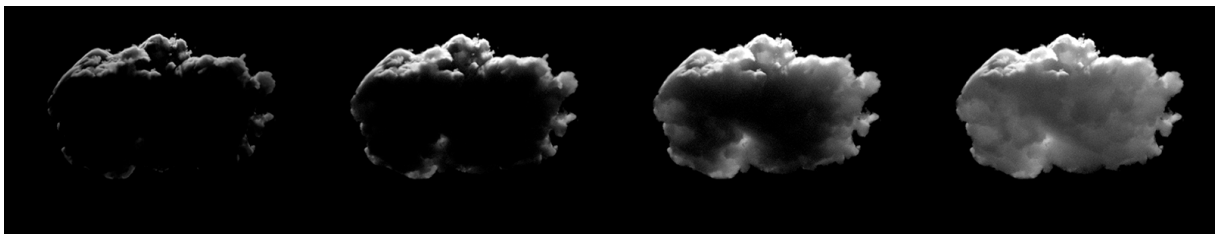
# Learning High-Order Scattering in Rendering from Data



Simon Kallweit

Master Thesis
September 2016

Prof. Dr. M. Gross

Advisors:
Jan Novák
Brian McWilliams

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Disney Research

cgl
computer graphics laboratory

# Abstract

Rendering of participating media is a well researched topic. Using Monte Carlo path tracing, participating media can be rendered in a physically accurate way. However, depending on the medium properties involved, the performance of this simple algorithm can degrade quickly. Rendering clouds in particular is a hard problem, as light can scatter many times inside the medium due to lack of absorption, leading to noisy images and slow convergence. We investigated methods for using machine learning techniques to predict the contribution of high-order scattering from ground truth data. We developed two models using neural networks, to predict light transport in participating media. We have implemented software to generate the datasets for learning and a renderer incorporating our models. We analyzed the performance and quality of our models and discuss further improvements and avenues for future research.

# Zusammenfassung

Die Bildsynthese von voluminösen Materialien ist ein gut erforschtes Gebiet. Dank Monte Carlo Pfadnachverfolgung können solche Materialien physikalisch korrekt berechnet werden. Die Effizienz dieses einfachen Berechnungsverfahren nimmt aber je nach verwendeten Eigenschaften des voluminösen Materials schnell ab. Die Bildsynthese von Wolken zum Beispiel, ist ein schwieriges Problem. Weil Licht in einer Wolke nur selten absorbiert und dadurch viele Male gestreut wird, führt einfache Pfadnachverfolgung zu viel Rauschen und langsamer Konvergenz. Wir haben verschiedene Methoden untersucht, welche maschinelles Lernen verwenden um den Lichtanteil von höhergradiger Streuung auf Basis von Referenzdaten vorauszusagen. Wir entwickelten zwei Modelle, unter Verwendung von neuronalen Netzwerken, um den Lichttransport in voluminösen Materialen vorauszusagen. Desweiteren entwickelten wir Programme zur Erstellung grosser Mengen an Referenzdaten, um unsere Modelle anzulernen, sowie zur Bildsythese unter Verwendung unserer Modelle. Wir analysierten die Leistung und Genauigkeit unserer Modelle und diskutieren Verbesserungen und Möglichkeiten für zukünftige Forschung.


Master's Thesis:

# Learning High-Order Scattering in Rendering from Data

## Introduction

The appearance of natural phenomena, such as clouds or smoke, is strongly affected by light interacting with volumetric media, also called participating media. Rendering such phenomena with traditional rendering algorithms is challenging due to the highly scattering nature of the media involved. For example, rendering physically realistic clouds by solving light transport using standard path tracing is computationally much too expensive to be used in practical applications such as movie production. Plenty of research has been devoted to efficiently solving the radiative transport equation (RTE), which can be used to generate plausible images of these hard to render phenomena. Most methods aim at finding approximations for the high-order scattering terms, as the low-order terms can be accurately computed using traditional methods such as path tracing. It is also common practice to completely discard the high-order scattering terms and tweak the medium properties to achieve plausible and aesthetically pleasing results.

## Task Description

In this thesis, we want to explore possibilities of using machine learning techniques to accelerate the rendering of highly scattering participating media. While there is no prior work in this specific area, recent work has successfully applied machine learning in related areas such as fluid simulation. We ultimately hope to find methods that are able to generate images exhibiting the typical visual features originating from high-order scattering, while reducing the required computational effort. In order to achieve this, we plan to work on the following problems:

- Find light transport models that are suitable for prediction
- Train these models using training data obtained through standard light transport algorithms
- Implement rendering methods using the trained models to predict light transport
- Compare results against existing rendering algorithms, investigate temporal stability

## Remarks

A written report and an oral presentation conclude the thesis. The thesis will be overseen by Prof. Markus Gross and supervised by Jan Novak and Brian McWilliams.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

## Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

---

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

**Title of work** (in block letters):

LEARNING HIGH-ORDER SCATTERING IN RENDERING FROM DATA

**Authored by** (in block letters):

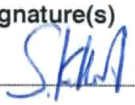
*For papers written by groups the names of all authors are required.*

| **Name(s):** | **First name(s):** |
|---|---|
| KALLWEIT | SIMON |
| | |
| | |
| | |
| | |

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

| **Place, date** | **Signature(s)** |
|---|---|
| Herzogenbuchsee, 20.09.2016 |  |
| | |
| | |
| | |
| | |

*For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.*

# Acknowledgments

I would like to thank both of my advisors, Jan Novák and Brian McWilliams, for guidance and many discussion over the course of this thesis. I would also like to thank Thijs Vogels, my fellow combatant, working on his own master thesis, for discussions and for sharing some of his machine learning tools.

I would also like to particularly thank my love Tina, for her constant support and for the greatest gift of my live, our 10 month old daughter. Thank you Leya, for reminding me of the important things in life and for cheering me up every single day, since you first saw the light of day.

Finally, I would like to thank my family and my friends, who supported my over all those years.

# Contents

# List of Figures

# List of Tables

# 1

# Introduction

In the 1970s, University of Utah hired Ivan Sutherland to teach a computer graphics class, which attracted many students that later became major contributors to the field. Among them, Edwin Catmull, who returned back to the university for graduate school. Although he always dreamed of becoming an animator, as he loved animation movies from his early childhood, he found great interest in the field of computer graphics. When he created his first 3D animated short movie now known as "A Computer Animated Hand", it became his vision to produce a feature-length animation movie exclusively created with computer graphics. He fulfilled his dream by co-founding Pixar Animation Studios in 1986 and releasing "Toy Story" in 1995, the first feature-length computer animated movie ever.

It took an average of 4 hours to render a single frame using a technique known as *rasterization*, which basically projects 3D objects onto the screen plane and computes lighting and shading effects pixel by pixel. In a recent experiment in 2011, Pixar re-rendered the movie on their current rendering cluster with an average render time of around 2 minutes per frame. Applying Moore's law, doubling the compute power every 18 months, one would expect a speedup of roughly a factor 1000. In practice, the speedup was only around a factor 100, but it still demonstrates the incredible pace at which technology evolved over the past decade. Naturally, the additional computation power is used to increase the visual quality of the rendered frames, rather than reducing the render time, or as Jim Blinn stated: "As technology advances, the rendering time remains constant".

A major increase in visual quality was achieved by incorporating *global illumination*, the effect of light scattering multiple times among surfaces in the scene. "Shrek 2", produced by DreamWorks Animation in 2004, was the first animation movie using an approximate solution for global illumination, followed by "Up", released by Pixar in 2009, and "Toy Story 3" in 2010. In recent years, rendering technology for movie production has shifted towards *path tracing*, an algorithm that renders images by tracing light paths from the camera, scattering at surfaces in the scene and accumulating light when paths intersect with light sources. Path tracing is a

very intuitive way of handling light transport, as it resembles how light moves in the physical world, allowing for better modeling of physically based materials and light sources. In addition to surfaces, path tracing can also be used for rendering skin, hair, fur and volumetric effects such as smoke and clouds.

Rendering volumetric materials, also called *participating media*, is a challenging problem, as light does not only scatter at the surface boundary of an object, but actually penetrates into the object and scatters potentially many times before exiting. Similar to the early techniques for rendering surfaces, where only a single scattering event was computed between the camera and the light source, participating media have for a long time been rendered with *single-scattering*. This is a crude approximation for many volumetric effects including clouds, where light can scatter hundreds of times before leaving. To render such effects more realistically, rendering algorithms need allow for light to scatter multiple times, also called *multiple-scattering*. With todays computational power it is usually feasible to compute multiple-scattering up to a low number of bounces, leaving out the *high-order scattering* contribution, but this can result in a huge loss of energy if the medium is not absorbing light, which for example is the case in clouds. "The Good Dinosaur", recently released by Disney and Pixar, features realistic renderings of clouds computed with path tracing. It took an average of 48 hours to render a single frame, while multiple-scattering effects for clouds have been accelerated by using some approximations [Wre15]. This clearly illustrates that computational power is still not abundant enough for rendering these kinds of volumetric effects in a physically correct way.

The appearance of clouds is typically rather diffuse and blurry, which is a consequence of high-order scattering. This allows for many approximations to be made when rendering clouds, however, these techniques usually result in a large bias compared to the ground truth. Artists can counter these errors to some extent by tweaking medium properties and get back more realistic results. Instead of using analytical approximations, our main idea is to learn how to predict the high-order scattering contribution based on ground truth data.

Recent development in machine learning, most prominently *deep learning* techniques, have inspired many applications where complex tasks are successfully solved by learning from large amounts of data, encouraging our endeavor. In this thesis, we have investigated different possibilities of learning high-order scattering from data. The result of our investigations is the development of two prediction models based on neural networks. We have implemented software to create large sets of ground truth data on a computer cluster to train our models and implemented a custom renderer incorporating the models for predicting high-order scattering. To test the quality and performance of our models, we have conducted various experiments and comparisons and provide an analysis and discussion of the results.

We introduce the theoretical background that builds the foundation for this thesis in Chapter 2. This includes the fundamentals of light transport in participating media, Monte Carlo integration, basic rendering algorithms and an introduction to machine learning with focus on nonlinear regression using neural networks.

In Chapter 3 we review related work in the fields of rendering participating media as well as in recent applications of machine learning in the field of computer graphics in general.

We investigate the effects of high-order scattering when rendering clouds in Chapter 4 and discuss different ideas for predicting its contribution.

Chapter 5 introduces our first model, that uses neural networks to predict the global light transport from an infinite directional light source to a single point and direction within a simple convex bounded homogeneous medium. We discuss the model, details on the training of the model as well as integrating the model into a renderer and provide analysis on the performance and quality of the model.

We extend our first model into a more general point-to-point transport model in Chapter 6, which predicts the transport between two points within a heterogeneous participating medium. We again discuss the model, training and usage of the model inside a renderer and discuss the inherent difficulties we faced during the development.

A conclusion of this thesis and a quick review of the contributions is provided in Chapter 7. We also discuss various ideas for future work based on insights gathered during our experiments.

# 2

# Theoretical Background

In this chapter we review the theoretical background we use throughout the rest of this thesis. First, we introduce the theory of *light transport*, with focus on participating media, which builds the foundation for rendering. Then we introduce *Monte Carlo integration*, a numerical framework for computing integrals of arbitrary dimensionality, which is one of the most commonly used techniques to solve light transport integrals in rendering. Next, we review two of the most commonly used *rendering algorithms*, namely *path tracing* and *bidirectional path tracing*. We use these algorithms both for generating datasets for learning as well as for rendering images. Finally, we review the *machine learning* techniques used for our models, with focus on non-linear regression using neural networks.

## 2.1. Light Transport

Light transport is the theory that describes how light propagates through a 3D environment. In this section, we review the terminology and mathematical foundation for light transport within participating media. Refer to [Jar08] for a more thorough discussion.

### 2.1.1. Radiometry

*Radiometry* is the study of measuring electromagnetic radiation, which also includes visible light. We will quickly review the radiometric quantities and units that are most relevant for reasoning about light transport.

To quantify light, we can think of individual photons that carry quantums of energy. The total amount of energy of a number of photons, denoted $Q$, is expressed in terms of joules $[J]$.

*Flux*, or power, measures the total amount of energy passing through a surface per unit time. Denoted $\Phi$, flux is expressed in terms of watts $[W = J \cdot s^{-1}]$ and defined as:

$$\Phi = \frac{dQ}{dt}. \tag{2.1}$$

As an example, we can quantify the total amount of light emitted from a light source by measuring the flux across a hypothetical sphere enclosing the light source.

*Solid angle*, denoted by $\Omega$, is the area on the unit sphere an object covers viewed from the center vertex of the sphere and is expressed in *steradians* $[sr]$. Differential solid angle can be expressed both in terms of an area element $d\mathcal{A}$ at distance $r$ or in polar coordinates:

$$d\vec{\omega} = \frac{d\mathcal{A}}{r^2} = \sin\theta \, d\theta \, d\phi. \tag{2.2}$$

We use use $\vec{\omega}$ to specify directions and assume they are unit length ($\|\vec{\omega}\| = 1$). Similarly, we write $\vec{\omega}_{\mathbf{xy}}$ to specify the normalized direction from $\mathbf{x}$ to $\mathbf{y}$.

*Radiant intensity* measures the directional density of flux. It is denoted by $I$ (units $[W \cdot sr^{-1}]$), and can be expressed in terms of flux:

$$I(\vec{\omega}) = \frac{d\Phi(\vec{\omega})}{d\vec{\omega}}. \tag{2.3}$$

As an example, an isotropic point light source with total power $\Phi$ has uniform intensity $I = \frac{\Phi}{4\pi}$.

*Radiance* measures the flux density per unit solid angle, per unit projected area. We denote *radiance* by $L$ (units $[W \cdot sr^{-1} \cdot m^{-2}]$) and express in it terms of flux:

$$L(\mathbf{x}, \vec{\omega}) = \frac{d^2\Phi(\mathbf{x}, \vec{\omega})}{d\vec{\omega} \, d\mathcal{A}^{\perp}(\mathbf{x})}, \tag{2.4}$$

where $d\mathcal{A}^{\perp}(\mathbf{x}) = d\mathcal{A}(\mathbf{x}) \cos\theta$ is projected area, with $\theta$ being the angle between the surface normal and direction $\vec{\omega}$.

In order to distinguish between incoming and outgoing radiance, we use the following convention:

- $L(\mathbf{x} \leftarrow \vec{\omega})$ is used to specify incoming radiance to $\mathbf{x}$ from direction $\vec{\omega}$,
- $L(\mathbf{x} \rightarrow \vec{\omega})$ is used to specify outgoing radiance from $\mathbf{x}$ into direction $\vec{\omega}$.

Radiance is arguably the most important quantity when computing light transport, as it remains constant along straight lines when inside a vacuum, meaning that radiance leaving at point $\mathbf{x}$ into direction of point $\mathbf{y}$ is equal to radiance arriving at point $\mathbf{y}$ from direction $\mathbf{x}$.

*Fluence* measures the flux at a specific point. We denote *fluence* by $\Phi(\mathbf{x})$ (units $[W \cdot m^{-2}]$) and define it as an integral of radiance:

$$\Phi(\mathbf{x}) = \int_{\Omega_{4\pi}} L(\mathbf{x} \leftarrow \vec{\omega})\, d\vec{\omega}, \tag{2.5}$$

where $\Omega_{4\pi}$ refers to integration over the sphere of directions.

### 2.1.2. Participating Media

While many computer graphics applications assume light to travel within the vacuum, we focus on light traveling inside *participating media*. In rendering, it is common practice to model participating media as collections of microscopic particles and to reason about the interactions between light and particles using a probabilistic view. This assumes that interactions with the medium are statistically independent of each other. In physics, it is known that there actually is correlation between the scattering events in clouds, however, this theory has generally not yet been adapted to computer graphics and is beyond the scope of this thesis.

### 2.1.3. Radiative Transfer Equation

As photons travel through a participating medium, they may interact with particles in the medium. The probability of an interaction between a photon and a particle is related to the extinction coefficient $\sigma_t$ (units $[m^{-1}]$), which depends both on the density and size of the particles. See Figure 2.1a for a visual comparison of the appearance of media with different extinction coefficients. When a photon interacts with a particle, it is either absorbed or scattered into a new direction, with relative probabilities given by the absorption coefficient $\sigma_a$ and scattering coefficient $\sigma_s$. The extinction coefficient is defined as $\sigma_t = \sigma_a + \sigma_s$ and its inverse $\frac{1}{\sigma_t}$ (units $[m]$) is known as the *mean free path* distance. The ratio $\alpha_s = \frac{\sigma_s}{\sigma_t}$ between the scattering and extinction coefficient is known as the *scattering albedo*. See Figure 2.1b for a visual comparison of the appearance of different scattering albedos.

The *radiative transport equation* describes the differential change of radiance $L$ along a given direction $\vec{\omega}$ and is defined as:

$$\begin{aligned}(\vec{\omega} \cdot \nabla) L(\mathbf{x} \to \vec{\omega}) = &\underbrace{-\sigma_a(\mathbf{x}) L(\mathbf{x} \to \vec{\omega})}_{\text{absorption}} \\ &\underbrace{+\sigma_a(\mathbf{x}) L_e(\mathbf{x} \to \vec{\omega})}_{\text{emission}} \\ &\underbrace{-\sigma_s(\mathbf{x}) L(\mathbf{x} \to \vec{\omega})}_{\text{out-scattering}} \\ &\underbrace{+\sigma_s(\mathbf{x}) L_i(\mathbf{x} \to \vec{\omega})}_{\text{in-scattering}}.\end{aligned} \tag{2.6}$$

The in-scattered radiance $L_i$ is defined as an integral over solid angle:

$$L_i(\mathbf{x} \rightarrow \vec{\omega}) = \int_{\Omega_{4\pi}} p(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega}) L(\mathbf{x} \leftarrow \vec{\omega}')\, d\vec{\omega}', \tag{2.7}$$

where $p(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega})$ is the *phase function* and $\Omega_{4\pi}$ refers to integration over the sphere of directions.

We can simplify the radiative transfer equation by combining the absorption and out-scattering terms into a single extinction term and removing the emission term, assuming that participating media are generally not emitting light:

$$(\vec{\omega} \cdot \nabla) L(\mathbf{x} \rightarrow \vec{\omega}) = \underbrace{-\sigma_t(\mathbf{x}) L(\mathbf{x} \rightarrow \vec{\omega})}_{\text{extinction}} \underbrace{+\sigma_s(\mathbf{x}) L_i(\mathbf{x} \rightarrow \vec{\omega})}_{\text{in-scattering}}. \tag{2.8}$$

### 2.1.4. Phase Function

The *phase function* $p(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega})$ describes the angular distribution of scattering and has units $[sr^{-1}]$. We use the convention of $\vec{\omega}$ and $\vec{\omega}'$ both pointing *away* from the scattering location $\mathbf{x}$. The phase function for uniform or *isotropic* scattering is defined as:

$$p_{iso}(\vec{\omega}' \leftrightarrow \vec{\omega}) = \frac{1}{4\pi}. \tag{2.9}$$

A commonly used phase function in computer graphics is the *Henyey Greenstein* phase function [HG41], which can model both forward and backward scattering:

$$p_{hg}(\vec{\omega}' \leftrightarrow \vec{\omega}) = \frac{1 - g^2}{4\pi \left(1 + g^2 + 2g \cos\theta\right)^{\frac{3}{2}}}, \tag{2.10}$$

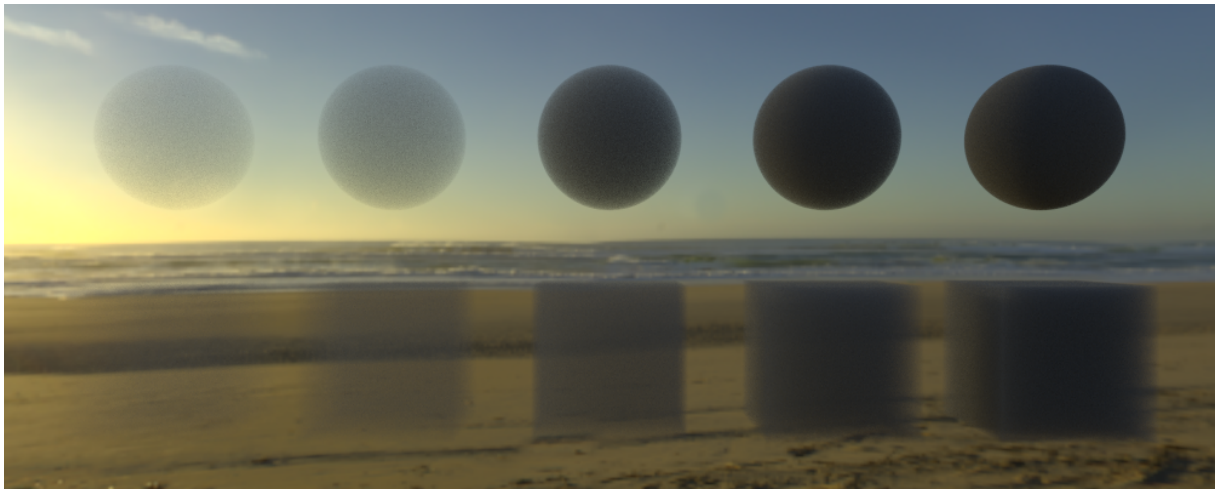
where $\cos\theta = \langle \vec{\omega}', \vec{\omega} \rangle$ and the parameter $g \in (-1, 1)$, known as *eccentricity*, determines the strength of forward or backward scattering. See Figure 2.1c for a visual comparison of the appearance with different values of $g$.

When the size of particles in a participating media are comparable to the wavelength of visible light, such as water particles in fog and clouds, the phase function can be numerically derived using Lorenz-Mie theory [Lor90, Mie08]. In rendering, empirically derived approximations for different types of media based on the Lorenz-Mie theory have been used. Among them, Rayleigh scattering [Ray71], to model clear atmospheres:

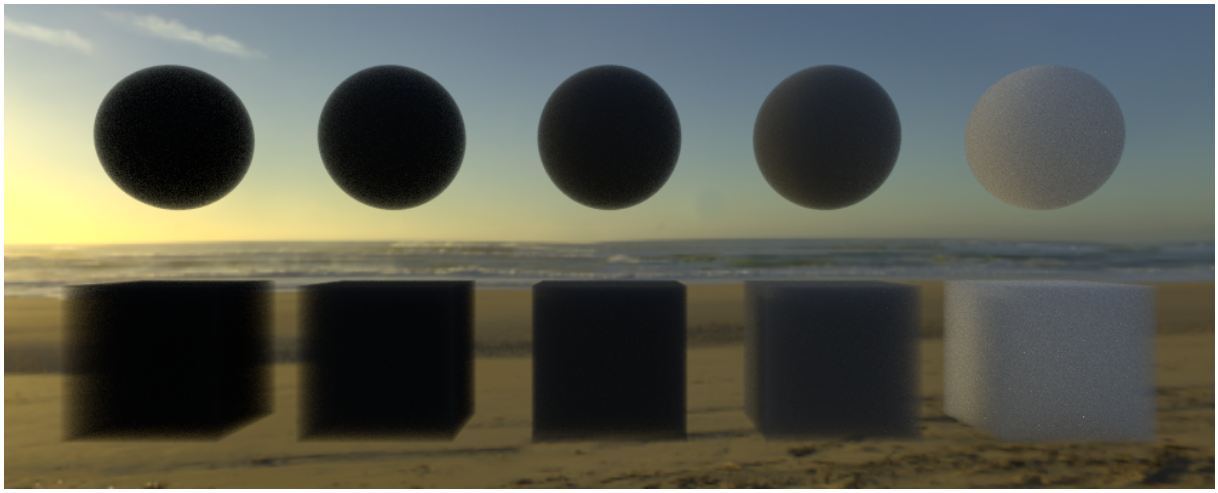
$$p_r(\vec{\omega}' \leftrightarrow \vec{\omega}) = \frac{3}{16\pi}(1 + \cos^2\theta), \tag{2.11}$$

as well as approximations to model *hazy* and *murky* atmospheres [NMN87]:

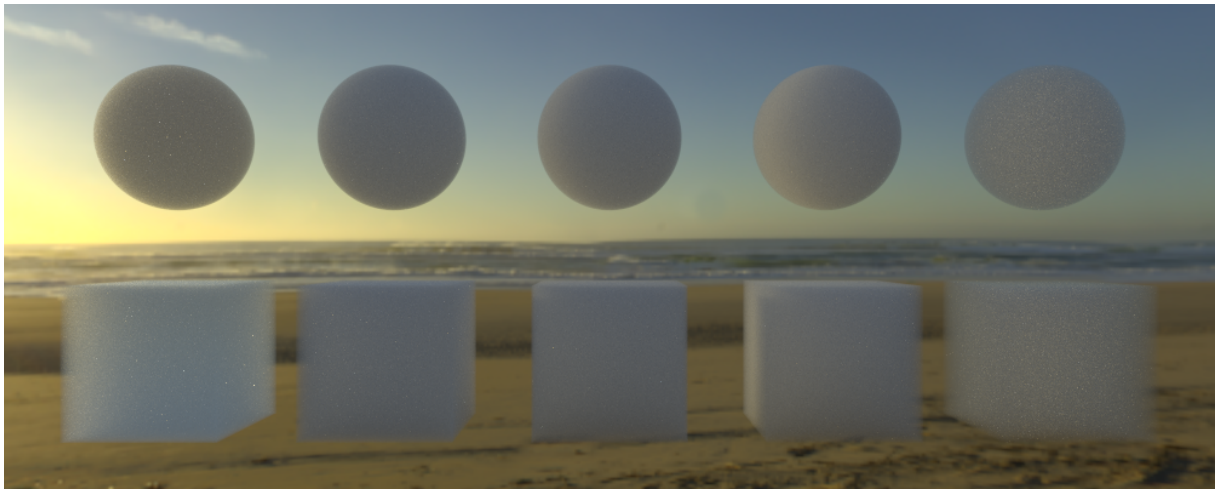
$$p_{mh}(\vec{\omega}' \leftrightarrow \vec{\omega}) = \frac{1}{4\pi}\left(\frac{1}{2} + \frac{9}{2}\left(\frac{1 + \cos\theta}{2}\right)^8\right), \tag{2.12}$$



**(a)** Extinction coefficient $\sigma_t$ is increased from left to right.

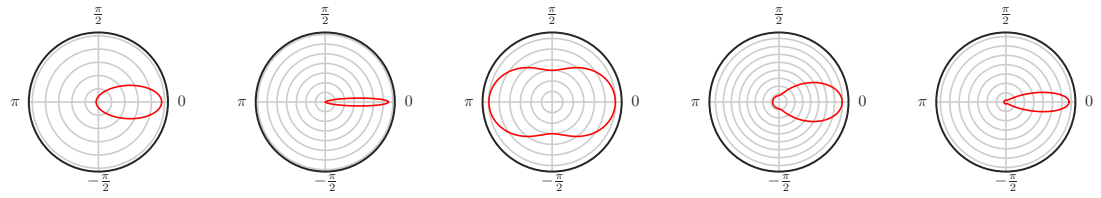


**(b)** Scattering albedo $\alpha_s$ is increased from left to right.



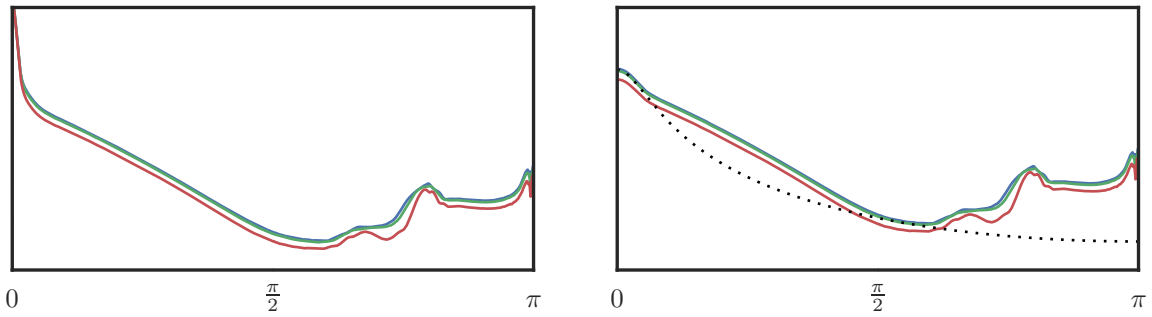
**(c)** Henyey Greenstein eccentricity value $g$ is $-0.9, -0.5, 0.0, +0.5, +0.9$ from left to right.

***Figure 2.1.:*** *Renderings showing the visual appearance of the primary properties of participating media.*



**(a)** **(b)** **(c)** **(d)** **(e)**

***Figure 2.2.:*** *Angular plots of different phase functions: (a) and (b) show the Henyey Greenstein phase function with $g = 0.5$ and $g = 0.85$, (c) shows the Rayleigh phase function, (d) shows the hazy and (e) shows the murky phase function.*



**(a)** **(b)**

***Figure 2.3.:*** *Plots of the Lorenz-Mie phase function for a typical water cloud. In (a), we show the phase function computed for the red, green and blue color channels. In (b), we show the chopped model by removing the peak and show the Henyey Greenstein phase function with $g = 0.85$ used as an approximation.*

$$p_{mm}(\vec{\omega}' \leftrightarrow \vec{\omega}) = \frac{1}{4\pi}\left(\frac{1}{2} + \frac{33}{2}\left(\frac{1+\cos\theta}{2}\right)^{32}\right). \tag{2.13}$$

See Figure 2.2 for angular plots of the different phase functions. To render realistic clouds, we would typically want to use the Lorenz-Mie theory to tabulate a phase function that corresponds to the size and density of water droplets found in actual clouds. The resulting phase function generally has a very strong forward peak, as shown in Figure 2.3a. Using the *chopped model* [Bou08], the strong forward peak is replaced with a delta distribution, which is equal to adjusting the extinction coefficient $\sigma_t$. The resulting chopped phase function can be approximated with a Henyey Greenstein phase function with eccentricity $g = 0.85$, as shown in Figure 2.3b.

### 2.1.5. Volume Rendering Equation

The *rendering equation* [Kaj86] describes in integral form how to compute radiance arriving at a camera in a scene built from surfaces and light sources. The *volume rendering equation*, extends

the rendering equation by handling participating media. We describe a simplified version based on the assumption that the scene does not contain surfaces.

The loss of radiance due to extinction is given by the first term of the radiative transfer equation (2.8). Integrating this term along a ray from $\mathbf{x}$ to $\mathbf{x}'$, we get the *transmittance*, the fraction of radiance remaining after traveling along the ray:

$$T_r(\mathbf{x} \leftrightarrow \mathbf{x}') = e^{-\tau(\mathbf{x} \leftrightarrow \mathbf{x}')}, \tag{2.14}$$

where $\tau(\mathbf{x} \leftrightarrow \mathbf{x}')$, the *optical depth*, is defined as:

$$\tau(\mathbf{x} \leftrightarrow \mathbf{x}') = \int_{\mathbf{x}}^{\mathbf{x}'} \sigma_t(\mathbf{x})\, ds(\mathbf{x}). \tag{2.15}$$

The volume rendering equation is the integral form of the radiative transfer equation (2.8) and is given by:

$$\begin{aligned} L(\mathbf{x} \leftarrow \vec{\omega}) = & \underbrace{T_r(\mathbf{x} \leftrightarrow \mathbf{x}_e) L_e(\mathbf{x}_e \rightarrow -\vec{\omega})}_{\text{reduced radiance}} \\ & + \underbrace{\int_{\mathbf{x}}^{\mathbf{x}_e} T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t \rightarrow -\vec{\omega})\, ds(\mathbf{x}_t)}_{\text{accumulated inscattered radiance}}, \end{aligned} \tag{2.16}$$

where $L_e$ is the radiance emitted by the environment, $r(\mathbf{x}, \vec{\omega})$ is the *ray function* that returns the nearest point starting at $\mathbf{x}$ along $\vec{\omega}$ past all participating media and $\mathbf{x}_e = r(\mathbf{x}, \vec{\omega})$.

## 2.2. Monte Carlo Integration

*Monte Carlo integration* is a numerical framework for approximating integrals by means of random sampling. Despite being a relatively old technique, having its origin during the development of the atomic bomb in 1949 [MU49], it is still the most commonly used method for computing light transport today. This section provides an overview of the technique, including more recent additions relevant for light transport computation such as *multiple importance sampling*. Refer to Pharr et al. [PH10] and Veach [Vea97] for a more thorough discussion.

### 2.2.1. Random Variables

A *random variable*, denoted by $X$, is a variable whose value is the outcome of some random process. Random variables can either be discrete or continuous, however, we focus on the continuous kind. The *cumulative distribution function*, or CDF, of a random variable $X$, is the probability that a randomly drawn value from the variable's distribution is less or equal to some threshold $x$:

$$cdf(x) = Pr[X \leq x]. \tag{2.17}$$

The *probability density function*, or PDF, is the derivative of the CDF:

$$pdf(x) = \frac{d}{dx} cdf(x). \tag{2.18}$$

Because CDFs are always monotonically increasing we can use:

$$Pr[a \leq x \leq b] = \int_a^b pdf(x)\, dx = cdf(b) - cdf(a). \tag{2.19}$$

The *expected value* and *variance* of a random variable $Y = f(X)$ over domain $\Omega(x)$ are defined as:

$$E[Y] = \int_{\Omega(x)} f(x) pdf(x)\, d\Omega(x), \tag{2.20}$$

$$\sigma^2[Y] = E[(Y - E[Y])^2], \tag{2.21}$$

where $\sigma$, the square root of *variance*, is the *standard deviation*.

### 2.2.2. Monte Carlo Estimator

The *Monte Carlo estimator* addresses the problem of numerically solving the integral:

$$F = \int_\Omega f(x)\, d\Omega(x), \tag{2.22}$$

for an arbitrary function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ and integration domain $\Omega \subset \mathbb{R}^n$. $F$ can be estimated by taking $N$ random samples from a distribution over $\Omega$ of a random variable $X$:

$$\langle F^N \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{pdf(X_i)}. \tag{2.23}$$

It is easy to show that the expected value of $\langle F^N \rangle$ is indeed equal to F:

$$\begin{aligned} E\left[\langle F^N \rangle\right] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{pdf(X_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N} E\left[\frac{f(X_i)}{pdf(X_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_{\Omega}\frac{f(x)}{pdf(x)}pdf(x)\,dx \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_{\Omega} f(x)\,dx \\ &= \int_{\Omega} f(x)\,dx \\ &= F. \end{aligned}$$

The convergence rate of the estimator $\langle F^N \rangle$ is given by:

$$\sigma[\langle F^N \rangle] = \frac{1}{\sqrt{N}}\sigma[Y] \propto \frac{1}{\sqrt{N}}, \tag{2.25}$$

where $Y_i = f(X_i)/pdf(X_i)$. To half the error, we generally need four times the number of samples. However, by reducing the variance of $Y$, the total variance of the estimator can be reduced significantly, as discussed in the next section. Many numerical quadrature rules offer better convergence rates for low dimensional integrals, but they suffer from the *curse of dimensionality*, leading to exponentially reduced convergence rates with increasing dimensionality of the integral. The convergence rate of Monte Carlo integration is independent of the dimensionality of the integral, making it a very useful technique for computing light transport.

### 2.2.3. Importance Sampling

The variance of a Monte Carlo estimator (2.23) can be reduced significantly by choosing samples from a distribution $pdf(x)$ that resembles the integrand $f(x)$. This is generally referred to as *importance sampling*. Intuitively speaking, importance sampling simply places more samples in regions of the integrand where it has high importance, e.g. results in large values.

### 2.2.4. Multiple Importance Sampling

For a complex integrand $f(x)$, multiple sampling techniques with distributions $pdf_i(x)$ may exist, each one putting importance on a different region of the integrand. Because variance is additive, averaging estimators using different sampling techniques does not result in overall reduced variance. By using multiple importance sampling [VG95] we can combine samples in

a way that provably reduces overall variance. The multiple importance sampling estimator has the form:

$$\langle F \rangle_{MIS} = \sum_{i=1}^{M} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{pdf_i(X_{i,j})}, \tag{2.26}$$

where $M$ is the number of different sampling techniques with distributions $pdf_i$, $N_i$ is the number of samples taken for sampling technique $i$, $X_{i,j}$ is the $j$-th sample of $i$-th sampling technique, and $w_i(x)$ is a weighting function. Two of the most commonly used weighting functions are the *power heuristic* and the *balance heuristic*. The power heuristic is defined as:

$$w_i(x) = \frac{[n_i \, pdf_i(x)]^\beta}{\sum_k [n_k pdf_k(x)]^\beta}, \tag{2.27}$$

where $\beta$ is typically set to 2. The balance heuristic is obtained by setting $\beta$ to 1.

### 2.2.5. Russian Roulette

Reducing variance by drawing more samples can become expensive when the cost of evaluating individual samples is high. With prior knowledge of the contribution of individual samples $f(x) \propto q$, *Russian roulette* can be used to cut down the number of evaluated samples by replacing the integrand with:

$$f(x)_{RR} = \begin{cases} c & \text{with probability } q, \\ \frac{f(x)-qc}{1-q} & \text{otherwise,} \end{cases} \tag{2.28}$$

where $c$ is a constant value, usually set to 0. Russian roulette will generally not reduce variance, but by omitting expensive evaluation on samples with low contribution, more time is spent evaluating samples with high contribution, leading to better convergence overall.

### 2.2.6. Confidence Interval Estimation

Monte Carlo estimators are an efficient technique for computing complex integrals. The number of samples $N$ is often increased iteratively until acceptable convergence is reached. Based on the *central limit theorem* and *interval estimation* [KK51], *confidence interval estimation* allows for estimating the number of samples necessary to reach convergence up to a specified error. We only provide a brief overview of the technique, see [Bou08] for the full derivation.

Let us restate the Monte Carlo estimator of an integral $F = \int_\Omega f(x) \, d\Omega(x)$:

$$\langle F^N \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{pdf(X_i)}. \tag{2.29}$$

The estimator is equal to the sample mean of Y:

$$\langle Y \rangle_N = \frac{1}{N} \sum_{i=1}^{N} Y_i, \tag{2.30}$$

with $Y_i = f(X_i)/pdf(X_i)$. The corresponding sample variance of Y is:

$$S_N^2 = \left\langle Y^2 \right\rangle_N - \langle Y \rangle_N^2 . \tag{2.31}$$

Sample mean and variance can easily be computed iteratively while running the Monte Carlo estimator, but some care needs to be taken to avoid a problem known as *catastrophic cancellation* [Wel62]. Confidence interval estimation allows to estimate the number of samples $N$ necessary for the sample mean $\langle Y \rangle_N$ to converge to a value within the *confidence interval* $[F - c, F + c]$ with a probability of $1 - \alpha$, called then *confidence level*. Because $F$ is unknown, we can instead define the confidence interval in relation to the sample mean using $c = k \langle Y \rangle_N$ and estimate the number of samples $N$ with:

$$N \geq \left( \frac{t_{a/2} S_N}{k \langle Y \rangle_N} \right)^2 , \tag{2.32}$$

where $t_{a/2}$ is given by the relation $\mathcal{N}(t_{a/2}) = 1 - a/2$, with $\mathcal{N}$ being the CDF of the normal distribution. The confidence interval can also be defined in relation to the sample variance $c = kS_N$, which results in:

$$N \geq \left( \frac{t_{a/2}}{k} \right)^2 . \tag{2.33}$$

## 2.3. Rendering Algorithms

In this section, we review two rendering algorithms, *path tracing* and *bidirectional path tracing*. Both algorithms use Monte Carlo integration for solving light transport by sampling random walks. To reason about the two algorithms, we use a simplified version of the path integral framework [JM12], only considering light transport in participating media, without taking into account surfaces and visibility. The pixel intensity $I$ on the image plane can be written as an integral over the domain $\Omega$ of light transport paths:

$$I = \int_{\Omega} f(\overline{\mathbf{x}}) \, dV(\overline{\mathbf{x}}), \tag{2.34}$$

with $\overline{\mathbf{x}} = \mathbf{x}_0, \ldots, \mathbf{x}_k$ being a path of $k \geq 1$ segments and $k + 1$ vertices, where $\mathbf{x}_k$ is on the light source, or *emitter*, $\mathbf{x}_0$ is on the camera, or *sensor*, and the remaining vertices are scattering locations within participating media. The differential measure $dV(\overline{\mathbf{x}})$ corresponds to volume

integration. The path contribution function $f$ is defined as product of geometry throughput $G(\overline{\mathbf{x}})$, scattering throughput $p(\overline{\mathbf{x}})$ and transmittance $T_r(\overline{\mathbf{x}})$:

$$f(\overline{\mathbf{x}}) = G(\overline{\mathbf{x}})p(\overline{\mathbf{x}})T_r(\overline{\mathbf{x}}), \tag{2.35}$$

where

$$G(\overline{\mathbf{x}}) = \prod_{i=0}^{k-1} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}), \qquad p(\overline{\mathbf{x}}) = \prod_{i=0}^{k} p(\mathbf{x}_i), \qquad T_r(\overline{\mathbf{x}}) = \prod_{i=0}^{k-1} T_r(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}).$$

The geometry term for segment $\mathbf{xy}$ is defined as $G(\mathbf{x} \leftrightarrow \mathbf{y}) = \frac{D(\mathbf{x}\to\mathbf{y})D(\mathbf{y}\to\mathbf{x})}{\|\mathbf{x}-\mathbf{y}\|^2}$, where $D(\mathbf{x} \to \mathbf{y}) = |\vec{\mathbf{n}}_x \cdot \vec{\omega}_{xy}|$, if $\mathbf{x}$ is on a surface with normal $\vec{\mathbf{n}}_x$, e.g. on the emitter or the sensor and $D(\mathbf{x} \to \mathbf{y}) = 1$ otherwise, and the same for $D(\mathbf{y} \to \mathbf{x})$. The transmittance term $T_r(\mathbf{x} \leftrightarrow \mathbf{y})$ is defined as in (2.14). The scattering function $p(\mathbf{x}_i)$ is defined as:

$$p(\mathbf{x}_i) = \begin{cases} L_e(\mathbf{x}_0 \to \vec{\omega}_{\mathbf{x}_0\mathbf{x}_1}) & \text{if } i = 0, \\ W_e(\mathbf{x}_k \leftarrow \vec{\omega}_{\mathbf{x}_k\mathbf{x}_{k-1}}) & \text{if } i = k, \\ p_m(\mathbf{x}_i, \vec{\omega}_{\mathbf{x}_i\mathbf{x}_{i-1}} \leftrightarrow \vec{\omega}_{\mathbf{x}_i\mathbf{x}_{i+1}})\sigma_s(\mathbf{x}_i) & \text{otherwise,} \end{cases} \tag{2.36}$$

where $L_e$ is the emission, $W_e$ is the sensor importance, $\sigma_s$ is the scattering coefficient and $p_m$ is the phase function of the medium. Figure 2.4 shows an overview of the terms involved in a single light path.

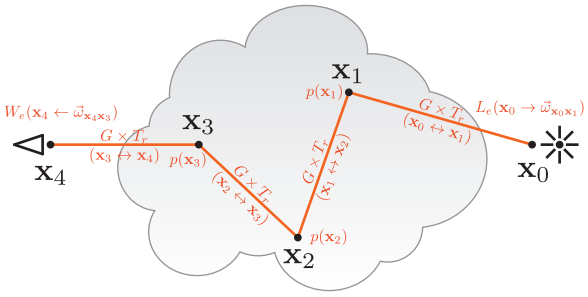



***Figure 2.4.:*** *Visualization of the terms involved in the transport along a light path $\overline{\mathbf{x}}$.*

The path probability density $pdf(\overline{\mathbf{x}})$ is defined as the joint probability density of the individual vertices, $pdf(\overline{\mathbf{x}}) = pdf(\mathbf{x}_0, \ldots, \mathbf{x}_k)$. Ideally, using Monte Carlo integration, the joint probability density $pdf(\overline{\mathbf{x}})$ should be proportional to the path contribution function $f(\overline{\mathbf{x}})$. In practice however, depending on the rendering algorithm, different terms of the contribution function cannot be importance sampled, leading to increased variance.

### 2.3.1. Path Tracing

In *path tracing*, also called *unidirectional path tracing*, light paths are incrementally constructed starting at the sensor. First, a vertex is sampled on the sensor with respect to sensor importance $W_e$. Additional vertices are generated by first sampling a distance with respect to transmittance $T_r$, followed by a new direction with respect to the phase function $p_m$. The light path is terminated, adding its contribution to the estimate of pixel intensity $I$, when it hits an emitter. This means that all terms of the contribution function $f$ other than the emission $L_e$ are importance sampled. This can lead to large variance, as the probability of hitting an emitter can be small.

Light paths can also be constructed with the first vertex on then emitter, called *light tracing*, but the problem remains the same, as the probability of hitting the sensor is small. Practical implementations of path tracing often employ a technique known as *next event estimation*, connecting the incrementally constructed path to an emitter at each scattering location. This dramatically reduces variance, as a full path is constructed at each vertex.

### 2.3.2. Bidirectional Path Tracing

In *bidirectional path tracing*, two light paths are constructed, one starting at the emitter, the other starting at a sensor, such that both the emission $L_e$ and the sensor importance $W_e$ are importance sampled. Full light paths are constructed by deterministically connecting the end vertices of two path prefixes, using $s$ vertices of the emitter path and $t$ vertices of the sensor path. Because the geometry term between the two connecting vertices has a singularity, which is not importance sampled, bidirectional path tracing suffers from infinite variance [Kal63]. This can be alleviated by using multiple importance sampling (2.26) over the $s + t - 1$ techniques, in which any full path can be connected, yielding finite and often quite low variance. Note that path tracing with next event estimation corresponds to bidirectional path tracing where the light path prefix always has $s = 1$ vertices.

### 2.3.3. Sampling free flight distance

An important problem in both rendering algorithms is importance sampling the transmittance term $T_r$, also known as sampling the *free flight distance*. In homogeneous media, we can sample from the probability density function $pdf(t) = \frac{T_r(t)}{\int_0^\infty T_r(t)\,d(t)} = \sigma_t e^{-\sigma_t t}$, obtained by normalizing the transmittance function $T_r$. We can then compute the cumulative density function $cdf(t) = \int_0^t pdf(t)\,dt = 1 - e^{-\sigma_t t}$ and invert it, to sample distances as $t = -\frac{\ln(1-\xi)}{\sigma_t}$, where $\xi$ is a uniform random variable in the range $[0, 1)$.

In heterogeneous media, one method to sample distances is based on *ray-marching*, computing the distance $t = \|\mathbf{x} - \mathbf{x}'\|$ along a ray where $T_r(\mathbf{x} \leftrightarrow \mathbf{x}') = \xi$. The probability density function resulting from this process is $pdf(t) = T_r(\mathbf{x} \leftrightarrow \mathbf{x}')\sigma_t(\mathbf{x}')$. Due to the step size used for ray-marching being on the order of the density grid spacing, this method can be very slow on large density grids. Also, due to computing the transmittance using a finite step size, the method is generally biased.

A more popular approach for sampling distances in heterogeneous media is known as *Woodcock tracking*, with its origin in particle physics [Col68]. The basic idea is to compute the upper bound of the extinction coefficient $\bar{\sigma}_t$ in the heterogeneous medium and use it to sample a distance in the same way as for a homogeneous medium, to obtain a potential scattering location $\mathbf{x}$. The scattering event is then accepted with a probability of $p = \frac{\sigma_t(\mathbf{x})}{\bar{\sigma}_t}$. If rejected, a new scattering location is sampled, repeating the same process but starting at the previous location $\mathbf{x}$. In contrast to ray-marching, Woodcock tracking is unbiased and potentially much faster, as it is not tied to a minimum step size. The main disadvantage of Woodcock tracking is that the probability density function for the sampled distances is not known, which is typically not a problem in path tracing, but prevents the computation of MIS weights accounting for the transmittance terms in bidirectional path tracing.

## 2.4. Machine Learning

*Machine learning* is the study of automatically learning programs from data instead of manually constructing them. Over the last decade, machine learning has rapidly spread throughout many different fields in computer science. In todays world, machine learning is ubiquitous in countless applications such as web search, recommender systems, stock trading, image recognition, language translation and many others. The success of machine learning in recent years is due to the ever growing availability of large datasets and computational power, combined with some major breakthroughs in the design of learning algorithms. Machine learning can be classified into three major categories:

- **Supervised learning** Based on inputs and corresponding outputs, or labels, the goal of the learning algorithm is to find a general mapping from inputs to outputs.
- **Unsupervised learning** Based only on inputs, or unlabeled data, the learning algorithm tries to find the general structure of the data to uncover hidden patterns.
- **Reinforcement learning** Based on interaction with a dynamic environment, the learning algorithm tries to maximize some reward.

Furthermore, machine learning can be categorized into different types of outputs that a learned system is desired to provide such as:

- **Classification** The input data is labeled with different classes. Using supervised learning, the goal of the system is to assign one or more classes to unseen input.
- **Regression** Similar to classification, the goal is to map unseen input to continuous output values.
- **Clustering** Using unsupervised learning, a set of inputs is divided into a number of clusters.

The goal of this thesis is to predict light transport, thus we are mainly focused on regression problems.

### 2.4.1. Regression

In the typical regression problem, a possibly unknown true function $f$ is approximated by a function $\hat{f} : X \to \mathbb{R}$, where $X \in \mathbb{R}^N$ is the input space. The quality of the approximation is usually measured using some error function such as *mean squared error*:

$$mse(\hat{f}) = E[(\hat{f} - f)^2]. \tag{2.37}$$

The function $\hat{f}$, called *predictor* or *model*, is obtained by running a learning algorithm on a *training set* consisting of $n_{train}$ data points $\{(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2)), \ldots, (\mathbf{x}_{n_{train}}, f(\mathbf{x}_{n_{train}}))\}$. The approximation error (2.37), or *generalization error*, can be estimated by evaluating the error function on a different set of $n_{test}$ data points, referred to as the *test set*:

$$mse(\hat{f}) \approx \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \left[\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i)\right]^2. \tag{2.38}$$

Although MSE is a common choice to define the generalization error, many other error functions can be used [WF05].

### 2.4.2. Linear Regression

The linear regression model assumes that the relationship between the input and output of $f$ is linear. The model can be written as:

$$\hat{f}(\mathbf{x}_i) = \mathbf{x}_i^T \mathbf{w}, \tag{2.39}$$

where $\mathbf{w}$ are the *model parameters*. To find the model parameters, one can apply the method of *least squares* [LH95], which minimizes the squared error between the input and the output:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \sum_{i=1}^{n_{train}} \left(\mathbf{x}_i^T \mathbf{w} - f(\mathbf{x}_i)\right)^2. \tag{2.40}$$

### 2.4.3. Nonlinear Regression

Some nonlinear models can be transformed to linear models, also called *linearization*. As an example, we can transform a simple nonlinear model such as:

$$\hat{f}(\mathbf{x}_i) = \alpha e^{\mathbf{x}_i^T \mathbf{w}} \tag{2.41}$$

into

$$\ln(\hat{f}(\mathbf{x}_i)) = \ln(\alpha) + \mathbf{x}_i^T \mathbf{w}, \tag{2.42}$$

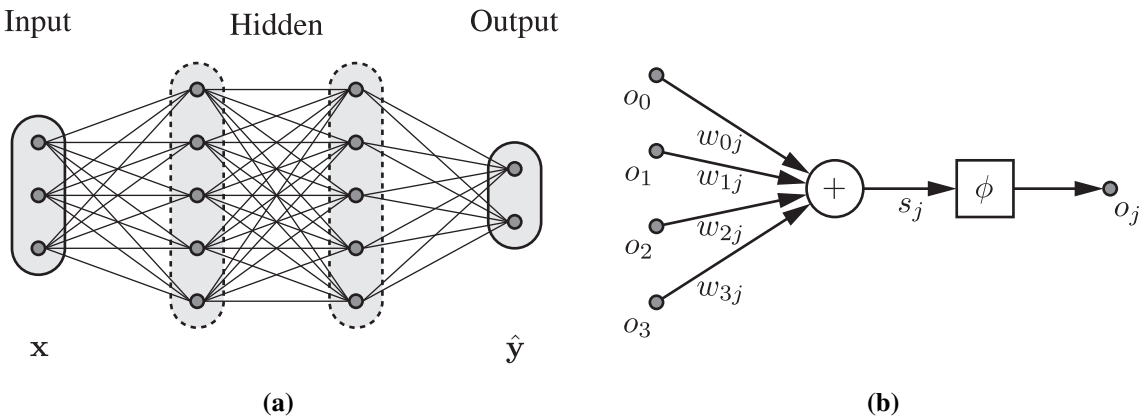



***Figure 2.5.:*** *Structure of a Multilayer Perceptron network: (a) shows a network with four layers where* $\mathbf{x}$ *is the input and* $\hat{\mathbf{y}}$ *is the predicted output, (b) shows a single unit in the network, where the outputs* $o_i$ *of the previous layer are weighted with* $w_{ij}$*, summed together and fed through the activation function* $\phi$ *to generate the units output* $o_j$*.*

and use least squares to obtain the model parameters $\alpha$ and $\mathbf{w}$. However, we have to consider the fact that we no longer minimize the squared error in the original space, but in the linearized space.

An improved technique for nonlinear regression is known by the *kernel method* [STC04], which relies on a user defined kernel computing the inner product between two data points, allowing to transform the original data points into a much higher dimensional space without inflating the size of the data points.

### 2.4.4. Artificial Neural Networks

Artificial neural networks, inspired by the humans central nervous system, consist of a set of nodes, also called *neurons* or *units*, connected together to form a network. With its origin in the early 1940s [MP43], it took several decades until ANNs became a practical computational model by development of an efficient learning algorithm known as *backpropagation* [Wer74], [RHW88]. With the advent of *deep learning* in the late 2000s, ANNs gained in popularity as they consistently outperformed well established machine learning techniques in many different areas.

### 2.4.5. Multilayer Perceptron

The multilayer perceptron, or MLP, is a feed-forward artificial neural network and consists of three or more layers. The first layer, or *input layer*, consists of a set of inputs to the network. The last layer, or *output layer*, consists of a set of outputs from the network. The layers in-between are called *hidden layers*. MLPs are fully connected networks, meaning that each unit of a layer connects to all units of the next layer. See Figure 2.5a for an example of an MLP.

Three layer MLPs are proven to be universal function approximators [HSW89], meaning that

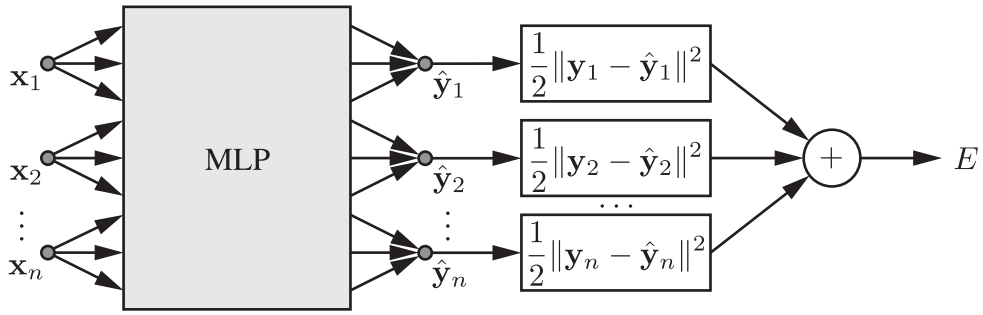



***Figure 2.6.:*** *The extended network to compute error $E$ from a training set with $\mathbf{x}_1, \ldots, \mathbf{x}_n$ as inputs and $\mathbf{y}_1, \ldots, \mathbf{y}_n$ as target values.*

any nonlinear function can be represented given enough units in the hidden layer. Depending on the type of *activation function* in the units, MLPs can be used for both classification and regression problems. We will focus on the latter and write the regression model using an MLP as:

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{x}), \tag{2.43}$$

where $\mathbf{x}$ is the input, $\hat{\mathbf{y}}$ is the output and $\hat{f}$ is the network function. The number of units in the input layer is equal to the dimensionality of $\mathbf{x}$. Similarly, the number of units in the output layer is equal to the dimensionality of $\hat{\mathbf{y}}$. When evaluating the model $\hat{f}$, the input layer is directly fed with the components of $\mathbf{x}$ and the components of $\hat{\mathbf{y}}$ are given by the outputs of the units in the output layer after propagating the inputs through the network. The output of each unit in the hidden and output layer is defined as:

$$o_j = \phi(s_j) = \phi\left(\sum_{i=1}^{n} w_{ij} o_i\right), \tag{2.44}$$

where $\phi$ is the activation function, $s_j$ is the input to the unit, $n$ is the number of units on the previous layer, $w_{ij}$ is the weight between units $i$ and $j$ and $o_i$ are the outputs of the units of the previous layer. Note that it is common practice to use an additional input of constant value 1, to allow biasing the input to the activation function. See Figure 2.5b for a visualization of a single unit in the network. For regression problems, a common activation function is the continuous differentiable *sigmoid* function, defined as:

$$\phi(z) = \frac{1}{1 + exp(-z)}, \tag{2.45}$$

$$\frac{d\phi}{dz}(z) = \phi(z)(1 - \phi(z)). \tag{2.46}$$

In order to learn the weights in an MLP, we first have to define an error function, or *objective function*, a typical choice being the mean squared error:

$$E = \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2, \tag{2.47}$$

where $n$ is the size of the training set, $\hat{\mathbf{y}}_i = \hat{f}(\mathbf{x}_i)$ are the *predicted values*, $\mathbf{y}_i = f(\mathbf{x}_i)$ are the *target values* and the factor $\frac{1}{2}$ is used to simplify the derivative. To minimize the error $E$, we first build an extended network as shown in Figure 2.6. Each training sample $\mathbf{x}_i$ is fed into the MLP and its predicted value $\hat{\mathbf{y}}_i$ is used to compute the error term $\frac{1}{2}\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2$, which is then summed up to the total error $E$. This extended network can compute the total error $E$ for a given training set. The weights of the MLP network are the only parameters that can be adjusted to minimize the error. Because $E$ is calculated by a composition of output functions at each unit, it is a continuous differentiable function of the $l$ weights $w_1, \ldots, w_l$ in the MLP network. The error $E$ can therefore be minimized using gradient descent based on the gradient of $E$:

$$\nabla E = \left( \frac{dE}{dw_1}, \ldots, \frac{dE}{dw_l} \right). \tag{2.48}$$

The weights are iteratively updated using the increments:

$$\Delta w_i = -\gamma \frac{dE}{dw_i} \quad \forall i \in \{1, \ldots, l\}, \tag{2.49}$$

where $\gamma$ is the learning rate. Efficiently learning the weights within the extended network can be done with the backpropagation algorithm, which on a high level performs an iteration of the following three steps:

1. **Feed-forward** The samples of the training set are fed through the network, the outputs and local derivatives with respect to each sample are computed for all units within the network.
2. **Backpropagation** The total error $E$ is propagated back through the network to obtain the partial derivatives for each weight.
3. **Weight update** The weights are updated based on a global learning rate $\gamma$.

Refer to [Roj96] for a more thorough discussion of the backpropagation algorithm. Due to memory constraints, it is typically not feasible to do gradient descent using the complete training set. An alternative is to compute the gradients based on single data points during gradient descent, known as *stochastic gradient descent*. In practice, it is preferred to use *mini-batches* to compute the gradients, which allows to take advantage of vectorization and often leads to smoother convergence.

In order to increase the rate of convergence during training, it is common practice to normalize the input and output values in the dataset [LBOM98]. This can be achieved by subtracting the mean and dividing by the standard deviation in each column of the dataset. Furthermore, the rate of convergence can strongly be affected by the initialization of the weight parameters in the network. Glorot et al. [GB10] propose to use weights from a uniform distribution:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n+n_p}}, \frac{\sqrt{6}}{\sqrt{n+n_p}}\right], \tag{2.50}$$

where $n$ is the size of the current layer and $n_p$ is the size of the previous layer.

# 3

# Related Work

## 3.1. Rendering of Participating Media

Rendering is a well explored and actively researched topic in computer graphics. In this section, we will focus on techniques used for rendering images with scenes containing participating media, proposed over the course of the past two decades. Most methods aim at solving the path integral to compute pixel intensities (2.34) in efficient ways, which is challenging due to its generally infinite dimensionality.

Pattanaik and Mudur [PM93] have used a Monte Carlo algorithm that uses random walks starting from light sources to render participating media. Lafortune and Willems [LW96] extended this algorithm to use bidirectional random walks, starting both from the camera and the light sources, and used multiple importance sampling [VG95] to combine contributions of paths obtained by connecting all valid path prefixes. Pauly et al. [PKK00] have extended the Metropolis Light Transport algorithm [VG97] by incorporating support for participating media.

Great effort has been put into researching different algorithms for sampling *free flight distances* inside participating media, a core component of all algorithms using Monte Carlo random walks. While this is a trivial problem for homogeneous media, as one can simply draw samples from an exponential distribution, the problem is much harder for heterogeneous media. Initially, *ray-marching* [PH89] and the *inversion method* have been used for sampling distances proportional to transmittance. While this method provides a proper PDF with respect to distance, it is biased and rather inefficient due to the overhead of ray-marching. An alternative for sampling distances with its origin in particle physics, known as *Woodcock tracking* [Col68], has later been adapted for unbiased rendering of heterogeneous media [MBJ+06, RSK08]. The efficiency of the method has been improved by using spatial subdivision [YIC+10, SKTM11] to allow for taking larger steps in optically thin regions of the medium. Unfortunately, Woodcock tracking can only be used to stochastically sample free flight distances but does not provide a

PDF. The noisy estimates of transmittance based on Woodcock tracking have been improved by applying the concept of *control variates* [NSJ14]. Georgiev et al. [GKH+13] have replaced the independent sampling of the phase function and free flight distance with a joint sampling scheme to improve variance.

An alternative to path tracing for solving light transport is a two stage algorithm known as *photon mapping* [Jen96]. During the first stage, random walks are started from the light sources and *photons* are stored at scattering locations inside the scene. In the second stage, the *photon map* is queried using density estimation kernels to evaluate incoming light at arbitrary locations in the scene. Despite adding bias, photon mapping has been widely adapted because of its efficiency. Jensen and Christensen [JC98] have extended photon mapping with support for participating media. The method has later been improved by using a better density estimation scheme [JZJ08]. In a similar approach to photon mapping, radiance is cached at various points within the medium while solving the light transport integral, in order to reuse expensive computation [JDZJ08].

In more recent work, the point based photons have been replaced with higher dimensional primitives such as photon beams [JNT+11]. Reusing much of the same framework as photon mapping, so called *many light methods* have replaced the point based photons with *virtual point lights* [WABG06] and photon beams with *virtual ray lights* [NNDJ12]. In most recent work, the strengths of various estimators for light transport have been combined into a unified rendering algorithm [KGH+].

In general it is not possible to find closed form solutions to light transport within participating media, which is among the reasons for the prevalence of Monte Carlo algorithms. One exception is light transport in an infinite homogeneous medium, where an analytical solution for multiple-scattering can be derived [LK11]. In other work, approximate analytical solutions for single-scattering within homogeneous media have been derived and applied in real-time rendering [SRNN05, PSS11].

Monte Carlo based rendering algorithms can be highly inefficient when rendering media with high scattering albedo, because light paths can scatter many times before hitting a light source and contributing to pixel estimates. Similar problems occur when dealing with strongly anisotropic phase functions. This gave rise to many biased rendering techniques that approximate light transport within strongly scattering media. One important area is rendering of *sub-surface scattering* effects, a model where light can penetrate through a surface and scatter within the medium behind it, such as skin. Among the most important techniques for rendering sub-surface scattering are diffusion based methods, originally brought to the graphics community by Stam [Sta95]. Jensen et al. [JMLH01, JB02] applied the method to rendering. Later, diffusion was incorporated into the photon mapping framework [DJ08] to allow for oblique and indirect illumination effects. The accuracy of the diffusion approximation has been improved further [dI11, D'E12], and various methods for improving accuracy of oblique illumination have been proposed [d'E13, HCJ13b, FHK14]. The diffusion theory has also been applied to compute light propagation in heterogeneous media defined on a grid [DKE14].

Given a radiance field with bounded directional frequencies, different sets of medium properties can result in the same solution of the RTE. This is known as *similarity theory* [WPW89a, WPW89b]. In a simplified first-order form, similarity theory allows to approximate an anisotropic medium with an isotropic medium with adjusted scattering parameters. This is commonly used

to derive medium properties for rendering with diffusion based methods [FCJ07]. In more recent work, similarity theory has been used in its high-order form, to reduce the scattering albedo in order to accelerate rendering using Monte Carlo methods [ZRB14].

Light transport inside clouds is a difficult problem, as light scatters many times due to lack of absorption. Real-time rendering of clouds can only be achieved with substantial approximations such as folding multiple-scattering effects into a model for direct evaluation [REK+04]. In other work, iterative methods have been used to propagate light along grids to account for multiple-scattering [SKSU05, Fat09, ERDS14]. Bouthors [Bou08] has thoroughly analyzed light transport within clouds and developed real-time and interavtive rendering algorithms [BNL06, BNM+08] based on data fitting of extensive Monte Carlo simulations performed within a slab of cloud. Some of his findings have also been applied to offline rendering [WKL13, Wre15], allowing to render complex clouds with multiple-scattering in full length feature movies.

## 3.2. Machine Learning in Rendering

In this section, we review related work in applications of machine learning to rendering problems. This is a relatively new research topic with many avenues to explore. Artificial neural networks have gained much attention with the recent successes of *deep learning* [LBH15] techniques, significantly reducing error rates of classification and regression problems in many fields. Bengio et al. [BCV12] discuss how data representation affects the performance of machine learning algorithms and show that learning data representations using general priors can be superior over manually designing them using domain specific knowledge.

Ren et al. [RWG+13] have used regression functions based on feed-forward neural networks for real-time estimation of indirect illumination from features at surface points, such as position, viewing angle and local lighting conditions. The regression functions have to be trained on a per-scene basis, with extensive precomputation based on offline renderings of the scene from many different viewpoints and lighting scenarios. In later work [RDL+15], they formulate light transport by a product of a transport matrix and a lighting vector and applied regression using neural networks to estimate the transport matrix based on a sparse set of input images.

Vorba et al. [VKŠ+14] have used machine learning on a per-scene basis to guide the sampling of light paths in bidirectional path tracing. They use a parametric mixture model to encode a spatially varying importance function for importance sampling local scattering directions, with respect to global light transport.

Nalbach et al. [NAM+16] have used deep convolutional networks for real-time estimation of screen-space shading effects, such as ambient occlusion, depth-of-field and subsurface scattering and others based on features from deferred shading buffers including position, depth, normal and material properties.

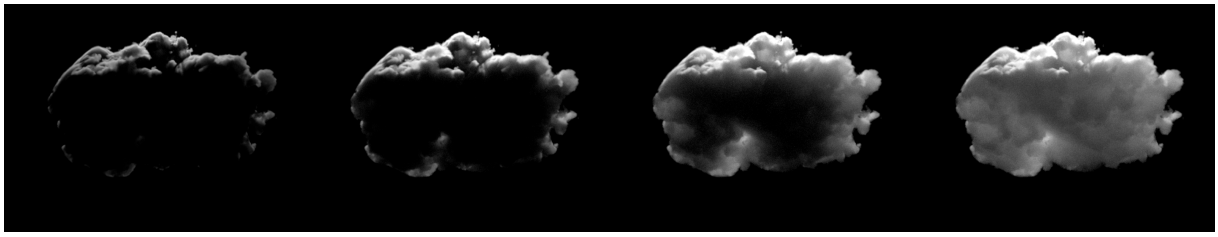
Kalantari et al. [KBS15] proposed to use non-linear regression for estimating filter parameters for denoising images from noisy Monte Carlo renderings.

In the field of particle fluid dynamics, regression forests have been used to estimate the accelera-

tion of individual particles based on a large training set of simulations obtained from traditional fluid solvers [LJS+15].

# 4

# Learning High-Order Scattering



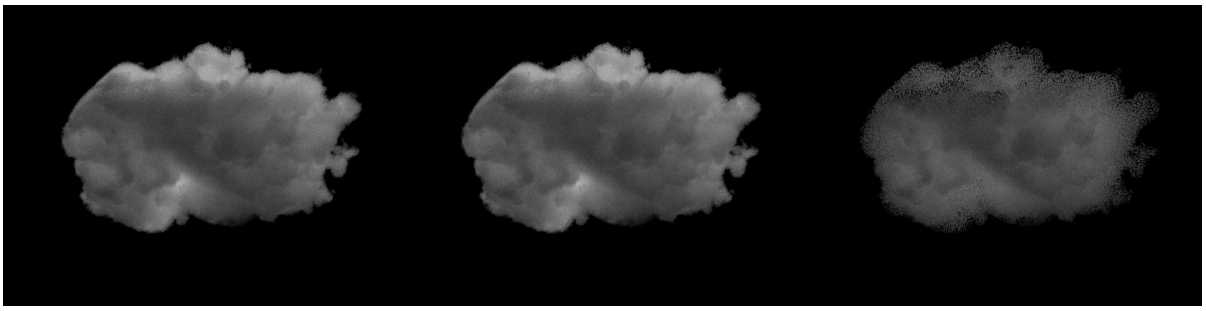
**(a)** 1 bounce, 1x **(b)** 4 bounces, 2x **(c)** 16 bounces, 5x **(d)** Full transport, 10x

***Figure 4.1.:*** *A cloud rendered up to different number of bounces and the corresponding relative render time using the same number of samples per pixel.*

In this chapter, we introduce our idea of using machine learning to predict the high-order scattering contribution when rendering clouds. Figure 4.1 illustrates how high-order scattering affects the visual appearance of a small cloud. For the medium properties we use a scattering albedo of $\alpha_s = 0.99$, a forward scattering Henyey Greenstein phase function with $g = 0.85$ and a mean free path of $10m$, assuming a cloud with diameter of roughly $1km$. We illuminate the cloud with a single directional light source. The scene was rendered with our path tracer using single-scattering, 4 bounces, 16 bounces and an unbounded number of bounces, capturing the full light transport. This comparison clearly illustrates the amount of energy that is lost by omitting high-order scattering and its importance to the overall plausibility of the rendered image.

We also compare the relative increase in render time using an equal number of samples per pixel, showing an increase of a factor of $10$ between rendering single-scattering and the full light transport. At first sight, this might seem like a reasonable cost to pay for the increase of visual quality, however, the numbers are misleading. First, at the same number of samples per pixel, the images rendered with less bounces are better converged. Second, in scenes with larger or denser clouds and cloudscapes, light can scatter many times more before contributing

to the pixel value, leading to much more expensive random walks and a significant increase in variance. For these reasons, it quickly becomes infeasible to render complex scenes with full light transport in a reasonable amount of time.



**(a)** 1 bounce **(b)** 4 bounces **(c)** 16 bounces

***Figure 4.2.:*** *Visualization of the missing high-order scattering contribution when using a fixed number of bounces.*

Figure 4.2 shows the contribution of high-order scattering in isolation, obtained by subtracting the rendered image of single-scattering, 4 and 16 bounces from the image rendered with full light transport. This contribution is what we have to predict in order to obtain the full light transport. We can observe, that the high-order scattering contribution is more diffuse and contains less variation compared to low-order scattering [Bou08]. Therefore, errors in the high-order scattering contribution should generally be less discernable than errors in the low-order contribution, allowing for some tolerance in the accuracy of the prediction.

## 4.1. Prediction in Screen-Space

In the early stages of our investigations we discussed the idea of using a prediction scheme in *screen-space*, where a number of *feature buffers* is used to predict the final output. Obviously, it is impossible to predict any reasonable output from a single image rendered with only single-scattering or a low number of bounces, as most areas of the image are dark and contain no usable information. By rendering the cloud multiple times, with progressively thinner participating media, we might obtain a number of feature buffers that contain more information and can be used to predict the final output. Additional feature buffers may be created from rendering auxiliary buffers such as transmittance of the participating media, again in a series of decreasing medium density. A possible set of feature buffers is visualized in Figure 4.3.

Similar to the work on screen-space shading [NAM+16], one could use convolutional neural networks to predict the final image from the feature buffers. We have not pursued the idea of predicting high-order scattering in screen-space for the following reasons:

- Artifacts at the border of the image.
- Artifacts due to missing information in the feature buffers.
- Precondition on reasonably well converged feature buffers to do the prediction, prohibiting the use of progressive rendering.

- Cost of acquiring the training set.

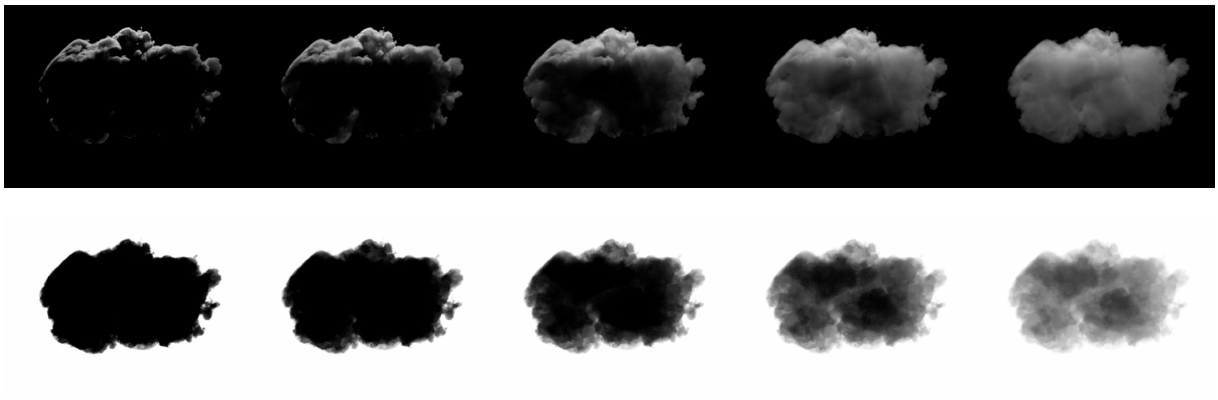


***Figure 4.3.:*** *A possible set of feature buffers for a screen-space prediction algorithm. The top row shows single-scattering, the bottom row shows transmittance, both in a series of decreasing density of the participating medium.*

Prediction in screen-space remains an interesting topic for future research and might be a good fit for interactive or real-time applications.

## 4.2. Prediction in World-Space

Predicting high-order scattering in *world-space* implies to learn how much light arrives at a given point in the scene. The quantity of light at each point and direction in the scene is encoded in the 5D radiance function $L(\mathbf{x} \leftarrow \vec{\omega})$, also known as the *plenoptic function*. Predicting the radiance function is an inherently difficult problem, as it depends on the specific configuration of the scene, including the spatial distribution of participating media and their medium properties, as well as all the light sources. Therefore, depending on the complexity of the scene, the function potentially depends on an infinite number of variables. Successfully predicting the radiance function relies on the assumption that the *features* provided to the model are a good enough representation of the scene. It follows naturally, that the complexity of the scene needs to be constrained in order to reduce the feature space to a reasonable dimensionality. We investigated two models using different constraints, which we discuss in the following two chapters.

# 5

# Global Transport Model

In this section, we introduce our *global transport model*, a model to predict the total light transport to a point within a constrained scene configuration. Our model can be used within a standard unidirectional path tracer, using random walks to sample the low-order scattering contribution, and replace sampling of longer random walks by a prediction from the model, to compute the high-order scattering contribution.

## 5.1. Model

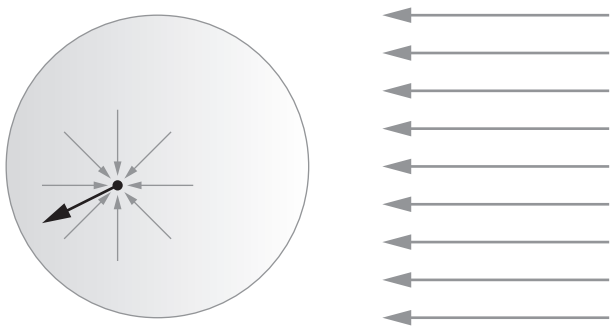


***Figure 5.1.:*** *Overview of our global transport model. We predict in-scattered radiance inside a homogeneous participating medium illuminated by a single directional light source.*

We constrain the scene configuration for our global transport model by using a single directional light source and a single homogeneous participating medium with a convex boundary, such as

a sphere or an ellipsoid. We predict the total light transport from the directional light source to a point within the medium. Instead of predicting incident radiance $L(\mathbf{x} \leftarrow \vec{\omega})$, left hand side of (2.16), we predict in-scattered radiance $L_i(\mathbf{x} \rightarrow \vec{\omega})$, left hand side of (2.7). First, the in-scattered radiance function is easier to learn, because it is generally smoother than the incident radiance function within a strongly anisotropic medium. Second, directly predicting the in-scattered radiance, removes 2 dimensions from the integral we have to evaluate during rendering, reducing both the cost and the variance. See Figure 5.1 for an overview of the scene configuration used in our global transport model.

The directional light source emits constant radiance $L_L = 1$ in direction $\vec{\omega}_L = (0, 0, -1)^T$. The homogeneous participating medium, with convex shape $S$, has extinction coefficient $\sigma_S$, scattering albedo $\alpha_S$ and uses the Henyey Greenstein phase function with eccentricity $g_S$. Let $S(\mathbf{x})$ be a binary function, that is 1 if the point $\mathbf{x}$ is inside to the convex shape $S$, and define the medium properties as follows:

$$\sigma_t(\mathbf{x}) = \begin{cases} \sigma_S, & \text{if } S(\mathbf{x}) = 1, \\ 0, & \text{otherwise,} \end{cases} \tag{5.1}$$

$$\sigma_s(\mathbf{x}) = \begin{cases} \alpha_S \sigma_S, & \text{if } S(\mathbf{x}) = 1, \\ 0, & \text{otherwise,} \end{cases} \tag{5.2}$$

$$p(\vec{\omega} \leftrightarrow \vec{\omega}') = p_{hg}(\vec{\omega} \leftrightarrow \vec{\omega}', g_S). \tag{5.3}$$

The volume rendering equation (2.16) for this constrained scene configuration simplifies to the following sum of reduced radiance and accumulated in-scattered radiance:

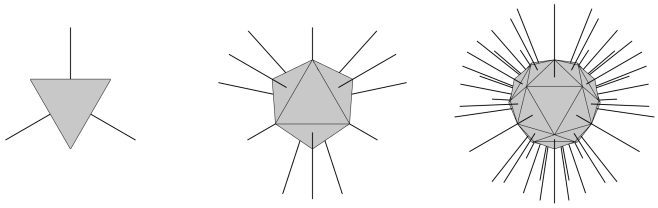
$$\begin{aligned} L(\mathbf{x} \leftarrow \vec{\omega}) = & \underbrace{T_r(\mathbf{x} \leftrightarrow r(\mathbf{x}, \vec{\omega}))\, \delta(1 - \langle \vec{\omega}, -\vec{\omega}_L \rangle)\, L_L}_{\text{reduced radiance}} \\ & + \underbrace{\int_{\mathbf{x}}^{\mathbf{x}_e} T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t) \sigma_s(\mathbf{x}_t) L_i(\mathbf{x}_t \rightarrow -\vec{\omega})\, ds(\mathbf{x}_t)}_{\text{accumulated in-scattered radiance}}, \end{aligned} \tag{5.4}$$

where $\delta$ is the dirac delta function, $1 - \langle \vec{\omega}, -\vec{\omega}_L \rangle$ is zero only if $\vec{\omega}$ is in opposite direction of $\vec{\omega}_L$ and $L_i(\mathbf{x}_t \rightarrow -\vec{\omega})$, the in-scattered radiance (2.7), is defined as previously:

$$L_i(\mathbf{x} \rightarrow \vec{\omega}) = \int_{\Omega_{4\pi}} p(\vec{\omega}' \leftrightarrow \vec{\omega}) L(\mathbf{x} \leftarrow \vec{\omega}') d\vec{\omega}'. \tag{5.5}$$

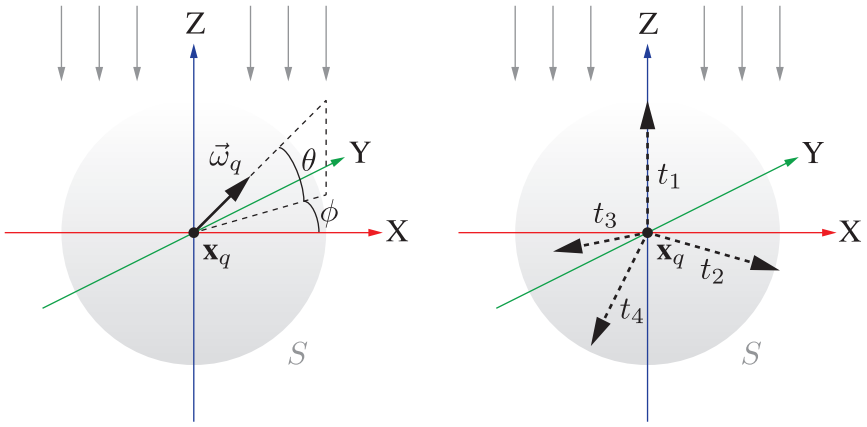
Because the reduced radiance term in the volume rendering equation has an analytic solution, we can efficiently estimate the in-scattered radiance integral using a simple Monte Carlo path tracer. For this reason, generating large training sets of ground truth data is quite efficient, as shown later.

With the definition of light transport in place, we continue to describe the parametrization of the prediction model. First, we define a set of directions $\vec{d}_i$ for $i = 1 \dots n_D$ over the sphere,



**(a)** $n_D = 4$ **(b)** $n_D = 16$ **(c)** $n_D = 64$

***Figure 5.2.:*** *Sets of directions with different size based on subdividing a tetrahedron.*




**(a)** Query direction features $\theta, \phi$ **(b)** Transmittance features $t_i$

***Figure 5.3.:*** *Visualization of the features used in the global transport model.*

using a subdivided tetrahedron to obtain progressively refined sets as shown in Figure 5.2. We align the set of directions with respect to the light source and fix $\vec{d}_1 = -\vec{\omega}_L = (0, 0, 1)^T$. Despite not being distributed perfectly uniform across the sphere, the tetrahedron subdivision scheme allowed us to reuse the same directions for sets with different sizes $n_D \in \{4, 16, 64\}$ during our experiments. We suggest to use a more uniform distribution once the optimal number of directions is determined. With in-scattered radiance as the target function $f$, we define its predictor $\hat{f}$ as:

$$L_i(\mathbf{x}_q \rightarrow \vec{\omega}_q) = f \approx \hat{f}(\alpha_S, g_S, \theta, \phi, t_1 \dots t_{n_D}), \tag{5.6}$$

where $\mathbf{x}_q$ is the query location, $\vec{\omega}_q$ is the query direction and the features for the predictor $\hat{f}$ are defined as follows:

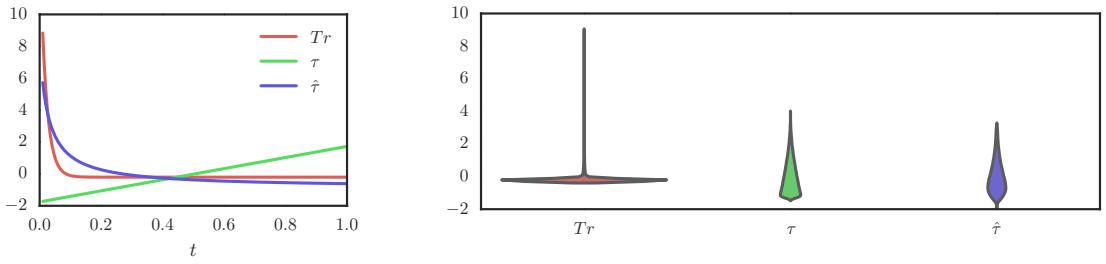
- $\alpha_S$ is the scattering albedo,
- $g_S$ is the phase function eccentricity,
- $\theta, \phi$ are the spherical coordinates of the query direction $\vec{\omega}_q$ (see Figure 5.3a),
- $t_i = T_r(\mathbf{x}_q \leftrightarrow r(\mathbf{x}_q, \vec{d}_i))$ is the transmittance along a ray starting at the query location $\mathbf{x}_q$

in direction of $\vec{d_i}$ (see Figure 5.3b).

Note that the transmittance features $t_i$ implicitly represent the shape of the medium $S$, its extinction coefficient $\sigma_S$, as well as the query location $\mathbf{x}_q$ in relation to the shape. In consequence, $S$, $\sigma_S$ and $\mathbf{x}_q$ are not used as explicit features.

## 5.2. Training

We decided to use nonlinear regression using an MLP network for learning our predictor (5.6). Being universal function approximators, MLPs are a good fit for this kind of problem and have been used successfully in other rendering applications [RWG+13]. We used the *mean squared error* as our loss function. The dataset for learning consists of 5 million data points, split into 4 million data points for training and the remaining 1 million for testing. To train the MLP network, we used *TensorFlow* [AAB+15], a Python framework by Google for machine learning applications with focus on deep neural networks.




**(a)** **(b)**

***Figure 5.4.:*** *(a) shows a comparison of the normalized values of transmittance $Tr$, optical depth $\tau$ and our inverse log-transformed optical depth $\hat{\tau}$, with respect to distance $t$, in a medium with $\sigma_t = 50$. (b) shows a violin plot, visualizing the actual distributions of the feature values in the training set for data points with $\sigma_t \approx 50$.*

### 5.2.1. Feature Transformations

In order to improve convergence during training, we used the standard practice of normalizing all feature values by subtracting the mean and dividing by the standard deviation. During early attempts, the MLP completely failed to learn the predictor. Analyzing the dataset revealed, that the transmittance features $t_i$ quickly tend to zero in media with high extinction coefficient, leading to a heavy-tailed distribution within the training set. Therefore, we replaced the transmittance features $t_i$ with optical depth features $\tau_i = \tau(\mathbf{x}_q \leftrightarrow r(\mathbf{x}_q, \vec{d_i}))$, which measure the same quantity in a different space. This improved the accuracy, but we found that the distribution of optical depth values in our training set was still strongly skewed. In the end, we settled with inverse log-transformed optical depth features that improved accuracy considerably:

$$\hat{\tau}_i = \log\left(\frac{1}{1+\tau_i}\right). \tag{5.7}$$

We used the inverse of $1+\tau_i$ because the resulting features $\hat{\tau}_i$ have stronger correlation with the target function $L_i(\mathbf{x}_q \rightarrow \vec{\omega}_q)$, which additionally eases the learning problem. We visualize the three variants of the transmittance features in Figure 5.4 and show their distribution within the training set.

### 5.2.2. Target Function Transformations

Even with the improved feature vector, the MLP still failed to learn the predictor in strongly forward scattering media. Further analysis showed, that the target function had significant correlation with the phase function evaluated for single-scattering. Therefore, we decided to normalize the target function by dividing by the single-scattering phase function:

$$\frac{L_i(\mathbf{x}_q \rightarrow \vec{\omega}_q)}{p(\mathbf{x}_q, \vec{\omega}_q \leftrightarrow \vec{\omega}_L)} = f_n \approx \hat{f}_n(\alpha_S, g_S, \theta, \phi, \hat{\tau}_1 \ldots \hat{\tau}_{n_D}). \tag{5.8}$$

We analyzed the distribution of the target function values and again found a heavy-tailed distribution, hindering the accuracy of the MLP network. We used a log-transform on the target function to improve its distribution:

$$\log\left(1 + \frac{L_i(\mathbf{x}_q \rightarrow \vec{\omega}_q)}{p(\mathbf{x}_q, \vec{\omega}_q \leftrightarrow \vec{\omega}_L)}\right) = f_{ln} \approx \hat{f}_{ln}(\alpha_S, g_S, \theta, \phi, \hat{\tau}_1 \ldots \hat{\tau}_{n_D}). \tag{5.9}$$

In addition, we also used the same normalization scheme on the target function values as we used for the feature values, subtracting the mean and dividing by the standard deviation. See Figure 5.5 for a visualization of the marginal and joint distributions of the original target function values $f$, the normalized function values $f_n$, the log-transformed normalized function values $f_{ln}$ and both the $\theta$ and $\hat{\tau}_1$ feature values. This illustrates how the transformations led to better distributions with less heavy tails, improving the accuracy of our predictor.

### 5.2.3. Network Architecture

We initially tried many different variations of the size and number of hidden layers for the MLP network. We found that more than 2 hidden layers did not improve the accuracy, but increased both the training time and the time to evaluate the network at rendertime. On the other hand, using only a single hidden layer would require it to be of large size to get reasonable accuracy. Therefore, we settled for a configuration with two hidden layers, an input layer consisting of the feature values and a single unit on the output layer corresponding to the target function.

For training the network, we used standard stochastic gradient descent with an empirically set learning rate and a batch size of 1000 data points. Training over 10 epochs using a single workstation based on an *Intel Core i7-3930K CPU @3.2GHz* took around 30 minutes. Due to the

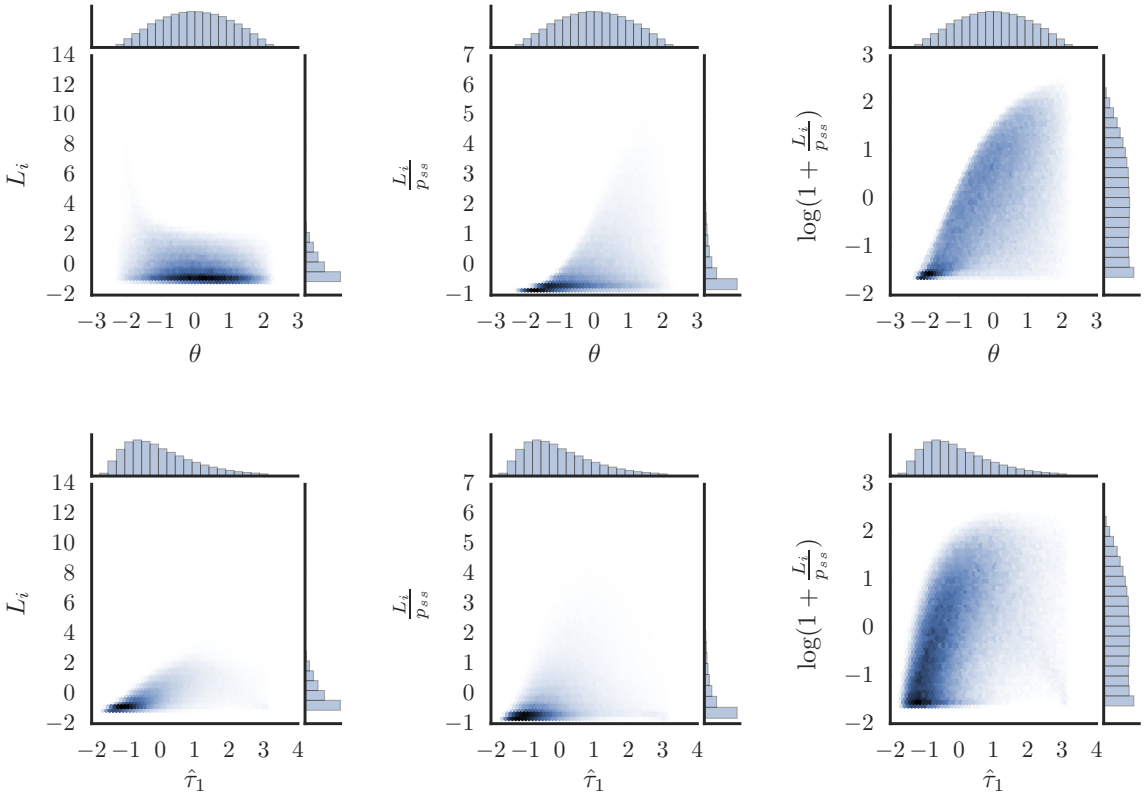



***Figure 5.5.:*** *Visualization of the marginal and joint distribution between various transformations of the target function values and the values of the two features $\theta$ and $\hat{\tau}_1$ found in the training set for $\sigma_t \approx 50$. Both axes represents the actual range of target and feature values in the training set after transformations and normalization.*

dataset being very large compared to the number of parameters in the network, we never ran into the problem of over-fitting. Figure 5.6 shows the training convergence for three network configurations with different number of units on the first and second hidden layer. Even though the configuration with $[200, 100]$ hidden units performed marginally better than the configuration with $[100, 50]$ hidden units, we used the latter due to the increased performance during rendertime.

## 5.3. Data Generation

We have written a small custom unidirectional path tracer in C++11 to generate the datasets for training. With optimizations possible due to the constrained scene configuration, we reached around a 10x speedup compared to the more generic *Mitsuba* renderer [Jak10], which we have used to validate the correctness of our results. The data generator is configured with the following parameters:

- extinction coefficient range $[\sigma_{S,min}, \sigma_{S,max}]$,

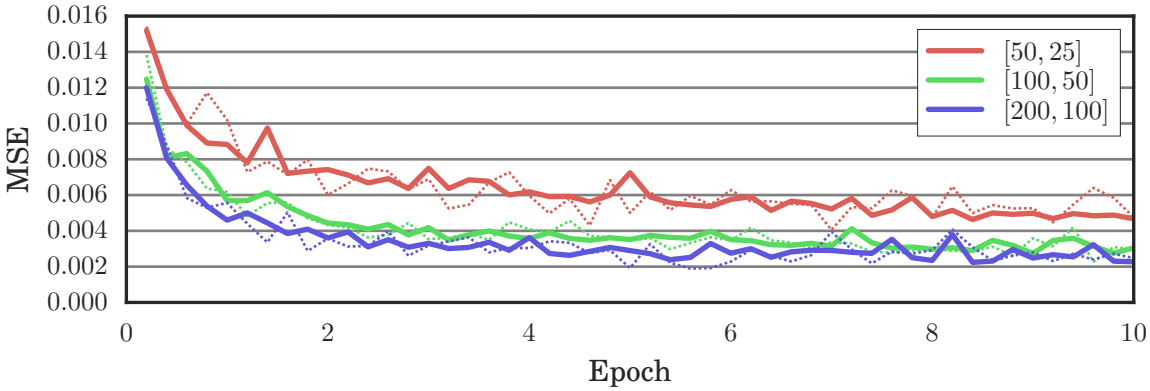



***Figure 5.6.:*** *Training convergence with different number of units on the two hidden layers. Thick lines represent the testing error, dotted lines the training error.*

- scattering albedo range $[\alpha_{S,min}, \alpha_{S,max}]$,
- phase function eccentricity range $[g_{S,min}, g_{S,max}]$,
- set of convex shapes $\mathcal{S}$ (e.g. unit sphere, unit box),
- enable/disable random rotation and scaling,
- maximum relative error and confidence level.

Based on the configured settings, the data generator performs the following steps to generate each data point:

1. Sample extinction coefficient $\sigma_S$ uniformly from $[\sigma_{S,min}, \sigma_{S,max}]$.
2. Sample scattering albedo $\alpha_S$ uniformly from $[\alpha_{S,min}, \alpha_{S,max}]$.
3. Sample phase function eccentricity $g_S$ uniformly from $[g_{S,min}, g_{S,max}]$.
4. Sample a convex shape $S$ uniformly from $\mathcal{S}$.
5. Transform shape $S$ by uniformly sampling a rotation/scaling matrix.
6. Sample query location $\mathbf{x}_q$ uniformly within $S$ and direction $\vec{\omega}_q$ uniformly on the sphere.
7. Compute in-scattered radiance $L_i(\mathbf{x}_q \rightarrow \vec{\omega}_q)$ using Monte Carlo path tracing to requested accuracy by using confidence interval estimation to control the number of samples.
8. Compute feature vector $[\alpha_S, g_S, \theta, \phi, \hat{\tau}_1 \dots \hat{\tau}_{n_D}]$ and target value $f_{ln}$ and store the data point.

For our experiments, we typically used a constant scattering albedo of $\alpha_S = 1$, a maximum relative error of $5\%$ and a confidence level of $95\%$. We used the *Euler* compute cluster running *Xeon E5-2697 v2* nodes to generate our training sets. Table 5.1 gives an overview of the typical time to compute a single data point on a single core. Even with $\sigma_S = 100$, a high enough extinction coefficient to render relatively dense clouds, the compute time for 5 million data points was consistently below 1 core year. This allowed for experimentation and running the

| | $\sigma_S = 10$ | $\sigma_S = 25$ | $\sigma_S = 50$ | $\sigma_S = 100$ |
|---|---|---|---|---|
| $g_S = 0.0$ | 0.15 | 0.79 | 3.19 | 13.66 |
| $g_S = 0.85$ | 0.12 | 0.44 | 1.71 | 6.39 |

***Table 5.1.:*** *Time in seconds to generate a single data point on a single thread of a Xeon E5-2697 v2, using a unit sphere shape $S$, scattering albedo $\alpha_S = 1$, maximum relative error of $5\%$ and confidence level of $95\%$.*

data generation process multiple times during the development of our model.

## 5.4. Rendering

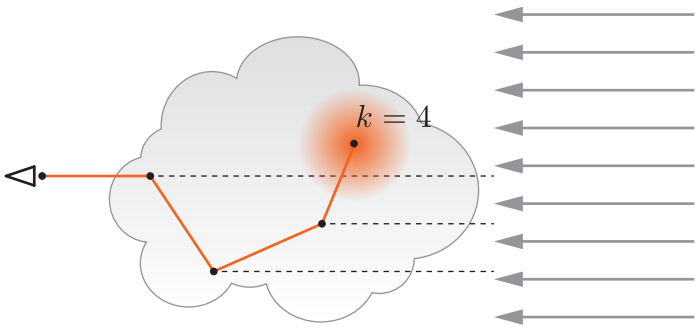



***Figure 5.7.:*** *Visualization of path tracing and predicting high-order scattering using our model. The first $k - 1$ bounces are computed using path tracing with next event estimation, in-scattered radiance on the $k$th bounce is predicted using our global transport model.*

We integrated our global transport model into a custom renderer written in C++11, allowing us to evaluate its quality and performance. The renderer uses standard unidirectional path tracing, computing direct lighting at each bounce by connecting to a random light source. Using the model is straight forward, as we can simply switch to the prediction once a configured number of $k$ bounces has been computed using random walks, as shown in Figure 5.7.

We re-implemented the evaluation of the MLP network in our renderer based on *Eigen* [GJ+10], using the weights learned during training, such that the renderer can be run standalone. For each directional light source in the scene, we construct a local coordinate system. This is used to both transform the query direction into the space used for the model, as well as transform the directions along which optical depth is evaluated into world space. The optical depth along the set of directions $\vec{d}_i$ is evaluated using ray-marching. To get the in-scattered radiance at the query location, we multiply the predicted value by the actual radiance emitted by the light source.

We found that the high-order scattering contribution from the prediction has generally much lower variance than the low-order scattering contribution computed by sampling random walks. For this reason, we use Russian roulette to cut down the number of predictions. We used an

empirically determined probability of $p = 0.25$ for evaluating the prediction. The probability $p$ could also be determined by computing a low number of samples at the start of the rendering process to estimate the ratio between variance of high-order scattering and low-order scattering.

## 5.5. Evaluation

To evaluate the quality and performance of our model, we used a set of 3 test scenes, furthermore denoted by A, B, and C. Each scene contains a single cloud defined by a density grid. We generated the clouds using a custom modeler software based on OpenVDB [MLJ+13]. The density grids are generated from simple hierarchies of spherical primitives, adding detail using displacement and advection based on noise functions. The scenes are chosen such that they should progressively reveal the limitations of the model, assuming a homogeneous medium bounded by a convex shape.

For scene A, we used a cloud shape that has an approximately convex boundary, a homogeneous interior and only spatially varying density around the surface. In scene B, we used a cloud shape that is more concave, but still has a homogeneous interior. The last scene C, uses a cloud that has a more complex shape and a heterogeneous interior created by multiplying the density grid with a 3D noise function. All scenes are lit by a single directional light source. While keeping the direction of the light source fixed, we positioned three cameras to capture different lighting scenarios, furthermore denoted by:

- **Toplit**, camera direction is perpendicular to the light direction,
- **Backlit**, camera direction is opposite to the light direction,
- **Frontlit**, camera direction is in line with the light direction.

For the medium properties, we used scattering albedo $\alpha_s = 1$, phase function eccentricity $g = 0.85$ and three different density scale values $\rho$. For scene A and B, we chose $\rho \in \{20, 40, 80\}$, roughly corresponding to mean free path distances of $50m$, $25m$ and $12.5m$, assuming the diameter of the cloud to be around $1km$. In scene C, we chose $\rho \in \{80, 160, 320\}$, because due to the heterogeneous interior, the cloud is generally less dense.

Using 3 scenes, 3 lighting scenarios and 3 density scales lead to a total of 27 configurations. We used cluster nodes based on *Xeon E5-2680 v3* with 24 threads to render each image. We rendered the following images in a resolution of $512 \times 512$ for $k \in \{1, 2, 4, 8\}$:

- **Low-order scattering reference**, computed by path tracing, stopping after $k$ bounces, using 2048 samples per pixel,
- **Full transport reference**, computed by path tracing, using 4096 samples per pixel,
- **Full transport using prediction**, computed with path tracing and our prediction model evaluated at the $k$th bounce, using 2048 samples per pixel.

Note that we used Russian roulette to terminate random walks during path tracing, using a probability equal to the path throughput. Furthermore, we also use Russian roulette with a fixed

probability of $p = 0.25$, to reduce the number of predictions, which leads to an average of around 500 predictions per pixel.

We compute the *bias* of our model as the mean of absolute error between the pixel values of the reference and prediction images in linear space. Because the low-order scattering contribution is the same in both images, the computed bias represents the actual error of our prediction. To avoid residual variance corrupting the bias estimate, we use a box filter on the images and downscale them by a factor of 16. In addition, we only consider pixels that contain information, e.g. have a value larger than zero, to avoid lowering the bias artificially.

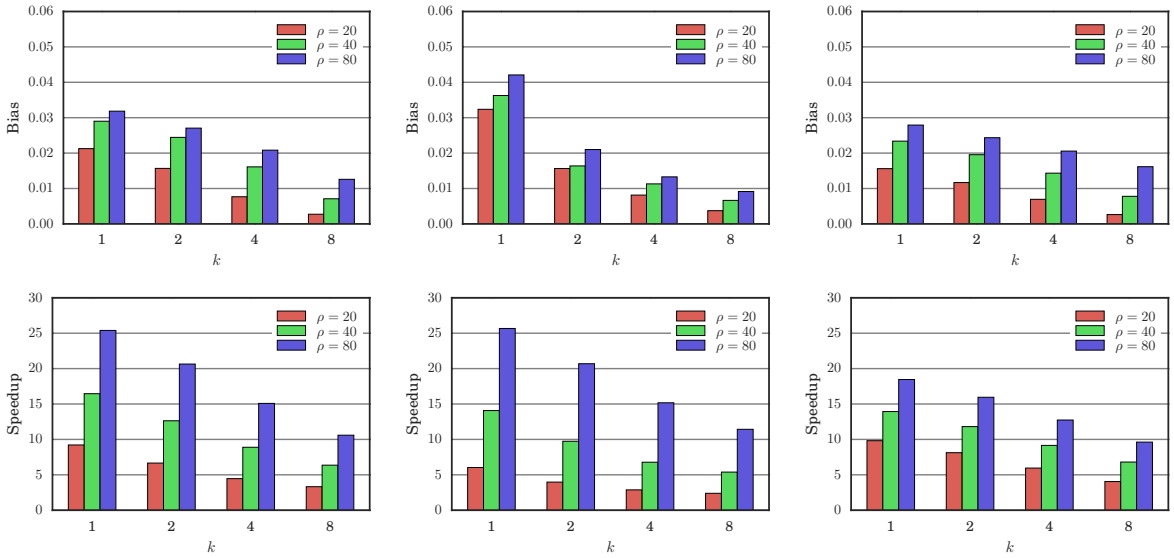
To compare the render time, we estimate the *time to unit variance*, furthermore denoted as TTUV, computed as the mean of the pixel variance multiplied by the render time. Again, we exclude pixels that contain no information, as they do not contribute to the render time. We compute the *speedup* of our method by dividing the TTUV of the reference by the TTUV of the prediction. Note that the speedup is with regards to variance only and does not take into account the bias of our method.

| | Reference | $k = 1$ | | $k = 2$ | | $k = 4$ | | $k = 8$ | |
|---|---|---|---|---|---|---|---|---|---|
| Toplit | | | | | | | | | |
| $\rho = 20$ | 390.0 | 42.3 | 9.2× | 58.5 | 6.7× | 87.4 | 4.5× | 117.4 | 3.3× |
| $\rho = 40$ | 842.8 | 51.2 | 16.5× | 66.7 | 12.6× | 94.8 | 8.9× | 132.4 | 6.4× |
| $\rho = 80$ | 1716.9 | 67.6 | 25.4× | 83.2 | 20.6× | 113.8 | 15.1× | 162.0 | 10.6× |
| Backlit | | | | | | | | | |
| $\rho = 20$ | 513.4 | 85.1 | 6.0× | 129.2 | 4.0× | 178.6 | 2.9× | 214.7 | 2.4× |
| $\rho = 40$ | 628.7 | 44.7 | 14.1× | 64.6 | 9.7× | 92.7 | 6.8× | 116.7 | 5.4× |
| $\rho = 80$ | 722.8 | 28.2 | 25.7× | 34.9 | 20.7× | 47.6 | 15.2× | 63.2 | 11.4× |
| Frontlit | | | | | | | | | |
| $\rho = 20$ | 381.7 | 38.9 | 9.8× | 47.0 | 8.1× | 64.1 | 6.0× | 94.2 | 4.1× |
| $\rho = 40$ | 1189.7 | 85.4 | 13.9× | 100.8 | 11.8× | 129.9 | 9.2× | 174.8 | 6.8× |
| $\rho = 80$ | 3049.2 | 165.2 | 18.5× | 191.1 | 16.0× | 239.3 | 12.7× | 317.0 | 9.6× |

***Table 5.2.:*** *TTUV in seconds and corresponding speedup for scene A under different lighting scenarios.*

We report the TTUV and speedup for scene A in Table 5.2 and show a comparison of both the bias and speedup in Figure 5.8. We visually compare our method against the reference for the same scene, limited to the toplit configuration, in Figure 5.9. We show three groups of images, corresponding to the 3 different density scales. In each group, the top row shows the full light transport rendered using our method, evaluating the prediction at the $k$th bounce, as well as the reference. The bottom row is split into 4 sub-images:

- **Top-left**, shows the low-ordering scattering contribution,
- **Top-right**, shows the reference high-order scattering contribution,




**(a)** Toplit **(b)** Backlit **(c)** Frontlit

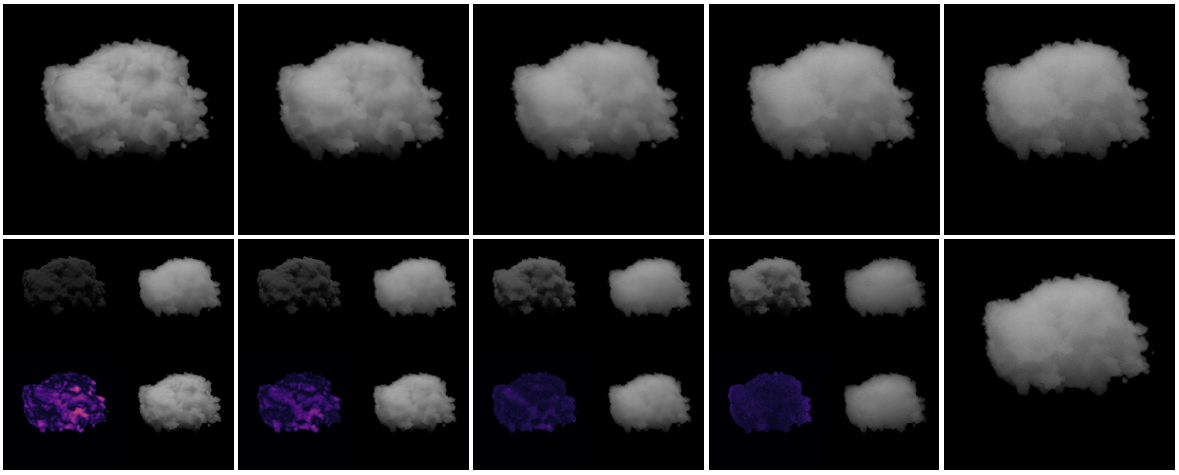
***Figure 5.8.:*** *Comparison of bias (top row) and speedup with respect to TTUV (bottom row) for scene A under different lighting scenarios.*

- **Bottom-right**, shows the predicted high-order scattering contribution,
- **Bottom-left**, shows the error of the predicted high-order scattering contribution.

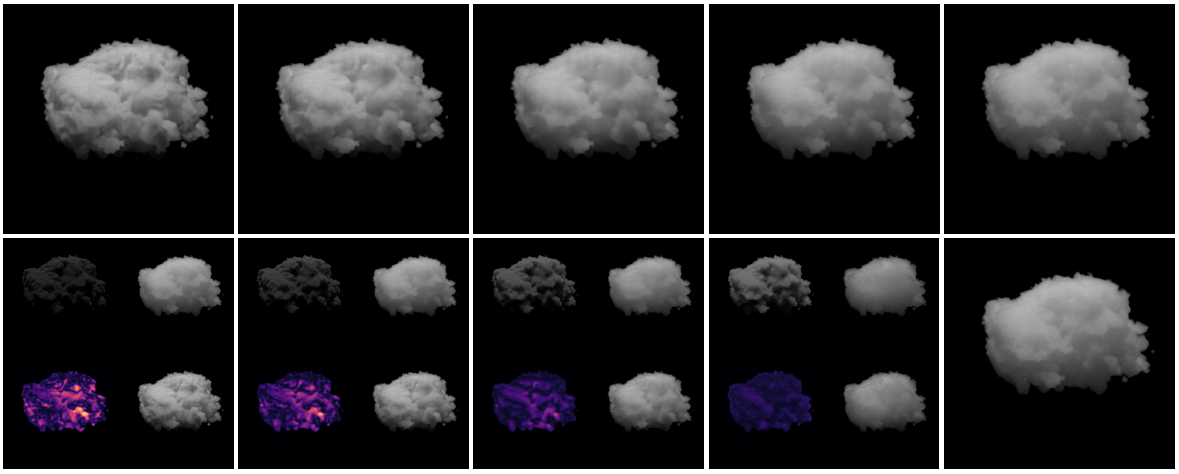
All images are tone-mapped using gamma correction. The error images show the absolute error of the high-order scattering contribution images using gamma correction, hence they do not exactly represent the bias, which is measured in linear space, but instead visualize the perceptual error of our method. We provide the full evaluation of all 27 configurations in Appendix B.

The evaluation reveals, that the bias introduced by our method consistently decreases with higher values of $k$, the number of bounces at which the prediction is made, as well as with lower values of $\rho$, the density scaling factor. This is explained by the fact that both parameters affect the total euclidean length of the paths up to the $k$th bounce, which in turn has an influence on the extents of the region inside the cloud, in which predictions for high-order scattering are made. Using a larger region generally increases the quality of the high-order scattering prediction, which can also be observed in the visual comparison. Both, increasing the number of bounces $k$ and decreasing the density scaling factor $\rho$, lead to more diffuse and blurred appearance of the high-order scattering contribution, converging towards the reference.

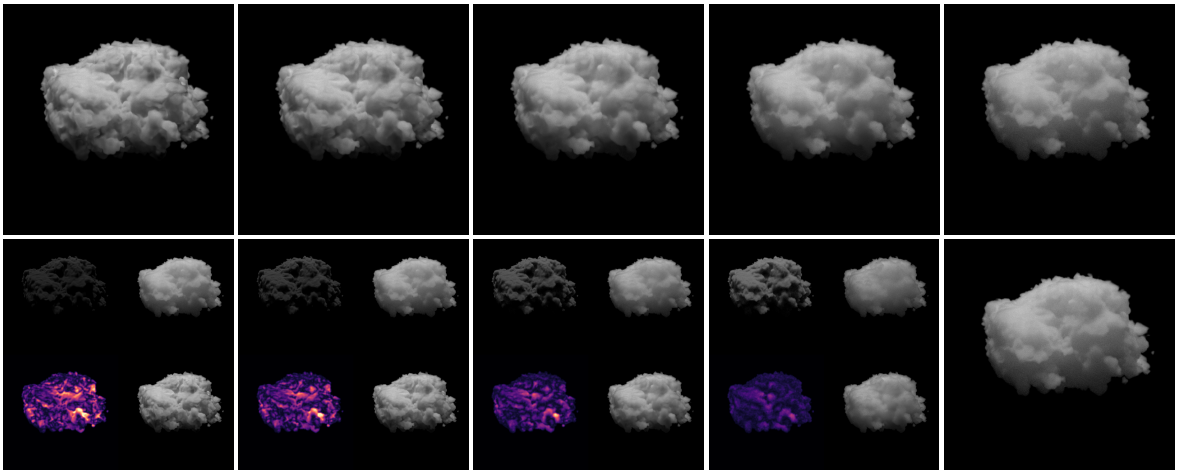
The speedup achieved by our method is generally proportional to the resulting bias, leading to a trade-off between computational efficiency and the quality of the prediction. Using higher number of bounces $k$ reduces the speedup for two reasons: First, computing the low-order scattering contribution becomes more expensive and its variance increases. Second, while the cost for predicting the high-order scattering contribution remains roughly the same, its variance increases due to the larger region over which predictions are made. Using the prediction in thicker media generally leads to a higher speedup, a consequence of the strongly increased TTUV for the reference whereas the TTUV for the prediction only increases marginally, as



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure 5.9.:*** *Visual comparison between our method and the reference for scene A in the toplit configuration.*

shown in Table 5.2.

For scene A, the errors are distributed quite evenly across the the image, which is a result of using a mostly convex shape, matching the constraints of our model. In scene B, the errors are clearly increased in regions where the cloud has a strongly concave shape, because the model fails to accurately predict light transport in these regions. The results for scene C are surprisingly good, given that the cloud has a heterogeneous interior, which the model is not designed to handle. However, it appears that the overall density of the cloud is lower than in scene A and B, hiding some of the artifacts.

## 5.6. Limitations

Due to the constrained scene configuration in our model, its applicability is naturally restricted to scenes with a similar configuration. The current model predicts illumination from a directional light source, preventing its use in scenes with different types of light sources. Support for environment maps would be a useful addition, allowing to render clouds under skylight illumination. With the current model, illumination from an environment map could be approximated by sampling directions from an environment map, using the emitted radiance of the sampled direction as the radiance for the directional light source. Supporting other types of light sources might be feasible by extending the model with additional features.

Another limitation comes from the model being agnostic to visibility, preventing it to predict shadowing from geometry as well as other participating media. Therefore, it is not possible to render more complex scenes, for example a cloud scape instead of a single cloud. Using a model based on a homogeneous medium with a convex boundary naturally limits its use to predict high-order scattering in more complex scenes with heterogeneous media. This lead to the development of our second model, introduced in the next chapter.

# 6

# Point-To-Point Transport Model

In this chapter, we introduce our *point-to-point transport model*, a model to predict light transport between two points within a heterogeneous participating medium. In contrast to the global transport model, this model can be used in a bidirectional path tracer, to connect the end vertices of two subpaths and predict the total transport across the connecting segment. The main advantage of this extended model is, that the constraints of the scene configuration are loosened, such that the model should be applicable in many more scenarios, for example handling different types of light sources, as well as heterogeneous participating media with arbitrary shapes.

## 6.1. Model

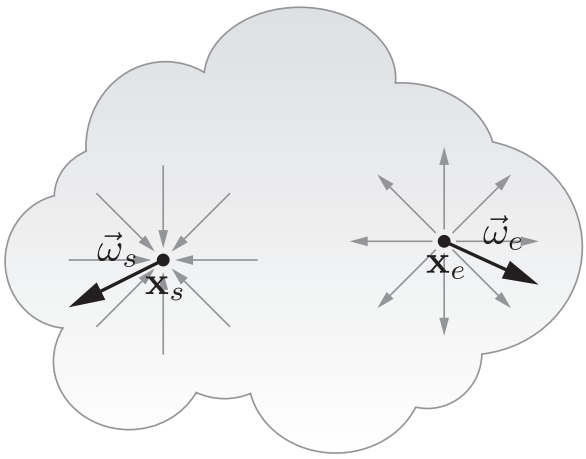



***Figure 6.1.:*** *Overview of the our point-to-point transport model. We predict the total light transport between an emitter and a sensor inside a heterogeneous participating medium.*

For the point-to-point transport model, we assume a scene that contains a heterogeneous medium with spatially varying extinction coefficient $\sigma_t$, constant scattering albedo $\alpha_s$ and a Henyey Greenstein phase function with constant eccentricity $g$. We place a point emitter with unit power at position $\mathbf{x}_e$ inside the medium, using a directional emission profile equal to the phase function, with fixed incident direction $\vec{\omega}_e$. Analogously, we place a point sensor at position $\mathbf{x}_s$ inside the medium, using a directional importance profile again equal to the phase function, but with fixed exitance direction $\vec{\omega}_s$. See Figure 6.1 for an overview of the scene configuration.

We define the total light transport $T$ between the emitter and the sensor as an integral over the domain $\Omega$ of light transport paths:

$$T(\mathbf{x}_e, \vec{\omega}_e \leftrightarrow \mathbf{x}_s, \vec{\omega}_s) = \int_{\Omega} f(\overline{\mathbf{x}})\, dV(\overline{\mathbf{x}}), \tag{6.1}$$

where $\overline{\mathbf{x}} = \mathbf{x}_0, \ldots, \mathbf{x}_k$ is a path of $k \geq 1$ segments with fixed start and end vertices $\mathbf{x}_0 \equiv \mathbf{x}_e$ and $\mathbf{x}_k \equiv \mathbf{x}_s$, $f$ is the contribution function as defined in (2.35) and the differential measure $dV(\overline{\mathbf{x}})$ corresponds to volume integration. We use an adapted scattering function $p(\mathbf{x}_i)$ for the contribution function $f$, defined as:

$$p(\mathbf{x}_i) = \begin{cases} p_{hg}(\vec{\omega}_e \leftrightarrow \vec{\omega}_{\mathbf{x}_0\mathbf{x}_1}) & \text{if } i = 0, \\ p_{hg}(\vec{\omega}_s \leftrightarrow \vec{\omega}_{\mathbf{x}_k\mathbf{x}_{k-1}}) & \text{if } i = k, \\ p_{hg}(\vec{\omega}_{\mathbf{x}_i\mathbf{x}_{i-1}} \leftrightarrow \vec{\omega}_{\mathbf{x}_i\mathbf{x}_{i+1}})\sigma_s(\mathbf{x}_i) & \text{otherwise,} \end{cases} \tag{6.2}$$

where $p_{hg}$ is the Henyey Greenstein phase function with constant eccentricity $g$ and $\sigma_s$ is the spatially varying scattering coefficient. Using the total light transport $T$ as the target function $f_t$, we define its predictor $\hat{f}_t$ as:

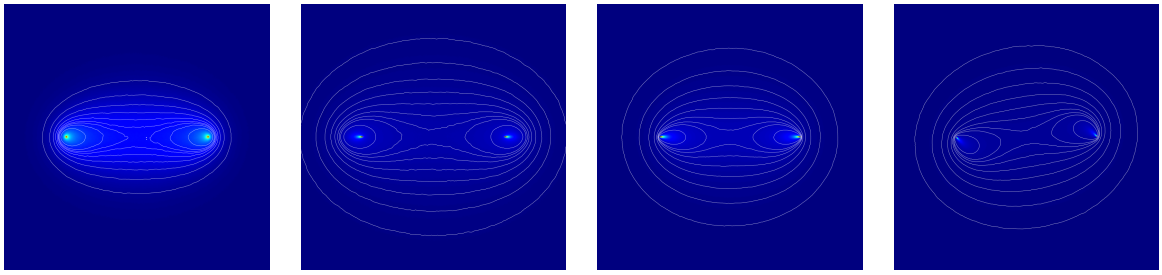
$$T(\mathbf{x}_e, \vec{\omega}_e \leftrightarrow \mathbf{x}_s, \vec{\omega}_s) = f_t \approx \hat{f}_t\left(\mathbf{x}_e, \vec{\omega}_e, \mathbf{x}_s, \vec{\omega}_s, \Sigma\right), \tag{6.3}$$

where $\Sigma$ is a discretization of the *density field* $\sigma_t(\mathbf{x})$.

### 6.1.1. Density Field Discretization

Because light transport is strongly affected by the density field, a representative discretization $\Sigma$ is essential for the accuracy of the predictor. In Figure 6.2, we visualize the fluence due to light transport between two points inside an infinite homogeneous medium, computed using a custom 2D light transport simulator. The resulting images can be interpreted as importance maps for the discretization of the density field. We found that the regions around the emitter and sensor positions are of higher importance than regions farther away, which comes at no surprise, as all contributing light paths start at the emitter and end at the sensor.

We initially planned to use a similar approach as proposed in a method for predicting fluid dynamics [LJS+15], where a summed volume table is used to quickly evaluate features based on the average response of a scalar field over many randomly positioned and sized boxes. However, due to the arbitrary orientation of the emitter and sensor with respect to the density grid,



**(a)** $g = 0$ **(b)** $g = 0.8$ **(c)** $g = 0.8$ **(d)** $g = 0.8$

***Figure 6.2.:*** *Visualization of fluence due to light transport between two points at unit distance in an infinite homogeneous medium with extinction coefficient $\sigma_t = 40$ and scattering albedo $\alpha_s = 0.99$. (a) uses an isotropic medium, (b), (c) and (d) use an anistropic medium with $g = 0.8$ and three different sets of emitter and sensor directions. The isolines represent boundaries of regions with multiples of $10\%$ of the total fluence.*

we would have to recompute a new summed volume table for each prediction, which completely defeats its purpose. We propose to use point-based lookups into filtered versions of the density field instead, which allows for arbitrary orientation of the emitter and sensor, as well as measuring the average response over larger regions.

Because generating ground truth data for the point-to-point model is prohibitively expensive, as shown later, we simplified our model by only supporting isotropic media. We define the density field discretization $\Sigma$ in a local coordinate system, with operator $F$ being a combination of translation, rotation and uniform scaling, transforming from world to local space.

We fix 6 degrees of freedom of $F$ using fixed positions of the emitter and sensor in local space, $F(\mathbf{x}_e) = \dot{\mathbf{x}}_e \equiv (0, 0, -\frac{1}{2})^T$ and $F(\mathbf{x}_s) = \dot{\mathbf{x}}_s \equiv (0, 0, \frac{1}{2})^T$. The remaining degree of freedom, the rotation around the axis through the emitter and sensor, is fixed using a random rotation. We define the density field in the local coordinate system as $\dot{\sigma_t}(\dot{\mathbf{x}}) = \|\mathbf{x}_e - \mathbf{x}_s\| \, \sigma_t(F^{-1}(\dot{\mathbf{x}}))$, where $\|\mathbf{x}_e - \mathbf{x}_s\|$ scales the densities in relation to the transformation, in order to preserve the light transport between the emitter and the sensor.

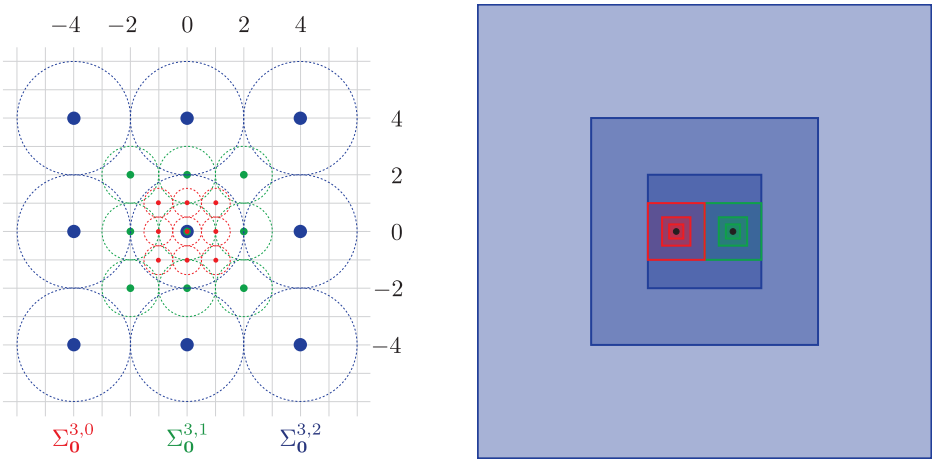
Next, we define a uniform grid of positions $\mathbf{b}^{n,l}_{i,j,k} = 2^l \left(b^n_i, b^n_j, b^n_k\right)^T$ with $b^n_i = \frac{2i}{n-1} - 1$ and $b^n_j, b^n_k$ likewise, where $n$ is the number of points along each axis, $l$ is the level specifying the extents of the grid $[-2^l, 2^l]^3$ and $i, j, k \in \{0, \ldots, n-1\}$ are the grid point indices. We define a single level of density features as:

$$\Sigma^{n,l}_{\mathbf{o}} = \left\{ \dot{\sigma_t}^{n,l}(\mathbf{o} + \mathbf{b}^{n,l}_{i,j,k}) \mid \forall i, j, k \in \{0, \ldots, n-1\} \right\}, \tag{6.4}$$

where $\mathbf{o}$ is the origin of the grid and $\dot{\sigma_t}^{n,l}$ is a gaussian filtered version of the density field $\dot{\sigma_t}$ with standard deviation $\sigma = \frac{2^l}{n-1}$, equal to half the grid point spacing along the principal axes as shown in Figure 6.3a. We define three categories of density features:

$$\Sigma^{n,l}_e \equiv \Sigma^{n,-l-1}_{\dot{\mathbf{x}}_e}, \quad \Sigma^{n,l}_s \equiv \Sigma^{n,-l-1}_{\dot{\mathbf{x}}_s}, \quad \Sigma^{n,l}_g \equiv \Sigma^{n,l}_{\mathbf{0}}, \tag{6.5}$$

where $\Sigma^{n,l}_e$ and $\Sigma^{n,l}_s$ cover the *local* neighborhood around the emitter and sensor and $\Sigma^{n,l}_g$ covers




**(a)** **(b)**

***Figure 6.3.:*** *In (a), we visualize three levels of density features. The dotted circles represent the gaussian filter standard deviation $\sigma$. In (b), we show a visualization of the total set of density features using three levels for local emitter and sensor (red and green) and global (blue) features.*

the *global* neighborhood centered around the origin $\mathbf{0} = (0, 0, 0)^T$. The total set of density features is given by the union of multiple levels in each category:

$$\Sigma_t^{n_l,n_g,l_l,l_g} = \cup \left\{ \Sigma_e^{n_l,0}, \ldots, \Sigma_e^{n_l,l_l-1}, \Sigma_s^{n_l,0}, \ldots, \Sigma_s^{n_l,l_l-1}, \Sigma_g^{n_g,0}, \ldots, \Sigma_g^{n_g,l_g-1} \right\}, \tag{6.6}$$

where the number of grid points $n_l, n_g$ and the number of levels $l_l, l_g$ are model parameters and determine the quality and scale of the discretization. See Figure 6.3b for a visualization of the total set of density features.

For our experiments, we used a density discretization with $n_l = 3, n_g = 3$ grid points, $l_l = 4$ local levels and $l_g = 5$ global levels, resulting in a total number of $351$ scalar density features. As an additional feature, we provided an estimate of light transport within an infinite homogeneous medium based on the improved diffusion equations [dI11].

We can rewrite the predictor for the model reduced to isotropic media as:

$$T(\mathbf{x}_e \leftrightarrow \mathbf{x}_s) = f_i \approx \hat{f}_i \left( \Sigma_t^{3,3,4,5}, \phi_D \right), \tag{6.7}$$

where $\Sigma_t^{3,3,4,5}$ is the density field discretization and $\phi_D$ is the light transport estimate based on the diffusion equation.

## 6.2. Training

We initially tried to use MLPs for learning our predictor (6.3), but found that we had to use large networks with millions of parameters in order to get acceptable accuracy. In consequence, the time to train a network quickly grew to days, using a single workstation based on an *Intel*

*Core i7-3930K CPU @3.2GHz*. Because TensorFlow supports training on both the CPU and GPU, we decided to run the training on a *Nvidia Geforce GTX 1080 Ti* GPU, which resulted in an average speedup of around $10\times$, reducing training times back to hours instead of days.

### 6.2.1. Deep Network

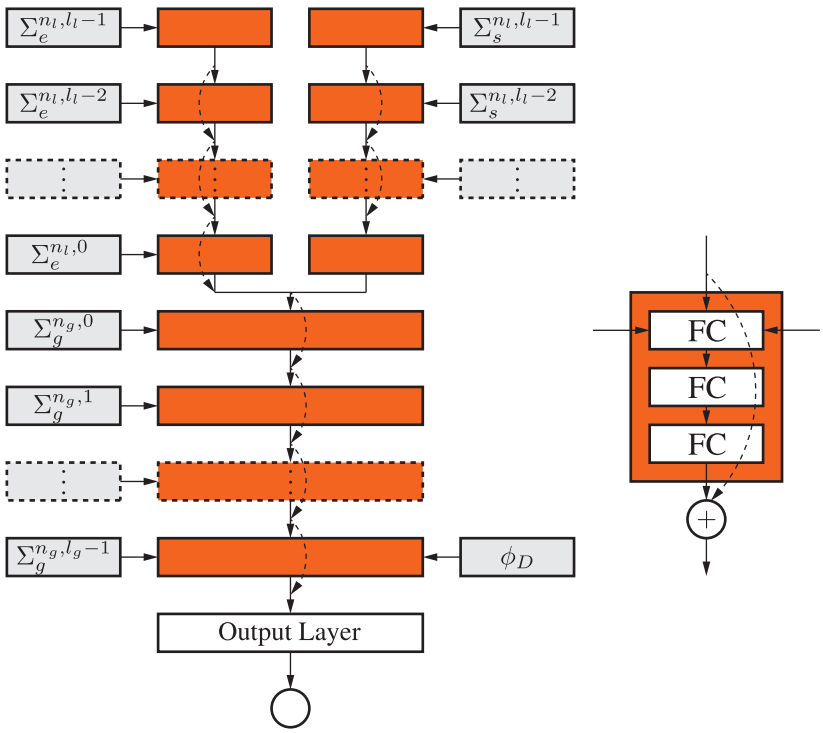



***Figure 6.4.:*** *Visualization of our deep residual network architecture using multiple blocks of fully connected layers with rectified linear units. Each level of density features is fed into a separate block, starting with the smallest scale of the local neighborhood and ending with the largest scale of the global neighborhood.*

As an alternative to the MLP network, we designed a deep network based on intuition of how the density field affects light transport. Instead of using all density features as inputs on the first layer, we feed the density features level by level into a deeper network as shown in Figure 6.4.

The network architecture uses an arrangement of blocks, each comprised of fully connected layers using rectified linear units with activation function $\phi(x) = \max(0, x)$. We found that 3 layers, each using 200 units, worked well for our purposes. Each level of density features is fed into a separate block. We begin with the local density features $\Sigma_e$ and $\Sigma_s$, starting with the highest level, the one covering the smallest region around the emitter and sensor. The output layer of each block, together with the density features of the next level act as the inputs of the next block. We repeat the same pattern for the global density features $\Sigma_g$, starting with the lowest level, again the one covering the smallest region around the emitter and sensor. We feed the additional feature based on diffusion as an input to the last block in the network. The network is terminated with a fully connected output layer.

The intuition for our deep network is based on the observation that light transport is mostly affected by the density field in the close neighborhood of the emitter and sensor, hence we start with the density features representing these regions first. We propagate learned quantities related to the total light transport through a series of blocks, which take into account a progressively larger neighborhood of the density field.

Based on the assumption of each block learning how the density field at the respective level affects light transport, we used a technique known as *deep residual networks* [HZRS15] to improve the accuracy and convergence of our network architecture. We create direct connections from the output of each block to the output of its successor block, which results in each block learning a residual based on the density features at the respective level.

### 6.2.2. Convergence

To evaluate the performance of our architecture, we trained two sets of models, furthermore denoted as *single* and *multi*. For the first set, we used a dataset using ground truth data computed on a single density grid, *cloud-1196* of the cloud set in Appendix A, consisting of 2.5 million data points. For the second set, we used a dataset based on ground truth data computed on multiple density grids, all 12 clouds in Appendix A, consisting of a total of 30 million data points. We used $75\%$ of the dataset for training and the remaining $25\%$ for testing.

We compared three different network architectures:

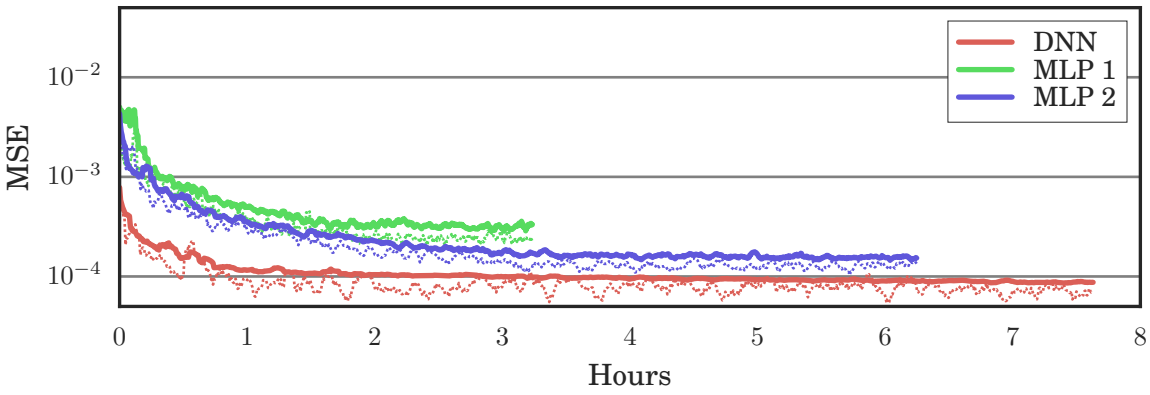
1. **DNN**, our deep residual network with $1.6$ million parameters,
2. **MLP1**, 3 hidden layers with $[2000, 1000, 500]$ units and $3.2$ million parameters,
3. **MLP2**, 4 hidden layers with $[4000, 2000, 1000, 500]$ units and $12$ million parameters.

Note that the two MLP architectures use $2\times$ and $7.5\times$ more parameters than our deep network architecture. We used the *mean squared error* loss function for all three architectures. For the first set of models we used a batch size of 500 data points and trained for 200 epochs, whereas for the second set we used a batch size of 2500 data points and 80 epochs. We used stochastic gradient descent with a momentum of $0.9$ and an empirically set schedule for the learning rate. We show the results in Figure 6.5.

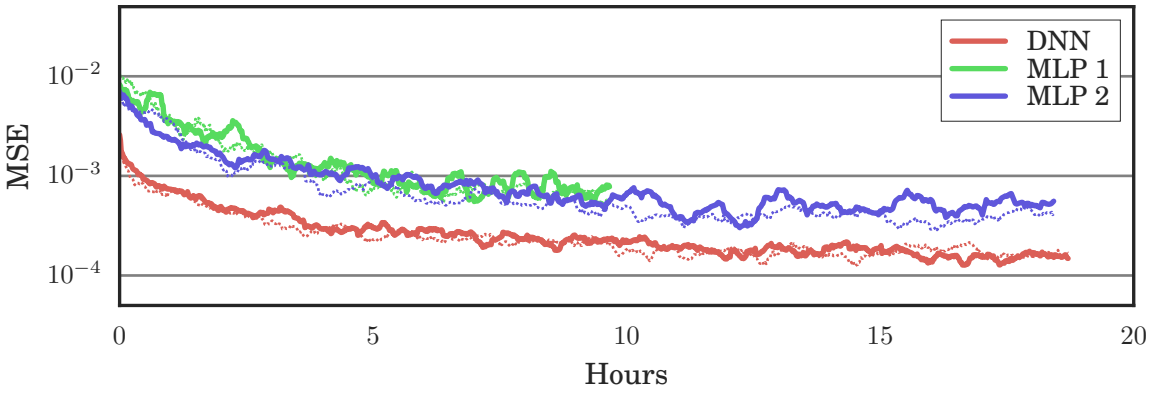
The convergence plots show that our network architecture is clearly superior to the MLP network, while using fewer parameters. Both the error and the rate of convergence is improved significantly. We found that training with the *multi* training set is much more demanding and using more than 80 epochs may be worthwhile.

## 6.3. Data Generation

We have written a custom bidirectional path tracer in C++11 to generate the datasets for training, solving the light transport integral (6.1). The data generator is configured with the following parameters:




**(a)** Convergence using the *single* training set.




**(b)** Convergence using the *multi* training set.

***Figure 6.5.:*** *Comparison of the training convergence of our deep network architecture (DNN) with two MLP networks. Thick lines represent the testing error, dotted lines the training error.*

- cloud density field $\sigma_t(\mathbf{x})$,
- density scaling factor $\rho$,
- scattering albedo $\alpha_s$,
- phase function eccentricity $g$,
- number of query bounces $k$,
- maximum relative error and confidence level.

Based on the configured settings, the data generator performs the following steps to generate each data point:

1. Sample a light path $\overline{\mathbf{x}}$ with $k$ segments from a random position and direction outside the medium and set the emitter position $\mathbf{x}_e = \mathbf{x}_k$ and emitter direction $\vec{\omega}_e = \vec{\omega}_{\mathbf{x}_k \mathbf{x}_{k-1}}$.
2. Sample a light path $\overline{\mathbf{x}}$ with $k$ segments from a random position and direction outside the medium and set the sensor position $\mathbf{x}_s = \mathbf{x}_k$ and sensor direction $\vec{\omega}_s = \vec{\omega}_{\mathbf{x}_k \mathbf{x}_{k-1}}$.

3. Compute light transport $T(\mathbf{x}_e, \vec{\omega}_e \leftrightarrow \mathbf{x}_s, \vec{\omega}_s)$ using bidirectional path tracing to requested accuracy by using confidence interval estimation to control the number of samples.
4. Compute feature vector $[\Sigma_t^{n_l,n_g,l_l,l_g}, \phi_D]$ and target value $f_t$ and store the data point.

### 6.3.1. Optimizations

We have implemented various optimizations to accelerate the computation of the light transport integral. First, instead of using ray-marching to sample free flight distances, we used Woodcock tracking, which proved to be much more efficient, even without the transmittance terms represented in the MIS weights. Second, we used Russian roulette to discard many of the connections between the subpaths. We tried two Russian roulette schemes: RR1 uses the product of the phase function terms evaluated at the connected vertices, RR2 extends RR1 by multiplying with the geometry term between the connected vertices.

We conducted a performance evaluation on the cluster, computing data points for different medium properties and comparing various strategies to compute the light transport. We used a maximum relative error of $10\%$ and a confidence level of $95\%$. The results are reported in Table 6.1.

Using Woodcock tracking instead of ray-marching results in a speedup between $5 - 15\times$. Using Russian roulette to prune path connections results in another $2\times$ speedup, where the RR2 scheme performed marginally better overall.

Even with these optimizations, generating large amounts of ground truth data was prohibitively expensive. In order to compute a dataset for our experiments, we decided to use relatively thin media with a density scaling factor of $\rho = 25.0$ and use an isotropic phase function. The latter allowed to reuse each computed data point multiple times in the final dataset, as we can evaluate the density features for different rotations around the axis between the emitter and sensor points. We computed 50000 data point for each of the 12 cloud density grids shown in Appendix A. Each data point was reused 50 times, using random rotations, resulting in a total of 2.5 million data points per cloud.

## 6.4. Rendering

We integrated the point-to-point transport model into our custom renderer, allowing us to evaluate its quality and performance. The renderer uses standard bidirectional path tracing for the low-order contribution and uses our model to predict the transport between the end vertices of both the emitter and sensor subpaths as shown in Figure 6.6. Using $k$ segments on both subpaths, we compute the low-order scattering contribution with $1, \ldots, 2k$ bounces using bidirectional path tracing and the high-order scattering contribution using our prediction model.

The singularity due to the geometry terms in the transport predictor (6.3) leads to a significant amount of variance. Because it is not possible to derive a PDF for the transport predictor, we were unable to use multiple importance sampling for reducing variance on paths connected by the prediction. Therefore, we introduce a distance threshold $r$, which is used to avoid predic-

| | | PT | BDPT | | | |
|---|---|---|---|---|---|---|
| Medium | | Woodcock | Ray-Marching | Woodcock | | |
| $\alpha_s$ | $g$ | | All | All | RR1 | RR2 |
| $\rho = 10.0$ | | | | | | |
| 0.99 | 0.00 | 8.43 | 22.78 | 1.65 | 0.60 | **0.57** |
| 0.99 | 0.85 | - | 53.83 | 3.64 | 3.93 | **1.82** |
| 1.00 | 0.00 | 3.99 | 24.41 | 1.73 | **0.52** | 0.53 |
| 1.00 | 0.85 | - | 50.44 | 4.30 | 3.81 | **1.84** |
| $\rho = 20.0$ | | | | | | |
| 0.99 | 0.00 | 164.95 | 230.84 | 40.81 | 8.48 | **6.77** |
| 0.99 | 0.85 | - | 144.56 | 18.13 | 10.94 | **7.22** |
| 1.00 | 0.00 | 47.11 | 279.45 | 56.72 | **9.74** | 10.51 |
| 1.00 | 0.85 | - | 121.44 | 18.45 | 11.30 | **5.13** |
| $\rho = 40.0$ | | | | | | |
| 0.99 | 0.00 | 207.72 | - | - | 142.10 | **133.49** |
| 0.99 | 0.85 | - | - | 161.08 | 46.11 | **33.03** |
| 1.00 | 0.00 | **160.66** | - | - | 312.34 | 373.34 |
| 1.00 | 0.85 | - | 392.06 | 144.36 | 44.13 | **29.85** |

***Table 6.1.:*** *Time in seconds to generate a single data point on a single thread of a Xeon E5-2697 v2, using a cloud density grid with a diameter of around* $1$*, different density scaling factors* $\rho$*, scattering albedos* $\alpha_s$*, eccentricities* $g$*, a maximum relative error of* $10\%$ *and confidence level of* $95\%$*. We compare two algorithms, path tracing (PT) and bidirectional path tracing (BDPT), two distance sampling strategies (Woodcock, Ray-Marching) and 3 different connection schemes: connecting all prefixes (All), using Russian roulette based on the phase function terms (RR1) and using Russian roulette based on the phase function terms and the geometry term (RR2). We highlight the best result for each scene configuration.*

tions altogether for query locations $\mathbf{x}_s$ and $\mathbf{x}_e$ if $\|\mathbf{x}_s - \mathbf{x}_e\| < r$. In these cases, we instead estimate transport using standard bidirectional path tracing, extend one subpath by another vertex and repeat the process.

## 6.5. Evaluation

To evaluate our point-to-point model, we rendered a scene containing the *cloud-1196* density grid, matching the cloud used to generate the dataset for our *single* model. We used two lighting

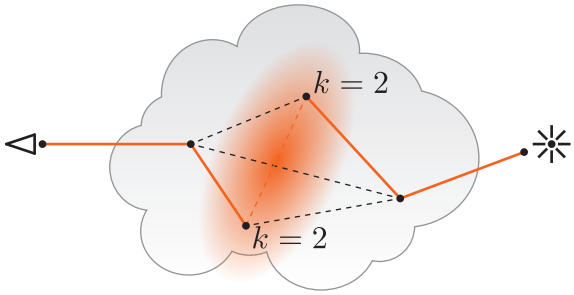



***Figure 6.6.:*** *Visualization of bidirectional path tracing and predicting high-order scattering using our model. The first $2k$ bounces are computed using bidirectional path tracing, the remaining high-order scattering contribution is predicted by our point-to-point transport model.*

scenarios:

- **Backlit**, cloud is illuminated from the back side, opposite to the view direction,
- **Frontlit**, cloud is illuminated from the front side, in line with the view direction.

For the medium properties, we used a density scaling factor $\rho = 25$, scattering albedo $\alpha_s = 0.99$ and an isotropic phase function. We used a distance threshold $r = 0.08$, corresponding to approximately 2 mean free paths. For different initial number of bounces $k \in \{1, 2, 3, 4\}$, we rendered the following images:

- **High-order scattering reference**, computed using bidirectional path tracing with 16k samples per pixel,
- **High-order scattering prediction**, computed using our prediction model with 4k samples per pixel.

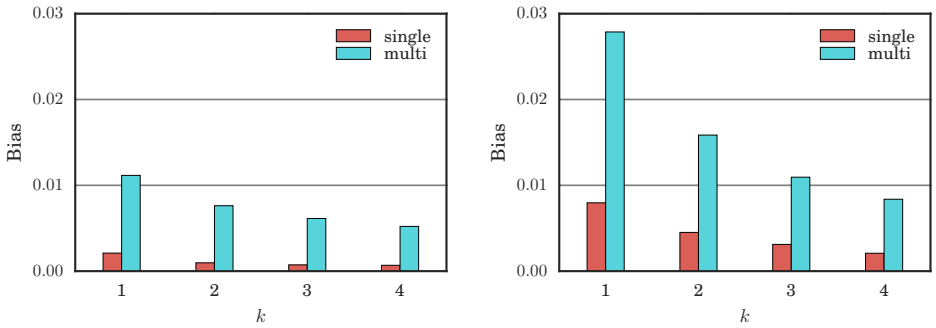
Each image using the prediction model was rendered twice, to get results for both the *single* and *multi* models. We used a simplified rendering algorithm to generate the images, because there were inconsistencies within our bidirectional path tracer. Therefore, we only computed the high-order scattering contribution in order to evaluate the accuracy of our model. We use two subpaths with $k$ segments and compute the total transport either by our model or by using the same bidirectional path tracer we used to generate the ground truth data. We completely reject light transport between paths if their end vertices are closer than the distance threshold $r$. This simplified rendering setup unfortunately results in a significant overhead when rendering the reference images, because the first $k$ segments of both subpaths are not weighted using MIS.

We report the *bias* introduced by our model in Table 6.2, computed as the mean of absolute error between the pixel values of the reference and prediction images in linear space. We show a comparison of the bias between our two models in Figure 6.7. In Figure 6.8 and Figure 6.9 we visually compare the reference to the prediction obtained by the two models in both the backlit and frontlit configurations. As before, all images are tone-mapped using gamma correction and the error images show the absolute error between the reference and the prediction images, visualizing the perceptual error.

We observe that the bias increases significantly using the model trained on the larger dataset. This can be explained either by the fact that the training has not fully converged after 80 epochs, as shown earlier, or that the model is not capable of handling the larger training set and would require more parameters or a finer grained density field discretization. We can also observe that the bias decreases with higher number of initial bounces $k$, which has two main causes: First, the magnitude of the estimated light transport decreases with higher values of $k$, leading to lower absolute error and second, due to the increasing length of the initial subpaths, a larger fraction of the complete light path is estimated with unbiased Monte Carlo integration. The reason for the generally higher bias in the frontlit configuration is again due to the magnitude of the light transport.

| | Backlit | | | | Frontlit | | | |
|---|---|---|---|---|---|---|---|---|
| Training Set | $k=1$ | $k=2$ | $k=3$ | $k=4$ | $k=1$ | $k=2$ | $k=3$ | $k=4$ |
| single | 0.0021 | 0.0010 | 0.0007 | 0.0007 | 0.0080 | 0.0045 | 0.0031 | 0.0021 |
| multi | 0.0112 | 0.0076 | 0.0061 | 0.0052 | 0.0279 | 0.0159 | 0.0109 | 0.0084 |

***Table 6.2.:*** *Bias introduced by the single and multi models using different number of initial bounces $k$ in both the backlit and frontlit configuration.*




**(a)** Backlit **(b)** Frontlit

***Figure 6.7.:*** *Comparison of bias between the single and multi models using different number of initial bounces $k$ in both the backlit and frontlit configuration.*

We report the TTUV, computed as the mean of the pixel variance multiplied by the render time in Table 6.3. Even though using the prediction is generally around 10 times faster than using bidirectional path tracing to compute the same high-order scattering contribution, we found that evaluating our model for individual light paths is generally too expensive to get reasonable performance. This is especially true with an isotropic and rather thin participating media used in our test scene. In this setting, simple path tracing actually performs much better, while computing the full light transport, as shown in the comparison. Using our method is up to 100 times slower.

To check if the model trained on a larger dataset is more general, we rendered all of the 12 clouds from our cloud set in Appendix A using $k = 4$. We only rendered the backlit configuration and

| | Training Set | PT | $k=1$ | | $k=2$ | | $k=3$ | | $k=4$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| Backlit | | 10.3 | | | | | | | | |
| | single | | 706.0 | 7.9× | 702.1 | 8.9× | 734.0 | 10.4× | 630.4 | 11.3× |
| | multi | | 815.4 | 6.8× | 750.5 | 8.3× | 785.6 | 9.7× | 676.0 | 10.5× |
| Frontlit | | 78.3 | | | | | | | | |
| | single | | 7099.8 | 6.0× | 5803.3 | 7.4× | 5551.7 | 8.3× | 4238.5 | 9.3× |
| | multi | | 8277.5 | 5.1× | 6486.3 | 6.6× | 6024.8 | 7.6× | 4583.5 | 8.6× |

***Table 6.3.:*** *TTUV and speedup using the single and multi models with different number of initial bounces $k$ in both the backlit and frontlit configuration. The speedup is with respect to computing the high-order scattering reference using bidirectional path tracing. For comparison, we also include the TTUV for computing the full transport using path tracing (PT).*

computed the bias as before. We compare the bias introduced by the two models in Figure 6.10 and show a visual comparison on a subset of the clouds in Figure 6.11.

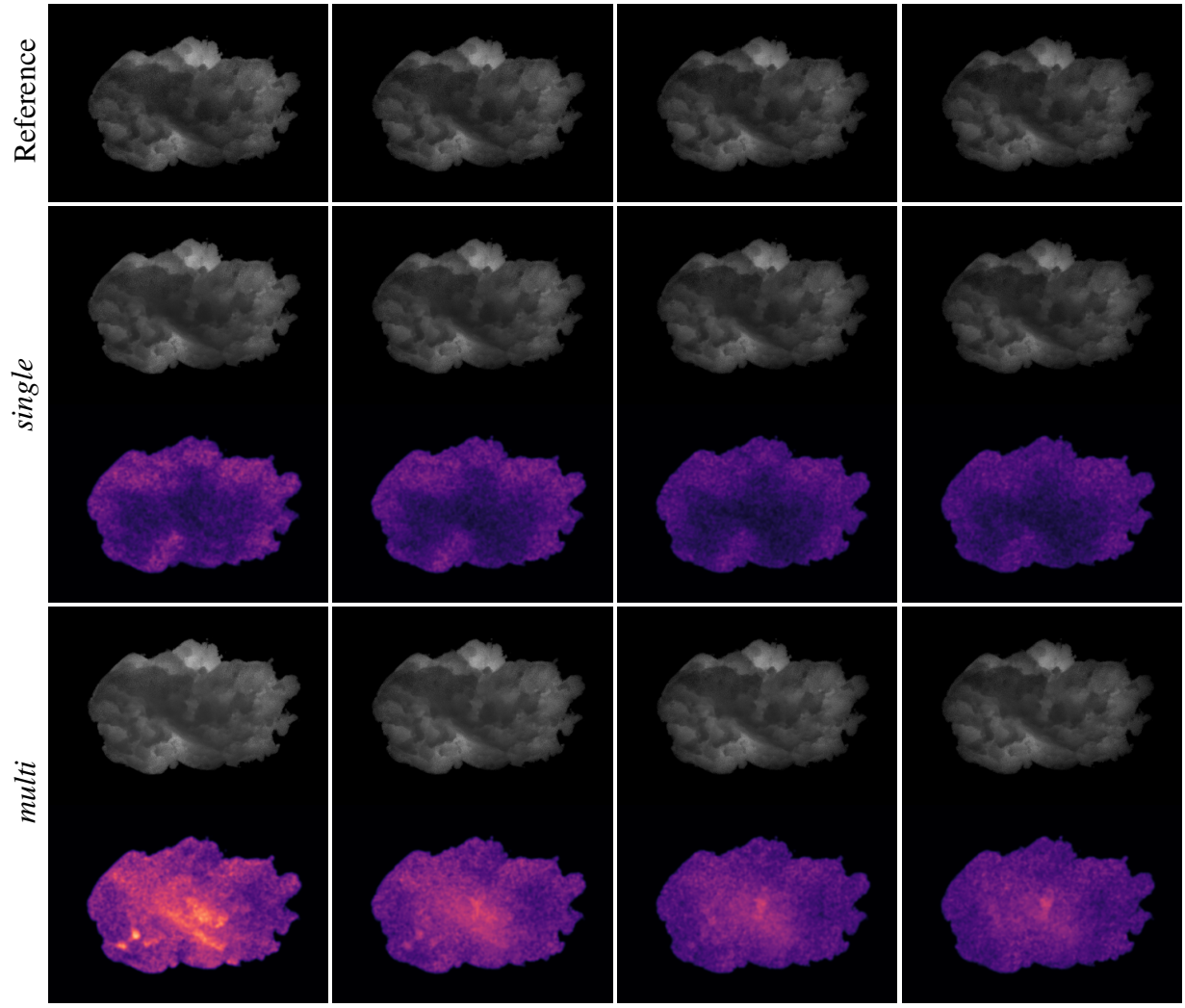
In general, the bias when using the *multi* model is significantly lower than with the *single* model, which is to be expected, as the larger training set actually contains ground truth data for these additional scenes. The only exceptions are *cloud-1191* and *cloud-1196*, which can be explained by the fact that the *single* single model is trained on ground truth data of the latter, and the *cloud-1191* is quite similar in shape and form. It is surprising however, that the bias in *cloud-1191* is significantly higher than for all other scenes. Experimenting with an increased number of density features or a network with more parameters might improve this, but it is difficult to argue where this rise in bias is coming from exactly.

## 6.6. Limitations

In theory, our point-to-point transport model has less limitations than the previously introduced global transport model. The model is able to predict light transport inside heterogeneous media and supports different types of light sources by using the bidirectional path tracing framework.

The model is currently limited from using a fixed phase function and a fixed scattering albedo, which in theory might be alleviated by using a similar discretization as for the density field. However, the additional dimensions would make the problem significantly harder, if not infeasible, both in terms of generating ground truth data as well as in terms of the model complexity.

The main issues we faced were related to generating ground truth data and using the model during rendering. Computing light transport between two points inside a participating medium is a difficult problem, leading to infinite variance when using unidirectional path tracing [Kal63]. Finite variance can be achieved with bidirectional path tracing using MIS, but generating ground truth data is still prohibitively expensive. Even with our optimizations we were not able to generate the amount of data we hoped for. Techniques such as joint importance sampling




**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 3$ **(d)** $k = 4$

***Figure 6.8.:*** *Visual comparison between images generated with our prediction models and the reference in the backlit configuration.*

[GKH+13] could further help the performance, but generating ground truth data would still remain a difficult problem.

During rendering, we faced the problem of not having a PDF with regards to the point-to-point transport function. In consequence, we cannot use MIS to weight paths using the prediction, leading to increased variance. Blindly connecting random subpaths again leads to infinite variance due to the geometry terms, which we have tried to avoid by using a distance threshold during rendering.

We also found that evaluating our model with 1.6 million parameters is too expensive to compute light transport along single light paths. Without the possibility to importance sample the point-to-point transport function, using bidirectional path tracing without the prediction is actually far more efficient in the scenes we tested. However, we have not been able to expand our model to handle strongly forward scattering phase functions and thicker media, which would

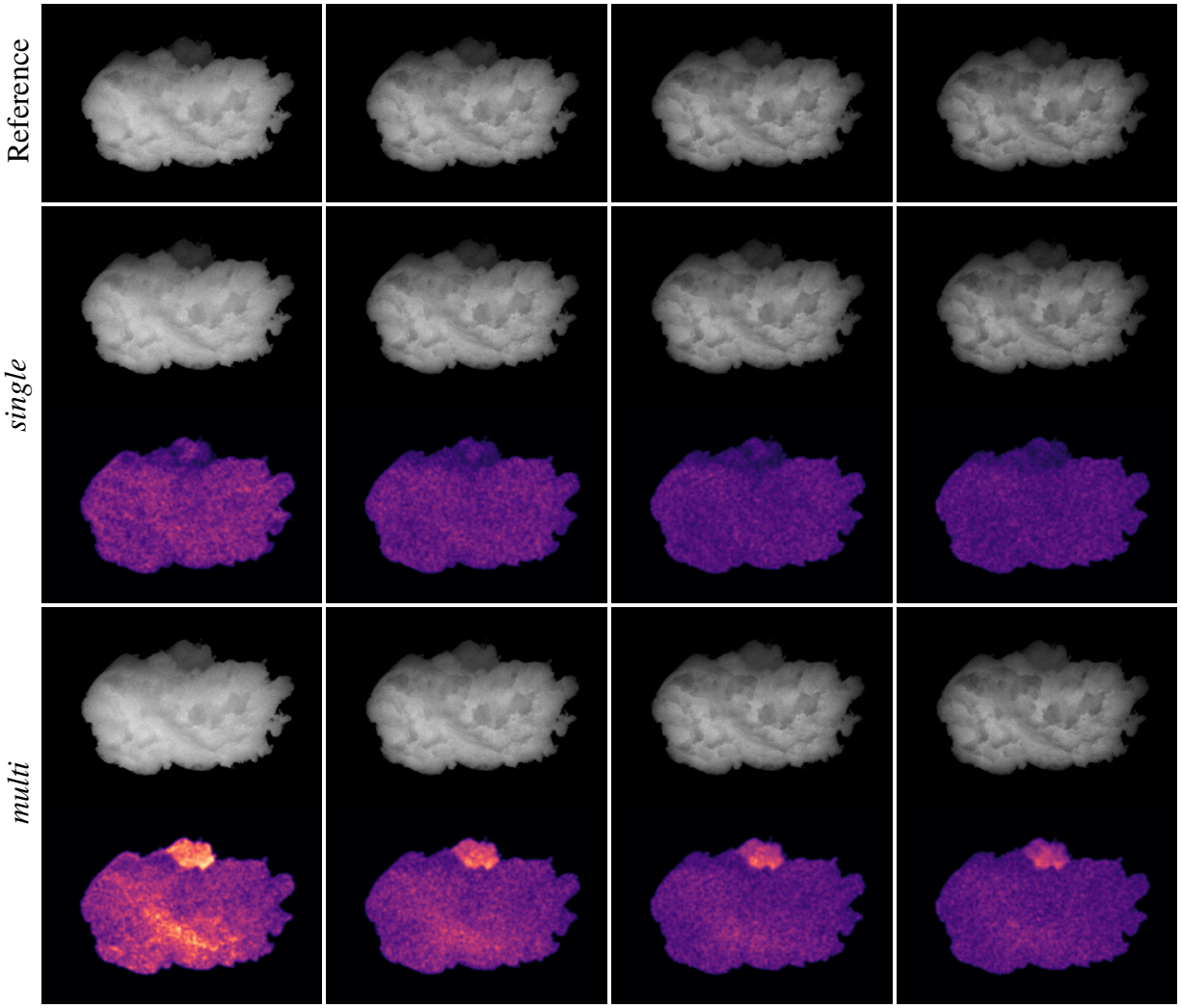



***Figure 6.9.:*** *Visual comparison between images generated with our prediction models and the reference in the frontlit configuration.*

strongly affect the performance of bidirectional path tracing and make using predictions more worthwhile.

Despite all the issues we faced, we have shown that our model is capable of predicting light transport within a heterogeneous participating medium. This confirms that our density discretization is able to capture the relevant information needed for predicting light transport and that our network architecture is able to learn the transport function.

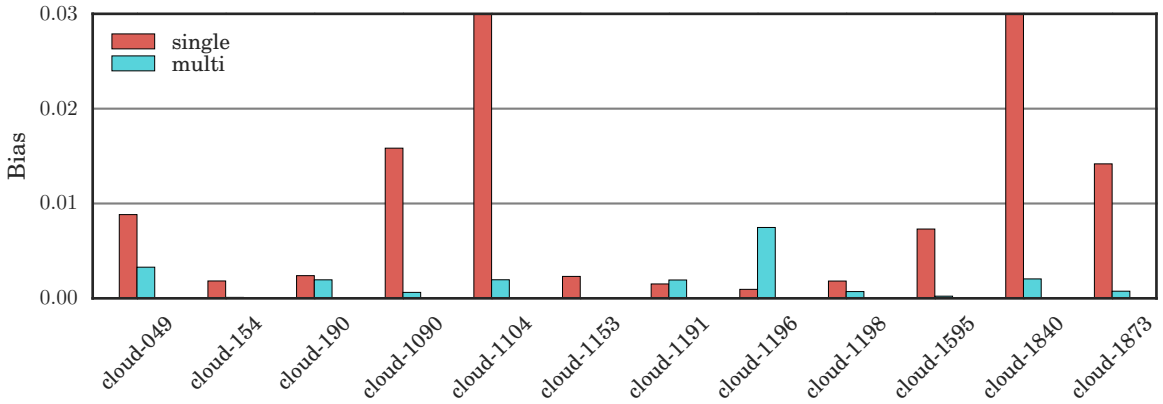
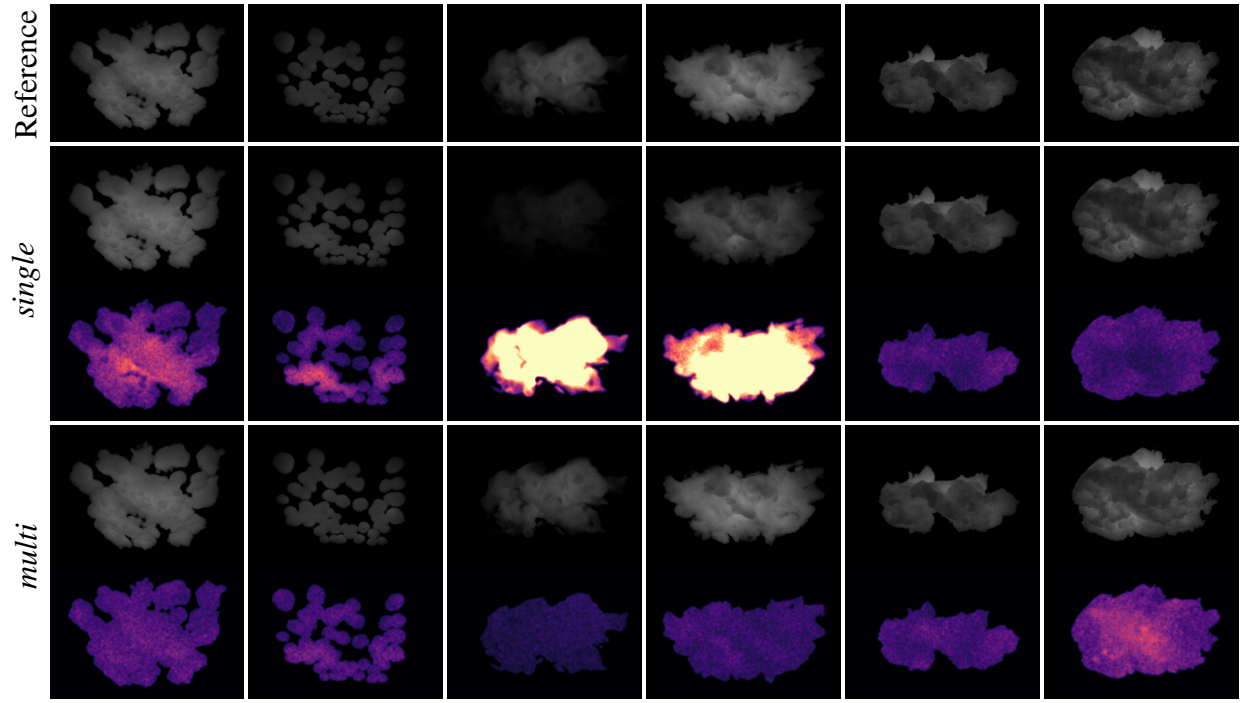



***Figure 6.10.:*** *Comparison of bias between the single and multi models in scenes containing all 12 clouds from our cloud set rendered in the backlit configuration.*




**(a)** cloud-049 **(b)** cloud-190 **(c)** cloud-1090 **(d)** cloud-1104 **(e)** cloud-1191 **(f)** cloud-1196

***Figure 6.11.:*** *Visual comparison between images generated with our prediction models and the reference in the backlit configuration.*

# 7

# Conclusion and Future Work

In this thesis, we have investigated the problem of predicting high-order scattering using neural networks. We have reviewed the theoretical background for rendering of participating media and machine learning with focus on regression problems using neural networks. We have reported on related work in the fields of rendering and machine learning applied to computer graphics.

Our main contribution is the development of two novel models for predicting light transport in participating media. Our *global transport model* predicts the total light transport from a directional light source to a point within a simple homogeneous medium. With our *point-to-point transport model*, we predict light transport between two points within a heterogeneous medium. We have implemented software to create ground truth datasets for learning, as well as using the models during rendering for predicting high-order scattering. We have conducted experiments to examine both the performance and accuracy of our methods.

We have shown, that the global transport model is able to predict high-order scattering reasonably well, given that the constraints of the model are respected. It was surprising to see that the model even produces usable results within a cloud with a heterogeneous interior, as long as the number of bounces for computing low-order scattering is high enough. This shows that the features based on measuring transmittance partly generalize from homogeneous to heterogeneous media, which we did not anticipate in the beginning.

In our experiments, using our model resulted in a speedup with respect to TTUV of around a factor 5 to 10, showing the potential for learning high-order scattering. Generating ground truth data for training is straight forward and fast enough to generate large datasets.

The main challenges during the development of our first model have been the design of the features, as well as their transformations, such that the neural network was able to learn the predictor. The resulting network was quite small, using only 200 units on the first and 100 units on the second hidden layer. We have never observed over-fitting during the training of our

predictor, which hints that the size of the training set and the effort for computing ground truth data might be reduced.

In future work, the global transport model could be extended with support for additional types of light sources. Environment maps in particular would be an interesting avenue, because they allowed for more natural lighting. Furthermore, adapting the model to a distant disk light should be straight forward and would allow to simulate light transport from the sun.

With the point-to-point transport model, our goal was to remove the main limitations of our first model, namely the constraints on the fixed type of light source and the convex homogeneous medium. In theory, the model is able to predict light transport in scenes with arbitrary light sources and more complex participating media.

Due to the cost of evaluating our model and difficulties of applying it in a bidirectional path tracer, we were unable to reduce the time for rendering high-order scattering. It would be interesting to investigate possibilities for deriving a probability density function of the point-to-point transport function and develop importance sampling techniques, as this would allow for much more efficient rendering.

Light transport within participating media is strongly affected by the density field, which we were able to capture with our discretization scheme and the deep network architecture used in our second model. It would be interesting to combine our density features with the global transport model, because this allowed predicting high-order scattering in much more complex scenes.

Both of our models predict total light transport and therefore rely on features that are representative for the whole scene. During our investigations, we have tried to find methods to predict transport in a hierarchical way, possibly allowing to reuse the same model on multiple scales. This would significantly reduce the complexity of the scene representation and help to generalize the model. Unfortunately we were unable to derive a hierarchical model, but we believe this is an interesting topic for future research.

Rendering in combination with machine learning techniques, such as deep neural networks, is still a novel field, with many possibilities for future research. We have conducted initial investigation of applying machine learning techniques to the problem of predicting high-order scattering and have shown that there is potential for future research. We are eager to see how this field will evolve in the coming years.

# A

# Cloud Set

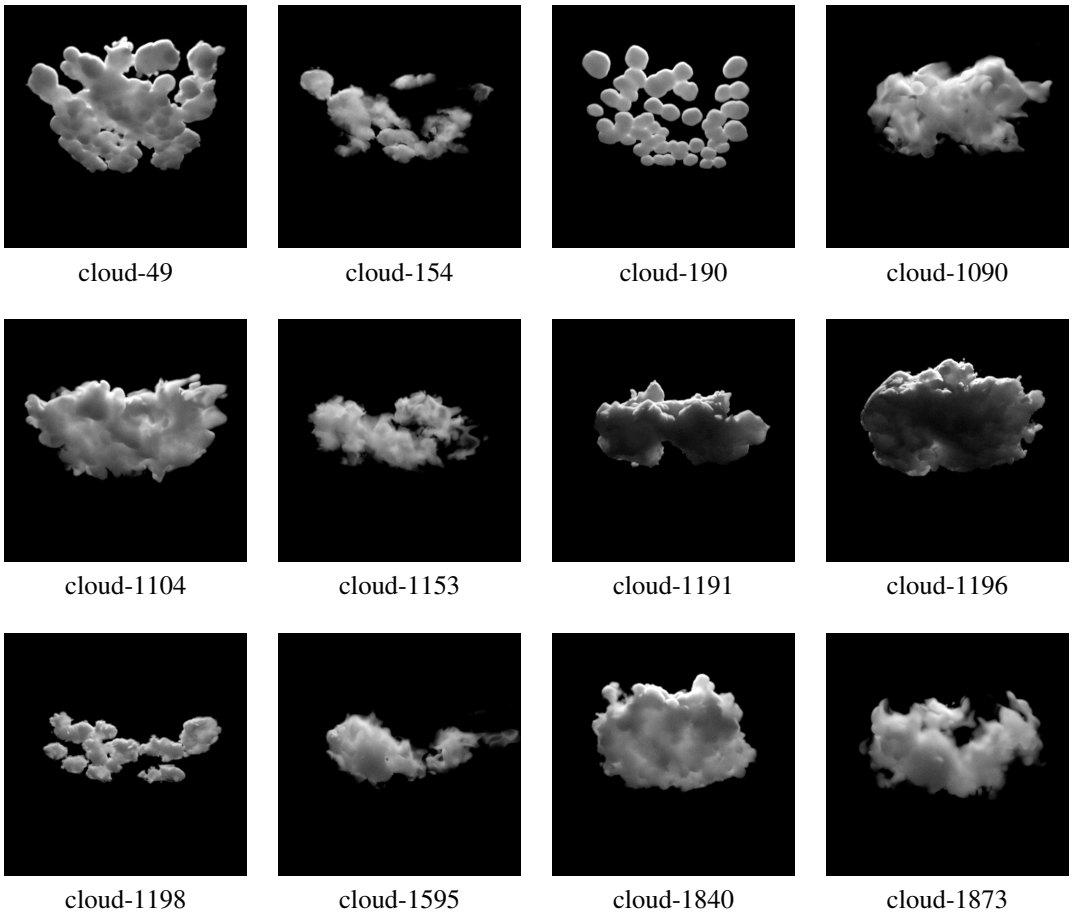



***Figure A.1.:*** *Set of cloud density grids generated with our custom cloud modeler utility.*

# B

# Global Transport Model Evaluation

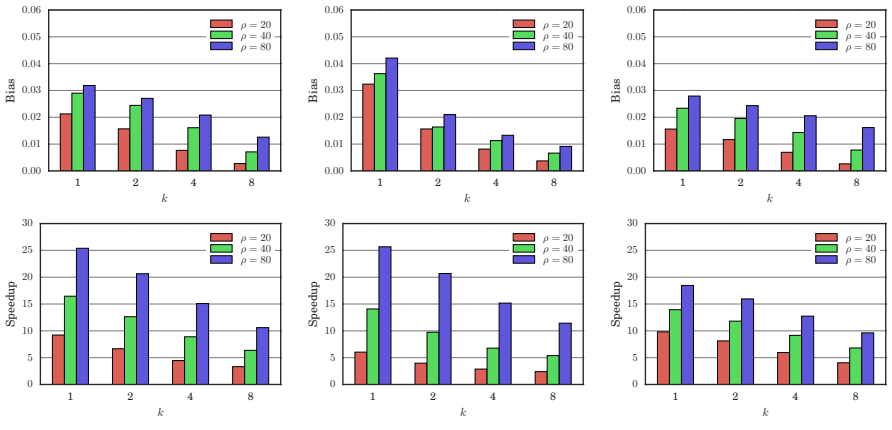
# B.1. Scene A

| | $k=1$ | $k=2$ | $k=4$ | $k=8$ |
|---|---|---|---|---|
| Toplit | | | | |
| $\rho=20$ | 0.0213 | 0.0157 | 0.0077 | 0.0027 |
| $\rho=40$ | 0.0290 | 0.0244 | 0.0161 | 0.0071 |
| $\rho=80$ | 0.0318 | 0.0271 | 0.0208 | 0.0126 |
| Backlit | | | | |
| $\rho=20$ | 0.0324 | 0.0156 | 0.0081 | 0.0037 |
| $\rho=40$ | 0.0363 | 0.0164 | 0.0113 | 0.0066 |
| $\rho=80$ | 0.0421 | 0.0210 | 0.0133 | 0.0091 |
| Frontlit | | | | |
| $\rho=20$ | 0.0156 | 0.0117 | 0.0069 | 0.0026 |
| $\rho=40$ | 0.0234 | 0.0195 | 0.0143 | 0.0078 |
| $\rho=80$ | 0.0279 | 0.0243 | 0.0206 | 0.0162 |

***Table B.1.:*** *Bias of our method compared to the reference.*

| | Reference | $k=1$ | | $k=2$ | | $k=4$ | | $k=8$ | |
|---|---|---|---|---|---|---|---|---|---|
| Toplit | | | | | | | | | |
| $\rho=20$ | 390.0 | 42.3 | 9.2× | 58.5 | 6.7× | 87.4 | 4.5× | 117.4 | 3.3× |
| $\rho=40$ | 842.8 | 51.2 | 16.5× | 66.7 | 12.6× | 94.8 | 8.9× | 132.4 | 6.4× |
| $\rho=80$ | 1716.9 | 67.6 | 25.4× | 83.2 | 20.6× | 113.8 | 15.1× | 162.0 | 10.6× |
| Backlit | | | | | | | | | |
| $\rho=20$ | 513.4 | 85.1 | 6.0× | 129.2 | 4.0× | 178.6 | 2.9× | 214.7 | 2.4× |
| $\rho=40$ | 628.7 | 44.7 | 14.1× | 64.6 | 9.7× | 92.7 | 6.8× | 116.7 | 5.4× |
| $\rho=80$ | 722.8 | 28.2 | 25.7× | 34.9 | 20.7× | 47.6 | 15.2× | 63.2 | 11.4× |
| Frontlit | | | | | | | | | |
| $\rho=20$ | 381.7 | 38.9 | 9.8× | 47.0 | 8.1× | 64.1 | 6.0× | 94.2 | 4.1× |
| $\rho=40$ | 1189.7 | 85.4 | 13.9× | 100.8 | 11.8× | 129.9 | 9.2× | 174.8 | 6.8× |
| $\rho=80$ | 3049.2 | 165.2 | 18.5× | 191.1 | 16.0× | 239.3 | 12.7× | 317.0 | 9.6× |

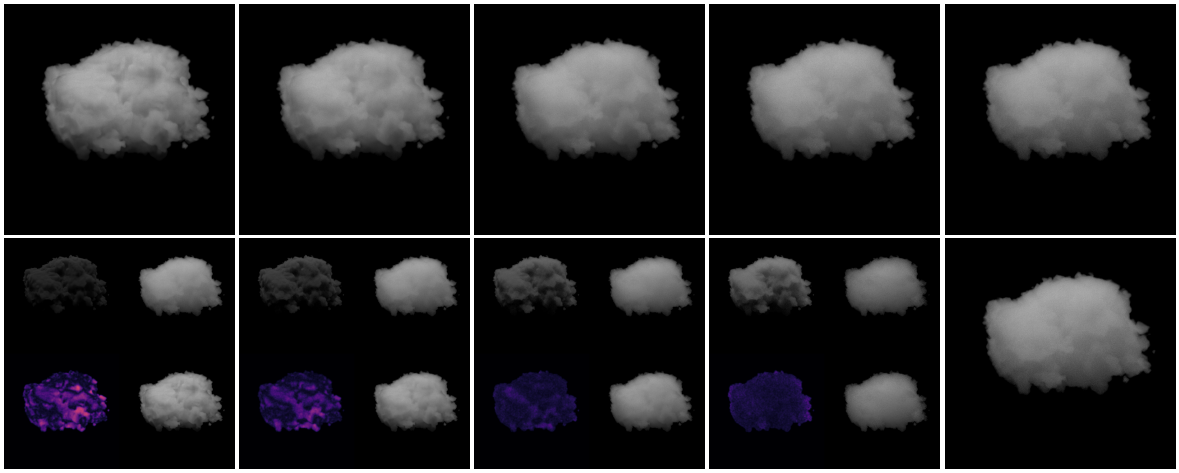
***Table B.2.:*** *Speedup with respect to TTUV of our method compared to the reference.*



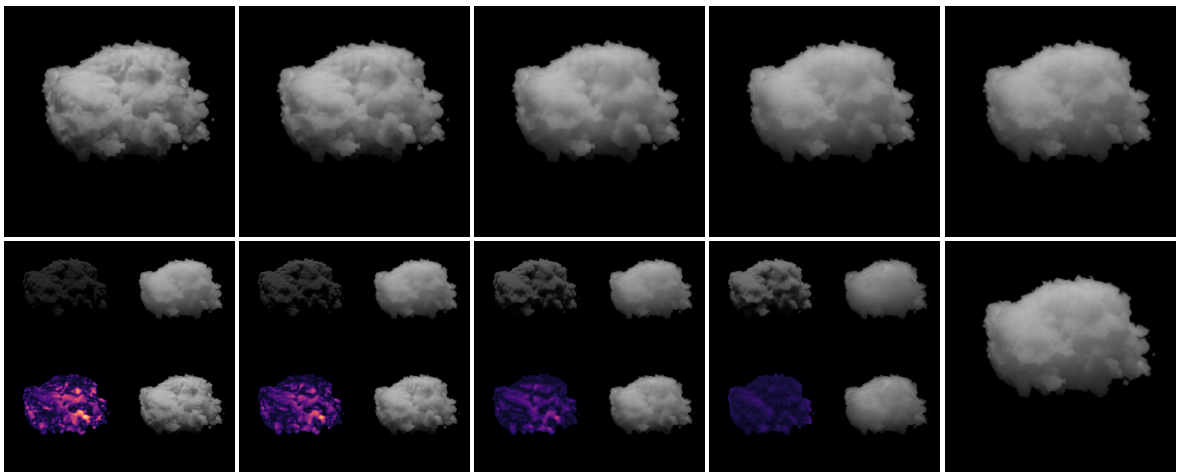

**(a)** Toplit **(b)** Backlit **(c)** Frontlit

***Figure B.1.:*** *Comparison of bias (top row) and speedup with respect to TTUV (bottom row).*

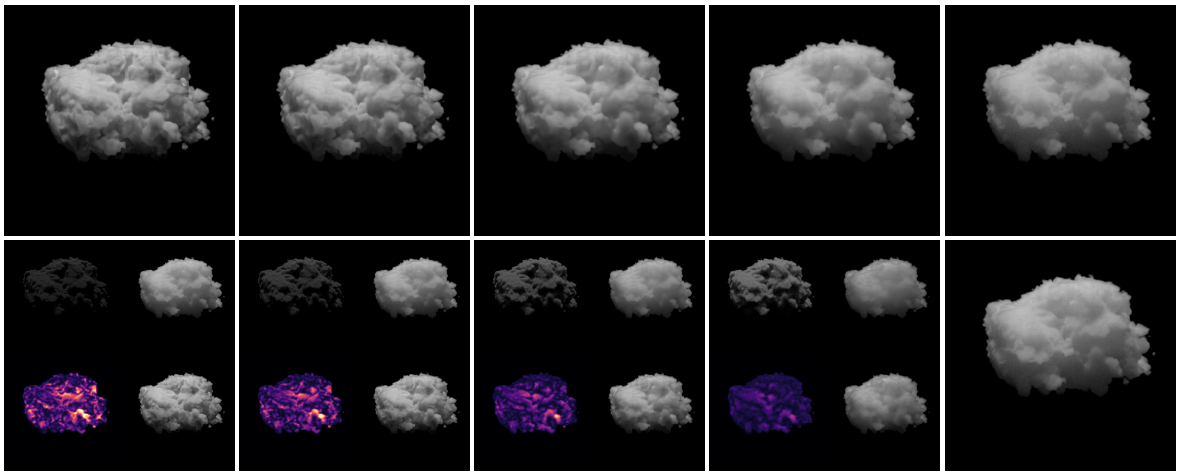
## Scene A - Toplit



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



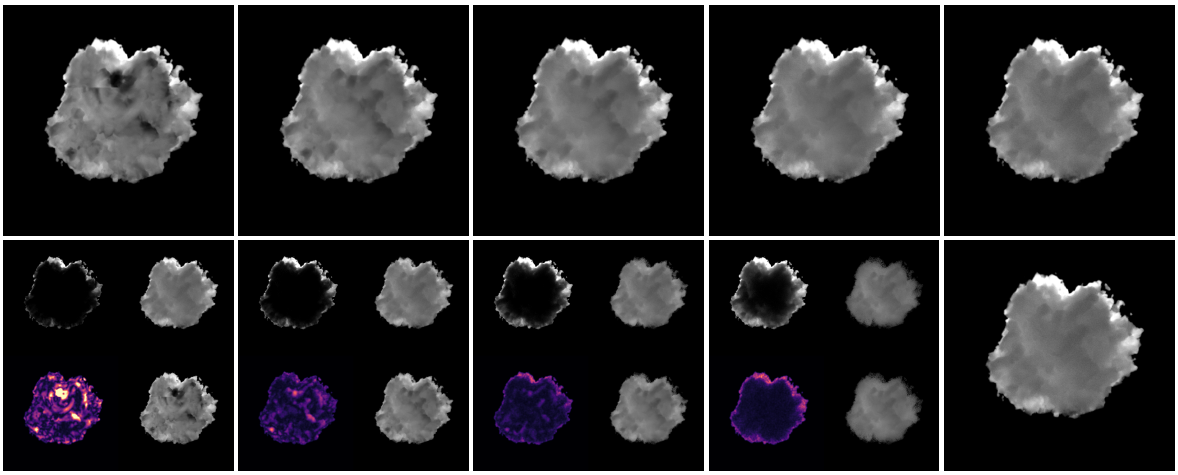
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



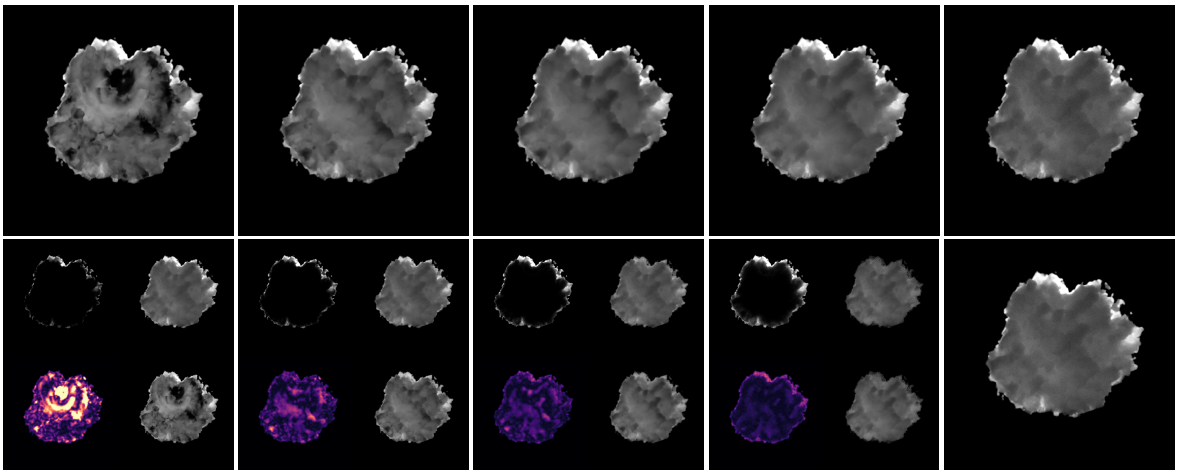
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure B.2.:*** *Visual comparison between our method and the reference.*

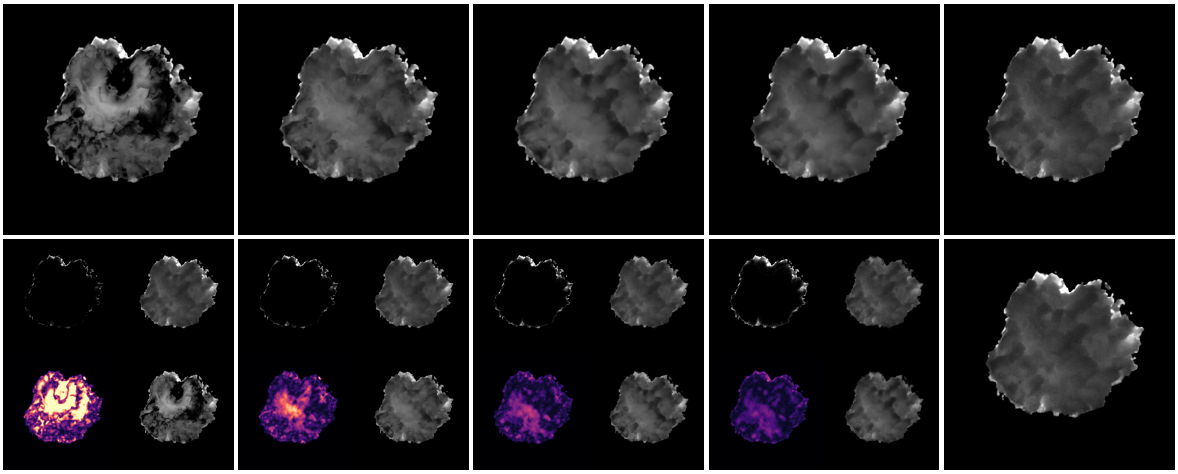
## Scene A - Backlit



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



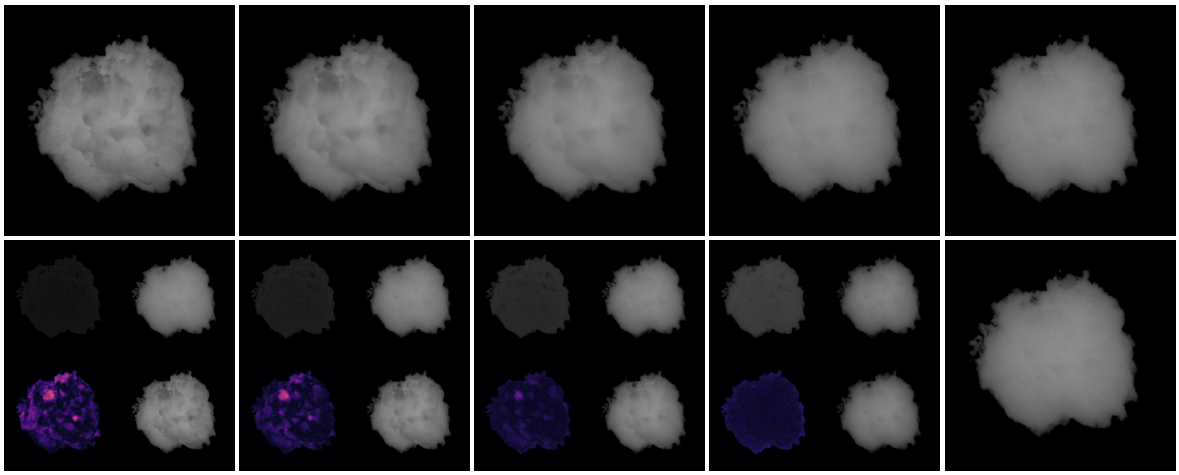
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



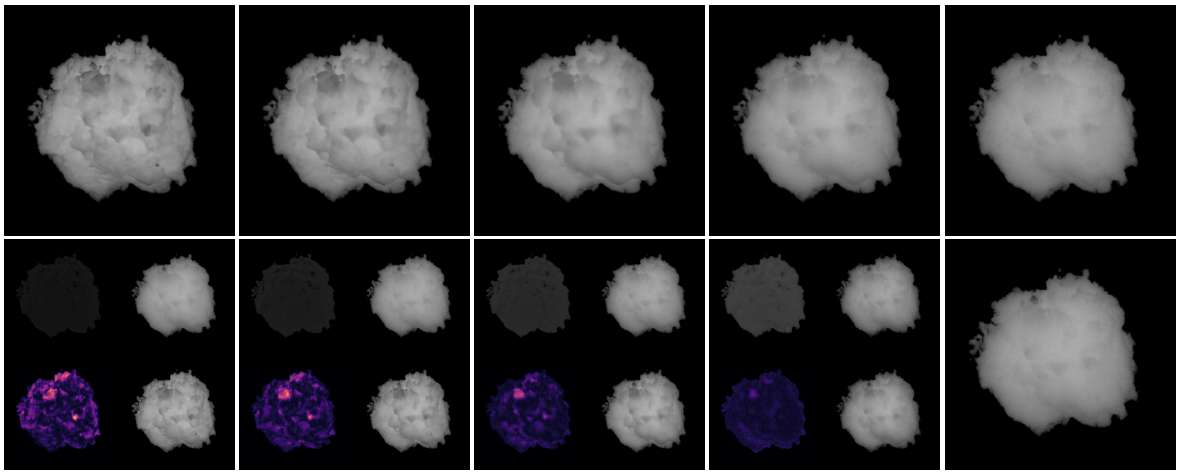
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure B.3.:*** *Visual comparison between our method and the reference.*

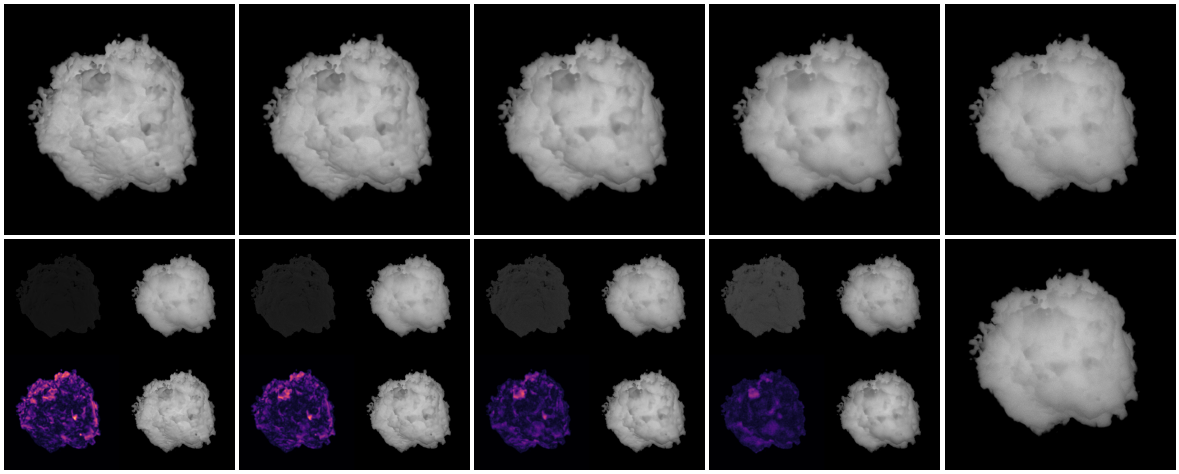
## Scene A - Frontlit



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure B.4.:*** *Visual comparison between our method and the reference.*

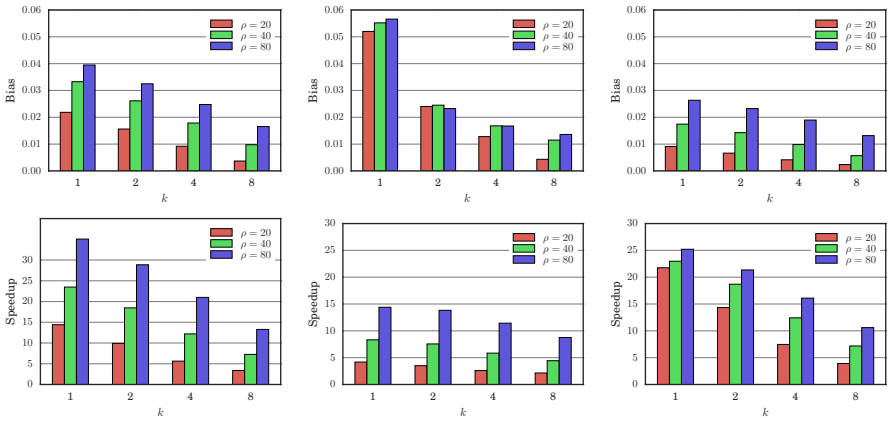
## B.2. Scene B

| | $k=1$ | $k=2$ | $k=4$ | $k=8$ |
|---|---|---|---|---|
| Toplit | | | | |
| $\rho=20$ | 0.0218 | 0.0156 | 0.0092 | 0.0037 |
| $\rho=40$ | 0.0333 | 0.0261 | 0.0179 | 0.0098 |
| $\rho=80$ | 0.0395 | 0.0325 | 0.0248 | 0.0165 |
| Backlit | | | | |
| $\rho=20$ | 0.0520 | 0.0240 | 0.0128 | 0.0043 |
| $\rho=40$ | 0.0552 | 0.0245 | 0.0168 | 0.0115 |
| $\rho=80$ | 0.0566 | 0.0233 | 0.0167 | 0.0136 |
| Frontlit | | | | |
| $\rho=20$ | 0.0091 | 0.0066 | 0.0041 | 0.0024 |
| $\rho=40$ | 0.0174 | 0.0143 | 0.0098 | 0.0057 |
| $\rho=80$ | 0.0263 | 0.0232 | 0.0190 | 0.0132 |

***Table B.3.:*** *Bias of our method compared to the reference.*

| | Reference | $k=1$ | | $k=2$ | | $k=4$ | | $k=8$ | |
|---|---|---|---|---|---|---|---|---|---|
| Toplit | | | | | | | | | |
| $\rho=20$ | 111.1 | 7.7 | 14.4× | 11.2 | 9.9× | 19.7 | 5.6× | 32.8 | 3.4× |
| $\rho=40$ | 280.3 | 11.9 | 23.5× | 15.2 | 18.5× | 23.0 | 12.2× | 38.6 | 7.3× |
| $\rho=80$ | 579.5 | 16.5 | 35.1× | 20.1 | 28.9× | 27.6 | 21.0× | 43.6 | 13.3× |
| Backlit | | | | | | | | | |
| $\rho=20$ | 351.7 | 84.1 | 4.2× | 100.3 | 3.5× | 135.7 | 2.6× | 163.9 | 2.1× |
| $\rho=40$ | 417.6 | 50.2 | 8.3× | 55.2 | 7.6× | 71.4 | 5.8× | 94.2 | 4.4× |
| $\rho=80$ | 514.8 | 35.8 | 14.4× | 37.2 | 13.8× | 45.0 | 11.4× | 58.8 | 8.8× |
| Frontlit | | | | | | | | | |
| $\rho=20$ | 66.8 | 3.1 | 21.8× | 4.7 | 14.3× | 8.9 | 7.5× | 17.1 | 3.9× |
| $\rho=40$ | 230.0 | 10.0 | 23.0× | 12.3 | 18.7× | 18.5 | 12.4× | 32.0 | 7.2× |
| $\rho=80$ | 697.8 | 27.7 | 25.2× | 32.7 | 21.4× | 43.3 | 16.1× | 65.8 | 10.6× |

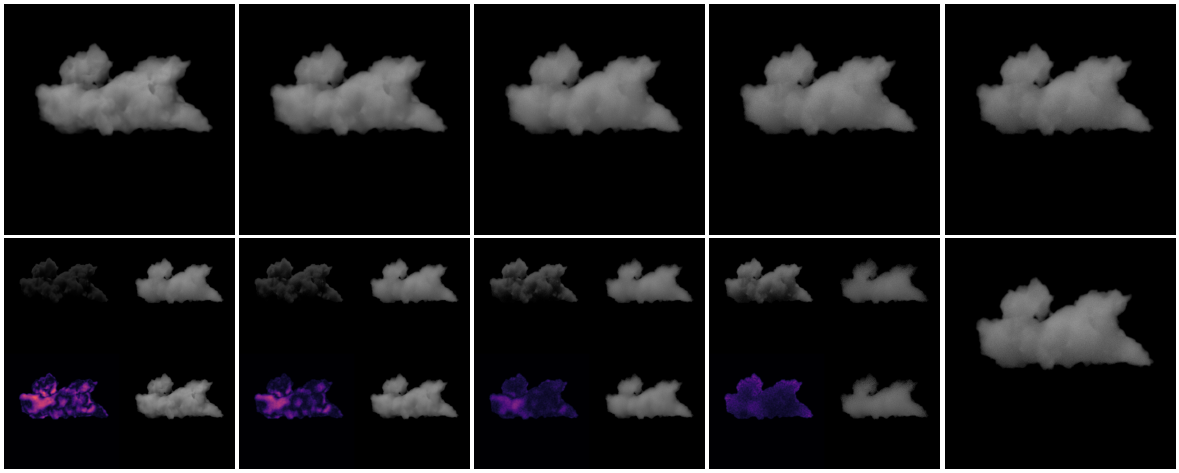
***Table B.4.:*** *Speedup with respect to TTUV of our method compared to the reference.*



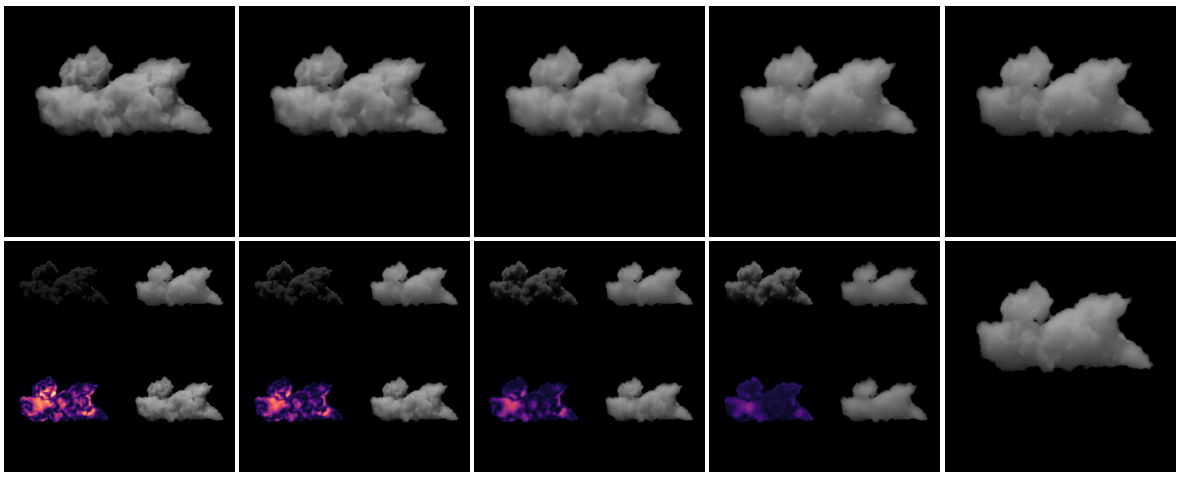

**(a)** Toplit **(b)** Backlit **(c)** Frontlit

***Figure B.5.:*** *Comparison of bias (top row) and speedup with respect to TTUV (bottom row).*

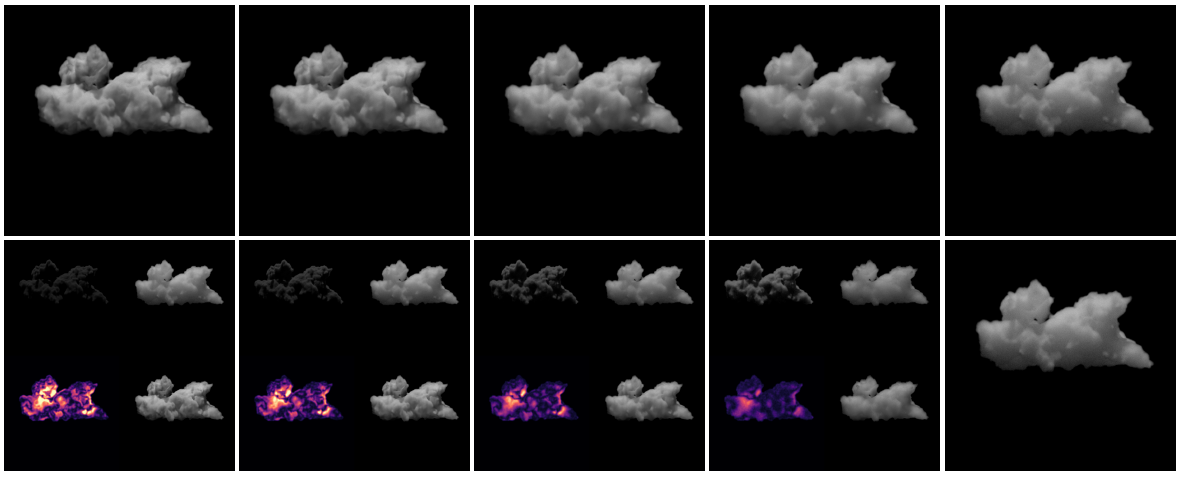
## Scene B - Toplit



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



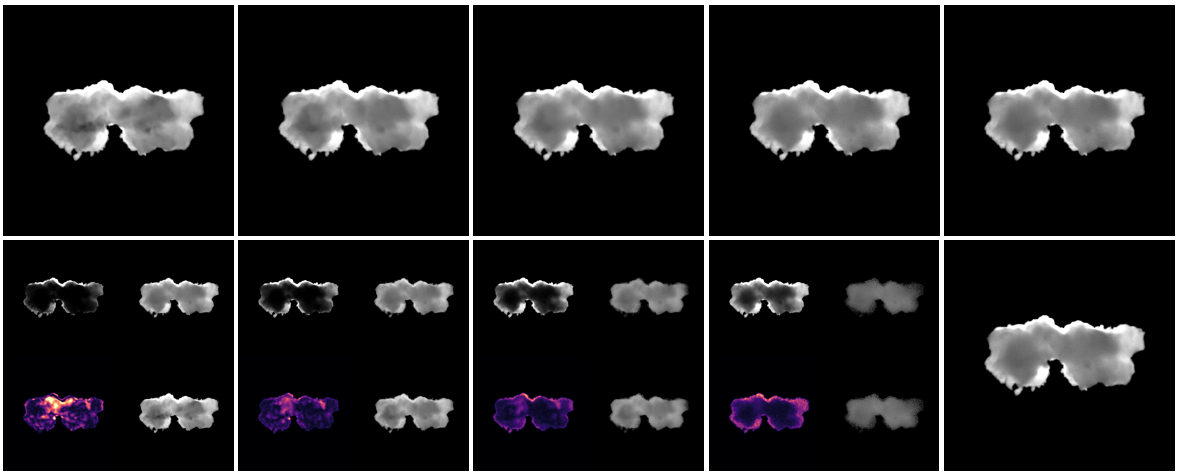
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



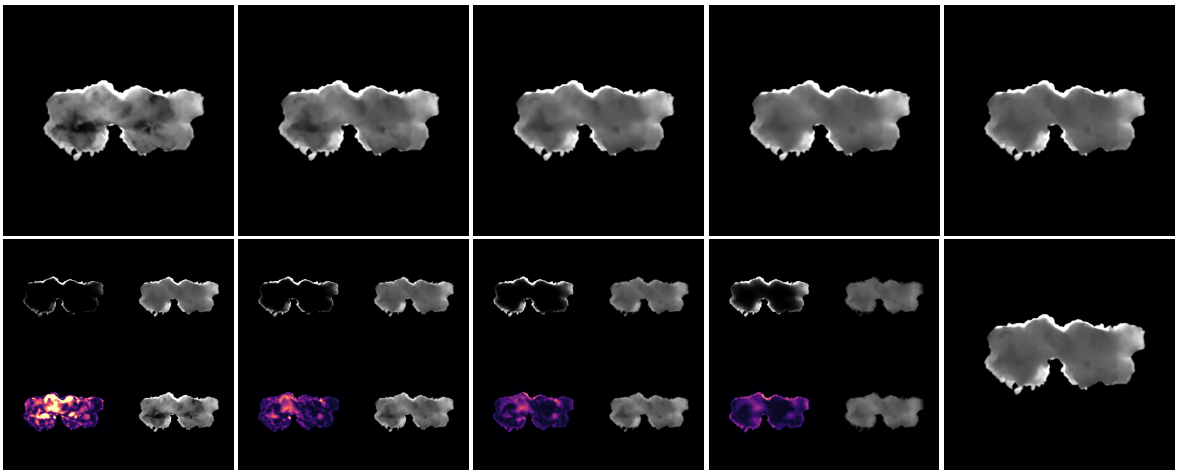
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure B.6.:*** *Visual comparison between our method and the reference.*

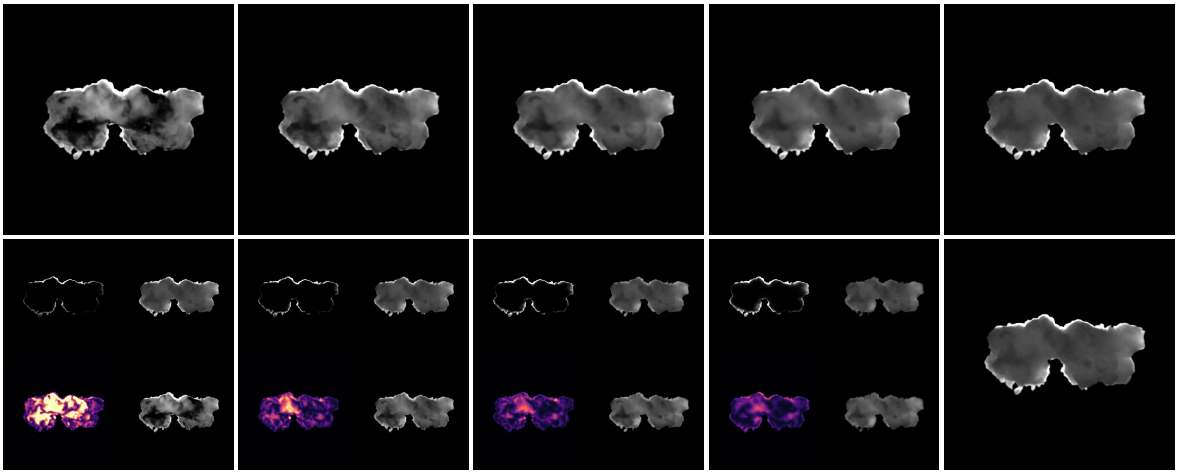
## Scene B - Backlit



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



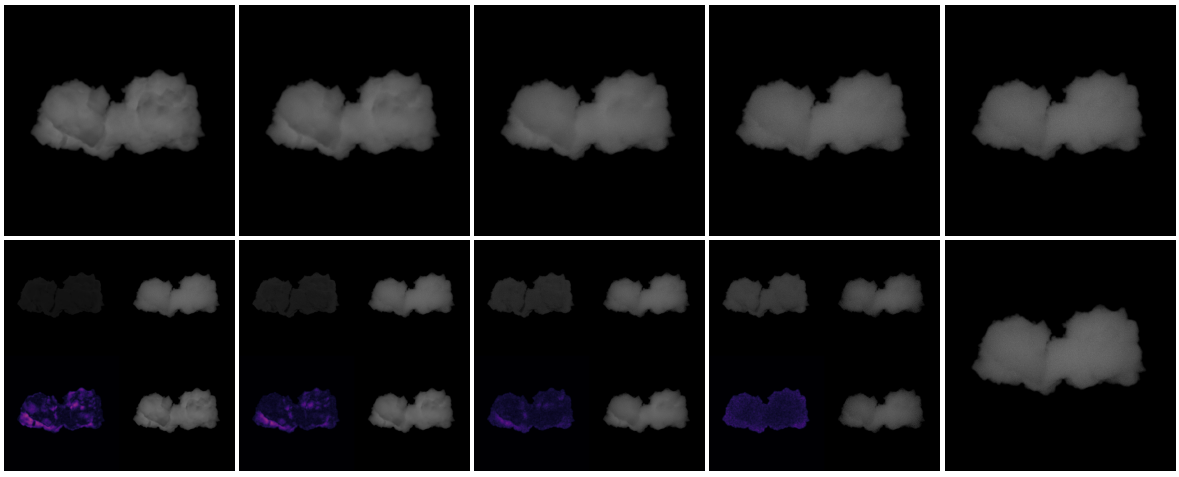
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



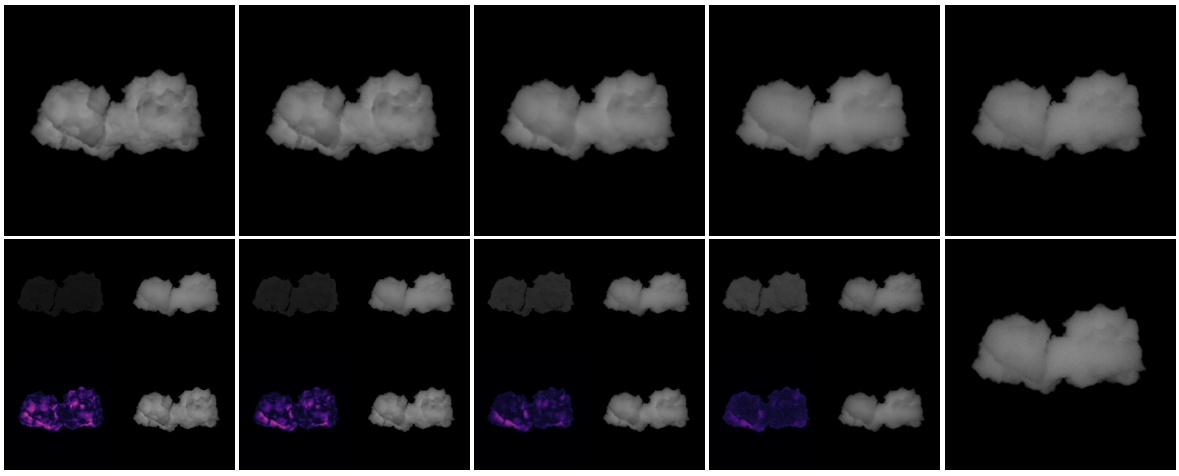
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure B.7.:*** *Visual comparison between our method and the reference.*

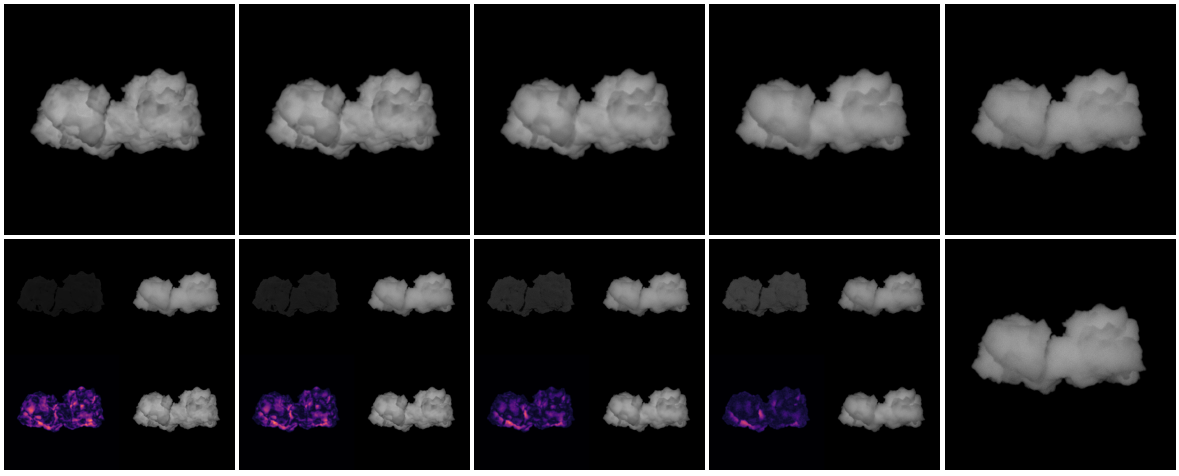
## Scene B - Frontlit



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure B.8.:*** *Visual comparison between our method and the reference.*

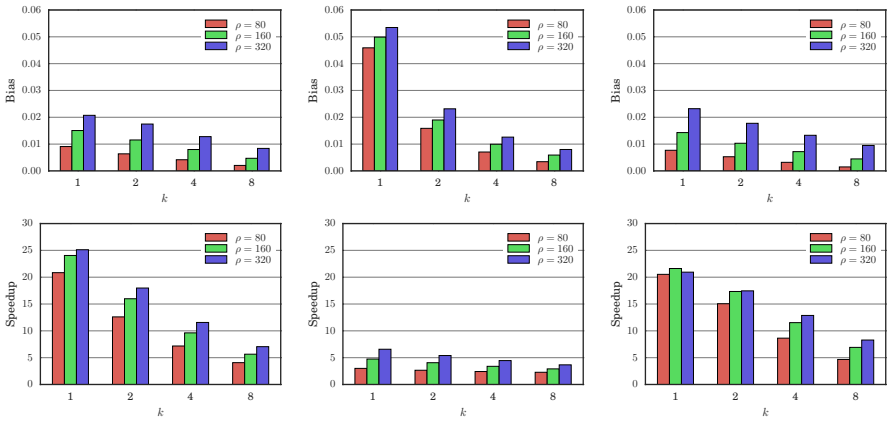
## B.3. Scene C

| | $k=1$ | $k=2$ | $k=4$ | $k=8$ |
|---|---|---|---|---|
| Toplit | | | | |
| $\rho=80$ | 0.0091 | 0.0064 | 0.0042 | 0.0021 |
| $\rho=160$ | 0.0151 | 0.0115 | 0.0080 | 0.0047 |
| $\rho=320$ | 0.0207 | 0.0175 | 0.0128 | 0.0084 |
| Backlit | | | | |
| $\rho=80$ | 0.0459 | 0.0159 | 0.0071 | 0.0034 |
| $\rho=160$ | 0.0499 | 0.0190 | 0.0100 | 0.0059 |
| $\rho=320$ | 0.0535 | 0.0232 | 0.0126 | 0.0080 |
| Frontlit | | | | |
| $\rho=80$ | 0.0077 | 0.0053 | 0.0032 | 0.0015 |
| $\rho=160$ | 0.0143 | 0.0103 | 0.0072 | 0.0045 |
| $\rho=320$ | 0.0232 | 0.0178 | 0.0133 | 0.0095 |

***Table B.5.:*** *Bias of our method compared to the reference.*

| | Reference | $k=1$ | | $k=2$ | | $k=4$ | | $k=8$ | |
|---|---|---|---|---|---|---|---|---|---|
| Toplit | | | | | | | | | |
| $\rho=80$ | 142.4 | 6.8 | 20.8× | 11.3 | 12.6× | 19.8 | 7.2× | 35.0 | 4.1× |
| $\rho=160$ | 385.4 | 16.0 | 24.0× | 24.1 | 16.0× | 40.1 | 9.6× | 68.2 | 5.6× |
| $\rho=320$ | 901.2 | 35.9 | 25.1× | 50.1 | 18.0× | 77.9 | 11.6× | 127.8 | 7.1× |
| Backlit | | | | | | | | | |
| $\rho=80$ | 1223.9 | 407.5 | 3.0× | 461.1 | 2.7× | 506.0 | 2.4× | 537.5 | 2.3× |
| $\rho=160$ | 2061.6 | 434.6 | 4.7× | 507.6 | 4.1× | 608.6 | 3.4× | 708.5 | 2.9× |
| $\rho=320$ | 3152.2 | 479.7 | 6.6× | 583.8 | 5.4× | 707.4 | 4.5× | 860.9 | 3.7× |
| Frontlit | | | | | | | | | |
| $\rho=80$ | 87.1 | 4.2 | 20.5× | 5.8 | 15.1× | 10.1 | 8.6× | 18.7 | 4.7× |
| $\rho=160$ | 352.7 | 16.3 | 21.6× | 20.3 | 17.3× | 30.6 | 11.5× | 51.0 | 6.9× |
| $\rho=320$ | 1157.1 | 55.3 | 20.9× | 66.3 | 17.5× | 89.9 | 12.9× | 139.6 | 8.3× |

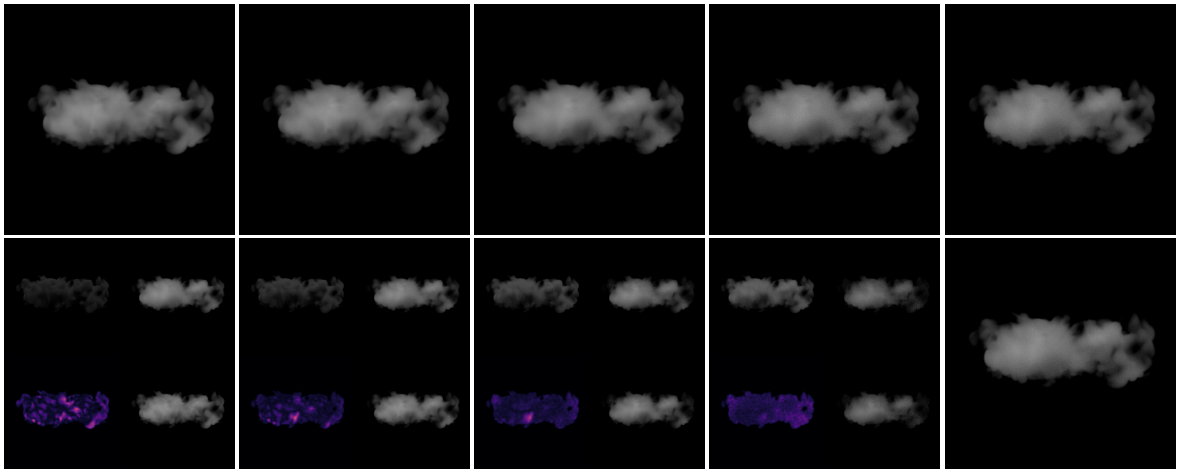
***Table B.6.:*** *Speedup with respect to TTUV of our method compared to the reference.*



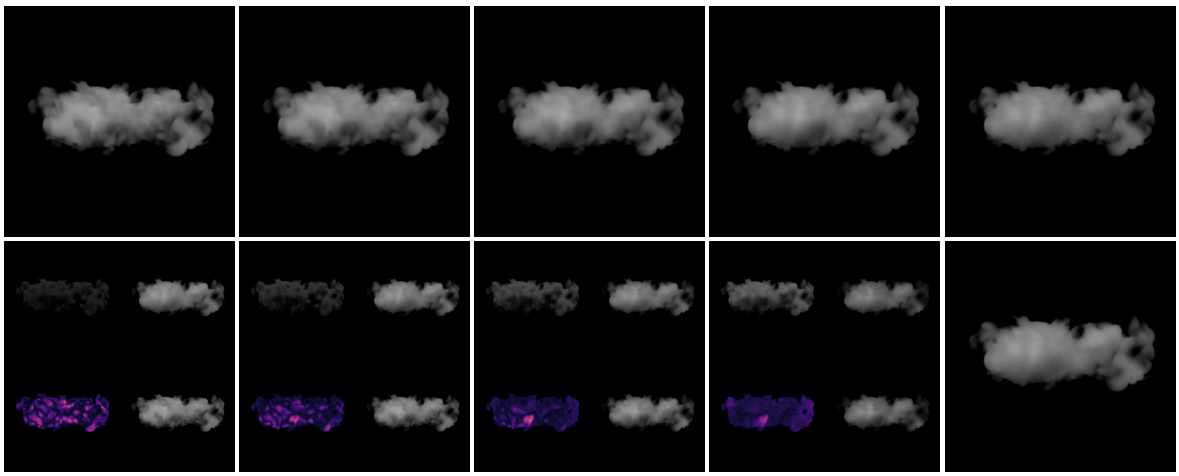

**(a)** Toplit **(b)** Backlit **(c)** Frontlit

***Figure B.9.:*** *Comparison of bias (top row) and speedup with respect to TTUV (bottom row).*

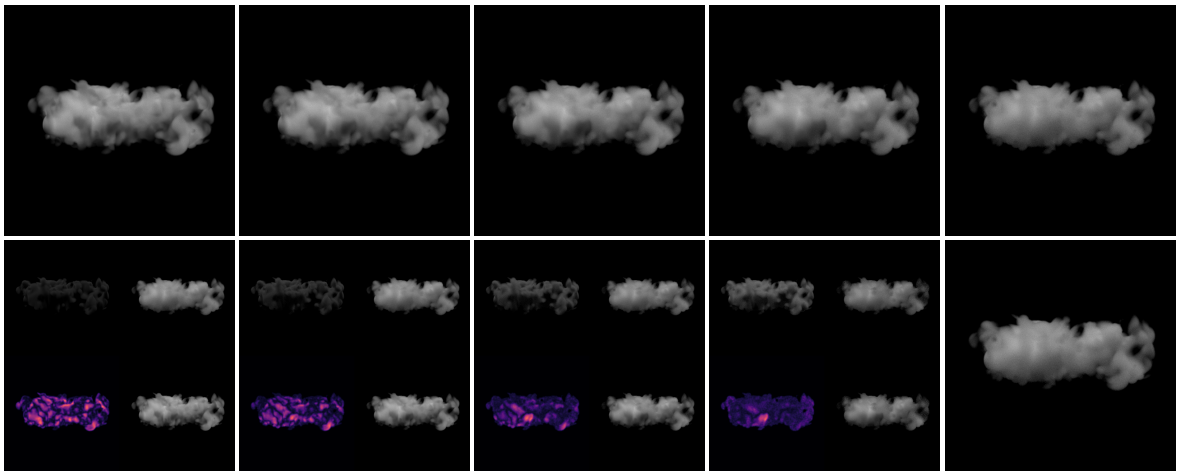
## Scene C - Toplit



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



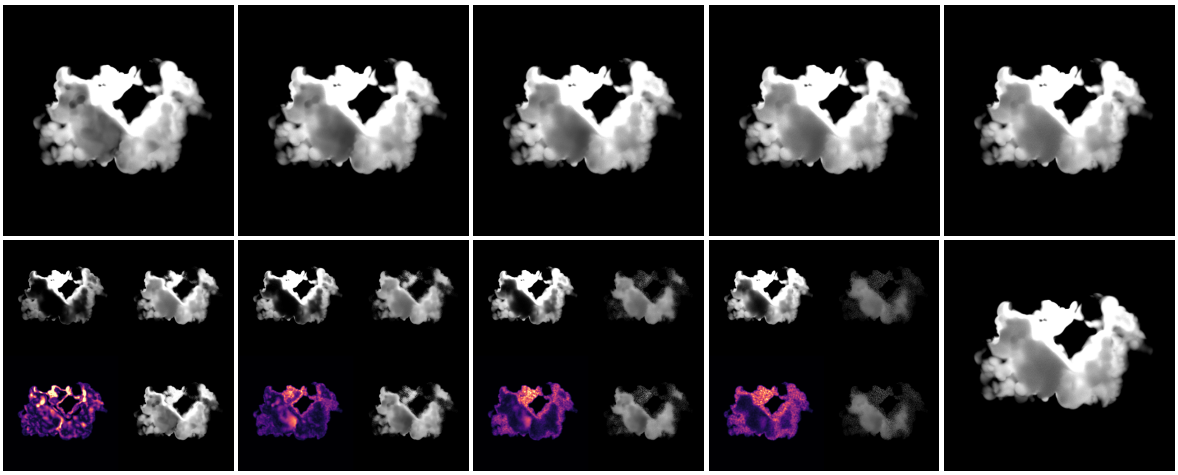
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



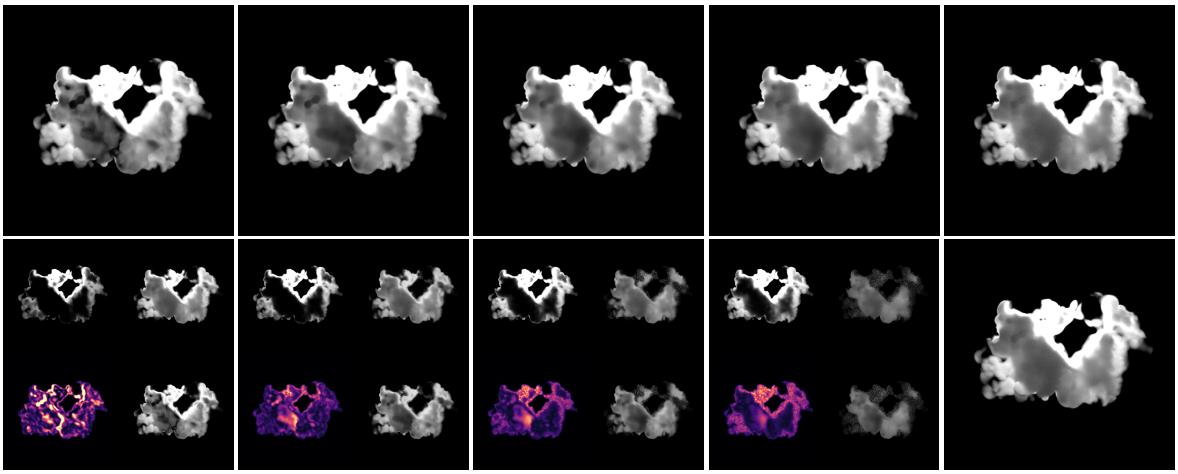
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure B.10.:*** *Visual comparison between our method and the reference.*

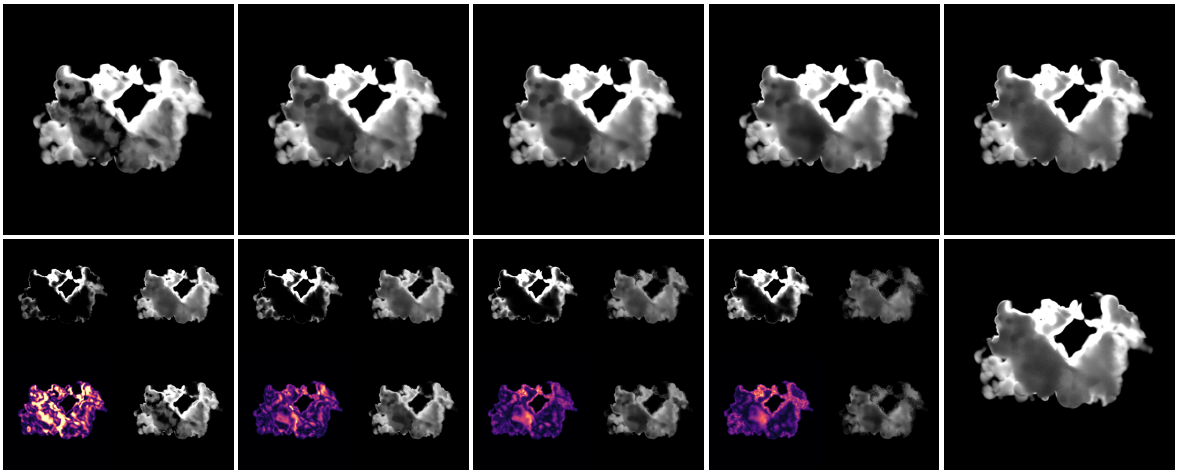
## Scene C - Backlit



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



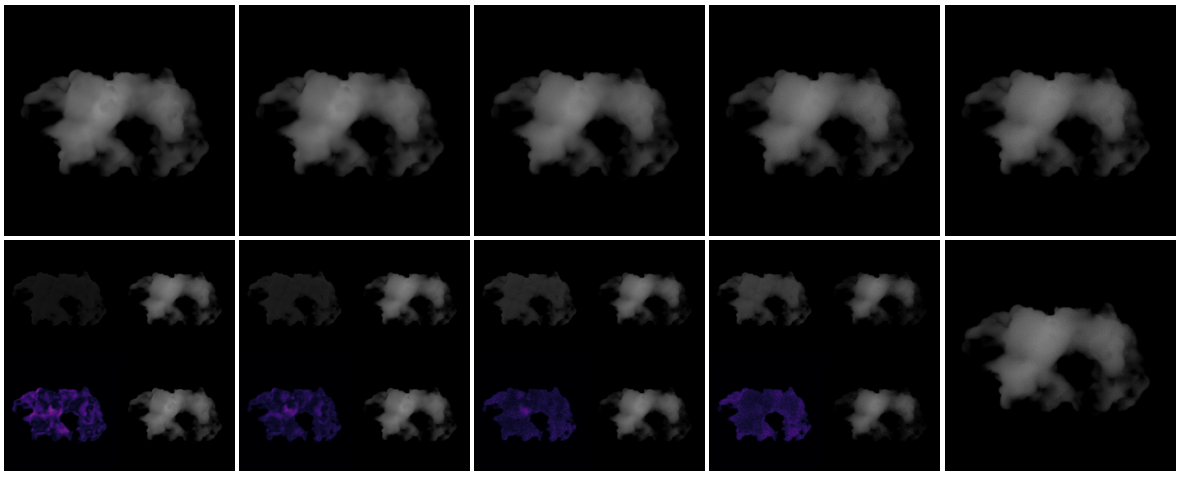
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



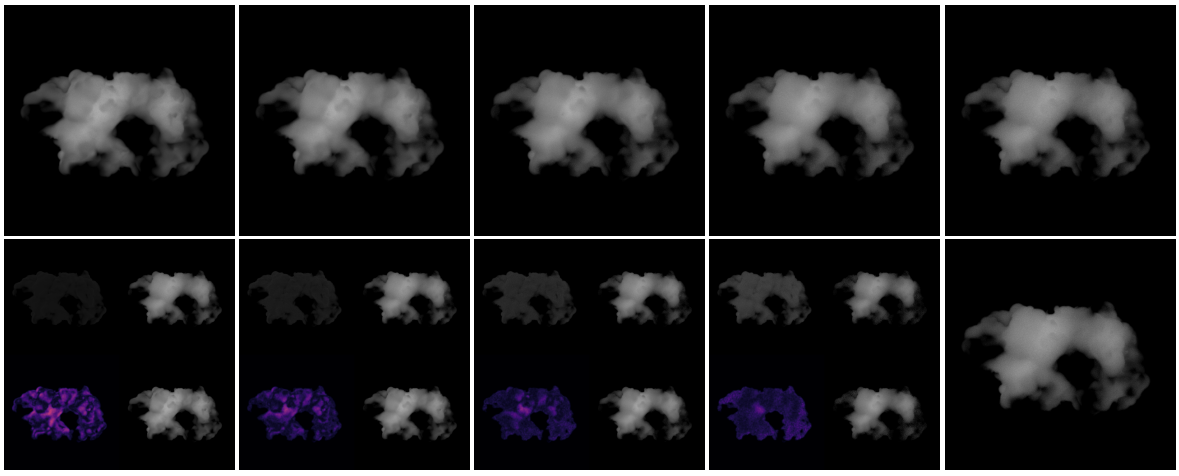
**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure B.11.:*** *Visual comparison between our method and the reference.*

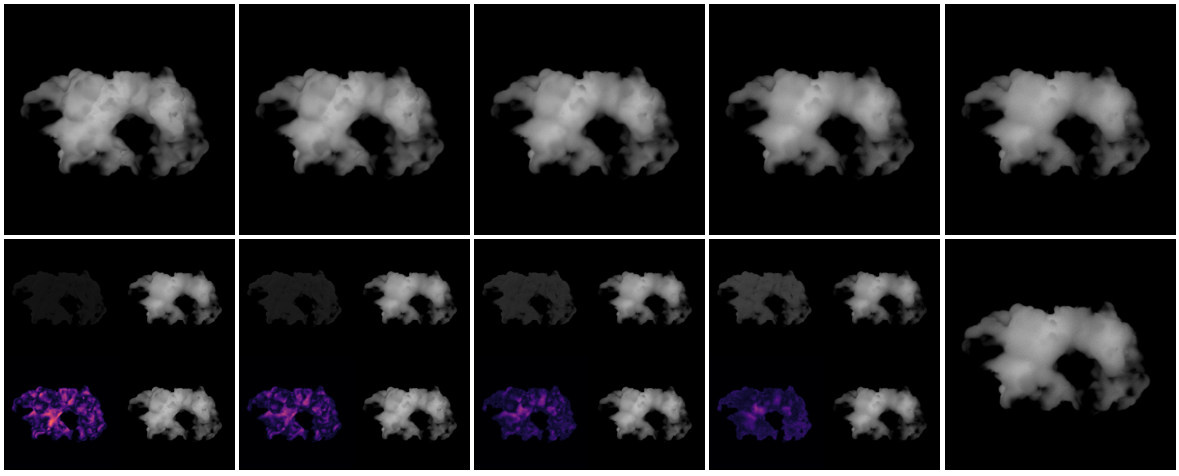
## Scene C - Frontlit



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference



**(a)** $k = 1$ **(b)** $k = 2$ **(c)** $k = 4$ **(d)** $k = 8$ **(e)** Reference

***Figure B.12.:*** *Visual comparison between our method and the reference.*

# Bibliography


[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[BCV12] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[BNL06] Antoine Bouthors, Fabrice Neyret, and Sylvain Lefebvre. Real-time realistic illumination and shading of stratiform clouds. In *Proceedings of the Second Eurographics Conference on Natural Phenomena*, NPH'06, pages 41–50, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[BNM+08] Antoine Bouthors, Fabrice Neyret, Nelson Max, Eric Bruneton, and Cyril Crassin. Interactive multiple anisotropic scattering in clouds. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, I3D '08, pages 173–182, New York, NY, USA, 2008. ACM.

[Bou08] Antoine Bouthors. *Realistic rendering of clouds in real-time / Rendu réaliste de nuages en temps-réel*. PhD thesis, Université Joseph Fourier, Grenoble, France, june 2008.

[Col68] W. A. Coleman. Mathematical verification of a certain Monte Carlo sampling tech-

nique and applications of the technique to radiation transport problems. *Nuclear Science and Engineering*, 32(1):76–81, April 1968.

[D'E12] Eugene D'Eon. A better dipole. Technical report, http://www.eugenedeon.com, 2012.

[d'E13] Eugene d'Eon. A dual-beam method-of-images 3d searchlight BSSRDF. *CoRR*, abs/1311.0955, 2013.

[dI11] Eugene d'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 30(4):56:1–56:14, July 2011.

[DJ08] Craig Donner and Henrik Wann Jensen. Rendering translucent materials using photon diffusion. In *ACM SIGGRAPH 2008 Classes*, pages 4:1–4:9, New York, NY, USA, 2008. ACM.

[DKE14] Filip Sadlo Thomas Ertl David Koerner, Jamie Portsmouth and Bernd Eberhardt. Flux-limited diffusion for multiple scattering in participating media. *CoRR*, abs/1403.8105, 2014.

[ERDS14] Oskar Elek, Tobias Ritschel, Carsten Dachsbacher, and Hans-Peter Seidel. Interactive light scattering with principal-ordinate propagation. In *Proceedings of Graphics Interface 2014*, GI '14, pages 87–94, Toronto, Ont., Canada, Canada, 2014. Canadian Information Processing Society.

[Fat09] Raanan Fattal. Participating media illumination using light propagation maps. *ACM Trans. Graph.*, 28(1):7:1–7:11, February 2009.

[FCJ07] Jeppe Revall Frisvad, Niels Jørgen Christensen, and Henrik Wann Jensen. Computing the scattering properties of participating media using lorenz-mie theory. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 26(3), jul 2007.

[FHK14] Jeppe Revall Frisvad, Toshiya Hachisuka, and Thomas Kim Kjeldsen. Directional dipole model for subsurface scattering. *ACM Transactions on Graphics*, 34(1):5:1–5:12, December 2014.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[GJ+10] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3, 2010. http://eigen.tuxfamily.org.

[GKH+13] Iliyan Georgiev, Jaroslav Křivánek, Toshiya Hachisuka, Derek Nowrouzezahrai, and Wojciech Jarosz. Joint importance sampling of low-order volumetric scattering. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 32(6):164:1–164:14, November 2013.

[HCJ13a] Ralf Habel, Per H. Christensen, and Wojciech Jarosz. Classical and improved diffusion theory for subsurface scattering. Technical report, Disney Research ZÃijrich, June 2013.

[HCJ13b] Ralf Habel, Per H. Christensen, and Wojciech Jarosz. Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 32(4), June 2013.

[HG41] L. G. Henyey and J. L. Greenstein. Diffuse radiation in the galaxy. *Astrophysical Journal*, 93:70–83, January 1941.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[Jak10] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[Jar08] Wojciech Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, September 2008.

[JB02] Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 21(3):576–581, July 2002.

[JC98] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proc. of SIGGRAPH 98*, Annual Conference Series, pages 311–320, New York, NY, USA, 1998. ACM.

[JDZJ08] Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. Radiance caching for participating media. *ACM Transactions on Graphics*, 27(1):7:1–7:11, March 2008.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In *Proc. of Eurographics Workshop on Rendering Techniques*, pages 21–30, London, UK, August 1996. Springer-Verlag.

[JM12] Wenzel Jakob and Steve Marschner. Manifold exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31(4):58:1–58:13, July 2012.

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proc. of SIGGRAPH 01*, Annual Conference Series, pages 511–518, New York, NY, USA, 2001. ACM.

[JNT+11] Wojciech Jarosz, Derek Nowrouzezahrai, Robert Thomas, Peter-Pike Sloan, and Matthias Zwicker. Progressive photon beams. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 30(6):181:1–181:12, December 2011.

[JZJ08] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proc. of Eurographics)*, 27(2):557–566, April 2008.

[Kaj86] James T. Kajiya. The rendering equation. *Computer Graphics (Proc. of SIGGRAPH)*, pages 143–150, 1986.

[Kal63] Malvin H. Kalos. On the estimation of flux at a point by Monte Carlo. *Nuclear*

*Science and Engineering*, 16:111–117, 1963.

[KBS15] Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. A machine learning approach for filtering monte carlo noise. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2015)*, 34(4), 2015.

[KGH+] Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*.

[KK51] J. F. Kenney and E. S. Keeping. Mathematics of statistics. part 2, 1951.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LBOM98] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Effiicient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, UK, 1998. Springer-Verlag.

[LH95] Charles L. Lawson and Richard J. Hanson. *Solving least squares problems*, volume 15 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995. Revised reprint of the 1974 original.

[LJS+15] L'ubor Ladický, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross. Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics*, 34(6):199:1–199:9, October 2015.

[LK11] André Liemert and Alwin Kienle. Analytical solution of the radiative transfer equation for infinite-space fluence. *Phys. Rev. A*, 83:015804, January 2011.

[Lor90] Ludwig Lorenz. Lysbevægelsen i og uden for en af plane lysbølger belyst kugle. In *Det Kongelige Danske Videnskabernes Selskabs Skrifter (trykt utg.): Naturvidenskabelig og Mathematisk Afdeling*, 1890.

[LW96] Eric P. Lafortune and Yves D. Willems. Rendering participating media with bidirectional path tracing. In *Proc. of Eurographics Workshop on Rendering Techniques*, pages 91–100, London, UK, August 1996. Springer-Verlag.

[MBJ+06] R. Keith Morley, Solomon Boulos, Jared Johnson, David Edwards, Peter Shirley, Michael Ashikhmin, and Simon Premože. Image synthesis using adjoint photons. In *Proc. of Graphics Interface*, pages 179–186, Toronto, Ont., Canada, 2006.

[Mie08] Gustav Mie. Beiträge zur optik trüber medien, speziell kolloidaler metallösungen. *Annalen der Physik*, 330:377–445, 1908.

[MLJ+13] Ken Museth, Jeff Lait, John Johanson, Jeff Budsberg, Ron Henderson, Mihai Alden, Peter Cucka, David Hill, and Andrew Pearce. Openvdb: An open-source data structure and toolkit for high-resolution volumes. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, pages 19:1–19:1, New York, NY, USA, 2013. ACM.

[MP43] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent

in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[MU49] Nicholas Metropolis and Stanislaw M. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, September 1949.

[NAM+16] Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, Hans-Peter Seidel, and Tobias Ritschel. Deep shading: Convolutional neural networks for screen-space shading. *CoRR*, abs/1603.06078, 2016.

[NMN87] Tomoyuki Nishita, Yasuhiro Miyawaki, and Eihachiro Nakamae. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. *Computer Graphics (Proc. of SIGGRAPH)*, pages 303–310, 1987.

[NNDJ12] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 31(4):60:1–60:11, July 2012.

[NSJ14] Jan Novák, Andrew Selle, and Wojciech Jarosz. Residual ratio tracking for estimating attenuation in participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 33(6), November 2014.

[PH89] Ken H. Perlin and Eric M. Hoffert. Hypertexture. *Computer Graphics (Proc. of SIGGRAPH)*, 23(3):253–262, July 1989.

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[PKK00] Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis light transport for participating media. In *Proc. of Eurographics Workshop on Rendering Techniques*, pages 11–22, London, UK, 2000. Springer-Verlag.

[PM93] Sumanta N. Pattanaik and S. P. Mudur. Computation of global illumination in a participating medium by monte carlo simulation. *The Journal of Visualization and Computer Animation*, 4(3):133–152, July–September 1993.

[PSS11] Vincent Pegoraro, Mathias Schott, and Philipp Slusallek. A mathematical framework for efficient closed-form single scattering. In *Proc. of the 37th Graphics Interface Conference*, pages 151–158, 2011.

[Ray71] John William Strutt Lord Rayleigh. On the scattering of light by small particles. *Philosophical Magazine*, 64:447–454, 1871.

[RDL+15] Peiran Ren, Yue Dong, Stephen Lin, Xin Tong, and Baining Guo. Image based relighting using neural networks. *ACM Transactions on Graphics*, 34(4):111:1–111:12, July 2015.

[REK+04] Kirk Riley, David S. Ebert, Martin Kraus, Jerry Tessendorf, and Charles Hansen. Efficient rendering of atmospheric phenomena. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques*, EGSR'04, pages 375–386, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[RHW88] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning repre-

sentations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[Roj96] Raul Rojas. The backpropagation algorithm. In *Neural networks*, pages 149–182. Springer, 1996.

[RSK08] Matthias Raab, Daniel Seibert, and Alexander Keller. Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 591–606. Springer, 2008.

[RWG+13] Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. Global illumination with radiance regression functions. *ACM Transactions on Graphics*, 32(4):130:1–130:12, July 2013.

[SKSU05] László Szirmay-Kalos, Mateu Sbert, and Tamás Ummenhoffer. Real-time multiple scattering in participating media with illumination networks. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, EGSR '05, pages 277–282, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[SKTM11] László Szirmay-Kalos, Balázs Tóth, and Milán Magdics. Free path sampling in high resolution inhomogeneous participating media. *Computer Graphics Forum*, 30(1):85–97, 2011.

[SRNN05] Bo Sun, Ravi Ramamoorthi, Srinivasa G. Narasimhan, and Shree K. Nayar. A practical analytic single scattering model for real time rendering. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1040–1049, 2005.

[Sta95] Jos Stam. Multiple scattering as a diffusion process. In *Proc. of Eurographics Workshop on Rendering Techniques*, pages 41–50. Springer-Verlag, 1995.

[STC04] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

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

[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 *Proc. of SIGGRAPH 97*, Annual Conference Series, pages 65–76, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[VKŠ+14] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. Online learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 33(4), August 2014.

[WABG06] Bruce Walter, Adam Arbree, Kavita Bala, and D P Greenberg. Multidimensional lightcuts. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 25(3):1081–1088, July 2006.

[Wel62] B. P. Welford. Note on a method for calculating corrected sums of squares and

products. *Technometrics*, 4(3):419–420, 1962.

[Wer74] Paul Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. 1974.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[WKL13] Magnus Wrenninge, Christopher D. Kulla, and Viktor Lundqvist. Oz: the great and volumetric. In *SIGGRAPH Talks*, page 46:1. ACM, 2013.

[WPW89a] Douglas R Wyman, Michael S Patterson, and Brian C Wilson. Similarity relations for anisotropic scattering in monte carlo simulations of deeply penetrating neutral particles. *Journal of Computational Physics*, 81(1):137–150, 1989.

[WPW89b] Douglas R. Wyman, Michael S. Patterson, and Brian C. Wilson. Similarity relations for the interaction parameters in radiation transport. *Appl. Opt.*, 28(24):5243–5249, December 1989.

[Wre15] Magnus Wrenninge. Art-directable multiple volumetric scattering. In *ACM SIGGRAPH 2015 Talks*, pages 24:1–24:1, New York, NY, USA, 2015. ACM.

[YIC+10] Yonghao Yue, Kei Iwasaki, Bing-Yu Chen, Yoshinori Dobashi, and Tomoyuki Nishita. Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 29(6):177:1–177:8, December 2010.

[ZRB14] Shuang Zhao, Ravi Ramamoorthi, and Kavita Bala. High-order similarity relations in radiative transfer. *ACM Transactions on Graphics*, 33(4):104:1–104:12, July 2014.