, B.11 and B.12) is even more diffuse than PANOS and also has the same issues. The too low illumination can also be seen in the tone-mapped renderings, especially noticeable at the height 5.1. Again, this is due to the small number of samples. The pattern and form of the illuminated area, however, is accurate and without artifacts.

**PERLUCE** (Tables B.13, B.14, B.15 and B.16) is approximated especially well. The NMSE is relatively small compared to the other luminaires. The luminous intensity distribution is almost uniform, which means that it can be considered a diffuse area light. Since scattered sampling is based on structured sampling, and structured sampling is developed to approximate diffuse area lights accurately, the small error is expected.

The main error is introduced by the non-diffuse luminous intensity distribution at the back of the area light. This error, however, is small and not noticeable in the tone-mapped images.

**MIREL** (Tables B.17, B.18, B.19 and B.20) is similar to PANOS and SLOTLIGHT with regards to its luminous intensity distribution. Therefore, the approximation results are similar. The illumination is generally too dark in the border areas.

has a emission profile where the most light is not emitted forwards. At height 0.1, the approximation results are similar to PANOS and SLOTLIGHT. But starting at height 1.1, an over-illumination appears in the center. This is caused by the structured samples in the corners. Because MIREL emits the most light slightly sidewards, the central points of the inside area receive a lot of illumination from each of the four corner points.

**INTRO** (Tables B.21, B.22, B.23 and B.24) is an asymmetric luminaire, where an artifact is visible where the plane of the area light intersects the ground plane. This artifact shows itself as a fine dark line, which appears due to a contradiction between the luminous intensity distribution and the used cubature technique. While the first one states there is a luminous flux in the direction of $\theta = \pi/2$, the cubature technique based on the Delaunay triangulation computes the weights based on the spherical excess of spherical triangles, which is 0 for all points in the light plane (meaning that the computed illuminance is 0). Because of this contradiction, the darkened line appears due to numerical issues where the plane of the area light intersects the ground plane. Moreover, the closest point appears to be not the best choice for this luminaire. It can be seen that the border area is not smooth but instead has light bumps. These artifacts are shown in Figure 6.6.

Another noticeable difference is the small spike away from the large illuminated area. While it is a single spike in the reference solution, the small number of samples leads to

Figure 6.6: The figure shows two of the artifacts generated by SCT when approximating INTRO. To the left, light bumps are visible instead of a smooth border area which is caused by using not enough samples. The black dotted line is an artifact resulting from numerical issues of SCT, as INTRO emits light inside the light plane where the subtended solid angle of the light source is 0.

an appearance of two spikes in the approximation. This is especially noticeable for the rotation step $\{5\pi/8, 3\pi/4, 7\pi/8\}$.

Besides these three artifacts, the approximation is close to the reference solution with regards of the illumination with a relatively small NMSE.

**LINETIK-S** (Tables B.25, B.26, B.27 and B.28) has the most complex luminous intensity distribution of all presented luminaires. Yet, the approximation with only 5 samples is close to the reference solution. There are no artifacts and hardly any visual differences.

Otherwise, it has the same problems as PANOS and SLOTLIGHT. Due to the small number of samples, contributions from a larger area of $S_L$ are missing, which leads to a too low illumination.

While the error images show differences in certain areas for various luminaires, it can be seen that differences can hardly be seen in the tone-mapped images. Apart from artifacts, as in the case of INTRO, scattered sampling approximates direct lighting by photometric area lights well. The approximations are visually plausible and match the illumination profiles of the reference solution such that differences can hardly be seen in the tone-mapped images. While scattered sampling is not suitable for exact evaluations of photometric area lights due to the errors in the inside area, it shows to be a good technique to provide fast and visually plausible approximations of them.

**PANOS**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 5.1 | 0.0026777 | 0.026367 | -0.13672 |

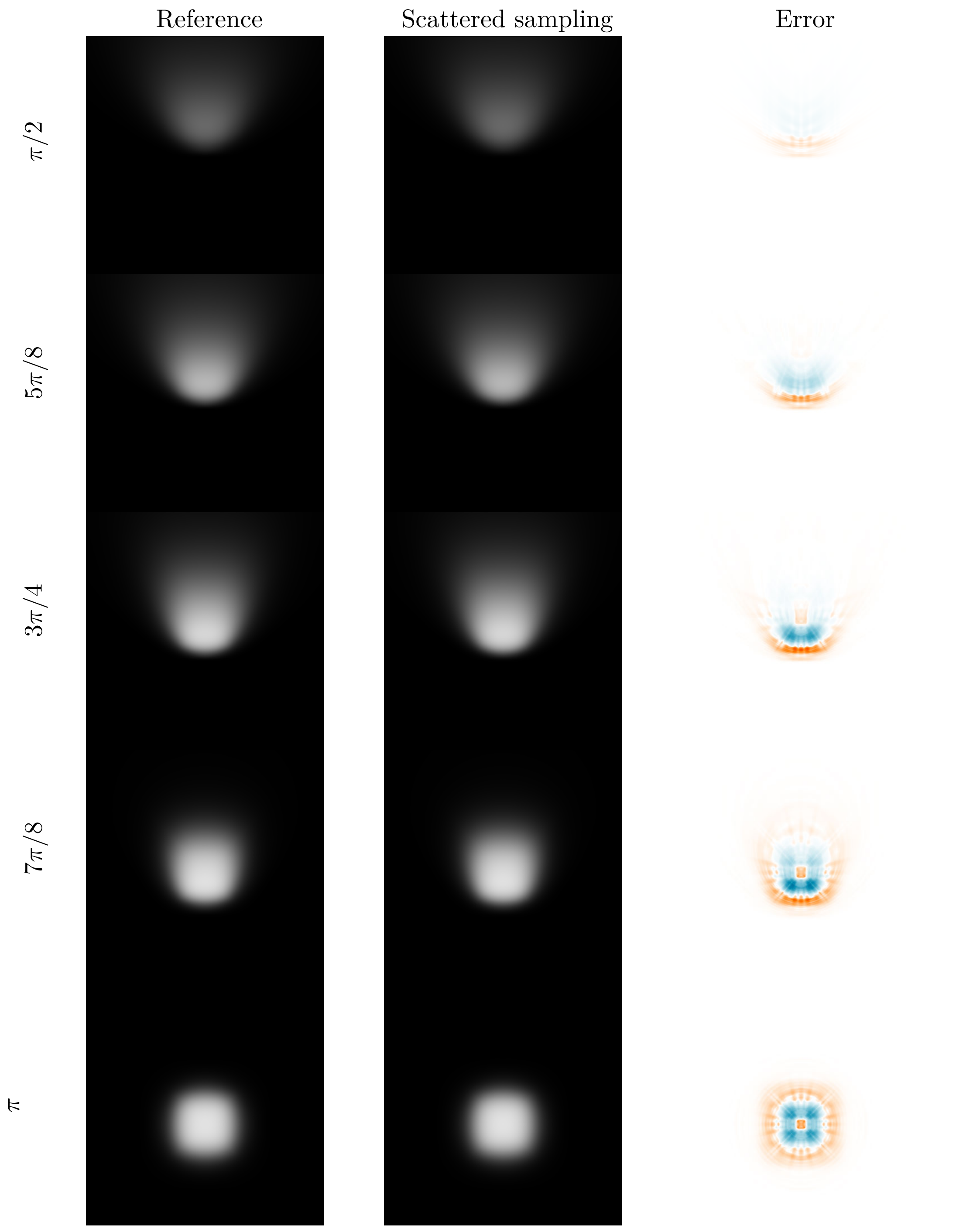


Table 6.2: Table showing the results for the PANOS at height 5.1. The table shows the normalized mean squared error (NMSE), the upper and the lower error bounds (UEB and LEB).

## 6.5 Comparison to Monte Carlo Integration

To provide a reference frame for the quality results of scattered sampling in Section 6.4, we compare them with Monte Carlo integration with light-source importance sampling as done by our reference renderer (see Section 6.1). However, instead of creating the images iteratively with a small number of Halton samples each frame, we use a constant set of Poisson samples and compute the direct lighting each frame with the same samples. The number of Poisson samples is chosen such that the performance of the reference renderer matches our scattered sampling reference implementation. To match scattered sampling with flipping (~300FPS), we use 40 Poisson samples. To match scattered sampling without flipping (~400FPS), we use 24 Poisson samples. The rendering performance of the reference renderer remains constant and independent of the scene and camera position because the same constant sample set is used for all fragments.

The plots in the Figures 6.7, 6.8, 6.9 and 6.10 show the NMSEs for each light and each technique for various heights, where each entry is the NMSE over all rotation steps. Scatterd sampling is abbreviated with *SCT* and Monte Carlo integration with light-source importance sampling with Poisson samples with *MCI-N*, where N denotes the number of Poisson samples. Scattered sampling without flipping will not be addressed separately, as we observed no flipping with our reference implementation, which means that the error of scattered sampling without flipping is equal to the error of scattered sampling with flipping.

The plots show that MCI-24 and MCI-40 generally perform better for all heights regarding the NMSE. This is expected as MCI-24 and MCI-40 use more samples. The exception is PERLUCE, an almost diffuse luminaire, where SCT always returns the smallest NMSE. The corners used with closest point show to be a good choice for a diffuse light source.

While MCI-24 and MCI-40 perform better in terms of NMSE than SCT (excluding PERLUCE), SCT performs better regarding artifacts at close distances. Artifacts become visible with MCI-24 and MCI-40 when the light source is close. An example for these artifacts, we refer to them as light bumps, is shown in Figure 6.11. Although MCI-24 and MCI-40 have a small NMSE, their approximations are visually not plausible due to the noticeable bumps. SCT, on the other hand, creates an illuminated area which is similar to the reference solution because of the dynamic sample.

The problem with light bumps is given at short distances and occurs due to the fixed sample set used by MCI-24 and MCI-40. When the light source comes close, the subtended solid angle of the light source increases quadratically and the density of the samples in the subtended solid angle decreases. As a result, some illuminated points receive less illumination and some receive more illumination depending on their relative position to the light source.

The dynamic sample used by SCT counters this efficiently. The closest point defines a position which returns a high illumination for luminaires with a Phong-like emission where the main emission direction is forward. For each illuminated point the corresponding

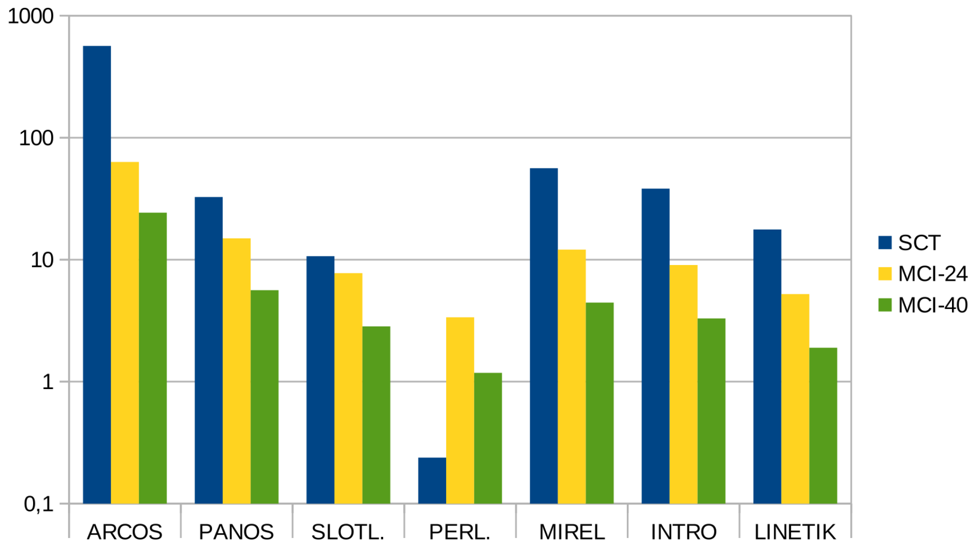


Figure 6.7: Normalized mean squared error of various techniques over all rotation steps with the light being 0.1 units above the ground.

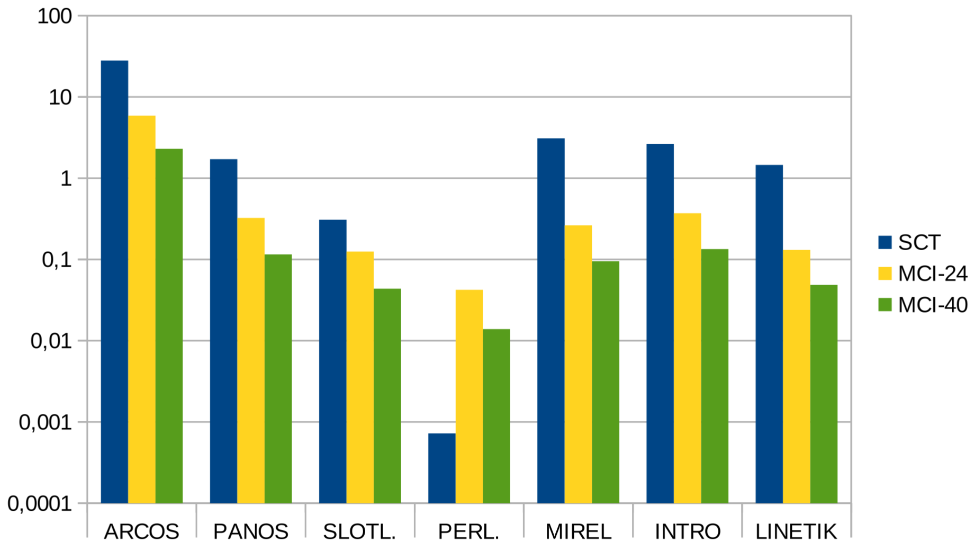


Figure 6.8: Normalized mean squared error of various techniques over all rotation steps with the light being 1.1 units above the ground.

closest point is used, which means that at least one sample exists for each point which returns a significant amount of illuminance, even if the distance between the samples becomes large.

How strongly these light bumps appear depends on several factors: the luminous intensity

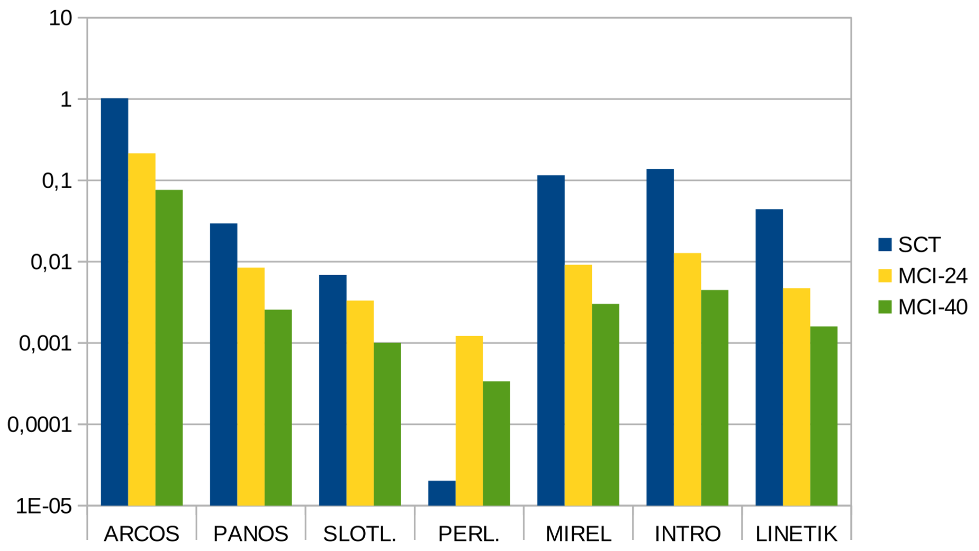


Figure 6.9: Normalized mean squared error of various techniques over all rotation steps with the light being 3.1 units above the ground.

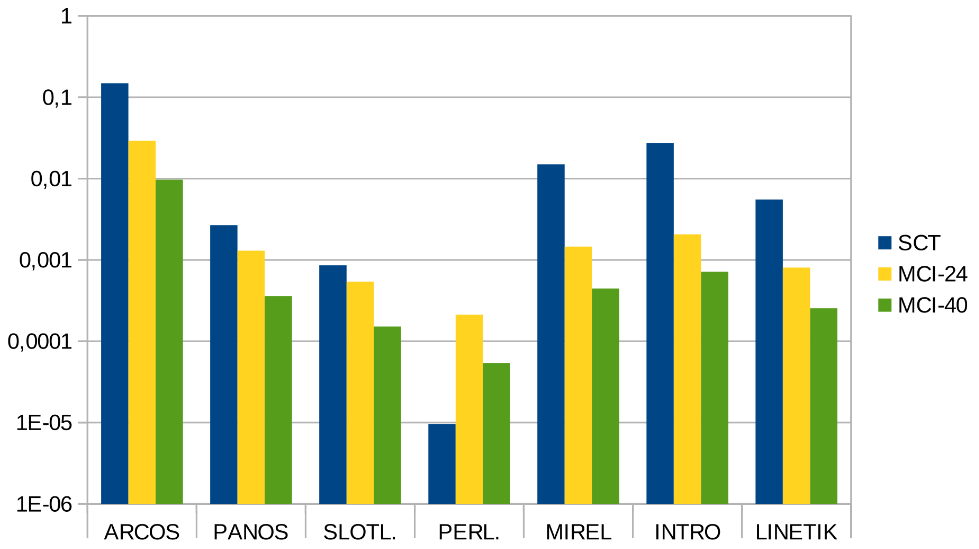
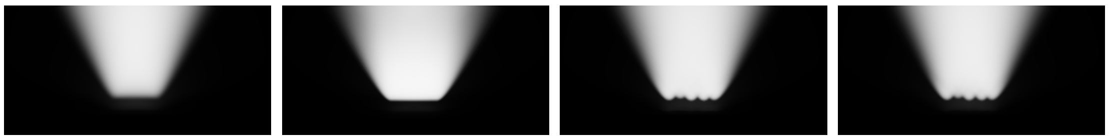


Figure 6.10: Normalized mean squared error of various techniques over all rotation steps with the light being 5.1 units above the ground.

distribution and its magnitude, the dimension of the light source and the spatial relation of illuminated surfaces to the light source. We observed that directional and brighter luminaires lead to more noticeable artifacts. MCI approximations of ARCOS, which is the strongest and most directional luminaire in our evaluation, have noticeable artifacts



(a) Reference (b) SCT (c) MCI-24 (d) MCI-40

Figure 6.11: Border area of ARCOS at height 0.1 with an angle of $3\pi/4$ for various approximations. It can be seen that MCI-24 and MCI-40 show light bumps in the border area while SCT creates a border area similar to the reference solution.

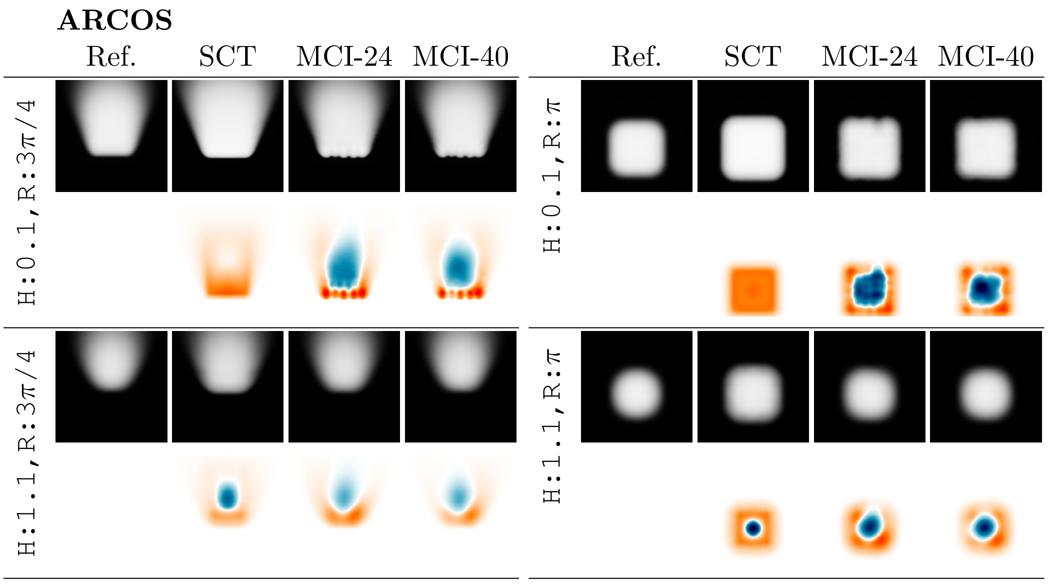


Table 6.3: Approximations of ARCOS and the corresponding error images generated with the scattered sampling (SCT) and Monte Carlo integration (MCI-24, MCI-40). $H$ defines the height and $R$ the rotation state. The images are tone mapped as described in Section 6.2 with a fixed scaling value of 0.1.

at close distances as shown in Table 6.3.

If the luminaires have a less directional luminous intensity distribution, like PANOS and SLOTLIGHT, the artifacts are weaker, yet still noticeable. This is shown in Table 6.4. However, the images show that these artifacts already start to vanish at slightly larger distances compared to ARCOS. The distance of the closest point increases due to the light source rotating around its center. At a rotation of $3\pi/4$, the shortest distance between the ground plane and the light source, which has had an initial height of 0.1, is ~0.246 units. Table 6.4 shows that the artifacts become barely visible for PANOS and SLOTLIGHT at this distance and vanish at further rotation steps. However, artifacts remain visible for ARCOS even after a rotation by $\pi$ radians, where the shortest distance to the ground plane is 0.6 units when the light source had an initial height of 0.1 units.

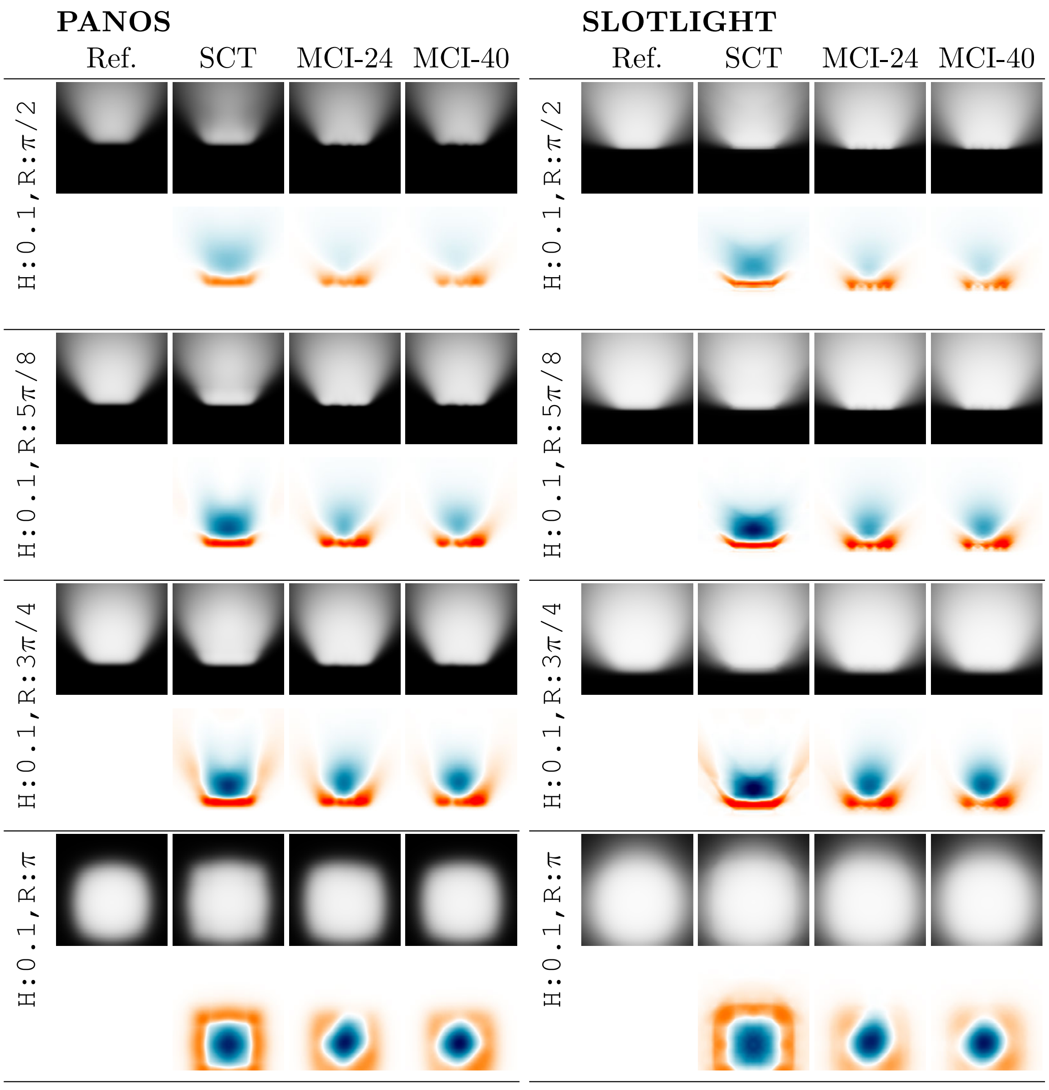


Table 6.4: Approximations of PANOS and SLOTLIGHT and the corresponding error images generated with the scattered sampling (SCT) and Monte Carlo integration (MCI-24, MCI-40).*H* defines the height and *R* the rotation state. The images are tone mapped as described in Section 6.2 with a fixed scaling value of 0.1.

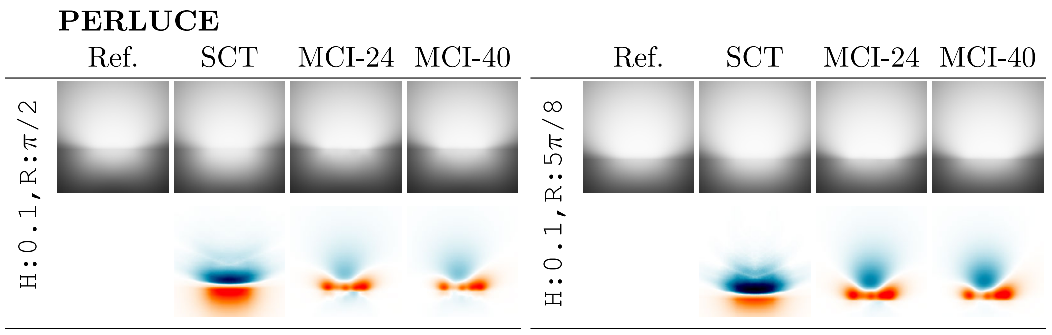


Table 6.5: Approximations of PERLUCE and the corresponding error images generated with the scattered sampling (SCT) and Monte Carlo integration (MCI-24, MCI-40). *H* defines the height and *R* the rotation state. The images are tone mapped as described in Section 6.2 with a fixed scaling value of 0.1.

PERLUCE is the most diffuse luminaire. Table 6.5 shows that there are no noticeable artifacts, even when the light source is close. While the error images show overexposed spots near the light source, they are so weak that they cannot be seen in the tone-mapped images.

MIREL emits light approximately Phong-like, yet differs such that the strongest emission direction is not forward. The approximations generated with MCI-24 and MCI-40 have bump artifacts at short distances similar to PANOS due to the directionality as shown in Table 6.6. SCT creates a border area similar to the reference solution. However, as the distance increases, the images show that artifacts appear in the border area with SCT, while the MCI approximations become more stable. The error images in Table 6.6 show that the illumination variance in the border area increases with an increasing distance for SCT, whereas the variance decreases for MCI.

The LINETIK approximations also have artifacts for all approximation techniques. Table 6.7 shows that MCI suffers from bump artifacts in the border area, while SCT differs noticeably from the reference solution in the inside area. As the light source moves further away, the bump artifacts with MCI vanish while the artifacts in the inside area remain noticeable with SCT.

All techniques create artifacts when approximating INTRO, which is the only luminaire where the main emission direction is to the side. Table 6.8 shows the approximations. The error images show that there are light bumps with MCI-24 and MCI-40 near the light plane when the light source is close, which can be seen in the tone-mapped images near the light source. SCT also creates visible artifacts inside the illuminated area. Moreover, all techniques fail to approximate the light emitted to left correctly and instead render a light spike, especially prominent with SCT. With increasing distance, all techniques render more plausible approximations. However, the light spike remains with SCT.

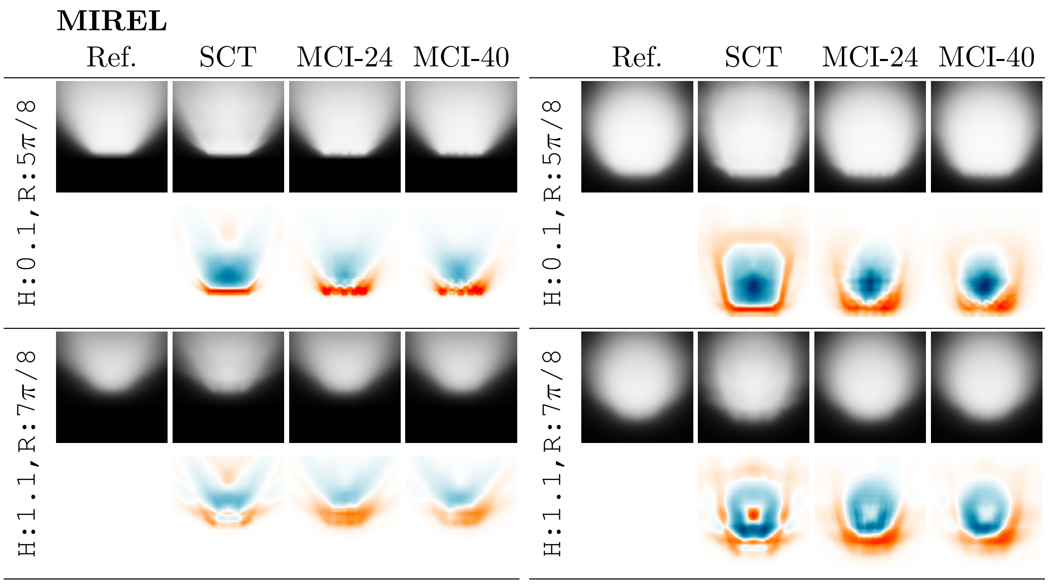


Table 6.6: Approximations of MIREL and the corresponding error images generated with the scattered sampling (SCT) and Monte Carlo integration (MCI-24, MCI-40).$H$ defines the height and $R$ the rotation state. The images are tone mapped as described in Section 6.2 with a fixed scaling value of 0.1.

## 6.6 Discussion

The results show that scattered sampling creates plausible approximations with a small number of samples when the light source is close. While errors remain because of the small number of samples, they can only be seen in direct comparison to the reference solution or in the error images. This is because of the error being consistent, which means that there is no high-frequent alternation of over- and under-illumination. Therefore the approximations look plausible.

Monte Carlo integration, on the other hand, suffers of such a high illumination alternation at close distances which shows as visible light bumps in the border area. How visible these light bumps are depends on the luminous intensity distribution and the distance of the illuminated surface to the light source. The results show that there is a direct correlation between the directionality of a luminaire and the light bumps. As the luminaire emits light more diffusely, the light bumps are less prominent at closer distances (e.g., ~0.246 units with SLOTLIGHT and 0.1 units with PERLUCE), while they remain visible even at large distances for highly directional luminaires (e.g.,0.6 units with ARCOS). Another important factor besides the form of the luminous intensity distribution is tone mapping. Depending on the tone mapping, bump artifacts may not be visible even at close distances.

However, scattered sampling suffers also from visible errors if the luminous intensity distribution is not Phong-like, like in the case of MIREL, LINETIK or INTRO. Our

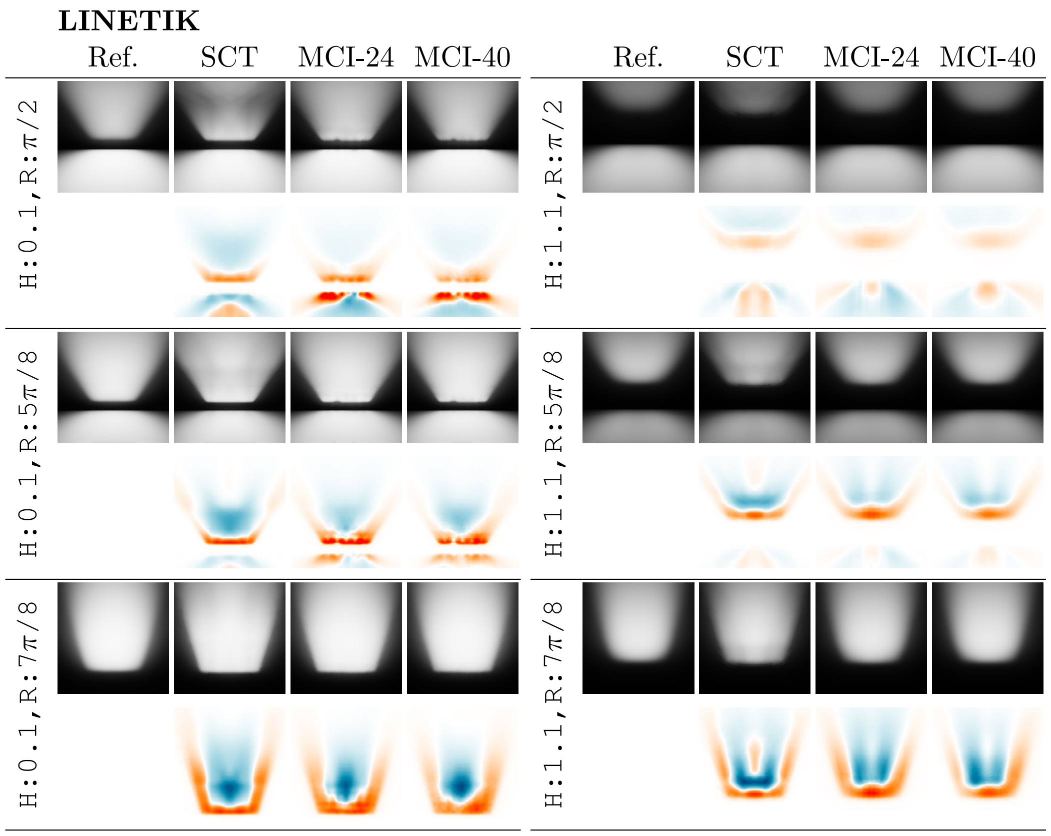


Table 6.7: Approximations of LINETIK and the corresponding error images generated with the scattered sampling (SCT) and Monte Carlo integration (MCI-24, MCI-40). $H$ defines the height and $R$ the rotation state. The images are tone mapped as described in Section 6.2 with a fixed scaling value of 0.1.

reference implementation uses the closest point, which proves to be an important sample for luminaires with a Phong-like light emission, as the most light is emitted forwards by Phong-like luminaires. This is not the case for MIREL, LINETIK or INTRO. Therefore, errors appear inside the illuminated area, whereas Monte Carlo integration approximates the illumination accurately. While light bumps vanish with Monte-Carlo integration at larger distances, the errors of scattered sampling become less prominent but remain visible.

The choice of the dynamic sample has a large influence on the results because the only other samples in our reference implementation are the structured samples in the corners. As there are no other samples inside the area light, the dynamic sample has to be positioned such that it compensates this lack of samples. While this is the case with the closest point for luminaires with a Phong-like emission, it does not work for other luminaires as our results show.

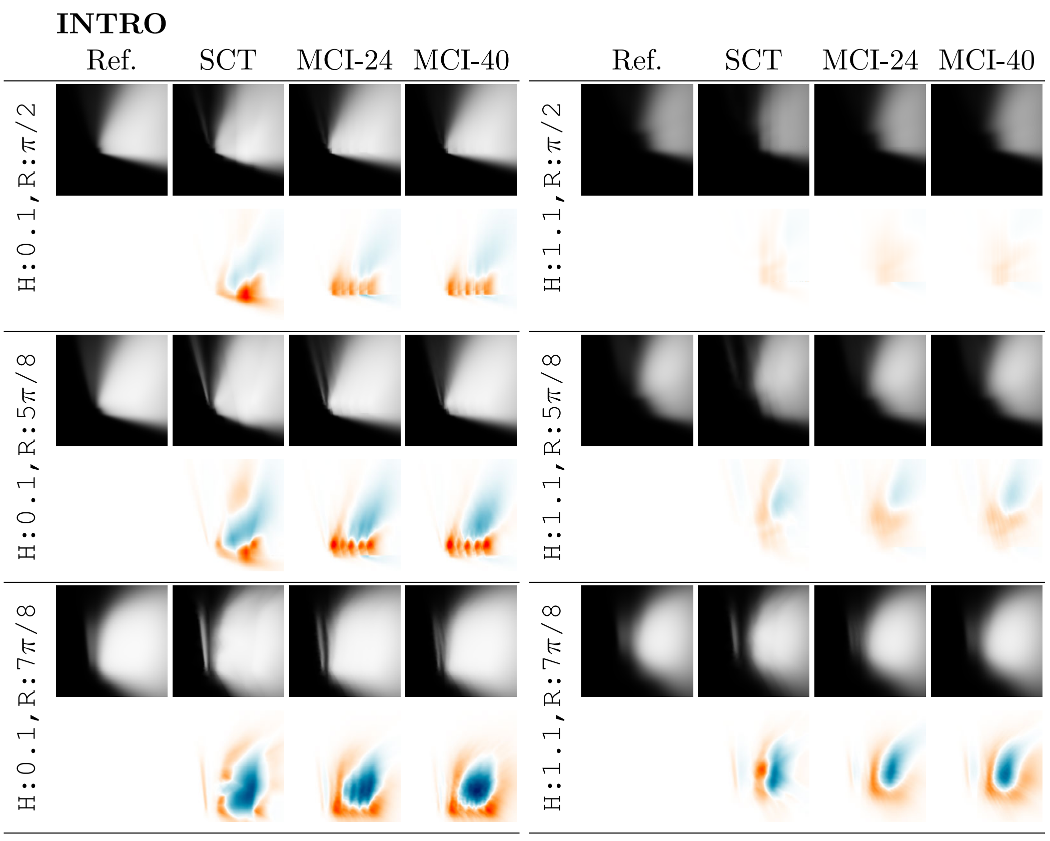


Table 6.8: Approximations of INTRO and the corresponding error images generated with the scattered sampling (SCT) and Monte Carlo integration (MCI-24, MCI-40).$H$ defines the height and $R$ the rotation state. The images are tone mapped as described in Section 6.2 with a fixed scaling value of 0.1.

Another issue, which is independent of the used dynamic sample, becomes visible with INTRO. The photometry of INTRO states that light is emitted inside the light plane. Scattered sampling, however, uses a cubature technique based on the Delaunay triangulation. The spherical excess of a triangle on the light source is 0 inside the light plane, which means that the illumination has to be zero by definition. This contradiction is a problem for which we currently do not have a solution. As a result, luminaires with a lateral emission prove to be an issue for scattered sampling at its current state.

Regarding performance, our implementation of scattered sampling with 5 samples (~300 FPS with flipping, ~400 FPS without) is as fast as our reference renderer with 40 Poisson samples (~300 FPS) or 24 Poisson samples (~400 FPS). This frame rate difference is due to Monte Carlo integration being less complex than scattered sampling, which allows a larger number of samples. While direct lighting can be evaluated with Monte Carlo integration by only summing over a list of samples, scattered sampling requires the modification and evaluation of a complex data structure.

To summarize: The results show that scattered sampling performs well when the light source is close, while Monte Carlo integration performs well when the light source is further away. What *close* and *further away* means depends on the luminaire. While a diffuse luminaire like PERLUCE can be approximated with Monte Carlo integration at close distance without artifacts, light bumps remain visible at large distances for highly directional luminaires like ARCOS.

A solution to generally avoid artifacts could be to combine scattered sampling and Monte Carlo integration with blending. For surface points where the light source is close scattered sampling could be used to create a plausible border area. As the distance of the points to the light source increases, the illumination could be blended to the value computed with Monte Carlo integration until only Monte Carlo integration is used. The results show that 24 Poisson samples are already enough to create a plausible approximation with Monte Carlo integration for a rectangular luminaire with the size of $1 \times 1$ units after a certain distance.

CHAPTER 7

# Conclusion and Future Work

In this thesis, we have presented scattered sampling, a novel approach to evaluate the direct lighting from photometric area lights in real time. The technique, which is based on structured sampling and the most representative point (both techniques currently used in game engines), allows rendering visually plausible approximations of scenes illuminated by photometric area lights with a small number of samples. While visible errors remain when used with complex, asymmetric luminaires, the approximations are visually close to the reference solutions in tone-mapped images.

An issue we observed when using Monte Carlo integration with a small and constant sample set for photometric area lights is that their arbitrary luminous intensity distribution can lead to bump artifacts. These bumps, especially noticeable at the border of the illuminated area, appear due to a sampling pattern on the light source that is not dense enough. Scattered sampling, which combines structured samples with dynamic samples (samples whose positions depend on the spatial relation of the area light to the illuminated point, e.g., the most representative point), addresses this issue effectively, as the dynamic sample covers the regions of the area light which are not covered by the structured samples without increasing the number of samples. The border areas are, as a result, close to the reference solution for luminaires that do not emit light inside the light plane. The latter shows to be a problem as the used cubature technique for scattered sampling depends on the subtended solid angles of triangles on the light source, which is 0 for points the light plane. This results in a contradiction, which again results in an erroneous illumination for all surfaces intersecting with the area light's plane.

The combination of sampling techniques comes at the cost of added complexity. Instead of simply computing a Monte Carlo estimate with a list of samples, scattered sampling requires the handling and managing of a triangle data structure on the GPU. Per fragment, the GPU must fill this data structure, manipulate it and derive a value from it. This added complexity and increase of required memory shows as the frame rate of our reference implementation of scattered sampling with 5 samples is comparable to the frame rate of Monte Carlo integration with 40 Poisson samples.

Besides this added complexity and additional computational costs, our implementation of scattered sampling is currently limited regarding the number of samples inside the area light due to numerical issues we were not able to solve. Deciding whether two samples are on the same position or whether a sample lies exactly between two samples is prone to numerical errors, yet needed to fill the data structure correctly. These issues arise when several samples are placed inside an area light, independent of them being dynamic or static.

Aside of the mentioned issues, the technique shows great potential, as scattered sampling with 5 samples creates visually plausible approximations. Moreover, it outperformed Monte Carlo integration with 40 Poisson samples when approximating a luminaire with a diffuse emission characteristic. Finding more efficient and numerically stable solutions for scattered sampling such that more samples can be used would allow the technique to show its full potential.

During the evaluation, we observed that the choice of the dynamic sample is crucial. We chose the closest point, which showed to be a good choice for luminaires with a Phong-like or diffuse emission. For luminaires with other emission characteristics, the closest point does not have such an important role, which increases the error as the number of contributing samples decreases. Finding more dynamic samples which fit other emission characteristics well remains future work.

APPENDIX A

# Algorithms on the Data Structure

This appendix lists algorithms for the data structure described in Section 5.2.1.

We use two notations to denote array accesses of EV, EE, EM and so on. For example, reading $v_0$ from edge $e$ can be written either as EV[e].$v_0$ or as EV[e][0]. The general notation relation for EV is given by { EV[j][0], EV[j][1], EV[j][2], EV[j][3] } == { EV[j].$v_0$, EV[j].$o_0$, EV[j].$v_1$, EV[j].$o_1$ }. The notation is equivalent for all arrays and used to notate value readings by index.

**Algorithm A.1:** Edge Flip

```
  Data: Index of edge to flip: e; Edges: EV, EE, EM; Faces: FV, FE; Stack: S
  Result: EV, EE, EM, FV, FE and S where edge e is flipped
1 UpdateQuadrilateralEdges(e, EV, EE, EM, S)
  // Store original face edges of e before it is flipped
2 $f_0$ = { EV[e].$v_0$, EV[e].$o_0$, EV[e].$v_1$ }
3 $f_1$ = { EV[e].$v_1$, EV[e].$o_1$, EV[e].$v_0$ }
  // Flip e by leftshifting the indices with rotation.
4 EV[e] = { EV[e].$o_0$, EV[e].$v_1$, EV[e].$o_1$, EV[e].$v_0$ }
5 EE[e] = { EV[e].$e_1$, EV[e].$e_2$, EV[e].$e_3$, EV[e].$e_0$ }
6 UpdateFaces(e, $f_0$, $f_1$, EV, EE, FV, FE)
```

**Algorithm A.2:** Update Quadrilateral Edges

```
Data: Index of edge to flip: e; Edges: EV, EE, EM; Stack: S
Result: EV, EE, EM and S where quadrilateral edges of e are updatet
1  for i = 0 to 3 do
       // Update all edges of the quadrilateral of e
2      e' = EE[e][i]
3      if i mod 2 == 0 then
           // e' is an edge going to an opposite vertex
4          if EV[e'].$o_0$ == EV[e][(i + 2) mod 4] then
5              EV[e'].$o_0$ = EV[e][(i + 3) mod 4]
6          else
7              EV[e'].$o_1$ = EV[e][(i + 3) mod 4]
8          end
9          EE[e'][i] = e
10         EE[e'][i + 1] = EE[e][(i + 3) mod 4]
11     else
           // e' is an edge coming from an opposite vertex
12         if EV[e'].$o_0$ == EV[e][i -1] then
13             EV[e'].$o_0$ = EV[e][(i + 2) mod 4]
14         else
15             EV[e'].$o_1$ = EV[e][(i + 2) mod 4]
16         end
17         EE[e'][i - 1] = EE[e][(i + 1) mod 4]
18         EE[e'][i] = e
19     end
20     if e' == inside edge $\wedge$ e' != marked then
21         Mark e'
22         Push e' on S
23     end
24 end
```

**Algorithm A.3:** Update Faces

```
Data: Index of edge to flip: e; Vertices of faces to update: $f_0$, $f_1$, Edges: EV,
      EE; Faces: FV, FE
Result: FV and FE where faces of edge e are updatet
for $i = 0$ to length of $FV$ do
    if $FV[i]$ has vertices $f_0$ then
        FV[i] = [EV[e].$v_0$, EV[e].$o_0$, EV[e].$v_1$]
        FE[i] = [EE[e].$e_0$, EE[e].$e_1$, e]
    end
    if $FV[i]$ has vertices $f_1$ then
        FV[i] = [EV[e].$v_1$, EV[e].$o_1$, EV[e].$v_0$,]
        FE[i] = [EE[e].$e_2$, EE[e].$e_3$, e]
    end
end
```

**Algorithm A.4:** Triangulate Face

```
Data: Index of vertex to insert: v; Index of face to triangulate: f; Edges: EV,
      EE, EM; Faces: FV, FE;
Result: EV, EE, EM, FV and FE where f is triangulated based on v
e = next free edge id
f' = next free face id
for $i = 0$ to 2 do
    e' = e + i
    // Insert new edge
    EV[e'][0] = FV[f][(i + 0) mod 3]
    EV[e'][1] = FV[f][(i + 1) mod 3]
    EV[e'][2] = v
    EV[e'][3] = FV[f][(i + 2) mod 3]
    EE[e'][0] = FE[f][(i + 0) mod 3]
    EE[e'][1] = e + (i + 1) mod 3
    EE[e'][2] = e + (i + 2) mod 3
    EE[e'][3] = FE[f][(i + 2) mod 3]
    Mark e' as inside
    // Update face edge
    $e_f$ = FE[f][i]
    if $EV[e_f].o_0 == FV[f][(i + 2) \bmod 3]$ then
        EV[$e_f$].$o_0$ = v
        EE[$e_f$][0] = e + (i + 1) mod 3
        EE[$e_f$][1] = e'
    else
        EV[$e_f$].$o_1$ = v
        EE[$e_f$][2] = e + (i + 1) mod 3
        EE[$e_f$][3] = e'
    end
    Mark $e_f$
    // Insert new face
    FV[f' + i] = { EV[e'][0], EV[e'][1], EV[e'][2] }
    FE[f' + i] = { EE[e'][0], EE[e'][1], e' }
end
Remove face f
```

**Algorithm A.5:** Split Edge

```
Data: Index of vertex to insert: v; Edge to split: e; Edges: EV, EE, EM; Faces:
      FV, FE;
Result: EV, EE, EM, FV and FE where f is triangulated based on v
e' = next free edge id
f = next free face id
Remove faces of e
for $i$ = $0$ to 3 do
    if EV[e][i] exists then // Insert new edge
        e" = LID2GID(e, e', i) // Map Local to Global ID
        EV[e"][0] = EV[e][i]
        EV[e"][1] = EV[e][(i + 1) mod 4]
        EV[e"][2] = v
        EV[e"][3] = EV[e][(i + 3) mod 4]
        eO0IsEmpty, eO1IsEmpty = true
        if EV[e][(i + 1) mod 4] exists then
            EE[e"][0] = EE[e][i]
            EE[e"][1] = LID2GID(e, e', (i + 1) mod 4)
            eO0IsEmpty = false
        end
        if EV[e][(i + 3) mod 4] exists then
            EE[e"][2] = LID2GID(e, e', (i + 3) mod 4)
            EE[e"][3] = EE[e][(i + 3) mod 4]
            eO1IsEmpty = false
        end
        e" is inside = $\neg$eO0IsEmpty $\wedge$ $\neg$eO1IsEmpty
        if $\neg$eO0IsEmpty then // Update edge going from $v_0$ to $o_0$
            $e_i$ = EE[e][i]
            if EV[$e_i$].$o_0$ exits $\wedge$ EV[$e_i$].$o_0$ == EV[e][(i + (i == 1 $\vee$ i == 3 ? 3 :
             2)) mod 4] then
                EV[$e_i$] = { EV[$e_i$][0], v, EV[$e_i$][2], EV[$e_i$][3] }
                EE[$e_i$] = { LID2GID(e, e', (i + 1) mod 4), e", EE[$e_i$][2], EE[$e_i$][3] }
            else
                EV[$e_i$] = { EV[$e_i$][0], EV[$e_i$][1], EV[$e_i$][2], v }
                EE[$e_i$] = { EE[$e_i$][0], EE[$e_i$][1], LID2GID(e, e', (i + 1) mod 4), e" }
            end
            // Add new face
            FV[f] = { EV[e][i], EV[e][(i + 1) mod 4], v }
            FE[f] = { $e_i$, LID2GID(e, e', (i + 1) mod 4), e" }
            f = f + 1
        end
    end
end
```

**Algorithm A.6:** Map Local to Global ID (LID2GID)

```
   Data: Freed ID: fid, Next free ID: nid, localId: id
   Result: Global ID to access edge arrays
1  if $id == 0$ then
2      return $id$
3  else
4      return $nid + id - 1$
5  end
```

APPENDIX B

# Results Scattered Sampling

This chapter presents the result for approximations generated with our reference implementation of scattered sampling for the luminaires listed in Section 6.3. The images and error measurements are generated with our evaluation setup presented in Section 6.2. Each table shows the normalized mean squared error (NMSE) over all rotation steps and the upper error bound (UEB) and lower error bound (LEB) for the error images. For each rotation step, a reference image, an approximation and an error image is given.

**ARCOS**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 0.1 | 565.1048 | 160.25 | -8.4375 |

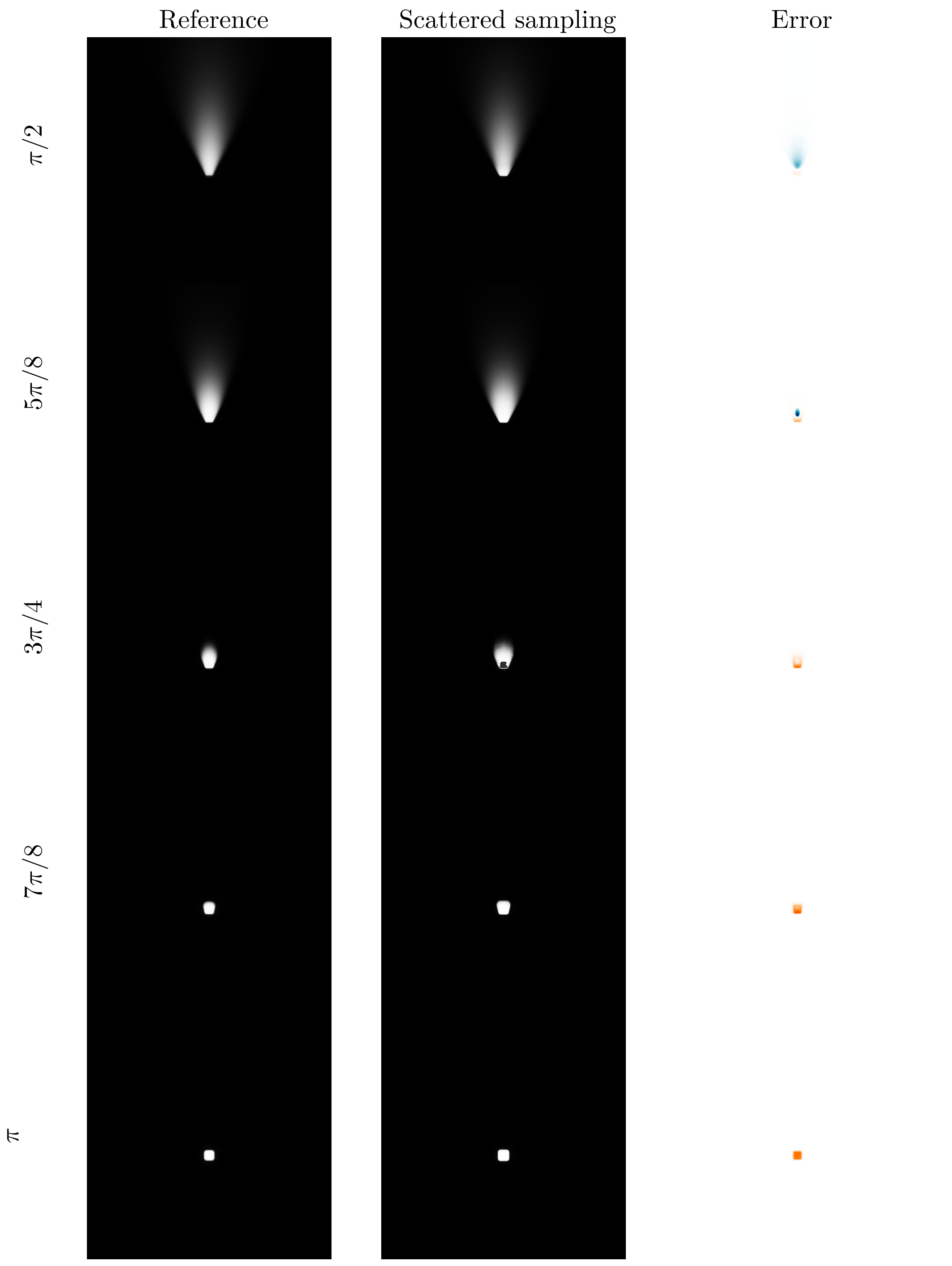


Table B.1: Table showing the results for ARCOS at height 0.1.

**ARCOS**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 1.1 | 27.9669 | 27.125 | -11.8125 |

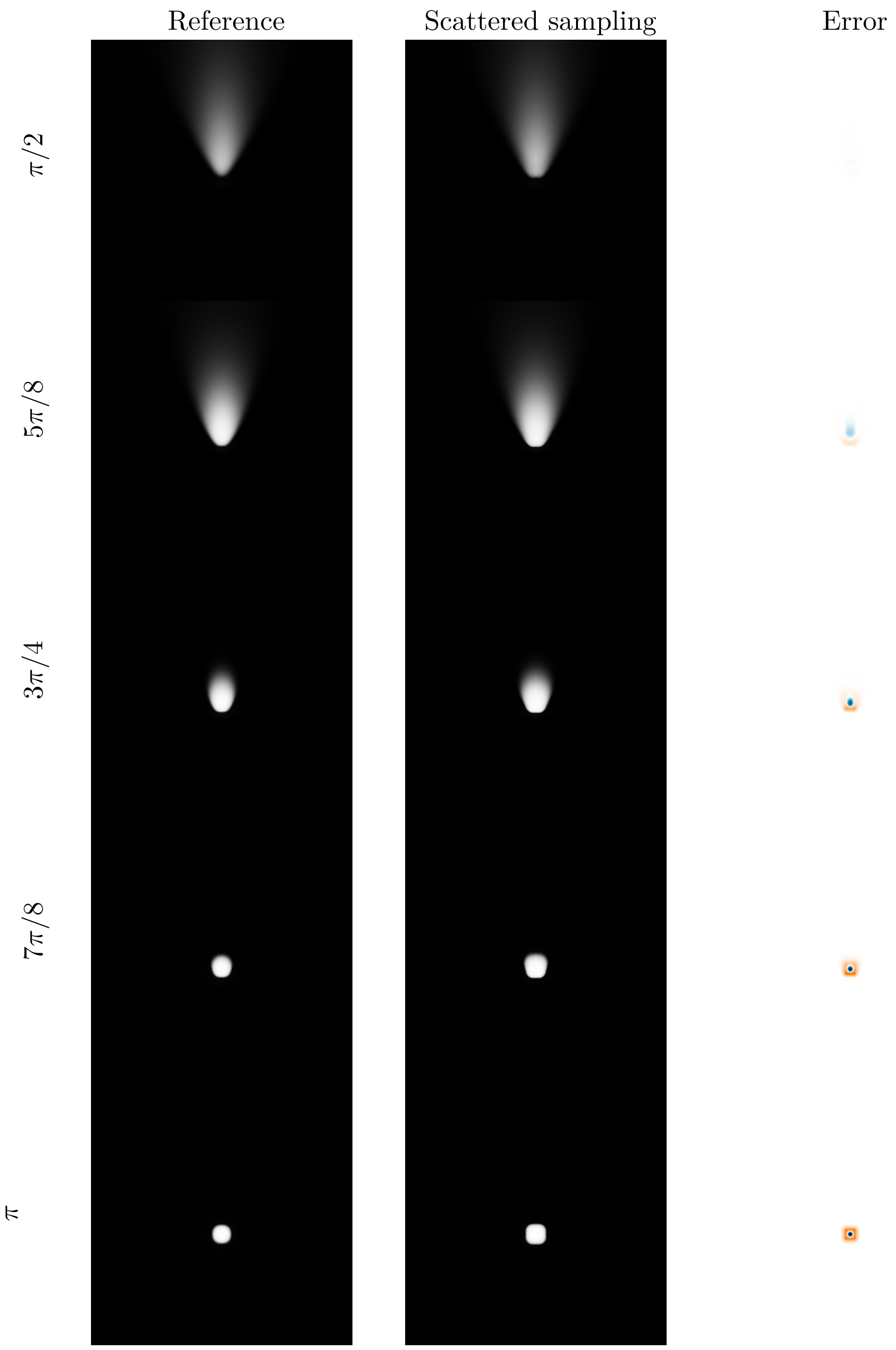


Table B.2: Table showing the results for ARCOS at height 1.1.

**ARCOS**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 3.1 | 1.0217 | 2.3984 | -4.0313 |

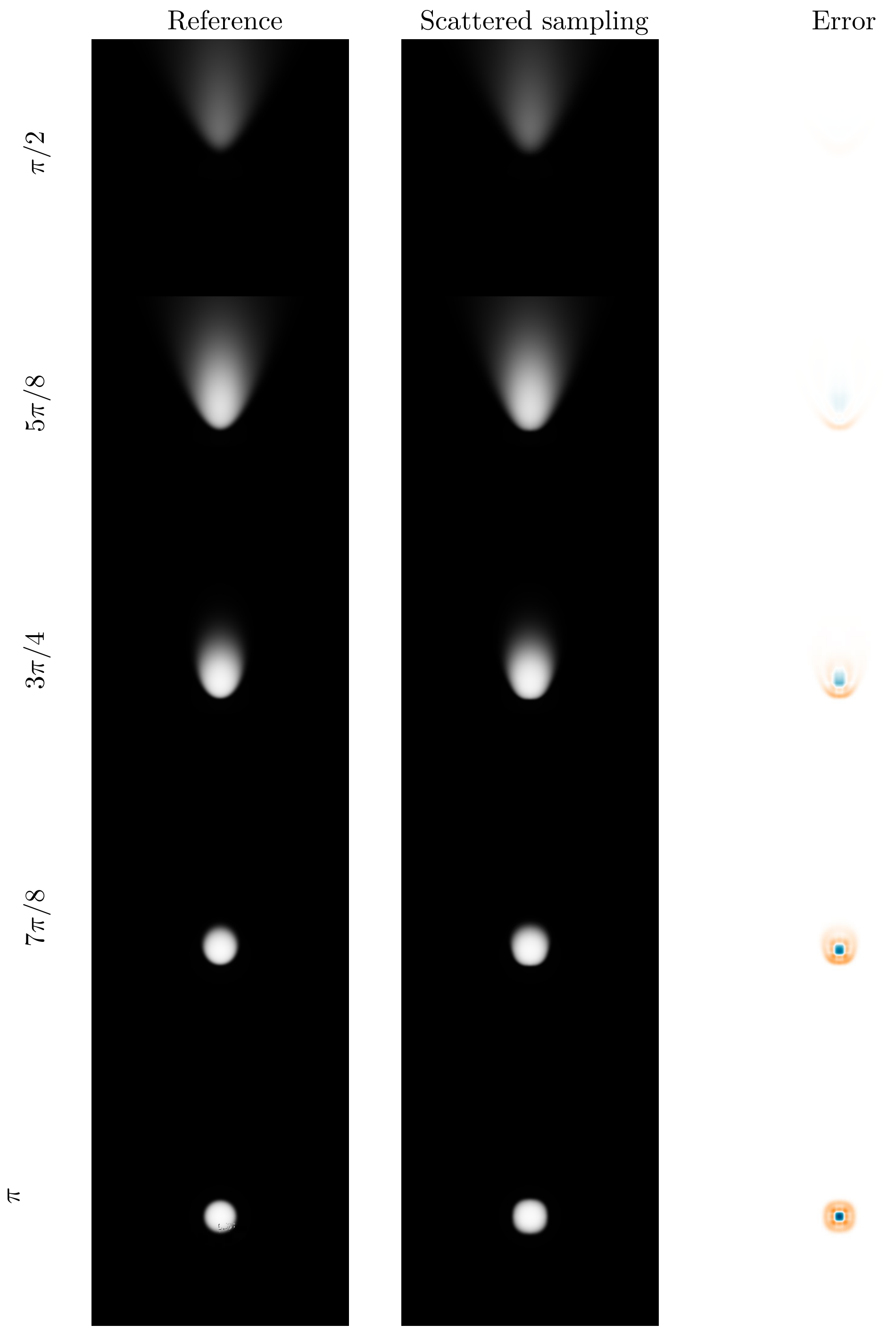


Table B.3: Table showing the results for ARCOS at height 3.1.

**ARCOS**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 5.1 | 0.1487 | 0.58594 | -1.3281 |

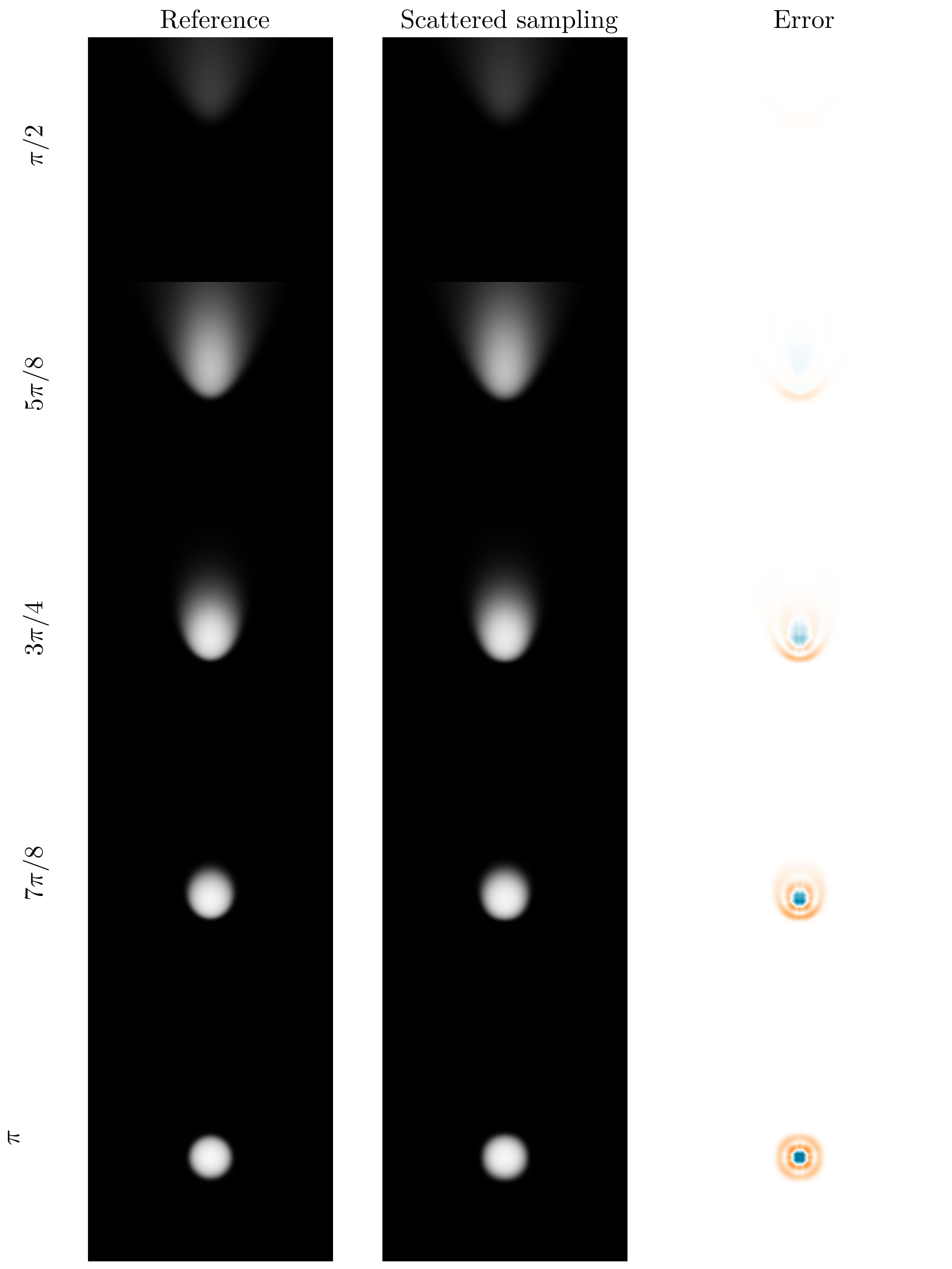


Table B.4: Table showing the results for ARCOS at height 5.1.

**PANOS**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 0.1 | 32.6394 | 10.5625 | -36.375 |

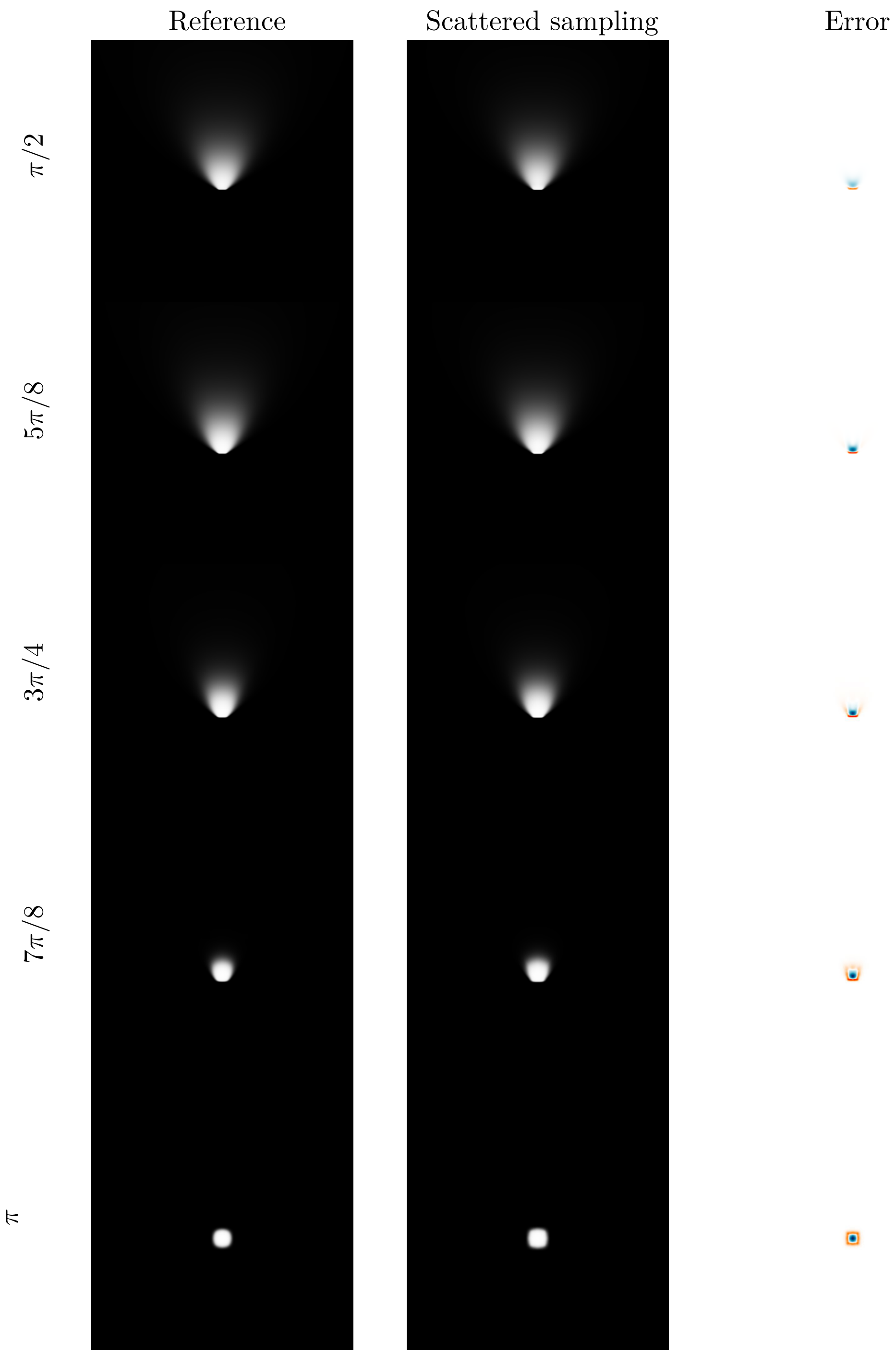


Table B.5: Table showing the results for PANOS at height 0.1.

**PANOS**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 1.1 | 1.7118 | 1.1563 | -6.5313 |

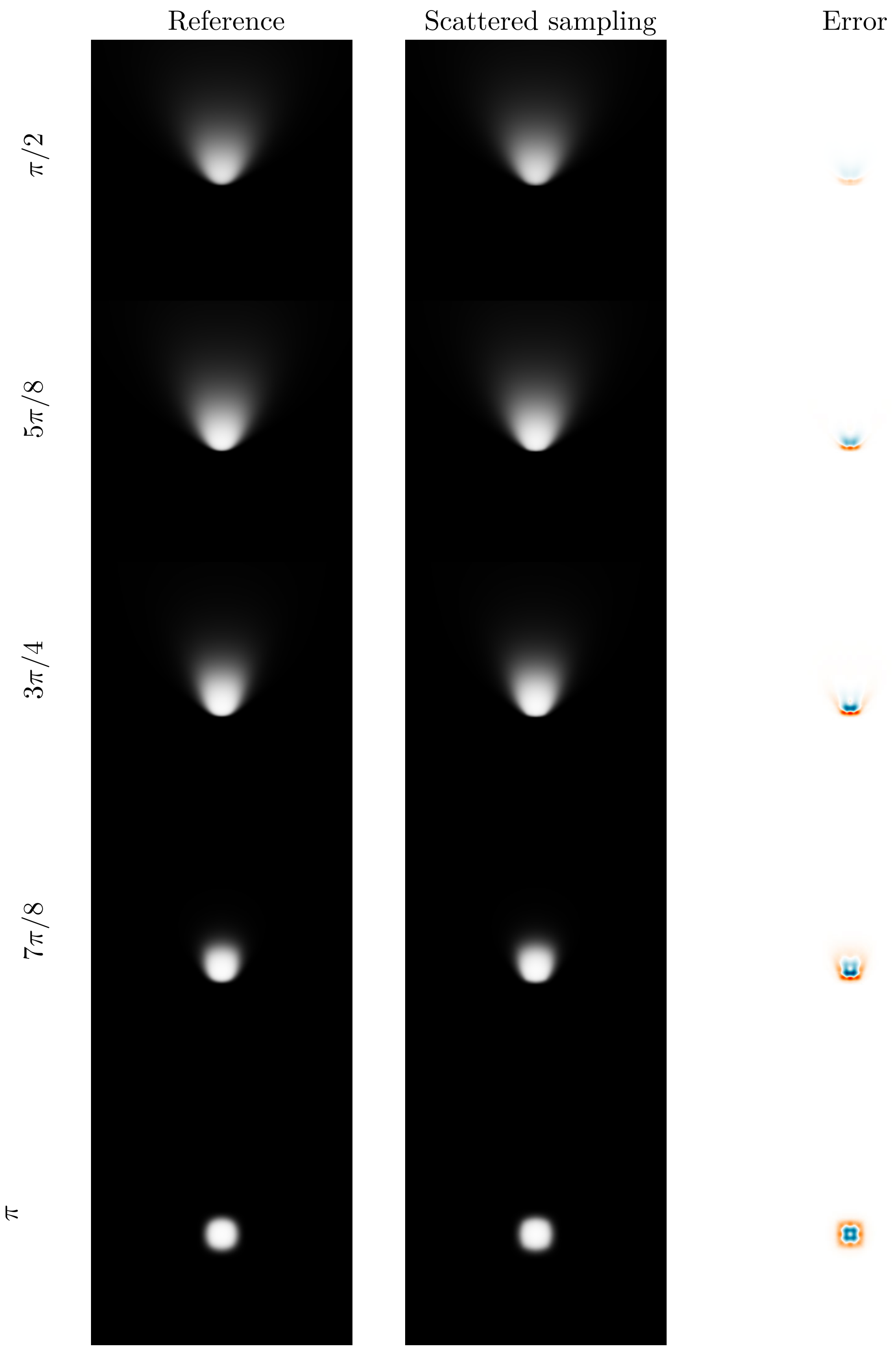


Table B.6: Table showing the results for PANOS at height 1.1.

**PANOS**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 3.1 | 0.029584 | 0.10156 | -0.60938 |

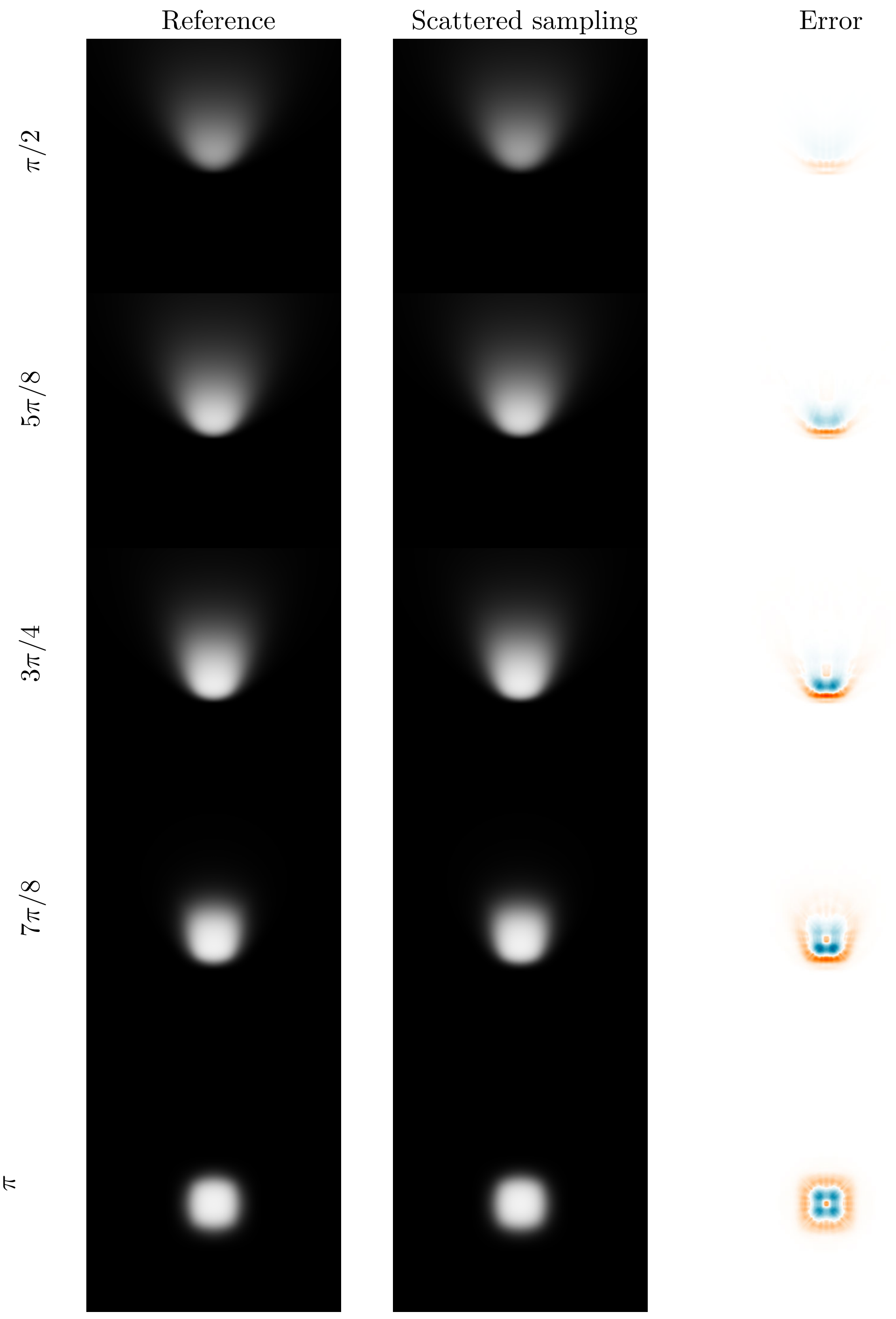


Table B.7: Table showing the results for PANOS at height 3.1.

**PANOS**

| Height | NMSE | UEB | LEB |
|---|---|---|---|
| 5.1 | 0.0026777 | 0.026367 | -0.13672 |

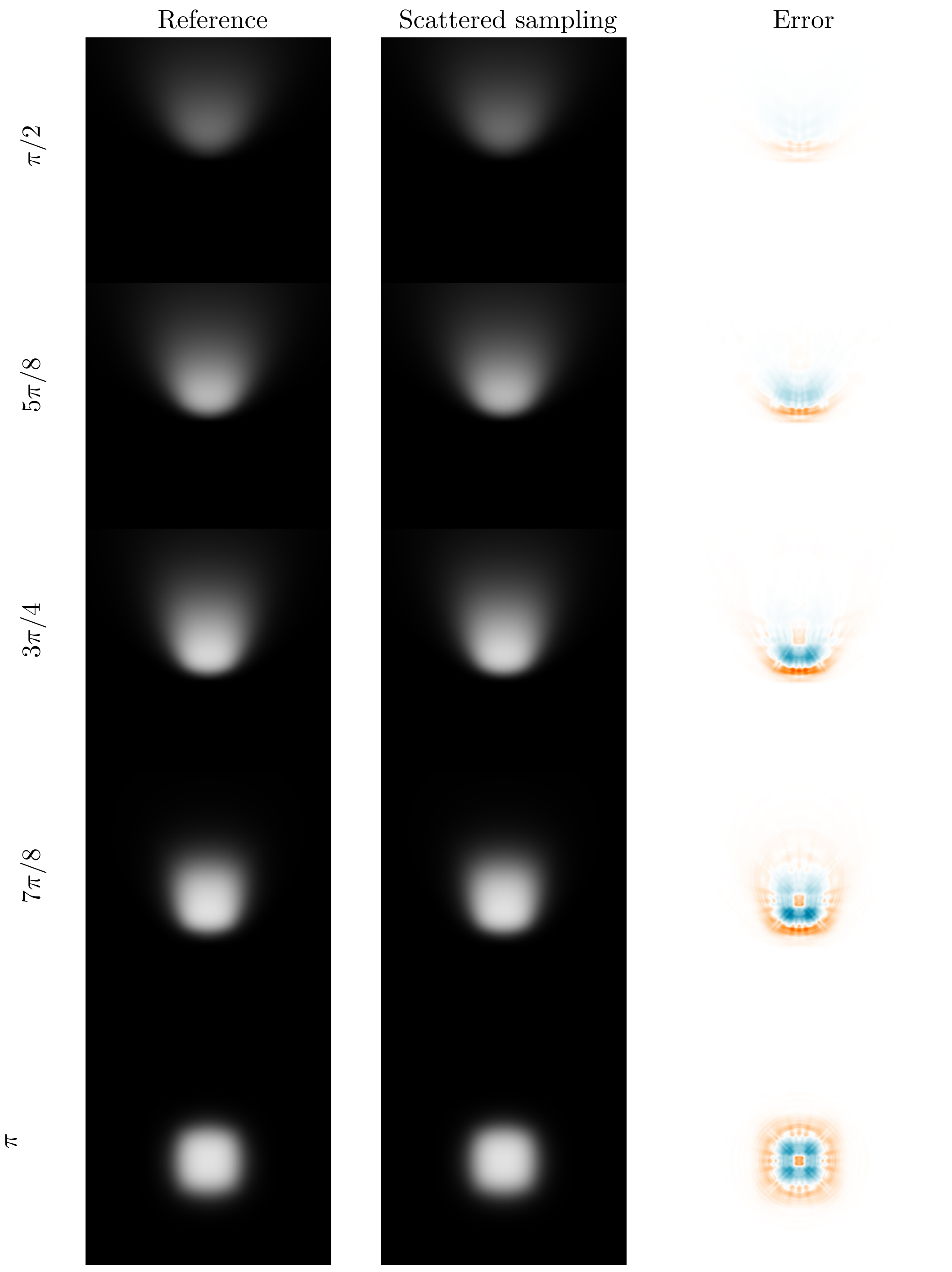


Table B.8: Table showing the results for PANOS at height 5.1.

**SLOTLIGHT**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 0.1 | 10.681 | 8.4375 | -49 |

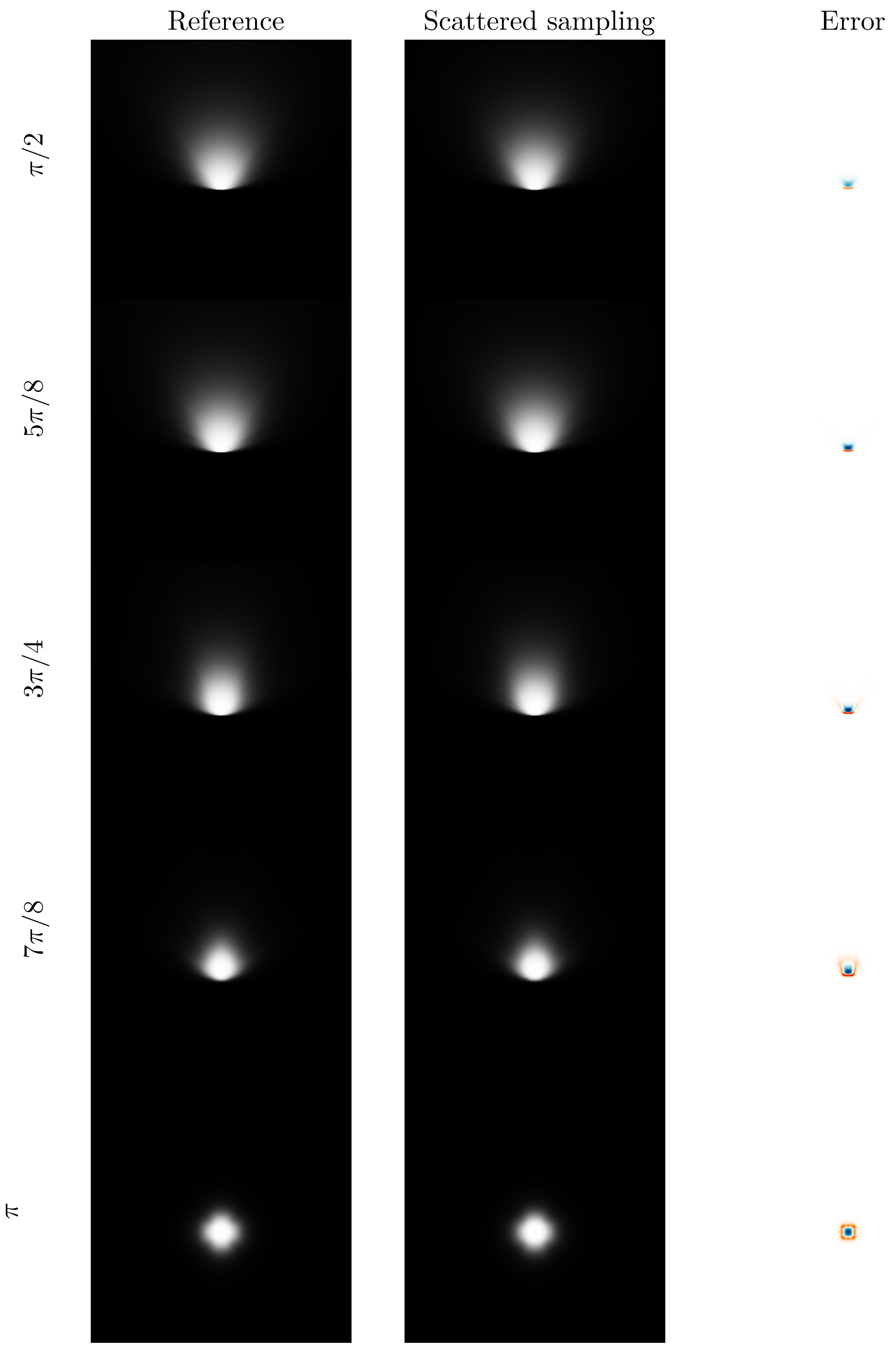


Table B.9: Table showing the results for SLOTLIGHT at height 0.1.

| | **SLOTLIGHT** | | | |
|---|---|---|---|---|
| | **Height** | **NMSE** | **UEB** | **LEB** |
| | 1.1 | 0.308 | 1.5625 | -6.0625 |

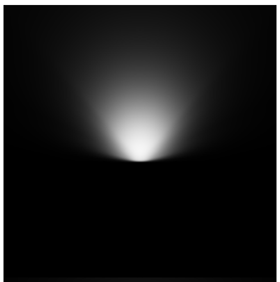
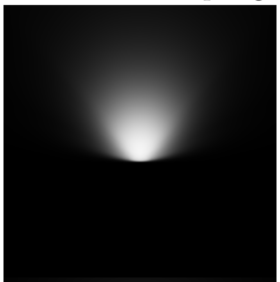
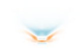
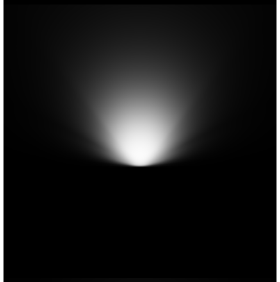
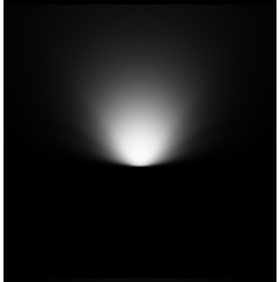
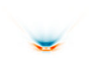
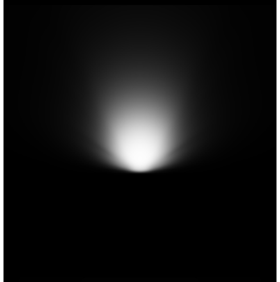
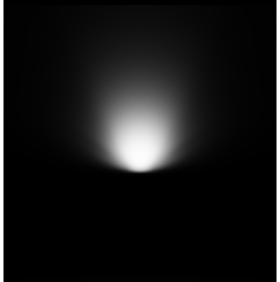
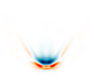
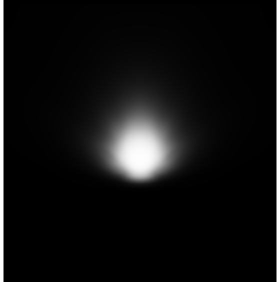
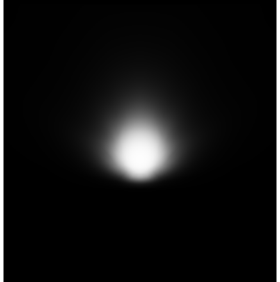
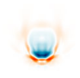
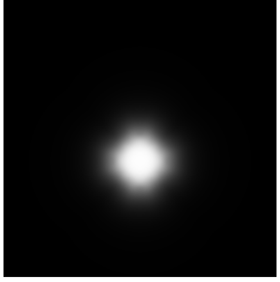
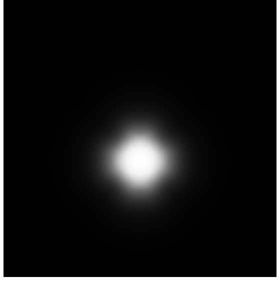
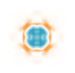
| | Reference | Scattered sampling | Error |
|---|---|---|---|
| $\pi/2$ |  |  |  |
| $5\pi/8$ |  |  |  |
| $3\pi/4$ |  |  |  |
| $7\pi/8$ |  |  |  |
| $\pi$ |  |  |  |

Table B.10: Table showing the results for SLOTLIGHT at height 1.1.

**SLOTLIGHT**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 3.1 | 0.006882 | 0.25 | -0.53125 |

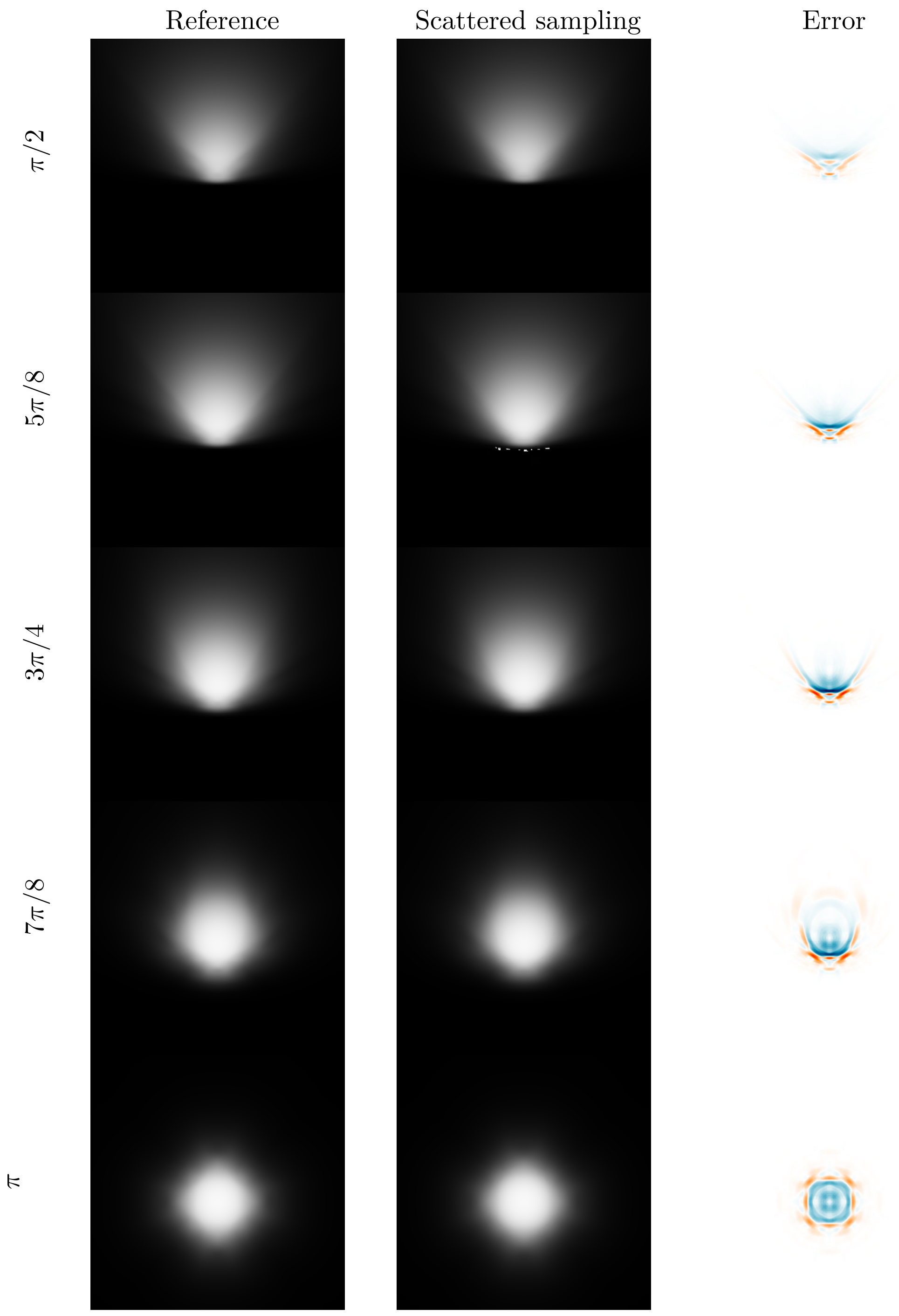


Table B.11: Table showing the results for SLOTLIGHT at height 3.1.

**SLOTLIGHT**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 5.1 | 0.00085754 | 0.082031 | -0.14844 |

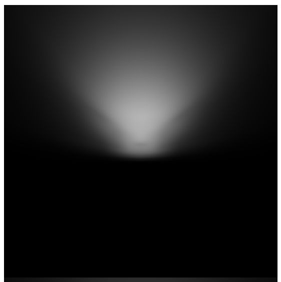
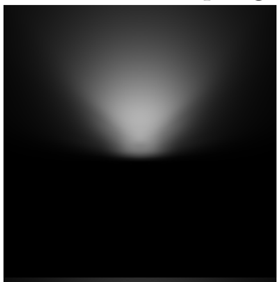
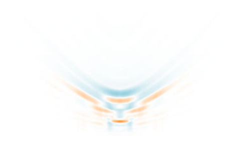
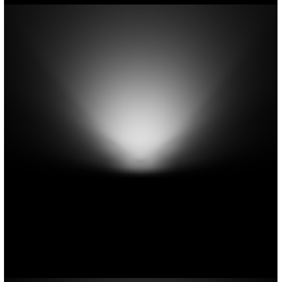
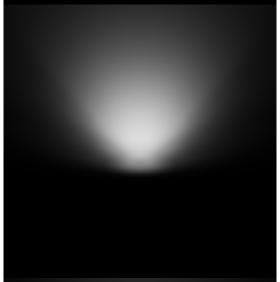
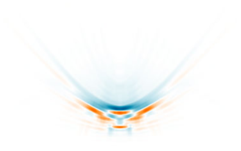
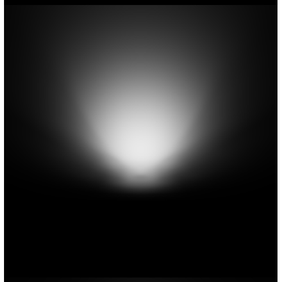
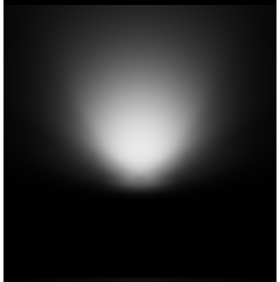
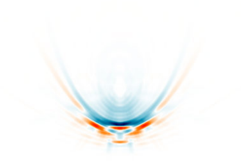
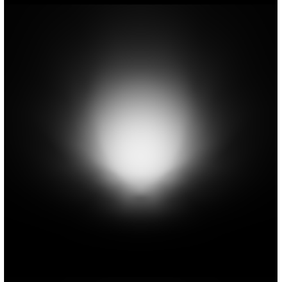
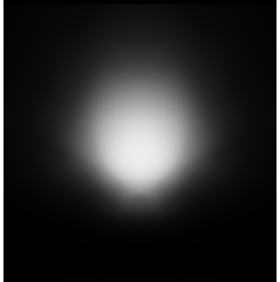
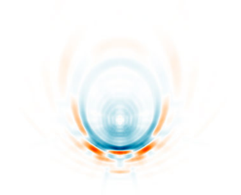
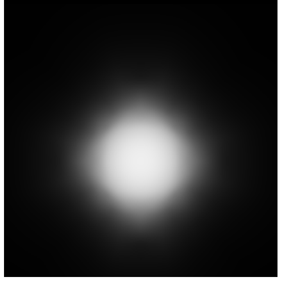
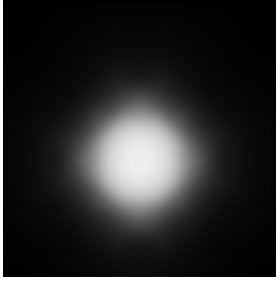
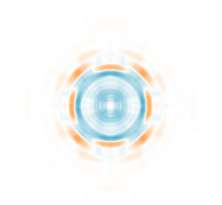
|  | Reference | Scattered sampling | Error |
|---|---|---|---|
| $\pi/2$ |  |  |  |
| $5\pi/8$ |  |  |  |
| $3\pi/4$ |  |  |  |
| $7\pi/8$ |  |  |  |
| $\pi$ |  |  |  |

Table B.12: Table showing the results for SLOTLIGHT at height 5.1.

**PERLUCE**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 0.1 | 0.2382 | 18.7969 | -4.25 |

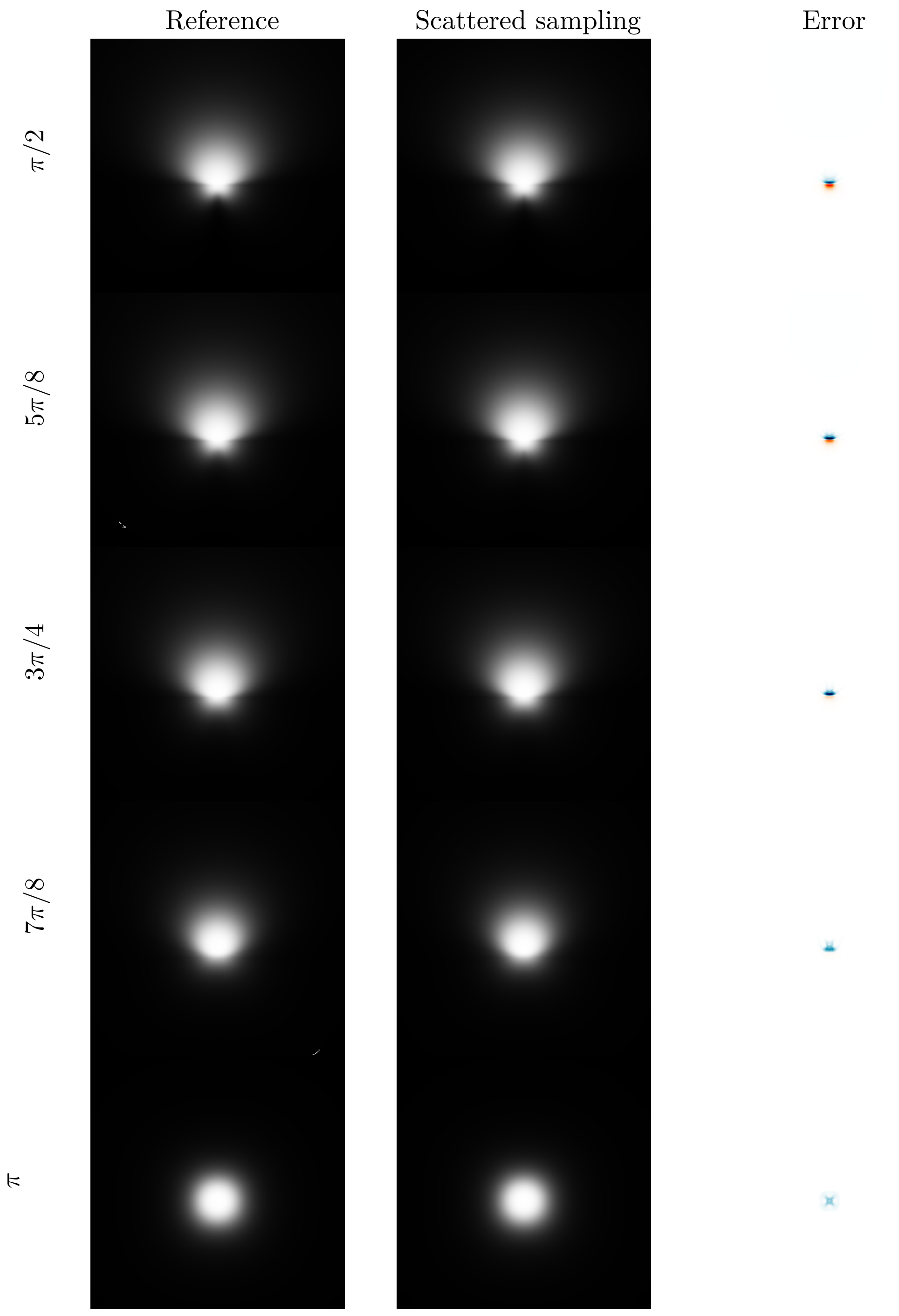


Table B.13: Table showing the results for PERLUCE at height 0.1.

**PERLUCE**

| Height | NMSE | UEB | LEB |
|---|---|---|---|
| 1.1 | 0.0007229 | 0.64453 | -0.25391 |

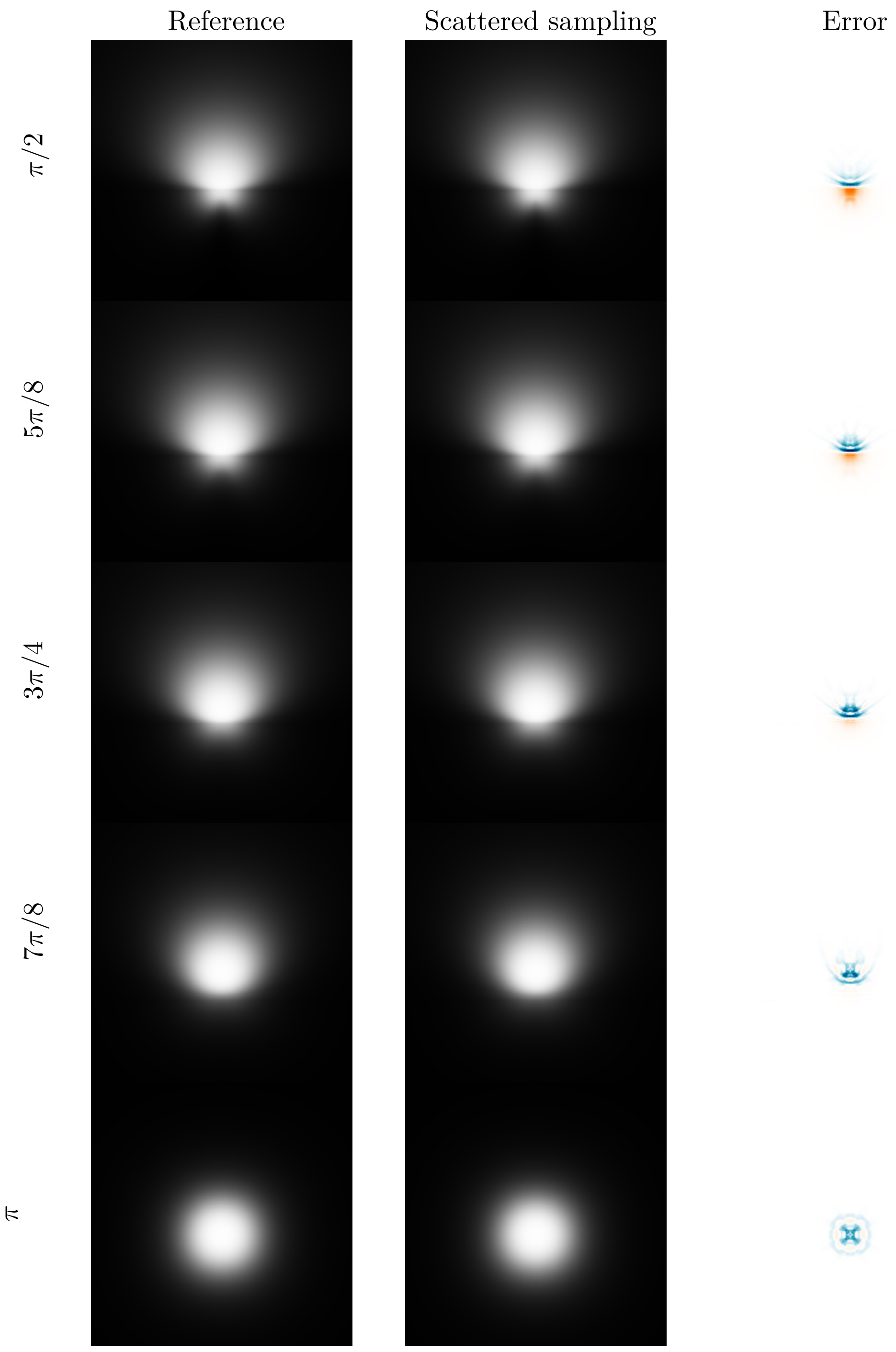


Table B.14: Table showing the results for PERLUCE at height 1.1.

**PERLUCE**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 3.1 | 2.0194e-05 | 0.074219 | -0.17834 |

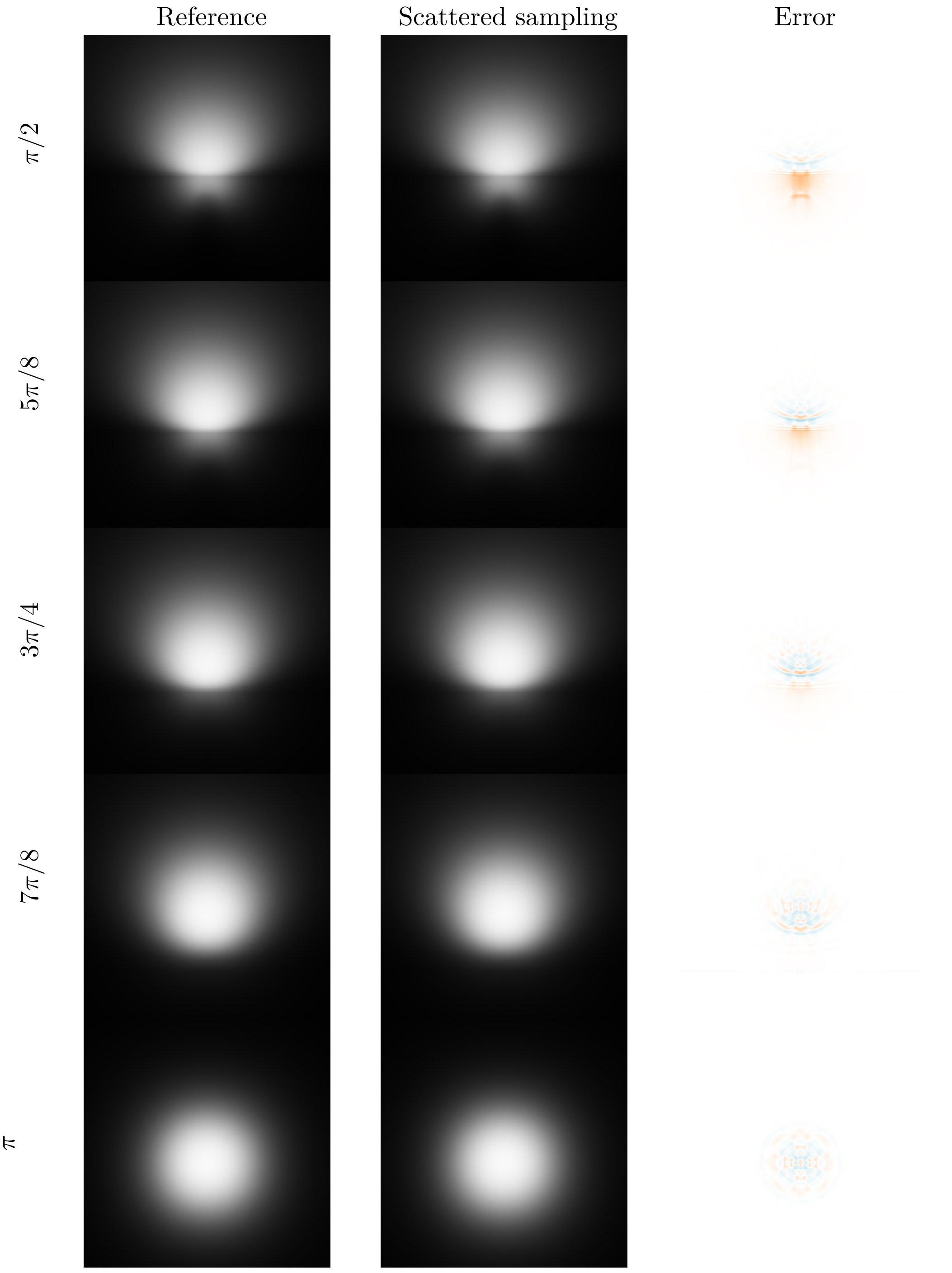


Table B.15: Table showing the results for PERLUCE at height 3.1.

**PERLUCE**

| Height | NMSE | UEB | LEB |
|---|---|---|---|
| 5.1 | 9.5972e-06 | 0.19495 | -0.16321 |

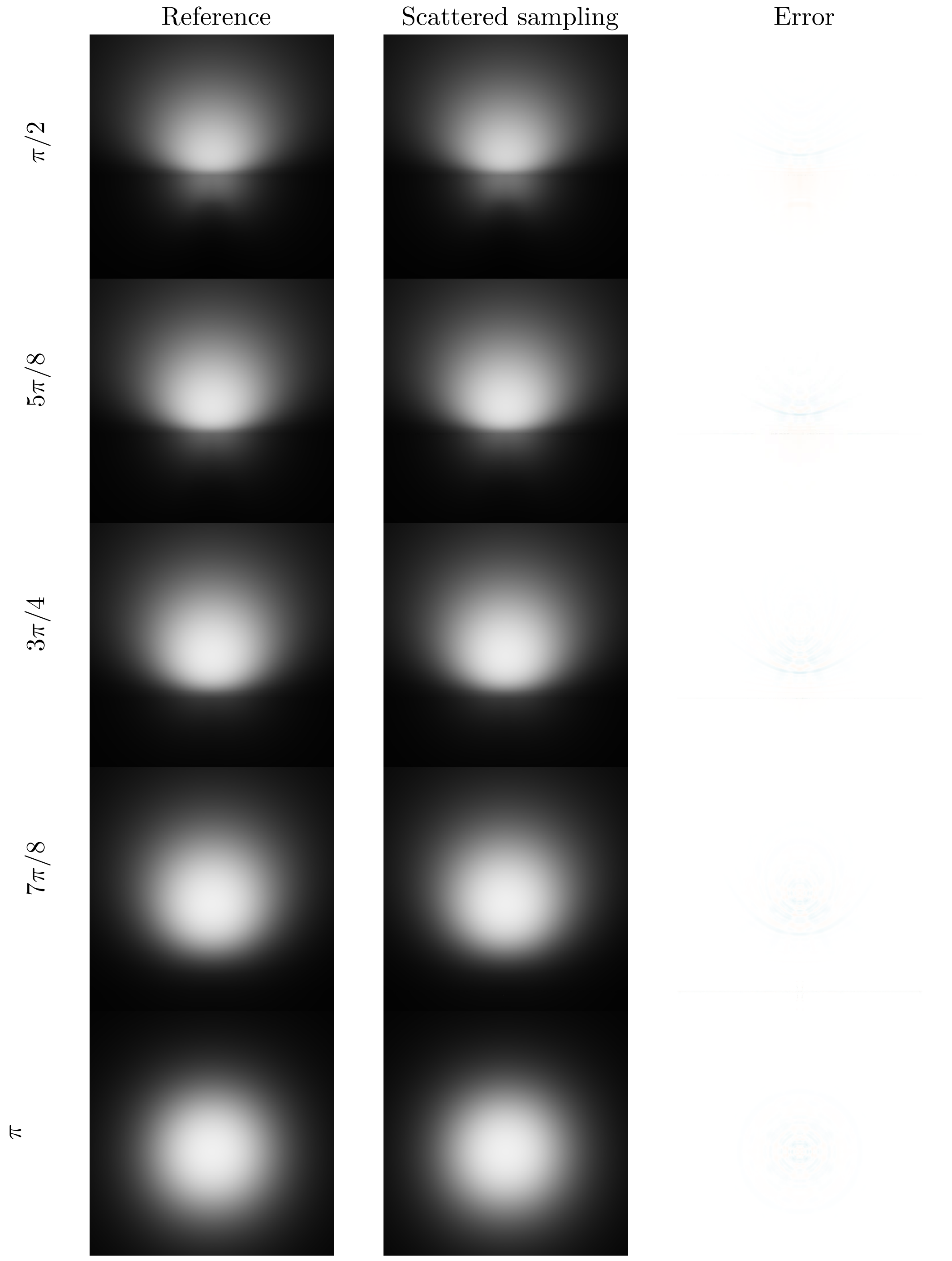


Table B.16: Table showing the results for PERLUCE at height 5.1.

**MIREL**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 0.1 | 56.2119 | 30.2813 | -129.75 |

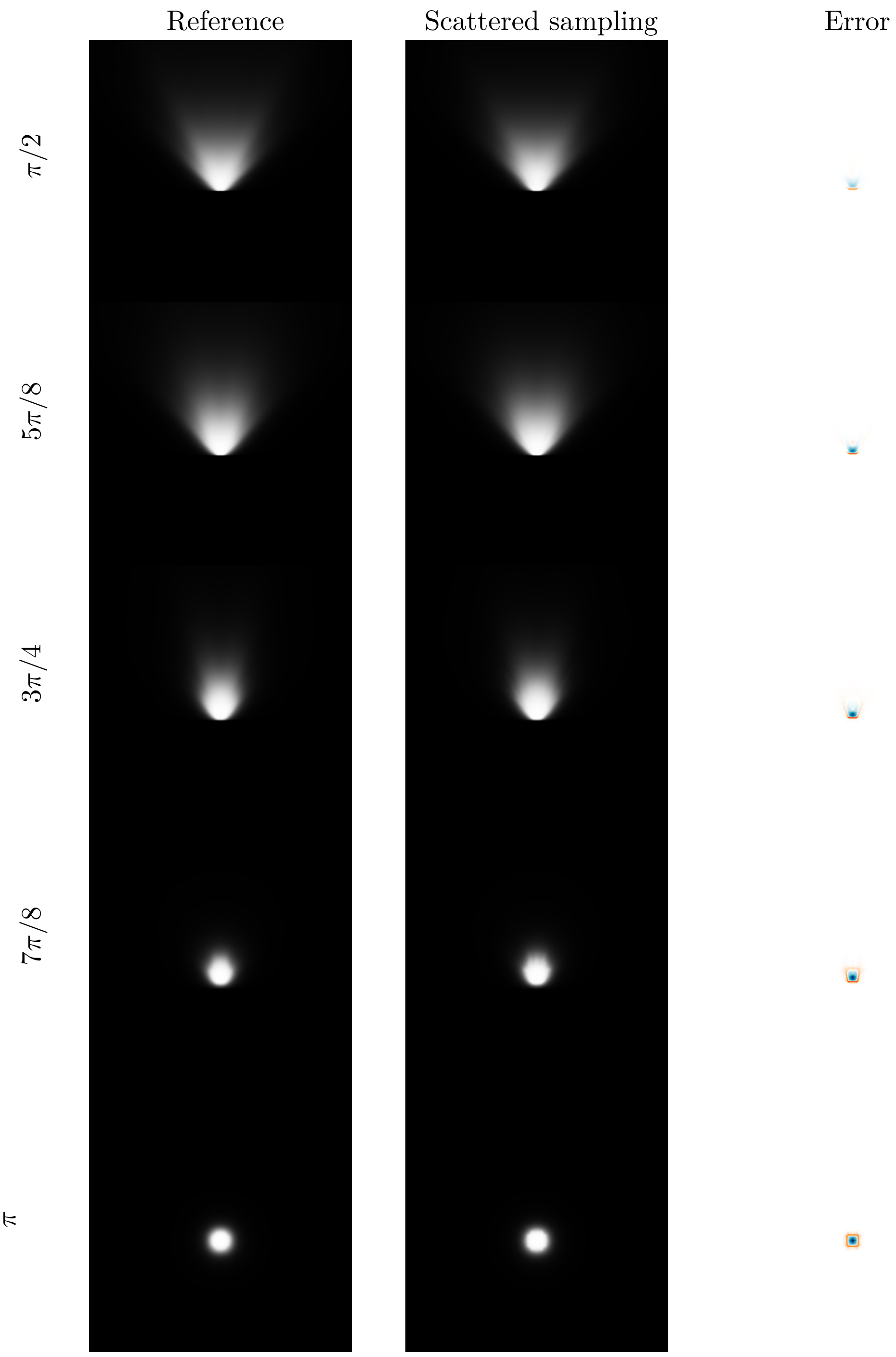


Table B.17: Table showing the results for MIREL at height 0.1.

**MIREL**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 1.1 | 3.0891 | 5.625 | -23.1875 |

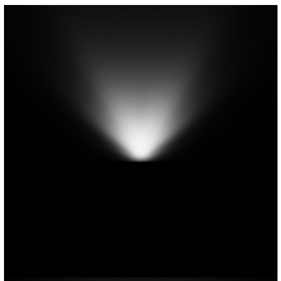
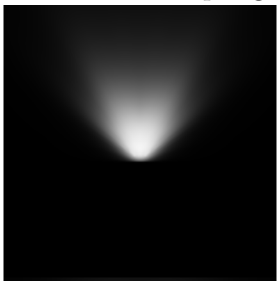
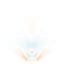
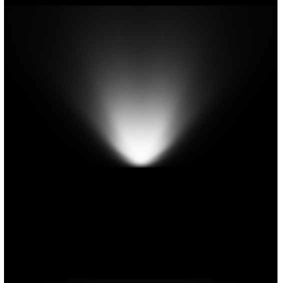
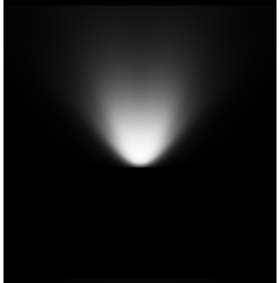
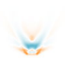
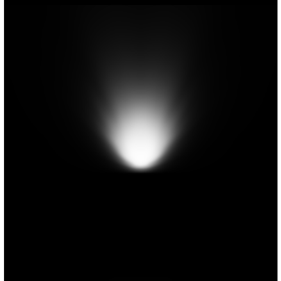
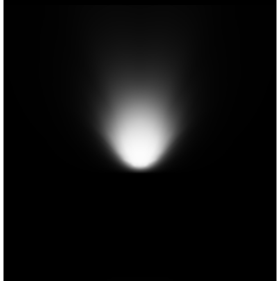
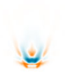
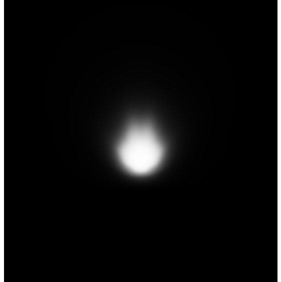
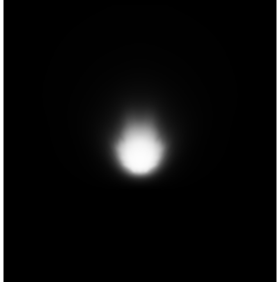
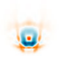
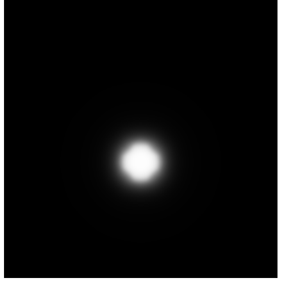
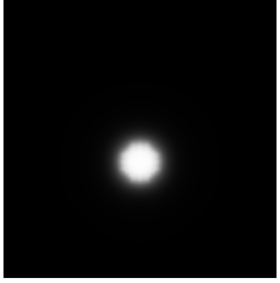
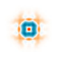
| | Reference | Scattered sampling | Error |
|---|---|---|---|
| $\pi/2$ |  |  |  |
| $5\pi/8$ |  |  |  |
| $3\pi/4$ |  |  |  |
| $7\pi/8$ |  |  |  |
| $\pi$ |  |  |  |

Table B.18: Table showing the results for MIREL at height 1.1.

**MIREL**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 3.1 | 0.11569 | 1.625 | -2.8906 |

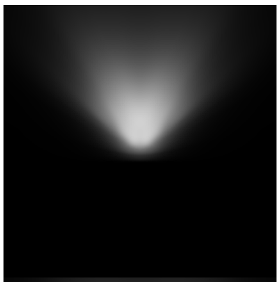
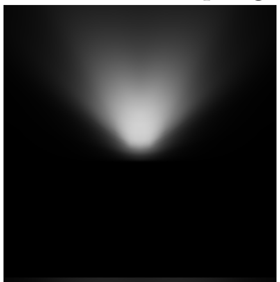
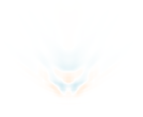
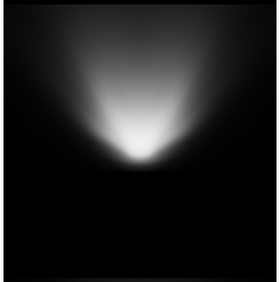
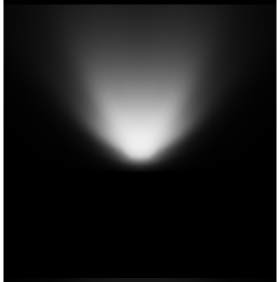
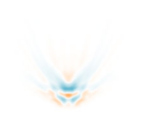
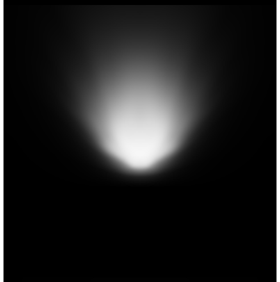
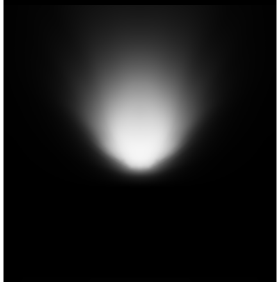
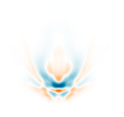
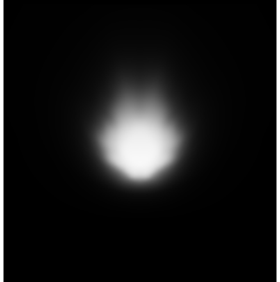
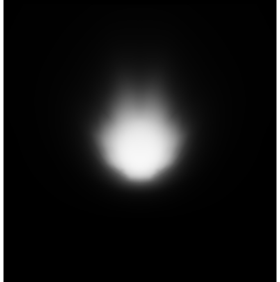
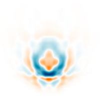
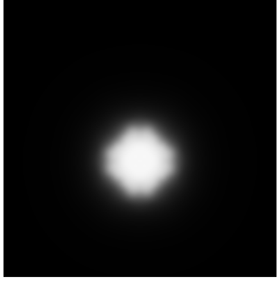
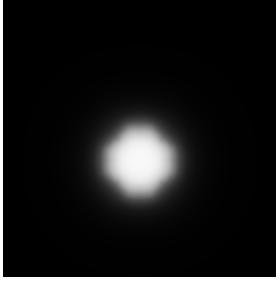
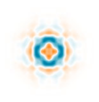
| | Reference | Scattered sampling | Error |
|---|---|---|---|
| $\pi/2$ |  |  |  |
| $5\pi/8$ |  |  |  |
| $3\pi/4$ |  |  |  |
| $7\pi/8$ |  |  |  |
| $\pi$ |  |  |  |

Table B.19: Table showing the results for MIREL at height 3.1.

**MIREL**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 5.1 | 0.014999 | 0.47656 | -0.73438 |

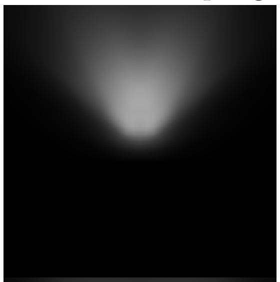
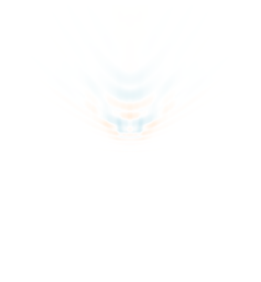
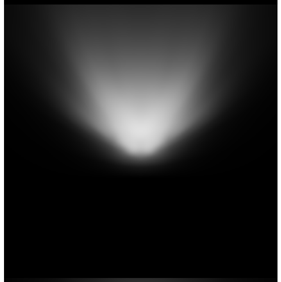
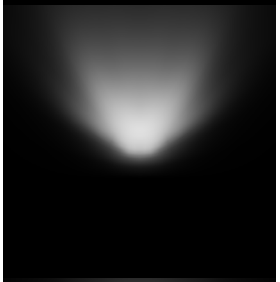
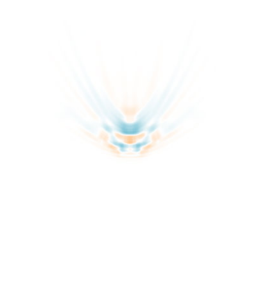
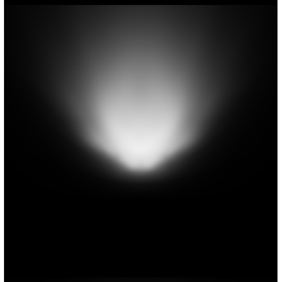
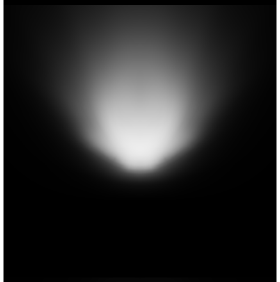
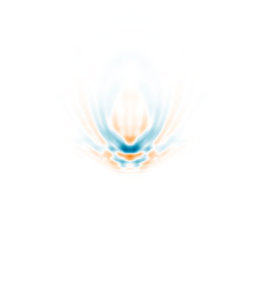
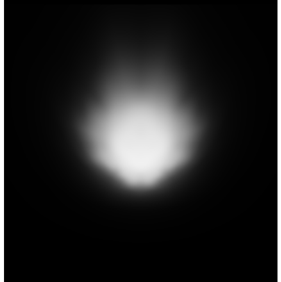
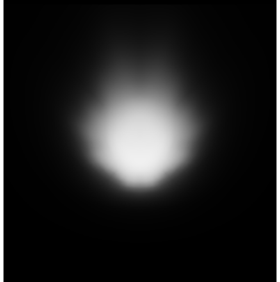
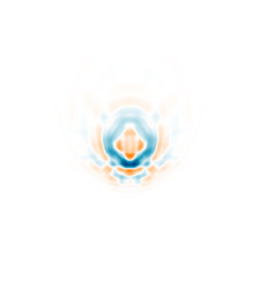
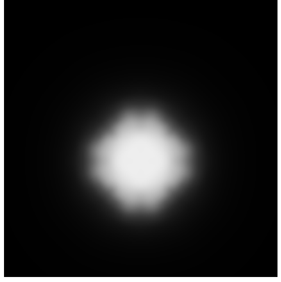
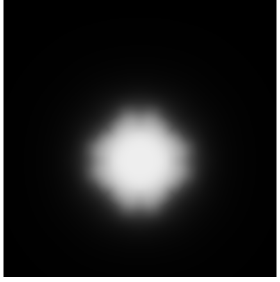
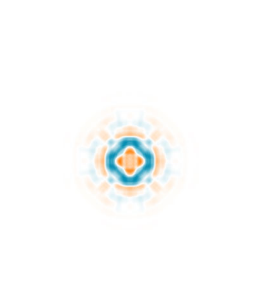
| | Reference | Scattered sampling | Error |
|---|---|---|---|
| $\pi/2$ | |  |  |
| $5\pi/8$ |  |  |  |
| $3\pi/4$ |  |  |  |
| $7\pi/8$ |  |  |  |
| $\pi$ |  |  |  |

Table B.20: Table showing the results for MIREL at height 5.1.

**INTRO**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 0.1 | 38.2276 | 39.4688 | -129.625 |

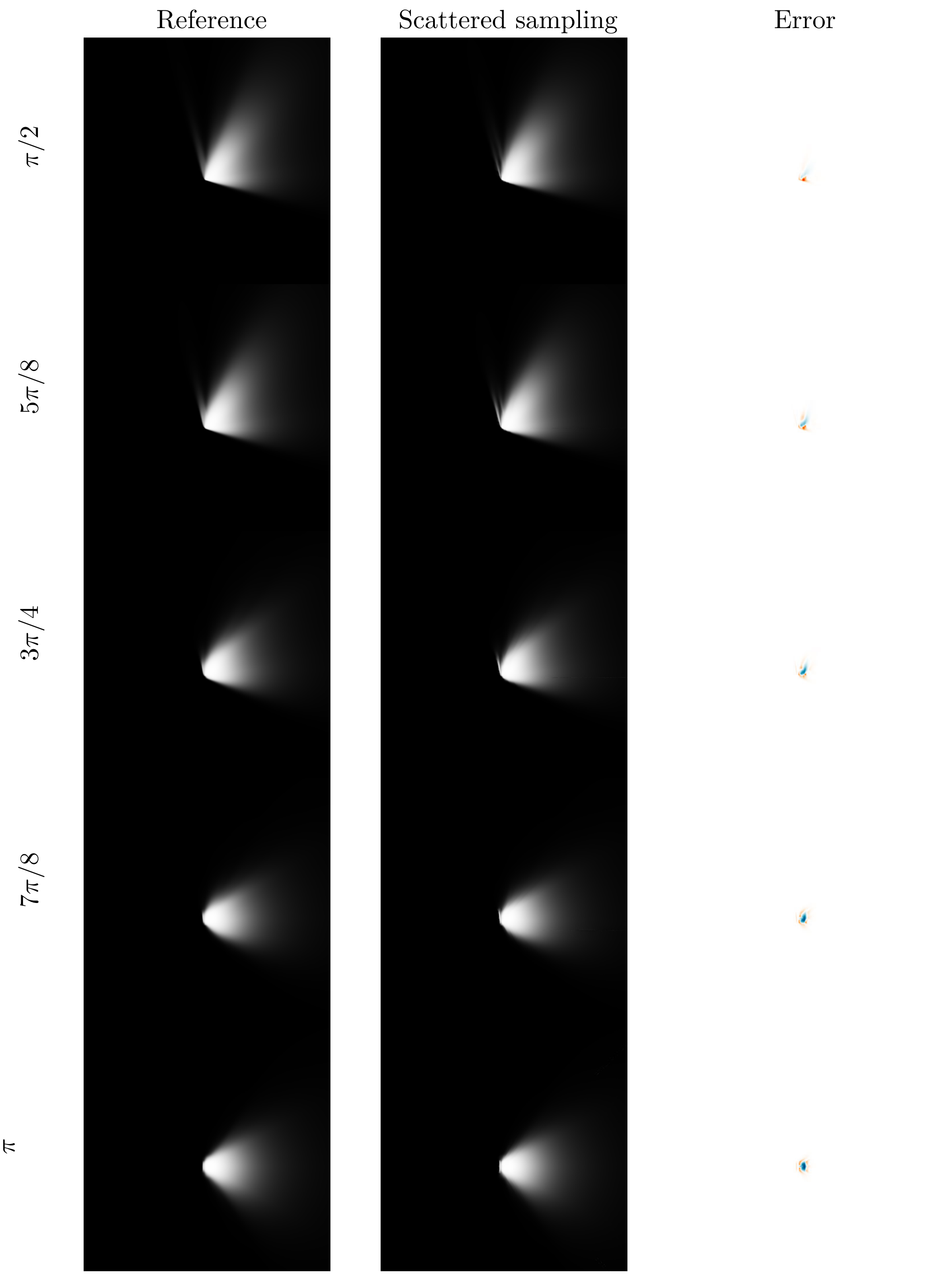


Table B.21: Table showing the results for INTRO at height 0.1.

**INTRO**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 1.1 | 2.6359 | 17.2813 | -33.25 |

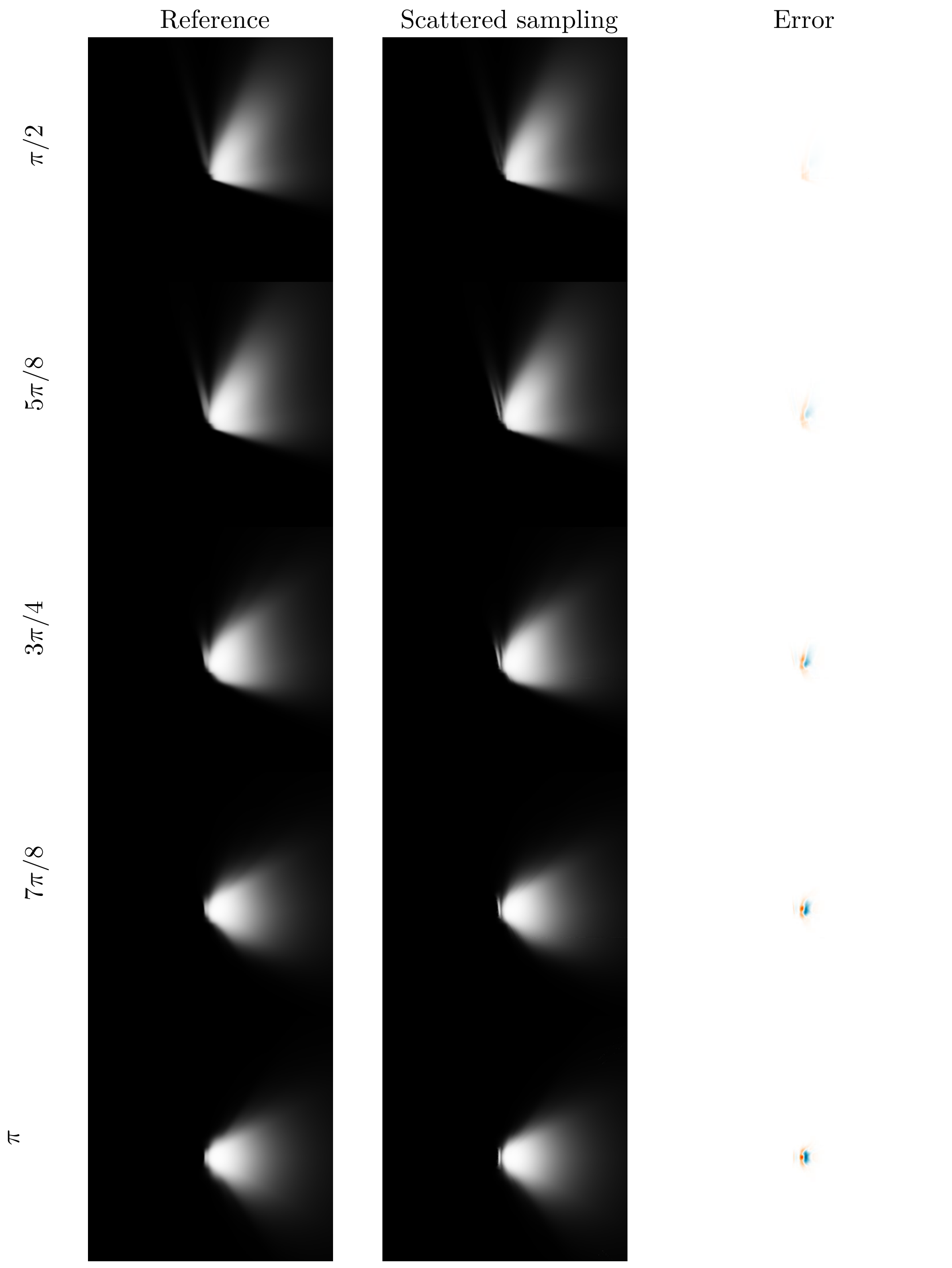


Table B.22: Table showing the results for INTRO at height 1.1.

**INTRO**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 3.1 | 0.13794 | 4.375 | -3.9375 |

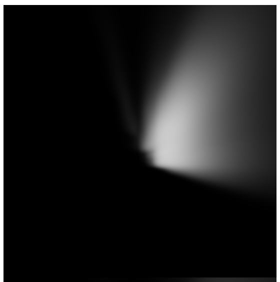
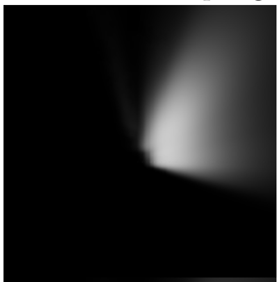
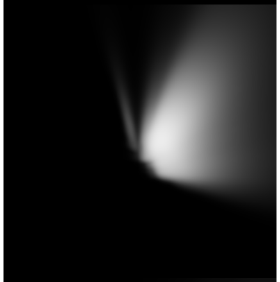
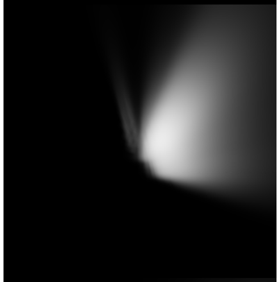
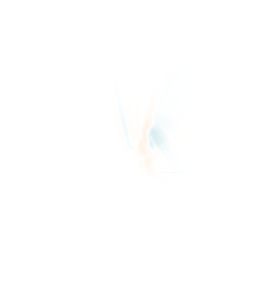
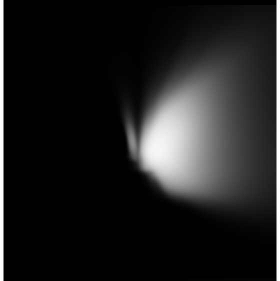
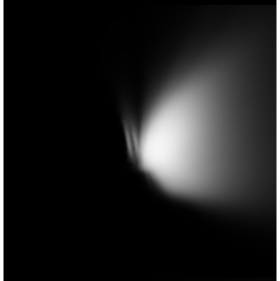
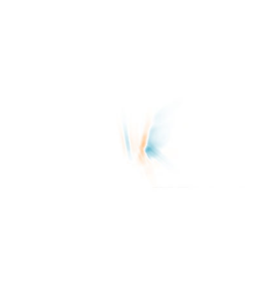
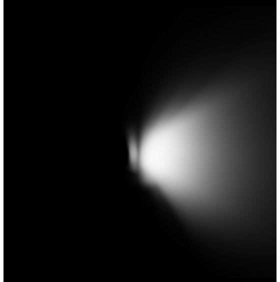
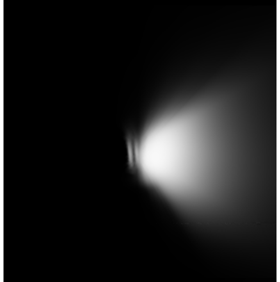
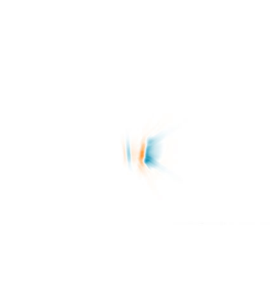
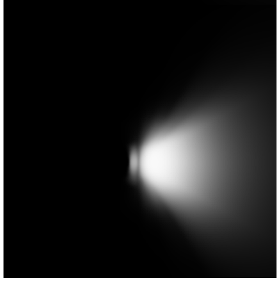
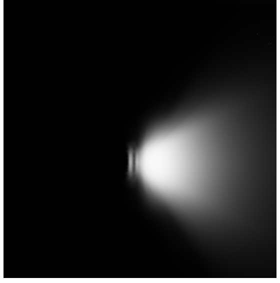
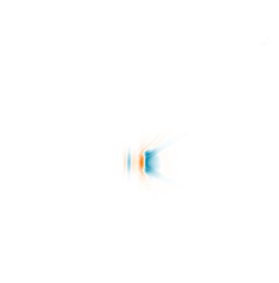
| | Reference | Scattered sampling | Error |
|---|---|---|---|
| $\pi/2$ |  |  | |
| $5\pi/8$ |  |  |  |
| $3\pi/4$ |  |  |  |
| $7\pi/8$ |  |  |  |
| $\pi$ |  |  |  |

Table B.23: Table showing the results for INTRO at height 3.1.

**INTRO**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 5.1 | 0.027428 | 1.5889 | -1.9375 |

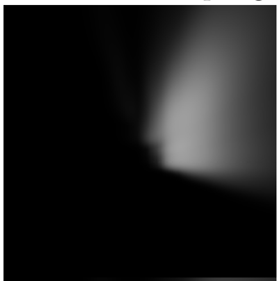
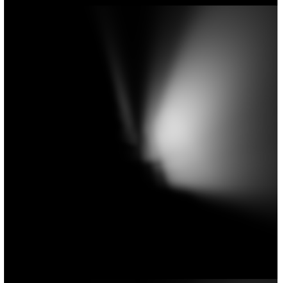
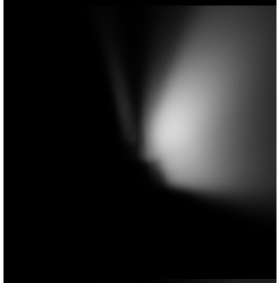
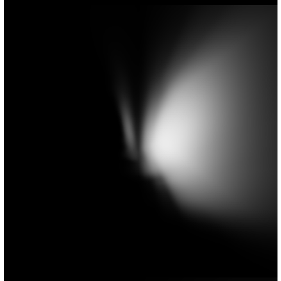
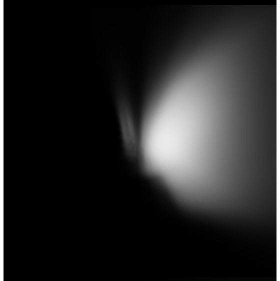
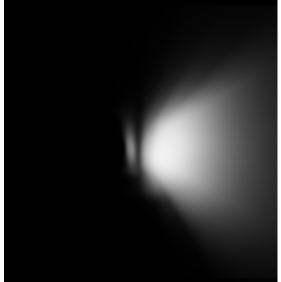
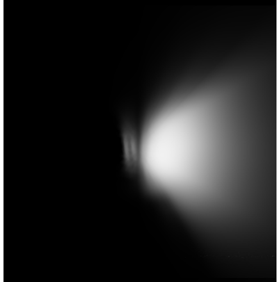
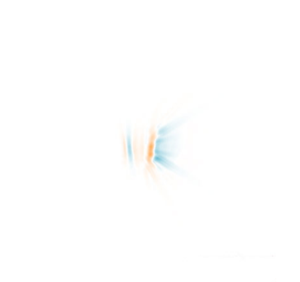
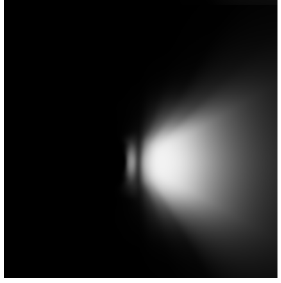
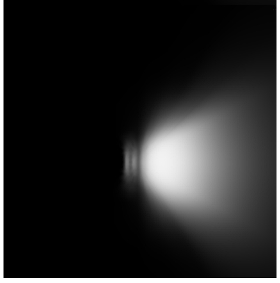
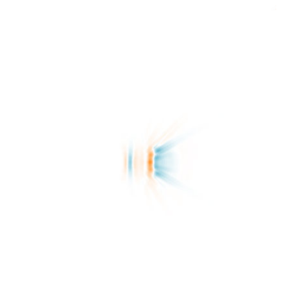
| | Reference | Scattered sampling | Error |
|---|---|---|---|
| $\pi/2$ | |  | |
| $5\pi/8$ |  |  | |
| $3\pi/4$ |  |  | |
| $7\pi/8$ |  |  |  |
| $\pi$ |  |  |  |

Table B.24: Table showing the results for INTRO at height 5.1.

**LINETIK-S**

| Height | NMSE | UEB | LEB |
|---|---|---|---|
| 0.1 | 17.6828 | 28.9531 | -105 |

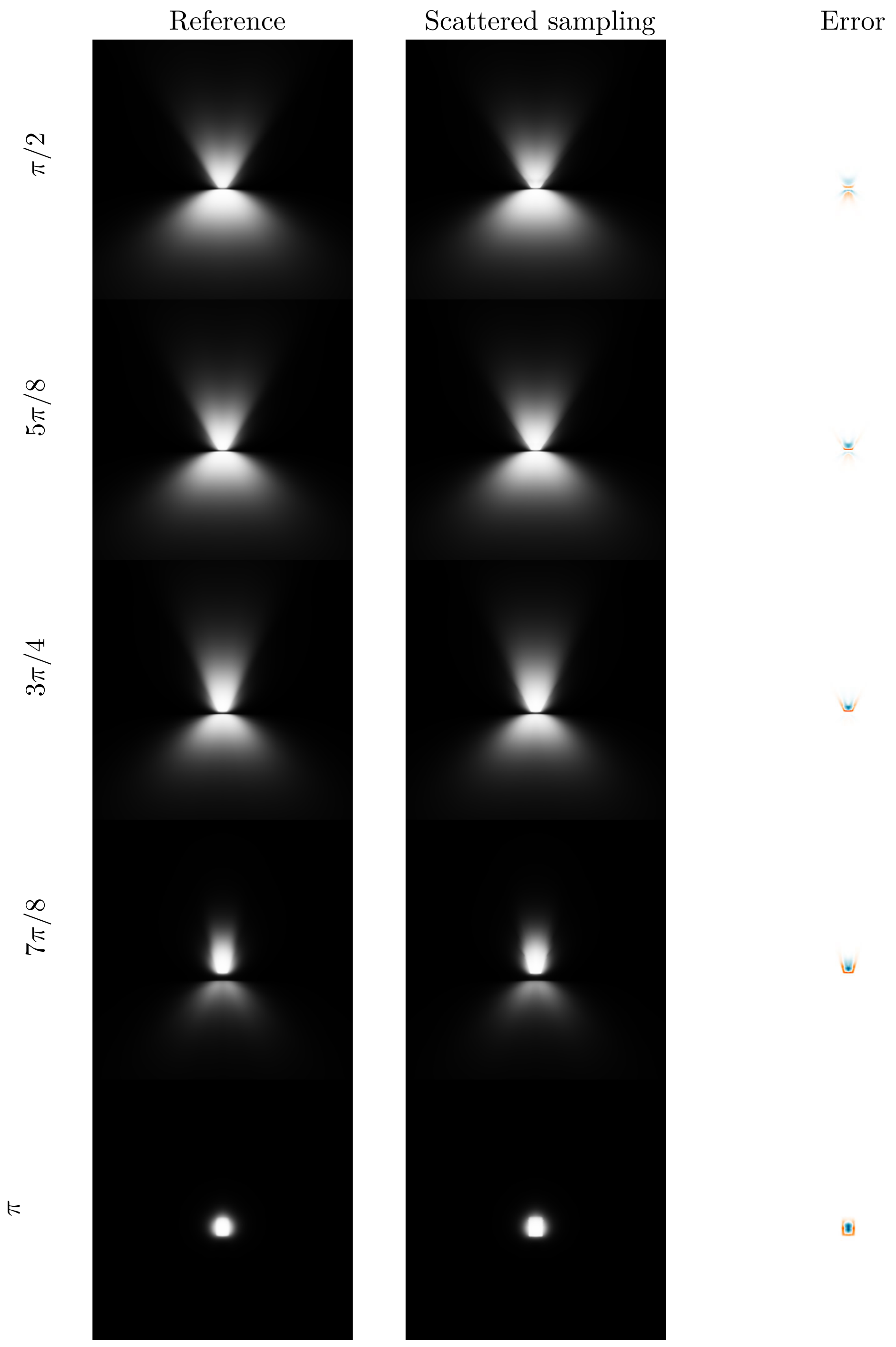


Table B.25: Table showing the results for LINETIK-S at height 0.1.

**LINETIK-S**

| Height | NMSE | UEB | LEB |
|---|---|---|---|
| 1.1 | 1.4551 | 7.1563 | -18.125 |

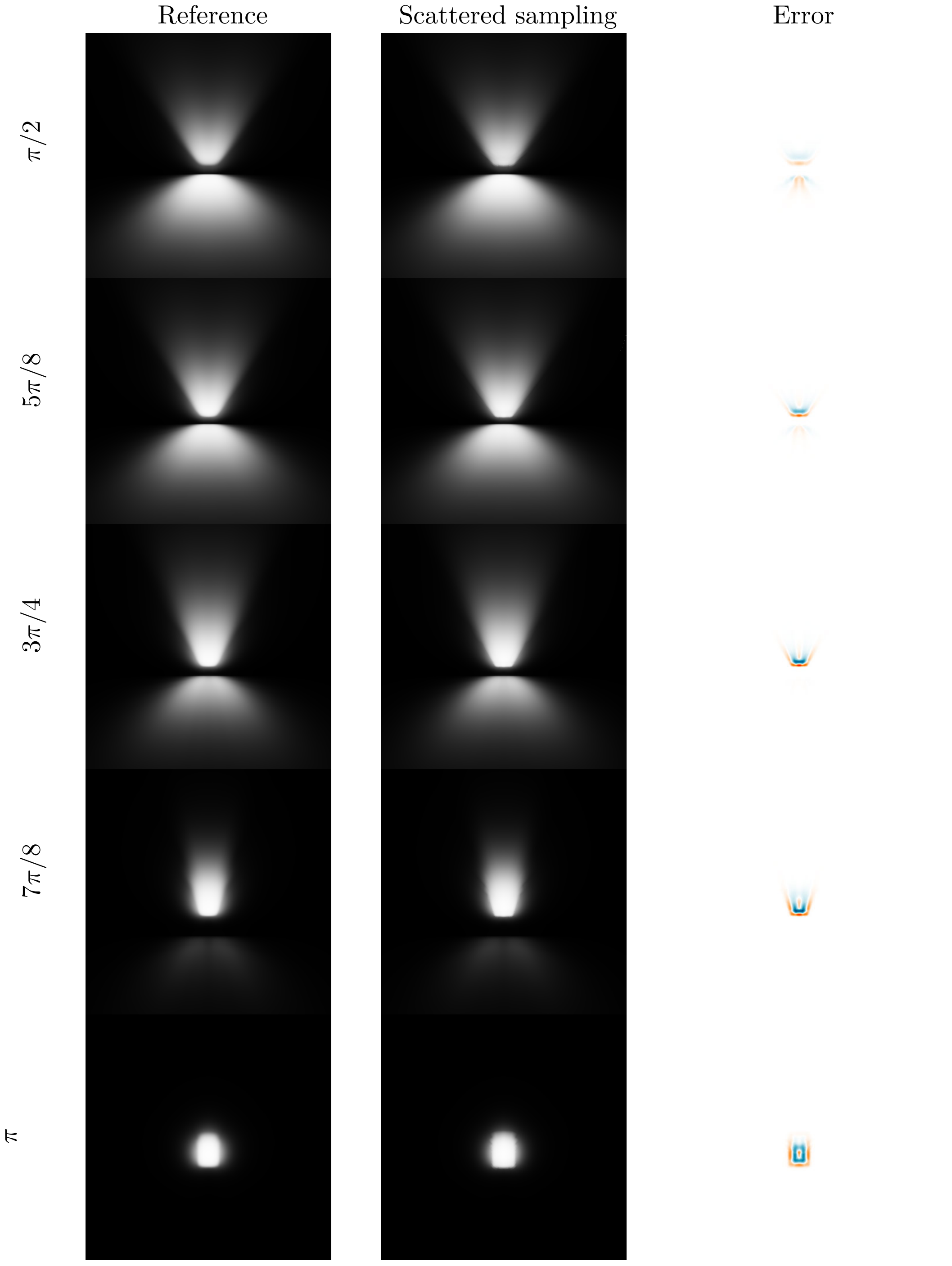


Table B.26: Table showing the results for LINETIK-S at height 1.1.

**LINETIK-S**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 3.1 | 0.044154 | 0.89063 | -1.5781 |

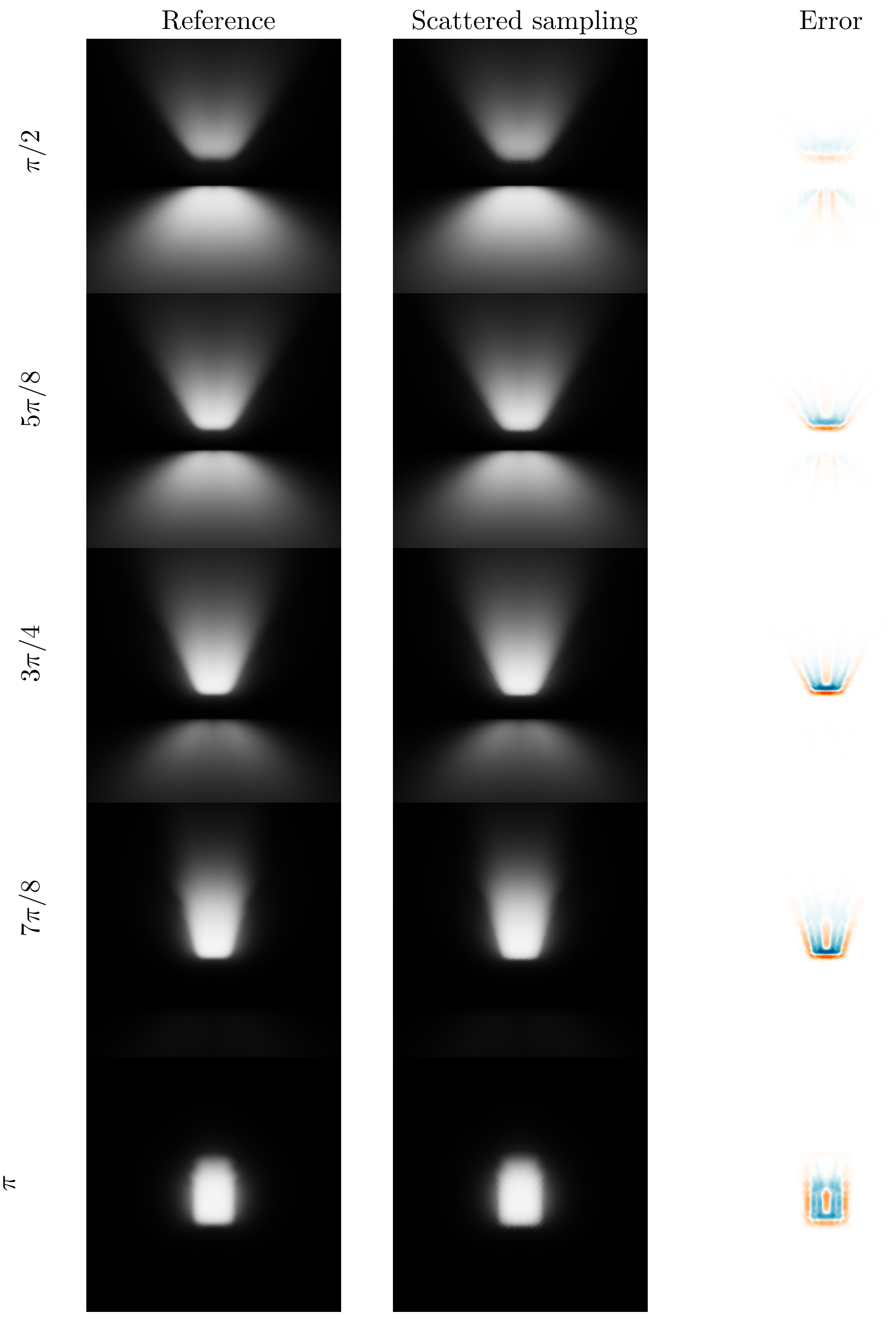


Table B.27: Table showing the results for LINETIK-S at height 3.1.

**LINETIK-S**

| **Height** | **NMSE** | **UEB** | **LEB** |
|---|---|---|---|
| 5.1 | 0.0055244 | 0.27539 | -0.40625 |

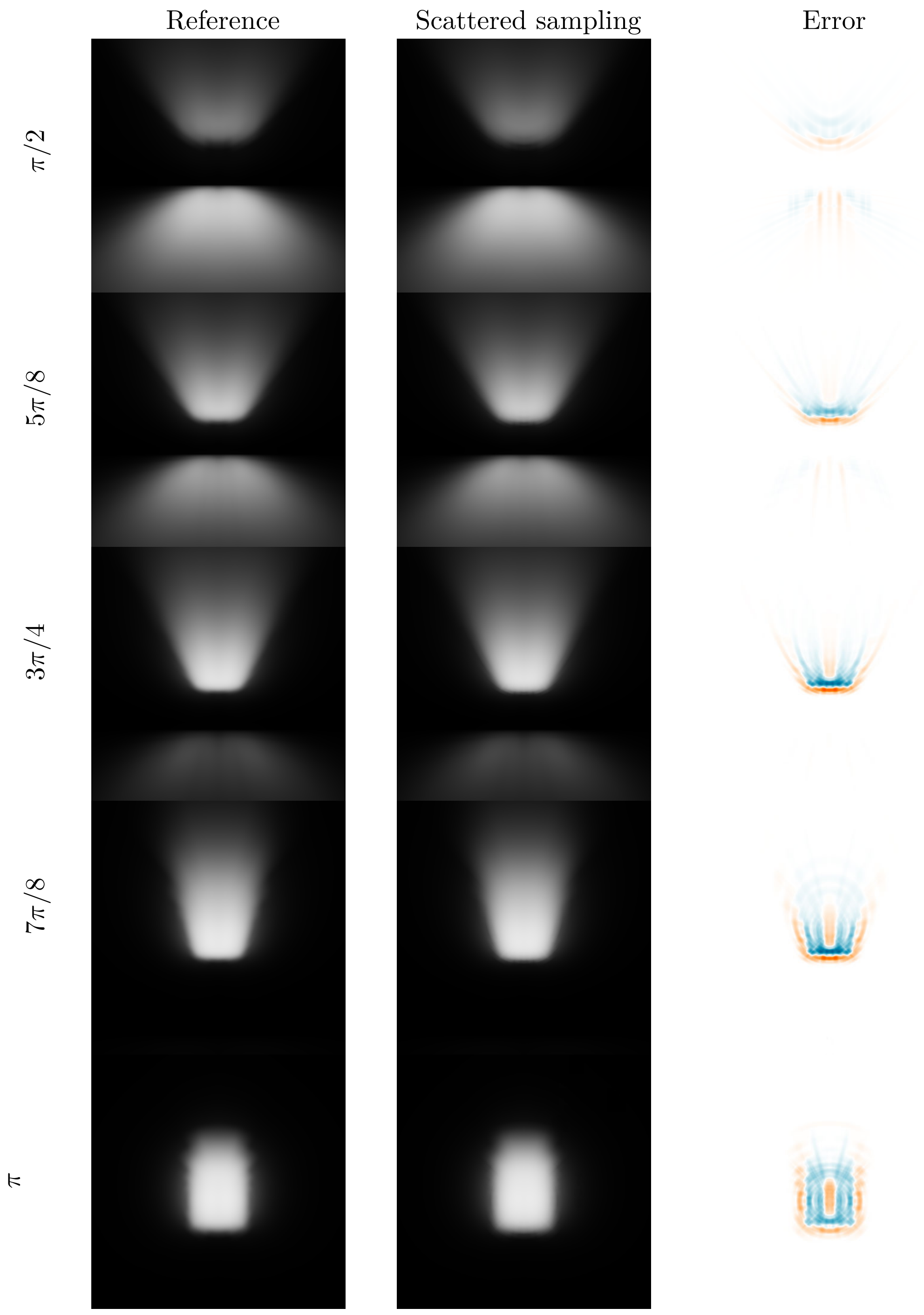


Table B.28: Table showing the results for LINETIK-S at height 5.1.

APPENDIX C

# Error Image Comparison

This chapter presents the error images of scattered sampling (SCT) and Monte Carlo integration with light-source importance sampling with 24 or 40 Poisson samples (MCI-24, MCI-40) next to each other for comparison. The error images visualize the approximation errors for the luminaires presented in Section 6.3 for various heights and rotation steps (see Section 6.2). The error bounds for the error images of each approximations are given by the upper error bound (UEB) and the lower error bound (LEB).

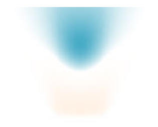
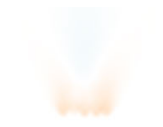
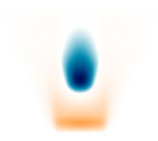
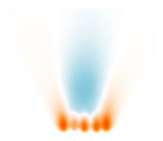
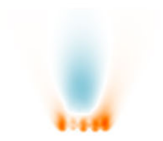
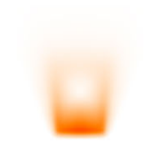
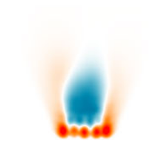
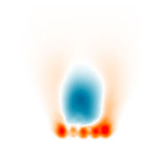
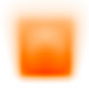
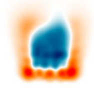
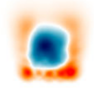
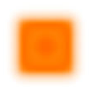
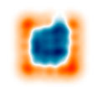
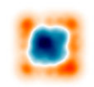
| **ARCOS** | Height: 0.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 1.60e+02 | 3.10e+01 | 1.88e+01 |
| **LEB** | -8.44e+00 | -4.02e+01 | -2.91e+01 |
| $\frac{\pi}{2}$ |  |  | |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.1: Table showing the error images for ARCOS at height 0.1.

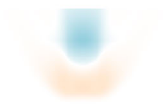
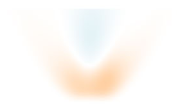
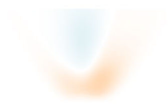
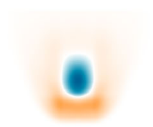
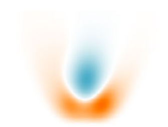
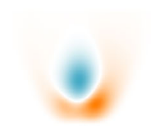
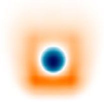
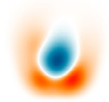
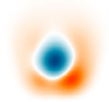
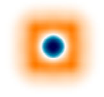
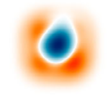
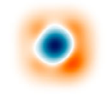
| **ARCOS** | Height: 1.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 2.71e+01 | 5.12e+00 | 3.77e+00 |
| **LEB** | -1.18e+01 | -1.54e+01 | -1.01e+01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.2: Table showing the error images for ARCOS at height 1.1.

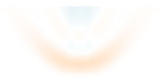
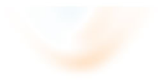
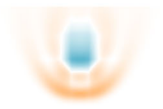
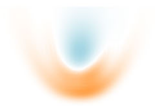
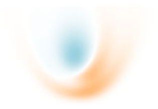
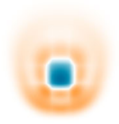
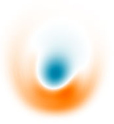
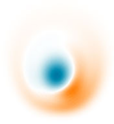
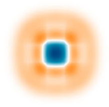
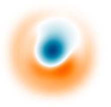
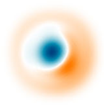
| **ARCOS** | Height: 3.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 2.40e+00 | 6.56e-01 | 5.08e-01 |
| **LEB** | -4.03e+00 | -2.44e+00 | -1.47e+00 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  | |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.3: Table showing the error images for ARCOS at height 3.1.

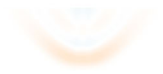
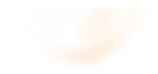
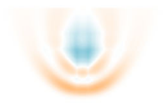
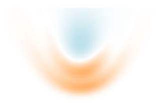
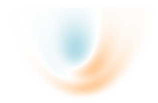
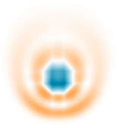
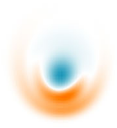
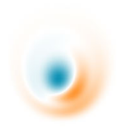
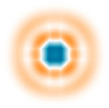
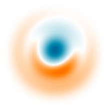
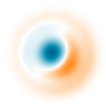
| **ARCOS** | Height: 5.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 5.86e-01 | 2.11e-01 | 1.68e-01 |
| **LEB** | -1.33e+00 | -6.72e-01 | -3.91e-01 |
| $\frac{\pi}{2}$ | | | |
| $\frac{5\pi}{8}$ |  | |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.4: Table showing the error images for ARCOS at height 5.1.

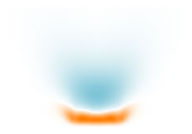
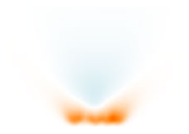
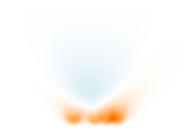
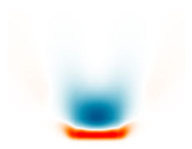
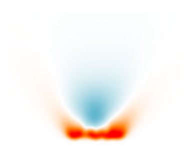
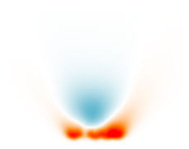
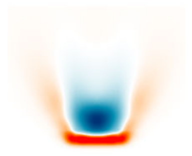
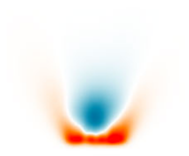
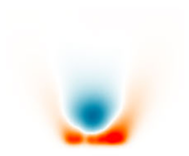
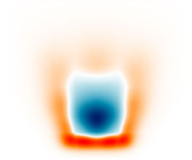
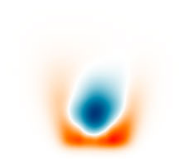
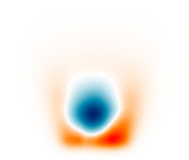
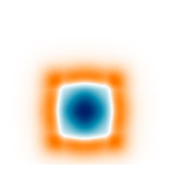
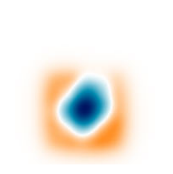
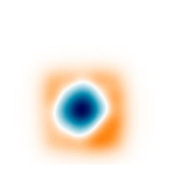
| **PANOS** | Height: 0.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 1.06e+01 | 1.60e+01 | 9.38e+00 |
| **LEB** | -3.64e+01 | -2.81e+01 | -1.80e+01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.5: Table showing the error images for PANOS at height 0.1.

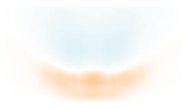
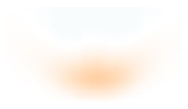
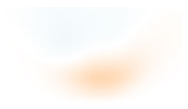
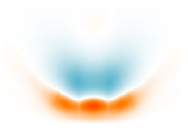
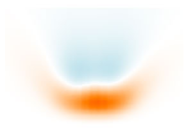
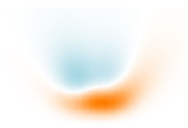
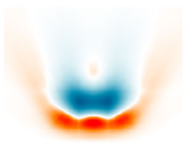
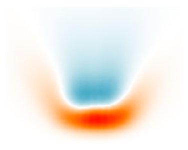
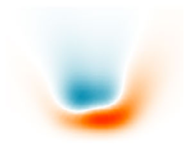
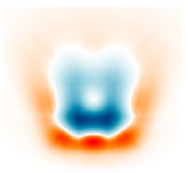
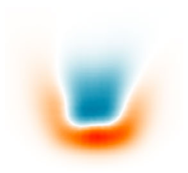
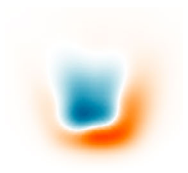
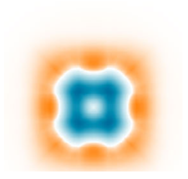
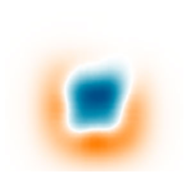
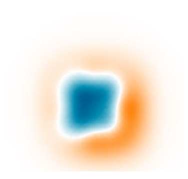
| **PANOS** | Height: 1.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 1.16e+00 | 1.53e+00 | 9.38e-01 |
| **LEB** | -6.53e+00 | -2.81e+00 | -1.81e+00 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.6: Table showing the error images for PANOS at height 1.1.

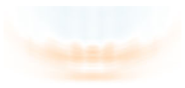
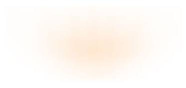
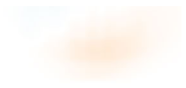
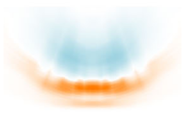
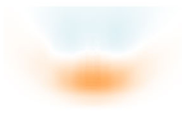
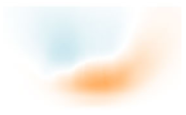
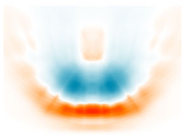
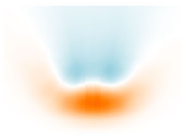
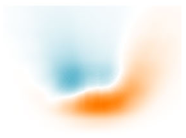
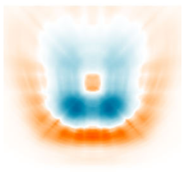
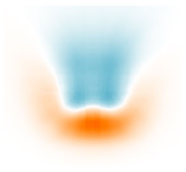
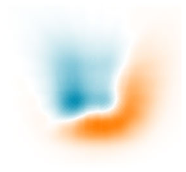
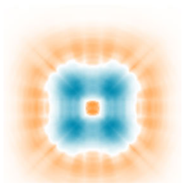
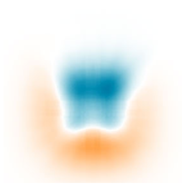
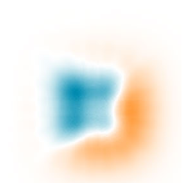
| **PANOS** | Height: 3.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 1.02e-01 | 2.23e-01 | 1.07e-01 |
| **LEB** | -6.09e-01 | -2.89e-01 | -1.88e-01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.7: Table showing the error images for PANOS at height 3.1.

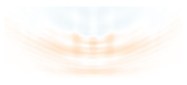
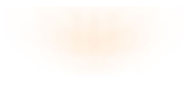
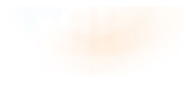
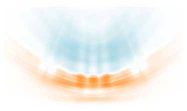
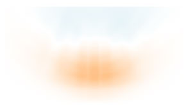
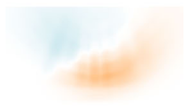
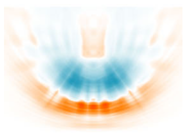
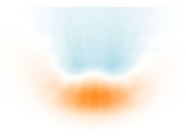
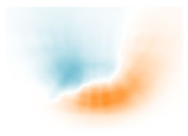
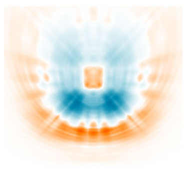
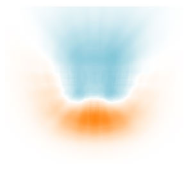
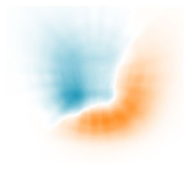
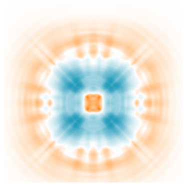
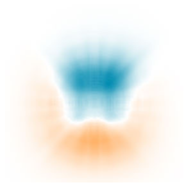
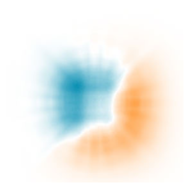
| **PANOS** | Height: 5.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 2.64e-02 | 7.42e-02 | 3.12e-02 |
| **LEB** | -1.37e-01 | -7.81e-02 | -4.69e-02 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.8: Table showing the error images for PANOS at height 5.1.

**SLOTLIGHT** Height: 0.1

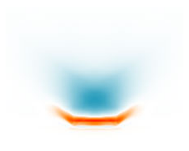
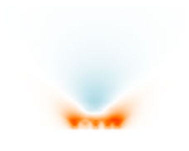
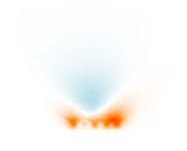
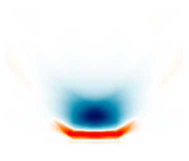
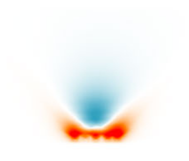
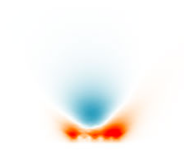
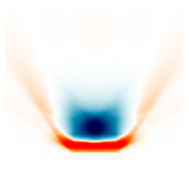
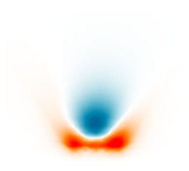
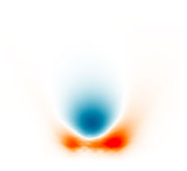
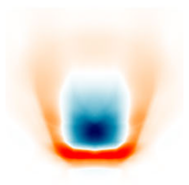
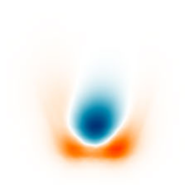
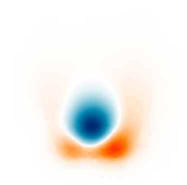
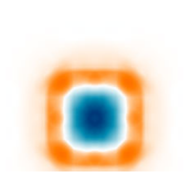
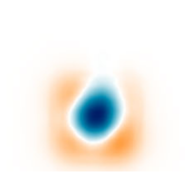
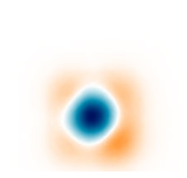
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 8.44e+00 | 3.36e+01 | 2.09e+01 |
| **LEB** | -4.90e+01 | -4.85e+01 | -3.15e+01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.9: Table showing the error images for SLOTLIGHT at height 0.1.

**SLOTLIGHT** Height: 1.1

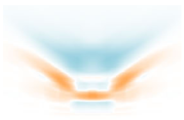
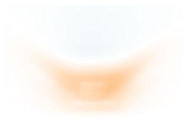
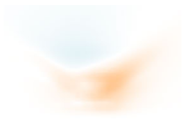
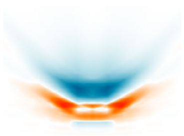
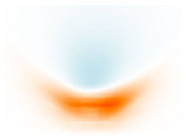
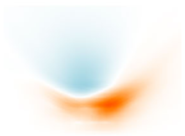
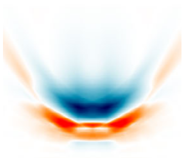
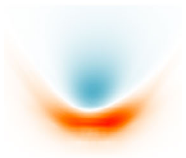
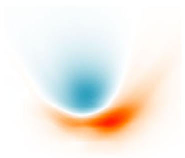
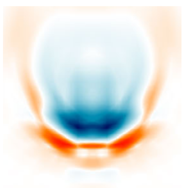
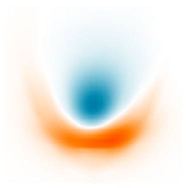
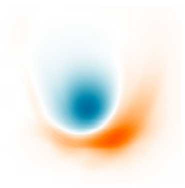
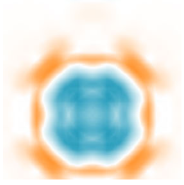
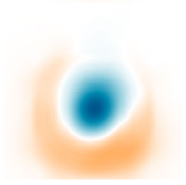
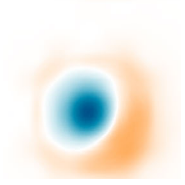
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 1.56e+00 | 2.64e+00 | 1.67e+00 |
| **LEB** | -6.06e+00 | -5.25e+00 | -3.12e+00 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.10: Table showing the error images for SLOTLIGHT at height 1.1.

**SLOTLIGHT** Height: 3.1

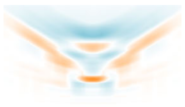
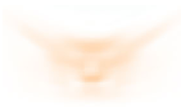
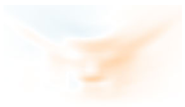
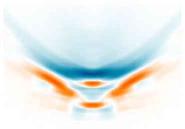
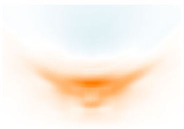
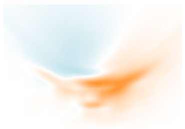
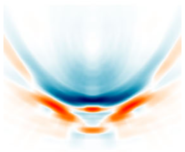
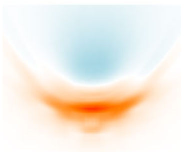
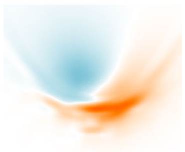
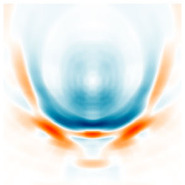
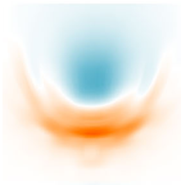
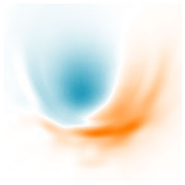
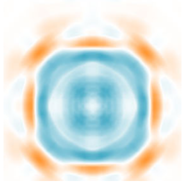
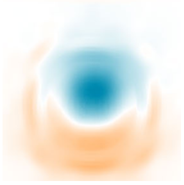
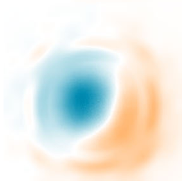
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 2.50e-01 | 3.28e-01 | 1.84e-01 |
| **LEB** | -5.31e-01 | -4.84e-01 | -2.81e-01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.11: Table showing the error images for SLOTLIGHT at height 3.1.

**SLOTLIGHT** Height: 5.1

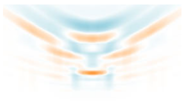
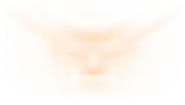
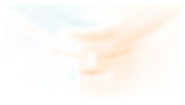
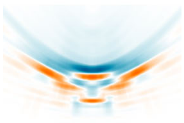
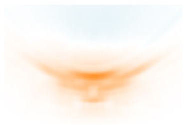
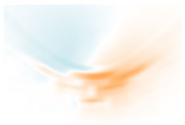
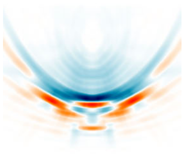
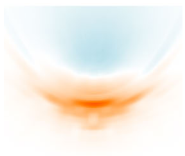
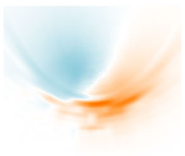
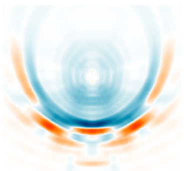
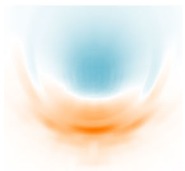
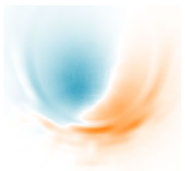
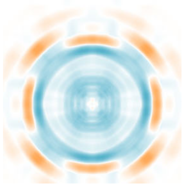
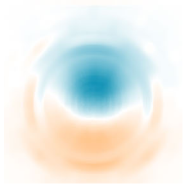
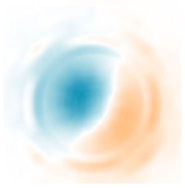
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 8.20e-02 | 1.13e-01 | 5.86e-02 |
| **LEB** | -1.48e-01 | -1.25e-01 | -7.03e-02 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.12: Table showing the error images for SLOTLIGHT at height 5.1.

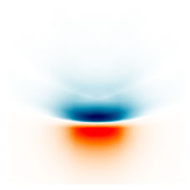
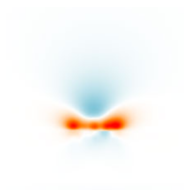
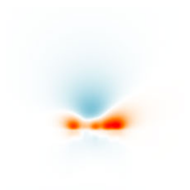
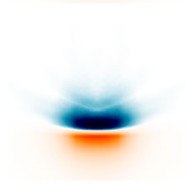
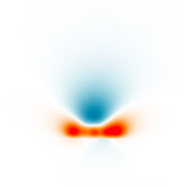
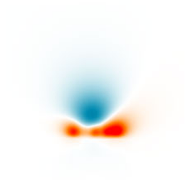
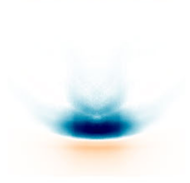
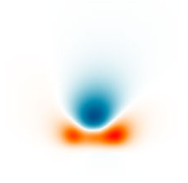
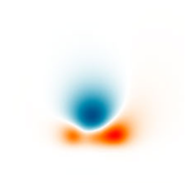
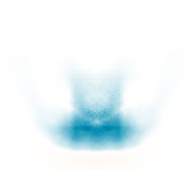
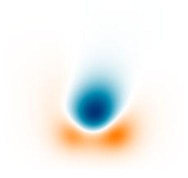
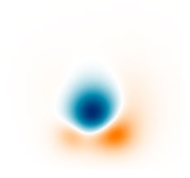
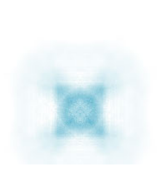
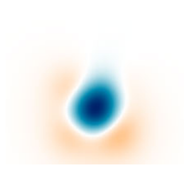
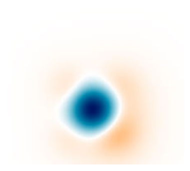
| **PERLUCE** | Height: 0.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 1.88e+01 | 4.75e+01 | 2.85e+01 |
| **LEB** | -4.25e+00 | -4.75e+01 | -3.08e+01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.13: Table showing the error images for PERLUCE at height 0.1.

**PERLUCE** Height: 1.1

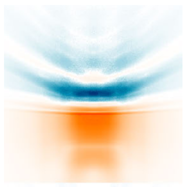
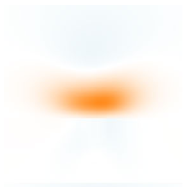
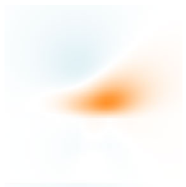
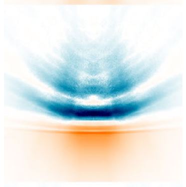
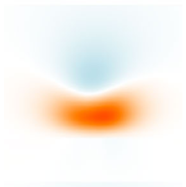
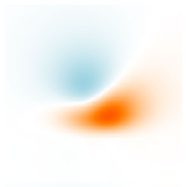
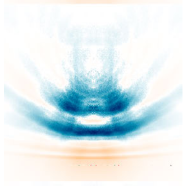
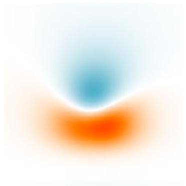
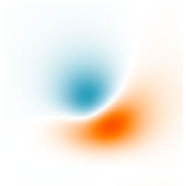
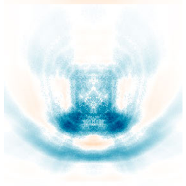
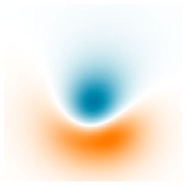
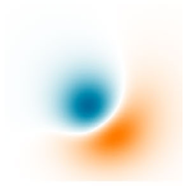
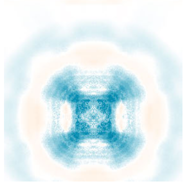
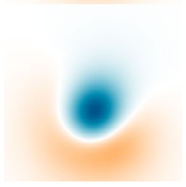
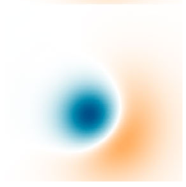
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 6.45e-01 | 2.66e+00 | 1.56e+00 |
| **LEB** | -2.54e-01 | -4.88e+00 | -2.88e+00 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.14: Table showing the error images for PERLUCE at height 1.1.

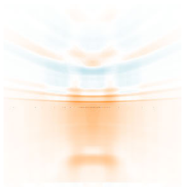
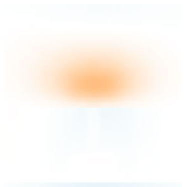
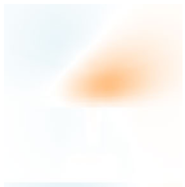
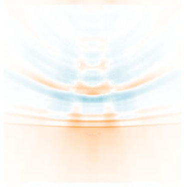
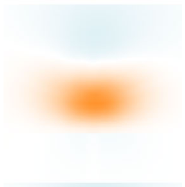
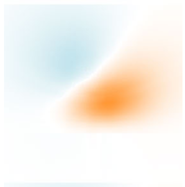
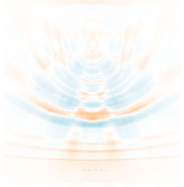
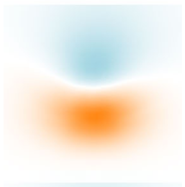
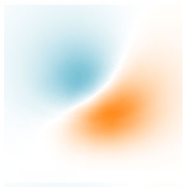
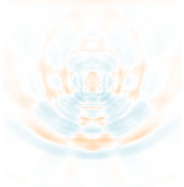
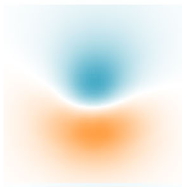
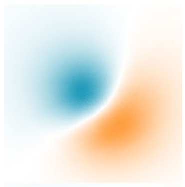
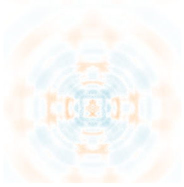
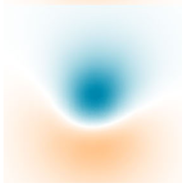
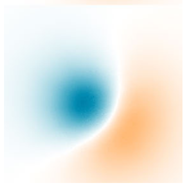
| **PERLUCE** | Height: 3.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 7.42e-02 | 3.59e-01 | 1.72e-01 |
| **LEB** | -1.78e-01 | -4.53e-01 | -2.50e-01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.15: Table showing the error images for PERLUCE at height 3.1.

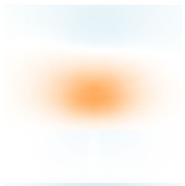
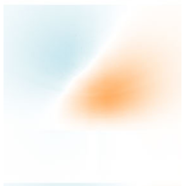
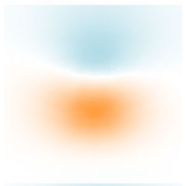
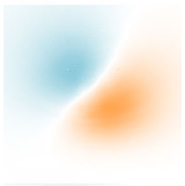
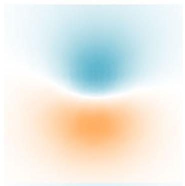
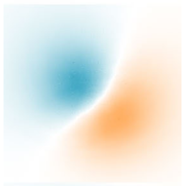
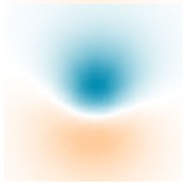
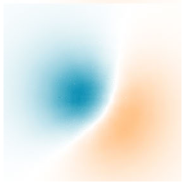
| **PERLUCE** | Height: 5.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 1.95e-01 | 1.17e-01 | 5.47e-02 |
| **LEB** | -1.63e-01 | -1.09e-01 | -6.25e-02 |
| $\frac{\pi}{2}$ | | | |
| $\frac{5\pi}{8}$ | |  |  |
| $\frac{3\pi}{4}$ | |  |  |
| $\frac{7\pi}{4}$ | |  |  |
| $\pi$ | |  |  |

Table C.16: Table showing the error images for PERLUCE at height 5.1.

**MIREL** Height: 0.1

| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 3.03e+01 | 3.71e+01 | 2.38e+01 |
| **LEB** | -1.30e+02 | -6.98e+01 | -4.02e+01 |

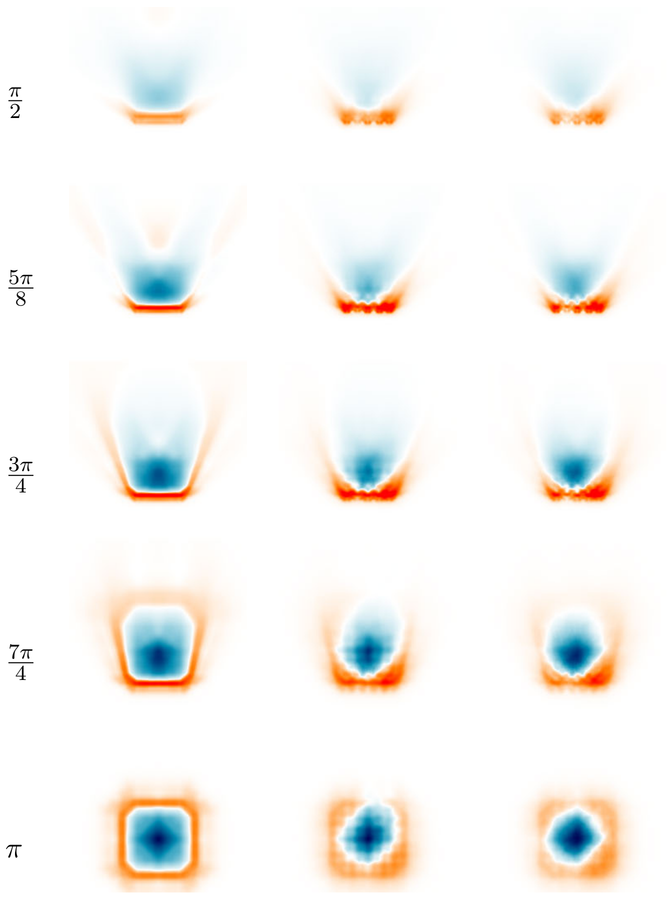


Table C.17: Table showing the error images for MIREL at height 0.1.

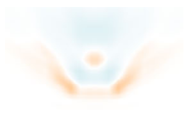
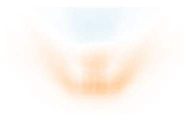
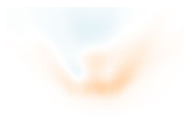
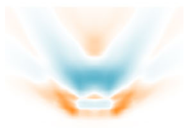
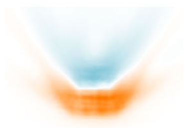
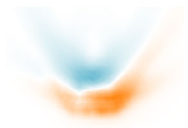
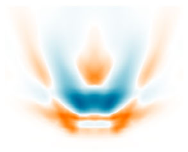
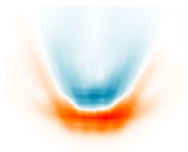
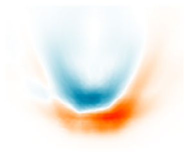
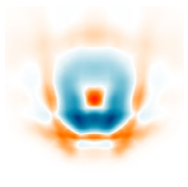
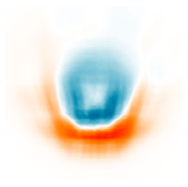
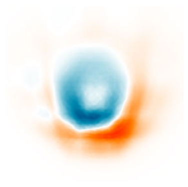
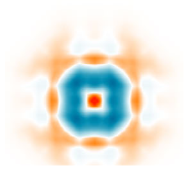
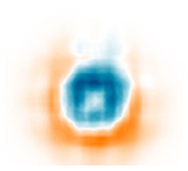
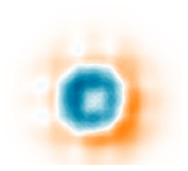
| **MIREL** | Height: 1.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 5.62e+00 | 3.56e+00 | 2.39e+00 |
| **LEB** | -2.32e+01 | -6.00e+00 | -4.12e+00 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.18: Table showing the error images for MIREL at height 1.1.

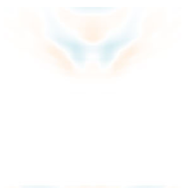
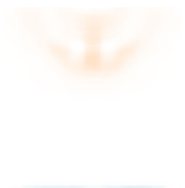
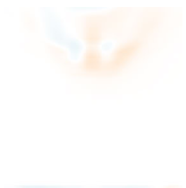
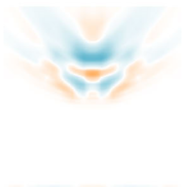
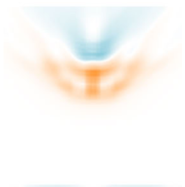
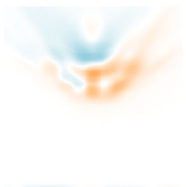
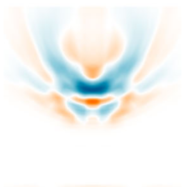
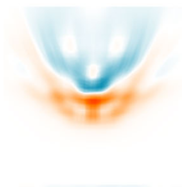
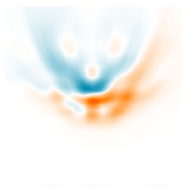
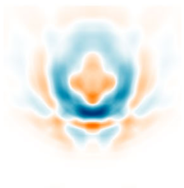
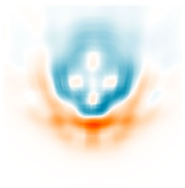
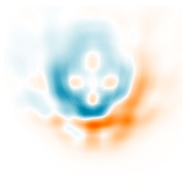
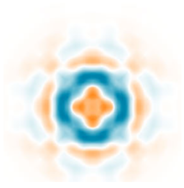
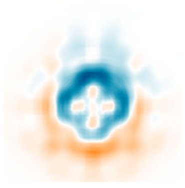
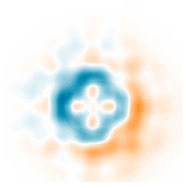
| **MIREL** | Height: 3.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 1.62e+00 | 6.56e-01 | 3.67e-01 |
| **LEB** | -2.89e+00 | -6.56e-01 | -5.00e-01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.19: Table showing the error images for MIREL at height 3.1.

**MIREL** Height: 5.1

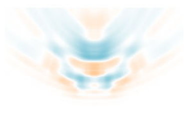
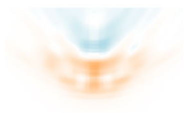
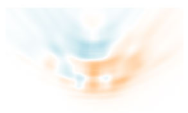
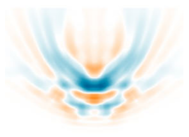
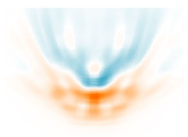
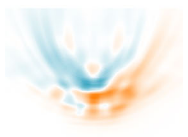
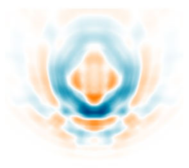
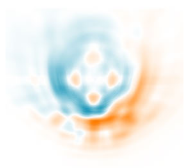
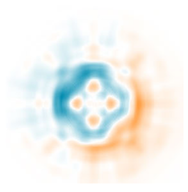
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 4.77e-01 | 2.27e-01 | 1.17e-01 |
| **LEB** | -7.34e-01 | -1.95e-01 | -1.48e-01 |
| $\frac{\pi}{2}$ | | | |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  | |  |
| $\pi$ | | |  |

Table C.20: Table showing the error images for MIREL at height 5.1.

**INTRO** Height: 0.1

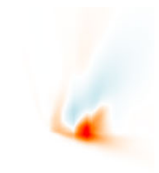
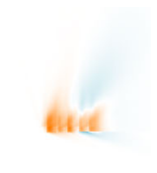
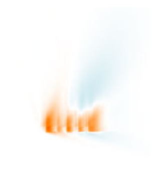
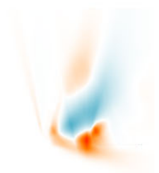
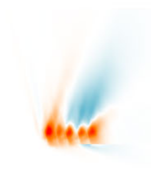
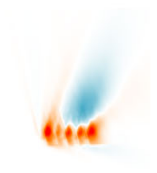
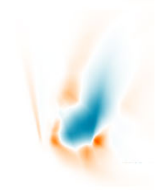
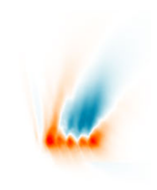
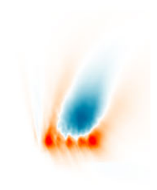
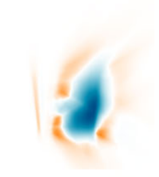
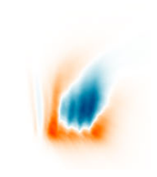
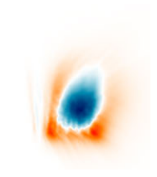
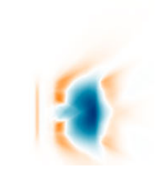
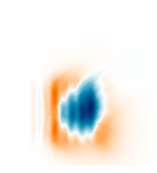
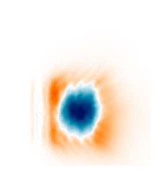
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 3.95e+01 | 3.77e+01 | 1.79e+01 |
| **LEB** | -1.30e+02 | -5.74e+01 | -3.39e+01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.21: Table showing the error images for INTRO at height 0.1.

**INTRO** Height: 1.1

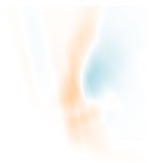
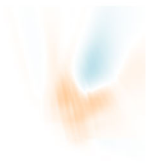
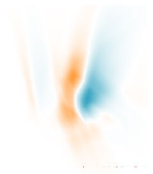
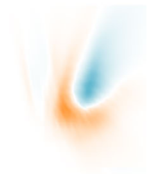
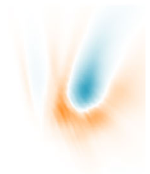
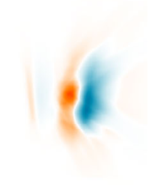
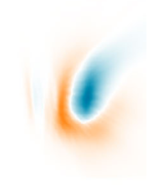
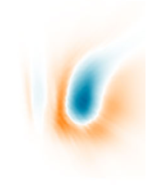
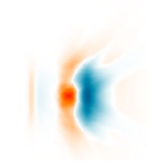
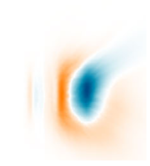
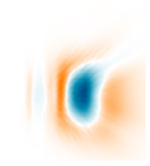
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 1.73e+01 | 7.00e+00 | 2.76e+00 |
| **LEB** | -3.32e+01 | -1.29e+01 | -7.81e+00 |
| $\frac{\pi}{2}$ | | | |
| $\frac{5\pi}{8}$ |  | |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.22: Table showing the error images for INTRO at height 1.1.

**INTRO** Height: 3.1

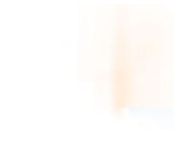
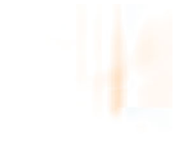
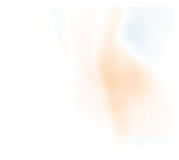
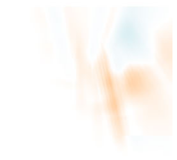
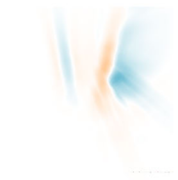
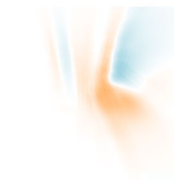
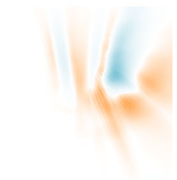
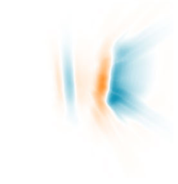
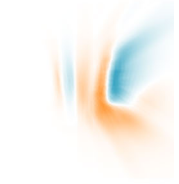
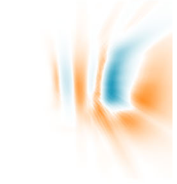
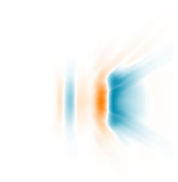
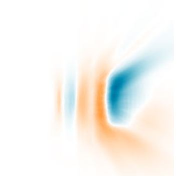
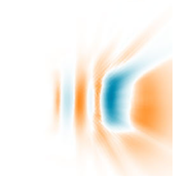
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 4.38e+00 | 8.28e-01 | 2.20e-01 |
| **LEB** | -3.94e+00 | -1.56e+00 | -1.16e+00 |
| $\frac{\pi}{2}$ | |  |  |
| $\frac{5\pi}{8}$ | |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.23: Table showing the error images for INTRO at height 3.1.

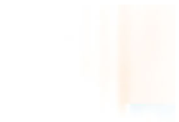
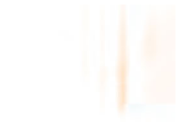
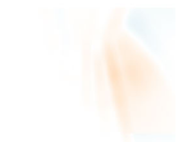
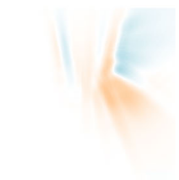
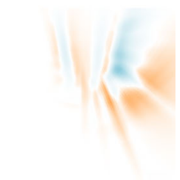
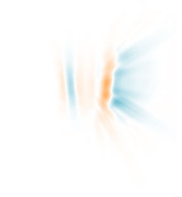
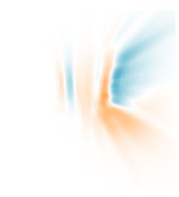
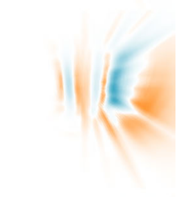
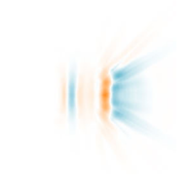
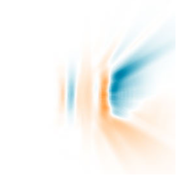
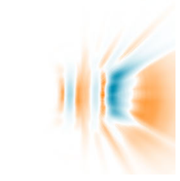
| **INTRO** | Height: 5.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 1.59e+00 | 3.05e-01 | 6.25e-02 |
| **LEB** | -1.94e+00 | -4.14e-01 | -3.44e-01 |
| $\frac{\pi}{2}$ | |  |  |
| $\frac{5\pi}{8}$ | |  | |
| $\frac{3\pi}{4}$ | |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.24: Table showing the error images for INTRO at height 5.1.

**LINETIK-S** Height: 0.1

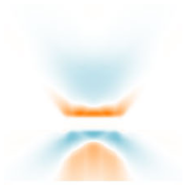
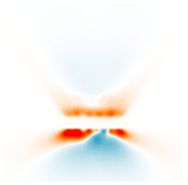
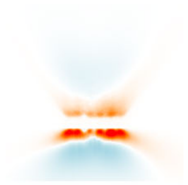
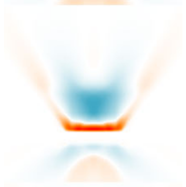
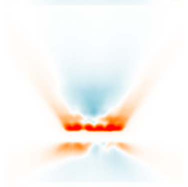
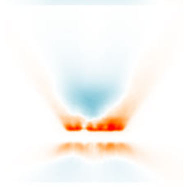
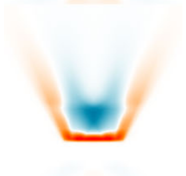
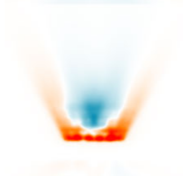
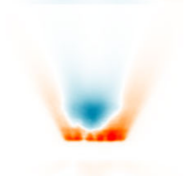
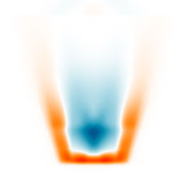
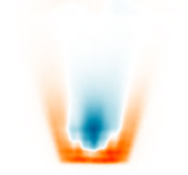
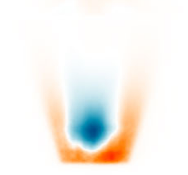
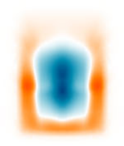
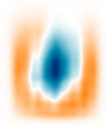
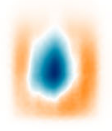
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 2.90e+01 | 2.68e+01 | 1.78e+01 |
| **LEB** | -1.05e+02 | -5.65e+01 | -3.35e+01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.25: Table showing the error images for LINETIK-S at height 0.1.

|  | **LINETIK-S** | Height: 1.1 |  |
|---|---|---|---|
|  | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 7.16e+00 | 3.33e+00 | 2.42e+00 |
| **LEB** | -1.81e+01 | -4.94e+00 | -3.44e+00 |

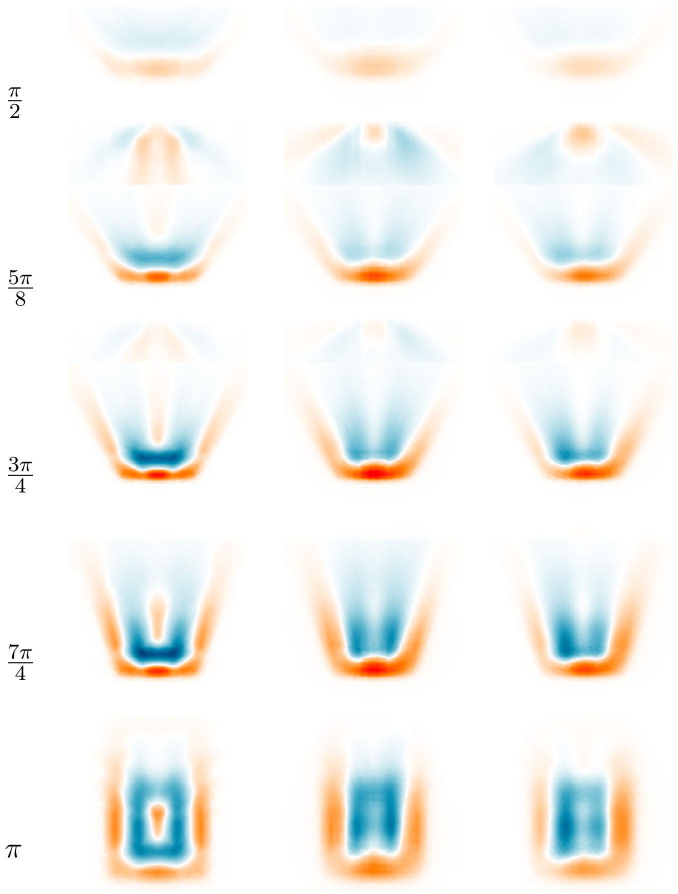


Table C.26: Table showing the error images for LINETIK-S at height 1.1.

**LINETIK-S** Height: 3.1

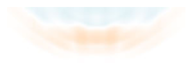
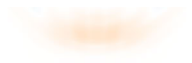
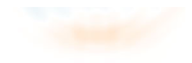
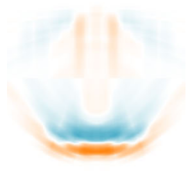
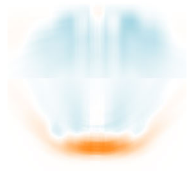
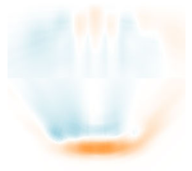
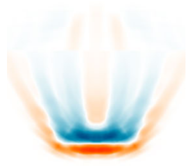
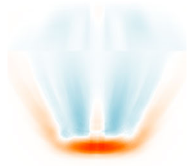
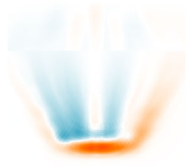
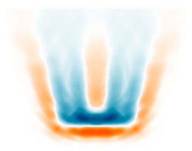
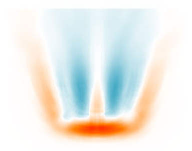
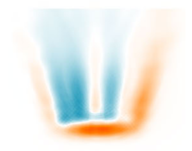
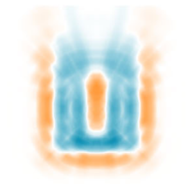
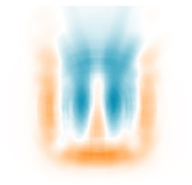
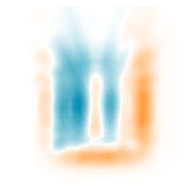
| | **SCT** | **MCI-24** | **MCI-40** |
|---|---|---|---|
| **UEB** | 8.91e-01 | 4.73e-01 | 2.89e-01 |
| **LEB** | -1.58e+00 | -4.84e-01 | -3.28e-01 |
| $\frac{\pi}{2}$ |  |  |  |
| $\frac{5\pi}{8}$ |  |  |  |
| $\frac{3\pi}{4}$ |  |  |  |
| $\frac{7\pi}{4}$ |  |  |  |
| $\pi$ |  |  |  |

Table C.27: Table showing the error images for LINETIK-S at height 3.1.

| **LINETIK-S** | Height: 5.1 | | |
|---|---|---|---|
| | **SCT** | **MCI-24** | **MCI-40** |
| **UEB** | 2.75e-01 | 1.52e-01 | 7.42e-02 |
| **LEB** | -4.06e-01 | -1.09e-01 | -7.81e-02 |

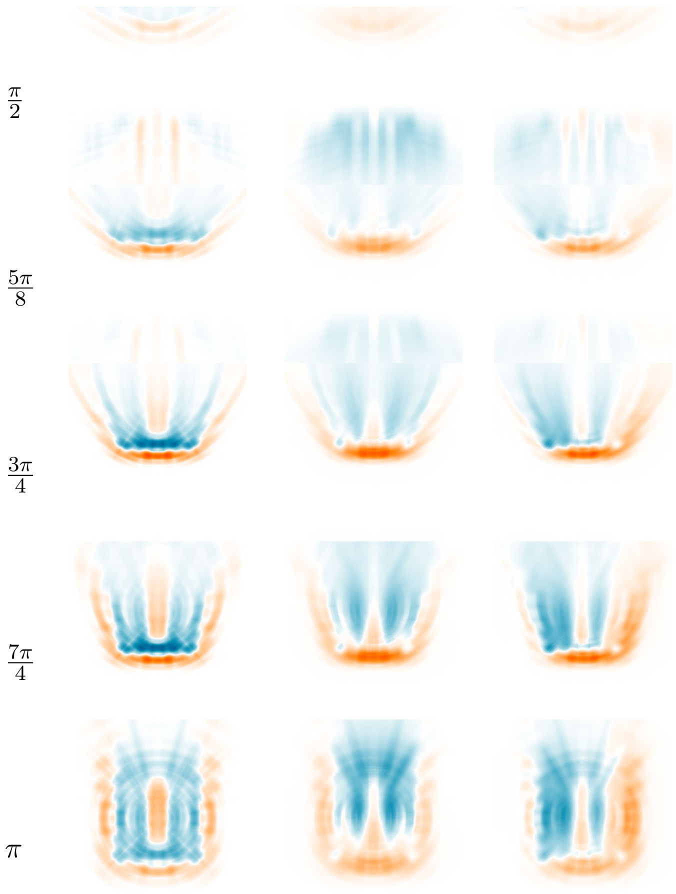


Table C.28: Table showing the error images for LINETIK-S at height 5.1.

# List of Figures

# List of Tables

# List of Algorithms

# Bibliography

[And99] Eric C Anderson. Monte Carlo Methods and Importance Sampling, October 1999.

[AR98] Ian Ashdown and Ron Rykowski. Making Near-Field Photometry Practical. *Journal of the Illuminating Engineering Society*, 27(1):67–79, 1998.

[Arv95a] James Arvo. Applications of Irradiance Tensors to the Simulation of non-Lambertian Phenomena. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 335–342, New York, NY, USA, 1995. ACM.

[Arv95b] James Arvo. Stratified Sampling of Spherical Triangles. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 437–438, New York, NY, USA, 1995. ACM.

[Ash93] I. Ashdown. Near-Field Photometry: A New Approach. *Journal of the Illuminating Engineering Society*, 22(1):163–180, January 1993.

[Ash95] Ian Ashdown. Near-field photometry: Measuring and Modeling Complex 3-D Light Sources. *ACM SIGGRAPH'95 Course Notes-Realistic Input for Realistic Images*, pages 1–15, 1995.

[Aur91] Franz Aurenhammer. Voronoi Diagrams—a Survey of a Fundamental Geometric Data Structure. *ACM Comput. Surv.*, 23(3):345–405, September 1991.

[Bau72] Bruce G Baumgart. Winged edge polyhedron representation. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1972.

[BE70] Marshall Bern and David Eppstein. Mesh Generation And Optimal Triangulation. February 1970.

[Bri11] Brian Sharpe. A fast and simple 32bit floating point hash function. https://briansharpe.wordpress.com/2011/11/15/a-fast-and-simple-32bit-floating-point-hash-function/, November 2011.

[BRW89] 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*, SIGGRAPH '89, pages 325–334, New York, NY, USA, 1989. ACM.

[BSBK02] Mario Botsch, Stephan Steinberg, Stephan Bischoff, and Leif Kobbelt. Openmesh-a generic and efficient polygon mesh data structure. 2002.

[CA00] Min Chen and James Arvo. Closed-Form Expressions for Irradiance from Non-Uniform Lambertian Luminaires. 2000.

[Coo86] Robert L. Cook. Stochastic Sampling in Computer Graphics. *ACM Trans. Graph.*, 5(1):51–72, January 1986.

[Dev86] L. Devroye. *Non-Uniform Random Variate Generation.* Springer New York, 1986.

[DHMS11] David L DiLaura, Kevin W Houser, Richard G Mistrick, and Gary R Steffy. *The Lighting Handbook: Reference and Application.* Illuminating Engineering Society of North America New York (NY), 2011.

[dl90] CIE Commision Internationale de l'Éclairage. *CIE 1988 2° Spectral Luminous Efficiency Function for Photopic Vision.* Bureau Central de la CIE Vienna, 1990.

[Dro14] Michal Drobot. Physically Based Area Lights. *GPU Pro*, 5:67–100, 2014.

[DW85] Mark A. Z. Dippé and Erling Henry Wold. Antialiasing Through Stochastic Sampling. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 69–78, New York, NY, USA, 1985. ACM.

[FNS15] Willi Freeden, M. Zuhair Nashed, and Thomas Sonar, editors. *Handbook of Geomathematics.* Springer-Verlag, Berlin Heidelberg, 2 edition, 2015.

[GKDS12] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. Light Transport Simulation with Vertex Connection and Merging. *ACM Trans. Graph.*, 31(6):192:1–192:10, November 2012.

[Gmb18] DIAL GmbH. DIALux. https://www.dial.de/de/dialux/, April 2018.

[GS10] Iliyan Georgiev and Philipp Slusallek. Simple and Robust Iterative Importance Sampling of Virtual Point Lights. In *Eurographics (Short Papers)*, pages 57–60, 2010.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the Interaction of Light Between Diffuse Surfaces. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 213–222, New York, NY, USA, 1984. ACM.

[Hal64] J. H. Halton. Algorithm 247: Radical-inverse Quasi-random Point Sequence. *Commun. ACM*, 7(12):701–702, December 1964.

[HDHN16] Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. Real-time Polygonal-light Shading with Linearly Transformed Cosines. *ACM Trans. Graph.*, 35(4):41:1–41:8, July 2016.

[HJ09] Toshiya Hachisuka and Henrik Wann Jensen. Stochastic Progressive Photon Mapping. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 141:1–141:8, New York, NY, USA, 2009. ACM.

[HKSS98] Wolfgang Heidrich, Jan Kautz, Philipp Slusallek, and Hans-Peter Seidel. Canned Lightsources. In George Drettakis and Nelson Max, editors, *Rendering Techniques '98*, pages 293–300, Vienna, 1998. Springer Vienna.

[HKWB09] Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. Virtual Spherical Lights for Many-light Rendering of Glossy Scenes. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 143:1–143:6, New York, NY, USA, 2009. ACM.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive Photon Mapping. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 130:1–130:8, New York, NY, USA, 2008. ACM.

[Hop98] Hugues Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1):27–36, 1998.

[HPB07] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Matrix Row-column Sampling for the Many-light Problem. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[iec18a] IEC 60050 - International Electrotechnical Vocabulary - Details for IEV number 845-01-25: "luminous flux". http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=845-01-25, May 2018.

[iec18b] IEC 60050 - International Electrotechnical Vocabulary - Details for IEV number 845-01-56: "luminous efficacy of radiation". http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=845-01-56, May 2018.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques' 96*, pages 21–30. Springer, 1996.

[Kaj86] James T. Kajiya. The Rendering Equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM.

[Kel97] Alexander Keller. Instant Radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[KG13] Brian Karis and Epic Games. Real shading in Unreal Engine 4. *Proc. Physically Based Shading Theory Practice*, 2013.

[KK06] Thomas Kollig and Alexander Keller. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 245–257. Springer, 2006.

[KW08] M.H. Kalos and P.A. Whitlock. *Monte Carlo Methods.* Wiley, 2008.

[Lag] Sébastien Lagarde. IES light format: Specification and reader. https://seblagarde.wordpress.com/2014/11/05/ies-light-format-specification-and-reader/.

[LDSM17] P. Lecocq, A. Dufay, G. Sourimant, and J. E. Marvie. Analytic Approximations for Real-Time Area #x00A0;Light Shading. *IEEE Transactions on Visualization and Computer Graphics*, 23(5):1428–1441, May 2017.

[LR14] S. Lagarde and C.D. Rousiers. Moving Frostbite to Physically Based Rendering. *part of ACM SIGGRAPH2014 Course: Physically Based Shading in Theory and Practice*, 2014.

[LTH$^+$13] Christian Luksch, Robert F. Tobler, Ralf Habel, Michael Schwärzler, and Michael Wimmer. Fast Light-map Computation with Virtual Polygon Lights. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '13, pages 87–94, New York, NY, USA, 2013. ACM.

[LW93] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal, December 1993.

[MMP08] Albert Mas, Ignacio Martín, and Gustavo Patow. Compression and Importance Sampling of Near-Field Light Sources. In *Computer Graphics Forum*, volume 27, pages 2013–2027. Wiley Online Library, 2008.

[Nie92] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1992.

[Ren97] Robert J. Renka. Algorithm 772: STRIPACK: Delaunay Triangulation and Voronoi Diagram on the Surface of a Sphere. *ACM Trans. Math. Softw.*, 23(3):416–434, September 1997.

[RLW$^+$11] G. Rong, Y. Liu, W. Wang, X. Yin, D. Gu, and X. Guo. GPU-Assisted Computation of Centroidal Voronoi Tessellation. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):345–356, March 2011.

[RSSF02] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic Tone Reproduction for Digital Images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 267–276, New York, NY, USA, 2002. ACM.

[RT06] Guodong Rong and Tiow-Seng Tan. Jump Flooding in GPU with Applications to Voronoi Diagram and Distance Transform. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, I3D '06, pages 109–116, New York, NY, USA, 2006. ACM.

[RT07] G. Rong and T. Tan. Variants of Jump Flooding Algorithm for Computing Discrete Voronoi Diagrams. In *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, pages 176–181, July 2007.

[SIP07] Benjamin Segovia, Jean Claude Iehl, and Bernard Péroche. Metropolis instant radiosity. In *Computer Graphics Forum*, volume 26, pages 425–434, 2007.

[SKW09] Jens Schneider, Martin Kraus, and Rüdiger Westermann. GPU-based real-time discrete Euclidean distance transforms with precise error bounds. In *VISAPP (1)*, pages 435–442, 2009.

[SWZ96] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte Carlo Techniques for Direct Lighting Calculations. *ACM Trans. Graph.*, 15(1):1–36, January 1996.

[TM06] R Tobler and St Maierhofer. A Mesh Data Structure for Rendering and Subdivision. *Proceedings of WSCG (International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision)*, pages 157–162, 2006.

[UnFK13] Carlos Ureña, Marcos Fajardo, and Alan King. An Area-Preserving Parametrization for Spherical Rectangles. In *Computer Graphics Forum*, volume 32, pages 59–66. Wiley Online Library, 2013.

[VDWG15] Edgar Velázquez-Armendáriz, Zhao Dong, Bruce Walter, and Donald P. Greenberg. Complex Luminaires: Illumination and Appearance Rendering. *ACM Trans. Graph.*, 34(3):26:1–26:15, May 2015.

[Vea97] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation.* Number 1610. Stanford University PhD thesis, 1997.

[VG84] C. P. Verbeck and D. P. Greenberg. A Comprehensive Light-Source Description for Computer Graphics. *IEEE Computer Graphics and Applications*, 4(7):66–75, July 1984.

[VG95] Eric Veach and Leonidas J. Guibas. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 419–428, New York, NY, USA, 1995. ACM.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis Light Transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 65–76, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[Wan94] Changyaw Wang. *The direct lighting computation in global illumination methods.* PhD thesis, Indiana University, 1994.

[web18a] 1988 CIE Photopic Luminous Efficiency Function. http://donklipstein.com/photopic.html, May 2018.

[web18b] HILITE - VRVis. https://www.vrvis.at/research/projects/hilite/, April 2018.

[web18c] How to use the Relux Raytracing calculation. https://support.relux.com/en/support/solutions/articles/17000040665-how-to-use-the-relux-raytracing-calculation, April 2018.

[web18d] ReluxNet. https://reluxnet.relux.com/de/, April 2018.

[WFA$^+$05] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: A Scalable Approach to Illumination. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1098–1107, New York, NY, USA, 2005. ACM.

[Wit] Daniel Witzel. DIALux evo – new calculation method.

[WLWF08] Lifeng Wang, Zhouchen Lin, Wenle Wang, and Kai Fu. One-Shot Approximate Local Shading. Technical report, Tech. rep, 2008.

[YRGW11] Z. Yuan, G. Rong, X. Guo, and W. Wang. Generalized Voronoi Diagram Computation on GPU. In *2011 Eighth International Symposium on Voronoi Diagrams in Science and Engineering*, pages 75–82, June 2011.

[Zal12] Sławomir Zalewski. Digital recording of photometric data for multisource luminaires. *Przegląd Elektrotechniczny*, 88(4a):220–222, 2012.

[Zum18] Zumtobel. Products - Zumtobel. https://www.zumtobel.com/com-en/products.html, May 2018.