Data-driven global importance sampling for physically-based rendering

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Florian Reibold geb. Simon

aus Esslingen am Neckar

Tag der mündlichen Prüfung: 01.07.2020
Erster Gutachter: Prof. Dr.-Ing. Carsten Dachsbacher
Zweiter Gutachter: Prof. Dr.-Ing. Hendrik P. A. Lensch

Für Augustin.

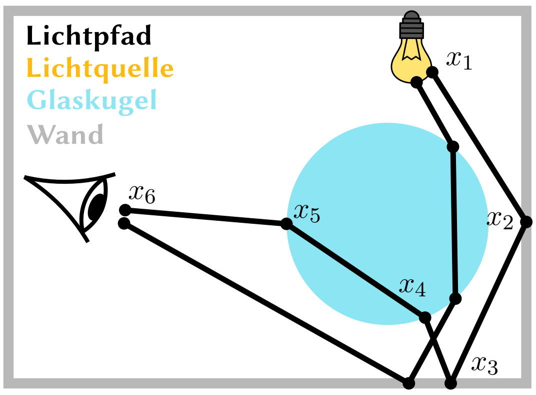
Zusammenfassung

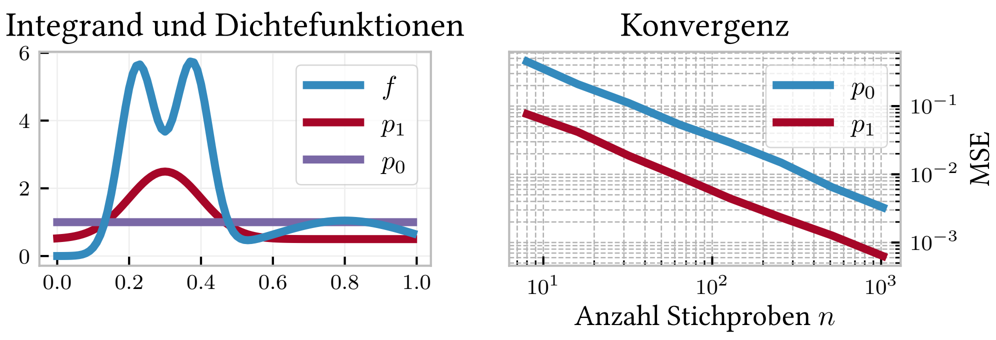
Die fotorealistische Bildsynthese ist ein aktives Forschungsfeld mit vielen Anwendungsbereichen wie z. B. der Architektur- und Produktvisualisierung sowie visuellen Effekten in Filmen. Dabei muss für einen virtuellen Bildsensor der eingehende Lichtfluss durch die Berechnung der Lichtausbreitung in einer virtuellen Szene berechnet werden.

Unter der Annahme geometrischer Optik kann der Pfad eines Photons, welches von einer Lichtquellenposition x_1 ausgeht und nach beliebig vielen Streueregnissen an einem Sensorpunkt x_k auftrifft, als Folge dieser Streuereignisse $X = (x_1, x_2, \ldots, x_{k-1}, x_k)$ beschrieben werden. Bezeichnet man den Raum aller möglichen Pfade beliebiger Länge mit $\mathcal{P}$, so lässt sich die Berechnung der Lichtausbreitung als Bestimmung des Integralwerts

$$I = \int_{\mathcal{P}} f(X)\mu(\mathbf{X})$$

formalisieren. Die *Messfunktion* f enthält dabei Informationen über emittiertes Licht, Sensorantwort, Streuereignisse und geometrische Konfiguration der Oberflächen. Der Pfadraum $\mathcal{P}$ ist unendlichdimensional mit zugehörigem Maß μ. Zur Berechnung werden üblicherweise Monte-Carlo-Verfahren eingesetzt, da deren Konvergenzrate, im Gegensatz zu anderen numerischen Verfahren, unabhängig von der Dimensionalität ist.





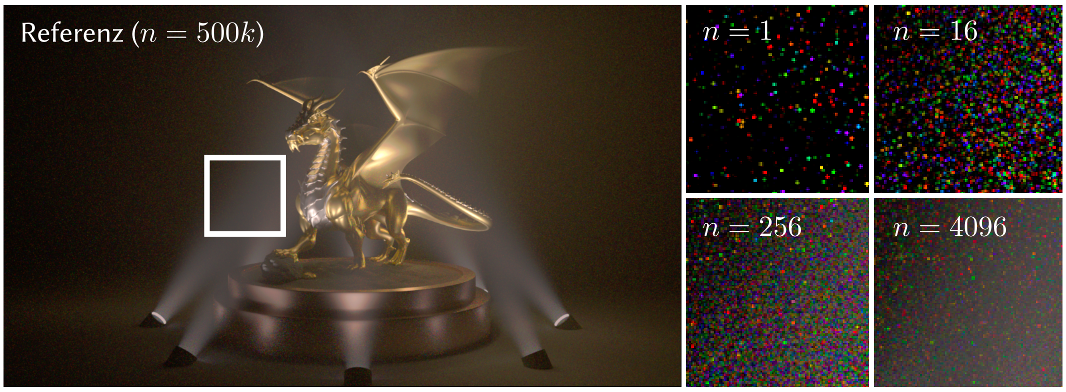
Für die Monte-Carlo-Integration werden unabhängig zufällige Pfade $X_1, \ldots, X_n$ nach einer beliebigen Wahrscheinlichkeitsdichtefunktion $p(X)$, welche die Bedingung $f(X) > 0 \Rightarrow p(X) > 0$ erfüllt, erzeugt. Der Wert

$$\langle I_n^p \rangle = \frac{1}{n} \sum_{i=1}^{n} \frac{f(X_i)}{p(X_i)}$$

ist ein *Monte-Carlo-Schätzer*, der sich nach dem Gesetz der großen Zahlen für zunehmendes n stochastisch immer mehr dem Wert des Integrals I annähert. Wie schnell diese Annäherung stattfindet, hängt dabei sehr stark von der Wahl von p ab. Eine optimale Wahl für p wäre $p \sim f$. Dies ist aber nicht praktikabel, da die Proportionalitätskonstante vom Wert des Integrals abhängt. In vielen Fällen lassen sich aber Wahrscheinlichkeitsdichten bestimmen, die ähnlich zu f, einfach zu normalisieren und schnell auszuwerten sind. Dieses Vorgehen, p möglichst ähnlich zu f zu wählen, bezeichnet man als *Importance Sampling*. Importance Sampling ist eine der wichtigsten Formen der Varianzreduktion in der fotorealistischen Bildsynthese und das zentrale Thema meiner Dissertation.

Die obige Abbildung illustriert an einem eindimensionalen Beispiel, dass schon eine leicht dem Integranden f folgende Wahrscheinlichkeitsdichte (p_1) im Vergleich zu einer uniformen Wahrscheinlichkeitsdichte (p_0) den mittleren quadratischen Fehler (MSE) um fast eine Größenordnung bei gleicher Stichprobengröße (n) verringern kann.

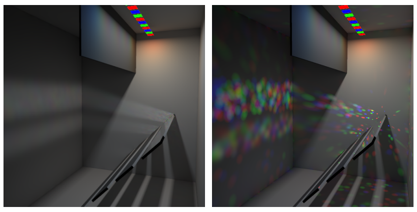
Die Varianz eines Monte-Carlo-Schätzers wirkt sich als störendes Rauschen auf die erzeugten synthetischen Bildern aus. In der folgenden Illustration ist das Monte-Carlo-Rauschen selbst nach 4096 stochastisch erzeugten Pfaden pro Bildpunkt immer noch wahrnehmbar. In dieser Arbeit werden mehrere Verfahren beschrieben, die durch Importance Sampling das Rauschen in synthetischen Bildern für spezielle Anwendungen bei gleicher Laufzeit deutlich reduzieren. Die grundlegenden Ideen dieser Verfahren sind im Folgenden kurz zusammengefasst.

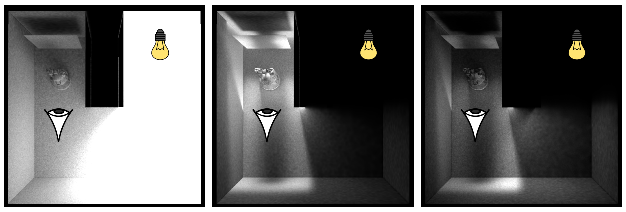


Virtuelle Punktlichtquellen für glänzende Oberflächen

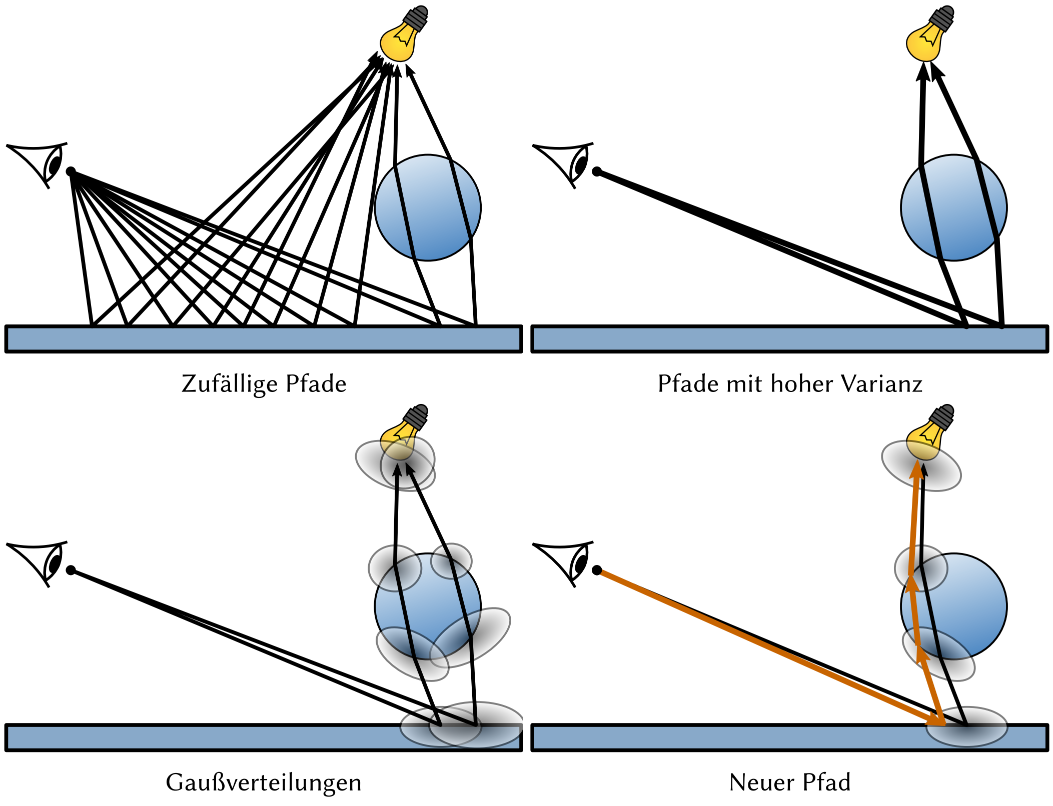
Many-Light-Methoden sind eine Klasse von Monte-Carlo-Verfahren, die rauschfreie Bilder erzeugen. Dabei werden Pfade ausgehend von einer Lichtquelle in der Szene verfolgt und virtuelle Punkt-Lichtquellen (VPLs) an den Streuereignissen platziert. Diese VPLs werden in einem zweiten Schritt für die Beleuchtungsberechnung bei der eigentlichen Bildsynthese verwendet. Jede VPL enthält dabei nur die Information des Lichtpfades, der sie erzeugt hat. In dieser Arbeit wird eine Form von VPLs vorgestellt, die Informationen über beliebig viele Lichtpfade enthalten kann. Speziell für glänzende Oberflächen liefert dieser Ansatz deutlich bessere Ergebnisse im Vergleich zu herkömmlichen VPLs.

Zudem wurde ein datengetriebener Ansatz vorgestellt, der VPLs nur an Stellen der Szene platziert, an denen sie einen Einfluss auf das zu berechnende Bild haben. Die erforderlichen Größen werden über eine Dichteschätzung von Partikeln, die von der Lichtquelle bzw. dem Sensor ausgehend in der Szene verfolgt wurden, ermittelt. Durch die Kombination der jeweiligen Partikeldichten kann man die optimalen Bereiche der Szene zur Positionierung von VPLs identifizieren. Für die somit platzierten VPLs werden zudem alle Informationen der Partikel, die von den Lichtquellen aus verfolgt wurden, genutzt, um eine Schätzung des ausgehenden Lichts zu berechnen.



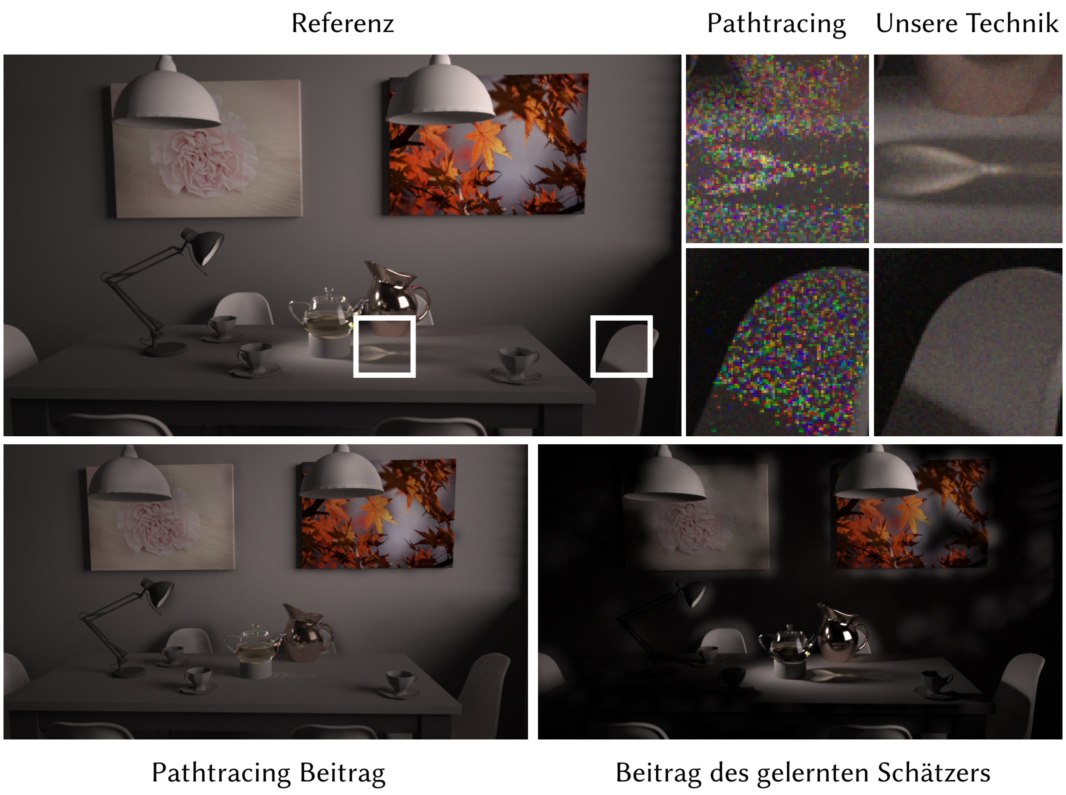


Datengetriebenes adaptives Importance Sampling mit ganzen Lichtpfaden



Das meistgenutzte Verfahren zur fotorealistischen Bildsynthese ist *Pathtracing*. Dieses Verfahren startet die Konstruktion eines Pfads mit einem zufälligen Punkt auf dem Sensor und entscheidet danach lokal und abhängig vom Material an jedem Streuereignis, in welche Richtung der Partikel abgelenkt wird. Für den Großteil der in der Praxis auftretenden Fälle berechnet Pathtracing den Lichttransport sehr effizient. In den Fällen, die Pathtracing nicht effizient behandeln kann, muss man entweder um Größenordnungen mehr Rechenzeit investieren, oder auf kompliziertere Lichttransport-Algorithmen zurückgreifen. In dieser Arbeit wird ein Verfahren beschrieben, dass die Teile des Lichttransports, die mit Pathtracing nicht effizient berechnet werden können, während der Laufzeit identifiziert und inkrementell eine adaptive Wahrscheinlichkeitsdichte lernt, mit deren Hilfe gezielt solche schwierigen Pfade erzeugt werden können.

Dafür werden iterativ Ausreißer, also Pfade X mit kleiner Wahrscheinlichkeit $p(X)$ und hohem Beitrag, gespeichert. Um gezielt ähnliche Pfade konstruieren zu können, werden für jeden “gelernten” Pfad mithilfe von Nachbarschaftsinformationen räumlich begrenzte



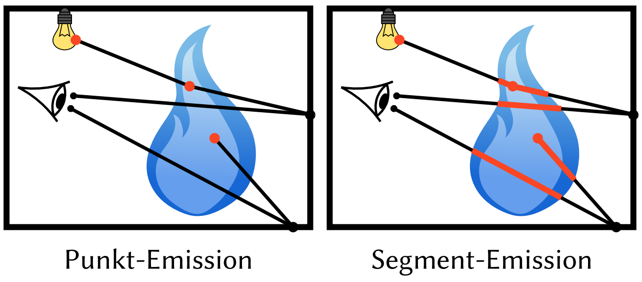
hochdimensionale Gaußverteilungen berechnet. In jeder Iteration werden neue Pfade mit Pathtracing und mithilfe der gelernten Pfaden erzeugt, sodass der Algorithmus sowohl global als auch lokal die schwierigen Beleuchtungseffekte lernt. Effektiv wird dadurch eine Mischverteilung mit multivariaten Gaußverteilungen

$$p_g(X) = \sum_{k=0}^{K} w_k \, \mathcal{N}(X, \mu_k, \Sigma_k)$$

erzeugt, sodass $p(X) = p_u(X) + p_g(X) \sim f(X)$, wobei $p_u(X)$ die zu Pathtracing gehörende Wahrscheinlichkeitsdichte ist.

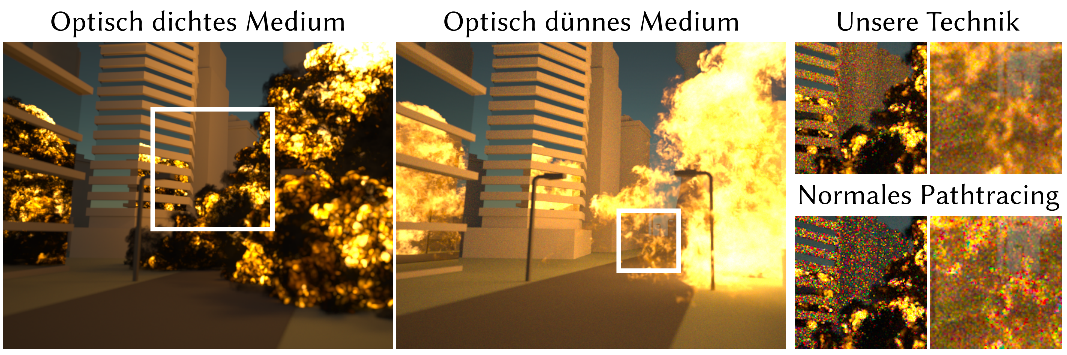
Die resultierende Pfaderzeugung ist ähnlich zur Pfaderzeugung von Pathtracing und wesentlich einfacher als die Pfaderzeugung komplexerer Lichttransport-Algorithmen. Ein weiterer Vorteil dieses datengetriebenen Algorithmus ist, dass es eine lokale Exploration ähnlich zu Markov-Ketten-Monte-Carlo-Verfahren ermöglicht, ohne deren mathematischen Einschränkungen wie detailliertes Gleichgewicht oder Reziprozität erfüllen zu müssen.

Robuste Bildsynthese für emittierende heterogene Medien



Emittierende heterogene Medien sind eine komplexe Form von Lichtquellen und können optisch sehr dünn (Kerzenflamme) oder optisch sehr dicht (Explosion mit Rußbildung) sein. Möchte man Mehrfachstreuung in Medien berechnen, muss eine Distanz im Medium bis zum nächsten Streuereignis bestimmt werden. Diese Distanz wird üblicherweise stochastisch und proportional zur optischen Durchlässigkeit gewählt. Für die Berücksichtigung der volumetrischen Emission wird an diesen Streuereignissen die Emission evaluiert. In dünnen Medien, in denen Streuereignisse sehr selten sind, führt das dazu, dass eine sehr große Anzahl an Pfaden konstruiert werden muss, um die Emission hinreichend abzutasten.

In dieser Arbeit wird Verfahren vorgestellt, welches erlaubt die Lichtemission entlang des *Pfadsegments* im Medium bis zum nächsten Streuereignis zu berücksichtigen. Damit können auch Pfade, die ein emittierendes Medium ohne Streuereignis passieren, zum Bild beitragen. Um diese Methode effizient in einen Pathtracing Algorithmus zu integrieren, wurde zudem ein Verfahren zur direkten Abtastung der Emission in heterogenen Medien vorgestellt, das Lichtemission auf Pfadsegmenten behandeln kann. Dieses Verfahren vermeidet außerdem, dass für dichte Medien die optische Durchlässigkeit über lange Strecken berechnet werden muss. Dadurch ist dieses Verfahren für optisch dichte Medien deutlich effizienter, da die Berechnung der optischen Durchlässigkeit über lange Strecken sehr aufwändig und bei dichten Medien das Ergebnis oft nahe Null ist.



Contents

1 Introduction

Physically-based rendering is the process of synthesizing images by simulating the physical properties of light as accurately as possible with the objective to create computer-generated images that are indistinguishable from images or films taken with a real camera.

The ability to generate such images has a wide variety of applications. Companies can show new products even before a potentially very costly physical prototype has been built. It allows fast iterations of product designs enabling customers to give feedback early. Architects can model and render the interior of a building to see if sufficient light can enter from the outside. Additionally, they can check whether sunlight gets reflected in distracting ways inside and modify the position or size of windows accordingly. Nowadays, customers can assemble their furniture in virtual reality, allowing them to adjust the color or material of the furniture seamlessly. Training neural networks for autonomous driving with photorealistic synthetic images allows testing these systems for extreme weather conditions and rare catastrophic events such as accidents in traffic upfront. Moreover, computer-generated images provide immersive experiences in computer games and movies. The realism of which has now reached a level of fidelity such that most people can not differentiate real and computer-generated environments or actors anymore.

Unfortunately, creating photorealistic images of complex virtual environments can come at an enormous computational cost. A single frame of a contemporary movie can take hundreds of hours to compute on a single core of a modern CPU. Therefore content creators often use expensive workstations and “render farms,” high-performance computer clusters with thousands of CPUs, to render the final image sequence of a movie. This high computational demand is one of the reasons why there is still a great interest in the research of more efficient algorithms for photorealistic image synthesis. Improvements in the underlying algorithms allow artists and designers to create ever more complex and realistic environments, and it also improves the availability and viability of images synthesis tools and workflows.

On a high level, modern, high-quality rendering algorithms emulate the underlying physical process. Photon trajectories are followed in a virtual scene from a light source to a virtual camera or vice versa in a stochastic manner. A virtual film sensor accumulates the energy of photons arriving at the camera, similar to a film sensor in a real digital camera. To make this process more tractable, some simplifying assumptions that are common in computer graphics are made in this thesis. Amongst others, we neglect the finite speed of photons, i.e., our scenes are so small that the propagation of light can be assumed to be instantaneous. We also assume geometric optics and disregard several wave-optical effects such as polarization, and we ignore effects from inelastic scattering such as fluorescence. Additionally, we only consider electromagnetic energy in the visible spectrum with regards to a human observer. Unfortunately, despite these simplifications, image synthesis can still be computationally expensive because the geometric configuration, materials, and light sources of the virtual scene can be arbitrarily complex.

In mathematical terms, the process of physically-based image synthesis corresponds to solving an integral equation over the infinite-dimensional space of light paths, i.e., all possible photon trajectories from a light source to the camera. Currently, Monte Carlo methods are the primary method to solve this integral equation in practice. Monte Carlo methods express the solution of the integral equation as the expected value of a random variable. The expected value is numerically approximated by stochastic sampling of the integration domain. The underlying random sampling manifests itself in noise in the image, similar to the noise in digital images in low light conditions. For this reason, it might be necessary to accumulate the contribution of a large number of samples or light paths to compute an image such that a human observer is not distracted by the remaining noise. However, generating a random light path usually involves tens of thousands of arithmetic operations. Therefore, finding smart ways to create these light paths plays a crucial role in implementing practical algorithms.

A commonly used strategy to improve the creation of light paths is importance sampling, which reduces the variance of the expected value estimation. In practice, most forms of importance sampling use local information to improve random choices. For example, when a photon scatters at a certain point in the scene, its scattering direction is stochastically chosen. In this case, importance sampling means taking the material properties at the corresponding point into account to increase the likelihood of generating directions in which a large amount of light gets reflected. Some of the contributions of this thesis are novel local importance sampling techniques, for example, to improve the rendering of volumetric sources of light such as flames.

However, performing several informed local decisions does not necessarily result in a light path that is also valuable in a global sense. Such global importance sampling is a much harder problem and usually involves more complicated numerical integration frameworks like Markov chain, Hamiltonian, or Population Monte Carlo. Another approach to facilitate global importance sampling that has recently attracted much attention from researchers and practitioners alike is data-driven. Here, information gained during the numerical integration process is stored and used to influence and improve future random sampling decisions. Generally speaking, these methods learn the light distribution in a scene and increase the probability of making multiple successive local decisions with a globally good outcome. Related to this, we will introduce a novel data-driven method for learning the problematic parts of light transport in this thesis.

Two Monte Carlo-based algorithms that have found broad practical applications due to their relative simplicity are path tracing and many-light methods. This thesis presents multiple local and global importance sampling strategies for improving these methods. Path tracing and many-light rendering have fundamental strengths and weaknesses, which means that they have applications in different contexts. For example, handling wavelength-dependent effects such as dispersion with high accuracy can be achieved easily with path tracing by using spectral rendering. On the other hand, spectral rendering is unsuitable for many-light rendering for practical reasons. Many-light methods rely on the more commonly known tristimulus rendering to resolve color, for example, by performing computations for RGB colors. However, as we will discuss in this thesis, tristimulus and spectral rendering can not be exchanged arbitrarily even in simple settings without wavelength-dependent effects. Therefore, it is essential to understand the consequences of using either approach for light transport simulation. Many-light rendering has the benefit of producing noise-free images by sacrificing some amount of physical accuracy. These are some of the reasons why path tracing is the first choice in high fidelity contexts. In contrast, many-light methods are a suitable choice for interactive preview rendering where the goal is to produce images as accurately as possible in a short amount of time.

1.1 Original contributions

This thesis has several contributions in the context of local and global importance sampling for physically-based rendering with path tracing and many-light methods. As discussed earlier, many-light methods rely on tristimulus rendering for resolving color information. Path tracing is more flexible in this regard and can efficiently use spectral rendering. We start by discussing non-trivial consequences and intricacies of using spectral or tristimulus rendering in chapter 3. Afterward, we discuss our contributions regarding importance sampling in the context of many-light rendering in chapter 4 and for path tracing in chapter 5 and chapter 6.

In more detail, the contributions of this thesis are:

1.1.1 Physically meaningful image synthesis using tristimulus colors

This chapter is not about importance sampling, but it illustrates problems that can occur when computing light transport with tristimulus values instead of spectral rendering where every light path has a stochastically chosen real-valued wavelength associated with it. Some rendering algorithms, such as the many-light method in chapter 4 have to perform light transport calculation using tristimulus colors to be efficient. However, we will show that some tristimulus values can result in physically invalid spectra if interpreted as reflectances. This problem is highly dependent on the chosen color space of the tristimulus values and especially relevant for wide-gamut color spaces, which are increasingly common in consumer-grade monitors. We introduce and evaluate several methods to alleviate this problem by mapping tristimulus values to the subspace of valid reflectance spectra.

1.1.2 Placement of virtual point lights

Many-light methods are a form of Monte Carlo method that produces noise-free images with very few samples by introducing correlation. In their basic form, many-light methods place so-called virtual point light sources at the scattering locations of multiple light paths. Afterward, many-light methods compute the value of every pixel with the same set of virtual light sources. For efficiency, it is vital to place these light sources at locations where they contribute to the image. Chapter 4 introduces a data-driven approach to detect suitable locations for the virtual light sources in a preprocess. The information acquired in this preprocess is later reused by a new type of virtual point light that can handle scenes with glossy materials more efficiently. Reusing the information is crucial to amortize the overhead of the preprocessing step.

1.1.3 Robust rendering of emissive heterogeneous volumes

Heterogeneous emissive volumes such as flames are a very challenging type of light source for Monte Carlo rendering. Depending on the overall density, the appearance of an emissive volume can range from thin flames to thick explosions. Chapter 5 introduces a technique that robustly and efficiently handles this wide range of densities. To achieve this, we introduce an integration method that accumulates volumetric emission along a path segment up to a specific scattering event either inside or outside the volume. This segment emission combines the benefit of previous methods which accumulated emission either at scattering events, which is preferable for dense volumes, or along the path segment through the whole volume, which is preferable for thin volumes. We also introduce a novel method to combine this integration method with explicit importance sampling of the direct illumination, which is crucial for efficient rendering.

1.1.4 Global importance sampling using path guiding

Whereas chapter 5 discusses a local importance sampling technique for path tracing for a specific use-case, chapter 6 introduces a more general novel data-driven method for global importance sampling. During runtime, this technique identifies and stores light paths that a particular Monte Carlo sampler cannot create efficiently. With the information about problematic paths, it provides an additional sampling technique to create more of these paths specifically. The key benefits of this method are its selective focus on the challenging portion of light transport and the novel sampling technique, which takes information about multiple previously sampled paths into account when creating a new one. Using information of previously sampled paths allows the creation of a path sampling method that exhibits local exploration behavior similar to Markov chain Monte Carlo without its mathematical restrictions.

1.2 Outline of this thesis

After this introduction, chapter 2 gives an overview of the relevant theoretical foundations to understand modern Monte Carlo rendering. Chapter 3 will discuss an often overlooked issue with tristimulus rendering and how to address it. The following chapters are the central part of this thesis. Chapter 4 introduces a data-driven global importance sampling method for the placement of virtual point lights and a matching way of handling glossy surfaces more efficiently. Chapter 5 introduces a robust way to handle the whole range of thin to dense heterogeneous emissive volumes and provides a technique to importance sampling their direct illumination. Chapter 6 introduces a novel data-driven path sampling approach that allows global importance sampling. Finally, chapter 7 concludes this thesis's contributions in the context of local and global importance sampling and gives an outlook to possible future directions of research.

Remark

The author of this thesis changed his family name from “Simon” to “Reibold”. He was first author on the following publications: [Simon et al. 2015], [Simon et al. 2017], and [Reibold et al. 2018]. He was also joint-first author of [Meng et al. 2015].

2 Fundamentals of physically-based rendering

This chapter introduces the fundamentals of physically-based image synthesis, which is the basis of the contributions in this thesis. It discusses radiometry and radiative transport theory, which allow us to measure electromagnetic radiation and compute the equilibrium distribution of light in a virtual scene. It also recapitulates the concepts of probability theory, which are necessary to understand the essential properties of Monte Carlo rendering algorithms. Additionally, this chapter reviews most modern Monte Carlo rendering algorithms on a high level as well as differences between tristimulus and spectral rendering. The content of this introduction has to be concise, and for details, we refer to the more elaborate resources [Veach 1998; Dutre et al. 2006; Pharr et al. 2017; Marschner and Shirley 2015], which inspired large parts of this chapter.

2.1 Spaces, measures, and notation

Every scene we render consists of a set of two-dimensional surfaces $S_1, \ldots, S_n \subset \mathbb{R}^3$ [B. O'Neill 2006, Chapter 4] and we denote the union of all sets as $\mathcal{M}$. We assume all surfaces to be smooth and locally flat. Therefore, every surface point $\mathbf{x} \in \mathcal{M}$ has a unique normal $\mathbf{n}_\mathbf{x}$ and corresponding *tangent plane* with basis vectors $\mathbf{t}$ and $\mathbf{b}$ called *tangent* and *bitangent*, respectively. Figure 2.1 illustrates a point $\mathbf{x}$ on an exemplary surface in a global coordinate system on the left, and a local coordinate system on the right. The local coordinate system shown is called *tangent space*, and it is often more convenient to perform computations in this space than in global coordinates directly. Our convention is that the normal $\mathbf{n}_\mathbf{x}$ corresponds to the basis vector $\mathbf{e}_1 = (0, 1, 0)^T$ in *local coordinates* which we define as pointing "up" ($\mathbf{t}$ corresponds to $\mathbf{e}_0 = (1, 0, 0)^T$ and $\mathbf{b}$ corresponds to $\mathbf{e}_2 = (0, 0, 1)^T$).

The tangent plane divides the unit sphere $\mathcal{S}^2 = \{\mathbf{x} \in \mathbb{R}^3 : \|\mathbf{x}\| = 1\}$ around $\mathbf{x}$ into two disjoint subspaces

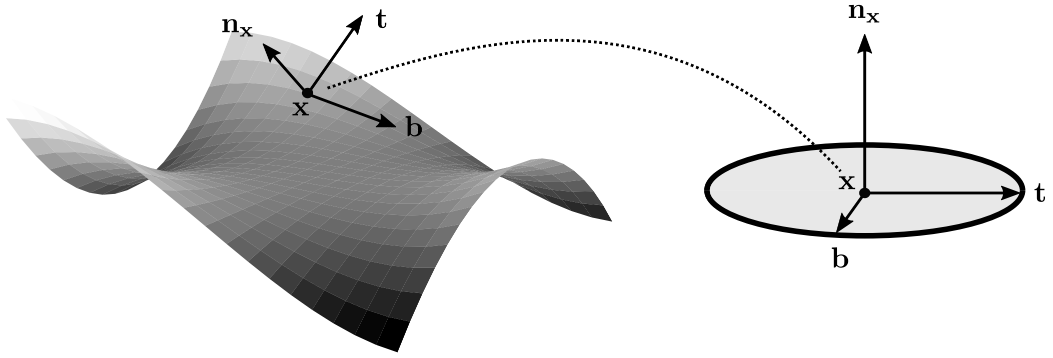


Figure 2.1: The two-dimensional surfaces we consider are assumed to be smooth and locally flat (left). Therefore, for every surface point $\mathbf{x}$, there exists a unique surface normal $\mathbf{n_x}$, and a corresponding tangent plane spanned by a tangent vector $\mathbf{t}$ and bitangent vector $\mathbf{b}$. The *tangent space* (right) is a local coordinate system in which it is more convenient to perform measurements (e.g., the total amount of incoming light) or define material properties. Our convention is that the normal corresponds to the basis vector $\mathbf{e}_1 = (0,1,0)^T$ pointing "up."

$$\mathcal{H}^+(\mathbf{x}) = \{\mathbf{y} \in \mathcal{S}^2 : \langle \mathbf{n_x}, \mathbf{y} \rangle > 0\} \text{ and} \tag{2.1}$$

$$\mathcal{H}^-(\mathbf{x}) = \{\mathbf{y} \in \mathcal{S}^2 : \langle \mathbf{n_x}, \mathbf{y} \rangle < 0\} \tag{2.2}$$

called the *upper* and *lower hemisphere*, respectively, as illustrated in figure 2.2.

For a watertight two-dimensional surface $\mathcal{S}$, i.e., a surface for which the notion of an *inside* and *outside* makes sense, we define $\mathbf{n_x}$ to be pointing outside. If the inside of $\mathcal{S}$ is unique, which means all points inside $\mathcal{S}$ are outside all other surfaces $\mathcal{M} \setminus \mathcal{S}$, it can have a refractive index that differs from its outside, which, for example, allows modeling of glass objects.

We further assume that length and area measurements are well defined on all surfaces. Let $A(S)$ denote the area of a surface $S \subseteq \mathcal{M}$ where A is the area measure or two-dimensional Lebesgue measure [Royden and Fitzpatrick 2010, Chapter 2], i.e.,

$$A(S) = \int_{\mathcal{M}} \mathbf{1}_S(\mathbf{x}) \, \mathrm{d}A(\mathbf{x}) = \int_S \mathrm{d}A(\mathbf{x}), \tag{2.3}$$

where $\mathbf{1}_S$ is the indicator function for the set S, i.e.,

$$\mathbf{1}_S(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in A \\ 0, & \text{otherwise} \end{cases} . \tag{2.4}$$

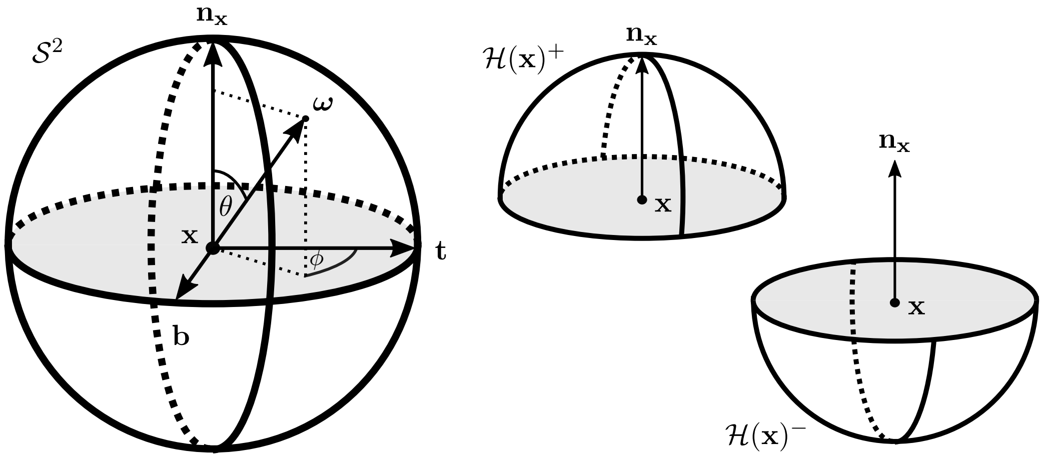


Figure 2.2: Every direction $\boldsymbol{\omega} = (\phi, \theta)$ corresponds to a point on the sphere $\mathcal{S}^2$ (right) which is divided by the tangent plane at $\mathbf{x}$ in two subsets called upper hemisphere $\mathcal{H}^+$ and lower hemisphere $\mathcal{H}^-$.

We assume all functions $f : \mathcal{M} \to \mathbb{R}$ considered in this thesis to be Lebesgue integrable, and denote the Lebesgue integral [Royden and Fitzpatrick 2010, Chapter 4] of f as

$$\int_S f(\mathbf{x})\, \mathrm{d}A(\mathbf{x}). \tag{2.5}$$

A direction $\omega \in \mathbb{R}^3$ is a vector of unit length, i.e., $\|\omega\| = 1$, and corresponds to a point on the unit sphere $\mathcal{S}^2$. A direction can also be represented by its spherical coordinates (ϕ, θ), where ϕ is the azimuthal, and θ is the longitudinal angle, as shown in figure 2.2.

We will often integrate spherical functions, i.e., functions of the form $f : \mathcal{S}^2 \to \mathbb{R}$, using a measure σ over directions called *solid angle measure*. The measure σ is the surface area measure on $\mathcal{S}^2$. A solid angle is the analog of an angle for a sphere instead of a circle [McCluney 2014, Chapter 1.3]. The solid angle of an object is defined as the surface area of its projection onto a unit sphere. The solid angle depends on the location of the sphere's center $\mathbf{x} \in \mathbb{R}^3$, and we say that an object subtends a solid angle of Ω with respect to $\mathbf{x}$. In simpler terms, the solid angle is a measure of how large an object appears to an observer looking at it from $\mathbf{x}$.

Another convenient measure of directions is the *projected solid angle measure* $\sigma^\perp$, which can be used to measure a set of directions $\Omega \subset H(\mathbf{x})^+$ or $\Omega \subset H(\mathbf{x})^-$ by projecting Ω onto the tangent plane at $\mathbf{x}$. To see why this measure is convenient, we will relate the surface area measure A, the solid angle measure σ and the projected solid angle measure $\sigma^\perp$ in the following.

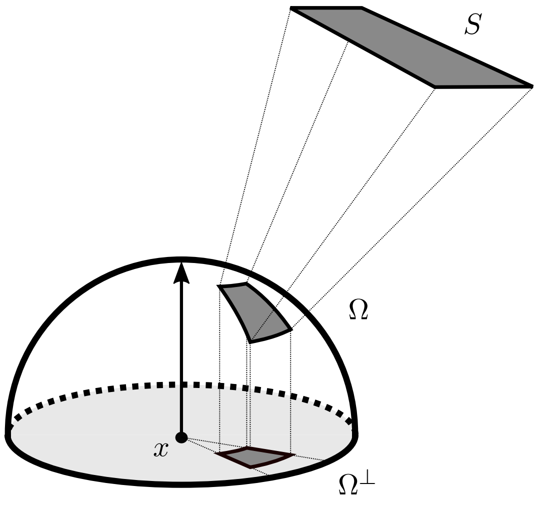


Figure 2.3: The solid angle Ω subtended by a surface S with respect to a point $\mathbf{x}$ is the projection of the surface onto the unit sphere around $\mathbf{x}$. The projected solid angle $\Omega^\perp$ is the projection of the solid angle Ω onto the unit disk in the tangent plane at $\mathbf{x}$.

For two points $\mathbf{x}, \mathbf{y} \in \mathcal{M}$, the notation $\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}$ is a shorthand for the direction from $\mathbf{x}$ to $\mathbf{y}$, i.e.,

$$\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}} = \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|}. \tag{2.6}$$

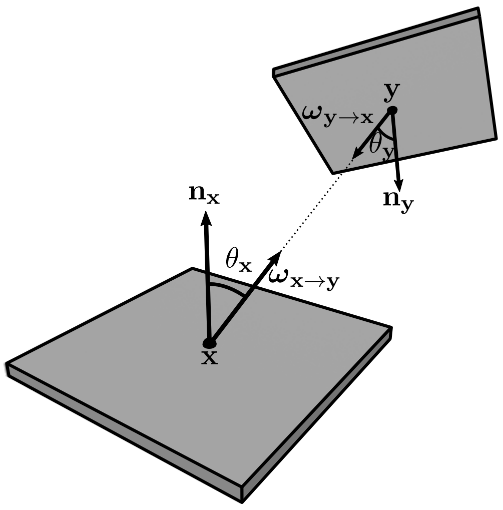
A *ray* starting at position $\mathbf{x}$ with direction $\boldsymbol{\omega}$ is the set of all points $\{\mathbf{x} + t\boldsymbol{\omega} \in \mathbb{R}^3 : t \in \mathbb{R}^+\}$, and a *ray cast* $\mathbf{y} = r(\mathbf{x}, \boldsymbol{\omega})$ determines the closest surface point $\mathbf{y} \in \mathcal{M}$ along the ray. A ray cast usually involves intersecting all surfaces with the ray and returning the closest intersection point. For a large number of surfaces, this operation can become very expensive, and a large body of research focuses on finding efficient acceleration structures to reduce the computation time for this ray cast operation [Glassner 1984; Havran 2000; Wald 2004; Wald et al. 2014]. The ray cast operation can be used to define a *visibility function*

$$V(\mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if } r(\mathbf{x}, \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}) = \mathbf{y} \\ 0, & \text{otherwise} \end{cases}, \tag{2.7}$$

which returns 1 if and only if two points $\mathbf{x}, \mathbf{y} \in \mathcal{M}$ are visible from each other.

The relashionship between the (projected) solid angle measure and the area measure is decribed by the *geometric term* between two points $\mathbf{x}, \mathbf{y} \in \mathcal{M}$:

$$G(\mathbf{x}, \mathbf{y}) = \frac{\langle \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}, \mathbf{n}_\mathbf{x} \rangle \langle \boldsymbol{\omega}_{\mathbf{y}\to\mathbf{x}}, \mathbf{n}_\mathbf{y} \rangle}{\|\mathbf{x} - \mathbf{y}\|^2} = \frac{\cos\theta_\mathbf{x} \cos\theta_\mathbf{y}}{\|\mathbf{x} - \mathbf{y}\|^2}. \tag{2.8}$$



$\mathbf{y}$
$\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{x}}$
$\theta_{\mathbf{y}}$
$\mathbf{n}_{\mathbf{y}}$
$\mathbf{n}_{\mathbf{x}}$
$\theta_{\mathbf{x}}$
$\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}$
$\mathbf{x}$

Figure 2.4: The geometric term G between $\mathbf{x}$ and $\mathbf{y}$ depends on the inverse squared distance of the points and the cosines $\cos\theta_{\mathbf{x}}$ and $\cos\theta_{\mathbf{y}}$ of the angle between the corresponding normals and directions between the points.

The geometric term is illustrated in figure 2.4, and the relashionship between the different measures is [McCluney 2014, Chapter 4.3]

$$\mathrm{d}\sigma^{\perp}(\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}) = \langle \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}, \mathbf{n}_{\mathbf{x}} \rangle \,\mathrm{d}\sigma(\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}) = G(\mathbf{x},\mathbf{y})\,\mathrm{d}A(\mathbf{y}). \tag{2.9}$$

The projected solid angle $\Omega^{\perp}$ subtended by a surface $S \subset \mathcal{M}$ with respect to $\mathbf{x}$ can be computed in multiple ways. One option is to integrate over projected solid angle and to check if a ray in the respective directions hits the surface S. Another option is to directly integrate over the area of the surface S using the area measure. Using an indicator function

$$\mathbf{1}_S(\mathbf{x},\boldsymbol{\omega}) = \begin{cases} 1, & \text{if } r(\mathbf{x},\boldsymbol{\omega}) \in S, \\ 0, & \text{otherwise} \end{cases}, \tag{2.10}$$

that tells us whether a ray hits the surface, the projected solid angle can be computed as

$$\Omega^{\perp} = \int_{\mathcal{H}^+} \mathbf{1}_S(\mathbf{x},\boldsymbol{\omega})\,\mathrm{d}\sigma^{\perp}(\boldsymbol{\omega}) = \int_S G(\mathbf{x},\mathbf{y})\,\mathrm{d}A(\mathbf{y}). \tag{2.11}$$

In the following, we will shorten the notation of the differentials to improve readability and define

$$\mathrm{d}\boldsymbol{\omega}^{\perp} = \mathrm{d}\sigma^{\perp}(\boldsymbol{\omega}), \tag{2.12}$$

$$\mathrm{d}\boldsymbol{\omega} = \mathrm{d}\sigma(\boldsymbol{\omega}), \text{ and} \tag{2.13}$$

$$\mathrm{d}A = \mathrm{d}A(\mathbf{x}). \tag{2.14}$$

We will convert between (projected) solid angle and area measure quite often in this thesis. Working with the projected solid angle measure is convenient because the transformation to area measure is simply the geometric term G, which is easier to remember and work with because it contains both cosines.

2.2 Radiometry and radiative transfer

To generate realistic synthetic images, we have to measure how much light, in the form of photons, reaches a virtual sensor. *Radiometry* provides techniques and physical units to quantify such forms of electromagnetic radiation [McCluney 2014].

2.2.1 Radiant flux and radiance

The fundamental quantity of radiometry is *radiant energy* Q measured in Joules $[J]$ [McCluney 2014, Chapter 1.4]. In our case, this energy is comprised of a large number of photons. Each photon carries an energy that is inversely proportional to its wavelength. Radiant energy can, for example, describe the total energy of all photons inside a three-dimensional region at a given point in time, as illustrated in figure 2.5 (top, left). It could also be used to describe the amount of electromagnetic radiation that is received by the earth from the sun in one day. However, since this quantity depends on the duration of a measurement, a more flexible quantity that can be used to determine the radiant energy for arbitrary time intervals is *radiant flux*

$$\Phi = \Phi(t) = \frac{\mathrm{d}Q(t)}{\mathrm{d}t}\ [W], \tag{2.15}$$

measured in Joules per second or Watts $W = Js^{-1}$. We will drop the time dependency in the notation for simplicity and because we will always be interested in radiometric quantities at one point in time. With this, we could describe how much radiant energy a surface $S \subset \mathcal{M}$ receives per unit time (figure 2.5, top. right). The next more general

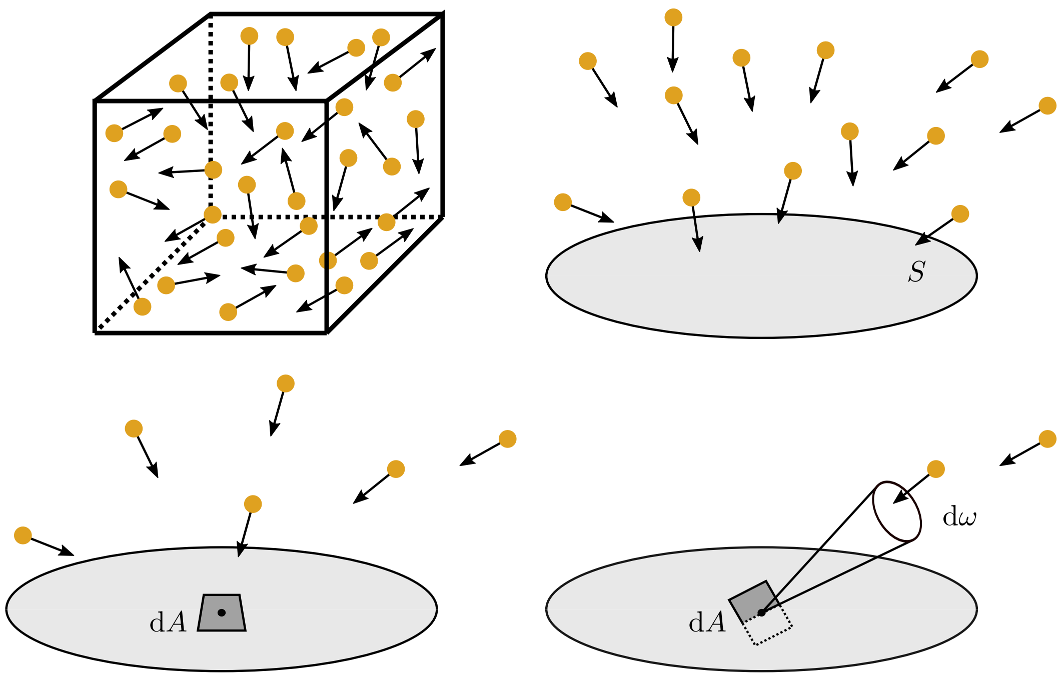


Figure 2.5: Here we illustrate the most important radiometric quantities for this thesis. *Radiant energy* can be used to describe the total energy of a collection of photons in a three-dimensional region (top, left). *Radiant flux* can be used to quantify the energy that passes through a surface S per unit time (top, right). *Irradiance* is the radiant flux incident at a differential surface patch $\mathrm{d}A$ (bottom, left). *Radiance* is the radiant flux incident at a differential surface patch $\mathrm{d}A$ from a differential solid angle $\mathrm{d}\boldsymbol{\omega}$ (bottom, right). Note that for radiance the differential area $\mathrm{d}A$ is perpendicular to the corresponding direction $\boldsymbol{\omega}$ wheres for irradiance, the differential area $\mathrm{d}A$ is aligned with the tangent plane.

quantity is *radiant flux density* or *irradiance* E, which is radiant flux per unit area incident at a point $\mathbf{x} \in \mathcal{M}$, i.e.,

$$E = E(\mathbf{x}) = \frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}A}\ [Wm^{-2}]. \tag{2.16}$$

The differential area $\mathrm{d}A$ corresponds to the area measure from equation 2.3. Irradiance is the incident flux at a differential surface patch $\mathrm{d}A$ over the whole hemisphere $\mathcal{H}^+(\mathbf{x})$ or $\mathcal{H}^-(\mathbf{x})$, as illustrated in figure 2.5 (bottom, left). It will be clear from the context which hemisphere we are considering, and otherwise denote the irradiance from the positive or negative hemisphere with E^+ and E^- respectively. A quantity that also takes the direction of radiant flux into account is *radiance*

$$L(\mathbf{x}, \boldsymbol{\omega}) = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \boldsymbol{\omega})}{\mathrm{d}A\,\mathrm{d}\boldsymbol{\omega}}\ [Wm^{-2}sr^{-1}]. \tag{2.17}$$

Here, the differential surface patch $\mathrm{d}A$ is perpendicular to $\boldsymbol{\omega}$ and $\mathrm{d}\boldsymbol{\omega}$ is the differential solid angle around $\boldsymbol{\omega}$ (see figure 2.5, bottom, right). Radiance is the fundamental quantity from which all others can be computed by integrating over time, directions, and surface (or volume). However, the radiance function is not continuous at surfaces. The radiance leaving a point in a direction is not the same as the radiance arriving at a point from the same direction. Therefore, the convention in this thesis is that $L(\mathbf{x}, \boldsymbol{\omega})$ is the radiance leaving $\mathbf{x}$ in direction $\boldsymbol{\omega}$ and $L_i(\mathbf{x}, \boldsymbol{\omega})$ will explicitly denote the *incident radiance.* To measure incident radiance at a differential surface patch dA which is not orthogonal to $\boldsymbol{\omega}$, we have to account for the foreshortening term $\cos\theta$, where θ is the angle between $\boldsymbol{\omega}$ and the corresponding surface normal, i.e.,

$$L_i(\mathbf{x}, \boldsymbol{\omega}) = \frac{\mathrm{d}E(\mathbf{x})}{\mathrm{d}\boldsymbol{\omega}^\perp} = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \boldsymbol{\omega})}{\mathrm{d}\boldsymbol{\omega}^\perp\,\mathrm{d}A} = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \boldsymbol{\omega})}{\mathrm{d}\boldsymbol{\omega}\cos\theta\,\mathrm{d}A}. \tag{2.18}$$

2.2.2 Surface reflection

If light hits the surface of an object, it can be reflected in complicated ways. This reflection interaction can be modeled with the *bidirectional reflection distribution function* (BRDF, [Nicodemus et al. 1977], [McCluney 2014, Chapter 6.6])

$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o), \tag{2.19}$$

which describes how much light incoming from $\boldsymbol{\omega}_i \in \mathcal{H}^+$ is reflected in direction $\boldsymbol{\omega}_o \in \mathcal{H}^+$ at a surface point $\mathbf{x}$. The total reflected radiance $L_r(\mathbf{x}, \boldsymbol{\omega}_o)$ usually depends on light incoming from all directions, and the differential portion due to light incident from a small solid angle $\mathrm{d}\boldsymbol{\omega}_i$ is proportional to the respective irradiance, i.e.,

$$\mathrm{d}L_r(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i) \propto \mathrm{d}E(\mathbf{x}, \boldsymbol{\omega}_i). \tag{2.20}$$

The bidirectional reflectance distribution function is simply defined as this proportionality constant, i.e.,

$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{\mathrm{d}L_r(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i)}{\mathrm{d}E(\mathbf{x}, \boldsymbol{\omega}_i)} = \frac{\mathrm{d}L_r(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i)}{L_i(\mathbf{x}, \boldsymbol{\omega}_i)\,\mathrm{d}\boldsymbol{\omega}_i^\perp}. \tag{2.21}$$

This definition leads naturally to the *reflection equation* that allows to compute the total reflected radiance by integrating over the incident radiance weighted by the BRDF, i.e.,

$$L_r(\mathbf{x}, \boldsymbol{\omega}_o) = \int_{\mathcal{H}^+} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) L_i(\mathbf{x}, \boldsymbol{\omega}_i)\,\mathrm{d}\boldsymbol{\omega}_i^\perp. \tag{2.22}$$

Measuring bidirectional reflectance distribution functions from real materials is time-consuming and data-intensive. For high-quality results, gonioreflectometers are usually used for these measurements [Murray-Coleman and A. Smith 1990; Ward et al. 1992]. Alternatively, BRDFs can be defined analytically using mathematical models with simplifying assumptions. An essential class of analytical BRDFs is based on microfacet theory [Walter et al. 2007; Heitz 2014]. To be physically plausible, BRDFs have to be *reciprocal*, i.e.,

$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = f_r(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i), \tag{2.23}$$

and ensure *energy conservation*, i.e.,

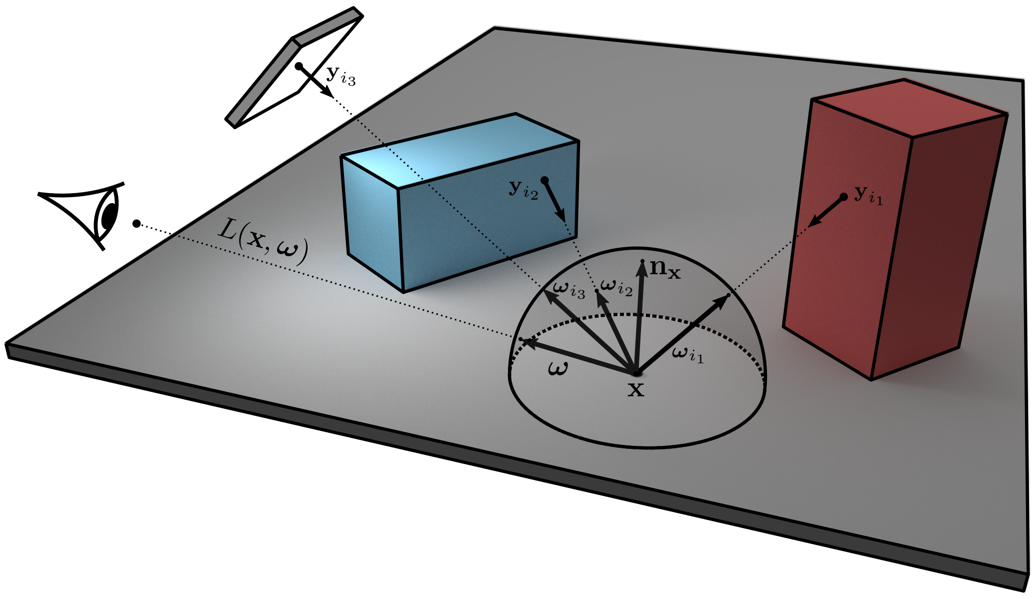
$$\int_{\mathcal{H}^+} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\,\mathrm{d}\boldsymbol{\omega}_i^\perp \leq 1 \quad \text{for all } \boldsymbol{\omega}_o \in \mathcal{H}^+. \tag{2.24}$$

Light can not only be reflected but also transmitted (e.g., glass). In this case, a *bidirectional transmittance distribution function* (BTDF) can be defined analogously to a BRDF, i.e.,

$$f_t(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{\mathrm{d}L_t(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i)}{\mathrm{d}E(\mathbf{x}, \boldsymbol{\omega}_i)} \tag{2.25}$$

where $\boldsymbol{\omega}_i \in \mathcal{H}^-$. Combining BRDF and BTDF results in the *bidirectional scattering distribution function* (BSDF), i.e.,

$$f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \begin{cases} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o), & \text{if } \boldsymbol{\omega}_i \text{ and } \boldsymbol{\omega}_o \text{ in same hemisphere} \\ f_t(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o), & \text{otherwise} \end{cases} \tag{2.26}$$



$\mathbf{y}_{i_3}$
$\mathbf{y}_{i_2}$
$\mathbf{y}_{i_1}$
$L(\mathbf{x}, \boldsymbol{\omega})$
$\mathbf{n}_\mathbf{x}$
$\boldsymbol{\omega}_{i_3}$
$\boldsymbol{\omega}_{i_2}$
$\boldsymbol{\omega}_{i_1}$
$\boldsymbol{\omega}$
$\mathbf{x}$

Figure 2.6: The *rendering equation* (equation 2.27) states that the outgoing radiance $L(\mathbf{x}, \boldsymbol{\omega})$ at a surface point $\mathbf{x}$ in direction $\boldsymbol{\omega}$ can be computed by integrating the incident radiance $L_i(\mathbf{x}, \boldsymbol{\omega}_i)$ over all directions $\boldsymbol{\omega}_i \in \mathcal{S}^2$ multiplied by the BRDF $f(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega})$. The incident radiance $L_i(\mathbf{x}, \boldsymbol{\omega}_i)$ depends recursively on the outgoing radiance $L(\mathbf{y}_i, -\boldsymbol{\omega}_i)$ where $\mathbf{y}_i = r(\mathbf{x}, \boldsymbol{\omega}_i)$ is the closest surface point hit by a ray cast from $\mathbf{x}$ in direction $\boldsymbol{\omega}_i$.

2.2.3 Rendering equation

In addition to scattering, the outgoing radiance can also be influenced by emission at the surface directly. The directional emission function $L_e(\mathbf{x}, \boldsymbol{\omega})$ describes radiance emitted in direction $\boldsymbol{\omega}$ at a surface point $\mathbf{x}$. With emission and scattering, we can formulate the *rendering equation* [Kajiya 1986] that describes the light transport in a vacuum, i.e.,

$$L(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}) L_i(\mathbf{x}, \boldsymbol{\omega}_i) \,\mathrm{d}\boldsymbol{\omega}_i^\perp. \tag{2.27}$$

As illustrated in figure 2.6, the incident radiance $L_i(\mathbf{x}, \boldsymbol{\omega}_i)$ recursively depends on the outgoing radiance $L(\mathbf{y}_i, -\boldsymbol{\omega}_i)$. Here $\mathbf{y}_i = r(\mathbf{x}, \boldsymbol{\omega}_i)$ is the closest surface point intersected by a ray with origin $\mathbf{x}$ and direction $\boldsymbol{\omega}_i$. We can also formulate the rendering equation directly over surface points using the visibility function $V(\mathbf{x}, \mathbf{y})$ and geometric term $G(\mathbf{x}, \mathbf{y})$ defined in equation 2.7 and equation 2.8 as

$$L(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{\mathcal{M}} f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}, \boldsymbol{\omega}) L(\mathbf{y}, \boldsymbol{\omega}_{\mathbf{y}\to\mathbf{x}}) V(\mathbf{x}, \mathbf{y}) G(\mathbf{x}, \mathbf{y}) \,\mathrm{d}A(\mathbf{y}). \tag{2.28}$$

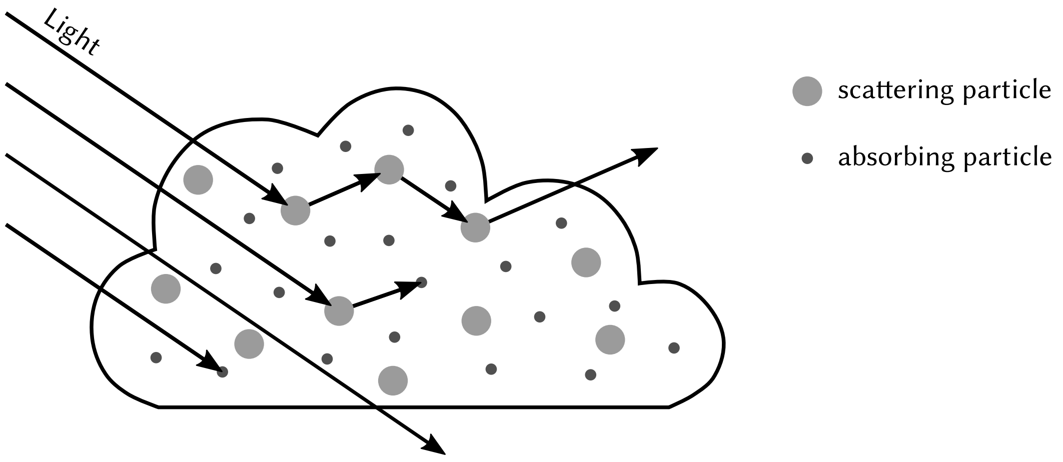


Figure 2.7: Participating media consist of particles that can interact with light traveling through the medium. These interactions can be *absorption* or *scattering*. Absorption means photons are taken up by the particles, and their energy is transformed into another form, for example, thermal energy. Scattering means photons are absorbed and re-emitted in a different direction. We only consider *elastic scattering* for which scattering does not affect the wavelength of the photon.

2.2.4 Participating media

The rendering equation fully describes light transport in a vacuum. However, many natural phenomena, like clouds or translucent fluids such as milk, are due to a participating medium, which fills the space between surfaces. In such a medium, light can interact with the particles it consists of, and get absorbed or scattered as illustrated in figure 2.7. We model such media with a statistical distribution of particles that interact with photons either by absorption or scattering. The overall density of these particles is denoted by $\rho(\mathbf{x})$ $[m^{-3}]$. The density of particles can vary spatially. Absorbing and scattering particles are assumed to be spatially independent and are characterized by their respective cross-sections σ_a $[m^2]$ and σ_s $[m^2]$. The overall absorption and scattering coefficients are determined by multiplying the cross-section with the density, i.e.,

$$\mu_a(\mathbf{x}) = \rho(\mathbf{x})\sigma_a \; [m^{-1}] \tag{2.29}$$

$$\mu_s(\mathbf{x}) = \rho(\mathbf{x})\sigma_s \; [m^{-1}] \tag{2.30}$$

These quantities determine the probability density of a photon undergoing an absorption or scattering event per unit distance traveled in the medium. The *extinction coefficient*

$$\mu_t = \mu_a + \mu_s \tag{2.31}$$

determines the probability density of a photon undergoing either type of event and the ratio

$$\alpha = \frac{\mu_s}{\mu_t}, \tag{2.32}$$

often called the *single-scattering albedo*, denotes the probability that a collision event results in the scattering of a photon. Emissive participating media can be modeled by a concentration of emissive particles μ_e and their respective cross-section σ_e. In practice, the spatially varying density values are given procedurally in analytic form [Perlin 1985; Perlin and Hoffert 1989; Ebert and Musgrave 2003] or as 3D grid consisting of finite volume elements (voxels), which is usually represented using an efficient hierarchical data structure [Museth 2013].

In this thesis, we will only consider *elastic scattering*, which means a photon can change its direction upon scattering but not its energy. The distribution of scattering directions is determined by the *phase function* f_p, which is the equivalent of a BSDF for participating media. A widely used phase function is the *Henyey-Greenstein phase function* [Henyey and Greenstein 1941],

$$f_p(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 + 2g(\langle \boldsymbol{\omega}_i, \boldsymbol{\omega}_o \rangle))^{3/2}} \tag{2.33}$$

where the asymmetry parameter $g \in [-1, 1]$ controls the shape of the scattering function. We will assume that the asymmetry parameter is constant over the whole volume. For $g < 0$, the medium is forward scattering, for $g = 0$, the scattering is isotropic, and for $g < 0$, it is backscattering.

2.2.5 Radiative transfer equation

The radiance along a ray through a volume can change due to four different processes. Radiance decreasing out-scattering and absorption, and radiance increasing in-scattering and emission. These processes are illustrated in figure 2.8. The *radiative transfer equation* (RTE) [Chandrasekar 1960; Kajiya and Von Herzen 1984],

$$(\boldsymbol{\omega} \cdot \nabla) L(\mathbf{x}, \boldsymbol{\omega}) = \mu_e L_e(\mathbf{x}, \boldsymbol{\omega}) + \mu_s L_s(\mathbf{x}, \boldsymbol{\omega}) - (\mu_a + \mu_s) L(\mathbf{x}, \boldsymbol{\omega}), \tag{2.34}$$

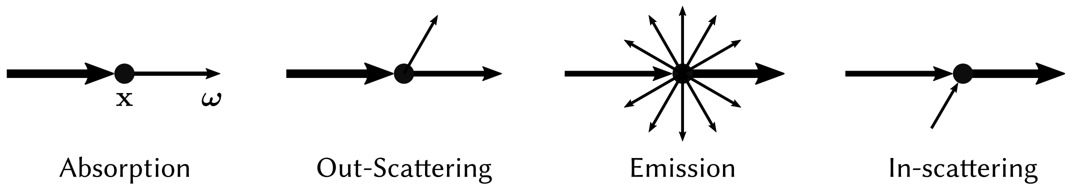


Figure 2.8: The radiance along a ray with direction $\boldsymbol{\omega}$ can change due to four different scattering effects. At each point $\mathbf{x}$ along the ray, the radiance can decrease due to absorption or out-scattering, and it can increase due to emission or in-scattering.

describes the change of radiance along a ray through a differential volume location $\mathbf{x}$ in direction $\boldsymbol{\omega}$. Here, L_s is the in-scattered radiance

$$L_s(\mathbf{x}, \boldsymbol{\omega}) = \int_{S^2} f_p(\boldsymbol{\omega}_i, \boldsymbol{\omega}) L_i(\mathbf{x}, \boldsymbol{\omega}_i) \, \mathrm{d}\boldsymbol{\omega}_i. \tag{2.35}$$

Please note that the scattering coefficients μ_a, μ_s, and μ_e implicitly depend on the location $\mathbf{x}$. If we ignore scattering (i.e., $\sigma_a = 0$), equation 2.34 simplifies to

$$(\boldsymbol{\omega} \cdot \nabla) L(\mathbf{x}, \boldsymbol{\omega}) = \mu_e L_e(\mathbf{x}, \boldsymbol{\omega}) - \mu_t L(\mathbf{x}, \boldsymbol{\omega}) \tag{2.36}$$

which is a standard first-order differential equation with the solution

$$L(\mathbf{x}, \boldsymbol{\omega}) = L(\mathbf{x}', \boldsymbol{\omega}) e^{-\int_0^t \mu_t(s) \, \mathrm{d}s} + \int_0^t \mu_e(s) L_e(s) e^{-\int_0^s \mu_t(s') \, \mathrm{d}s'} \, \mathrm{d}s. \tag{2.37}$$

Here, L_e, μ_e, and μ_t are reparameterized (i.e., $L_e(s) = L_e(\mathbf{x} - s\boldsymbol{\omega})$), and $L(\mathbf{x}', \boldsymbol{\omega})$ is the radiance at the surface geometry, which serves as a boundary condition. The *transmittance* term

$$T(t) = e^{-\int_0^t \mu_t(s) \, \mathrm{d}s} \tag{2.38}$$

corresponds to the loss of light along a distance t in the medium due to extinction. The transmittance allows computing the fraction of radiance that is not scattered or absorbed when traveling between two points, i.e.,

$$T(\mathbf{x}, \mathbf{y}) = e^{-\tau(\mathbf{x}, \mathbf{y})}, \tag{2.39}$$

$$\tau(\mathbf{x}, \mathbf{y}) = \int_0^d [\mu_a + \mu_s](\mathbf{x} - s\boldsymbol{\omega}) \, \mathrm{d}s = \int_0^d \mu_t(\mathbf{x} - s\boldsymbol{\omega}) \, \mathrm{d}s, \tag{2.40}$$

where $d = \|\mathbf{x} - \mathbf{y}\|$, $\boldsymbol{\omega} = \boldsymbol{\omega}_{\mathbf{y} \to \mathbf{x}}$. The function τ is called *optical depth* or *optical thickness.*

In the general case (i.e., $\sigma_s \neq 0$), we get a similar albeit more complicated formula for computing the radiance $L(\mathbf{x}, \boldsymbol{\omega})$ by integrating the emitted and in-scattered radiance weighted by the transmittance along the ray $r(t) = \mathbf{x} - t\boldsymbol{\omega}$, i.e.,

$$L(\mathbf{x}, \boldsymbol{\omega}) = \int_0^\infty T(\mathbf{x}, \mathbf{x} - \boldsymbol{\omega}t)\ [\mu_e L_e + \mu_s L_s](\mathbf{x} - \boldsymbol{\omega}t, \boldsymbol{\omega})\,\mathrm{d}t, \tag{2.41}$$

In the presence of surfaces, the rendering equation (equation 2.27) serves as a boundary condition to the RTE. If a ray $r(t) = \mathbf{x} - t\boldsymbol{\omega}$ intersects a surface at location $\mathbf{x}' \in \mathcal{M}$, we compute the radiance $L(\mathbf{x}, \boldsymbol{\omega})$ as

$$\begin{aligned} L(\mathbf{x}, \boldsymbol{\omega}) &= T(\mathbf{x}, \mathbf{x}')L(\mathbf{x}', \boldsymbol{\omega}) && (2.42)\\ &+ \int_0^{\|\mathbf{x}-\mathbf{x}'\|} T(\mathbf{x}, \mathbf{x} - \boldsymbol{\omega}t)\ [\mu_e L_e + \mu_s L_s](\mathbf{x} - \boldsymbol{\omega}t, \boldsymbol{\omega})\,\mathrm{d}t. && (2.43) \end{aligned}$$

In this thesis, we assume that the volume is isotropic in the sense that the particle cross-sections do not depend on the direction of propagation. We also assume the particle distribution to be uncorrelated and that the index of refraction only changes at the interface of the volume. Methods that generalize volumetric transport and lift these assumptions can be found in [Jakob et al. 2010; Ament et al. 2014; Jarabo et al. 2018; Bitterli et al. 2018]. A much more detailed description of volume rendering and the radiative transport equation can be found in [Hege et al. 1996].

2.2.6 Path integral formulation

The radiative transfer equation can be recursively expanded into a more explicit form that provides an intuitive description of how light transport can be calculated [Veach 1998]. This formulation states that the measurement of a pixel sensor j can be computed by integrating the differential flux transported over every light transport path that connects this sensor to the light sources, i.e.,

$$I_j = \int_{\mathcal{P}} f_j(\mathbf{X})\mu(\mathbf{X}). \tag{2.44}$$

The integration domain $\mathcal{P}$ is called *path space* and is the set of all paths $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ of arbitrary length $k \in \mathbb{N}$. The measure $\mu(\mathbf{X})$ is called *path space measure*, and the integrant f_j is called the *measurement contribution.* These terms are defined in the following.

Let $\mathcal{P}_k$ be the set of all light paths $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ of length k. Then, path space is the infinite union

$$\mathcal{P} = \bigcup_{k=2}^{\infty} \mathcal{P}_k. \tag{2.45}$$

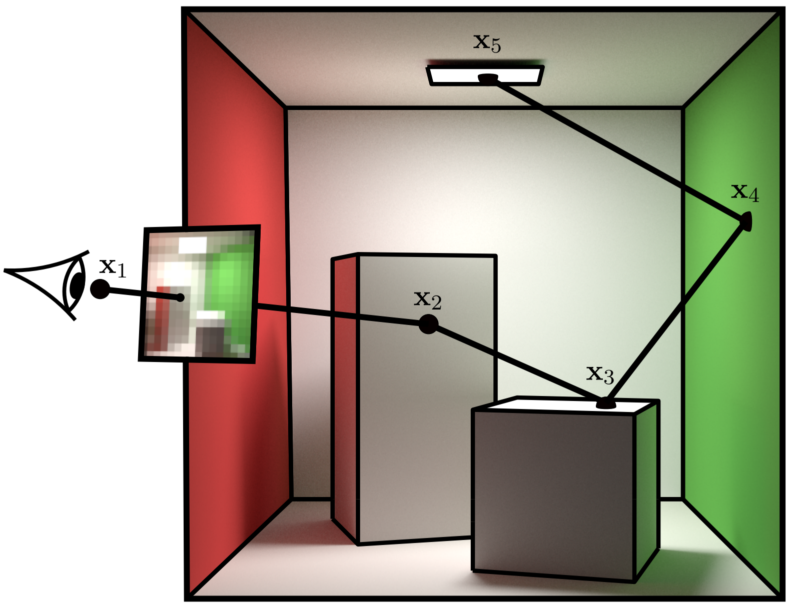


Figure 2.9: An exemplary path of length 5.

We ignore paths of length $k = 0$ and $k = 1$ since they do not contribute to the light transport in a scene. For every vertex $\mathbf{x}_i$ along a path, we define a corresponding *vertex measure* $\mathrm{d}\mathbf{x}_i$. If the vertex is a surface vertex, the corresponding vertex measure equals the surface area measure. Otherwise, the vertex measure equals the volume measure. For each set $\mathcal{P}_k$, we then define a measure μ_k as the product of all vertex measures $\mathrm{d}\mathbf{x}_i$, i.e., for a set of paths $\mathcal{D}_k \subset \mathcal{P}_k$ we define

$$\mu_k(\mathcal{D}_k) = \int_{\mathcal{D}_k} \mathrm{d}\mathbf{x}_1 \cdots \mathrm{d}\mathbf{x}_k. \tag{2.46}$$

Since all $\mathcal{P}_k$ are disjoint, we can define the path space measure μ as

$$\mu(\mathcal{D}) = \sum_{k=2}^{\infty} \mu_k(\mathcal{D} \cap \mathcal{P}_k), \tag{2.47}$$

where $\mathcal{D} \subset \mathcal{P}$ is a set of paths of arbitrary length.

The sensitivity of a sensor to arriving light can depend on the location on the sensor and the incident direction. Such effects can be modeled with a sensor sensitivity function $W_j(\mathbf{x}, \boldsymbol{\omega})$, which describes the sensitivity of the sensor to light arriving at $\mathbf{x}$ from direction $\boldsymbol{\omega}$. To define the measurement contribution f_j, we define a scattering function f and geometric term G for path vertices $\mathbf{x}, \mathbf{y}, and \mathbf{z}$, which can either be surface or volume vertices as

$$f(\mathbf{x},\mathbf{y},\mathbf{z}) = \begin{cases} f_s(\mathbf{y}, \boldsymbol{\omega}_{\mathbf{y}\to\mathbf{z}}, \boldsymbol{\omega}_{\mathbf{y}\to\mathbf{x}}), & \text{if } \mathbf{y} \text{ on surface} \\ \mu_s(\mathbf{y}) f_p(\mathbf{y}, \boldsymbol{\omega}_{\mathbf{y}\to\mathbf{z}}, \boldsymbol{\omega}_{\mathbf{y}\to\mathbf{y}}), & \text{if } \mathbf{y} \text{ in volume} \end{cases} \tag{2.48}$$

$$D(\mathbf{x},\mathbf{y}) = \begin{cases} \langle \mathbf{n}_\mathbf{x}, \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}} \rangle, & \text{if } \mathbf{x} \text{ on surface} \\ 1, & \text{if } \mathbf{x} \text{ in volume} \end{cases} \tag{2.49}$$

$$G(\mathbf{x},\mathbf{y}) = \frac{D(\mathbf{x},\mathbf{y})D(\mathbf{y},\mathbf{x})}{\|\mathbf{x}-\mathbf{y}\|^2}. \tag{2.50}$$

Now, for a path $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$, the measurement contribution function is

$$f_j(\mathbf{X}) = W_j(\mathbf{x}_1, \boldsymbol{\omega}_{\mathbf{x}_2\to\mathbf{x}_1}) G(\mathbf{X}) T(\mathbf{X}) f(\mathbf{X}) L_e(\mathbf{x}_k, \boldsymbol{\omega}_{\mathbf{x}_k\to\mathbf{x}_{k-1}}), \tag{2.51}$$

where T, G and f are the products of all geometric terms, transmittances, and scattering distribution functions of all path segments or inner path vertices respectively, i.e.,

$$T(\mathbf{X}) = \prod_{i=1}^{k-1} T(\mathbf{x}_i, \mathbf{x}_{i+1}), \tag{2.52}$$

$$G(\mathbf{X}) = \prod_{i=1}^{k-1} G(\mathbf{x}_i, \mathbf{x}_{i+1}), \text{ and} \tag{2.53}$$

$$f(\mathbf{X}) = \prod_{i=2}^{k-1} f(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}). \tag{2.54}$$

2.3 Probability theory and Monte Carlo integration

In this section, we will recapitulate basic concepts of probability theory [Grimmett and Welsh 2014], give an introduction to the Monte Carlo method [Metropolis and Ulam 1949] and a variance reduction technique called (multiple) importance sampling [Veach and Guibas 1995].

2.3.1 Random variables and the probability density function

Random experiments can be modeled with a *probability triple* $(\Omega, \mathcal{F}, P)$ which consists of a *sample space* Ω, a σ-Algebra $\mathcal{F}$ over Ω, and a *probability measure* P [Grimmett and Welsh 2014, Chapter 1]. The sample space Ω is the set of all possible outcomes of the experiment, and P assigns a probability to each element in $\mathcal{F}$.

A measurable function $X : \Omega \to D$, which maps random outcomes to a measurable set D, is called a *random variable*. In this thesis, we are only interested in *discrete random variables* where $D \subseteq \mathbb{Z}$ or *continuous random variables* where $D \subseteq \mathbb{R}$. If $f : D \to \mathbb{R}$ is Lebesgue measurable, then $Y := f(X)$ is also a random variable.

A discrete random variable X can be characterized by a probability mass function (PMF), which assigns every outcome $x \in D$ of the random experiment a probability

$$\begin{aligned} p &: D \to [0,1], \\ p(x) &= P(X = x) = P(\{\omega \in \Omega : X(\omega) = x\}), \end{aligned} \tag{2.55}$$

and fulfills the properties

$$p(x) \geq 0 \; \forall x \in D \text{ and } \sum_D p(x) = 1. \tag{2.56}$$

For example, X could model the throw of a fair dice with $D = \{1, \dots, 6\}$, and the PMF would be constant, i.e., $p(x) = P(X = x) = 1/6$ for all $x \in D$.

Continuous random variables have no PMF since the probability mass of every individual $x \in D \subseteq \mathbb{R}$ must be zero. Instead, continuous random variables are characterized using a cumulative distribution function (CDF)

$$\begin{aligned} F &: D \to [0,1], \\ F(x) &= P(X \leq x) = P(\{\omega \in \Omega : X(\omega) \leq x\}). \end{aligned} \tag{2.57}$$

or its corresponding probability density function (PDF)

$$\begin{aligned} p &: D \to [0,1], \\ p(x) &= \frac{\mathrm{d}F}{\mathrm{d}x}(x). \end{aligned} \tag{2.58}$$

The probability density function has to fulfill the following properties:

$$p(x) \geq 0 \; \forall x \in D \text{ and } \int_D p(x) \,\mathrm{d}x = 1. \tag{2.59}$$

In the following, we will consider only probability density functions. The properties of random variables we discuss also hold for discrete random variables by straightforwardly replacing the PDF with a PMF and integration by summation.

From now on, we will characterize a random variable X by its PDF p and write $X \sim p$, which means that the random realizations or *samples* $x_1, \dots x_n \in D$ of X are distributed according to p. In this case, we will write $x_i \sim p$ or $x_i \sim X$ synonymously. We are usually not interested in which $\omega_i \in \Omega$ resulted in $X(\omega_i) = x_i$ and assume that the underlying probability triple was chosen accordingly.

2.3.2 Conditional and marginal distributions and independence

Two random variables $X, Y : \Omega \to D$ have a *joint probability density function*

$$p_{XY}(x, y) = \frac{\partial^2}{\partial x \partial y} F_{XY}(x, y), \tag{2.60}$$

where

$$F_{XY}(x, y) = P(X \leq x \text{ and } Y \leq y) \tag{2.61}$$

is the *joint cumulative distribution function.* Two important concepts are the *marginal distribution*

$$p_X(x) = \int_D p_{XY}(x, y) \, \mathrm{d}y, \tag{2.62}$$

and the *conditional distribution*

$$p_{X|Y}(x|y) = \frac{p_{XY}(x, y)}{p_Y(y)}. \tag{2.63}$$

The two random variables X and Y are said to be *independent random variables* if their respective conditional distribution equals the marginal distribution, i.e., if $p_{X|Y}(x|y) = p_X(x)$ and $p_{Y|X}(y|x) = p_Y(y)$. For discrete random variables, this means that the probability of an occurring event (x, y) is the product of the individual events, i.e.,

$$P(X = x \text{ and } Y = y) = P(X = x) \cdot P(Y = y). \tag{2.64}$$

2.3.3 Inversion method and rejection sampling

Randomness in a computer implementation is usually accomplished using *pseudo-random number generators* (RNG). Pseudo-random number generators with excellent statistical properties, and the ones we use in our implementations, are Mersenne twister [Matsumoto and Nishimura 1998] and PCG generators [M. E. O'Neill 2014]. Most RNGs implement a random variable with *uniform distribution* in $[0, 1)$ also denoted as $U[0, 1]$, with PDF $p_{U[0,1]}(x) = 1 \; \forall x \in [0, 1)$. In the following, we use this uniform distribution as a building block to construct random variables with more complex distributions.

We can create a random variable $X \sim p$ if the corresponding CDF F_X is invertible using the *inversion method* [Devroye 1986]. The inversion method uses a uniformly distributed random variable $Y \sim U[0,1]$ and transforms it using the inverse of F_X, i.e.,

$$X := F^{-1}(Y) \sim p. \tag{2.65}$$

If the CDF F_X is not invertible, we can resort to a method called *rejection sampling* [Devroye 1986]. Suppose we can create an intermediate distribution p_z, for example with the inversion method, and we can bound p with some $k \in \mathbb{R}$ such that

$$p(x) < k \cdot p_z(x) \; \forall x \in D. \tag{2.66}$$

Then repeatedly sampling $x \sim p_z$ and $u \sim U[0,1]$ until

$$u < \frac{p(x)}{k \cdot p_z(x)} \tag{2.67}$$

results in samples $x \sim p$. This algorithm has unbounded runtime and is given in algorithm 1. The number of rejections depends on how well the intermediate distribution p_z approximates p. The better p_z approximates p, and the tighter the bound k is, the faster the rejection sampling procedure terminates.

Algorithm 1 Rejection Sampling

```
1: procedure RejectionSampling($p$, $p_z$, $k$)
2:     $x \leftarrow 0$
3:     do
4:         sample $x \sim p_z$
5:         sample $u \sim U[0,1]$
6:     while $u \cdot k \cdot p_z(x) > p(x)$
7:     return $x$
8: end procedure
```

2.3.4 Expected value and law of large numbers

A random variable X is a measurable function with underlying measure P and its *expected value* is

$$\mathbb{E}(X) = \int_\Omega X(\omega)\,\mathrm{d}P(\omega). \tag{2.68}$$

For a discrete random variable $X : \Omega \to \mathbb{Z}$ or a continuous random variable $Y : \Omega \to \mathbb{R}$, the expected value can be computed using the corresponding PMF p_X or PDF p_Y as

$$\mathbb{E}(X) = \sum_{\mathbb{Z}} x p_X(x) \tag{2.69}$$

$$\mathbb{E}(Y) = \int_{\mathbb{R}} x p_Y(x) \, \mathrm{d}x. \tag{2.70}$$

We will assume that $\mathbb{E}(X)$ always exists, i.e., $\mathbb{E}(X) < \infty$. The rest of this section focuses on a continuous random variable $X : \Omega \to D \subseteq \mathbb{R}$ with $X \sim p$. Analytical computation of the expected value $\mathbb{E}(X)$ can be tedious or impossible. In such cases, we can estimate the expected value using the mean of many random samples $x_1, \dots, x_n$, i.e.,

$$\mathbb{E}(X) \approx \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{2.71}$$

This estimation is justified by the (strong) law of large numbers [Grimmett and Welsh 2014, Chapter 8]. This law states that for an infinite sequence of independent random variables $X_1, X_2, \dots \; p$, and a random variable

$$\bar{X}_n := \frac{1}{n} \sum_{i=1}^{n} X_i, \tag{2.72}$$

the mean value $\bar{X}_n$ will converge to $\mathbb{E}(X)$ *in probability*, i.e.,

$$P\Big(\lim_{n\to\infty} \bar{X}_n = \mathbb{E}(X)\Big) = 1. \tag{2.73}$$

2.3.5 Variance and Chebyshev's inequality

A natural question to ask is how many samples have to be drawn until the estimate $\bar{X}_n$ is a good approximation for $\mathbb{E}(X)$ with a high probability. For example, we could ask how many samples are necessary, such that the probability of $\bar{X}_n$ differing 1% or more from the expected value $\mathbb{E}(X)$ is at most 1%, i.e.,

$$P\left(\left|\bar{X}_n - \mathbb{E}(X)\right| \geq \frac{\mathbb{E}(X)}{100}\right) \leq \frac{1}{100}. \tag{2.74}$$

The answer to this question depends on the *variance* $\sigma^2 := \mathrm{Var}(X)$ of the random variable which is its expected squared deviation from its mean $\mu := \mathbb{E}(X)$, i.e.,

$$\sigma^2 = \mathrm{Var}(X) = \mathbb{E}\big((X-\mu)^2\big) = \int_D (x-\mu)^2 p(x)\,\mathrm{d}x. \tag{2.75}$$

As with the expected value, we assume that the variance always exists, i.e., $\sigma^2 < \infty$. We know from fundamental properties of the expected value (linearity) and variance that the expected value of $\bar{X}_n$ is equal to the expected value μ of X since

$$\mathbb{E}\big(\bar{X}_n\big) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n}\sum_{i=0}^{n}\underbrace{\mathbb{E}(X_i)}_{=\mu} = \frac{1}{n}\sum_{i=0}^{n}\mu = \mu, \tag{2.76}$$

and the variance decreases linearly with n because

$$\mathrm{Var}\big(\bar{X}_n\big) = \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\sum_{i=0}^{n}\underbrace{\mathrm{Var}(X_i)}_{=\sigma^2} = \frac{\sigma^2}{n}. \tag{2.77}$$

Using *Chebyshev's inequality* [Grimmett and Welsh 2014, Chapter 8], which states that

$$P(|X-\mu| \geq a) \leq \frac{\mathrm{Var}(X)}{a^2} \tag{2.78}$$

for any value $a \in \mathbb{R}^+$, we can determine the requirement for equation 2.74 from above since

$$P\Big(\big|\bar{X}_n - \mu\big| \geq \frac{\mu}{100}\Big) \leq \frac{1}{100} \Leftrightarrow \frac{100^2 \cdot \mathrm{Var}\big(\bar{X}_n\big)}{\mu^2} \leq \frac{1}{100} \Leftrightarrow \frac{\sigma^2}{n} \leq \frac{\mu^2}{100}. \tag{2.79}$$

Therefore, the number of samples n has to be at least $100 \cdot \sigma^2/\mu^2$, and this number increases linearly with the variance of the random variable.

2.3.6 Monte Carlo integration

The approximation of the expected value of a random variable as the mean of many samples can be used to integrate any measurable function $f : D \to \mathbb{R}$ numerically. This approximation is called *Monte Carlo Method* or *Monte Carlo Integration* [Metropolis and Ulam 1949], and it only requires a probability density function

$$\begin{gathered} p : D \to \mathbb{R} \text{ with} \\ p(x) > 0 \text{ whenever } f(x) > 0. \end{gathered} \tag{2.80}$$

The integral value I can be written as an expected value of a random variable $X \sim p$ as follows:

$$I = \int_D f(x)\,\mathrm{d}x = \int_D \frac{f(x)}{p(x)} p(x)\,\mathrm{d}(x) = \mathbb{E}\left(\frac{f(X)}{p(X)}\right). \tag{2.81}$$

The law of large numbers states that the *Monte Carlo estimate*

$$\langle I_n^p \rangle = \frac{1}{n} \sum_{i=1}^{n} \frac{f(x_i)}{p(x_i)} \tag{2.82}$$

will converge (in probability) to I for increasing number of samples $x_1, \dots, x_n$. The estimator $\langle I_n^p \rangle$ itself is also a random variable, and

$$\mathbb{E}(\langle I_n^p \rangle) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\left(\frac{f(X)}{p(X)}\right) = \mathbb{E}\left(\frac{f(X)}{p(X)}\right) = I, \tag{2.83}$$

which means that the expected value of the estimator is the value it estimates. Estimators with this property, i.e., $\mathbb{E}(\langle I_n^p \rangle) = I$ are called *unbiased*.

Estimators without this property are called *biased* and the difference between the value of the integral and the expected value of the estimator

$$\mathbb{B}(\langle I_n^p \rangle) = I - \mathbb{E}(\langle I_n^p \rangle) \tag{2.84}$$

is called *bias*. If a biased estimator converges (in probability) to the right solution, i.e.,

$$P\Big(\lim_{n\to\infty} \langle I_n^p \rangle = I\Big) = 1 \Leftrightarrow P\Big(\lim_{n\to\infty} \mathbb{B}(\langle I_n^p \rangle) = 0\Big) = 1, \tag{2.85}$$

it is called consistent. Unbiased, biased, and consistent estimators are all used in Monte Carlo rendering, and we will discuss examples of each kind of estimator later in this chapter. Even though unbiased estimators are generally the preferred choice for rendering algorithms, it is sometimes possible to use a biased or consistent estimator, which compensates for the introduced bias by having significantly less variance.

The probability density function p can be chosen arbitrarily as long as it fulfills the property in equation 2.80. Usually, the choice of p influences the accuracy of the estimator $\langle I_n^p \rangle$ for a specific number of samples n. In statistics, the quality of an estimator $\langle I_n^p \rangle$ is often measured using the *root mean squared error*

$$\mathrm{RMSE}(\langle I_n^p \rangle) = \sqrt{\mathbb{E}((\langle I_n^p \rangle - I)^2)}, \tag{2.86}$$

which, for an unbiased estimator, equals its *standard deviation* σ, i.e.,

$$\mathrm{RMSE}(\langle I_n^p \rangle) = \sqrt{\mathrm{Var}(\langle I_n^p \rangle)}. \tag{2.87}$$

Similar to above, the variance of the estimator depends on its underlying (one sample variance)

$$\sigma^2 := \mathrm{Var}\left(\frac{f(X)}{p(X)}\right) \tag{2.88}$$

and the number of samples, i.e.,

$$\mathrm{Var}(\langle I_n^p \rangle) = \frac{1}{n^2} \sum_{i=1}^{n} \mathrm{Var}\left(\frac{f(X)}{p(X)}\right) = \frac{\sigma^2}{n} \tag{2.89}$$

Therefore, the root mean squared error of the estimator is

$$\mathrm{RMSE}(\langle I_n^p \rangle) = \sqrt{\mathrm{Var}(\langle I_n^p \rangle)} = \frac{\sigma}{\sqrt{n}}, \tag{2.90}$$

which means that we can improve the estimate by increasing the number of samples n or choosing a probability density function with smaller underlying variance σ^2. Independent of the choice of the probability density function, the convergence rate of Monte Carlo integration is always $\mathcal{O}(n^{-\frac{1}{2}})$, which means that in order to half the expected error of the estimator, we have to quadruple the number of samples used. This convergence rate is slow compared to other numerical integration methods (e.g., gaussian quadrature formulas [Stroud and Secrest 1966]). However, the convergence rate is independent of dimensionality, and there are no additional constraints (e.g., smoothness) for the integrand, which makes it more suitable for Monte Carlo rendering than other integration methods [Širca 2016, Chapter 13].

We will use the root mean squared error to evaluate our estimators in this thesis. However, we are more interested in an estimator's efficiency, which is the root mean squared error after a certain period of computation time. The efficiency of an estimator is more informative than the RMSE with a certain number of samples because the quality of an estimator is usually proportional to its computation demands for sampling, and a higher quality estimator will usually generate fewer samples per unit time. For this reason, we always state the render times in addition to the number of samples and RMSE when we compare Monte Carlo rendering algorithms later in this thesis.

2.3.7 Importance sampling

As we have seen above, the convergence of a particular Monte Carlo estimator is in $\mathcal{O}(n^{-\frac{1}{2}})$, and it is often beneficial to search for a probability density functions p such that

$$\sigma^2 = \mathrm{Var}\left(\frac{f(X)}{p(X)}\right) \tag{2.91}$$

is small. One way to do this is *importance sampling* [Širca 2016, Chapter 13], which means choosing p in a way that it is similar to f, i.e., taking on small density values where f is small and taking on large density values where f is large. The optimal importance sampling PDF would be $p \propto |f|$, i.e.,

$$p(x) = C \cdot |f(x)| \text{ for some } C \in \mathbb{R}^+, \tag{2.92}$$

since the corresponding estimator would have no variance because

$$\mathrm{Var}\left(\frac{f(X)}{p(X)}\right) = \mathbb{E}\left(\frac{1}{C^2}\frac{f^2(X)}{f^2(X)}\right) - \mathbb{E}\left(\frac{1}{C}\frac{f(X)}{|f(X)|}\right)^2 = \frac{1}{C^2} - \frac{1}{C^2} = 0. \tag{2.93}$$

However, for $p(x) = C \cdot f(x)$ to be a PDF it has to be normalized, i.e.,

$$\int_D p(x)\,\mathrm{d}x = 1 \Leftrightarrow C = \frac{1}{\int_D f(x)\,\mathrm{d}x}, \tag{2.94}$$

and the denominator is the value we want to estimate in the first place. For efficient importance sampling, p should be as similar to f as possible. Additionally, generating samples $x_i \sim p$ and evaluating $p(x_i)$ should be fast.

2.3.8 Multiple importance sampling

It is challenging to construct a single PDF that is similar to the integrand f on the whole domain D. In many of our cases, the integrand is a product of several functions (see equation 2.51), for example, $f(x) = g(x) \cdot h(x)$. It can be much simpler to construct suitable PDFs for g and h individually. *Multiple importance sampling* (MIS) [Veach and Guibas 1995] is a general framework that allows the combination of different sampling PDFs and their corresponding estimators in an optimal way. Optimal in this context means, that without further knowledge about the integrand, no other combination scheme will be significantly better than multiple importance sampling (see Theorem 1 and Theorem 2 in [Veach and Guibas 1995]).

Let $p_1, \ldots, p_n$ be a set of sampling PDFs (also called sampling techniques) and let n_i be the number of samples x_{ij} $(j = 1, \ldots, n_i)$ with $x_{ij} \sim p_i$ $(i = 1, \ldots, n)$. Then, a *multi-sample estimator*

$$\langle E \rangle = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(x_{ij}) \frac{f(x_{ij})}{p_i(x_{ij})} \tag{2.95}$$

is unbiased if the weights w_i fulfill the properties

- $\sum_{i=1}^{n} w_i(x) = 1$ whenever $f(x) \neq 0$, and
- $w_i(x) = 0$ whenever $p_i(x) = 0$.

The multi-sample estimator is unbiased because of the following identity:

$$\begin{aligned}
\mathbb{E}(\langle E \rangle) &= \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbb{E}\left(w_i(X_{ij}) \frac{f(X_{ij})}{p_i(X_{ij})} \right) && (2.96) \\
&= \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} \int_D w_i(x) \frac{f(x)}{p_i(x)} p_i(x) \, \mathrm{d}x && (2.97) \\
&= \int_D \left(\sum_{i=1}^{n} w_i(x) \right) f(x) \, \mathrm{d}x && (2.98) \\
&= \int_D f(x) \, \mathrm{d}x. && (2.99)
\end{aligned}$$

There are several choices for the weight function w that are used in practice. However, in this thesis, we will only consider the *power heuristic*, i.e.,

$$w_i^{\alpha}(x) = \frac{n_i^{\alpha} p_i^{\alpha}(x)}{\sum_{j=1}^{n} n_j^{\alpha} p_j^{\alpha}(x)}, \tag{2.100}$$

for $\alpha \in \mathbb{R}^+$. The special case of $\alpha = 1$ is also called *balance heuristic*, and it is optimal in the sense that any other multi-sample estimator cannot improve upon the balance heuristic significantly (Theorem 1 in [Veach and Guibas 1995]).

An often occurring scenario is that multiple sampling techniques $p_1, \ldots, p_n$ should be combined but only to generate one sample (e.g., choosing between following the reflection or transmission direction when sampling a light path). In this case, one of the sampling techniques is chosen randomly with probability c_i where $\sum_{i=1}^{n} c_i = 1$. We formalize the process of choosing a sampling strategy, generating a sample according to this strategy, and evaluating an appropriate weight function as the *one-sample estimator*

$$\langle E \rangle = \frac{w_I(x_I) f(x_I)}{c_I p_I(x)}, \tag{2.101}$$

where $I \in 1, \ldots n$ is a random variable distributed according to the probabilities c_i. For the one-sample estimator the balance heuristic is optimal (see Theorem 1 in [Veach and Guibas 1995]), and the resulting estimator is

$$\langle E \rangle = \frac{f(x)}{\sum_{i=1}^{n} c_i p_i(x)}, \tag{2.102}$$

which allows to encapsulate different sampling techniques conveniently in a way, which behaves as just a single sampling technique p^* with

$$p^*(x) = \sum_{i=1}^{n} c_i p_i(x). \tag{2.103}$$

2.4 An example: Bringing it all together

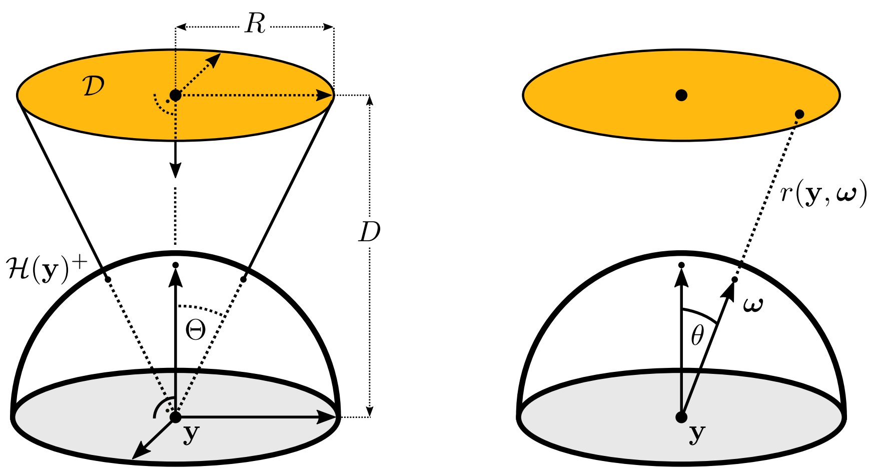


Figure 2.10: In this section, we will compute the irradiance at point $\mathbf{y}$ due to the light source $\mathcal{D}$ for this simple configuration (left) to illustrate Monte Carlo integration. The simplest way of doing so, is by sampling random directions $\boldsymbol{\omega}$ in the upper hemisphere $\mathcal{H}(\mathbf{y})^+$ and determining whether the ray $r(\mathbf{y}, \boldsymbol{\omega})$ hits the light source $\mathcal{D}$ (right).

Before discussing Monte Carlo rendering algorithms in detail in section 2.5, we will recapitulate the mathematical basics with an example that can be thought of as a simplified version of the evaluation of direct illumination. For this, we compute the irradiance from an emitting disk $\mathcal{D}$ in the straightforward geometric configuration shown in figure 2.10. We will assume the emission of the disk to be constant, i.e.,

$$L_e(\mathbf{x}, \boldsymbol{\omega}) = 1 \text{ for all } \mathbf{x} \in \mathcal{D} \text{ and } \boldsymbol{\omega} \in \mathcal{S}^2. \tag{2.104}$$

In this simple setting, we can compute the irradiance at the surface point $\mathbf{y}$ analytically using spherical coordinates as

$$\begin{aligned} E(\mathbf{y}) &= \int_{\mathcal{H}^+} L_i(\mathbf{y}, \boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega}^\perp = \int_{\mathcal{H}^+} L_i(\mathbf{y}, \boldsymbol{\omega}) \cos\theta \, \mathrm{d}\boldsymbol{\omega} && (2.105)\\ &= \int_0^{2\pi} \int_0^{\Theta} L_i(\mathbf{y}, (\phi, \theta)) \cos\theta \sin\theta \, \mathrm{d}\phi \, \mathrm{d}\theta && (2.106)\\ &= 2\pi \int_0^{\Theta} 1 \cdot \cos\theta \sin\theta \, \mathrm{d}\theta = 2\pi \left[-\frac{1}{2} \cos^2\theta \right]_0^{\Theta} && (2.107)\\ &= \pi(1 - \cos^2\Theta) && (2.108) \end{aligned}$$

If the radius of the disk is $R = 1$ and the distance of the disk center is $D = 1$, we get

$$E(\mathbf{y}) = \pi\left(1 - \frac{D^2}{D^2 + R^2}\right) = \pi\left(\frac{1}{\frac{D^2}{R^2} + 1}\right) = \frac{\pi}{2} \; [Wm^{-2}]. \tag{2.109}$$

We can see that, as expected, the irradiance decreases with the square of the distance D for a fixed radius R. We can also compute an estimate for $E(\mathbf{y})$ using Monte Carlo integration by randomly sampling uniformly distributed directions $\boldsymbol{\omega}_i$ in the upper hemisphere $\mathcal{H}(\mathbf{y})^+$ and testing if the rays $r(\mathbf{y}, \boldsymbol{\omega}_i)$ hit the disk $\mathcal{D}$ (see figure 2.10 right). Hitting the disk is equivalent to

$$\boldsymbol{\omega}_i \in \mathcal{C}_\Theta := \{\boldsymbol{\omega} \in \mathcal{S}^2 : \theta < \Theta\}, \tag{2.110}$$

where $\mathcal{C}_\Theta$ defines a cone with opening angle Θ and θ is the corresponding longitudinal angle of $\boldsymbol{\omega} = (\phi, \theta)$. If we assume $\mathbf{n}_\mathbf{y} = (0, 1, 0)^T$, then a uniformly distributed direction $\boldsymbol{\omega} = (x, y, z)$ in the upper hemisphere can be constructed in cartesian coordinates as

$$\boldsymbol{\omega} = \begin{pmatrix} \cos(2\pi\xi_2)\sqrt{1 - \xi_1^2} \\ 1 - \xi_1 \\ \sin(2\pi\xi_2)\sqrt{1 - \xi_1^2} \end{pmatrix}, \tag{2.111}$$

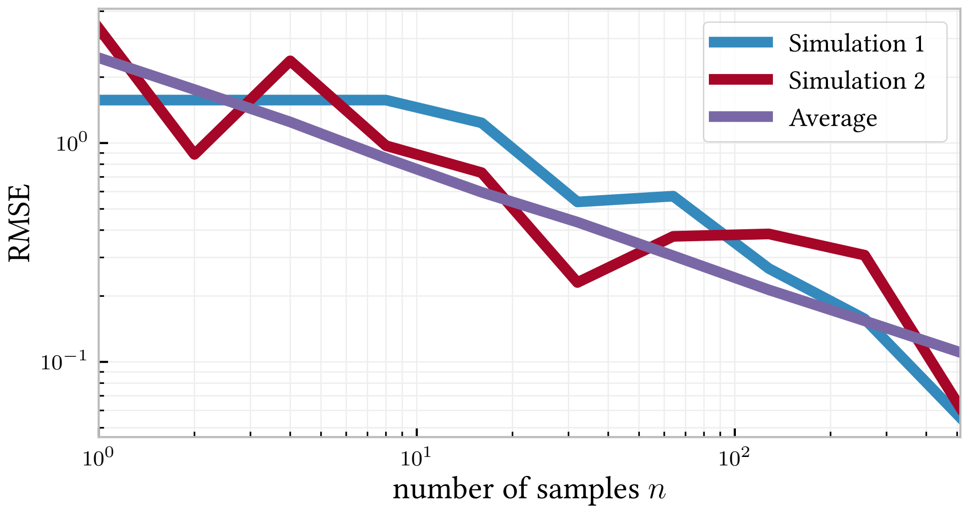


Figure 2.11: Running a Monte Carlo simulation and plotting the RMSE after a certain number of samples n results in graphs that are not monotonically decreasing and that exhibit randomness from the underlying (pseudo-)random numbers. For two different seeds for the (pseudo-)random number generator, the graphs look quite different (Simulation 1 and Simulation 2). It is more insightful to plot the expected RMSE graph, which can be generated by averaging multiple Monte Carlo simulations. Here the gray graph is the average of 1024 simulations with different random seeds.

where $\xi_1, \xi_2 \sim U[0,1]$ and $p_\sigma(\boldsymbol{\omega}) = \frac{1}{2\pi}$ in solid angle measure. The resulting Monte Carlo estimator is

$$\langle I_n^{p_\sigma} \rangle = \frac{1}{n} \sum_{i=1}^{n} \frac{\mathbf{1}_{\mathcal{C}_\Theta}(\boldsymbol{\omega}_i) L_i(\mathbf{y}, \boldsymbol{\omega}_i) \cos\theta_i}{p_\sigma(\boldsymbol{\omega}_i)} = \frac{2\pi}{n} \sum_{i=1}^{n} \mathbf{1}_{\mathcal{C}_\Theta}(\boldsymbol{\omega}_i) \cos\theta_i. \tag{2.112}$$

We can run a Monte Carlo simulation and plot the RMSE for increasing n in the log-log scale. The log-log scale is convenient because the convergence plots of unbiased Monte Carlo estimators are always straight lines. Figure 2.11 shows the results of two simulations, which are initialized with different seeds to the (pseudo-)random number generator. As we can see, RMSE($\langle I_n^{p_\sigma} \rangle$) is itself a random variable. A more insightful graph, which illustrates the convergence behavior much better is the expected RMSE. This graph can be estimated by averaging multiple simulations with different random seeds (figure 2.11).

Importance Sampling

We can create a better estimator by importance sampling the cosine term from equation 2.105. We can construct random directions $\boldsymbol{\omega} \in \mathcal{H}^+$ that follow the cosine distribution similar to before with $\xi_1, \xi_2 \sim U[0,1]$ and

$$w = \begin{pmatrix} \cos(2\pi\xi_2)\sqrt{1-\xi_1^2} \\ \sqrt{1-\xi_1} \\ \sin(2\pi\xi_2)\sqrt{1-\xi_1^2} \end{pmatrix}. \tag{2.113}$$

The corresponding probability density distribution in solid angle measure is $p_\sigma^c(\boldsymbol{\omega}) = \frac{\cos\theta}{\pi}$. It is equivalent to a uniform distribution in projected solid angle measure with PDF $p_\sigma^\perp(\boldsymbol{\omega}) = \frac{1}{\pi}$. In the resulting Monte Carlo estimator the cosine term now cancels out by the cosine term in the PDF leading to the estimator

$$\left\langle I_n^{p_\sigma^\perp} \right\rangle = \frac{\pi}{n} \sum_{i=1}^{n} \mathbf{1}_{\mathcal{C}_\Theta}(\boldsymbol{\omega}_i). \tag{2.114}$$

Constructing a PDF such that terms can be canceled out is often the goal of importance sampling since it usually reduces variance. The graphs in figure 2.12 show, that this estimator has the same convergence rate as before. However, the RMSE is always lower for a fixed sample count compared to the uniform variant.

We can create a better estimator by reasoning about the geometric configuration of the light source. If the disk is small (or far away), most of the sampled directions will miss the disk. As an alternative, we can estimate the irradiance by explicitly sampling the disk.

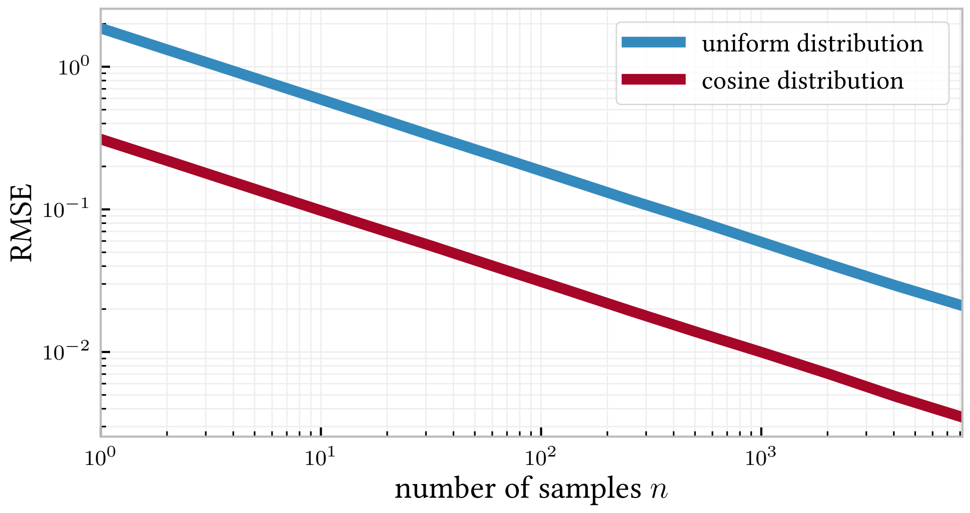


Figure 2.12: Constructing random directions that follow a cosine distribution results in a better Monte Carlo estimator for equation 2.105 compared to uniformly distributed directions. Even though the rate of convergence is the same, the RMSE values vary significantly between the two estimators for any sample count n.

Uniform sampling of points $z \in \mathcal{D}$ on the disk with the PDF $p_A(z) = \frac{1}{\pi R^2}$ in surface area measure, results in the estimator

$$\langle I_n^{p_A} \rangle = \frac{1}{n} \sum_{i=1}^{n} \frac{L_e(\mathbf{z}, \boldsymbol{\omega}_{\mathbf{z}\to\mathbf{y}}) G(\mathbf{z}, \mathbf{y})}{p_A(\mathbf{z})} = \frac{\pi R^2}{n} \sum_{i=1}^{n} G(\mathbf{z}, \mathbf{y}), \tag{2.115}$$

where G is the geometric term defined in equation 2.8. In the following, we will leave the radius of the disk fixed as $R = 1$. We compare the estimator, which samples directions according to the cosine distribution, with the estimator, which samples points on the light source $\mathcal{D}$. This comparison is performed for varying distances D of the disk center to $\mathbf{y}$. In figure 2.13, we show the normalized (expected) RMSE values for both estimators with $n = 32$ samples, depending on D. We normalize the RMSE values by the integral value (which also depends on D) to get an error illustration that is invariant of the magnitude of the value to be estimated. As we can see, direction sampling ($p_{\sigma^\perp}$) performs better for small distances, whereas point sampling (p_A) performs better for large distances.

Multiple Importance Sampling

In this simple example, we can see that the area sampling estimator is preferable for distances $D > 0.7$. However, it is not trivial to determine which of several estimators will

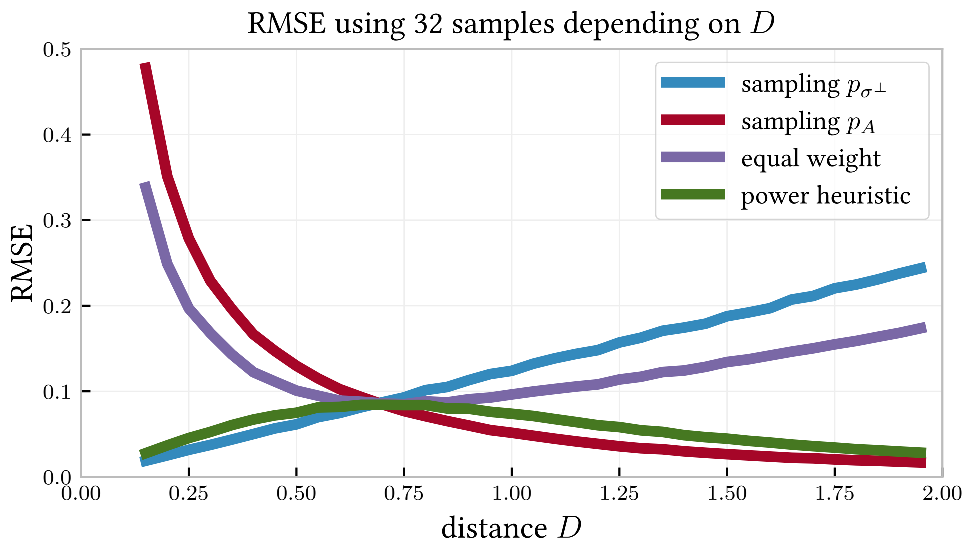


Figure 2.13: This figure shows the RMSE of different sampling strategies for a fixed number of samples (32) and varying distances of the light source D. Usually, importance sampling strategies are more or less suited, depending on properties, which are usually unknown during a complex Monte Carlo simulation. Here, the efficiency of the importance sampling strategies $p_{\sigma^\perp}$ and p_A depends on the distance D. Several sampling strategies can be combined to create more robust estimators. However, care has to be taken to combine them beneficially. For example, a combination with equal weights (red) results in an estimator that performs similar to the worst of the individual estimators. The framework of multiple importance sampling provides a superior combination of several estimators (green).

have the lowest variance in general. Therefore, we would like to combine the estimators and benefit from their individual strengths. The simplest way to combine the two estimators is to weight them equally, i.e.,

$$\frac{1}{2}\left\langle I_n^{p_\sigma^\perp}\right\rangle + \frac{1}{2}\langle I_n^{p_A}\rangle. \tag{2.116}$$

However, as can be seen in figure 2.13, this combination is not optimal because it performs similar to the worst of the estimators. Combining the two estimators with multiple importance sampling using the power heuristic with $\alpha = 2$ (shown in figure 2.13 in green) is much better and always performs similarly to the best of the two estimators over the whole range of values of D. The MIS weights for the estimators are

$$w_A(\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{z}}) = \frac{p_A(\mathbf{z})^2}{p_A(\mathbf{z})^2 + (p_\sigma^\perp(\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{z}})G(\mathbf{y},\mathbf{z}))^2} \tag{2.117}$$

and

$$w_\sigma(\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{z}}) = \frac{p_\sigma^\perp(\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{z}})^2}{p_\sigma^\perp(\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{z}})^2 + (p_A(\mathbf{z})/G(\mathbf{y},\mathbf{z}))^2} \tag{2.118}$$

where the geometric term G is needed to transform both the probabilities into the same measure.

2.5 Monte Carlo rendering

Now, equipped with the physical and mathematical fundamentals, we will describe the Monte Carlo-based rendering algorithms, which are the most relevant for this thesis. First, however, we will briefly discuss how the building blocks of a Monte Carlo rendering system are modeled.

Scene representation

All surfaces in this thesis are represented as polygon meshes consisting of triangles or quads, as shown in figure 2.14. Each point on a surface is associated with a BSDF that describes the scattering properties of the surface. A surface can be *textured*, which means that material properties, such as the BSDF parameters and the albedo (color), or geometric properties, such as the normal, are defined by an image texture that is mapped onto the surface. To accelerate intersection tests between rays and surfaces, we use bounding volume hierarchies (BVH) [Rubin and Whitted 1980; Dammertz et al. 2008; Wald et al. 2014], which are commonly used in production rendering for this purpose.

Homogeneous participating have constant scattering and absorption coefficients for the whole domain of the volume. These media are either “everywhere,” and replace the vacuum

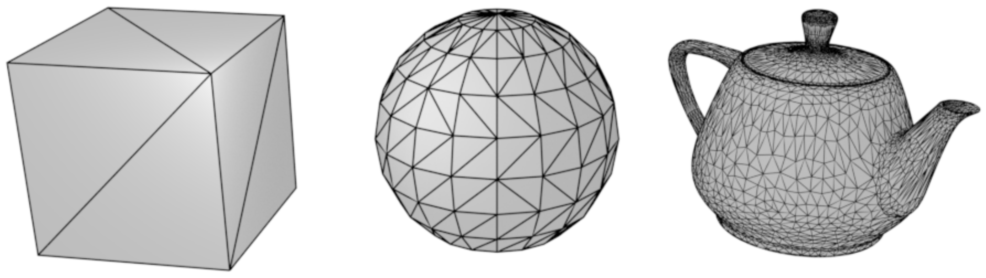


Figure 2.14: This figure shows several triangle meshes with increasing complexity. All surfaces in this thesis are represented as meshes consisting of triangles or quads.

that is usually assumed to be between surfaces, or they are enclosed inside a watertight surface mesh, which can optionally have a surface BSDF. Heterogeneous participating media are commonly represented as a spatially discretized density field, such as a regular 3-dimensional grid. Since such grids can quickly become very large for even moderate resolutions, hierarchical data structures are often used to reduce the memory requirements and allow for more detailed volumetric data sets [Museth 2013].

Light sources

Modeling real light sources is often not feasible because they exhibit a high geometrical complexity. Instead, we use several simplifying models of light sources, which allow us to create realistic scenes nonetheless.

Point lights are defined by a point in space and emit energy in every direction. Point lights can have a directionally varying emission profile (see chapter 4). The energy of a point light is concentrated at one point in space, which is physically impossible and can lead to artifacts. These artifacts are discussed later in this chapter. Despite these potential artifacts, they are still an often-used light source due to their simplicity.

Spotlights emit their energy into a cone of directions, usually with a falloff to the cone boundary. *Area lights* are surface patches that emit energy. *Environment lights* are specified by high dynamic range images of the whole spherical domain and are usually captured from real environments. Every texel in the environment map defines the incident radiance for all directions corresponding to this texel.

Volumetric lights are a complicated form of light source, and the main topic of chapter 5. In figure 2.15, a simple scene is illuminated by a point, spot, area, and environment light, respectively. In the insets, the emission profiles of the lights are visualized by placing the light source into a homogeneous medium. This visualization is inspired by a foggy night, which makes the emission profile of headlights of cars or street lights visible.

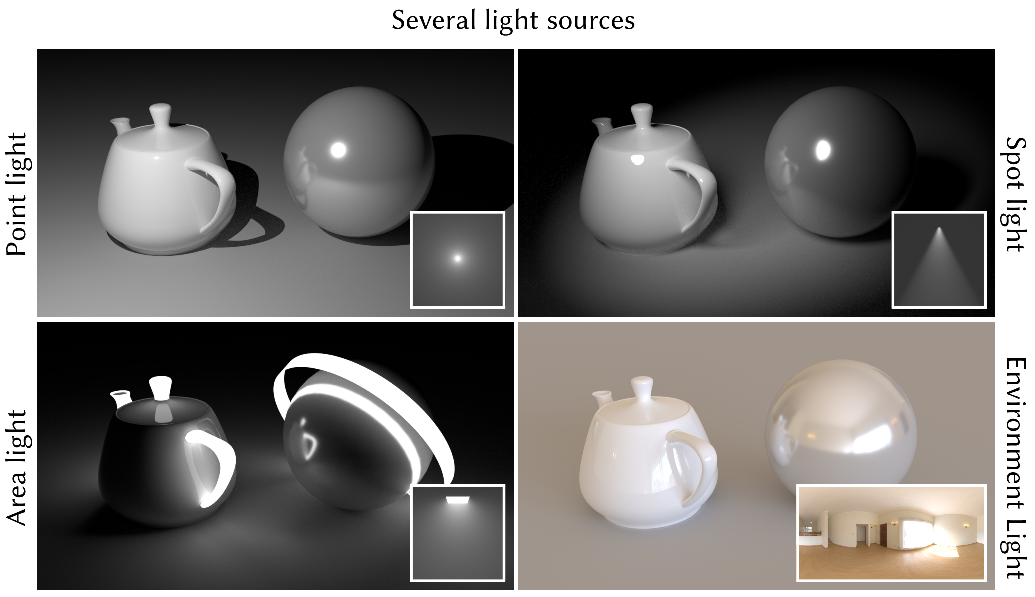


Figure 2.15: This figure shows a scene that is illuminated by a point, spot, area, and environment light, respectively. The insets illustrate the emission profile of the light sources. The emission profile is visualized by placing the light source into a homogeneous medium (like fog). The bottom-right inset shows the captured environment, which was used to illuminate the scene.

Camera Models

Camera models with sophisticated lens systems that can simulate complex effects like chromatic aberrations and barrel distortion can be integrated into the Monte Carlo rendering framework [Hullin et al. 2012; Hanika and Dachsbacher 2014; Schrade et al. 2016]. However, we use a thin lens camera model [Barsky et al. 2003] in this thesis for simplicity. In this model, a camera consists of a film (on which the raster image is placed), aperture, focal length, and position. The pixels of the raster image can be thought of as small sensors that measure incoming light. By modifying the focal length and aperture size, certain regions of a scene can be put into focus, whereas others get blurred. This *depth of field* effect is a fundamental artistic tool that virtually every photographer uses. To generate a ray originating at a pixel and exiting the camera system into the scene, we need to sample a point on the film and a point on the usually disk-shaped aperture. The origin of the ray is the sampled point on the aperture. The ray's direction is defined by the intersection of the focal plane with a ray from the sampled position through the center of the aperture. Figure 2.16 illustrates this configuration. An aperture with size zero results in the *pinhole camera model* [Barsky et al. 2003], which is often used in image synthesis.

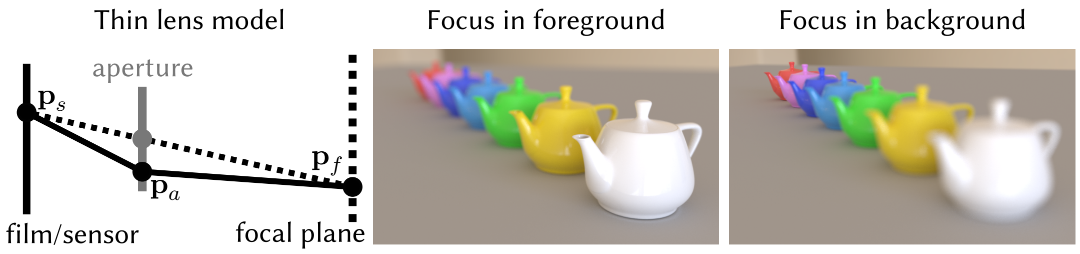


Figure 2.16: The images in this work are rendered using the *thin lens* camera model, which allows focusing on a specific part of the scene and blurring of others (middle and right). In this model, a ray is generated by sampling a point on the camera's sensor $\mathbf{p}_s$ and a point on the aperture $\mathbf{p}_a$. The resulting ray is $r = (\mathbf{p}_a, \boldsymbol{\omega}_{\mathbf{p}_a \to \mathbf{p}_f})$, where $\mathbf{p}_f$ is the intersection $\mathbf{p}_f$ of the focal plane and the ray originating at $\mathbf{p}_s$ through the center of the aperture.

2.5.1 Path tracing

The first image synthesis method we discuss is *path tracing*, which was first described by James Kajiya in his seminal work [Kajiya 1986] that introduced the rendering equation to the graphics research community. Path tracing constructs a light path by incrementally sampling path vertices starting from the sensor until the path reaches a light source or until a termination criterion is met. A path is started by sampling a point on the camera sensor, which specifies the pixel in which the path contribution will be added, and a point on the aperture that determines the direction in which the path will be extended. Next, a ray cast is performed to determine the next surface point. If the ray does not hit a surface in the corresponding direction, we return the emission of the environment light. Otherwise, we will successively sample the next direction, by importance sampling the BSDF f, and cast a ray to get the next surface point. At each point, we check if the surface point is emissive, in which case we add the path contribution to the image and terminate the path.

With this simple procedure, it can be unlikely to hit a light source. For example, it would not be able to render scenes that are only illuminated by point lights, as it is impossible to hit them randomly. Therefore, we will additionally sample light sources at each path vertex explicitly, which is a technique called *next event estimation.* For this, we stochastically pick a point on a light source and evaluate the illumination from that point (i.e., checking visibility and evaluating the BSDF). For environment lights, we will sample a direction proportionally to the energy distribution of the environment texture instead of a point.

With next event estimation, there are two ways of sampling a particular light path. Either *implicitly*, by sampling the BSDF and hitting the light source with the ray cast, or *explicitly*, by sampling the point on the light source directly. Both techniques are combined using multiple importance sampling, which weights their contribution according to the respective sampling probabilities.

To determine the contribution of a path $\mathbf{X}$, we have to compute the path PDF $p_i(\mathbf{X})$ and $p_e(\mathbf{X})$ in vertex area measure. The PDF p_i and p_e are the PDF of constructing the path implicitly or explicitly. Since BSDF sampling results in a direction $\boldsymbol{\omega}$ with PDF $p_\sigma^\perp(\boldsymbol{\omega})$ in projected solid angle measure, we have to convert this probability into area measure, by multiplying with the corresponding geometric term as described in equation 2.9. Therefore, we can compute the PDF $p_i(\mathbf{X})$ of a path $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ as

$$p_i(\mathbf{X}) = p_A(\mathbf{x}_0) \prod_{i=0}^{k-1} p_\sigma^\perp(\boldsymbol{\omega}_{\mathbf{x}_i \to \mathbf{x}_{i+1}}) G(\mathbf{x}_i, \mathbf{x}_{i+1}). \tag{2.119}$$

The PDF of a path constructed with next event estimation is

$$p_e(\mathbf{X}) = p_A(\mathbf{x}_0) \left(\prod_{i=0}^{k-2} p_\sigma^\perp(\boldsymbol{\omega}_{\mathbf{x}_i \to \mathbf{x}_{i+1}}) G(\mathbf{x}_i, \mathbf{x}_{i+1}) \right) p_A(\mathbf{x}_k). \tag{2.120}$$

The MIS weights using the balance heuristic for a path constructed with the two techniques are

$$w_i(\mathbf{X}) = \frac{p_i(\mathbf{X})}{p_i(\mathbf{X}) + p_e(\mathbf{X})} \quad \text{and} \quad w_e(\mathbf{X}) = \frac{p_e(\mathbf{X})}{p_i(\mathbf{X}) + p_e(\mathbf{X})}, \tag{2.121}$$

respectively. A high-level version of path tracing is given in pseudo-code in algorithm 2.

Images rendered with path tracing and other Monte Carlo rendering methods exhibit noise due to the variance of the underlying Monte Carlo estimator (see figure 2.17). As more light paths are sampled, and their contributions averaged, the noise vanishes. However, the convergence is always $\mathcal{O}(\frac{1}{\sqrt{n}})$ where n is the number of paths per pixel. In practical terms, this means that the image converges with a fast rate for low sample counts, but after this initial phase residual noise requires increasing effort to be resolved. The number of light paths required to produce a perceptually noise-free image can be too large for practical use if an illumination effect can not be sampled efficiently.

The efficiency of path tracing depends heavily on the next event estimation. If next event estimation is not possible at a surface point, for example, because a small light source is "blocked" by a glass object, the corresponding region can be very noisy. This noise arises because we have to trace through the glass object and hit the light source by chance. In general, rendering caustic effects where light from a small light source is bundled or concentrated, for example, by a glass sphere, can be quite challenging for path tracing. These paths are often outliers, i.e., samples with very high contribution due to low probability, which can take a lot of additional sampling effort to resolve.

There are several options to solve this problem. First, we could ignore the outliers and not add them to the image. Ignoring outliers is a valid and safe approach if they are rare outliers due to numerical artifacts. However, this is not a solution if we want to compute challenging illumination effects.

The second option is to construct more sophisticated Monte Carlo rendering algorithms, like the ones that will be introduced in the remainder of this section. However, these algorithms all have their respective drawbacks as well. They are generally more complicated to implement in practice, and they might cause significant computational overhead.

Another option is to improve a simple Monte Carlo rendering method with an additional component, which allows it to handle a specific effect efficiently. This option is also the approach of the techniques introduced in this thesis. The technique in chapter 4 extends the many-light algorithm, which will be introduced below, to handle glossy surfaces better. In chapter 5, we introduce extensions to path tracing, which allow robust handling of a wide variety of heterogeneous emissive volumes. Lastly, chapter 6 introduces an adaptive path sampling method similar to path tracing, that can sample complex illumination effects, such as caustics, more efficiently.

Apart from these options to remove noise, low levels of residual image noise are often resolved using *denoising algorithms* [Zwicker et al. 2015] instead of generating more samples in production. However, apart from the outlier removal in chapter 6, we are not performing denoising of images in this thesis.

2.5.2 Bidirectional path tracing

Path tracing can become inefficient when next event estimation is not possible. For example, when a glass object is between the light source and the current path vertex. In this case, we could trace paths starting from the light source and connect to the camera. However, this approach, called *light tracing* [Arvo 1986], suffers from the same problems as path tracing when we exchange the positions of camera and light source. A more robust way of constructing paths is *bidirectional path tracing* [E. Lafortune and Y. Willems 1993; Veach and Guibas 1994], which constructs a subpath starting from the camera and a subpath starting from a light source. Connecting all vertices of the camera path to all vertices of the light path results in many full light paths.

With bidirectional path tracing, there are k possible ways to construct a path of length k. The connection of the camera and light path vertex could be at every of the $k - 1$ path

Algorithm 2 Path tracing with next event estimation

```
procedure PathTracing( )
    $\mathbf{X} \leftarrow \emptyset$                                   ▷ Path is empty
    $j, \mathbf{x}, \boldsymbol{\omega} \leftarrow$ SamplePixelAndAperture( )
    $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{x}$                         ▷ Append aperture vertex $\mathbf{x}$ to $\mathbf{X}$
    $\mathbf{x} \leftarrow$ NextPathVertex($\mathbf{x}, \boldsymbol{\omega}$)
    $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{x}$                         ▷ Append first inner vertex to $\mathbf{X}$
    while Length($\mathbf{X}$) $<$ MaxPathLength do
        $\mathbf{x}_e \leftarrow$ SampleLightSources( )                ▷ next event estimation
        if Visible($\mathbf{x}, \mathbf{x}_e$) then
            $\mathbf{X}_e \leftarrow \mathbf{X} + \mathbf{x}_e$
            $P_j \leftarrow P_j + w_e(\mathbf{X}_e)\frac{f_j(\mathbf{X}_e)}{p_e(\mathbf{X}_e)}$         ▷ add contribution to pixel value
        end if
        $\boldsymbol{\omega} \leftarrow$ SampleDirection($\mathbf{x}, \boldsymbol{\omega}$)
        $\mathbf{x} \leftarrow$ NextPathVertex($\mathbf{x}, \boldsymbol{\omega}$)
        $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{x}$
        if $f_j(\mathbf{X}) > 0$ or $\mathbf{x}$ on environment then
            $P_j \leftarrow P_j + w_i(\mathbf{X})\frac{f_j(\mathbf{X})}{p_i(\mathbf{X})}$           ▷ add contribution to pixel value
            return
        end if
    end while
end procedure
```

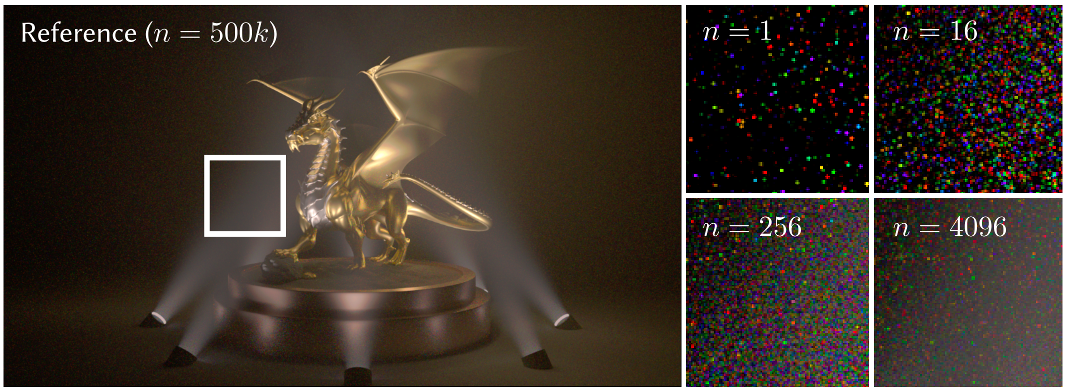


Figure 2.17: Images generated using Monte Carlo rendering exhibit noise due to the variance of the underlying Monte Carlo estimators. This noise vanishes as the number of sampled light paths per pixel n increases. However, if a Monte Carlo sampler cannot sample a complex illumination effect efficiently, the number of paths required to produce a perceptually noise free image can become too large for practical purposes.

segments. Additionally, the path could be the full camera path with no connection at all. To shorten the notation, we will define a subpath PDF $p_{i,j}(\mathbf{X})$ for a path $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ as

$$p_{i,j}(\mathbf{X}) = \begin{cases} \prod_{t=i}^{j-1} p_\sigma^\perp(\boldsymbol{\omega}_{\mathbf{x}_t \to \mathbf{x}_{t+1}}) G(\mathbf{x}_t, \mathbf{x}_{t+1}) & \text{if } i \leq j, \\ \prod_{t=i}^{j+1} p_\sigma^\perp(\boldsymbol{\omega}_{\mathbf{x}_t \to \mathbf{x}_{t-1}}) G(\mathbf{x}_t, \mathbf{x}_{t-1}) & \text{otherwise.} \end{cases} \tag{2.122}$$

If the path $\mathbf{X}$ was constructed with a connection at the l-th segment, we compute the corresponding PDF $p_l^k(\mathbf{X})$ for $l = 1, \ldots, k$ as

$$p_l^k(\mathbf{X}) = \begin{cases} p_A(\mathbf{x}_1) p_{1,k}(\mathbf{X}) & \text{if } l = k, \\ p_A(\mathbf{x}_k) p_{k,2}(\mathbf{X}) & \text{if } l = 1, \\ p_A(\mathbf{x}_1) p_{1,l}(\mathbf{X}) p_{k,l+1}(\mathbf{X}) p_A(\mathbf{x}_k) & \text{otherwise.} \end{cases} \tag{2.123}$$

The MIS weights using the balance heuristic for this path is computed are

$$w_l^k(\mathbf{X}) = \frac{p_l^k(\mathbf{X})}{\sum_{i=1}^k p_i^k(\mathbf{X})}. \tag{2.124}$$

A high-level version of bidirectional path tracing is given in pseudo-code in algorithm 3.

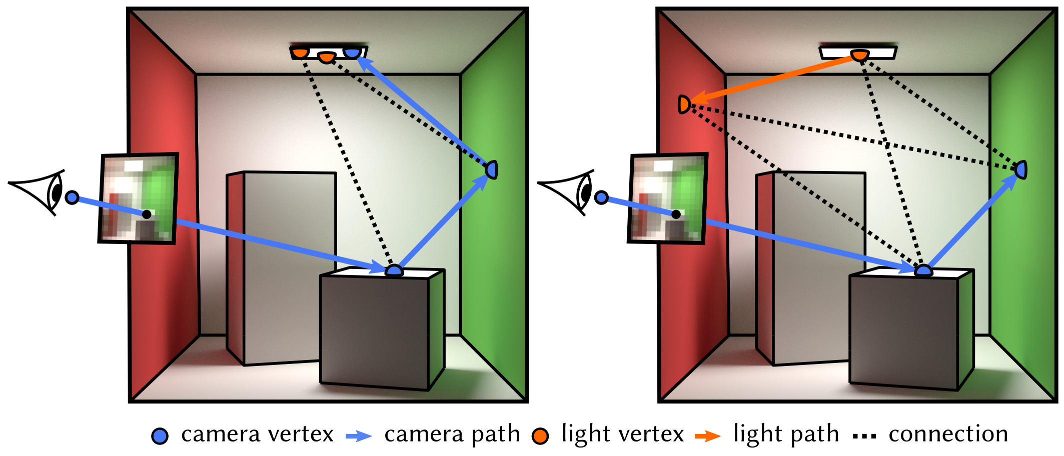


Figure 2.18: Path tracing (left) constructs paths starting from the camera by incrementally sampling the BSDF and casting a ray in the corresponding direction. At every path vertex, the direct illumination from a randomly chosen point on a light source is determined (next event estimation). Bidirectional path tracing (right) creates camera and light subpaths in the same incremental way as path tracing. It generates a multitude of light paths by connecting every vertex of a camera subpath to every vertex of a light subpath.

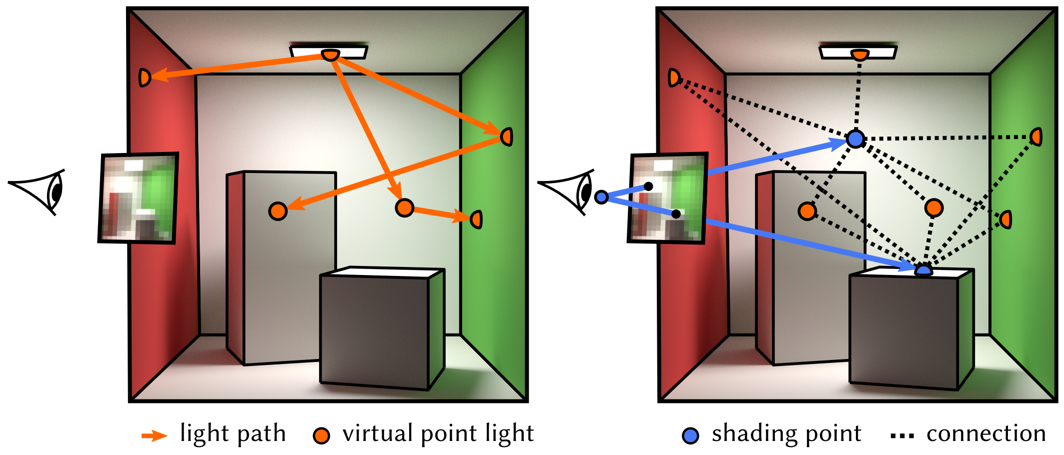


Figure 2.19: Many-light rendering consists of two steps. First, light subpaths are created, and each vertex is stored along with the incident direction and information about the BSDF at the vertex location (left). The stored vertices are called virtual point lights (VPLs). The VPLs are then used in the second step to illuminate shading points, which are sampled from the camera (right). The specific feature of many-light methods is that the information of all VPLs is used to compute the illumination at every shading point, which results in noise-free images.

Algorithm 3 Bidirectional path tracing with MIS

```
1: procedure BidirectionalPathTracing( )
2:     $\mathbf{X}_L \leftarrow \emptyset$                                         ▷ Light path is empty
3:     $\mathbf{x}, \boldsymbol{\omega} \leftarrow$ SampleLightSourceAndDirection( )
4:     $\mathbf{X} \leftarrow \mathbf{X}_L + \mathbf{x}$                              ▷ Append light vertex to $\mathbf{X}_L$
5:     while Length($\mathbf{X}_L$) $<$ MaxPathLength do                ▷ Construct light path
6:         $\mathbf{x}_1, j \leftarrow$ ConnectToCamera($\mathbf{X}_L$)          ▷ Aperture vertex and corresp. pixel
7:         $\mathbf{X} \leftarrow \mathbf{x}_1 + \mathbf{X}_L$                              ▷ Full light path
8:         $k \leftarrow$ Length($\mathbf{X}$)
9:         $P_j \leftarrow P_j + w_1^k(\mathbf{X}) \frac{f_j(\mathbf{X})}{p_1^k(\mathbf{X})}$               ▷ add contribution to pixel value
10:        $\mathbf{x}' \leftarrow$ NextPathVertex($\mathbf{x}, \boldsymbol{\omega}$)
11:        if $L_e(\mathbf{x}', \boldsymbol{\omega}_{\mathbf{x}' \to \mathbf{x}}) > 0$ or $\mathbf{x}'$ on environment then
12:            break
13:        end if
14:        $\mathbf{x} \leftarrow \mathbf{x}'$
15:        $\boldsymbol{\omega} \leftarrow$ SampleDirection($\mathbf{x}, \boldsymbol{\omega}$)
16:    end while
17:    $k_l \leftarrow$ Length($\mathbf{X}_L$)                                   ▷ Length of light path
18:    $\mathbf{X}_C \leftarrow \emptyset$                                       ▷ Camera path is empty
19:    $j, \mathbf{x}, \boldsymbol{\omega} \leftarrow$ SamplePixelAndAperture( )
20:    $\mathbf{X}_C \leftarrow \mathbf{X}_C + \mathbf{x}$                          ▷ Append aperture point vertex $\mathbf{X}$
21:    while Length($\mathbf{X}_C$) $<$ MaxPathLength do
22:        $\mathbf{x}' \leftarrow$ NextPathVertex($\mathbf{x}, \boldsymbol{\omega}$)
23:        if $L_e(\mathbf{x}', \boldsymbol{\omega}_{\mathbf{x}' \to \mathbf{x}}) > 0$ or $\mathbf{x}'$ on environment then
24:            $\mathbf{X} \leftarrow \mathbf{X}_C + \mathbf{x}'$
25:            $k \leftarrow$ Length($\mathbf{X}$)
26:            $P_j \leftarrow P_j + w_k^k(\mathbf{X}) \frac{f_j(\mathbf{X})}{p_k^k(\mathbf{X})}$
27:            break
28:        else
29:            $l \leftarrow$ Length($\mathbf{X}_C$)
30:            for $i \leftarrow 1, \ldots, k_l$ do
31:                $\mathbf{X} = \mathbf{X}_C + \mathbf{X}_L[i, \ldots, k_l]$      ▷ Connect camera path to light subpath
32:                $k \leftarrow$ Length($\mathbf{X}$)
33:                $P_j \leftarrow P_j + w_l^k(\mathbf{X}) \frac{f_j(\mathbf{X})}{p_l^k(\mathbf{X})}$
34:            end for
35:        end if
36:        $\boldsymbol{\omega} \leftarrow$ SampleDirection($\mathbf{x}, \boldsymbol{\omega}$)
37:        $\mathbf{x} \leftarrow$ NextPathVertex($\mathbf{x}, \boldsymbol{\omega}$)
38:    end while
39: end procedure
```

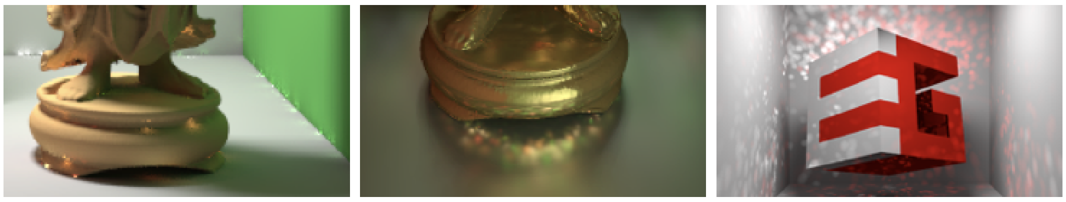


Figure 2.20: Many-light methods struggle from several artifacts: Bright splotches in cavities and corners due to the small distance between virtual point light (VPL) and shading point (left), splotchy reflections of individual VPLs in glossy surfaces (middle), and splotches due to the narrow emission profile of VPLs on glossy surfaces (right). In chapter 4 we introduce Rich-VPLs, which help alleviate all of these problems.

2.5.3 Many-light methods

Path tracing and bidirectional path tracing can capture most lighting phenomena in practice [Fascione et al. 2017] but often produce noisy images, even with very long computation times. Many-light methods, first introduced by [Keller 1997], have been prominent in computing (near) artifact-free images with low noise levels within predictable render times. Many-light methods compute the light transport in a scene by accumulating the direct illumination from many virtual point lights (VPLs). This approach offers a simple solution to many, but not all, rendering problems. Nowadays, many-light methods are popular for preview rendering. The objective of preview rendering is to generate an approximate solution in a given restricted time budget that is as good as possible.

In many-light rendering, image synthesis is divided into two phases. First, a set of light paths is traced from the light sources. A VPL is placed at each path vertex, and information about the incident radiance and BSDF is stored. In the second phase, visible surface points are sampled by constructing camera subpaths, and the illumination of all previously created VPLs is evaluated and accumulated. Among several other benefits, this method has the desirable advantage that it is scalable in the sense that the accuracy of the approximation and render time can be controlled by adjusting the number of VPLs.

In many cases, a small number of VPLs can produce a realistic image with indirect illumination, and the many-light method can be efficiently implemented on the GPU for real-time global illumination [Keller 1997]. However, in scenes with glossy materials or participating media, the number of VPLs needed to render an image with high fidelity will be too large for real-time applications. Often the illumination of millions of VPLs has to be accumulated in the second phase of image synthesis. For these scenarios, hierarchical data structures that cluster VPLs have been proposed for efficient rendering [Walter et al. 2005; Walter et al. 2006; Walter et al. 2012].

Many-light rendering can be seen as a special case of bidirectional path tracing where each camera path uses the same set of VPLs (i.e., connections to light vertices). This approach results in a noise-free image; however, it also introduces correlation, which can result in severe artifacts. These artifacts arise because the energy of a VPL is concentrated at a singular point in space. Additionally, the emission of a VPL is concentrated in a narrow cone of directions, which can cause problems in the presence of glossy surfaces. These artifacts can be seen as bright splotches in corners and cavities, reflections of individual VPLs in glossy surfaces, and illumination splotches, if the scene contains highly glossy materials, as shown in figure 2.20. In general, most existing many-light methods struggle with glossy surfaces, and VPLs are often assumed to be diffuse, which means they emit their energy equally in all directions [Walter et al. 2012].

Alternatively, their contributions can be clamped to avoid spikes in the illumination due to a narrow emission profile that only illuminates a small portion of a scene. However, clamping introduces bias (see section 2.3.6) and impacts the perception of glossy materials in a scene [Křivánek et al. 2010]. Virtual spherical lights (VSLs) [Hašan et al. 2009] reduce artifacts and avoid clamping by spreading the energy of a virtual light over a non-singular domain such as a disk or sphere. However, this comes at the expense of higher shading cost, and large numbers of virtual lights are still required to capture glossy light transport faithfully. In chapter 4, we introduce a novel method to improve many-light rendering for scenes with glossy materials.

2.5.4 Photon mapping

Another bidirectional approach for light transport simulation is photon mapping [Jensen 1996]. In contrast to bidirectional path tracing, photon mapping is based on *kernel density estimation*, which computes radiometric quantities with densities of particles. Similar to many-light methods, photon mapping has a phase in which light paths are traced from the light sources. At each path vertex, the location $\mathbf{x}_k$, incoming direction $\boldsymbol{\omega}_k$, and incoming flux $\Delta\Phi_k$ of a photon are stored in a photon map. The photon map represents an approximation of the light distribution in the scene. In the second stage, paths are traced starting from the camera until a non-specular surface point is reached. Outgoing radiance is computed by performing density estimation by searching all particles within a sphere of a certain radius. Let K be the number of photons within a radius r. By using the definition of incident radiance from equation 2.18 and the rendering equation (equation 2.27), the outgoing radiance can be estimated as follows:

$$
\begin{aligned}
L_o(\mathbf{x}, \boldsymbol{\omega}_o) &= \int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) L_i(\mathbf{x}, \boldsymbol{\omega}_i)\, \mathrm{d}\boldsymbol{\omega}_i^{\perp} && (2.125)\\
&= \int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \frac{\mathrm{d}\Phi_i(\mathbf{x}, \boldsymbol{\omega}_i)}{dA} && (2.126)\\
&\approx \sum_{k=1}^{K} f_s(\mathbf{x}, \boldsymbol{\omega}_k, \boldsymbol{\omega}_o) \frac{\Delta\Phi_k}{\Delta A} && (2.127)\\
&= \sum_{k=1}^{K} f_s(\mathbf{x}, \boldsymbol{\omega}_k, \boldsymbol{\omega}_o) \frac{\Delta\Phi_k}{\pi r^2}. && (2.128)
\end{aligned}
$$

In addition to an equal weighting of photons, density estimation can also use filter kernels. With a filter kernel, the contribution of photons can vary depending on their distance to the query point. Additionally, a more adaptive K-nearest neighbor search can be used instead of a fixed size radius. For example, the outgoing radiance can be estimated with a *2D Epanechnikov kernel* [Epanechnikov 1969] as

$$
L_o(\mathbf{x}, \boldsymbol{\omega}) \approx \frac{1}{\pi d_K^2} \sum_{k=1}^{K} f_s(\mathbf{x}, \boldsymbol{\omega}_k, \boldsymbol{\omega}_o) \left(1 - \frac{d_k}{d_K}\right) \Delta\Phi_k, \qquad (2.129)
$$

where $d_k = \|\mathbf{x} - \mathbf{x}_k\|^2$ is the distance of the k-th photon to the query point $\mathbf{x}$ and $d_K = \max_k d_k$ is the maximum distance of all photons to $\mathbf{x}$.

In contrast to bidirectional path tracing, photon mapping is not unbiased. There are different sources of systematic error. For example, the BSDF at a photon location $\mathbf{x}_k$ can be different from the BSDF at $\mathbf{x}$. There could also be geometry between $\mathbf{x}$ and $\mathbf{x}_k$, and ignoring this will lead to light leaking into parts of the scene where it should not be. The amount of bias depends on the query radius r and the total amount of photons distributed in the scene. However, there are methods to reduce the amount of bias in photon mapping [Qin et al. 2015], and there are ways to make photon mapping consistent (see section 2.3.6) by progressively decreasing the query radius [Hachisuka et al. 2008b; Hachisuka and Jensen 2009; Knaus and Zwicker 2011; A. S. Kaplanyan and Dachsbacher 2013].

2.5.5 VCM

Bidirectional path tracing and photon mapping seem to be fundamentally different approaches to solving the rendering equation since photon mapping is using the kernel density estimation. However, they can be unified in a generalized path integral frame-

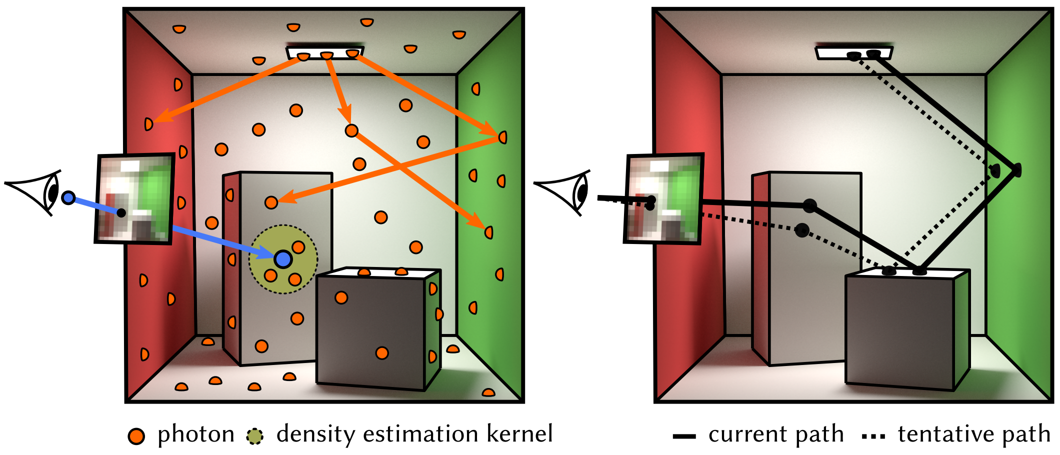


Figure 2.21: Similar to many-light methods, photon mapping (left) is a two-pass algorithm. Photon mapping uses kernel density estimation to approximate the outgoing radiance using nearby photons. In contrast to the other algorithms, Metropolis light transport (right) is based on Markov chain Monte Carlo. Here, paths cannot only be sampled from scratch but also mutated to create similar paths. A mutated path is rejected or accepted probabilistically. Mutating paths allows local exploration of path space.

work [Georgiev et al. 2012a; Hachisuka et al. 2012]. In vertex connection and merging (VCM) [Georgiev et al. 2012a], the photon lookup is not interpreted as a density estimation. Instead, the query point (last camera path vertex) is merged with the photon (last light path vertex) to create a full path. This merged path is then just another way to construct paths within the MIS framework. This technique results in an increased number of light paths for which MIS weights have to be computed. Therefore, these approaches have significant memory and computational overhead.

2.5.6 Markov chain Monte Carlo

Metropolis light transport (MLT) [Veach and Guibas 1997], based on the Metropolis-Hastings algorithm [Hastings 1970], is a different approach to light transport simulation. In contrast to other Monte-Carlo methods, it does not create light paths independently. Instead, Metropolis light transport performs a random walk in path space by generating a sequence of path mutations, which are either accepted or rejected using a carefully chosen probability. This method computes a histogram in image space, which converges to the correct solution if the Markov chain fulfills the *detailed balance* criterion (see below). The mutation strategies define a *tentative transition function* $T(\mathbf{X} \to \mathbf{X}')$, which is the probability density that a mutation of $\mathbf{X}$ results in $\mathbf{X}'$. The measurement contribution function can be split into the *image contribution function* f, which contains all terms that are independent of the pixel j, and h_j, which contains the remaining factors, i.e.,

$$I_j = \int_{\mathcal{P}} f_j(\mathbf{X})\mu(\mathbf{X}) = \int_{\mathcal{P}} h_j(\mathbf{X}) f(\mathbf{X})\mu(\mathbf{X}). \tag{2.130}$$

The Markov chain has the *stationary distribution* f, if the acceptance probability $a(\mathbf{X} \to \mathbf{X}')$ fulfills the *detailed balance* property, i.e.,

$$f(\mathbf{X})T(\mathbf{X} \to \mathbf{X}')a(\mathbf{X} \to \mathbf{X}') = f(\mathbf{X}')T(\mathbf{X}' \to \mathbf{X})a(\mathbf{X}' \to \mathbf{X}). \tag{2.131}$$

We can achieve this by defining the acceptance probability a as

$$a(\mathbf{X} \to \mathbf{X}') = \min\left\{1, \frac{f(\mathbf{X}')T(\mathbf{X}' \to \mathbf{X})}{f(\mathbf{X})T(\mathbf{X} \to \mathbf{X}')}\right\}. \tag{2.132}$$

A high-level overview of Metropolis light transport is given in algorithm 4. One of the main advantages of MLT is the local exploration of path space. For example, if MLT samples a light path, which is difficult to find, it will explore the corresponding feature by creating similar paths using mutations. In such cases, path mutation can be much more efficient than independent sampling, which has to start from scratch. In chapter 6, we introduce a technique, which allows local exploration of path space without the Markov chain framework and its detailed balance restriction.

Algorithm 4 Metropolis light transport

```
 1: procedure METROPOLITLIGHTTRANSPORT( )
 2:     $\mathbf{X} \leftarrow$ SAMPLEPATH( )                          ▷ e.g., with path tracing
 3:     for $i \leftarrow 1, \ldots, N$ do
 4:         $\mathbf{X}' \leftarrow$ MUTATEPATH($\mathbf{X}$)
 5:         $a \leftarrow= \min\left\{1, \frac{f(\mathbf{X}')T(\mathbf{X}' \to \mathbf{X})}{f(\mathbf{X})T(\mathbf{X} \to \mathbf{X}')}\right\}$
 6:         $\xi \leftarrow$ RANDOM( )              ▷ uniform random number in $[0, 1)$
 7:         if $\xi < a$ then
 8:             $\mathbf{X} \leftarrow \mathbf{X}'$
 9:         end if
10:         $j \leftarrow$ GETPIXEL($\mathbf{X}$)                   ▷ pixel corresp. to path
11:         $P_j \leftarrow P_j + 1/N$
12:     end for
13: end procedure
```

2.5.7 Rendering participating media

All rendering algorithms discussed previously have been extended to handle participating media. This includes photon mapping [Jensen and Christensen 1998], bidirectional path tracing [E. P. Lafortune and Y. D. Willems 1996], and Metropolis Light Transport [Pauly et al. 2000]. Virtual ray/beam lights [Novák et al. 2012b; Novák et al. 2012a] generalize the concept of virtual point lights for participating media. The point-based kernel density estimation was generalized for beams and planes [Jarosz et al. 2008; Bitterli and Jarosz 2017], and these estimators have been unified with point estimators and bidirectional path tracing using the general framework of multiple importance sampling [Křivánek et al. 2014].

In this thesis, we will focus solely on path tracing to handle participating media. Enabling path tracing to handle participating media requires to add the functionality of sampling the phase function, and sampling a distance to the next scattering event in the medium, also called free path sampling. Free path sampling is usually done proportionally to transmittance and tends to create short path segments in dense and long path segments in thin media.

The most general way to accomplish this is Woodcock or Delta tracking [Woodcock et al. 1965; Raab et al. 2008; Kutz et al. 2017]. However, this procedure cannot compute the transmittance up to the sampled distance explicitly. Apart from this implicit method, free path sampling can also be implemented by ray marching until the transmittance falls below a stochastically chosen threshold [Pharr et al. 2017]. However, ray marching with fixed step sizes is biased.

Regular tracking [Amanatides and Woo 1987] is a similar approach, which be used if the volume is given in a voxel representation. Apart from the voxel discretization, regular tracking is unbiased. It analytically integrates transmittance along a ray by steping from voxel boundary to voxel boundary. This analytical integration is also possible with trilinear interpolation [Szirmay-Kalos et al. 2011]. In chapter 5 we give more details on regular tracking and how volumetric light sources can be handled efficiently in the path tracing framework.

2.6 Tristimulus and spectral rendering

Until now, we were describing and calculating the total transported energy or flux over all wavelengths and did not differentiate between the wavelength of individual photons. However, to render colored images, we need to resolve wavelength information during the simulation of light transport.

In the real world, light consists of electromagnetic radiation of different wavelengths. The distribution of energy over wavelength can be expressed in the form of a *spectral energy density function* or *spectrum*

$$s(\lambda) : \mathbb{R}^+ \to \mathbb{R}^+ \tag{2.133}$$

which is a function that assigns an energy value to a wavelength. We differentiate between *emission spectra*, describing the emitted light of a light source, *reflectance spectra*, describing the reflective properties of a material, and *responsivity spectra*, which describe the sensitivity of a sensor to light that hits the sensor.

The human eye has three different types of sensors, called *cone cells* or *cones*, which are sensitive to different parts of the wavelength domain and therefore enable the perception of color. These cones are called S-cones, M-cones, and L-cones because they are most sensitive to light with a short (≈430 nm), medium (≈530 nm), and long (≈570 nm) wavelength respectively. Since the sensitivity of the cones to specific wavelengths can differ from person to person, the international commission on illumination *Commission Internationale de l'Éclairage* defined the *CIE standard observer* to represent the idealized average chromatic response of a human. It consists of the three responsivity spectra $\bar{x}$, $\bar{y}$, and $\bar{z}$, which are called *color matching functions* [Wright 1928; T. Smith and Guild 1931] and are shown in figure 2.22. We follow the recommendations given in [CIE 2004] and use the 1931 2° 5 nm step color matching functions defined over the abridged interval of wavelengths $\Lambda = [380, 780]$ nm. The color matching functions can be interpreted as three different sensors or light detectors with a linear responsivity. They produce a corresponding *tristimulus value* XYZ, which, in turn, represents a color. The tristimulus value for a given spectrum $s(\lambda)$ can be computed by integrating the product of spectrum and color matching function over the wavelength domain, i.e.,

$$X = \int_\lambda s(\lambda)\bar{x}(\lambda)\,\mathrm{d}\lambda \tag{2.134}$$

$$Y = \int_\lambda s(\lambda)\bar{y}(\lambda)\,\mathrm{d}\lambda \tag{2.135}$$

$$Z = \int_\lambda s(\lambda)\bar{z}(\lambda)\,\mathrm{d}\lambda. \tag{2.136}$$

These inner products represent a projection of a spectrum into three-dimensional space, and converting a spectrum to a tristimulus value is accompanied by a loss of information. This projection also explains *metamerism*, i.e., the fact that an observer perceives many different spectra as the same color.

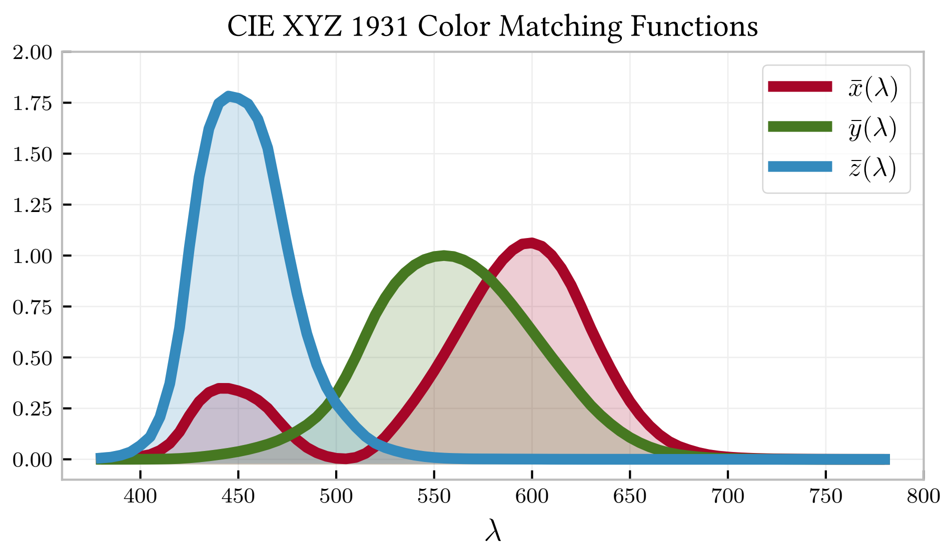


Figure 2.22: This illustration shows the color matching functions of the CIE standard observer. They can be interpreted as the response curves of three linear light detectors and determine the XYZ tristimulus value for an arbitrary spectrum.

The resulting space of all tristimulus values that can be obtained by the color matching functions is called the *CIE XYZ* color space. It contains all color sensations that can be perceived by an average person.

A closely related and more intuitive color space is the *CIE xyY* color space. In CIE xyY, a tristimulus value is separated into a chromaticity xy and a brightness Y. The Y parameter in the CIE XYZ color space corresponds to the luminance and is the same in XYZ and xyY. The chromaticity values x and y are determined by a projection of the corresponding XYZ values onto the plane $X + Y + Z = 1$, i.e.,

$$x = \frac{X}{X + Y + Z} \tag{2.137}$$

$$y = \frac{Y}{X + Y + Z} \tag{2.138}$$

These values span a two-dimensional subspace encompassing all chromaticities that are visible to the average human. It is called the *CIE chromaticity diagram* and is shown in figure 2.23. This horseshoe-shaped set of chromaticity values is also referred to as the *gamut of human vision*. The boundary of the gamut is called the *spectral locus* and corresponds to *monochromatic* light, i.e., light of a single wavelength. Spectra near the spectral locus are very narrow and peaky and become wider and smoother towards the middle. This

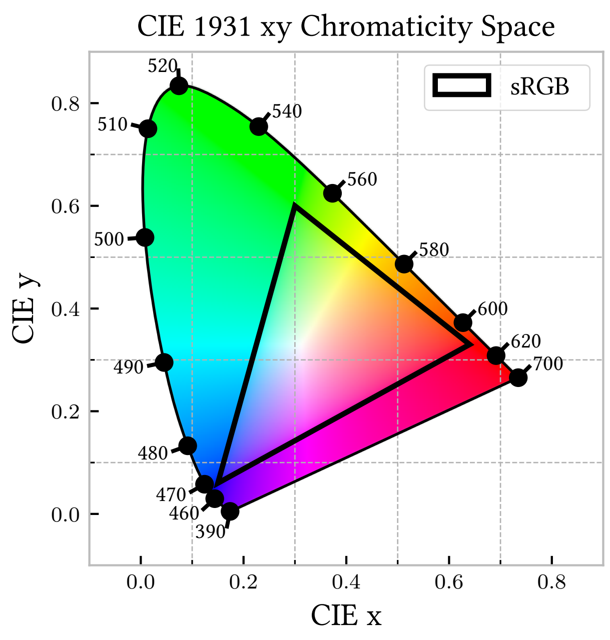


Figure 2.23: The CIE 1931 chromaticity diagram along with the gamut of the sRGB color space. The curved boundary of this horseshoe-shaped form is the spectral (or monochromatic) locus, with wavelengths shown in nanometers. This diagram entails all chromaticities that are visible to the human observer, but because of the limited output gamut of a computer display not all chromaticities are actually displayed.

means the corresponding colors become less saturated towards the center with pure white at $x = y = 1/3$.

2.6.1 RGB color spaces

The three different cone types of the human visual system and its linearity motivate color models based on the additive or subtractive mixture of a (small) set of primary colors. For displaying digital images on output devices such as monitors or television screens, the additive RGB color model with three primary colors is the most suitable.

Many color spaces are based on the RGB color model, which mostly differ in three primaries and *white point*. The white point corresponds to an illuminant with a particular emission spectrum. It is the chromaticity of an object that appears white under this illuminant for a human observer. We will only consider the *CIE illuminant D65*, corresponding to daylight and the equal energy *CIE illuminant E*, which has a constant spectrum.

The *CIE RGB color space* is defined by three monochromatic primary colors of 700 nm (red), 546.1 nm (green), and 435.8 nm (blue) and an equal energy white point E. The *gamut* of the CIE RGB color space is all colors inside the triangle that is defined by its primaries.

There are many other RGB color spaces such as sRGB and its linear (i.e., non-gamma corrected) variant Rec.709 [ITU 2002] or Adobe RGB. The progress on display technology also leads to an increased demand for RGB color spaces with a wider gamut, such as Adobe Wide Gamut RGB, DCI-P3, or ACES. This ever-growing number of color spaces, combined with the dependence of rendering results on the particular color space used, requires rendering systems to carefully account for the different color spaces and their respective white points [Agland 2014]. Additional care has to be taken if material properties such as albedo are specified by an image texture, which stores tristimulus values in a particular color space. As we will see in chapter 3, depending on the color space and values of a texture, the corresponding reflectance properties might not even be physically plausible under certain assumptions (see chapter 3).

CIE Lab

The CIE Lab color space [CIE 2008] is a perceptually uniform color space with respect to human color perception, which means that the distance of two colors in CIE Lab corresponds to the perceived difference. This is convenient because it allows measuring color differences with the standard euclidean distance metric. In CIE Lab color is separated into its lightness (L) and red-green (a) and blue-yellow (b) color component, respectively.

2.6.2 Tristimulus rendering

The most common way to handle color in image synthesis is tristimulus rendering, where light transport is computed for the three RGB primaries simultaneously. Using RGB primaries is convenient because input data such as textures are usually given in a RGB format (or RGBA if opacity is also specified). It is also very intuitive to define the color of a light source or the reflectance of an object using RGB tristimulus values. In contrast to spectral rendering described below, tristimulus rendering does not exhibit color noise and is thus more suitable for preview or interactive rendering. However, a problem arises when an algorithm has to make a wavelength-dependent decision, such as determining the refraction direction when hitting a glass object. Due to dispersion, any chosen direction will only be valid for one wavelength. For such lighting phenomena, it is often more convenient to use spectral rendering where a path has a unique wavelength associated with it. Another difficulty of tristimulus rendering is that the rendering result depends on the choice of the working color space in a nonlinear way [Agland 2014].

2.6.3 Spectral rendering

Spectral rendering helps to simplify color management and wavelength-depend decisions by simulating light transport either using full spectra [Meyer 1988; Peercy 1993; Sun et al. 1999] or by random sampling of a particular wavelength for a path. In this work, we will only consider the latter approach, which extends the dimensions of the Monte Carlo integration to the wavelength domain. Wavelengths are sampled randomly (usually uniformly between 380 nm and 780 nm), and a path is constructed for this particular wavelength. Having a specific wavelength for a path has the benefit that wavelength-dependent effects such as dispersion or scattering for participating media can be handled in a straight forward and simple manner. It is also possible to handle fluorescence with a spectral path tracer [Mojzík et al. 2018], but this effect is out of the scope of this thesis. A spectral renderer can also handle more complex illumination, such as a fluorescent light whose emission spectra cannot be faithfully represented using tristimulus colors since they have most of their energy in a narrow band of wavelengths.

A drawback of spectral rendering is that input such as textures given as RGB values have to be upsampled to a complete spectrum. Due to metamerism, this is not a trivial task since many possible spectra will result in the same tristimulus color when converted back. Moreover, as we will see, the choice of a spectrum influences light transport and, therefore, the rendered image. Another drawback of spectral rendering is that it increases the variance of a Monte Carlo estimator by introducing an additional integration dimension. If unconverged, the resulting images can exhibit distracting color noise. This drawback can be alleviated by computing light transport for several wavelengths simultaneously and combining the results using multiple importance sampling. Hero-wavelength sampling [Wilkie et al. 2014] chooses one wavelength randomly, which is then used for all wavelength-dependent decisions during path construction. It additionally performs light transport for several other wavelengths that are equidistantly distributed in the wavelength domain. Finally, some light transport algorithms are not well suited for spectral rendering. For example, many-light rendering would require many more VPLs to cover the wavelength domain. This would also be contrary to the improvement for many-light rendering presented in chapter 4.

Tristimulus and spectral rendering are both viable approaches for image synthesis. In chapter 3 we will analyse the differences between spectral and RGB transport, and we will consider an often overlooked issue with ensuring the physical plausibility of reflectances if defined as a RGB value in some potentially wide-gamut color space.

2.6.4 Spectral upsampling

There are several methods to upsample tristimulus input values in a spectral rendering system. The method of Smits [Smits 1999] uses the Rec.709 primaries adapted to white point E. Therefore, it is restricted to sRGB. It uses several precomputed piecewise constant spectra that are optimized for smoothness. The resulting spectra can have values greater than 1, which will be clipped to 1 if necessary to guarantee energy conservation. By design, this method depends on the sRGB color space and does not work well for color spaces with a wider gamut.

MacAdam [MacAdam 1935b; MacAdam 1935a] implicitly defines a XYZ to spectrum conversion that results in a spectrum with the highest possible brightness for a color with a specific saturation. These spectra are defined as a single box function wrapping around the wavelength domain and are therefore very simple and can be stored in a compact way. However, these spectra, designed for printing applications, are not smooth, and using them as reflectance spectra in physically-based rendering can result in significant color shifts in the indirect illumination where light bounces multiple times [Meng et al. 2015].

The method in [Meng et al. 2015] precomputes smooth spectra similar to Smits. However, the spectra are precomputed for a discretization of the complete chromaticity diagram. It uses a simple linear interpolation to compute a spectrum for arbitrary XYZ values. The benefit of this method is that it works on almost the whole gamut. It is also optimized for round trips, which means that the XYZ tristimulus value of an upsampled spectrum matches the original XYZ value with negligible error. The method aims to compute naturally looking or physically plausible reflectance spectra. Since these spectra are generally smooth [Maloney 1986], the method uses a nonlinear optimization procedure that maximizes the smoothness of the precomputed spectra.

A data-driven approach is proposed in [Otsu et al. 2018], which uses a set of measured spectra that are clustered and represented in a compact way using principal component analysis (PCA). The precomputed basis spectra of the corresponding clusters are combined as a weighted sum to upsample a XYZ value. The resulting spectra are smooth because of the smoothness of the underlying measured spectra and because of the PCA. However, the upsampled spectra can have negative values that have to be clamped to 0, and round trips can have noticeable error.

The method in [Jakob and Hanika 2019] models reflectance spectra as a nonlinear space of functions with three parameters and allows converting the parameters to a continuous spectrum quickly.

A Fourier domain representation of spectra, based on the theory of moments, is introduced in [Peters et al. 2019]. This method provides a compact way of storing spectra as a small number of coefficients that can also be quantized effectiveley.

Spectral Energy Conservation

When upsampling tristimulus input to reflectance spectra used to describe surface reflection (BSDF), energy conservation has to be considered. Energy conservation has to hold per wavelength, e.g., in the case of a diffuse BSDF, the condition

$$1 \geq \int_{\mathcal{S}^2} \frac{\rho(\lambda)}{\pi} \, \mathrm{d}\boldsymbol{\omega}^\perp = \rho(\lambda) \int_{\mathcal{S}^2} \frac{1}{\pi} \, \mathrm{d}\boldsymbol{\omega}^\perp = \rho(\lambda). \tag{2.139}$$

has to hold for every $\lambda \in \Lambda$. Similarly, the coefficients of other BSDF models can be bound to ensure energy conservation. The aforementioned upsampling methods do not guarantee energy conservation, and the resulting spectra have to be modified to ensure the convergence of the light transport simulation. Chapter 3 discusses several approaches to modify spectra in a way that they are energy-conserving.

3 Physically meaningful tristimulus rendering

Tristimulus and spectral rendering are two ways of handling colors in Monte Carlo rendering systems. Even though spectral rendering is conceptually simple and can handle more illumination effects, it is not always the preferred choice over tristimulus rendering. Many-light methods, for example, rely on tristimulus light transport to be efficient. Also, the lack of color noise makes tristimulus rendering more appealing for preview or interactive rendering applications. However, tristimulus rendering has some lesser-known intricacies. Even though light transport converges mathematically using tristimulus rendering, the input color values can be physically implausible under certain assumptions. This chapter discusses this topic in detail and proposes some methods to ensure physically meaningful results when using tristimulus rendering.

3.1 Introduction

Apart from geometry and material information, the input into a physically-based rendering system usually consists of information about colors, often given in the form of tristimulus values either individually or as image textures.

For wide-gamut color spaces, the chromaticities of tristimulus values can be close to the spectral locus. Such tristimulus values do not necessarily represent physically valid reflectances in the sense that they are too saturated and bright at the same time [Schrödinger 1919]. Reproducing such colors would only be possible with additional emission, for example, in the form of fluorescence.

MacAdam [MacAdam 1935b; MacAdam 1935a] analyses the upper limit to the brightness a color with a specific color saturation can have. This upper limit results in the *solid of valid reflectances*, which defines the subset of colors that can have an energy-conserving spectrum. However, even if a color is inside the solid of valid reflectances, problems can arise with tristimulus rendering. Near the boundary of the solid of valid reflectances, the corresponding spectra become peakier. In contrast, naturally occurring reflectance spectra

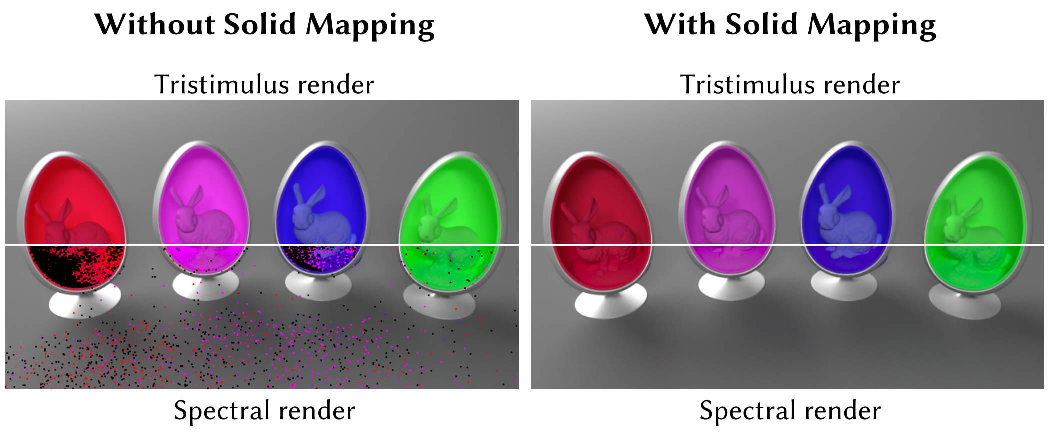


Figure 3.1: Four chairs with diffuse reflectance defined as Rec.709 tristimulus values. The results obtained from a tristimulus path tracer (top left) exhibit an unnatural glow in the indirect illumination behind the bunnies. A spectral path tracer reveals why (bottom left): to reproduce the same color saturation and brightness, *physically plausible* (i.e., smooth and energy-conserving) reflectance spectra must violate energy conservation, causing the render to diverge. With the solid mapping techniques we propose, color values can be modified in a way that both the tristimulus and the spectral render produce physically plausible results (right images).

are generally smooth [Maloney 1986]. Smooth spectra distribute their energy over a larger wavelength domain, and if they have the same overall brightness as MacAdam's spectra, the corresponding colors are necessarily less saturated.

For example, figure 3.1 shows four chairs with diffuse reflectances defined in Rec.709 $\in [0,1]^3$ rendered using tristimulus and spectral rendering, respectively. In the results rendered with tristimulus values (top left), the indirect illumination seems to glow, for example, in the chairs behind the bunnies. A spectral path tracer reveals why (bottom left): to reproduce the same color saturation and brightness, smooth reflectance spectra must violate energy conservation, which causes the spectral render to diverge.

The Grid-based upsampling method [Meng et al. 2015] optimizes spectra for smoothness (and low round-trip error), to generate spectra that behave similar to naturally occurring reflectance spectra. We use this method to define the *solid of natural reflectances* analogously to the solid of valid reflectances. This solid contains all colors for which a *physically plausible* (i.e., smooth and energy-conserving) reflectance spectrum exists.

To ensure physically meaningful tristimulus renderings, we propose three *solid mapping* techniques, which project colors outside the solid of natural reflectances to colors inside the solid of natural reflectance.

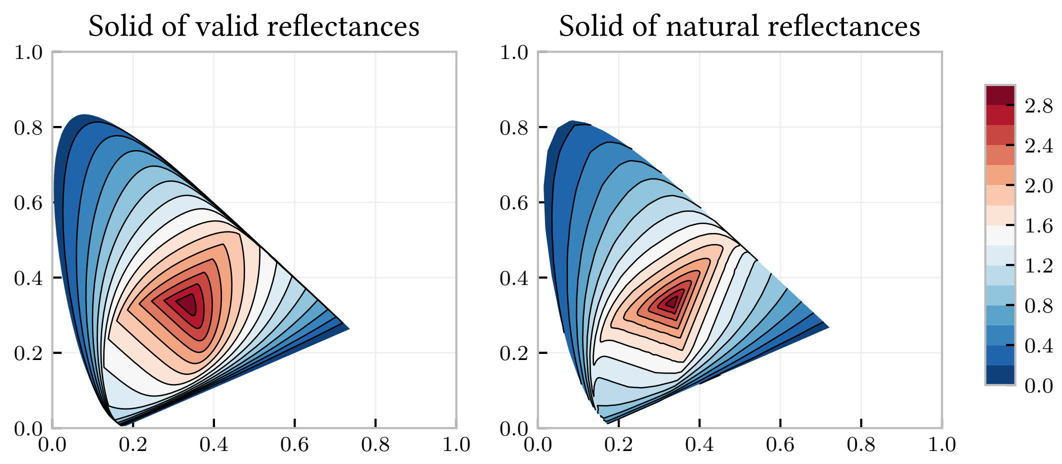


Figure 3.2: This illustration shows the largest theoretically possible solid of valid reflectances (MacAdam's limit) (left) and the solid of natural reflectances. The solid of natural reflectances corresponds to smooth spectra similar to those occurring in natural materials (right). The contour plots visualize the maximal brightness $b(x, y)$ over the base of the (x, y) chromaticity diagram.

3.2 Solid of natural reflectances

Every upsampling method can be used to determine a maximum brightness $b(x, y)$ that a tristimulus value with chromaticity (x, y) can have without violating energy conservation. Determining the maximum brightness requires upsampling the corresponding tristimulus value to a spectrum s with the particular method, and calculating b such that $b \cdot s(\lambda) \leq 1 \forall \lambda$. Following MacAdam [MacAdam 1935a], we define a *solid of reflectances* as the subspace of XYZ tristimulus values with brightness $X + Y + Z < b(x, y)$, where (x, y) is the corresponding chromaticity. The difference to [MacAdam 1935a] is the definition of brightness as $b = X + Y + Z$ instead of the visual efficiency Y.

The largest possible solid of reflectances can be obtained by using MacAdam's upsampling method since the resulting box-spectra determine the maximal brightness of a tristimulus value with a particular saturation. This solid is called the *solid of valid reflectance* since every tristimulus value outside this solid can not be produced by an energy-conserving spectrum.

In contrast to MacAdam's spectra, naturally occurring spectra are generally smooth [Maloney 1986]. Therefore we define the *solid of natural reflectances* as the solid of the Grid-based upsampling method [Meng et al. 2015]. The solid of natural reflectances contains all tristimulus values that result in a smooth and energy-conserving spectrum. Figure 3.2 visualizes the solid of valid and the solid of natural reflectances.

3.3 Solid mapping

Similar to two-dimensional gamut mapping, i.e., mapping colors from a wide gamut working space to a smaller gamut of an output device (e.g., printing [Stone et al. 1988]), we map XYZ tristimulus values to values inside a solid of reflectances. We call this procedure *solid mapping* since it is conceptually similar to gamut mapping but in three-dimensional space. In the following, we will denote the upsampled spectrum to a corresponding (x, y) chromaticity as

$$s(x, y, \lambda), \lambda \in \Lambda.$$

Scaling

To scale the spectrum uniformly such that its maximum is mapped to 1 we can use maximal brightness values $b(x, y)$ directly as a scaling factor:

$$S(s(x, y, \lambda)) = s(x, y, \lambda) \cdot b(x, y), \text{ or} \tag{3.1}$$

$$S(X, Y, Z) = (X, Y, Z) \cdot b(x, y). \tag{3.2}$$

This solid mapping preserves smoothness and chromaticity. However, it may drastically alter brightness.

Clipping

Clipping the spectrum at a value of 1 results in a valid reflectance spectrum:

$$C(s(x, y, \lambda)) = \min\{1, s(x, y, \lambda)\}. \tag{3.3}$$

This method is particularly useful for spectral rendering because it can be performed on the fly for a specific wavelength, independent without additional information. However, this method potentially introduces kinks into the spectrum.

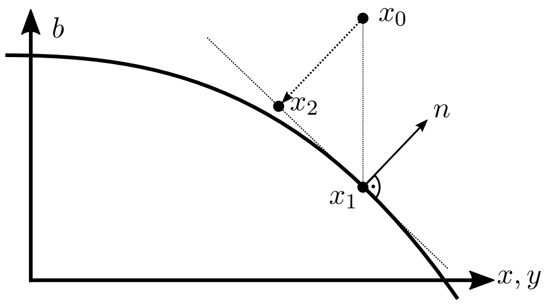


Figure 3.3: This Figure illustrates the predictor/corrector method for $\min \Delta E$ solid mapping. We start with the predictor step at point x_0 and perform a projection onto the solid's boundary at x_1 by scaling with $b(x, y)$. Then, the surface's normal n is estimated at x_1 for correction. The result of this predictor/corrector step x_2 is found by projecting x_0 onto the plane with the estimated normal. The point x_2 has the shortest distance in CIE Lab space to the first-order approximation of the solid's surface. The process is repeated and usually converges after 3-10 iterations.

Minimal distortion

The *method of minimal distortion* ($\min \Delta E$) is more involved than the other two and useful as a reference. It achieves minimum color shift (in terms of CIE 1976 ΔE error) by changing both brightness and chromaticity:

$$M(s(x, y, \lambda)) = s(x', y', \lambda), \text{ with} \tag{3.4}$$

$$(x', y') = \arg\min \left\| (X', Y', Z') - (X, Y, Z) \right\|_{\Delta E}. \tag{3.5}$$

CIE 1976 ΔE is the euclidean distance in the CIE Lab color space [CIE 2008]. We use a predictor/corrector method in CIE Lab space to geometrically search for the point on the surface of the solid with minimal distance to the original point.

The predictor projects the current CIE Lab point onto the surface of the solid by converting it to XYZ, scaling by $b(x, y)$, and converting back to CIE Lab. The corrector estimates the normal of the solid's surface in CIE Lab space at this point using finite differences and projects the input point onto the plane defined by this normal (see figure 3.3). The process is repeated and usually converges after 3-10 iterations.

Note that the brightness values $b(x, y)$ can be precomputed and stored in an image texture for a large number of (x, y) chromaticity values. This precomputation allows scaling and $\min \Delta E$ mapping of XYZ tristimulus values without explicit upsampling. All three strategies can be performed using MacAdam's or the Grid-based upsampling method.

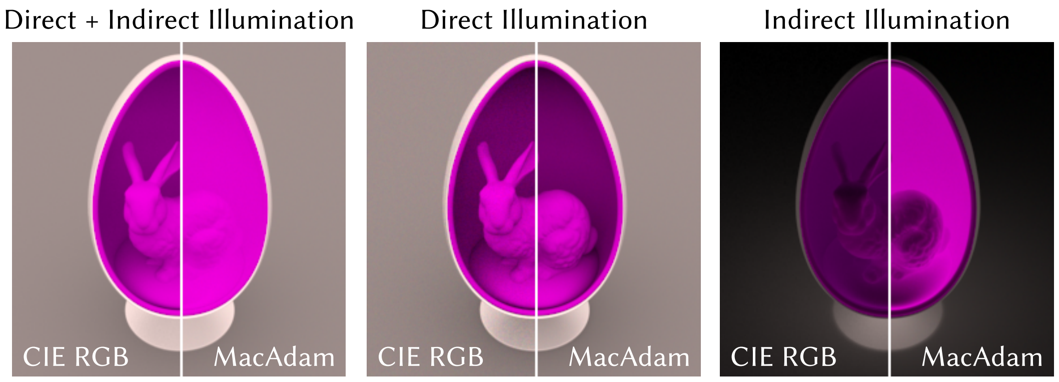


Figure 3.4: Tristimulus renderings with CIE RGB reflectance of $(0.440, 0.016, 0.510)$ for the bunny and inner chair and a spectral rendering using the corresponding MacAdam spectrum. While the direct illumination matches, the indirect illumination behaves quite differently for spectral and tristimulus rendering. For each surface interaction, the values of the CIE RGB tristimulus become smaller, whereas the spectral values remain 1 for wavelengths $\lambda \in [455, 620]$ nm.

Matrix transforms

A common approach in gamut mapping applications consists of applying several matrix transformations to map values from one gamut into another while retaining as much of the appearance as possible [Stone et al. 1988]. However, these methods usually modify all input values and not only the values outside the gamut. These gamut mapping methods want to preserve the color differences of input values in the target gamut. In contrast, we only want to modify the values outside the gamut. This ensures energy conservation and naturally looking reflectance for physically based rendering with indirect illumination. However, evaluating and modifying this technique for solid mapping could be an exciting topic for future research.

3.4 Results

Tristimulus vs spectral rendering using MacAdam's spectra

The primaries of CIE RGB and wide-gamut RGB color space have mono-wavelength spectral intensities (435.8 nm, 546.1 nm, 700 nm, and 450 nm, 525 nm, 700 nm, respectively). Tristimulus rendering in these color spaces corresponds to spectral rendering with three wavelengths. Therefore, they are the most physically meaningful color spaces for tristimulus rendering. Unfortunately, tristimulus rendering in these color spaces also produces results that are very different from simulating the full spectrum, as can be seen in fig-

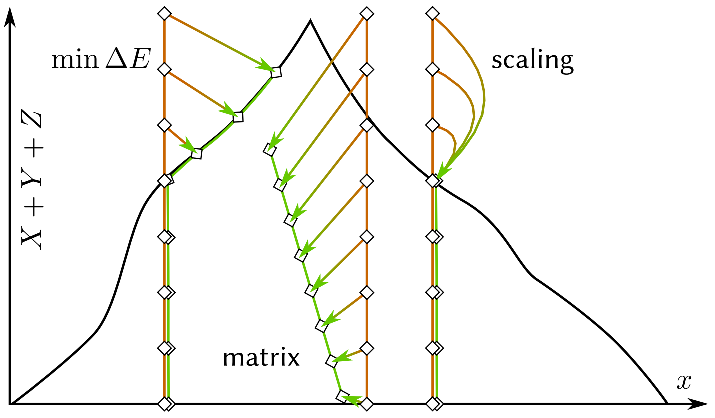


Figure 3.5: Side view of the solid of natural reflectances, with three mapping strategies, illustrating the smoothness of the mapping with respect to albedo. Every line represents the input of a constant chromaticity coordinate (x, y) at different albedo and how the mapping affects these values. We illustrate that matrix transforms used in gamut mapping applications modify all input values which we do not want for our application.

ure 3.4. The same issue applies to other RGB working space [Mansencal et al. 2015] and is an inherent limitation of tristimulus transport. However, we can now provide values that correspond to natural reflectances.

Continuity of the Mapping

In figure 3.5, we schematically illustrate three mapping strategies ($\min \Delta E$, scaling, and matrix transforms [Stone et al. 1988]). All three are C^1 continuous, but $\min \Delta E$ and scaling introduce kinks in the derivative. Scaling and $\min \Delta E$ map multiple tristimulus values outside the solid of valid reflectances to the same output. This ambiguity might be undesirable for general gamut mapping applications. The matrix transform does not introduce these drawbacks, but it might result in a much smaller output solid and also modify perfectly valid input reflectances. For maximum color accuracy, we think it is best to leave valid reflectances unchanged and inform content creators instantly about problematic tristimulus values during texture painting.

Mapping to Valid Reflectances

Figure 3.7 shows the importance of solid mapping for reflectances. Even at moderate brightness of $X + Y + Z = 1$, the chromaticity of the red Rec.709 primary does not define valid reflectances and needs to be adjusted. Figure 3.7 also shows the steep falloff in brightness for color with $X + Y + Z = 3$ near the white point $(1, 1, 1)$. The more complicated method optimizing for ΔE pays off for very saturated tones close to the spectral locus, especially. The Grid-based solid mapping methods show some grid interpolation artifacts. To avoid these grid interpolation artifacts, we propose that even in a spectral renderer, the input should be solid mapped before upsampling using MacAdam's limit. Alternatively, the grid resolution can be increased to diminish these artifacts further.

Surface Transport

The scene in figure 3.1 and figure 3.6 consists of four colored chairs, with diffuse reflectance defined as Rec.709 tristimulus values (1.00, 0.01, 0.10), (1.00, 0.10, 1.00), (0.10, 0.10, 1.00) and (0.10, 1.00, 0.10).

The first three rows in figure 3.6 show spectral light transport using the indicated upsampling method in each row and the indicated solid mapping method in each column. Note that $\min \Delta E$ corresponds to the mapping into the solid of valid reflectances (MacAdam) for Smits and MacAdam, and the solid of natural reflectances for the Grid-based method.

The bottom three rows show tristimulus transport in Rec.709. The rows are labeled Smits, Grid, and MacAdam, to indicate the underlying set of spectra used for the respective solid mapping techniques, the three tristimulus renders without solid mapping are identical.

The tristimulus renders exhibit unnatural glow in the indirect lighting, especially in the red and green tones. For the Grid and Smits' method, spectral rendering of the scene with these values diverges without solid mapping.

Note that tristimulus transport using solid mapping with Smits' upsampling and clipping (row 4, column 2) diverges as well. This divergence is due to tristimulus values greater than one, caused by the lack of round-trip precision of Smits' method; i.e., upsampling a Rec.709 triple $\in [0, 1]^3$ and converting it back to Rec.709 again will not necessarily result in values within $[0, 1]^3$.

The solid mapped variants show improved tonality, which gives more definition to the shape of the bunny, and a more realistic appearance of the indirect light.

3.5 Conclusion and future work

We showed that today's tristimulus-based rendering pipelines are not physically meaningful in general, even though no energy is produced, and the values per color channel do not increase. That is, the images converge mathematically but have no physical counterpart: no real material could produce such color.

With fluorescent dyes, which have been considered in the corresponding literature concerned with printing, materials can appear brighter and more saturated. However, light transport becomes significantly more complicated to implement in this case, and the effect on color in indirect light is not easily described in closed form.

We define the solid of natural reflectances using the Grid-based upsampling method [Meng et al. 2015]. This method is motivated by the smoothness of naturally occurring reflectance spectra [Maloney 1986] and creates spectra that are optimized for smoothness and low round-trip error.

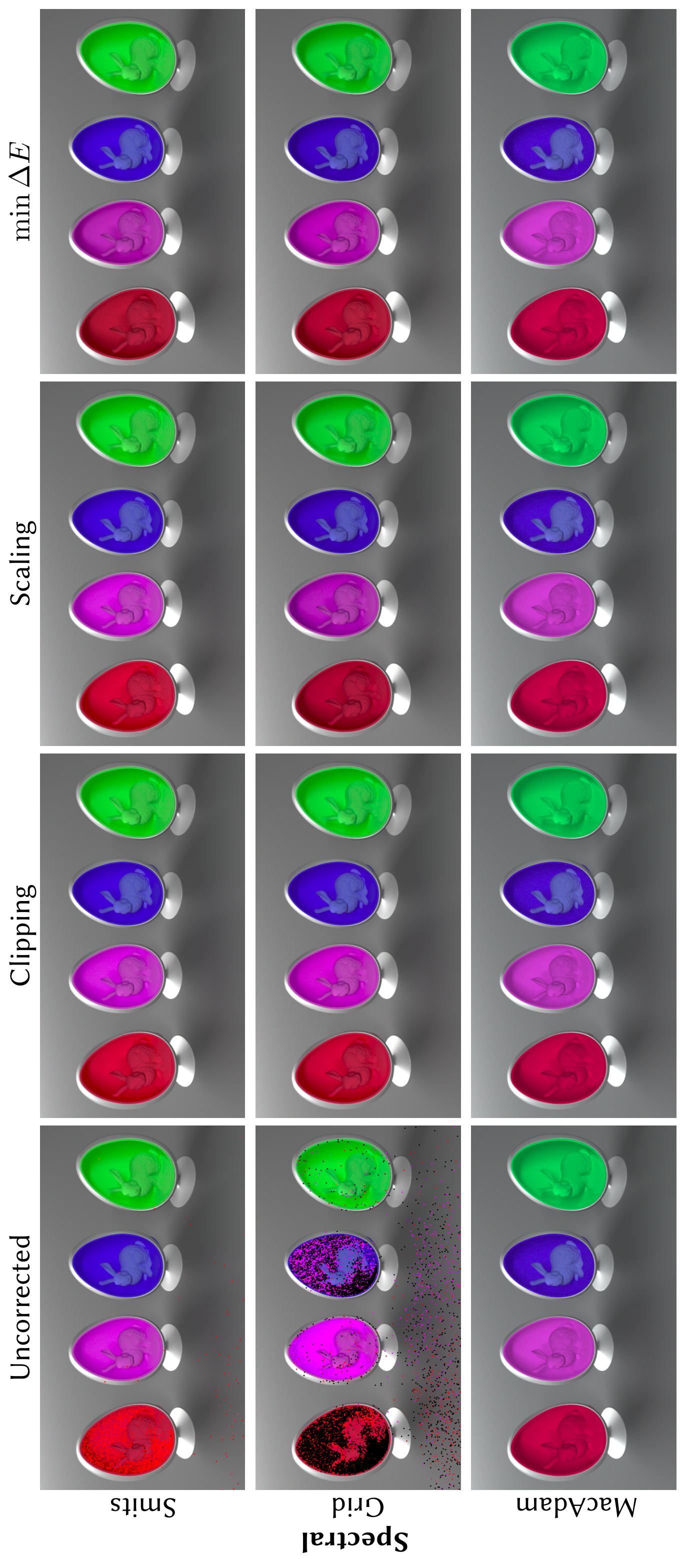
We introduced a set of methods to perform solid mapping to the solid natural reflectances to map arbitrary tristimulus values to physically meaningful values. Physically meaningful means that there is a smooth and energy-conserving spectrum that can produce this color.

The quantization of the Grid-based upsampling method [Meng et al. 2015] comes with slight artifacts when solid mapping has to be performed after upsampling. If this is an issue for an application, it is possible to increase the grid resolution.

The motivation to optimize spectra for smoothness for the solid of natural reflectances is similar is the motivation of Smits [Smits 1999]. However, we can not guarantee that the spectra used are necessarily the brightest, physically plausible reflectance spectra. For example, the data-driven appraoch in [Otsu et al. 2018] uses real measured spectra for upsampling. This method could result in a larger solid of reflectances compared to the Grid-based upsampling method.

Another recent upsampling method optimizes spectra in the three-dimensional domain compared to our approach in (x, y) chromaticity space [Jakob and Hanika 2019]. It employs a low-dimensional parametric model to create spectra similar to MacAdam. These spectra are energy-conserving by design and cover the sRGB gamut perfectly. However, the spectra become similarly peaky to MacAdam's spectra near the spectral locus and potentially represent colors that are too peaky for naturally occurring reflectance spectra.

We hope that in the future, content creation applications will inform artists when they use problematic tristimulus values that could create an unwanted glowing effect. These would be all tristimulus values that are not physically meaningful in the sense that they are too bright and saturated and could only be reproduced by fluorescent dyes in the real world.


min ΔE
Scaling
Clipping
Uncorrected
Smits
Grid
MacAdam
Spectral

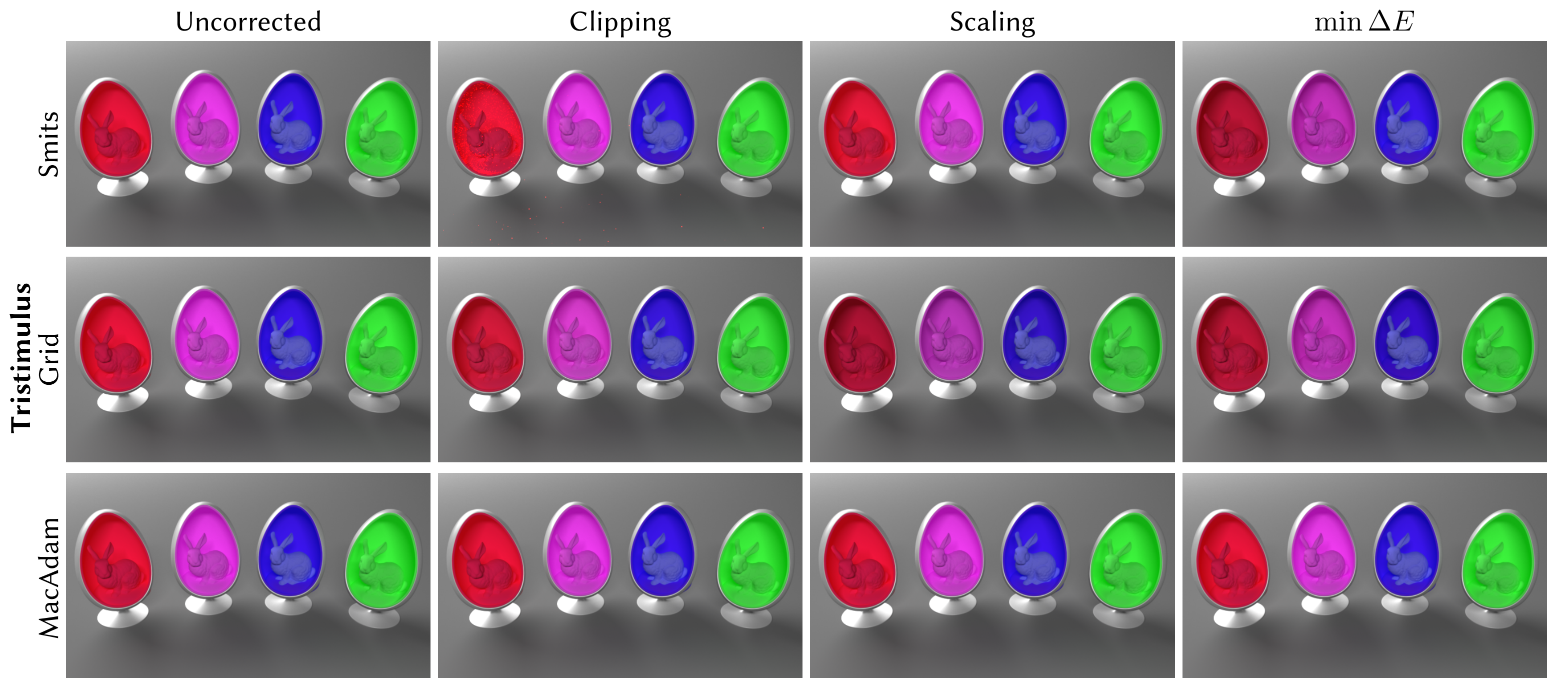


Figure 3.6: In this scene with very saturated color, tristimulus rendering in Rec.709 creates unrealistically bright colors (first column, bottom three rows). Creating spectra with Smits' or the Grid method requires additional solid mapping to fulfill energy conservation (first column, top three rows). We can solid map the RGB input values by clipping, scaling, or with $\min \Delta E$ (columns 3-4, respectively) such that the tristimulus transport images (rows 4-6), as well as the spectral renders (rows 1-3), produce more natural and converging results.

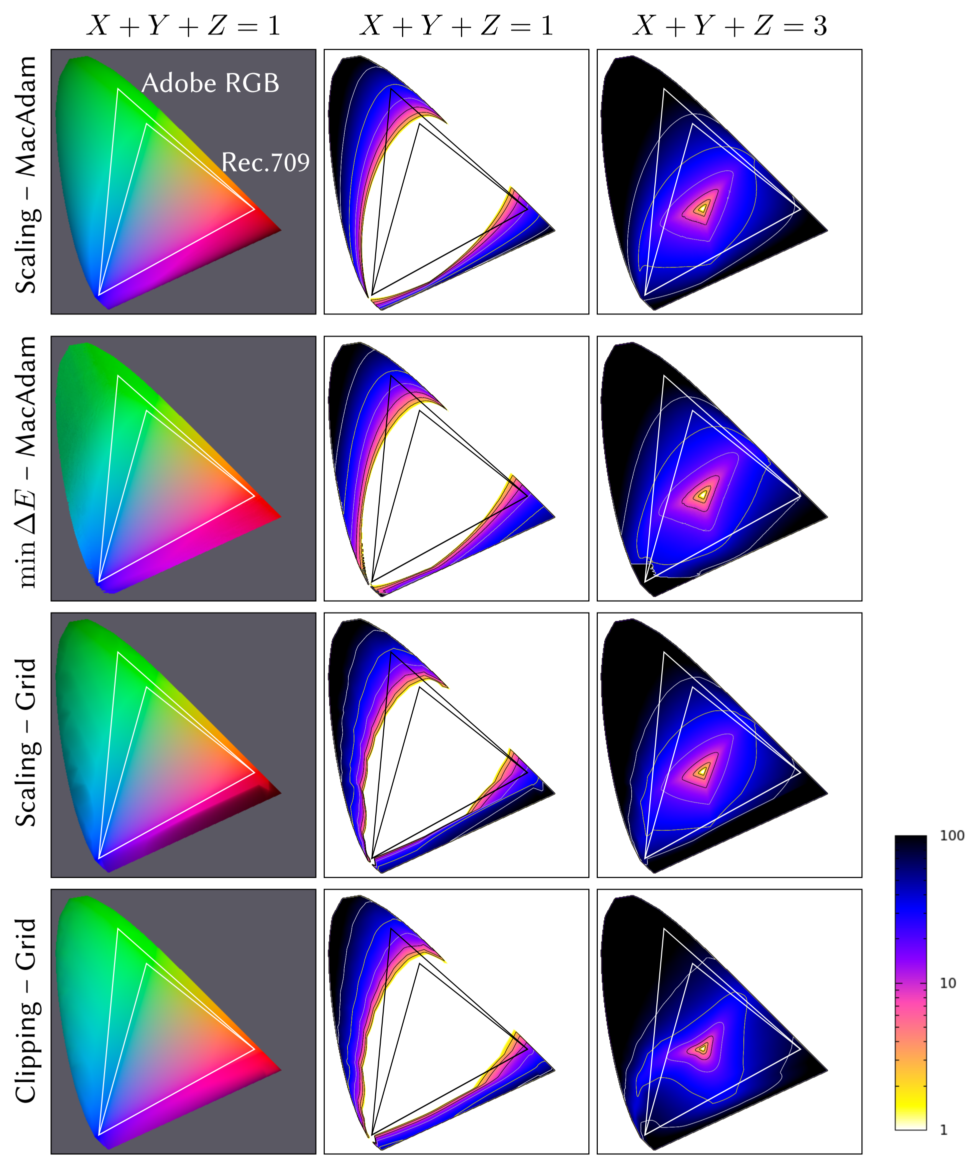


Figure 3.7: Different solid mapping strategies applied to a chromaticity diagram with brightness $X+Y+Z=1$ (left two columns) and $X+Y+Z=3$ (right column). Even for the limited gamut of Rec.709, the red primary is outside the solid of valid reflectances, even at moderate brightness. The difference images visualize logarithmic perceptual distance (CIE 1976 ΔE) to the unchanged input. The first two rows work directly on the XYZ data, the bottom two first convert the tristimulus values to a spectrum using the Grid-based upsampling method [Meng et al. 2015] and then perform scaling or clipping.

4 Many-light rendering with Rich-VPLs

Many-light methods are a family of Monte Carlo rendering techniques, which use a simple form of bidirectional path construction. Many-light methods produce noise-free images, even with a low number of samples. This property makes them especially interesting for preview and interactive rendering. However, to produce artifact-free images, some simplifying assumptions are commonly made. One of these assumptions is that virtual point lights are diffuse, which means they emit their energy equally in all directions. This assumption is especially crucial in scenes with highly glossy materials. However, in this case, the impact of this simplification can also be quite dramatic, and noticeably change the appearance of objects with glossy materials. In this chapter we will investigate the problems of many-light rendering in the presence of highly glossy materials. We propose a novel data-driven global importance sampling technique for choosing locations for virtual point lights. We also present an extension of regular virtual point lights that reuses information gathered by the location sampling step to handle glossy materials more efficiently.

4.1 Introduction

Many-light methods introduced in section 2.5.3 approximate the light transport in a scene by computing the direct illumination from many virtual point light sources (VPLs) and render low-noise images covering a wide range of performance and quality goals. However, they are very inefficient at representing glossy light transport. A VPL on a glossy surface illuminates only a small fraction of the scene, and a tremendous number of VPLs might be necessary to render acceptable images.

To expand the range of effects that many-light methods can render efficiently, we introduce a novel virtual light type, the Rich-VPL, that represents many incident light paths. Encompassing information of multiple light paths increases the efficiency compared to standard VPLs. The emission profiles are, for example, stored in textures and can be pre-filtered and mollified in the angular domain. Mollification reduces artifacts and enables

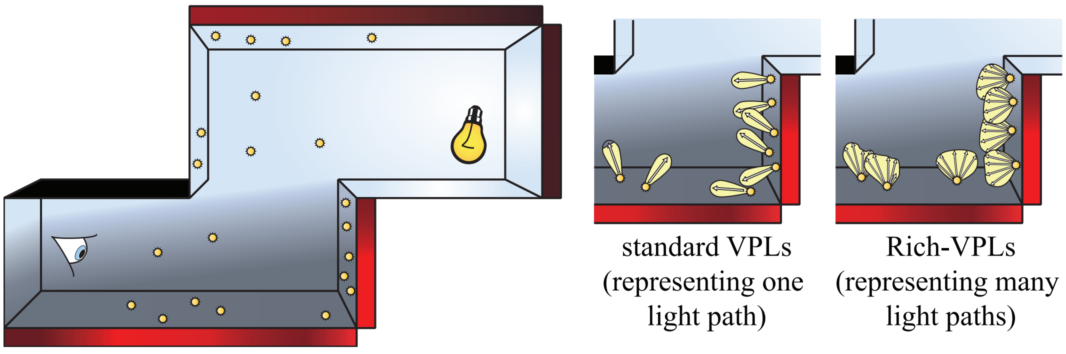

standard VPLs (representing one light path)
Rich-VPLs (representing many light paths)

Figure 4.1: This figure shows a scene with one primary light, several VPLs, and moderately glossy surfaces on the left. The red bars visualize the sensor importance. We describe a method to importance sample VPL locations according to the product of importance and radiance and introduce Rich-VPLs, which represent many light paths and have a more efficient emission profile than standard VPLs (right).

the rendering of near-specular light transport. Rich-VPLs also integrate well into state-of-the-art many-light methods, such as Virtual Spherical Lights [Hašan et al. 2009] and Lightcuts [Walter et al. 2005; Walter et al. 2006; Walter et al. 2012].

Our second contribution is a data-driven global importance sampling technique to improve the placement of (Rich-)VPLs. Ideally, (Rich-)VPLs should be created where they contribute considerably to the image. We propose a method to sample their locations proportional to the densely sampled product of sensor importance [Veach 1998] and radiance. We further introduce an efficient iterative relaxation scheme to obtain (Rich-)VPL locations with blue noise characteristics, which improves the spatial stratification of (Rich-)VPLs considerably.

We demonstrate that many-light methods benefit from Rich-VPLs and the novel placement method. We also show how to directly obtain the emission profiles of Rich-VPLs from the radiance sampled during the placement of VPLs.

4.2 Related work

The entire evolution of many-light methods began with Instant Radiosity [Keller 1997] where the concept of VPLs has been introduced. It has been recently portrayed in a comprehensive state-of-the-art report on this topic [Dachsbacher et al. 2013]. To this end, we concentrate on the most closely related work from this field.

VPL placement

It is obvious that VPLs that do not contribute (significantly) to an image only waste computation. To this end, Segovia et al. [Segovia et al. 2006] trace paths from the camera to place VPLs on surfaces one bounce after the surfaces visible in the image. This idea has been further improved by employing the Metropolis-Hastings algorithm which creates VPL location at positions proportional to the full measurement contribution of one individual path [Segovia et al. 2007]. In contrast, we make sure the location is equally good for all shading points by using the importance from the sensor similar to the importance driven photon map of Peter and Pietrek [Peter and Pietrek 1998]. However, our method constructs two independent maps for importons and photons which is simpler to implement and allows reusing the photon map for other parts of our method. Furthermore, their photon shooting is based on a discrete directional probability distribution of importance at every photon interaction. This is costly and requires an excessive number of importons for glossy BRDFs. Georgiev and Slusallek [Georgiev and Slusallek 2010] achieve the same goal by stochastically rejecting VPLs that do not significantly contribute to the image, which is evaluated for a sub-sampled image. However, while being simple to implement, this approach typically generates many candidate VPLs before one VPL is accepted. All three algorithms drive the distribution of VPLs by their contribution to the image, however, we use a significantly denser sampling of sensor importance and light paths. Davidovic et al. [Davidovič et al. 2010] also generate VPLs from the camera, denoted as local VPLs, which are used to locally increase the density of VPLs and to capture short-range interreflections. Local VPLs only contribute to a small portion of the image, and it is assumed that there is no occlusion for these VPLs. The global (long-range) transport is still computed with conventionally generated VPLs and could benefit from our VPL generation method.

Scalability

The general idea of scalable methods is that within a set of VPLs, not every VPL contributes equally to *a shading point*. Typically VPLs are clustered and hierarchically organized such that the aggregate effect of a group of VPLs can be approximated by evaluating a single, brighter, representative VPL. Lightcuts [Walter et al. 2006] determines a cut (the set of clusters) for every shading point using an analytic error bound and a perceptual metric. Lightcuts scales excellently for large numbers of VPLs, however, the error bounds ignore occlusion which results in wasteful cuts in scenes with complex visibility. Multidimensional Lightcuts [Walter et al. 2006] further introduces a hierarchy of shading points to achieve scalable performance for effects such as depth of field or motion blur. Bidirectional Lightcuts [Walter et al. 2012] extends the previous approaches to avoid clamping artifacts and handle a wider range of materials, such as glossy reflections, subsurface scattering and short-range indirect illumination. In all Lightcuts variants, VPLs only represent diffuse reflection. Matrix Row-Column Sampling [Hašan et al. 2007] (MRCS) computes only one global cut for the whole image. Again, the contribution of representatives is estimated using a sub-sampled image. Ou et al. [Ou and Pellacini 2011] combine the idea of locally adapted cuts and MRCS. The underlying observation is that while adaptation is important, many cuts share a certain set of VPLs which can be reused for local cluster refinements. Lastly, Georgiev et al. [Georgiev et al. 2012b] propose to choose the most relevant VPLs for a given shading point based on importance caching. In a pre-process they compute the contribution of all VPLs to the sub-sampled image, and use this cached contribution nearby a shading point as a discrete probability distribution to sample VPLs. Rich-VPLs and our placement can be used with any of the above algorithms for improving the scalability of many-lights rendering, which we demonstrate exemplarily for Lightcuts.

Avoiding the singularity

A constant companion of many-light methods are singularities in the geometry term due to shading with point lights and with glossy BRDFs. The simplest workaround is to clamp the VPLs' contributions; however, this removes short-distance light transport and alters the appearance of materials [Křivánek et al. 2010]. The clamped residual energy can be recovered with bias compensation [Kollig and Keller 2004] which is very costly, or approximate [Novák et al. 2011].

Virtual spherical lights (VSLs) [Hašan et al. 2009] address the source of the singularity directly: they replace the point-to-point shading evaluation by an integration over the solid angle subtended by a spherical light, essentially averaging the BRDFs over the solid angle and distributing the energy over nearby surfaces (visibility is still evaluated point-to-point). VSLs can be combined with scalable methods, such as Lightcuts or MRCS, and also with our Rich-VPLs.

4.3 Theory and motivation

In this section, we start from the path integral formulation of light transport and derive the motivation of Rich-VPLs. When computing light transport, we strive to sample light paths

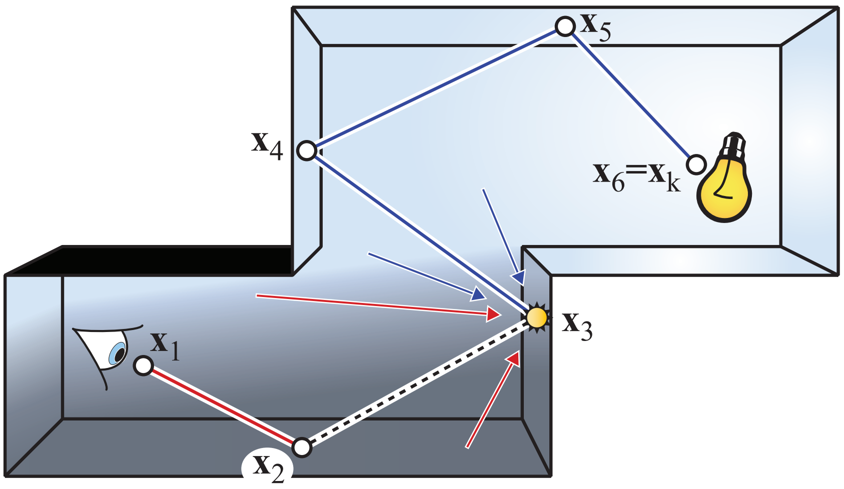


Figure 4.2: Notation for a transport path connecting the camera x_1 to the light source x_k. A VPL is placed at x_3 illuminating a visible surface x_2. The probability of placing a VPL should depend on all importance arriving a surface point (indicated by the red arrows) and it should ideally represent not just one incident light direction in its emission (blue arrows).

$\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ with vertices x_1 at the camera and x_k on a light source proportional to the measurement contribution function (equation 2.51):

$$f_j(\mathbf{X}) = W_j(\mathbf{x}_1, \boldsymbol{\omega}_{\mathbf{x}_2\to\mathbf{x}_1}) \left(\prod_{i=1}^{k-1} G(\mathbf{x}_i, \mathbf{x}_{i+1}) \right) \left(\prod_{i=2}^{k-1} f_r(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) \right) L_e(x_k), \tag{4.1}$$

where W is the importance of the j-th sensor (pixel), and $L_e(x_k)$ the emitted radiance. We are ignoring participating media and transparent surfaces in this chapter for simplicity and therefore the T term of equation 2.51 is missing and we replaced the general scattering distribution function f with the BRDF f_r. For the following considerations, we assume a path length of $k \geq 4$ and write $f_j(\mathbf{X})$ as a product of the sensor importance $W(\mathbf{x}_3)$ reaching $\mathbf{x}_3$, the BRDF at $\mathbf{x}_3$, and the radiance $L(\mathbf{x}_3)$ reaching $\mathbf{x}_3$, i.e.,

$$f(\mathbf{X}) = W(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) f_r(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) L(\mathbf{x}_3, \mathbf{x}_4, \ldots, \mathbf{x}_k), \tag{4.2}$$

with

$$W(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = W_j(\mathbf{x}_1) G(\mathbf{x}_1, \mathbf{x}_2) f_r(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) G(\mathbf{x}_2, \mathbf{x}_3) \tag{4.3}$$

$$L(\mathbf{x}_3, \ldots, \mathbf{x}_k) = L_e(\mathbf{x}_k) \left(\prod_{i=3}^{k-1} G(\mathbf{x}_i, \mathbf{x}_{i+1}) \right) \left(\prod_{i=4}^{k-1} f_r(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) \right). \tag{4.4}$$

This division of terms corresponds to having created a VPL at x_3 with emission

$$L_e(\mathbf{x}_3, \boldsymbol{\omega}) = f_r(\mathbf{x}_3, \boldsymbol{\omega}_{\mathbf{x}_4 \to \mathbf{x}_3}, \boldsymbol{\omega}) L(\mathbf{x}_3, \ldots, \mathbf{x}_k), \tag{4.5}$$

illuminating a surface point $\mathbf{x}_2$ visible to the camera (see figure 4.2). Instant radiosity [Keller 1997] creates VPLs proportional to L by particle tracing from the light sources, while more elaborate methods, e.g., [Segovia et al. 2007; Georgiev and Slusallek 2010], consider the product of W, L and the BRDF at $\mathbf{x}_3$ when creating VPLs. However, they do so by sparsely sampling paths or VPL contributions to the image.

Our work is motivated by the observation that VPLs on glossy surfaces have a narrow emission profile due to the BRDF at $\mathbf{x}_3$, while VPLs illuminating glossy surfaces have a spatially limited contribution to the image due to the BRDF at $\mathbf{x}_2$. In these cases evaluating VPL contributions by sub-sampling in image space often misses important features especially if the BRDF at $\mathbf{x}_3$ and $\mathbf{x}_2$ is glossy. This problem could be alleviated, if the emission of a VPL would account for incident radiance from all possible light subpaths reaching $\mathbf{x}_3$ and if VPL positions would be chosen with a probability density reflecting the importance for all shading points.

To this end, we propose to sample VPL locations proportional to the product of the *total importance* $\hat{W}(\mathbf{x}_3)$ reaching a surface point $\mathbf{x}_3$, i.e.,

$$\hat{W}(\mathbf{x}_3) = \int_{\mathcal{M}} W(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \, \mathrm{d}\mathbf{x}_2, \tag{4.6}$$

and the *total incident radiance* $\hat{L}(x_3)$ reaching x_3, i.e.,

$$\hat{L}(\mathbf{x}_3) = \sum_{k=4}^{\infty} \int_{\mathcal{M}} \cdots \int_{\mathcal{M}} L(\mathbf{x}_3, \cdots, \mathbf{x}_k) \, \mathrm{d}\mathbf{x}_4 \ldots \mathrm{d}\mathbf{x}_k. \tag{4.7}$$

The BRDF at x_3 is omitted in this importance sampling goal as Rich-VPLs account for incident radiance from multiple directions which means that their emission profiles are typically wide enough for efficient shading even on highly glossy surfaces. We could, however, include the maximum BRDF-value into the sampling to prevent creating VPLs on very dark surfaces. In practice, VPLs are not used for shading (near-)specular surfaces as the contribution of a VPL is easily missed. Instead the camera subpath is continued in this case until it reaches a moderately glossy or diffuse surface for which the illumination from the VPLs is then accumulated. Consequently, $\hat{W}$ also includes importance of longer camera subpaths if these begin with consecutive specular interactions.

4.4 Placement of virtual lights

In this section we detail our method for determining the VPL locations according to the aforementioned criteria. We also propose an optional iterative relaxation step that improves the distribution in a post-process.

4.4.1 Importance sampling of VPL locations

In order to sample VPL locations, we need to provide means to compute the total incident radiance $\hat{L}(\mathbf{x})$ and total importance $\hat{W}(\mathbf{x})$ at a surface point $\mathbf{x}$. To this end, we compute a photon map [Jensen 1996] and an importance map [Peter and Pietrek 1998] and estimate the two quantities by performing K-nearest neighbor density estimations with a 2D Epanechnikov kernel, i.e.,

$$\hat{L}(\mathbf{x}) = \frac{2}{\pi\|\mathbf{x} - \mathbf{x}_K\|^2} \sum_{i=1}^{K} \Phi_i \cdot w_i, \tag{4.8}$$

$$\hat{W}(\mathbf{x}) = \frac{2}{\pi\|\mathbf{x} - \mathbf{x}_K\|^2} \sum_{i=1}^{K} \Psi_i \cdot w_i. \tag{4.9}$$

with weights

$$w_i = 1 - \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\|\mathbf{x} - \mathbf{x}_K\|^2}, \tag{4.10}$$

where $\mathbf{x}_i$ is the position of i-th photon (importon), and Φ_i (Ψ_i) is the incident flux (importance) of the i-th photon (importon). We assume the K-nearest neighbors to be sorted, i.e.,

$$\|\mathbf{x} - \mathbf{x}_1\|^2 \leq \|\mathbf{x} - \mathbf{x}_2\|^2 \leq \cdots \leq \|\mathbf{x} - \mathbf{x}_K\|^2. \tag{4.11}$$

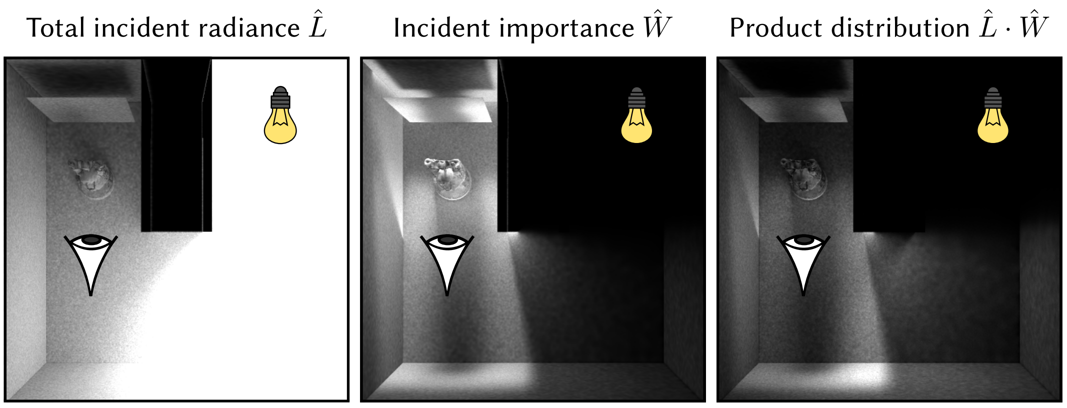

Total incident radiance $\hat{L}$
Incident importance $\hat{W}$
Product distribution $\hat{L} \cdot \hat{W}$

Figure 4.3: This is a qualitative visualization of the distribution of incident radiance $\hat{L}$ and incident importance $\hat{W}$ and their product in this U-Shape Scene. The images are individually tone-mapped for better presentation.

The incident flux of a randomly sampled photon with a path $\mathbf{X} = (\mathbf{x}_3, \dots, \mathbf{x}_k)$ where $\mathbf{x}_k$ is on a light source is

$$\Phi = \left(\prod_{i=3}^{k-2} \frac{G(\mathbf{x}_i, \mathbf{x}_{i+1}) f_r(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2})}{p_A(\mathbf{x}_i)} \right) \frac{G(\mathbf{x}_k, \mathbf{x}_{k-1})}{p_A(\mathbf{x}_{k-1})} \frac{L_e(\mathbf{x}_k)}{N_P p_A(\mathbf{x}_k)}. \tag{4.12}$$

Here, N_P is the number of sampled photon paths and $p_A(\mathbf{x}_i)$ is the probability of sampling $\mathbf{x}_i$ in area measure. The incident importance Ψ of an importon with path $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ where $\mathbf{x}_1$ is on the sensor is evaluated similarly. In contrast to the original work [Peter and Pietrek 1998], we store importons only at the interaction after the first non-specular bounce as we to create VPLs only on these surfaces.

A naive solution to obtain the VPL locations is to perform uniform area sampling of the scene's surfaces, and use rejection sampling according to $\hat{L}(\mathbf{x})\hat{W}(\mathbf{x})$. However, this might require generating a huge number of samples to create VPLs, as many candidate locations might lie in unlit regions or regions not, or weakly, illuminating visible surfaces.

To avoid excessive rejection we take a different approach: we randomly choose one photon and take its position $\mathbf{x}_p$ as a candidate location. Photons are distributed proportional to L, which is only an estimator for $\hat{L}$. However, the estimation is sufficiently accurate and since we will take the other light paths (photons) into account when creating a Rich-VPL

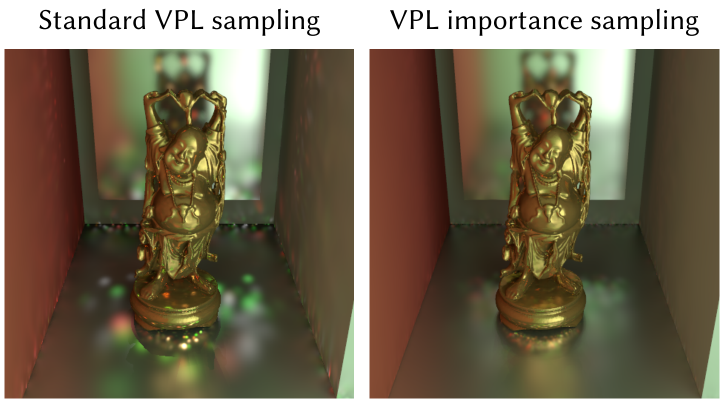


Figure 4.4: The U-Shape scene of figure 4.3. Even with 1.4 million VPLs the standard sampling [Keller 1997] does not yield good results. The same number of VPLs sampled according to the product distribution. Note how the reflection of the Buddha is better captured (render time 3 h 40 min in both cases).

we did not observe any difference in our experiments. Next, we evaluate $\hat{W}(\mathbf{x}_p)$ and create a VPL with an acceptance probability of:

$$P(\mathbf{x}_p) = \min\left(1, \frac{\hat{W}(\mathbf{x}_p)}{q\bar{W}}\right), \tag{4.13}$$

where $\bar{W}$ is the average importance of all photons locations, and q is the ratio of the number of photons to the number of VPLs to be generated. This sampling strategy can be used with any many-light method. In our case, we will reuse the photon map in the later stages of our algorithm. Figure 4.3 shows the differences of importon and photon densities and the target product distribution. Figure 4.4 demonstrates the impact of the VPL placement strategy.

4.4.2 Iterative relaxation of VPL locations

A low discrepancy distribution of VPL locations is desirable as this leads to less clumping artifacts in the shading. We can stratify light subpaths using quasi-random sequences (akin to Keller [Keller 1997]) when generating VPLs, however, some noise in the distribution remains [Spencer and M. Jones 2009].

Following the idea of photon relaxation [Spencer and M. Jones 2009], we can optionally relax the VPL locations after sampling them. This does not move around energy in the scene in our approach as the emission of VPLs is determined afterwards.

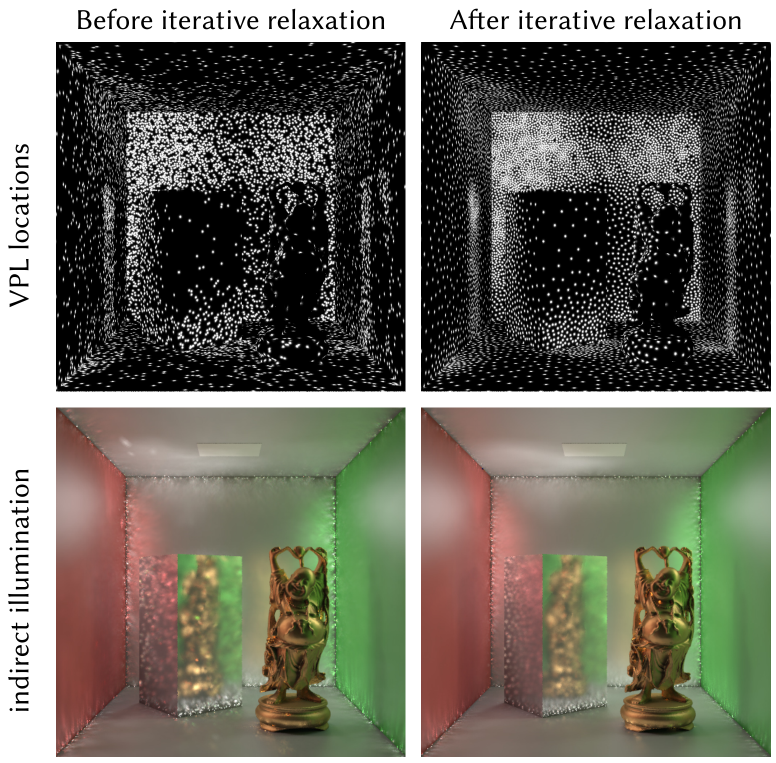


Figure 4.5: We show 10 k Rich-VPLs sampled according to the product distribution before and after relaxation. The illustrations of VPL location shows that after 20 iterations the blue noise characteristic of the Rich-VPL distribution is clearly visible while the global distribution is well preserved. We intentionally disabled clamping to highlight the improvements visible in the indirect illumination of the corresponding renderings.

We use the iterative relaxation scheme developed by Spencer and Jones from Turk's point repulsion method [Turk 1991]. In every iteration we search the K-nearest neighbors of a VPL and modify its location by

$$\Delta \mathbf{x} = \frac{1}{K} \sum_{i=1}^{K} (\mathbf{x} - \mathbf{x}_i) \left(\frac{\|\mathbf{x} - \mathbf{x}_{K+1}\|^2}{\|\mathbf{x} - \mathbf{x}_i\|^2 + \epsilon} - 1 \right). \tag{4.14}$$

One important difference to photon relaxation [Spencer and M. Jones 2009] is that we must prevent VPLs from moving off the surfaces or under the surface. We can achieve this with negligible cost by reusing the photon map that was created to estimate the total incident radiance. This photon map provides a dense sampling of the scene's surfaces with orders of magnitudes more photons than VPLs. After each relaxation step the position of each VPL is snapped to the location of the closest photon. This ensures that VPLs always reside on surfaces. The snapping did not negatively impact the relaxation according to our experiments.

Although we reuse the photon map as a dense surface sampling in this step, the relaxation itself is cheap to compute as the number of VPLs is significantly lower than the number of photons. In all our experiments we used $K = 6$ and performed 20 relaxation iterations which took less than 1 second to compute in all our experiments; figure 4.5 shows the VPL locations and renderings without and with relaxation.

4.5 Generating Rich-VPLs

After having determined the location $\mathbf{p}$ of a Rich-VPL we need to determine and store its emission profile, i.e.,

$$L_e^{\mathbf{p}}(\boldsymbol{\omega}) = \int f_r(\mathbf{p}, \boldsymbol{\omega}_i, \boldsymbol{\omega}) L_{in}(\mathbf{p}, \boldsymbol{\omega}_i) \, \mathrm{d}\boldsymbol{\omega}_i. \tag{4.15}$$

In principle it would be possible to sample $L_{in}(\mathbf{p}, \boldsymbol{\omega})$ using path tracing, or estimating it from another set of conventionally created VPLs [Segovia et al. 2006]. However, this is either costly, or again sub-samples the light transport leading to problems with glossy surfaces and many primary lights.

Instead we can reuse the same photon map which we created to determine the VPL locations and take photons in the proximity of $\mathbf{p}$ as an estimate of the incident radiance. To this end, we query the photon map for the K-nearest photons at a position $\mathbf{p}$ and estimate the exitant radiance for an outgoing direction $\boldsymbol{\omega}$ as

$$L_e^{\mathbf{p}}(\boldsymbol{\omega}) = \frac{1}{K} \sum_{j=1}^{K} w(\mathbf{x}_j, \mathbf{p}) f_r(\mathbf{p}, \boldsymbol{\omega}_j, \boldsymbol{\omega}) \Phi_j, \tag{4.16}$$

where Φ_j is the incident flux, $\boldsymbol{\omega}_j$ is the incident direction of photon j and w is a filter kernel that weights the contribution of a photon to the estimation depending on its distance to the query point $\mathbf{p}$ with

$$\sum_{j=1}^{K} w(\mathbf{x}_j, \mathbf{p}) = 1. \tag{4.17}$$

This function approximation is different from density estimation as we do not divide by surface area. Similar to photon mapping, this estimation introduces bias which depends on K and the total number of photons. However, this bias is negligible compared to the

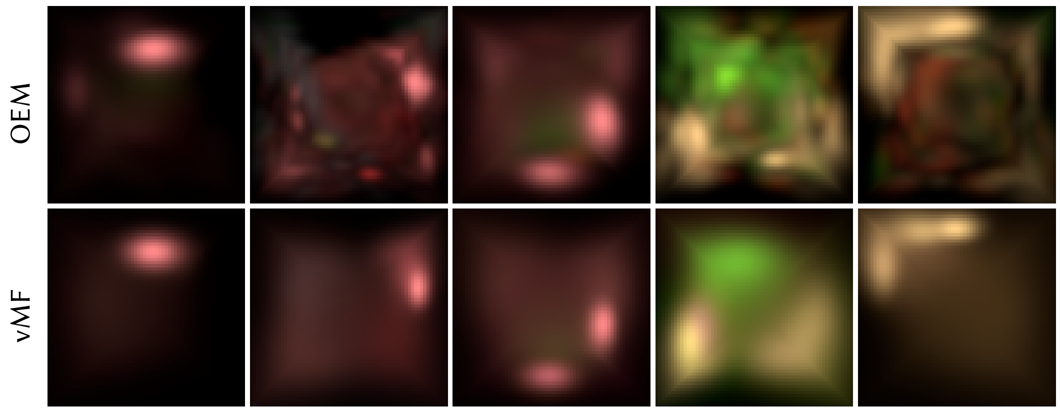


Figure 4.6: Top row: The emission of Rich-VPLs in the upper hemisphere tabulated and stored as octahedron environment maps (OEM). Bottom row: Approximation of the emission using 5 von Mises-Fisher-lobes (vMF) per color channel (see section 6.6).

bias introduced by necessary clamping of VPL contributions due to glossy materials or singularities.

As we want a Rich-VPL to represent many light paths, the number of photons N is typically large and it would be costly to query the incident radiance and evaluate the reflected radiance during shading. Instead we propose to tabulate $L_e^{\mathbf{p}}(\boldsymbol{\omega})$ and store it as a small environment map which can be used during shading with fixed look-up cost, independent of the number of photons or incident light paths.

4.5.1 Computing the emission of Rich-VPLs

We use two different ways to compute the tabulated emission depending on the glossiness of the surface at $\mathbf{p}$. For highly glossy or specular surfaces where the reflection is focused to a narrow solid angle, we importance sample outgoing directions $\boldsymbol{\omega}$ for each photon according to the BRDF $f_r(\mathbf{x}_k, \boldsymbol{\omega}_k, \boldsymbol{\omega})$ and accumulate the reflected radiance. Here $\mathbf{x}_k$ is the position, and $\boldsymbol{\omega}_k$ the incident direction of the k-th photon.

For diffuse or moderately glossy surfaces we first tabulate the accumulated incident radiance of all N photons before convolving it with the BRDF. However, if we accumulate first and then convolve, we cannot use the BRDFs at the location of the photons and instead use the BRDF at the VPL position. This strategy is the most-efficient when N is large.

For VPLs on Lambertian (perfectly diffuse) surfaces the emission is independent of the outgoing direction and their tabulate $L_{\mathbf{p}}(\omega)$. The singularity from perfectly specular BRDFs (e.g., perfect mirrors) is resolved by spreading the energy over at least the solid angle of one texel in the environment map. Unless otherwise noted, we used an octahedron

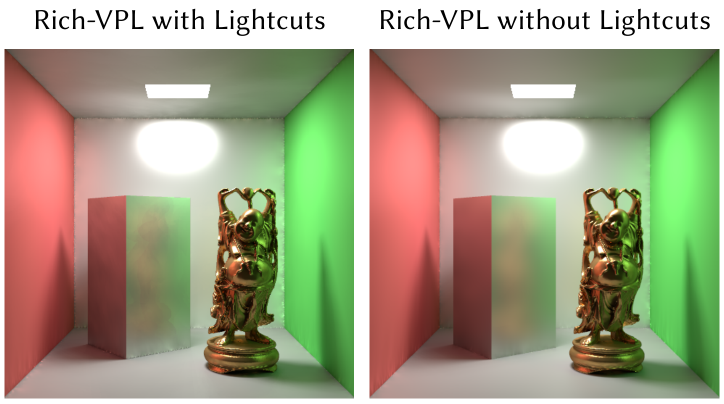


Figure 4.7: Rich-VPL Lightcuts with an error threshold of 0.5% and naive Rich-VPLs. The pure Shading times without Rich-VPL sampling and generation were reduced form 13 min to 2 min 30 s using Lightcuts for 42 k virtual lights.

map [Engelhardt and Dachsbacher 2008] with 32×32 texels for the hemisphere of outgoing directions (figure 4.6); the angular resolution then is approximately 4 to 5 degrees. We used a bilinear filter during the accumulation steps. Other filters can be used to increase the mollification (see also section 6.6).

4.5.2 Shading with Rich-VPLs

Naive many-light rendering with Rich-VPLs is almost identical to traditional VPL shading, except that Rich-VPLs directly store outgoing radiance and the BRDF is not evaluated on the fly. When accessing the environment map during shading, we use bilinear filtering to obtain a smooth radiance field.

4.6 Many-light scalability and fighting artifacts

Naive shading with Rich-VPLs, computing the illumination of every virtual light to every shading point, benefits from our improved placement and the richer emission profiles. However, other typical problems of many-light methods remain: singularities, clamping and scalability, as not every VPL is equally important for every shading point.

Lightcuts with Rich-VPLs

Lightcuts [Walter et al. 2005] clusters VPLs according to spatial proximity and orientation. For each cluster a representative VPL is chosen and its emission equals the total emission of all VPLs in the cluster. In its original form, it does not support glossy VPLs as the clustering metric is not well-suited for strongly varying emission. Rich-VPLs can be easily clustered for spatial proximity and we can directly obtain the emission of a representative by summing up the environment maps of all Rich-VPLs in a cluster. Since the orientation of the hemisphere in which a Rich-VPL emits energy varies, we need to store the full spherical domain in the environment map for cluster representatives.

To estimate the error bound during cut refinement, we need to compute the maximum incident radiance from a cluster onto a shading point. Therefore, we compute the solid angle subtended by the bounding box of a cluster and then determine the maximum exitant radiance at the representative within the same solid angle centered around the direction to the shading point [Walter et al. 2005, Sec.4.1]. As we store the emission in an environment map, we can easily compute a max-mip map hierarchy for the emission and directly access the appropriate mip-level to obtain a conservative estimate for a cone of directions.

During our experiments we found that the error threshold of the original heuristic for cut refinement has to be lowered for highly glossy VPLs as otherwise distracting artifacts appear. This results in larger cuts and reduces the efficiency of Lightcuts, but still provides improved scalability (see figure 4.7). Lightcuts requires storing twice as many emission profiles due to inner nodes; moreover for these we need to store full spherical emission.

Rich-VSLs

We can combine Rich-VPLs with the idea of VSLs [Hašan et al. 2009] which address the problem of shading singularities by distributing the energy of a VPL over nearby surfaces. During VSL shading, the BRDFs at the shading point and the VSL are importance-sampled. Rich-VSLs work almost identical, except that we do not importance sample the BRDF at the VSL location. This would require costly sampling according to a discrete probability density when storing the emission as environment map and, additionally, it is not necessary since even on glossy surfaces the emission of a Rich-VPLs/Rich-VSLs is spread more evenly.

4.7 Implementation and results

We implemented our VPL placement and the different many-light methods (standard VPLs, Rich-VPLs, Lightcuts, VSLs) in our own rendering framework, which also supports path tracing for reference images and photon mapping. In our (Rich-)VPL shading we are only clamping the geometric term, while the BRDF values are never clamped. For our VPL location sampling (section 4.4) we cast 32 importon paths per pixel, and use a photon map which contains q =100-times as many photons as we want to create VPLs. When querying the photon map to compute the emission of a Rich-VPL, we collect photons within a spherical proximity with a radius equal to the distance to the second closest Rich-VPL. In

Scene	#VPLs	VPLs	VPLs+IS	Rich-VPLs
Figure 4.7 (Box)	25 k	436 s	456 s	589 s
Figure 4.3,figure 4.4 (U-Shape)	35 k	340 s	550 s	555 s
Figure 4.11 (Garage)	13 k	372 s	565 s	576 s
Figure 4.10 (Kitchen)	42 k	814 s	833 s	837 s

Scene	#VPLs	IS	relaxation	creation	shading
Figure 4.7 (Box)	25 k	11 s	< 1 s	11 s	567 s
Figure 4.3,figure 4.4 (U-Shape)	35 k	13 s	< 1 s	8 s	533 s
Figure 4.11 (Garage)	13 k	18 s	< 1 s	2 s	554 s
Figure 4.10 (Kitchen)	42 k	12 s	< 1 s	10 s	808 s

Table 4.1: The top table shows total runtimes for VPLs, VPLs using our proposed importance sampling (IS, section 4.4), and Rich-VPLs with IS.The bottom table shows a breakdown of individual timings in seconds for importance sampling (IS), relaxation, Rich-VPL creation, and shading. All renders used $q = 100\times$ more photons than Rich-VPLs and 20 relaxation iterations.

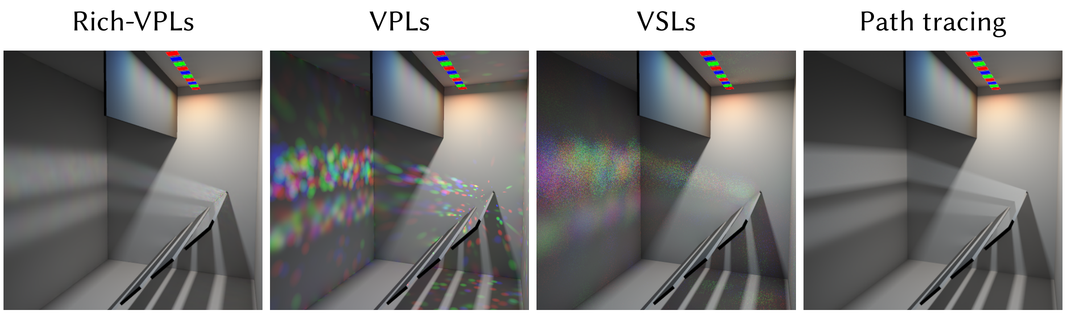


Figure 4.8: The Disco scene with multiple colored light sources and reflected highly-glossy caustics where colors add to white light. Rendered with Rich-VPLs, standard VPLs, virtual spherical lights [Hašan et al. 2009] (VSL), and path tracing reference. All many-light methods use 25 k virtual lights. Rich-VPLs introduce an overhead of about 50% in this geometrically simple scene on top of standard VPL rendering, but significantly improves the ability to capture glossy and near-specular light transport. VSL require about 20% additional rendering time on top of standard VPLs due to stochastic sampling.

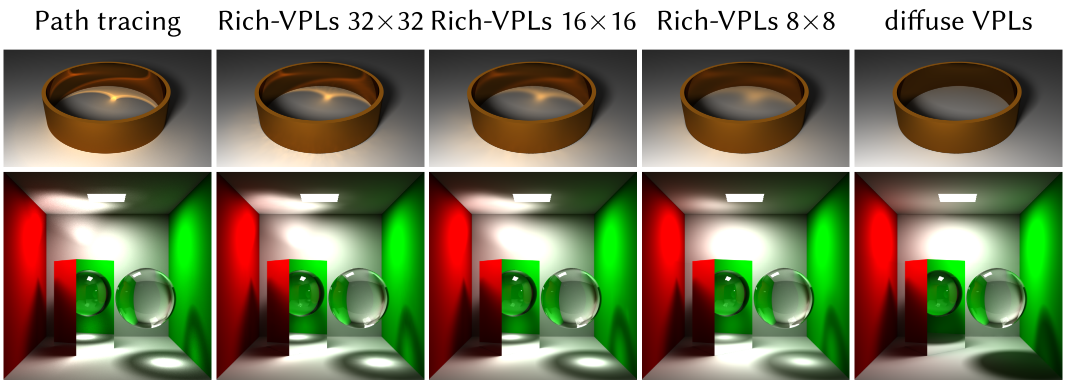


Figure 4.9: A perfectly specular metal ring (top row) and a Cornell Box with a perfectly specular mirror box and glass sphere (bottom row) rendered with different resolutions for the tabulated exitant radiance. On the far left is the path tracing reference and on the far right a VPL rendering with diffuse VPLs. In between are resolutions for the tabulation of the exitant radiance of 32×32, 16×16, and 8×8.

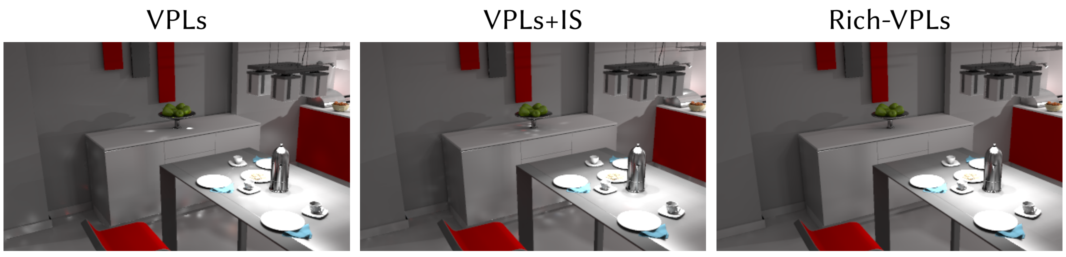


Figure 4.10: The Kitchen scene rendered with 42 k (Rich-)VPLs. From left to right: standard VPLs (13 minutes), VPL with our importance sampling (14 minutes), and Rich-VPLs with importance sampling (14 minutes). In this scene the overhead for Rich-VPLs is small as the render time is dominated by shadow connections.

our renderings with Rich-VSLs we use the same parameters as Hašan et al. [Hašan et al. 2009], i.e., radii M-times the distance to the 10th nearest neighbor VPL, with $M = 4..10$.

To compute the exitant radiance of Rich-VPLs (section 4.5) we execute the BRDF importance sampling on the CPU, and the convolution of the accumulated incident radiance on the GPU using CUDA. For all results, Rich-VPLs use a hemispherical environment map of 32×32 texels. During shading we use interleaved sampling with a 3×3 pixel sub-sampling scheme just as [Hašan et al. 2009; Davidovič et al. 2010]. All steps of our algorithm besides the aforementioned CUDA kernel are implemented on the CPU. All components except for the kD-tree construction for photon and importon maps are multi-threaded.

We evaluate and compare Rich-VPLs running on an Intel Core i7-3770 CPU with 3.40 GHz and 16 GB ram, using eight threads. In table 4.1 timings of our method are shown. For

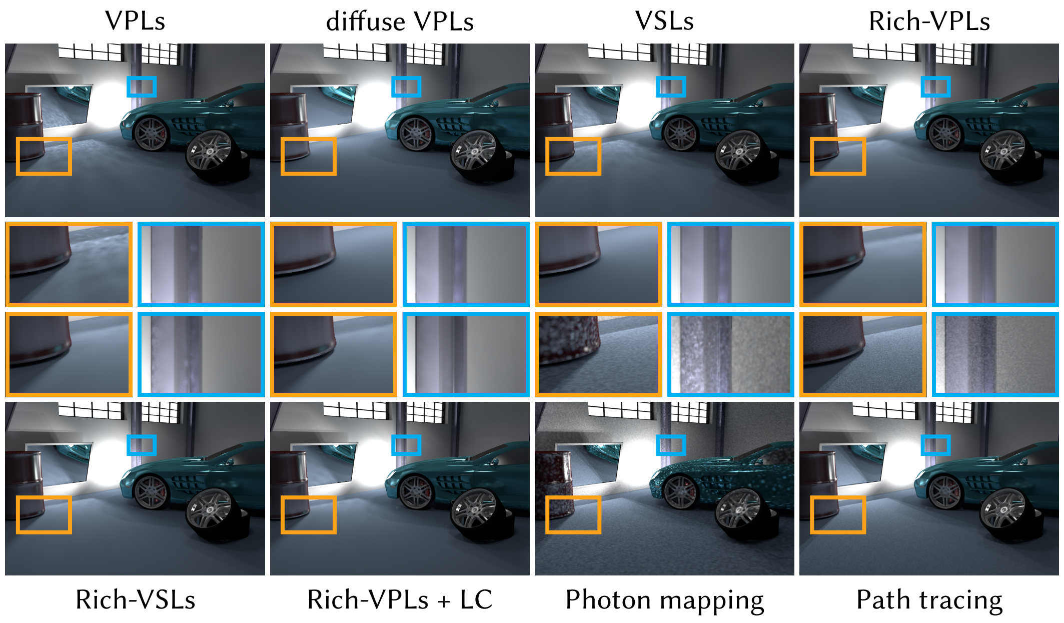


Figure 4.11: The Garage-scene rendered with 250 k virtual lights (created from 11 primary light sources). Rich-VPLs can capture highly glossy light transport much better than standard VPLs. In many methods, VPLs are assumed to be diffuse which results in wrong or missing light transport when compared to the path tracing reference. VSLs handle glossy reflections better than VPLs, but are still more prone to artifacts than Rich-VPLs. Combined with Lightcuts (Rich-VPL+LC) render times with Rich-VPLs are reduced by about 20% (1% threshold). All VPL-based methods used our importance sampling for VPL placement (section 4.4).

the same number of virtual lights, Rich-VPLs with our location sampling require approximately 50% more computation time which is due to the slightly increased cost of emission computation, shading, and the importance sampling.

The impact of our VPL placement can be best observed in the U-Shape scene in figure 4.4. Due to difficult visibility, the benefit of good VPL importance sampling is tremendous. To demonstrate the effectiveness of the Rich-VPLs, we show the disco scene in figure 4.8. This example intentionally has very simple visibility to highlight the benefits for glossy reflections and multiple primary light sources.

We can trade visual accuracy for memory by modifying the resolution of the tabulated exitant radiance. To demonstrate the impact of varying angular resolutions we rendered a scene with a metal-like ring and a modified CornellBox with path tracing, diffuse VPLs, and Rich-VPLs with different resolutions shown in figure 4.9. These scenes contain perfectly specular objects which would be impossible to render with standard VPLs without some kind of mollification.

To show the performance of the algorithm in more realistic scenes, we show the Kitchen-scene (figure 4.10) and the Garage-scene (figure 4.11). The kitchen shows roughly the same increase in quality from VPL importance sampling as well as from using Rich-VPLs. The garage was rendered with all diffuse VPLs (as used in most many-light methods, e.g., Bidirectional Lightcuts [Walter et al. 2012]), a direct rendering of a photon map (122 million photons) and a path tracing reference (250k samples per pixel). We compare these to VPLs, Rich-VPLs, VSLs and Rich-VSLs for all of which we used our VPL importance sampling. While diffuse VPLs produce a very smooth image, they fail to capture features such as the shadow boundary at the barrel completely. For glossy virtual lights, VSLs should ameliorate the remaining problems with blotches, however, the original heuristic was difficult to adjust (the parameter M which controls the VSL radii) for the very high VSL density on the mirror without overblurring the other parts (see for example the reflection of the tires in the door of the car). We believe that a better heuristic can easily be designed. The insets in figure 4.11 show how Rich-VPLs faithfully maintain the shadow boundaries, with just a little blur due to the environment map quantization; also the typical VPL blotchiness is reduced significantly. When using Lightcuts on top of Rich-VPLs we gain an overall speedup of about 20% (compared to brute force Rich-VPLs). The cut sizes in this scene varied strongly due to the glossy surfaces between 900 and 70 k virtual lights.

In figure 4.12 we show equal-time comparisons of the Rich-VPLs to path tracing and photon mapping, and the resulting mean squared error (MSE) compared to a path tracing reference. We also show the relative error $|r - v|/r$ where r is the averaged (over RGB) pixel value of the path tracing reference. In figure 4.13 we show a scene rendered using

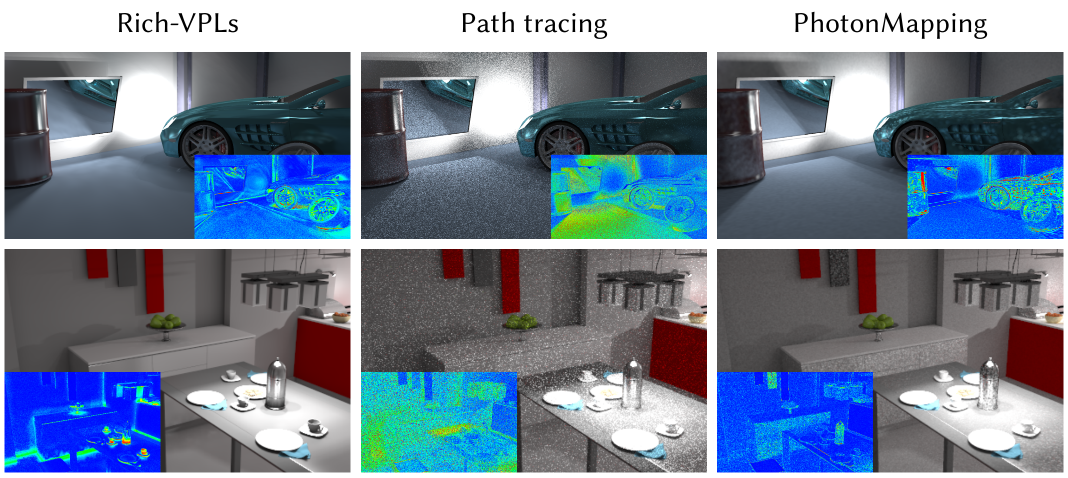


Figure 4.12: Equal-time rendering of the Garage (5 h) and Kitchen (1 h) scenes with Rich-VPLs (left), path tracing (middle) and photon mapping (right). The MSE compared to a path tracing reference is 4.70×10^{-3}, 12.35×10^{-3} and 4.73×10^{-3} in the Garage, and 0.24×10^{-3}, 0.59×10^{-3} and 0.29×10^{-3} in the Kitchen, respectively. The insets show the relative error compared to the path tracing reference.

38 k Rich-VPLs and VPLs and a version that uses all 3.8 million photons that where created for the Rich-VPLs as individual VPLs. The rendering time of Rich-VPLs is much lower than the version with 3.8 million VPLs but it results in a rendering with similar quality. This indicates that the 3.8 million VPLs induce a large amount of unnecessary oversampling.

4.8 Discussion and future work

Richness and mollification

Rich-VPLs represent an arbitrary number of incident light paths and are thus better-suited for glossy transport than traditional VPLs/VSLs, and they efficiently handle scenes with many primary lights (figure 4.8). They inherently mollify reflections off (near-)specular surfaces, which would be needed anyway for VPLs to render this transport. An interesting future work would be to explicitly control the mollification for progressive rendering [A. Kaplanyan and Dachsbacher 2013] or to render with fewer Rich-VPLs trading accuracy for speed.

Photon mapping

A legitimate question is how Rich-VPLs position themselves compared to photon mapping, in particular as we compute a photon map. Many-light methods in general can be interpreted

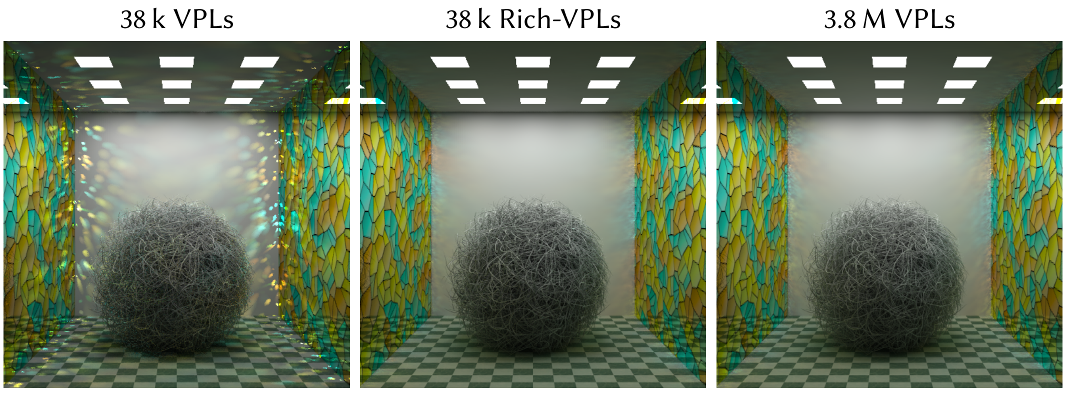


Figure 4.13: A scene with textured mirror walls rendered using 38 k standard VPLs ($\approx$ 18 min), 38 k Rich-VPLs ($\approx$ 18 min) and a rendering where all 3.8 million created photons are used as VPLs ($\approx$ 30 h).

as a final shooting instead of the final gathering algorithm. A Rich-VPL can furthermore be seen as a cache storing the result of a radiance kernel estimation using many photons. During final gathering for photon mapping this kernel estimation would be evaluated much more often. In the end it depends on the particular application whether noise due to shooting gather rays, or smooth shading yet correlated light transport with many-light rendering is better suited.

Clamping and bias

Rich-VPLs do not address clamping or bias compensation problems. As demonstrated, artifacts can be reduced with VSLs, but not eliminated completely. We believe that Rich-VPLs are a good complement to the Bidirectional Lightcuts method [Walter et al. 2012] which significantly reduces bias, enabling it to handle more transport using virtual lights (e.g., caustics as in figure 4.9 and figure 4.11 are not handled at all by [Walter et al. 2012]).

Memory

For very large numbers of Rich-VPLs the memory required for storing their emission might become an issue (e.g., for 32×32 texels, each holding a RGB-float triple, we need 12 kilobytes). Therefore we evaluated whether the emission can be sufficiently well represented using Gaussian-like distributions. To this end, we computed a fit of a von Mises-Fisher (vMF) distribution with 5 lobes per color channel and 32 iterations using expectation-maximization. To reduce memory consumption during rendering, we compute the fit for each completed Rich-VPL and only keep the coefficients for subsequent shading. For each lobe we store 4 floats (2 for direction, inverse width and amplitude), i.e., 240 bytes per

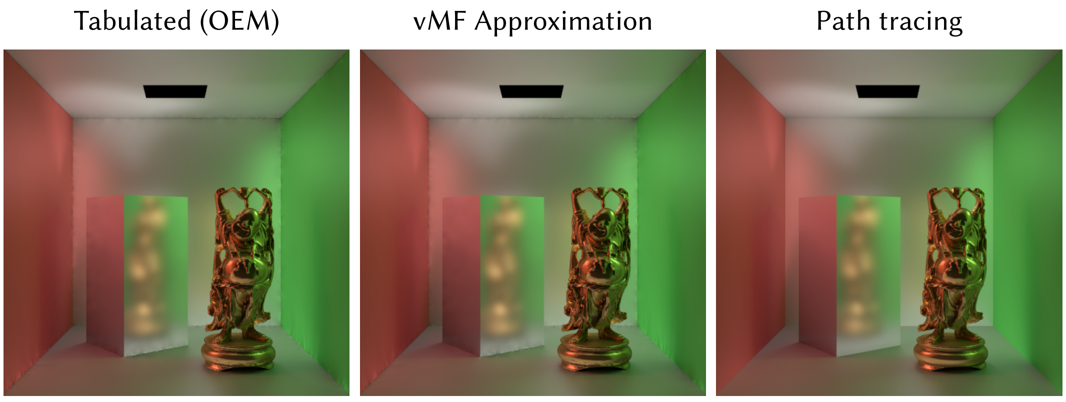


Figure 4.14: Comparison showing only indirect illumination using tabulated emission as octahedron environment maps (OEM) and von Mises-Fisher (vMF) approximation using 5 lobes per color channel for Rich-VPLs and path tracing reference.

Rich-VPL. Figure 4.6 shows several original emission profiles tabulated as octahedron environment maps [Engelhardt and Dachsbacher 2008] (OEM) and their vMF-fits. Figure 4.14 shows that vMF-fits result in very similar rendering quality. In all other examples we used tabulated emission instead of our unoptimized proof-of-concept vMF-implementation. Another simple way to reduce memory requirements is to adapt the environment map resolution to surface glossiness. We could also compute smaller photon maps and compute the emission of the Rich-VPLs progressively, i.e., compute emission from a smaller photon map, discard the photon map and repeat with a new one.

Placement heuristics

We believe that our VPL placement could flexibly incorporate heuristics for more fine-grained control. For example, we could artificially increase the importance (deviating from physics) when an importon travelled along a short path segment. This would enforce more VPLs in cavities which are often undersampled with many-light methods.

4.9 Conclusion

In this chapter we introduced Rich-VPLs which increase the efficiency of many-light rendering in the presence of highly glossy materials. This new lighting primitive accounts for the contribution of many, instead of one, incident light paths. Rich-VPLs enable angular filtering of light transport and extend the set of lighting features that can be handled well with many-light methods. Along with the new light type, we propose a VPL placement strategy that accounts for the total importance with respect to the whole image as well as the total incident radiance of potential VPL locations. We demonstrated the benefits of our method and its ability to complement many-light methods addressing orthogonal problems such as scalability and bias compensation.

Even tough our approach is not fast enough for applications in preview or interactive renderings yet, recent research shows promising modifications and adaptations of the ideas presented in this chapter for interactive rendering. For example, virtual spherical Gaussian lights [Tokuyoshi 2015] model the emission of a virtual point light as a spherical Gaussian which is similar to a von Mises-Fisher-lobe. Assuming that the virtual point light distribution and material reflection is Gaussian, they achieve highly glossy light transport with interactive frame rates. To reduce flickering artifacts in interactive applications [Hedman et al. 2016] use an adaptive resampling approach to distribute VPLs in important regions in a temporally stable fashion. Like the method presented in this chapter, they focus VPLs in important regions and account for information of multiple light paths per VPLs. Assuming only diffuse indirect illumination they achieve real-time performance for scenes in which camera and light sources are dynamic.

5 Forward next event estimation

While many-light rendering can be the method of choice for light transport simulation in a vacuum, especially for interactive and preview rendering, point light primitives are less suitable for handling participating media. Most of the research on using many-light methods for volumetric light transport focuses on the issue of singularities [Engelhardt et al. 2012; Novák et al. 2012b]. These singularities become especially problematic for multiple scattering and emissive volumes.

Recent work introduced a hierarchical structure to render heterogeneous emissive media using the many-light approach [Yuksel and Yuksel 2017]. However, the result is biased, and the issue of robustly rendering the directly visible component of the emissive volume remains. For these reasons, we resort to path tracing for rendering heterogenous emissive volumes.

In this chapter, we introduce an importance sampling method which handles the whole density range of emissive volumes, form thin flames to dense and smoky explosions, efficiently and robustly.

5.1 Introduction

The rendering of realistic images typically requires the inclusion of heterogeneous participating media. However, the presence of participating media often increases the rendering times of Monte Carlo rendering algorithms significantly. In recent years, notable progress has been made in rendering scattering and absorbing media [Kulla and Fajardo 2012; Novák et al. 2014; Křivánek et al. 2014; Zhao et al. 2014]. However, there is considerably less work on rendering heterogeneous emissive volumes, such as fire, explosions, and flames.

In principle, one can easily account for emission by accumulating the contribution at scattering events of Monte Carlo random walks. However, for thin volumes, like flames, free path sampling, according to transmittance, will only rarely sample a path vertex inside the volume. Since the emission is picked up at path vertices only, it is poorly sampled, which results in noisy images (see figure 5.1). Alternatively, one can deterministically step through the whole volume and collect the emission using a quadrature rule. While this deterministic approach has no variance, it can be wasteful to access all the volume data in case the volume contains dense areas, shadowing the emission behind.

For efficient rendering, the emission is usually directly sampled using next event estimation (NEE) [Villemin and Hery 2013]. However, this approach neglects the transmittance between a shading point (path vertex) and the sampled emissive point inside the volume.

In dense media, this can lead to many samples with low contribution, and to high cost as the volume needs to be traversed to compute the transmittance between possibly distant locations. Accessing volume data or evaluating procedural volumes, usually dominates the render cost. Marching through the entire volume or along long segments can quickly increase render times by a significant factor.

In this chapter, we propose a novel *forward next event estimation* (FNEE) technique, which generalizes regular NEE and unifies it with both transmittance sampling and deterministic integration. First, we introduce a *line integration technique* to gather the emission along entire segments between scattering events and show how to derive a correctly weighted measurement contribution function.

Second, we combine this with the importance sampling of emission. In essence, we sample an emitting path vertex as in regular NEE, but then discard the vertex location and only keep the direction into which we trace a ray. The length of the segment for which emission is gathered is determined by free path sampling proportionally to the transmittance.

Note that this is not sampling the product of emission and transmittance. However, it does consider emission (leading to higher contribution) and transmittance (leading to higher efficiency because it touches less volumetric data). In conjunction with line integration, our technique can outperform regular NEE in many cases.

5.2 Background and previous work

As introduced in section 2.2.4, a participating medium can be fully characterized by its phase function f_p along with its coefficients for scattering μ_s, absorption μ_a, emission μ_e, and extinction $\mu_t = \mu_s + \mu_a + \mu_e$. To clearly differentiate between the scattering cross section and the emitted radiance of a particle, we chose to explicitly model the source term as the product $\mu_e L_e$ of a separate coefficient for emissive particles μ_e and the emitted radiance L_e (which is then in Watts per square meter, analogous to the surface case). This will let us reason more clearly about emissive media which are thin (low μ_e) and bright (high L_e), such as candle lights.

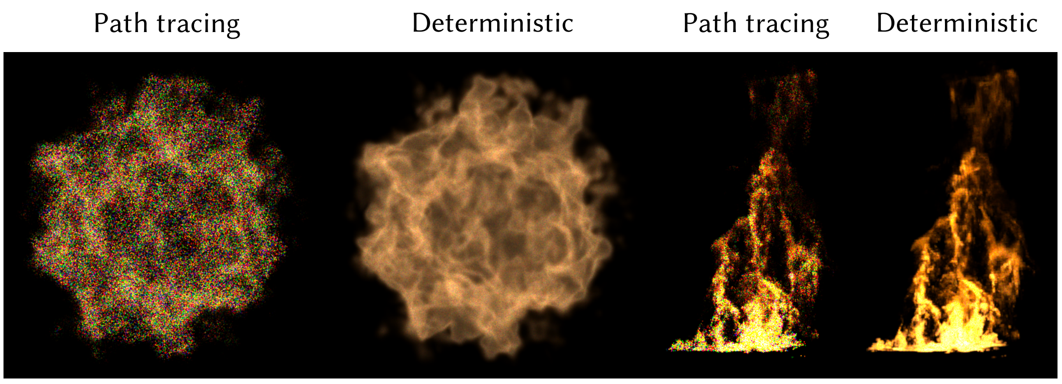


Figure 5.1: Path tracing thin emissive media is challenging: path segments pass through the volume most of the time, resulting in noisy images (images: path tracing). Deterministically collecting the total emission along a ray computes better results (images: deterministic). However, this can be inefficient if the volume also has dense parts.

Sampling distances differently, e.g., according to the product of transmittance and emission, can be done by constructing a discrete cumulative distribution function (CDF) as in [Kulla and Fajardo 2012]. However, this can be costly if the volume contains fine high frequency details.

Usually some form of direct light evaluation (next event estimation, NEE, [Kajiya 1986]), which creates endpoints of paths directly on a light source, is used to increase the efficiency of path tracing. Our forward next event estimation is similar to [Morley et al. 2006] in that it creates directions to a light source instead of points on it, but it still resembles next event estimation because the path is terminated at the resulting scattering point.

Many-light methods [Dachsbacher et al. 2013] and the higher dimensional extensions to photon density estimation [Bitterli and Jarosz 2017] create emissive auxiliary lighting primitives in the scene and thus also in volumes but none of these methods specifically targets the rendering of emissive volumes.

Rendering emissive volumes

Villemin and Hery [Villemin and Hery 2013] also focus on handling emissive volumes in the context of Monte Carlo rendering. They use multiple importance sampling (MIS) [Veach and Guibas 1995] to combine emission found by chance using importance sampling of the scattering function, with emission that was directly sampled with NEE. However, sampling according to emission alone, without accounting for the transmittance between the sampled point and the shading point, can be problematic: it can result in long path segments through the volume for which the transmittance evaluation is expensive. And in

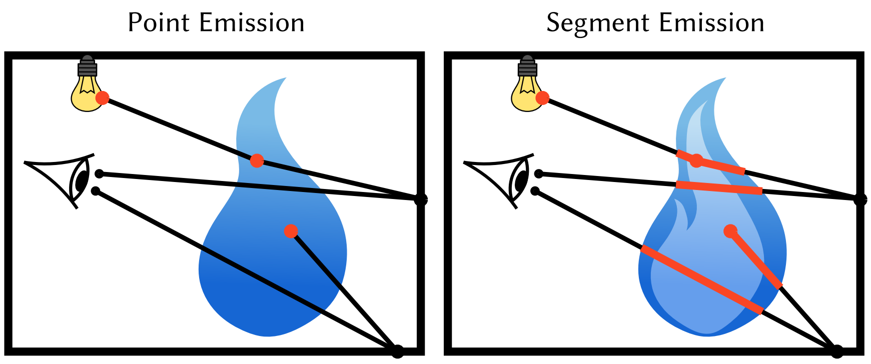


Figure 5.2: Path tracing successively samples interaction points with surfaces or within participating media. The emission is evaluated only at the interaction points along the resulting transport paths (left). Therefore, for thin volumes like flames, the emission is poorly sampled since most path segments pass through the medium, yielding noisy images. Right: we propose a new Monte Carlo estimator that takes into account the emission at both the interaction points as well as along path segments.

case the transmittance is low, the effort is wasted as the sample does not contribute much to the illumination. Furthermore, the emission is only evaluated for the sampled point; in contrast, we propose to account for both transmittance and emission along path segments.

5.3 Computing volumetric emission

In this section, we derive the steps leading to our forward next event estimation technique. This estimator works robustly for the whole spectrum between thin and dense media, samples according to emission and transmittance, and gathers emission along the next event path segments.

It is based on the observation that if L_e is stored closely in memory to μ_t or density values, this will result in the same or very similar data accesses as just sampling a free distance, and thus mostly runs at the same speed (see section 5.4 for details). We believe that similar considerations hold for procedurally generated volumes.

Our technique consists of the following three components. First, whereas in a standard Monte Carlo estimator, volumetric emission is gathered at points chosen by free path sampling we use a concept known in the field of neutron transport [Spanier and Gelbard 1969] and perform *line integration* (section 5.3.2), which gathers the volumetric emission along the ray segments up to the sampled point (see figure 5.2). By this we include more information into the estimator. To yield an unbiased result, however, due to multiple overlapping segments, the emission needs to be weighted appropriately to compensate for

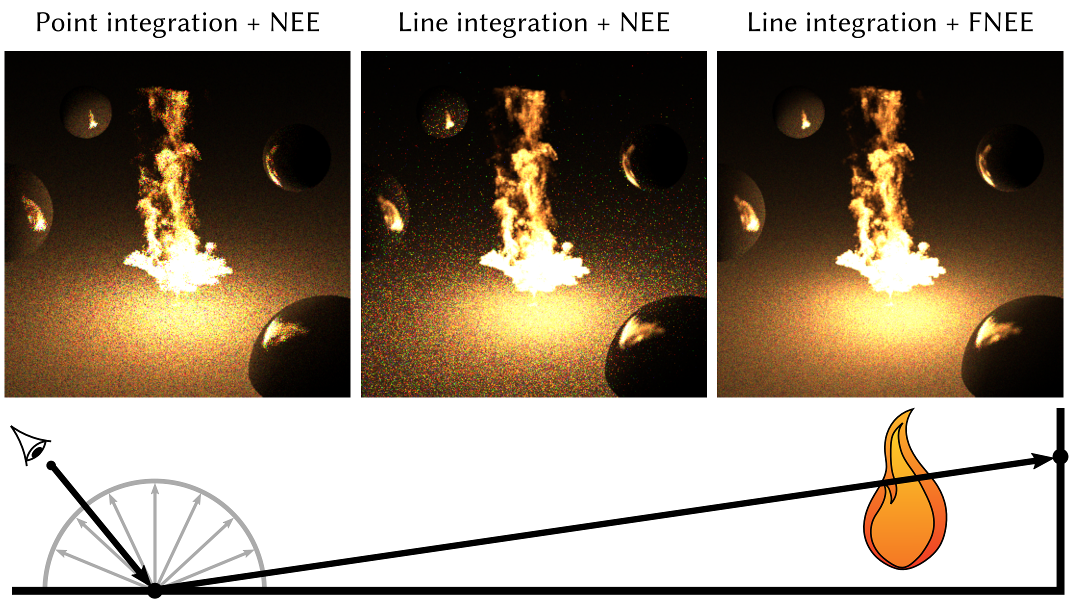


Figure 5.3: Equal-sample comparison. Left: regular next event estimation (NEE) yields noisy results in the directly visible thin flame. Center: integrating emission along path segments improves the rendering of the flame. However, additional noise on the floor arises, as line integration after BSDF sampling and NEE cannot be combined using MIS. Right: our forward NEE also accounts for emission along segments and can thus be combined with line integration.

the multiple gathering. Second, we show how to combine the point and line integration estimators into a unified estimator, which is well-suited for both thin and dense volumes (see section 5.3.3). And last, we show a solution to the problem that line integration and NEE cannot be combined using MIS because line integration can be used for segments passing through the medium, while NEE cannot create such paths and the MIS weight would be zero for such cases (see figure 5.3). We introduce *forward NEE*, which importance samples the volume emission, and converts the sample location into a direction. We use line integration along the corresponding direction, which allows a meaningful application of MIS. This method also leads to improved next event efficiency for dense volumes (see figure 5.4).

5.3.1 Point integration

Our goal is to compute the incident radiance at a point $\mathbf{x}$ in the scene from a direction $\boldsymbol{\omega}$ due to volumetric emission. Ignoring surface geometry and multiply scattered light (which is handled by the Monte Carlo path generation), it follows from equation 2.34 that we

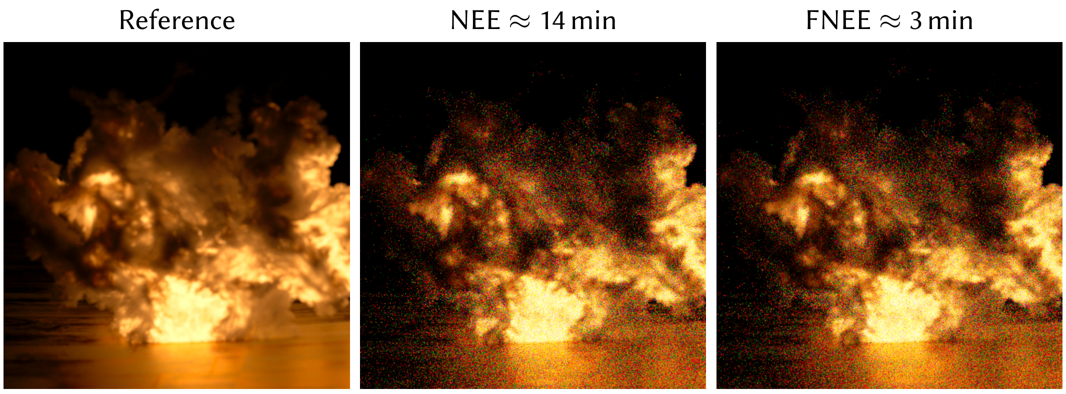


Figure 5.4: Next event estimation for dense volumes can be wasteful as only nearby points contribute to a path due to high extinction. This is reflected in large time differences between regular NEE and our FNEE. Both images use 32 samples per pixel and have roughly equal quality. The volume properties are $\sigma_t = 32$, $\sigma_s = 30$, $\sigma_e = 0.1$.

have to integrate the emitted radiance $\hat{L}_e(t) = \mu_e(t)L_e(t)$ weighted by the corresponding transmittance $T(t)$ along the ray $r(t) = \mathbf{x} + t\boldsymbol{\omega}$:

$$\int_0^\infty \hat{L}_e(t)T(t)\,\mathrm{d}t. \tag{5.1}$$

Since this integral has no analytic solution in the general case, we perform numerical integration. Using N randomly sampled distances t_i with probability density $p(t_i)$, we can compute a Monte Carlo estimate of equation 5.1 with

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\hat{L}_e(t_i)T(t_i)}{p(t_i)}, \tag{5.2}$$

which we will denote as the *point estimator*, as the emission is only evaluated at the discrete points $r(t_i)$ along the ray.

The distance sampling probability is usually chosen to be proportional to the transmittance, i.e., $p(t) \propto T(t)$. By this, however, regions with high emission can easily be missed, e.g., in very thin (low μ_t and μ_e, respectively) and bright flames (large L_e). Note that a typical path tracing implementation samples only one distance t_i for the next scattering event and integrates over many light paths per pixel instead. However, the arguments in the following section might be more intuitive for one ray and larger N.

5.3.2 Line integration

When computing one sample t_i of the point estimator (equation 5.2), we also compute the transmittance along the segment $[0; t_i]$. The motivation for the line integration is to also consider the emission along that segment to improve the estimator, assuming that the underlying data structure allows simultaneously gathering emission at only little additional cost. However, simply replacing $\hat{L}_e(t_i)T(t_i)$ by

$$\int_0^{t_i} \hat{L}_e(t)T(t)\,\mathrm{d}t \tag{5.3}$$

in equation 5.2 does obviously not work, as the emission of closer segments along the ray is gathered multiple times (see also figure 5.5). To compensate for this, we introduce an emission weight function, $w_t(s)$, whose choice and derivation we discuss in this section. Note that t is the parameter that we sample for distances, and s the ray parameter along one of the segments. We need to choose $w_t(s)$ such that

$$\text{Equation 5.1} \overset{!}{=} \int_0^\infty \left(\int_0^t w_t(s)\hat{L}_e(s)T(s)\,\mathrm{d}s \right) \mathrm{d}t. \tag{5.4}$$

We can now formulate two conditions to be met by the weight function: first of all, we want $w_t(s) = 0\,\forall\, s > t$, which means that in order to compute the emission along a ray segment up to a sampled location, i.e., $[0; t]$, we do not want to consider emission values at locations further away than t. Second, $w_t(s)$ has to reweight the contributions properly: if we fix an emissive point s on the ray (figure 5.5), the weight function needs to be normalized over all possible sample points t, i.e., (note $w_t(s) = 0\,\forall\, s > t$)

$$\int_0^\infty w_t(s)\,\mathrm{d}t = \int_s^\infty w_t(s)\,\mathrm{d}t = 1. \tag{5.5}$$

We can verify that equation 5.4 yields the same result as equation 5.1 by first extending the interval of the inner integral ($w_t(s) = 0\,\forall\, s > t$), then swapping the order of integration, and reordering the terms:

$$\begin{aligned} \text{Equation 5.4} &= \int_0^\infty \left(\int_0^\infty w_t(s)\hat{L}_e(s)T(s)\,\mathrm{d}s \right) \mathrm{d}t \\ &= \int_0^\infty \left(\int_0^\infty w_t(s)\,\mathrm{d}t \right) \hat{L}_e(s)T(s)\,\mathrm{d}s \\ &= \int_0^\infty \hat{L}_e(s)T(s)\,\mathrm{d}s. \end{aligned} \tag{5.6}$$

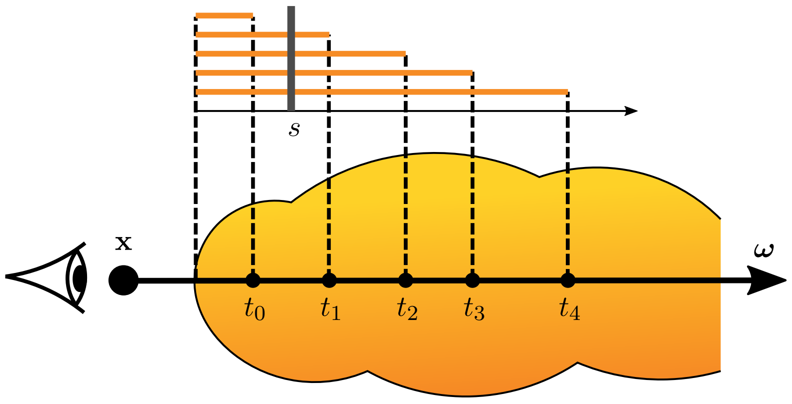


Figure 5.5: Line integration computes the incident radiance due to volumetric emission along a ray $\mathbf{x} + t\boldsymbol{\omega}$ by randomly sampling distances t_i and accumulating the emission along the segments $[0, t_i]$. To account for multiple gathering of the emission at a point $s \in [0, t_i]$, we weight the contribution with a function $w_t(s)$ with $\int_0^\infty w_t(s)\,\mathrm{d}t = 1$.

Choice of weight function

The above definition allows for different weight functions. Choosing $w_t(s) = \delta(s, t)$, for example, demonstrates that the line integral formulation in equation 5.4 is a generalization of the point integral from equation 5.1.

We propose to use a weight function motivated by transmittance sampling: the transmittance $T(s)$ is the probability of sampling a location $t > s$ along the ray and therefore also the probability for accounting for the emission at s. In order to compensate for the multiple gathering of $\hat{L}_e(s)$ during line sampling, we want

$$w_t(s) \propto 1/T(s) \text{ for } s \leq t \text{ and} \tag{5.7}$$

$$w_t(s) = 0 \text{ otherwise.} \tag{5.8}$$

In addition, we need a normalization term for $w_t(s)$, which also reflects the probability of sampling $t > s$. The probability density of sampling a distance t according to transmittance is $p(t) = \mu_t(t)T(t)$. Consequently, the probability of sampling a location $t > s$ and thus accounting for $\hat{L}_e(s)$ again is:

$$\int_s^\infty \mu_t(t)T(t)\,\mathrm{d}t = T(s) \Leftrightarrow \int_s^\infty \frac{\mu_t(t)T(t)}{T(s)}\,\mathrm{d}t = 1. \tag{5.9}$$

Comparing equation 5.5 and equation 5.9 shows that a valid weight function with the aforementioned proportionality is:

$$w_t(s) = \begin{cases} \frac{\mu_t(t)T(t)}{T(s)}, & \text{if } s \leq t \\ 0, & \text{otherwise.} \end{cases} \tag{5.10}$$

If the volume is bounded at a distance t', either because there is a surface or the volume ends, the remaining weight has to be assigned to the case when the sampled distance t_i exceeds t'. This is exactly the same as for the transmittance sampling probability density function (PDF).

Line estimator

The Monte Carlo estimator for equation 5.4 with a valid weight function becomes:

$$\frac{1}{N}\sum_{i=1}^{N} \frac{\int_0^{t_i} w_{t_i}(s)\hat{L}_e(s)T(s)\,\mathrm{d}s}{p(t_i)}. \tag{5.11}$$

Using our weight function from equation 5.10 and transmittance sampling for the distances t_i, the Monte Carlo estimator can be simplified to

$$\frac{1}{N}\sum_{i=1}^{N} \int_0^{t_i} \hat{L}_e(s)\,\mathrm{d}s. \tag{5.12}$$

Discussion

Equation 5.12 yields the same result in the limit as the point estimator in equation 5.2, but it uses more information for a single sample by collecting the emission along the line segment. The special case of the line estimator with transmittance sampling (equation 5.12) is also known as the track length estimator derived by Spanier and Gelbard [Spanier and Gelbard 1969] in neutron transport theory. If the distances t_i are sampled proportional to transmittance it is easy to show that the track length estimator

$$\frac{1}{N}\sum_{i=1}^{N} \int_0^{t_i} \hat{L}_e(s)\,\mathrm{d}s, \tag{5.13}$$

is an estimator for

$$\int_0^{\infty} \hat{L}_e(t)T(t)\,\mathrm{d}t. \tag{5.14}$$

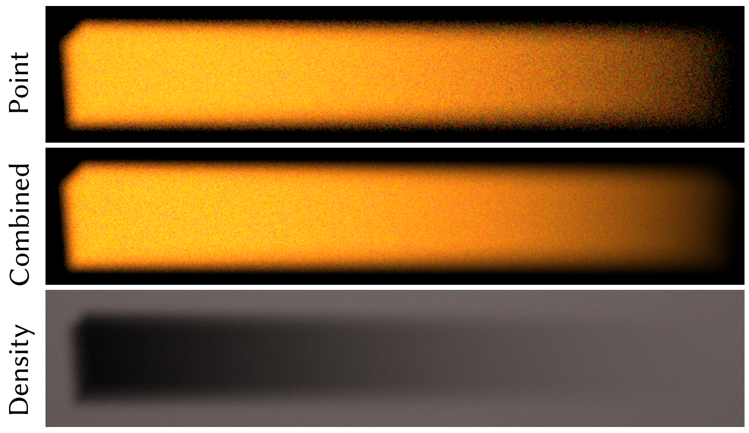


Figure 5.6: A simple cube-shaped medium with decreasing density from left to right. The combined point+line estimator performs better than the point estimator alone (top row). This is best visible in the thin parts of the volume.

First, we define a probabilistic estimator for the transmittance using a random distance s with $p(s) \propto T(s)$ as

$$T'_s(t) = \begin{cases} 1, & \text{if } t \leq s \\ 0, & \text{otherwise.} \end{cases} \tag{5.15}$$

This estimator is unbiased because

$$\mathbb{E}\big[T'_s(t)\big] = \int_0^\infty T'_s(t)p(s)\,\mathrm{d}s = \int_t^\infty p(s)\,\mathrm{d}s = T(t) \tag{5.16}$$

and since the distances of the track length estimator are sampled proportional to transmittance we can write

$$\begin{aligned} \mathbb{E}\left[\int_0^{t_i} \hat{L}_e(t)\,\mathrm{d}t\right] &= \mathbb{E}\left[\int_0^\infty \hat{L}_e(t)T'_{t_i}(t)\,\mathrm{d}t\right] \\ &= \int_0^\infty \hat{L}_e(t)\mathbb{E}\big[T'_{t_i}(t)\big]\,\mathrm{d}t \\ &= \int_0^\infty \hat{L}_e(t)T(t)\,\mathrm{d}t. \end{aligned} \tag{5.17}$$

In contrast to Spanier and Gelbard [Spanier and Gelbard 1969] we derive a more general Monte Carlo estimator which can be combined with other distance sampling methods as well. For example, sampling according to μ_s instead of μ_t for highly scattering media or equi-angular sampling [Kulla and Fajardo 2012].

5.3.3 Combining the line and point estimators

The previously introduced line estimator is superior to the point estimator in thin media as it gathers emission even when no scattering event was sampled inside the volume. However, it does not always have lower variance than the point estimator. For example, if $\hat{L}_e$ and μ_t are nearly constant along the line segment, the point estimator will have little variance if transmittance sampling is used:

$$\frac{\hat{L}_e(t_i)T(t_i)}{p(t_i)} \approx \frac{\hat{L}_e}{\mu_t} \approx const. \tag{5.18}$$

However, the contribution of the line estimator in this case is

$$\int_0^{t_i} \hat{L}_e(s)\,\mathrm{d}s \approx \hat{L}_e t_i, \tag{5.19}$$

where the random segment length t_i introduces variance.

For this reason, we want to combine the strengths of both estimators: we want the point estimator to have a higher contribution than the line estimator in dense regions and vice versa. To this end, we propose weighting their contributions at a particular location in the medium according to the transmittance along the ray up to that point. Note that this is not equivalent to a convex combination of equation 5.2 and equation 5.11 because of the spatially-varying weight. We obtain the combined estimator by first splitting equation 5.1:

$$\int_0^{\infty} (1 - T(t))\hat{L}_e(t)T(t)\,\mathrm{d}t + \int_0^{\infty} T(t)\hat{L}_e(t)T(t)\,\mathrm{d}t, \tag{5.20}$$

and use the point estimator for the left, and the line estimator for the right integral. We compute the latter analogously to our previous derivation with

$$\int_0^{\infty} T(t)\hat{L}_e(t)T(t)\,\mathrm{d}t = \int_0^{\infty}\int_0^{\infty} w_t(s)T(s)\hat{L}_e(s)T(s)\,\mathrm{d}s\,\mathrm{d}t \tag{5.21}$$

and obtain the combined estimator as

$$\frac{1}{N}\sum_{i=1}^{N}\left((1 - T(t_i))\frac{\hat{L}_e(t_i)T(t_i)}{p(t_i)} + \frac{\int_0^{t_i} w_{t_i}(s)T(s)\hat{L}_e(s)T(s)\,\mathrm{d}s}{p(t_i)}\right). \tag{5.22}$$

If we sample t_i proportional to transmittance, then the combined estimator conveniently simplifies to:

$$\frac{1}{N}\sum_{i=1}^{N}\left((1 - T(t_i))\frac{\hat{L}_e(t_i)}{\mu_t(t_i)} + \int_0^{t_i} \hat{L}_e(s)T(s)\,\mathrm{d}s\right). \tag{5.23}$$

Note the additional spatially-varying transmittance weight inside the integral of the line estimator. Figure 5.6 shows equal-sample results of the combined estimator compared to the point estimator.

Discussion

The combination of the point and line estimator is similar to MIS when distances are sampled according to transmittance (the transmittance is part of the PDF in this case). However, MIS normalizes the contribution of a discrete set of estimators, whereas we normalize over a continuous domain using a weight function. We will show in section 6.5, that our combination is, similar to MIS, not perfect. We experimented with other combinations and weight functions, but they were either much more complicated or needed information along the full ray through the volume. The introduced weight functions and combination of the point and line estimator are the most robust and efficient we found, and they proved to work well for a wide variety of volumes (see section 6.5).

5.3.4 Forward next event estimation

With line integration as part of the combined estimator, the emission along every path segment contributes to the image, even when there is no scatter event inside the volume. Combining such *pass-through paths* with volume NEE [Villemin and Hery 2013] using MIS is problematic. Since NEE only samples emissive points it cannot create these pass-through paths and thus their corresponding MIS-weights become zero, which means that MIS becomes useless. For example, consider pass-through paths from BSDF-sampling as shown in figure 5.3. Here, the volume subtends only a small solid angle and the probability of hitting the volume with a random BSDF-sampled direction off the rough floor is very low and the path cannot be created by NEE. Here NEE means that an emissive point is sampled inside the volume. For NEE with other light sources see section 5.3.5.

To overcome this problem we introduce *forward next event estimation* (FNEE). The key idea is to reinterpret the point sampling process of volume NEE as a solid angle direction sampling. For this, we discard a sampled point but keep the direction to it. Thereafter, we sample a distance according to transmittance along that direction, and use the combined estimator from equation 5.23 in the same was we do for phase function or BSDF scattering. This technique has the advantage that we can again use the combined estimator to gather emission along segments. Also it can sample pass-through paths and has meaningful MIS weights for every contributing path. In addition to that, sampling distances according to transmittance avoids deep penetration of dense volumes and thus improves the efficiency of direct light computation significantly. The main challenge is that multiple points in the

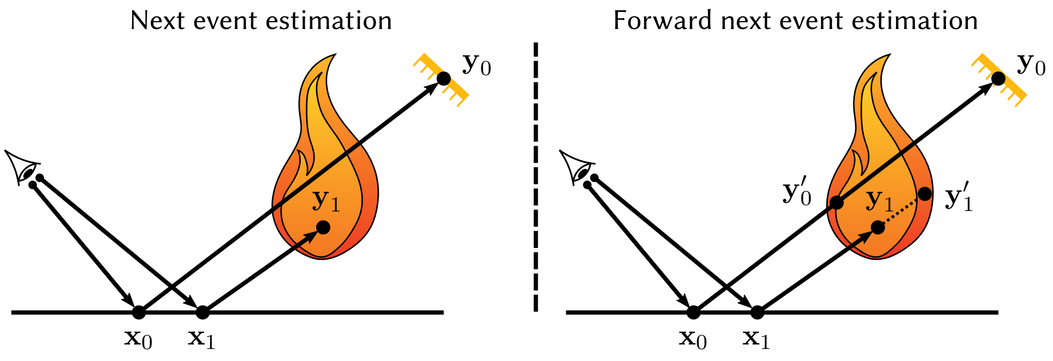


Figure 5.7: Path tracing with regular next event estimation (NEE) constructs paths in two ways: sampling $\mathbf{y}$ via outgoing directions from $\mathbf{x}$ and distance by transmittance, or by sampling a point on a light source. Forward next event estimation (FNEE) provides a third way. At first, a vertex $\mathbf{y}'$ inside the volume is sampled according to emission. Then, a distance towards $\mathbf{y}'$ is sampled by transmittance. This process may pass through the volume and result in a vertex $\mathbf{y}$ on a geometric light source. Such paths will have to be weighted against NEE on surfaces.

volume result in the same direction $\boldsymbol{\omega}$ from a shading point $\mathbf{x}$. To compute the solid angle measure probability of the resulting direction, $p_\sigma(\boldsymbol{\omega})$, we need to integrate the probability over all points, i.e., we have to compute

$$p_\sigma(\boldsymbol{\omega}) = \int_0^\infty p_{\mathbf{x}}(t) t^2 \,\mathrm{d}t, \tag{5.24}$$

where $p_{\mathbf{x}}(t)$ is the vertex volume sampling probability of a point $P(t) = \mathbf{x} + t\boldsymbol{\omega}$, and t^2 is the transformation Jacobian from volume to solid angle measure.

This integration is computationally expensive as it requires stepping from $\mathbf{x}$ in direction $\boldsymbol{\omega}$ through the entire volume to accumulate the point sampling probabilities $p_{\mathbf{x}}(t)$. Obviously, this would not be cheaper than gathering the entire emission directly. However, we can perform point sampling and PDF integration at a low-resolution version of the volume without noticeable differences in the rendering. Even though fine emission details could be missed – which would have a significant impact on regular NEE – line integration has a high chance to pick up the high frequency details again. Distance sampling and line integration are still performed on the full volume resolution and therefore the estimator is unbiased. We evaluate this sampling in figure 5.10.

Probabilistic transmittance for NEE

FNEE is faster for dense volumes because the additional transmittance sampling prevents unnecessarily deep penetration. The same speed benefit can be acquired for NEE with an

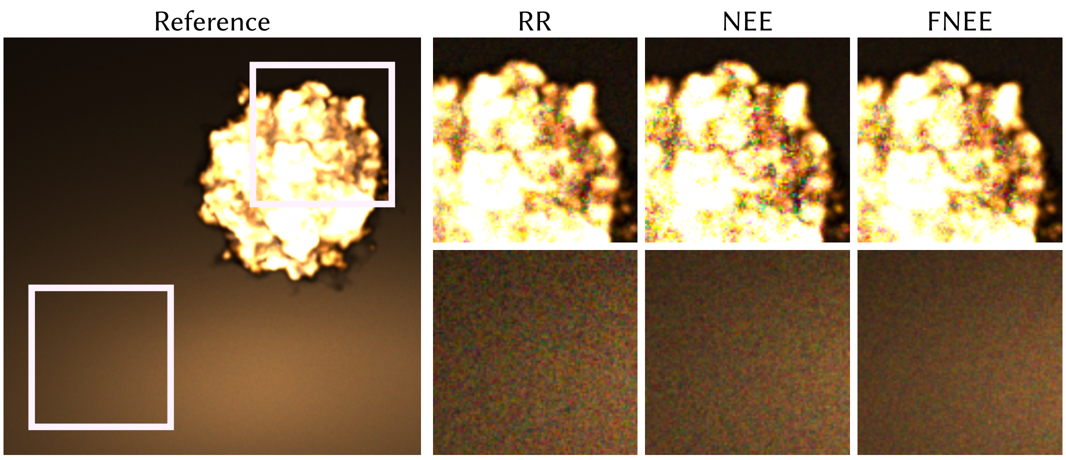


Figure 5.8: Equal-time comparison of NEE with probabilistic transmittance sampling (RR, 487 spp, left), regular NEE (232 spp, middle) and our FNEE (394 spp, right). The quality of all techniques is roughly the same in the directly visible flame (top row), however, the noise in the RR version is higher compared to NEE and even more so compared to FNEE in the illuminated parts (bottom row). All renders took 3 minutes. The RMSE over the cropped view in the bottom row are (from left to right): 0.87, 0.65, 0.49.

unbiased probabilistic transmittance approximation. For this, we sample a distance along the ray to the sampled NEE point according to transmittance and if the distance sampling stops before the NEE point, the transmittance will be zero, otherwise one. This can be viewed as a form of Russian roulette (RR). We found however, that this approach can actually increase variance resulting in higher equal-time RMSE (see figure 5.8) compared to regular NEE. On the other hand, FNEE uses line integration which increases the likelihood of useful contributions and always outperformed NEE with and without RR in our experiments. All comparisons apart from figure 5.8 are with regular NEE since it is the published state of the art at this time [Villemin and Hery 2013].

5.3.5 Other light sources

FNEE can create path segments that pass through the volume and potentially hit other light sources, e.g., area lights or environmental illumination (see figure 5.7). Such paths can be constructed by FNEE in the volume or NEE of the light source. We model this in the MIS framework by stochastically choosing surface NEE or volume FNEE via the one-sample MIS model [Veach and Guibas 1995]. If there are multiple emissive volumes in a scene, they can all contribute to the combined PDF of a path. The easiest way to account for this is to treat them as a single aggregate volume.

5.4 Implementation

In this section, we provide details on the implementation of forward next event estimation and line integration in a path tracing framework, and begin with details of the volumetric data structure.

Volumetric data structure

We use a voxel back end to represent spatially varying volume properties; in particular, we use a 4D storage similar to [Wrenninge 2016]. The basic structure is a hierarchical grid with a 8×8×8 voxel branching factor, similar to the one used in OpenVDB [Museth 2013]. For computing the images in this chapter, we stored density d and temperature in every voxel. We use a constant phase function and reconstruct the spatially varying scattering parameters as $\mu_* = \sigma_* \cdot d$, where $\sigma_{s,a,e}$ are user parameters which can be changed at run time. A shader is used to compute the emission L_e from the temperature, in most of our results by implementing a black body emitter. We construct a full binary tree for every grid to form a light hierarchy. This enables us to sample a voxel randomly according to $d \cdot L_e$ (as in [Villemin and Hery 2013]). Ultimately, we want to sample by $\mu_e L_e = \sigma_e \cdot d \cdot L_e$, but the scalar user constant σ_e is factored out as it only affects normalization. The concepts derived in this chapter are largely independent of the exact data structure used. We also implemented the technique in a production renderer using a *k*-d tree instead of hierarchical grids.

Free path sampling

To compute the distance to the next scattering event of a path, we traverse the volume along a ray and accumulate the transmittance on the leaf level of the voxel hierarchy until it falls below a randomly chosen threshold [Szirmay-Kalos et al. 2011]; this is possible as the transmittance is normalized. Traversing the voxels makes the evaluation of the segment emission integral in equation 5.23 straightforward. For our results, we assumed homogeneous media inside voxels, i.e., constant μ_e, L_e, μ_t, μ_s, which allows to analytically compute

$$\int_0^{t'} \mu_e L_e T(s)\,\mathrm{d}s = \mu_e L_e \int_0^{t'} e^{-\mu_t s}\,\mathrm{d}s = \frac{\mu_e}{\mu_t} L_e (1 - e^{-\mu_t t'}), \tag{5.25}$$

where t' is the length of the ray segment inside the voxel. Pseudo-code of the free path sampling and the combined estimator can be found in alg. 5. Computing segment emission along with the transmittance between two points works analogously.

Algorithm 5 Free Path Sampling with Segment Emission

```
procedure SampleFreePath($\mathbf{x}, \boldsymbol{\omega}, \xi$)
    $T \leftarrow 1$ // transmittance
    $L_e^c \leftarrow 0$ // combined estimate
    for all voxels along $\mathbf{x} + t\boldsymbol{\omega}$ do
        $\{t_0, t_1\} \leftarrow$ distance of voxel entry/exit point
        $\{\mu_t, \mu_e L_e\} \leftarrow$ extinction/emission of voxel
        $d \leftarrow t_1 - t_0$
        $T' \leftarrow T \cdot e^{-\mu_t d}$
        if Scatter($\xi, T, T'$) then
            $t_s \leftarrow$ distance to scatter point in voxel
            $d \leftarrow t_s - t_0$
            $L_e^c \leftarrow L_e^c + T \cdot L_e \frac{\mu_e}{\mu_t} \cdot (1 - e^{-\mu_t d})$
            $T \leftarrow T \cdot e^{-\mu_t d}$
            $L_e^c \leftarrow L_e^c + (1 - T) L_e \frac{\mu_e}{\mu_t}$
            return $\{t_0 + d, L_e^c\}$
        end if
        $L_e^c \leftarrow L_e^c + T \cdot L_e \frac{\mu_e}{\mu_t} \cdot (1 - e^{-\mu_t d})$
        $T \leftarrow T'$
    end for
    return $\{\infty, L_e^c\}$
end procedure
```

We also tested Woodcock tracking in combination with a coarse grid storing the majorant $\mu_t^{\max}$, however, for all volumes in our scenes it resulted in more memory accesses and longer run times. In cases where Woodcock tracking jumps over thin homogeneous regions and uses fewer memory accesses, it also skips potentially important emission.

Point sampling for NEE

To sample a light vertex for next event estimation, we use 3D sample warping on a full binary tree around the voxels of each grid. The target distribution is proportional to $\mu_e L_e$ averaged over wavelength. The intermediate levels sample the averages over the time domain, intermediate levels coinciding with the inner nodes of the hierarchical grid, however, also resolve the probabilities in time.

Forward NEE directional PDF

We compute the PDF (in solid angle measure) for the direction sampling of FNEE by evaluating the integral in equation 5.24 for homogeneous voxels as shown in algorithm 6. While free path sampling (also during FNEE) and point sampling for regular NEE are always performed on the highest resolution of our voxel data structure, we use a coarser

Algorithm 6 Compute direction PDF

```
1: procedure AccumPdf($\mathbf{x}, \boldsymbol{\omega}$)
2:     $p \leftarrow 0$
3:     for all voxels along $\mathbf{x} + t\boldsymbol{\omega}$ do
4:         $\{t_0, t_1\} \leftarrow$ distance of voxel entry/exit point
5:         $p_v \leftarrow$ probability of choosing voxel with point sampling
6:         $p \leftarrow p + (t_1^3 - t_0^3)/3 \cdot p_v$
7:     end for
8:     return $p$
9: end procedure
```

representation for FNEE direction sampling for efficiency reasons. This simply changes the list of voxels to traverse in algorithm 6. The term $(t_1^3 - t_0^3)/3$ is the analytic solution of the integral in equation 5.24 for homogeneous volumes after pulling out the constant PDF $p_{\mathbf{x}}(t)$.

5.5 Results

All images in this chapter were generated with a custom Monte Carlo renderer written in C using spectral rendering. The renderer used a single wavelength per path, but line integration and FNEE would also work with multiple wavelengths or RGB rendering. The images are inherently dark, to keep the fine detail in the emissive volumes. However, that impairs the quality for printing and we recommend to view the images on a computer screen. The renderings in all images and equal-time measurements where done using a machine with a quad-core Intel i7-3770 and 16 GB RAM, except for figure 5.13, which was rendered on a 128-core machine with 4 TB of main memory.

We use six volumes in the chapter with varying complexity (see Tab. 5.1). All volumes use isotropic scattering (i.e., a mean cosine of zero). Except for the second scene in figure 5.11, which uses a custom shader to mimic bioluminescence, all volumes are black body emitters.

5.5.1 Evaluation of the individual estimators

To evaluate the different estimators (point, line and combined) we rendered the fireball volume for varying densities across four orders of magnitude in figure 5.9. The emission is scaled to produce roughly the same total brightness in each case. We can see that the line and combined estimators gain advantage over the traditional point estimator for decreasing thickness of the volume. However, for very dense volumes the point estimator is better than the line estimator since the segment length introduces additional variance. The combined

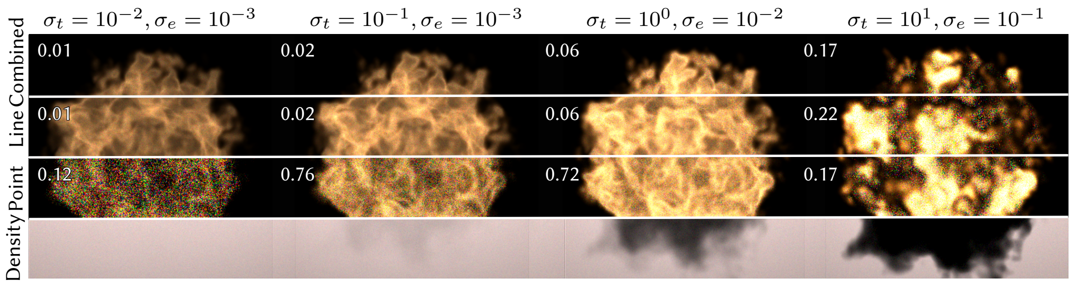

$\sigma_t = 10^{-2}, \sigma_e = 10^{-3}$
$\sigma_t = 10^{-1}, \sigma_e = 10^{-3}$
$\sigma_t = 10^{0}, \sigma_e = 10^{-2}$
$\sigma_t = 10^{1}, \sigma_e = 10^{-1}$
Combined
Line
Point
Density
0.01
0.02
0.06
0.17
0.01
0.02
0.06
0.22
0.12
0.76
0.72
0.17

Figure 5.9: The fireball from figure 5.1 rendered with the point (section 5.3.1), line (section 5.3.2) and the combined point+line estimator (section 5.3.3) for varying densities. The emission is scaled to maintain roughly the same brightness. The density values are scaled from left to right, $\sigma_s = 0$ for all cases. The RMSE after 128 samples per pixel is given for each image. Compared to our integration method, which is robust for several orders of magnitude of density values, the point integration becomes worse the thinner the medium gets.

estimator performs better than the line estimator in this case, although it does not reach the quality of the point estimator for dense volumes.

5.5.2 Forward next event estimation

Comparison to NEE

In figure 5.3 a simple test scene shows the problem arising when line integration is used with regular NEE (middle image). The high variance paths created by forward BSDF sampling cannot be combined with next event estimation using MIS in this case, as NEE cannot create pass-through paths which contribute due to segment emission. On the other hand, forward next event estimation is able to create pass-through paths. Combining BSDF sampling with FNEE using MIS results in an overall superior result, also compared to point integration with NEE (right image).

Volume	Size	#Voxels	Figure
wedge	10 MB	$\approx 2\times10^6$	5.6
flame	28 MB	$\approx 3\times10^6$	5.1, 5.3, 5.11
swirl	150 MB	$\approx 21\times10^6$	5.11
fireball	390 MB	$\approx 39\times10^6$	5.1, 5.9, 5.8
explosion	3.5 GB	$\approx 349\times10^6$	5.12, 5.4, 5.10
sun	30 GB	$\approx 6903\times10^6$	5.13

Table 5.1: Size of the volumetric data sets used in this chapter.

Computing the solid angle PDF

FNEE requires computing the solid angle PDF for a sampled direction (equation 5.24). As detailed in section 5.4 we compute the PDF by integrating along a ray through a coarser version of the volume. In our implementation we can simply choose one of the hierarchical grid levels. Next we evaluate the impact of using coarser levels.

For this test, we computed the irradiance for two locations p_0 and p_1, illustrated in figure 5.10, outside a volume where next event estimation is the most useful. Interestingly, the equal-sample comparison of the first row show that the choice of coarseness does not have a large impact on the sample quality in this test, even though the solid angle subtended by the volume differs greatly at these two locations. The line integration is tolerant to slightly more inaccurate initial point sampling, and compensates for that by gathering the emission at high resolution along path segments.

The middle column in figure 5.10 shows equal-time comparisons. FNEE-0 has a larger RMSE because it is much more expensive than FNEE-1 and FNEE-2. All renders use the second finest grid (FNEE-1) for forward next event estimation except for the explosion in figure 5.12 and figure 5.4 where FNEE-2 is used. This shows that even though the explosion data set has 349×10^6 voxels, we can use a 64^3-version of the volume to perform forward next event estimation.

Volume density

The density of a volume also influences the efficiency of next event estimation. If the volume is dense, then it is likely that emission at a sampled point is irrelevant because of high extinction in the medium. The last column of figure 5.10 shows the RMSE after 100ms plotted for varying densities. We can see that FNEE gains advantage over NEE for increasing σ_t.

5.5.3 Test scenes

We performed equal-time renders for different scenes and compared next event estimation using point integration and forward next event estimation using line integration. In all our test cases, FNEE delivers results with significantly less residual noise than NEE.

In figure 5.12 we rendered a city containing the explosion data set with two different values for σ_t. The scattering parameters are $\sigma_t = 0.2$, $\sigma_s = 0.01$, $\sigma_e = 0.1$ for the thin version (left, 10 min), and $\sigma_t = 40$, $\sigma_s = 0.1$, $\sigma_e = 64$ for the dense version (right, 30 min). The overall density of the resulting volumes is shown in insets where the volume is rendered with $\sigma_e = 0$. The image resolution is 1024×576.

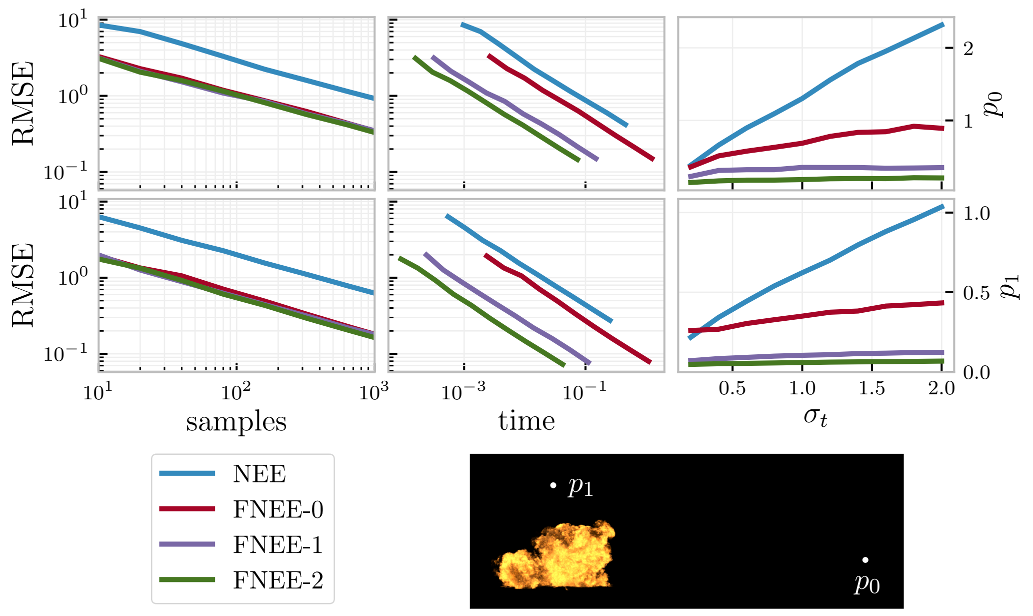
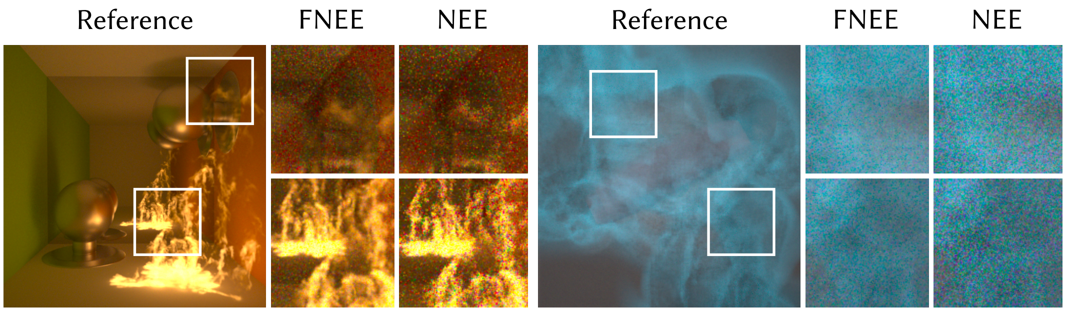


Figure 5.10: The irradiance is computed for two locations p_0 and p_1 outside the volume, where next event estimation is the most useful. The RMSE is plotted for an equal amount of samples (left column) and time (middle column). The notation FNEE-i means that the grid i levels above the leaf level is used for FNEE (FNEE-0 is the finest). The last column shows the RMSE after 100ms plotted for varying densities demonstrating the advantage of FNEE for dense volumes. Note how inefficiently NEE behaves for increasingly dense media. The first two columns correspond to $\sigma_t = 1$.

The thin version greatly benefits from line integration as can be seen in the insets showing directly visible parts of the volume. FNEE is much more efficient than NEE in the dense version: FNEE rendered more than twice as many samples as NEE in the same time frame producing a much better result than NEE.

Figure 5.11 shows equal-time comparisons between NEE and FNEE in two scenes (resolution 512×512). The upper scene contains a thin flame in a simple closed box. The flame is reflected by the mirroring back wall as well as the glossy objects. The renders are equal-time (15 min). The insets show that FNEE outperforms NEE not only in parts where the flame is directly visible, but also in indirectly lit parts of the scene. In this scene FNEE is slower than NEE (FNEE: 47 spp, NEE: 70 spp) because the volume is thin and FNEE passes through the volume most of the time. Still, the RMSE compared to a converged reference is greatly reduced (FNEE: RMSE 0.13, NEE: RMSE 0.21). The bottom scene shows a frame of a fluid simulation where a rotating object mixes two initially separated layers of fluid with different densities. The volumetric emission in the fluid is proportional to the velocity,



Scene	Time	Samples per pixel		RMSE	
		FNEE	NEE	FNEE	NEE
Box	15 min	47	70	0.13	0.21
Swirl	10 min	134	174	0.05	0.11

Figure 5.11: Equal-time comparisons of forward next event estimation and line integration (FNEE) and next event estimation with point integration (NEE) in a closed box containing a thin flame that is reflected in the mirror back wall and a fluid simulation containing a rotating object causing a swirl. In this scene volumetric emission is proportional to the velocity of the fluid which imitates bioluminescence. Even though FNEE results in less samples in these scenes per pixel the images contain less residual noise. Render times, sample count and remaining RMSE values are given in the table.

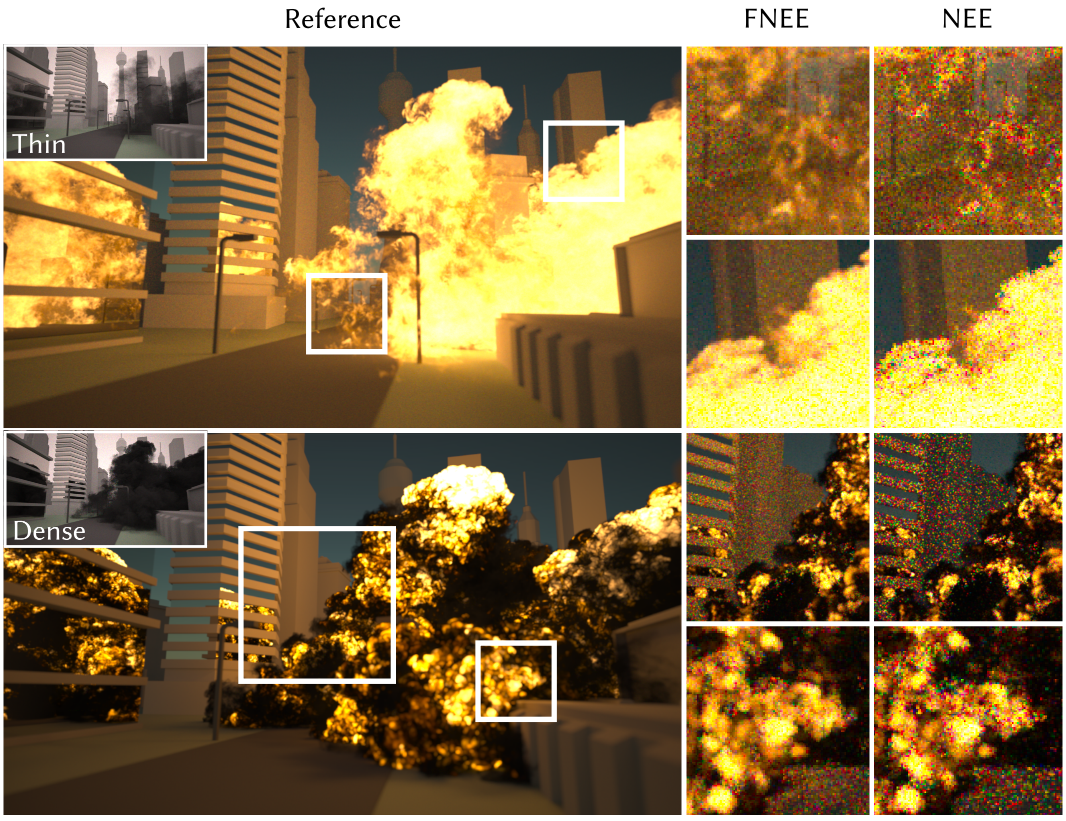
imitating bioluminescence. The renders took 10min and FNEE again has lower remaining noise (RMSE 0.05) compared to NEE (RMSE: 0.11) even with lower sample count (FNEE: 134 spp, NEE: 174 spp).

In figure 5.13, we rendered a 30 GB scene to evaluate the cost of FNEE for larger volumes. As expected, the computational cost increases (PDF accumulation, longer segments in thin regions), but FNEE still pays off for equal-time renders.

5.6 Limitations

This chapter specifically addressed the challenges with rendering emissive volumes. Next event estimation in thin scattering fog, for example, is a different problem and is usually addressed with equi-angular sampling [Kulla and Fajardo 2012] or by creating a CDF along a ray. The latter approach may also prove useful for emission, e.g., if very bright lights are hidden behind thick layers of absorbing smoke.

While our techniques would transparently work with Kelemen Metropolis [Kelemen et al. 2002], it is not as straightforward for path space methods derived from Metropolis Light Transport [Veach and Guibas 1997]. This might be an interesting direction for future work.



Scene	Time	Samples per pixel		RMSE	
		FNEE	NEE	FNEE	NEE
Thin	10 min	48	43	0.748	0.983
Dense	30 min	196	73	0.572	0.892

Figure 5.12: These images show equal-time comparisons of a heterogeneous emissive volume, in two versions that differ in overall density. The thin version greatly benefits from the line integration estimator that accumulates emission for path segments and not only at path vertices. In the dense version of the volume forward next event (FNEE) samples the length of next event segments proportional to transmittance. This prevents long segments that do not contribute to the final image due to high extinction and increases the efficiency of path tracing compared to regular next event estimation (NEE). The table shows sample counts and remaining RMSE error values for forward next event estimation and line integration (FNEE) compared to regular next event estimation with point integration (NEE).

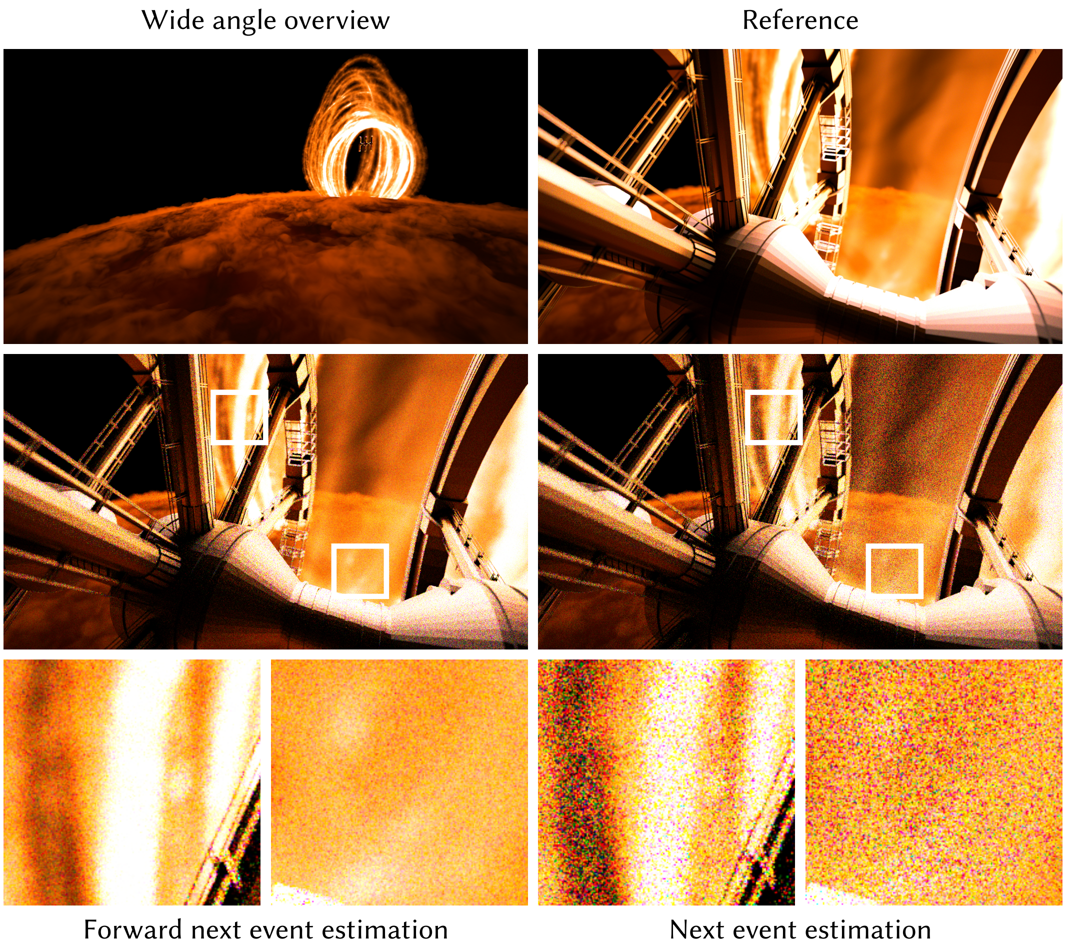


Figure 5.13: Scalability test on a 30 GB volume. These images show equal sample counts (128 spp) with a RMSE of 52.6 for NEE and 36.7 for FNEE. The time for NEE was 7.5 and for FNEE 11.2 core hours, i.e., the time one core would need to create the image. Most of the light comes from the relatively thin loop – a case which works reasonably well with regular NEE. Note that this is a difficult scenario for FNEE as it creates long segments through the thin medium, i.e., it has a higher cost per sample than NEE. However, this is still amortized by collecting the emission along the path segment: the RMSEs for equal-time (60 core hours) are 14.9 FNEE (704 spp) and 18.6 NEE (1024 spp).

We did not try using segment emission in bidirectional path tracing but it might be interesting to investigate this in future work. For deterministic connections, the segment emission can be computed in the same way as we already do for NEE paths to other light sources. However, MIS may need to be adjusted to account for the fact that paths sharing a common prefix close to the sensor can result in the same contributions (similar to when picking up point emission along all path vertices, not just the end points).

5.7 Conclusion

We presented a line integration estimator that accumulates emission for path segments, not only at path vertices. This now allows Monte Carlo light paths to contribute to the image in cases when no scattering event inside the volume has been sampled. This greatly improves Monte Carlo rendering in thin parts of emissive media.

We also proposed forward next event estimation, which combines sampling points according to emission and segment length according to transmittance. This prevents long segments that do not contribute to the final image in volumes with high extinction.

Whereas line integration works best for thin emissive volumes, FNEE is particularly beneficial for dense media with high extinction. Used together, they provide robust and efficient rendering of a wide range of emissive heterogeneous volumes, and we showed that resulting images can have greatly reduced Monte Carlo noise.

6 Selective path guiding

In chapter 4, we discussed a specialized global importance sampling technique for choosing good locations for virtual point lights in the many-light rendering context. However, finding practical global importance sampling strategies for general Monte Carlo light transport is challenging.

While estimators using local methods such as BSDF sampling or (forward) next event estimation (chapter 5) often work well in the majority of a scene, small regions of path space can sometimes be insufficiently sampled. In this chapter we propose a novel data-driven guided sampling method which selectively adapts to such problematic regions and complements the unguided estimator.

It is based on complete transport paths, which enables it to resolve correlation due to BSDFs and free flight distances in participating media. It is conceptually simple and places anisotropic truncated Gaussian distributions around guide paths to reconstruct a continuous probability density function (guided PDF). Guide paths are iteratively sampled from the guided as well as the unguided PDF and only recorded if they cause high variance in the current estimator.

Conventional Monte Carlo techniques sample paths independently, and Markov chain-based methods perturb a single path. In contrast, our technique considers multiple previously sampled paths to construct a new one by determining reconstruction kernels from a set of neighboring paths. This enables local exploration of the integrand without detailed balance constraints or the need for analytic derivatives.

We show that our method can decompose path space into a region that is well sampled by the unguided estimator and one that is handled by the new guided sampler.

6.1 Introduction

Photorealistic image synthesis is essential for product visualization, architecture, and visual effects in movie production. In these areas, Monte Carlo light transport simulation has found widespread adoption due to its ability to simulate complex lighting effects accurately. In this class of techniques, path tracing is still the most frequently used technique among practitioners, due to its simplicity and easy maintenance. This relatively simple solution — starting at the camera sensor, iteratively extending transport paths to more bounces, combined with next event estimation (i.e., deterministically connecting to the light sources) — can efficiently account for the vast majority of rendering problems in practice [Fascione et al. 2017].

However, path tracing often encounters transport phenomena, which it fails to sample sufficiently. Although the source of the difficulty may be very localized, it can cause noise virtually everywhere in the image.

More robust solutions have been proposed, e.g., [Veach 1998; Georgiev et al. 2012a; Hachisuka et al. 2012]), but have not found broad adoption so far. The reasons are many: they often suffer from temporal flickering or blurry artifacts, and some cannot handle special requirements such as hair strands or participating media – or the computational or memory overhead they introduce is just too high.

In this chapter, we present a novel method that addresses these issues and balances local exploration and global discovery of transport paths. The global discovery uses stratified sampling with uniformly distributed points, e.g., a path tracer using the Halton sequence or bidirectional path tracing (BDPT, [Veach 1998, chapter 10]). As a basis for the local exploration, we store samples causing high variance as *guide paths*. The key is to store full paths, as opposed to a disconnected set of endpoints [Vorba et al. 2014] or path segments [Jensen 1995]. By this, we can make use of all the available information during local exploration by creating new paths which are sampled using Gaussian distributions around the guide paths. These newly spawned paths are often selected as new guide paths for subsequent sampling.

In particular, our method is selective in the sense that we introduce an iterative learning process that identifies sub-spaces of the integration causing high variance. It also uses information from complete paths and accounts for path length, BSDF, visibility, and free path length in participating media.

To sample a new path, we use information from a *set* of neighboring paths, which were already sampled. This is a crucial difference to conventional Monte Carlo methods, which construct independent samples, and Markov chains, which mutate from a single current state sample.

We show that our approach can work in a variety of scene configurations, including cases that are difficult for previous work such as participating media, hair fibers.

6.2 Background and previous work

In this chapter we are again concerned with computing the light transport in a scene. This involves computing the amount of radiant energy I_j incident on a pixel j as described by the path integral formulation of light transport equation 2.44, i.e.,

$$I_j = \int_{\mathcal{P}} f_j(\mathbf{X})\mu(\mathbf{X}), \tag{6.1}$$

where f_j is the measurement contribution function (equation 2.51), $\mathcal{P}$ is the path space and a path $\mathbf{X}$ of length k is formed by a list of vertices $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_1, .., \mathbf{x}_k)$; $\mathbf{x}_1$ being on the sensor and $\mathbf{x}_k$ on a light. Many Monte Carlo (MC) [Sobol 1994] or Markov chain Monte Carlo (MCMC) approaches [Metropolis et al. 1953; Hastings 1970] have been devised to solve integrals of the form of equation 6.1 and have been applied to light transport [Pharr et al. 2017; Kelemen et al. 2002; Veach 1998]. While MC is usually good at uniformly discovering islands in the path space, MCMC approaches are better at exploring them once found. A hybrid approach exists [Cline et al. 2005], but often requires manual, scene-dependent intervention to determine what should be explored by MC or MCMC to avoid wasteful sampling and unnecessary flickering in animations. Our technique explicitly determines which parts of the path space get explored well by MC, and will then spend additional work only on the remaining parts. We achieve this by storing transport paths in subspaces which are intricate to sample and by this acquire information about the transport in a scene. This is in contrast to MCMC methods which always only depend on a single current state.

Advanced (MC)MC light transport methods

Previous work exploits additional information to guide local exploration in MCMC. In particular, geometric derivatives [Jakob 2013] have been used to globally optimize all vertices of a path simultaneously. This work clearly shows the interdependencies of path vertices along a full path in form of a block-tridiagonal matrix, which maps half vector constraints to vertex area locations (cf. [Jakob and Marschner 2012, Figure 2(c)] or [A. Kaplanyan et al. 2014, Equation (11)]). This motivates our approach of working with complete transport paths.

Analytic derivatives of the measurement contribution have been explored to help with computing step sizes [Li et al. 2015; Fasiolo et al. 2018]. Instead, we employ an observed light field, since fine displacements and visibility make analytical derivatives unstable [Hanika et al. 2015].

MCMC has also been augmented with large caches (photon maps) to represent visibility and information beyond a single state [Hachisuka and Jensen 2011; Šik et al. 2016; Gruson et al.

2016]. However, we want our method to focus only on small subspaces of the path space where exploration is difficult. Besides, photon maps do not encode the full information about complete transport paths, but represent a 2D marginal distribution (4D if incoming directions are stored).

High-dimensional importance sampling

Metropolis light transport can importance sample complete paths [Veach 1998, chapter 11] in the limit by performing many small mutations. There are approaches to incorporate more global information into regular MC sampling. Joint importance sampling [Georgiev et al. 2013] successively derives joint probability density functions (PDFs) for sampling 2 to 3 vertices in participating media at once. Weber et al. [Weber et al. 2017] use an uncached, deterministic path construction to construct in the order of 10 vertices inside a medium towards a light source as an extended next event estimation. Closed form approximations have been used to guide scattering events towards a light source in dense media [Křivánek and d'Eon 2014; Meng et al. 2016]. Note that all of these are strictly limited to participating media.

Particle filters

Sequential Monte Carlo or particle filters [Del Moral 1996] are usually used to discover the posterior distribution of a hidden Markov model (for instance tracking one or multiple moving 2D targets over time). They can be used to estimate a PDF by first sampling an initial set of particles followed by a move and resampling step. As this approach suffers from progressive degeneration, i.e., it tends to get stuck in a single mode, many countermeasures have been proposed, see e.g., [Gilks and Berzuini 2001]. In our context, however, we are computing a static equilibrium distribution of light, so we do not move our discovered samples over time. Also resampling means discarding some information from the previous iteration which did not work well for light transport. However, we do employ a similar resampling step for dynamic scenes.

Population Monte Carlo (PMC) [Cappé et al. 2004] and adaptive importance sampling [Cappé et al. 2008] also derive parameters for a PDF from previous samples. PMC used to be most effective in 2D and has been used to determine various 2D kernels for instance for lens perturbations [Fan 2006] and energy redistribution path tracing [Lai et al. 2007]. Recent work extends it to higher dimensional cases, e.g., particle MCMC [Andrieu et al. 2010].

Previous work suggests that light transport lends itself well for a decomposition into 2D subdomains (one per bounce), e.g., as done with padded replications for low discrepancy sampling [Kollig and Keller 2002a; Kollig and Keller 2002b]. However, maintaining the correlation between bounces as dictated, for instance, by non-diffuse BSDFs is not possible. Thus we cannot use a padded 2D replication of particle filters.

Learning-based methods

Despite many existing elaborate sampling techniques, learning-based approaches can improve the simulation by using information from previously sampled paths and guiding MC samples to regions where more samples are required. Care has to be taken to maintain the unbiasedness of an estimator, even when working in image space only [Kirk and Arvo 1991]. Guiding can also be performed by learning the 2D incoming light field at cache records distributed throughout the scene [Vorba et al. 2014], and has been combined with reinforcement learning [Dahm and Keller 2017a; Dahm and Keller 2017b]. The work of Hey and Purgathofer [Hey and Purgathofer 2002] uses kernels around guide samples to reconstruct a signal and is in that sense similar to our approach, but only for a 2D hemispherical signal.

Note that all of these approaches only consider a 2D slice of the path space at once and thus need to explicitly differentiate modes at each cache point. Even including BSDF information requires costly processing [Herholz et al. 2016]. Instead of clustering, we store complete paths and by this keep the entire information present during path sampling as well as the dimensions of the path vertices separated. This allows us to fit simpler unimodal distributions to the local surroundings of a path sample, and include correlation between the bounces such as information about the path length.

Spatial subdivision schemes for adaptive sampling, e.g., [Müller et al. 2017; Hachisuka et al. 2008a], are appealing because of their simplicity, but are intractable in higher dimensions. Subdividing every dimension only once in a 32-dimensional space would result in 2^{32} bins to store. In contrast, we store a subset of full transport paths only, and define a continuous PDF around them for guiding.

6.3 A selective guiding framework for complete paths

Computing the integral over a function f with a regular, unguided MC estimator often works well for some part of the integrand, but others are not well represented by the underlying unguided probability density function (unguided PDF). The goal of our work is to derive and add a new *guided PDF* to the existing one, as to make the sum of both proportional to the integrand f (section 6.3.1). This is equivalent to adding a PDF in the context of multiple importance sampling [Veach 1998, chapter 9], using the balance heuristic. Since data-driven guiding is expensive, we want to apply it only where needed, i.e., where

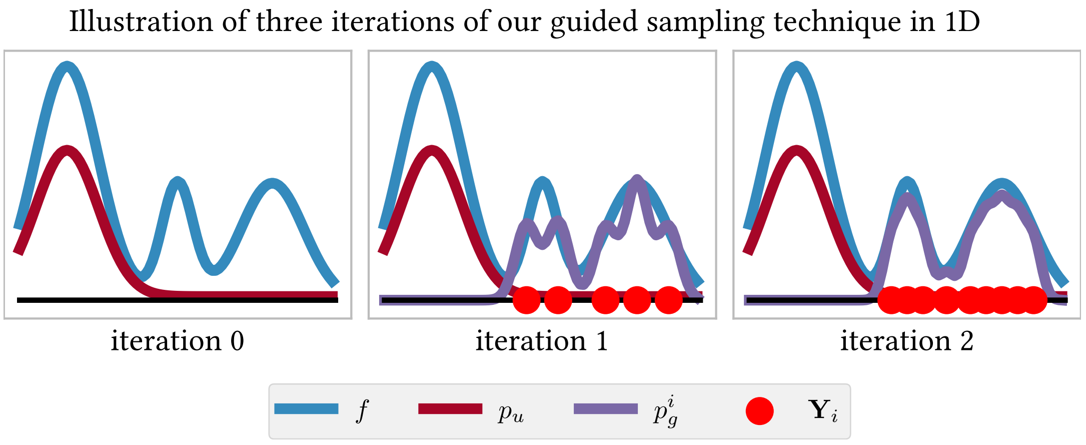


Figure 6.1: A 1D illustration of guided sampling. Initially, parts of the integrand f are well represented by a PDF p_u, whereas other parts are not. We wish to approximate the difference between the integrand and the given PDF p_u. Guiding records $\mathbf{Y}_i$ are iteratively placed and then used to reconstruct a continuous guided PDF p_g^i in the i-th iteration. These two PDF are then combined with multiple importance sampling (balance heuristic) to yield a good Monte Carlo estimator for the integral.

existing techniques fail to converge in acceptable time. To construct this new PDF, we draw samples from the unguided PDF, and record samples which cause high variance in a cache along with their sample weight, f divided by the sampling density, as *guide samples* (section 6.3.2). We use complete transport paths as samples, to retain as much information as possible. From these, we reconstruct a continuous PDF using a Gaussian mixture model. To sample from this PDF, we randomly pick one guide sample by its weight, and then sample from a Gaussian distribution centered around it (section 6.3.3). The Gaussian kernel's size depends on the neighboring guide samples (section 6.3.4). Learning the guided PDF is sped up by running this process iteratively on small sample batches, also sampling from the guided PDF learned so far.

6.3.1 Overview

In figure 6.1, an example integrand $f(\mathbf{X})$ is shown in blue, and the existing MC estimator draws samples from the PDF shown in orange, which we denote as the *unguided PDF* $p_u(\mathbf{X})$. More formally, we are approximating the integral using the estimator:

$$\langle I(\mathbf{X})\rangle_u = \frac{f(\mathbf{X})}{p_u(\mathbf{X})}. \tag{6.2}$$

In our example, the PDF p_u represents the left smooth mode of the integrand well, but misses the other features (two more modes).

Thus we construct the additional *guided PDF* $p_g^i(\mathbf{X})$ (shown in green) which is iteratively refined for $i = 0, 1, ...$, and captures the differences between the integrand $f(\mathbf{X})$ (blue) and the unguided PDF $p_u(\mathbf{X})$ (orange). The converged guided PDF is shown in figure 6.1, right. Eventually, we want to combine these two PDFs using multiple importance sampling (MIS). More precisely, we will be using a single-sample model with the balance heuristic [Veach 1998, chapter 9] resulting in a combined estimator with a mixing weight u, i.e.,

$$\langle I(\mathbf{X})\rangle_g^i = \frac{f(\mathbf{X})}{up_u(\mathbf{X}) + (1-u)p_g^i(\mathbf{X})}. \tag{6.3}$$

The guided PDF will be constructed incrementally and with every iteration i we adjust $p_g^i(\mathbf{X})$ to further reduce the variance of equation 6.3.

Intuitively, we proceed as illustrated in figure 6.2. We begin by letting $p_g^0(\mathbf{X}) \equiv 0$ and effectively sampling $\mathbf{X}$ using only $p_u(\mathbf{X})$ (figure 6.2a). From the paths $\mathbf{X}$, we select only a few new guide paths $\mathbf{Y}$ motivated by importance sampling: we first determine whether a path is an outlier causing high variance (via *density-based outlier rejection* [DeCoro et al. 2010]) and only from these we pick the N with the highest contribution to the estimate in equation 6.3. We will iteratively add batches of guide paths $\mathbf{Y}_j$ sampled from both p_u and the current guided pdf p_g^i to the cache, and once they are added we keep them unchanged until the end. This is illustrated in figure 6.2b and detailed in section 6.3.2.

The guide paths $\mathbf{Y}_j$ are turned into a continuous function using a Gaussian reconstruction kernel (figure 6.2c). We use a high-dimensional neighbor search to determine large enough reconstruction kernels to close the gaps between paths and to achieve smooth coverage. One challenge is that the PDF evaluation can become slow in Gaussian mixture models. Therefore we use truncated Gaussian kernels to be able to cull away samples efficiently. Our Gaussians are truncated at $\approx 3.330\,\sigma$.

To sample from the cache, we first select a guide path $\mathbf{Y}_j$ using a cumulative density function (CDF) built on path weights w_j which are updated every iteration i to reflect the new guided PDF $p_g^i(\mathbf{X})$ (see section 6.3.4). Then we sample a new path vertex $\mathbf{x}_v$ following the Gaussian reconstruction kernel $(\Sigma, \mu)_v$ around every guide path vertex $\mathbf{y}_v$ (figure 6.2d), starting at the sensor (see section 6.3.3).

In the subsequent iteration $i + 1$, we draw samples from both $p_u(\mathbf{X})$ and $p_g^i(\mathbf{X})$, with probabilities u and $1 - u$, respectively. No matter which technique was used, new samples

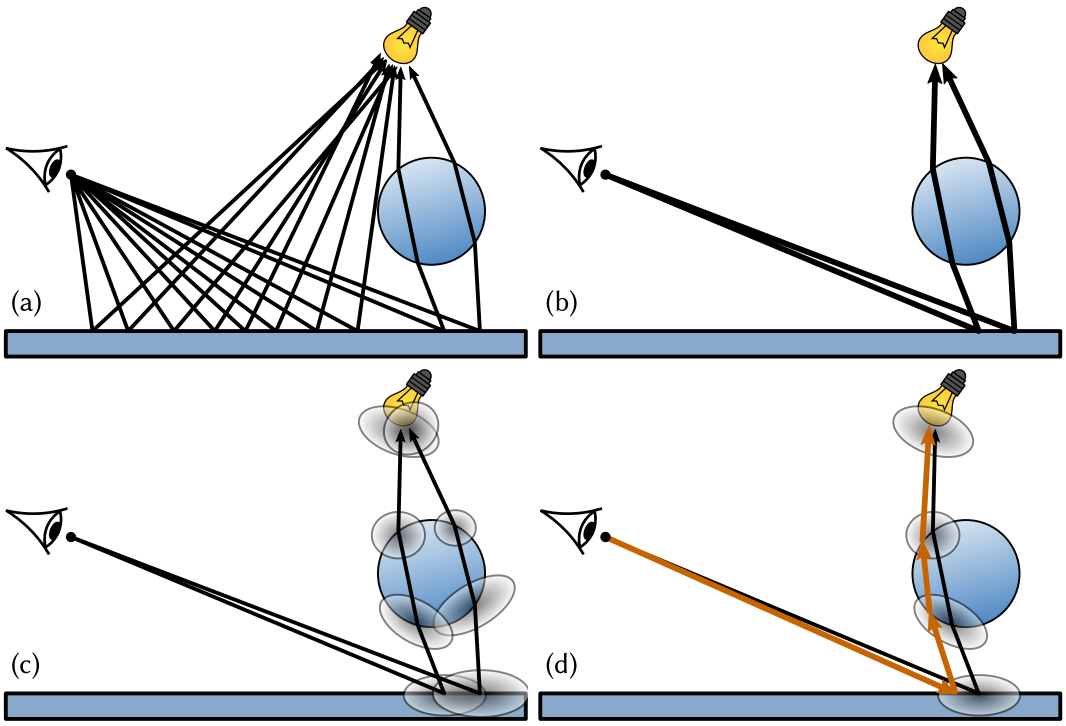


Figure 6.2: Schematic overview of the guiding method employing full paths. We start with all path traced trajectories, depicted in (a). From these, we select the ones that are most poorly sampled by the underlying Monte Carlo estimator, shown in (b). For each of these guide paths, we compute anisotropic reconstruction kernels at every path vertex, to fill in the gaps between the samples. This is visualized in (c). To sample from this structure, we first pick a guide path and then successively sample path vertices according to the reconstruction kernels (d). Evaluating the PDF requires to sum up the PDF associated with all guide paths that may create the given path.

will be considered to be recorded as new guide samples for the next iteration, depending on their contribution $\langle I(\mathbf{X})\rangle_g^i$.

There are concerns whether an estimator is still unbiased when changing weights and kernel parameters of a PDF based on particles adaptively [Kirk and Arvo 1991; Douc et al. 2007]. In our case we will use multiple importance sampling (MIS) with an unguided estimator, which guarantees an unbiased combination. For that we require $p_u(\mathbf{X})$ to form a complete unbiased estimator, in particular for all paths $\mathbf{X}$ we require

$$f(\mathbf{X}) > 0 \Rightarrow p_u(\mathbf{X}) > 0. \tag{6.4}$$

This will not be the case for our guided PDFs $p_g^i(\mathbf{X})$ which may be zero in areas where the original estimator is deemed sufficiently good already. We chose the balance heuristic to combine the PDFs as it is close to optimal [Veach 1998, chapter 9]; note that it is not relevant for the final sampling quality, as the combination of both PDFs $p_u(\mathbf{X})$ and $p_g^i(\mathbf{X})$ adapts to be proportional to $f(\mathbf{X})$.

Learning and sampling from $p_g^i(\mathbf{X})$ has several key advantages. The algorithm leverages both global stratification via regular (quasi-)Monte Carlo sampling and local exploration due to the Gaussian. Also, sampling is fast, since only one guide sample needs to be chosen and the corresponding Gaussian needs to be sampled. We only need a careful implementation of the PDF evaluation for good performance. A key property of this algorithm is, that it focuses on parts of the integration domain where unguided sampling performs poorly. Compared to MCMC where step sizes are often guesswork, we use neighboring samples to derive sampling spread. This is more robust than using analytical derivatives which can fail, e.g., due to fine displacements and visibility. We also show that guide path caches can easily be reused across frames in an animation (section 6.3.6).

6.3.2 Selecting new guide paths

In every iteration i we adapt the guiding cache to the residual variance present in the current estimator $\langle I\rangle_g^i$ (equation 6.3). The cache stores full transport paths consisting of one point on the sensor, one point on an emitter, and a list of vertices in between.

We generate a batch of such path samples from both the unguided and guided estimator (e.g., one sample per pixel), and select paths that are causing the highest variance in the current combined estimator $\langle I\rangle_g^i$. These samples need to have a high contribution in order to cause a big difference to the expected value. High contribution can be caused by high measurement as well as low PDF. The former does not necessarily cause high variance if sampled well (e.g., looking directly into the sun). We therefore use *density-based outlier* *rejection* (DBOR) [DeCoro et al. 2010; Zirr et al. 2018] to distinguish between high sample contributions caused by bright measurement and those caused by low PDF value. From the latter, we keep the N paths with the highest contribution as new guide paths. This avoids selecting new guide paths in regions that have already been sampled well in previous iterations. For implementation details regarding DBOR, see section 6.4.4. As sample weight associated with each selected path $\mathbf{Y}_j$, we store the value of the estimator equation 6.3 (section 6.3.4 details how the weights are computed).

We chose to add only N (in the order of several hundred) new samples in every iteration, as in our experiments this led to improved learning performance; an observation supported by previous work [Chopin et al. 2013]. We add unguided as well as guided samples to the cache since both have the ability to discover new features: one globally, the other locally. In contrast to particle filters, once a path is added to the guiding cache, it is never removed or resampled, since in our experiments this led to loss of information, oscillation, and clumping of guide paths (cf. section 6.3.4, section 6.6). Only the associated weight and kernel parameters change over time.

6.3.3 Sampling from the cache

To sample from $p_g(\mathbf{X})$, we first select a guide path $\mathbf{Y}$ by sampling the CDF built on the path weights w_j. Having selected a guide path representing the difference between the unguided PDF $p_u(\mathbf{X})$ and the measurement contribution function $f(\mathbf{X})$, we need to reconstruct a continuous PDF $p_g(\mathbf{X})$ around it. This is accomplished with a reconstruction kernel: at every high-dimensional control point (a guide path) we place a truncated Gaussian with a covariance matrix of the same dimensionality adapted to a number of nearest neighbors. Since the measurement contribution function is a separable product of terms and since we will be sampling full paths incrementally by constructing one vertex at a time, we assume that this high-dimensional covariance matrix has a block-tridiagonal structure. In section 6.3.5 we illustrate why this is a valid assumption. Thus we only need to compute a sparse set of coefficients relating up to three consecutive path vertices to each other.

We create path vertices $\mathbf{x}_v$ one by one, starting at the camera: First, we sample $\mathbf{x}_1$ on the camera lens as for regular path tracing. Then, successively for every vertex $v > 0$, we compute the parameters Σ_v, μ_v of a truncated Gaussian distribution $\mathcal{N}[\Sigma_v, \mu_v]$ from which we will sample $\mathbf{x}_v$.

For volumes, we sample a 3D truncated Gaussian. For surface points $\mathbf{x}_v$, we first sample $\mathbf{x}'$ according to the 2D Gaussian in the frame of the next guide vertex, and then trace a ray

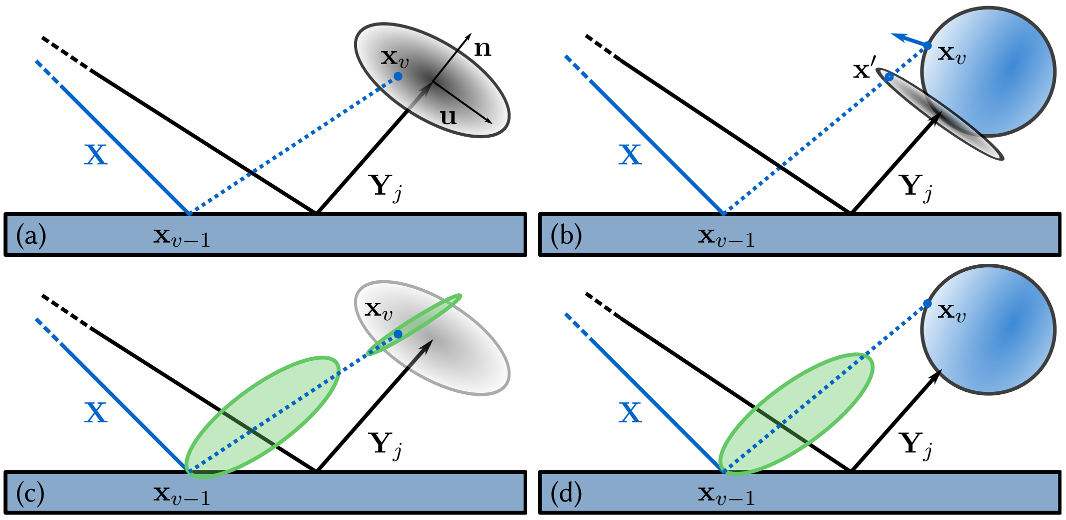


Figure 6.3: Sampling the next vertex $\mathbf{x}_v$ in volumes (a,c) and on surfaces (b,d). The Gaussian distributions are defined in a frame $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ at the guide path vertex, where $\mathbf{n}$ is aligned with the incident path segment. In the surface case (b), the sampled point $\mathbf{x}'$ needs to be projected to the surface geometry. For highly specular BSDF (c,d), we sample from the BSDF instead of from the guide path Gaussian. The volume case (c) then only samples the distance from a 1D Gaussian which is the 3D volume Gaussian conditioned to the sampled ray.

to project $\mathbf{x}'$ to the surface geometry which yields $\mathbf{x}_v$ (see figure 6.3). This step incurs a Jacobian determinant to account for the change of measure:

$$p_g(\mathbf{x}_v|\mathbf{x}_{v-1}) = \mathcal{N}[\Sigma_v, \mu_v](\mathbf{x}')G(\mathbf{x}_{v-1}, \mathbf{x}_v)/G(\mathbf{x}_{v-1}, \mathbf{x}'). \tag{6.5}$$

Obviously (near-)specular BSDFs will pose a problem if outgoing directions are chosen this way: the measurement contribution will likely evaluate to (near-)zero. To cover this case, we mix in outgoing directions sampled from the BSDF as well. We stochastically select between BSDF and Gaussian sampling depending on the Beckmann equivalent roughness of the surface material of the current vertex $\mathbf{x}_{v-1}$. The mode of interaction (reflect or transmit) is constrained to be the same as the guide path's. This case can also be combined with volume sampling: we choose the 2D direction by BSDF sampling, and sample the distance by a 1D Gaussian, by taking the conditional of the 3D Gaussian for this fixed direction (see figure 6.3c).

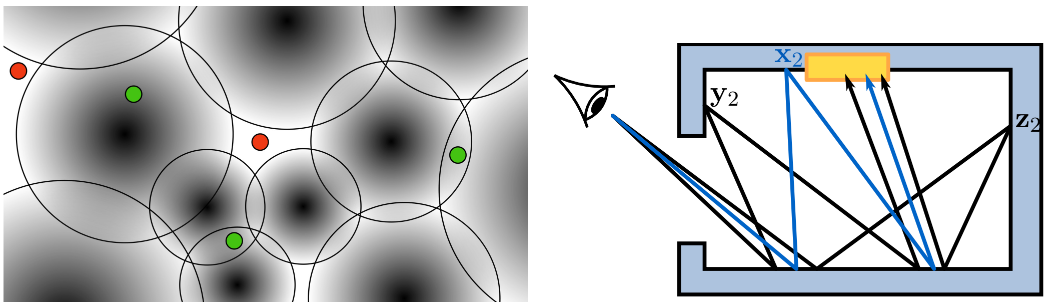


Figure 6.4: Left: A 2D illustration of the guided PDF (black Gaussians). If the unguided sampler creates high variance samples inside the support of the guided PDF (green) their contribution will be effectively reduced by MIS. High variance samples outside of the guided PDF's support (red) get added to the cache in the learning phase. However, during rendering the cache stays fixed and such outliers contribute fully to the image. Right: A diffuse path $\mathbf{X}$ (blue) with 5 vertices sampled in between two guide paths $\mathbf{Y}$ and $\mathbf{Z}$. Their covariance matrices Σ_2 around vertex $v = 2$ need to be very large to define an overlap with $\mathbf{x}_2$. Otherwise, $\mathbf{X}$ cannot be constructed by either guide path, even though they are very close in all other dimensions, and the gap will be filled by the unguided sampler (potentially with higher variance).

To evaluate the guided PDF $p_g(\mathbf{X})$ for a given path $\mathbf{X}$, we need to sum up the PDF of sampling $\mathbf{X}$ given any of the guide paths $\mathbf{Y}_j$, i.e.,

$$p_g(\mathbf{X}) = \frac{\sum_j w_j p_g(\mathbf{X}|\mathbf{Y}_j)}{\sum_j w_j}, , \tag{6.6}$$

where

6.3.4 Determining reconstruction kernel parameters

It remains to compute the Gaussian parameters Σ and μ, as well as the path weights w_j.

Reconstruction kernel size

The kernels should be wide enough to close the gaps between the guide paths in the path space. Otherwise the guided PDF leaves part of the difficult sampling domain to the unguided PDF $p_u(\mathbf{X})$ (figure 6.4). This still results in an unbiased estimator, but may lead to outliers generated by $p_u(\mathbf{X})$. Thus we employ a high-dimensional nearest neighbor search on all guide paths $\mathbf{Y}_j$ of same configuration (length k and succession of reflect, transmit, hair fiber, or volume scattering events). As distance metric, we use the sum of the squared Euclidean distances of the path vertices:

$$d(\mathbf{Y}, \mathbf{Z}) = \sum_{v=0}^{k-1} \|\mathbf{y}_v - \mathbf{z}_v\|^2. \tag{6.7}$$

For every guide path we collect a small number of neighboring paths (we are using 10, but any number between 4 to 20 yielded similar results). To derive the Gaussian to sample the respective next path vertex $\mathbf{x}_v$ given vertex $\mathbf{x}_{v-1}$, we first compute a covariance matrix $\bar{\Sigma}_v$ and the corresponding mean $\bar{\mu}_v$:

$$\bar{\Sigma}_v = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}, \quad \bar{\mu}_v = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \tag{6.8}$$

where the block Σ_{11} expresses how the v-th vertices of the neighbor paths are distributed around their mean μ_1, and $\Sigma_{11} \in \mathbb{R}^{2\times2}$ if $\mathbf{x}_v$ is on a surface, or $\Sigma_{11} \in \mathbb{R}^{3\times3}$ if $\mathbf{x}_v$ is in a volume. Σ_{22} is the same for the previous vertex $\mathbf{x}_{v-1}$, and the dimensionality of the off-diagonal blocks follows accordingly. This means that $\bar{\Sigma}_v$ can be 4×4, 5×5 or 6×6-dimensional and captures how the v-th path vertex behaves with respect to the $(v-1)$-th path vertex. We compute the covariance matrix in an orthonormal frame for every path vertex, where one basis vector is aligned with the incoming ray direction $\mathbf{y}_v - \mathbf{y}_{v-1}$ of the guide path.

To derive an accurate sampling distribution, we compute the conditional $\bar{\Sigma}_v|\mathbf{x}_{v-1}$ to obtain a truncated Gaussian distribution $\mathcal{N}[\bar{\Sigma}_v, \bar{\mu}_v]$ from which we will sample $\mathbf{x}_v$:

$$\Sigma_v = \bar{\Sigma}_v|\mathbf{x}_{v-1} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}, \tag{6.9}$$

$$\mu_v = \bar{\mu}_v|\mathbf{x}_{v-1} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_{v-1} - \mu_2). \tag{6.10}$$

Since equation 6.9 does not depend on $\mathbf{x}_{v-1}$, we precompute and store Σ_v, $(\Sigma_{12} \cdot \Sigma_{22}^{-1})$, μ_1, and μ_2 at the guide path vertex $\mathbf{y}_{v-1}$: We perform a singular value decomposition of Σ_v, regularize the singular values, and keep the rotation matrix and singular values suitable to transform an isotropic Gaussian random variable.

The conditional in equation 6.9 results in small Gaussians, facilitating our goal to yield good sampling efficiency such that, for instance, the sampled rays do actually intersect the light source they aim for (see figure 6.5).

The size of the Gaussian for the first vertex $\mathbf{y}_1$ is not based on the neighboring paths. Instead, we use a simple radius shrinking scheme similar to progressive photon mapping

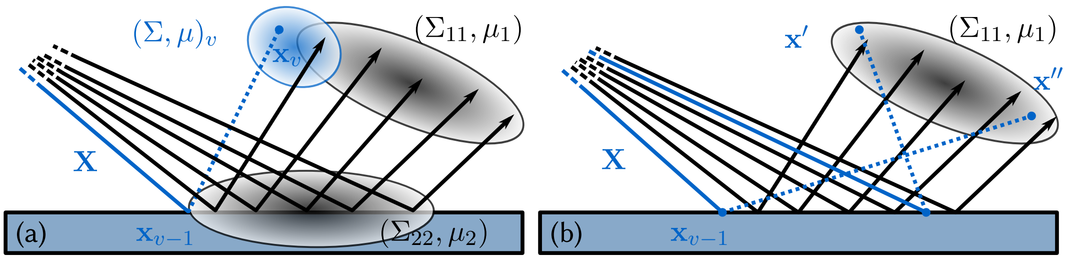


Figure 6.5: For most accurate sampling of $\mathbf{x}_v$, we compute a covariance matrix (5×5 in this example, since $\mathbf{x}_{v-1}$ is a surface point and $\mathbf{x}_v$ is in a volume). One block in the matrix, Σ_{11}, expresses how $\mathbf{x}_v$ should be distributed according to the collected data from the guide paths (shown in black). Because we already have a fixed previous vertex $\mathbf{x}_{v-1}$, we derive the conditional of the 5×5 Gaussian accordingly, resulting in a new distribution $\mathcal{N}[\Sigma_v, \mu_v]$ (blue). If there is a non-zero covariance between the coordinates of the guide path vertices $\mathbf{y}_v$ and $\mathbf{y}_{v-1}$ the conditional Gaussian Σ_v will typically be a lot more focused than Σ_{11}. In this example, disregarding the covariance would amount to sampling according to Σ_{11} with much larger spread, as illustrated in (b). Using Σ_v results in higher quality samples which are more likely to be valid paths, as well as faster PDF evaluation due to culling.

[Knaus and Zwicker 2011] which is based on a ray differential footprint. We successively shrink the size of the Gaussian at $\mathbf{y}_1$ which allows us to control screen space stratification, and ensures that the first Gaussians are not overlapping excessively, making PDF evaluation faster.

An illustration of the progressive sample placement in screen space can be seen in figure 6.6, showcasing the separation into the contribution from the guided and the unguided part of our technique as well as global and local exploration. In figure 6.7 we show a visualization of a guide path, its nearest neighbors from the cache and some sampled paths for the same scenes.

Guide path weights

In contrast to particle filtering methods, we do not employ a resampling step in the learning phase, and we never remove guide paths. Our experiments confirmed that in regions where only very scarce information can be obtained from the unguided sampler, discarding any path can be a substantial loss. Also, learning performs much faster if important paths attract more samples. Thus we aim to keep all guide paths and reevaluate the weights w_j once new paths are added. If the density of paths becomes higher, the weight should

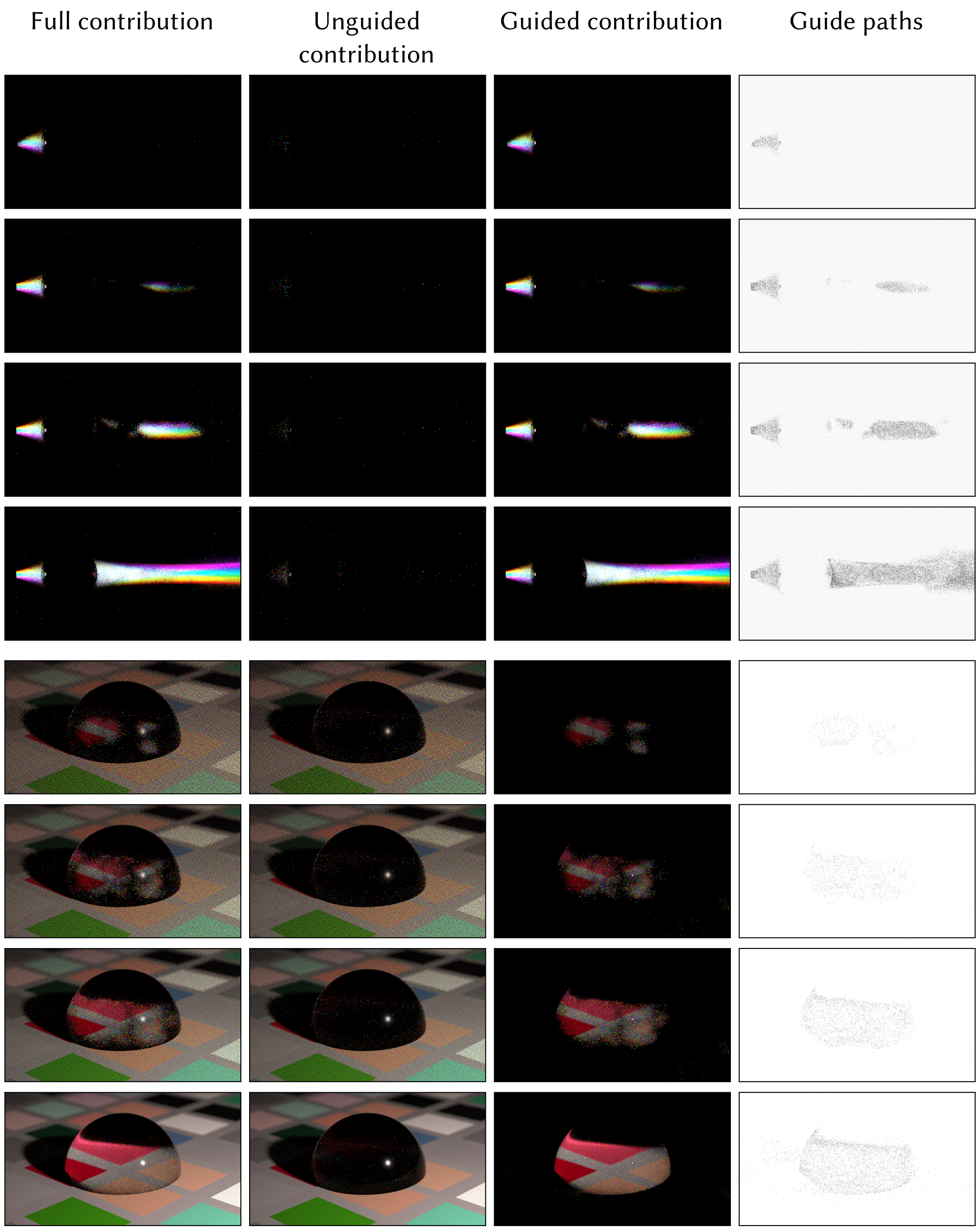


Figure 6.6: Top half: A volumetric caustic cast by a colored spot light through a specular dielectric sphere. Bottom half: A glass sphere on a plane. The rows represent a selection of the consecutive learning progressions from $i = 2$ to $i = 29$. The renders are broken down into contributions from the guided and unguided samplers, as indicated on the columns. While in the volume caustic, virtually all paths require guiding for efficient sampling, the diffuse ground plane in the right part is robustly separated out by our approach. Additionally, we show the guide path locations as black dots to visualize both the even spacing and the local learning. For illustration, we used a constant 10-pixel wide Gaussian for the first vertex in these images.

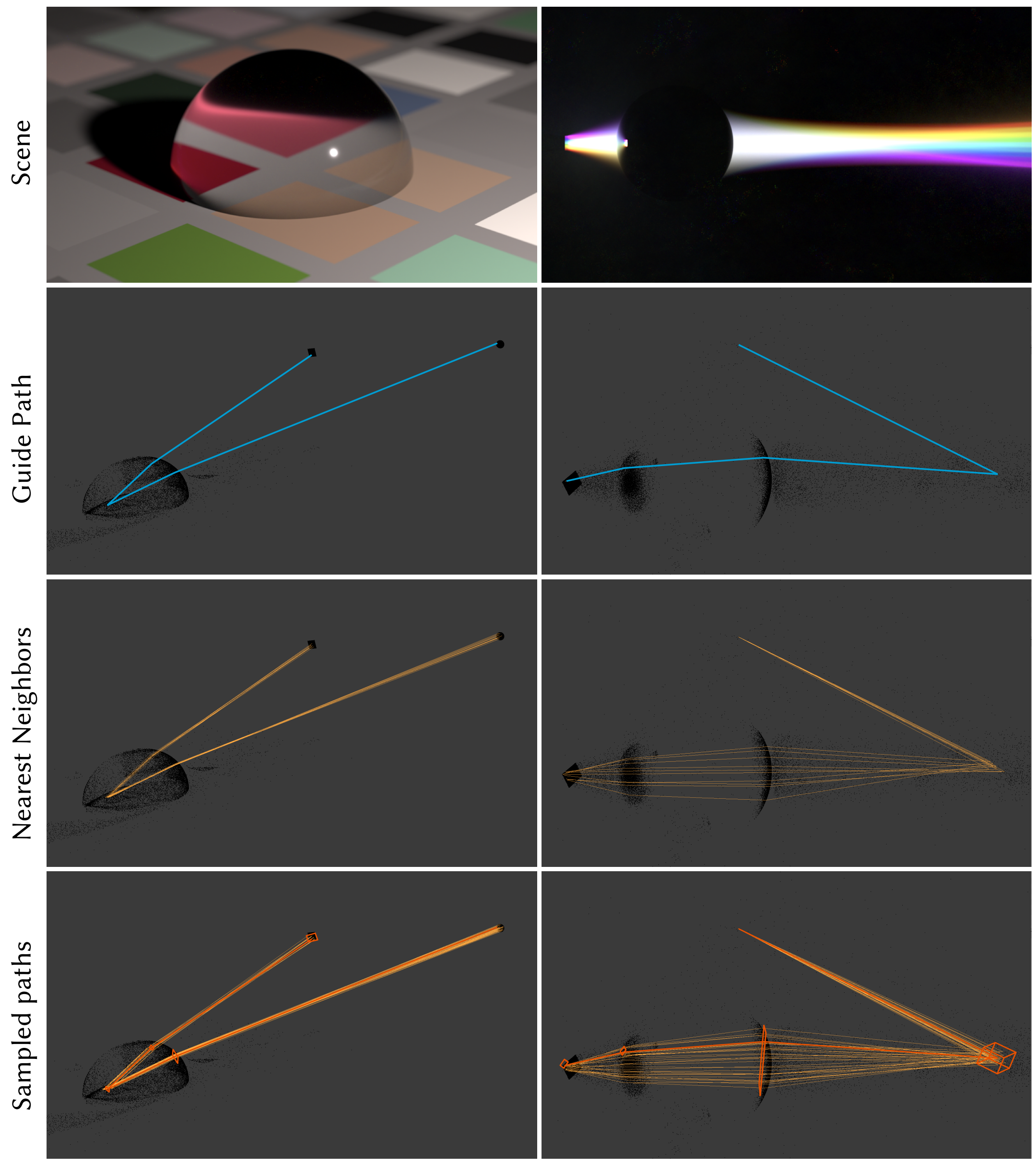


Figure 6.7: A visualization of our guided sampling for the scenes from figure 6.6 (top row). The second row shows a guide path exemplar (blue) and the third row the corresponding 10 nearest guide paths in the cache (yellow). The black points are the path vertices of all guide paths in the cache. The last row shows 32 randomly sampled paths (yellow), including partial paths for which sampling fails during construction. It also shows the bounding volumes (rectangle for 2D surface vertices, box in 3D for volume vertices) of the Gaussians used for one of those sampled path (orange).

become smaller. For importance sampling of guide paths, we should select them by their contribution to the estimator in equation 6.3:

$$w_j = \frac{\langle I(\mathbf{Y}_j)\rangle_g^i}{\sum_l \langle I(\mathbf{Y}_l)\rangle_g^i}. \tag{6.11}$$

In every new iteration i, we need to recompute these weights for all paths in the cache. Unfortunately, in iteration i equation 6.3 depends on the guided PDF $p_g^i(\mathbf{Y}_j)$ which in turn depends on updated weights w_j based on paths sampled in iterations $[0, i]$. This leads to a circular dependency between equation 6.6 and equation 6.11.

As a solution, we propose to use the PDF of the previous iteration $p_g^{i-1}(\mathbf{Y}_j)$ instead. To avoid oscillation, we take an exponential average of previous weights:

$$w_j^i = t\frac{\langle I(\mathbf{Y}_j)\rangle_g^{i-1}}{\sum_l \langle I(\mathbf{Y}_l)\rangle_g^{i-1}} + (1-t)w_j^{i-1}, \tag{6.12}$$

where we choose $t = 0.5$ for paths that were previously contained in the cache, and $t = 1$ for new additions. Note that since t depends on j, these weights do not sum to one any more, so they have to be explicitly normalized in equation 6.6.

6.3.5 Per vertex covariance matrices

It might be surprising, that we can determine useful covariance information for all path vertices by using only 10 nearest neighbors. It becomes clearer, that this approach works in practice by considering that the full covariance matrix of all path vertices is actually quite sparse. This was already developed in previous work [Jakob and Marschner 2012; A. Kaplanyan et al. 2014] and the fundamental observation is that the measurement contribution function (in the surface transport case) mainly changes with the half vector at path vertices. This half vector affects the evaluation of the distribution of microfacet orientations main which is the main factor of most BSDF models. Other factors such as Fresnel term or shadowing and masking geometry terms have less of an impact. The change in half vectors

$\Delta\mathbf{h}$ of a transport path can be expressed in terms of change of vertex locations $\Delta\mathbf{x}$, as a matrix transformation

$$\Delta\mathbf{h} = M \cdot \Delta\mathbf{x} \tag{6.13}$$

$$M = \begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & c_3 & \\ & \dots & & & \\ & & & a_k & b_k \end{pmatrix}, \tag{6.14}$$

where a_i, b_i, c_i are blocks of 2×2, 2×3, 3×2 or 3×3 matrices depending on whether the vertices are in the volume (3D) or on a surface (2D). This matrix M (employed in manifold walks [Jakob and Marschner 2012]) formally reveals the structure of the transport operator: the throughput of the path can be separated into blocks of three consecutive vertices which fully determine the BSDF. In the general case this is also true for transmittance, geometry terms, and more general BSDF than plain microfacet models.

We want to find a local approximation of an ideal sampling PDF, by assuming that in a sufficiently small region around a path, the PDF can be represented by a unimodal Gaussian. If the path has 5 path vertices which are all surface events, the resulting covariance matrix $\mathbf{S}$ of the local Gaussian distribution will be $\mathbf{S} \in \mathbb{R}^{10\times10}$. Similar to the matrix M above, each 2×2 block in $\mathbf{S}$ encodes how the distribution of vertex locations of one particular path vertex depends on the distribution of a certain other path vertex (for instance the first vertex in the scene and the vertex on the light).

Since we need to employ ray tracing to project vertices to the surface and test for visibility, we construct the path incrementally, i.e., the next vertex can only depend on the vertices sampled so far (not the others yet to be sampled). Additionally inserting the domain knowledge that the measurement contribution function is a separable product of terms which at most depend on three consecutive path vertices, we have the following structure in our covariance matrix:

$$\mathbf{S} = \begin{pmatrix} S_{11} & S_{12} & & & \\ S_{21} & S_{22} & S_{23} & & \\ & S_{32} & S_{33} & S_{34} & \\ & \dots & & & \\ & & & S_{k(k-1)} & S_{kk} \end{pmatrix}. \tag{6.15}$$

As a consequence, sampling one additional vertex given a guide path and an already sampled path prefix should depend on two previous vertices. Strictly speaking we should thus form 6D covariance matrices around neighboring guide paths and take the conditional at the current path prefix to compute a 2D covariance matrix and sample a new vertex from that (in the surface case). Since we follow the full path configuration from the start vertex, i.e., the current path prefix and the neighboring guide paths should be very similar, we opted for a simplified variant which just takes 4D covariance matrices around two consecutive vertices and computes the conditional by only removing two dimensions instead of four. This was a big step up from not performing any conditionalization at all (since the conditional is in general much sharper and more localized), but there may be potential in using the full 6D version instead, especially for short transport path segments.

In the setting we use in this work, we will need to compute a 4D covariance matrix for every guide path vertex. We do this with the nearest neighbors of a guide path. That is, we compute 10 distinct entries in the symmetric matrix from 10 neighboring paths (which evaluates to 40 input dimensions, since we take two vertices with two dimensional coordinates from every path). A high ratio of input samples vs. output dimensions makes sure we arrive at useful estimates of covariance matrices. Volume vertices increase both the number of entries in the covariance matrices and the input dimensions.

6.3.6 Extension to dynamic scenes

To explicitly improve temporal stability, we employ a resampling step which corresponds to the resampling step in particle filters for target tracking. When rendering a new frame of an animation, we keep the guide path cache of the previous frame and use it as an initial guided PDF to sample a few paths from. The difference between frames is often small due to temporal coherence, so this step often yields good samples and makes sure that previously discovered effects have a good chance to be discovered in the subsequent frame, too. After this initial seeding step, the old guide paths are discarded and the learning phase continues as usual on the resampled guide paths. In our experiments resampling 40% of guide paths from the previous cache provided good results. The remaining 60% of guide paths are then created without knowledge of the previous frame as only the resampled paths faithfully reflect the updated geometric configuration of the scene. Resampling the previous guide path cache exploits temporal coherence in a similar way as our learning phase exploits spatial coherence between transport paths.

We tested our guided sampling scheme for temporal stability in the Tumbler scene. As expected, we found that guiding without resampling is stable only in the features that can be detected reliably during the learning phase, and can introduce flickering for more difficult effects such as small glints. The resampling approach alleviates most of these issues and combined with DBOR it creates a useable and temporally stable animation except for the very first few frames until most important lighting features have been found and are represented in the guide path cache. The TUMBLER is a hard scene to test temporal stability since any path tracing algorithm can struggle to find all the small and potentially indirect glints which occur on the fine details of the fluid surface in every frame of an animation.

6.4 Implementation details

We implemented our method in a custom spectral rendering system. In our evaluation we found 10 k–50 k guide paths to be sufficient in most cases and therefore we did not optimize our implementation for run time speed or memory.

We store the guide paths and their vertices in two separate linear buffers. A path vertex needs 128 bytes of memory: 112 bytes for sampling information including the eigenvalue decomposition, 12 bytes for the world space position and the remaining 4 bytes are used for flags (e.g., volume, surface, transmit, reflect) and shader information. For each path we store its wavelength, weight, measurement contribution, the individual PDF values p_u and p_g, the index of the first path vertex in the vertex buffer, as well as some additional debugging information. Storing 50 k guide paths therefore requires about 150 MB of memory.

6.4.1 PDF evaluation

To evaluate the guided PDF $p_g(\mathbf{X})$ in equation 6.6 for a given path $\mathbf{X}$, we have to identify all guide paths that could have sampled $\mathbf{X}$. To reduce the computation, we employ a variety of culling approaches: first, we use a BVH to find only guide paths $\mathbf{Y}_j$ with overlapping truncated Gaussians for $\mathbf{x}_1$ because the PDF conditional on other guide paths would evaluate to zero. Each vertex has a corresponding bounding box that encloses its truncated sampling Gaussian. For surfaces, we determine all potential guide paths by intersecting this structure with the camera ray, enumerating all intersecting 2D Gaussians. For volumes, we can perform a simpler point query to find all overlapping 3D Gaussians. Afterwards, we do a quick culling step based on the path configurations (path length and reflect/transmit/volume scattering sequence have to match). Additionally, we cull a guide path as soon as one Gaussian evaluates to zero. Since the Gaussians are designed to cover a small number of neighbors, we can expect the culling to be effective.

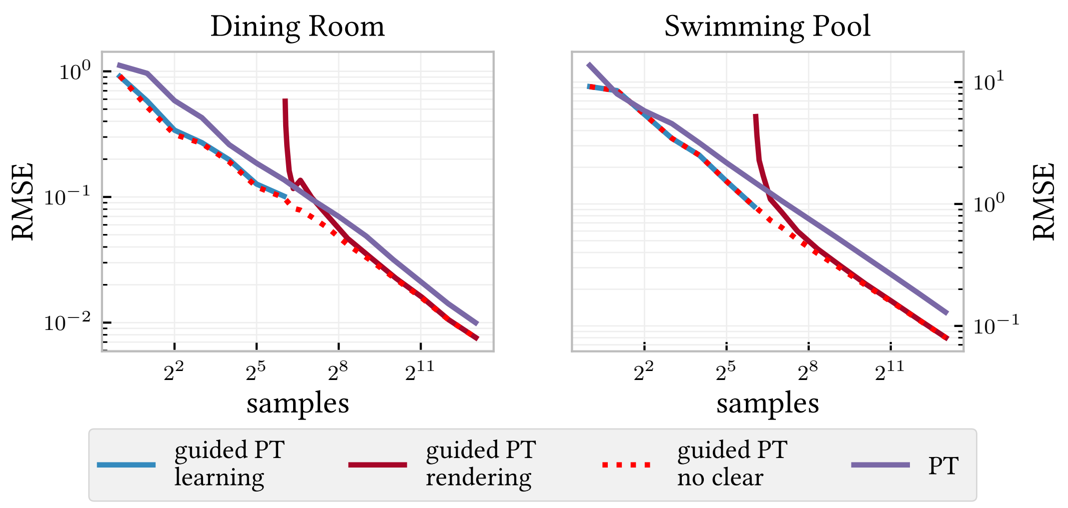


Figure 6.8: Convergence plots for the DiningRoom and Pool scene illustrating that asymptotically our proposed method behaves like regular Monte Carlo, but with lower error than plain path tracing. The orange line depicts the error if the learning phase is explicitly terminated after a while and the frame buffer is cleared. The guided PDF retained in the path cache makes sure the convergence quickly recovers. The dashed line indicates the error if the frame buffer is not cleared after learning stopped at 2^6 samples.

6.4.2 Covariance matrices

To compute the covariance matrices (cf. section 6.3.4) for a guide path, we search for similar guide paths using a high-dimensional nearest neighbor search. Since the number of guide paths is usually low (10 k–100 k) and culling using the same BVH as for the PDF calculation is very effective, we employ a simple brute force approach and compute the pairwise distances between the guide paths according to equation 6.7. We realize that this is very inefficient asymptotically, but it was not a bottleneck in our experiments. If the need for more guide paths arises, an appropriate acceleration structure for the nearest neighbor search will be needed.

The eigenvalue decomposition of the covariance matrices Σ_v can result in negative eigenvalues due to numerical problems. Therefore we clamp them from below to the pixel footprint at the vertex $\mathbf{x}_1$ of a path. This also counteracts sub pixel clumping of samples.

6.4.3 Learning mode/Rendering mode

Our method can be used fully progressively, and similar to Müller et al. [Müller et al. 2017] it does not strictly require a dedicated learning phase. Figure 6.8 illustrates the error behavior

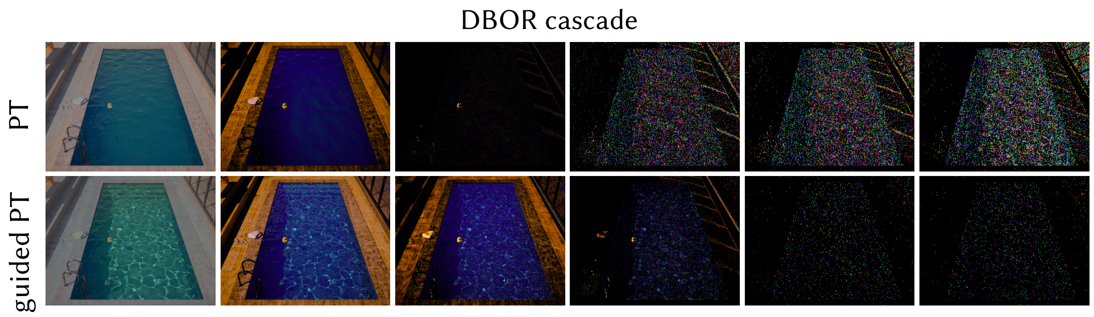


Figure 6.9: The cascade of DBOR buffers for the pool scene after 30min of render time using path tracing (PT) and our guided path tracing (guided PT). In this scene outlier removal basically means ignoring the higher levels of the DBOR cascade. As can be seen, the higher levels of the guided PT cascade contain significantly less energy compared to the PT cascade and the caustics are already resolved in the lower levels of the DBOR cascade. For better visibility only the three first and three last buffers are shown (the remaining buffers are almost black) and the i-th cascade level is scaled by 2^i.

(without DBOR) with sample count if we explicitly split the algorithm into learning and rendering phase versus if we just keep rendering. The main difference is that clearing the frame buffer after learning leads to a fresh start without any potentially accumulated outlier samples. If DBOR is used to remove any outliers from the learning phase, we conclude that clearing the frame buffer after learning is not necessary.

Stopping the learning process

For time-limited renders, it can be an advantage to stop the learning process at some point, since excessive learning can fill the cache with a large number of paths, leading to slow PDF evaluation. It is difficult to define a criterion to decide when the learning phase was sufficiently long: theoretically we can never be sure to have explored all important features in the path space since we have to rely on the unguided sampler to find them (which can take arbitrarily long). We found that allotting 10% of the total run time to the learning phase, and the rest accordingly for the rendering phase, works well in our test scenes. We have encountered cases where the DBOR guide path selection detects that all effects are already sufficiently explored and does not admit any more guide paths into the cache. In this case, the algorithm transparently stops recomputing the guide path cache. There are, however, fail cases to this, for instance in scenes where all transport is equally difficult. This leads to an indefinitely growing cache. We would like to explore more robust criteria in the future.

6.4.4 Density-based outlier rejection

We use density-based outlier rejection [DeCoro et al. 2010; Zirr et al. 2018] during the learning phase to select guide paths (see section 6.3.2) and to remove the remaining outliers after rendering to get clean results.

Our implementation follows the cascaded framebuffer scheme as proposed by [Zirr et al. 2018]. The cascaded framebuffer is a data structure similar to the bilateral grid [Chen et al. 2007] consisting of a set of downsampled framebuffers, one slice for every interval in a logarithmically spaced contribution to the frame buffer. Our implementation reduces the size of the DBOR grid by a factor of two, i.e., four image pixels correspond to one DBOR pixel. We chose this variant of density-based outlier rejection because it does not involve the construction and maintanance of a kd-tree as in [DeCoro et al. 2010] and the outlier removal is strictly a post-process and as such adjusts to progressive rendering.

During rendering, every sample is splatted into the DBOR cascade using a Blackman-Harris filter with 4×4 pixels support. We splat a constant one for each sample into the two levels in the brightness cascade which are closest to the sample contribution. This means the buffer is used to count the number of samples with similar contribution. During evaluation, the grid is simply accessed by reading out a single pixel for each of the two neighboring brightness levels of the sample (slicing), which are interpolated to yield a single value. This value is used to count the number of similar samples.

We use a separate DBOR cascade for learning and rendering depending on whether the framebuffer is cleared after learning. For an illustration of the buffers, see figure 6.9.

DBOR for guide path selection

We use DBOR during the learning phase to distinguish between high-contribution samples which are frequent enough to not cause high variance, and those which are important as guide paths. DBOR provides information about the frequency of paths with a certain contribution and we decide based on this frequency whether this contribution is already sufficiently well sampled or needs to be included as a new guide path with a user adjustable frequency threshold. This threshold has an impact on learning: lowering the threshold increases the chances to add irrelevant samples to the guiding cache which the unguided sampler would have been able to handle. This can decrease the efficiency of the guided sampler. On the other hand, increasing the threshold reduces the learning speed, as only very rare samples are added to the cache, and consequently more unguided samples are needed.

Remaining high variance samples

Stopping the learning process means that there can be regions in the path space that have not been explored yet. If these are sampled during the rendering phase by the unguided sampler, its high variance samples possibly contribute fully to the image as bright outliers. Resolving this is possible by an extended learning phase or by outlier removal. In figure 6.10 we show that removing the remaining outliers discards significantly less energy than using unguided sampling only and removing all outliers. We perform the outlier removal with DBOR, which we already use for path classification during the learning phase.

6.4.5 Parameters

Our implementation has parameters that we want to discuss in the following. First, we usually choose the unguided mixing weight from equation 6.3 as $u = 0.5$. We ran experiments adjusting it such that the observed variance in the guided and unguided contributions to the image are of equal magnitude. This led to results very close to 0.5, but may be worth reconsidering in the future.

Second, we need to set the number of samples added to the cache in every iteration. We found that using a budget of 0.1–1% of the total number of generated samples per iteration works well in our tests. Since DBOR can be unreliable in the beginning of the learning phase, we use a conservative maximum of 0.2%. We use one heap per thread to efficiently manage the list of highest-ranked paths.

Progressive Gaussian shrinking

As mentioned before, we use a progressive radius shrinking scheme based on a ray differential footprint for the size of the Gaussians at the first vertex. We start by a screen space radius of 50 pixels and progressively shrink it during the learning phase depending on the scene. Starting with a large radius is important for discovery in the beginning of the learning phase and allows to share guide paths between pixels. The radius at the end of shrinking has two important impacts. First, a smaller radius results in less overlapping Gaussians and therefore more effective culling and faster PDF evaluation. Second, a smaller radius results in larger cache sizes because guide paths can only be shared between fewer pixels which increases PDF evaluation time. Choosing an optimal minimal radius automatically is challenging and remains future work. Right now, it is a user parameter that is chosen between 1 (which we use for the Pool and Tumbler scenes) and 10 pixels (which we use for all other scenes in this work).

6.5 Results

Our results have been rendered on a AMD Ryzen 7 1800X with 64 GB of main memory. In all our experiments with the guided sampler, half the samples are guided ($u = 0.5$). The maximum path length is 32 path vertices for all images.

6.5.1 Evaluation of guided sampling

We evaluated our guided sampling in a variety of scenes with different challenging lighting settings shown in figure 6.10. We performed equal-time comparisons between path tracing with next event estimation (PT) and guided path tracing (guided PT). The details about learn and render time, number of guide paths and error metrics compared to a path traced reference are also shown in figure 6.10.

DiningRoom

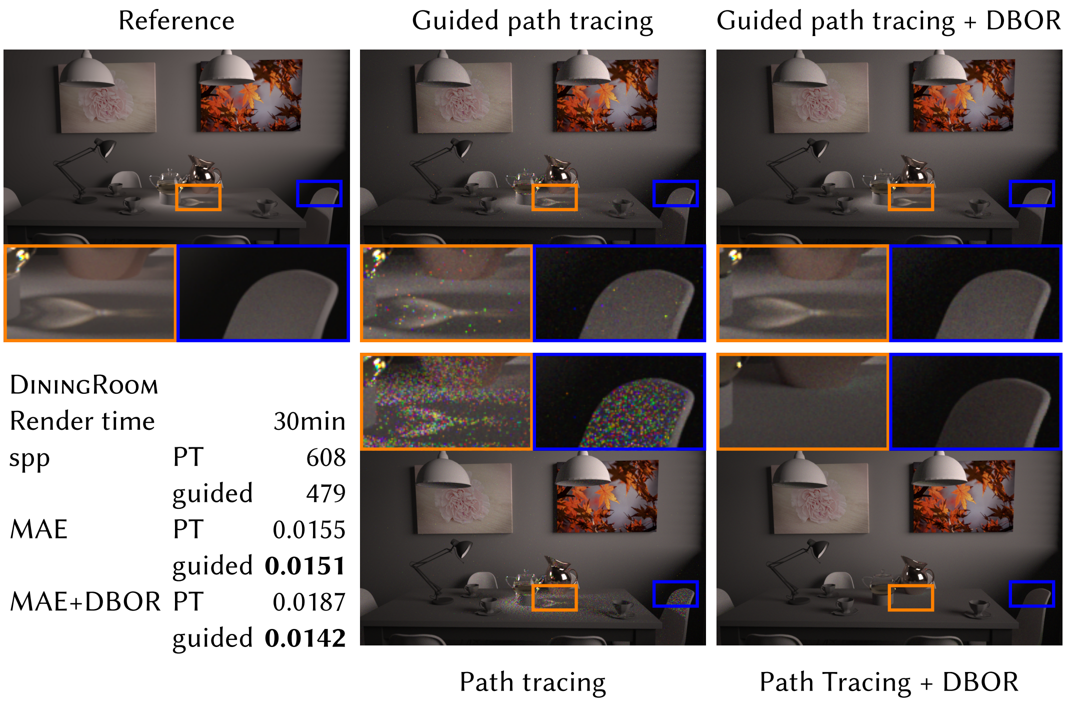
The first scene in figure 6.10 shows an indoor scene illuminated by a large light outside and a small bright spotlight inside on the table. PT handles the outside illumination that dominates the overall lighting in this scenes very well. However, it struggles to resolve the caustic caused by the lamp and teapot on the table. Our algorithm robustly selects the difficult regions and is able to sample the reflected caustic effectively. The insets show the results of our guided PT with outlier removal. For PT there was no outlier removal otherwise the caustic would be missing completely.

Pool

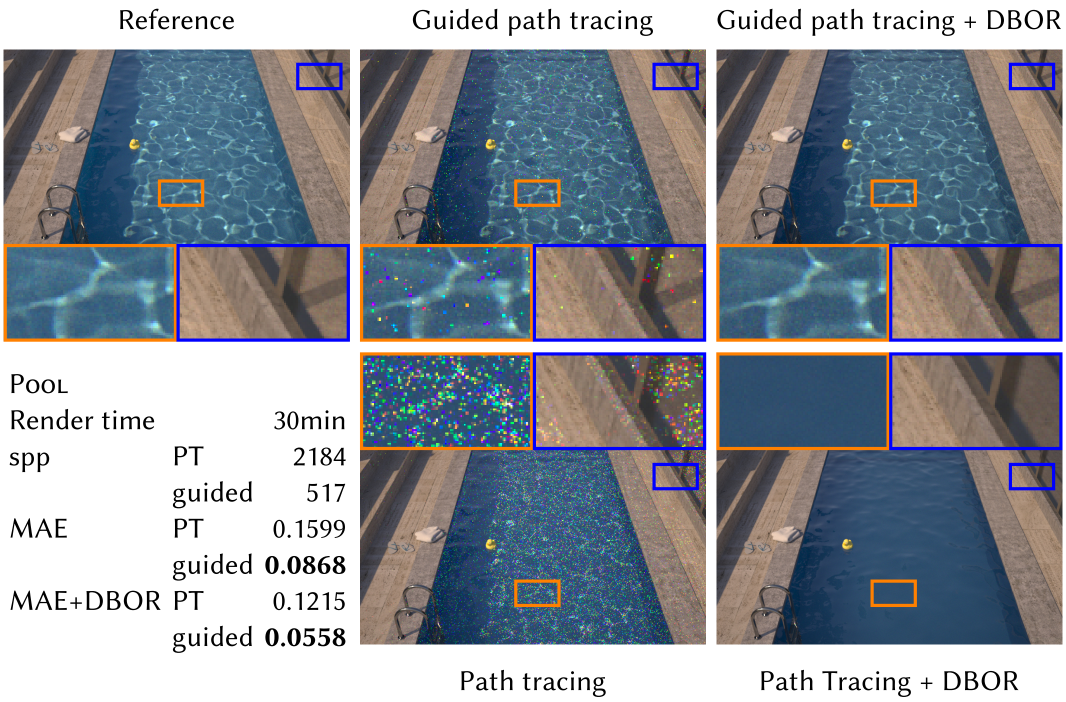
A scene where the difficult caustic affects a much bigger portion of the image is the second scene in figure 6.10. Here a swimming pool is illuminated by environment lighting containing a bright sun which causes intricate caustics on the bottom of the pool. Our guided PT handles this scene equally well and divides the illumination of the sun, which is handled by the guided sampler, from the more uniform rest of the environment illumination that is handled by the sampler. In all comparisons in figure 6.10 we used outlier removal with the same threshold for both PT and guided PT.

Tumbler

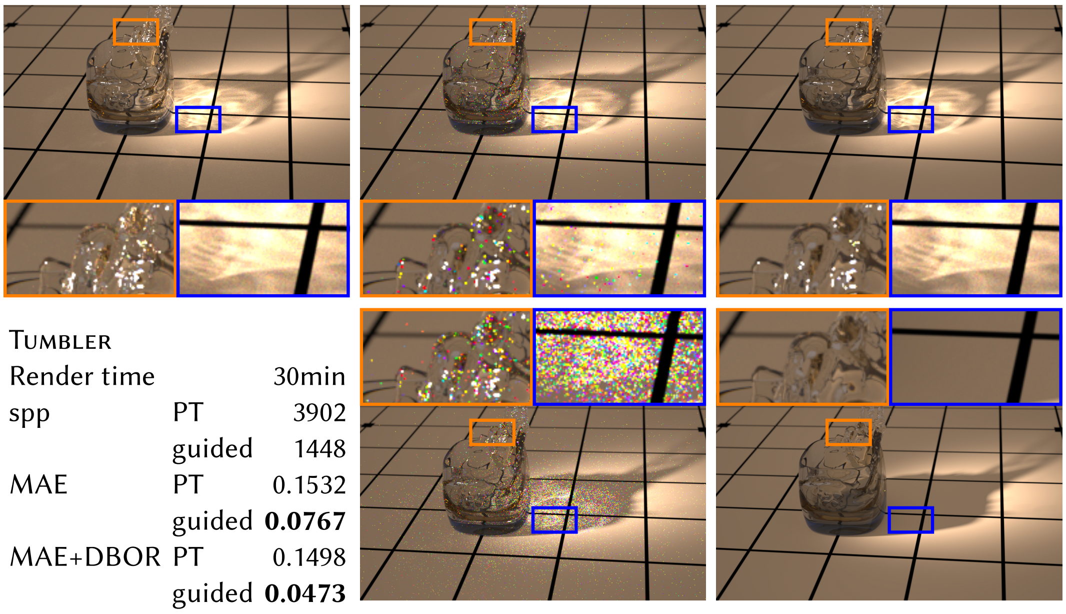
The third scene in figure 6.10 contains a complicated caustic and many small glints on the water surface and the surface of the tumbler. This scene is challenging due to its many direct and indirect highlights. Again the illumination is robustly divided into a high frequency component for the guided sampler and a low frequency component for the



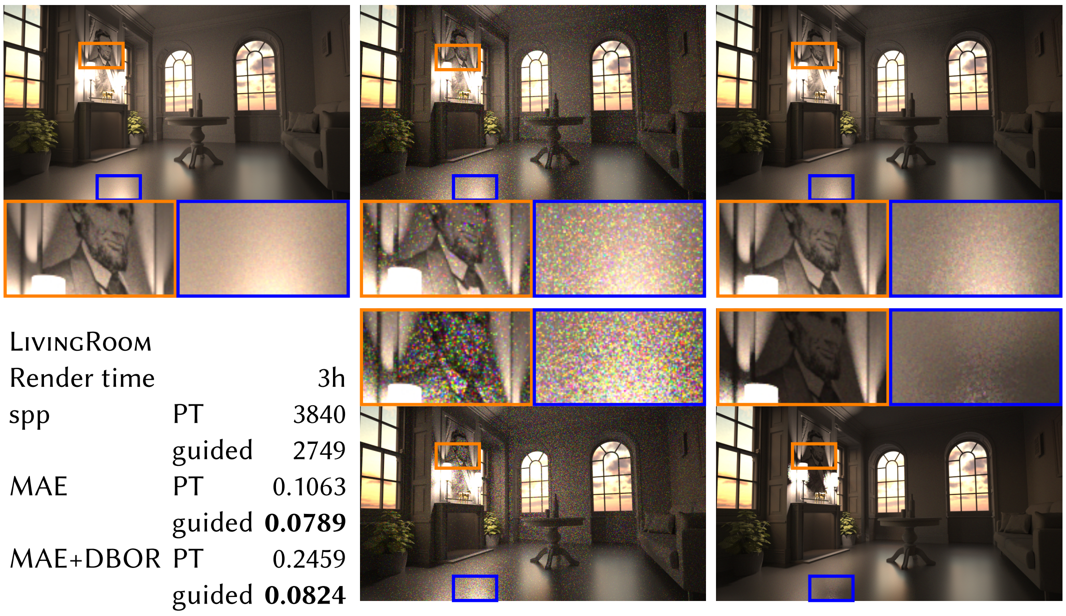
DiningRoom		
Render time		30min
spp	PT	608
	guided	479
MAE	PT	0.0155
	guided	**0.0151**
MAE+DBOR	PT	0.0187
	guided	**0.0142**



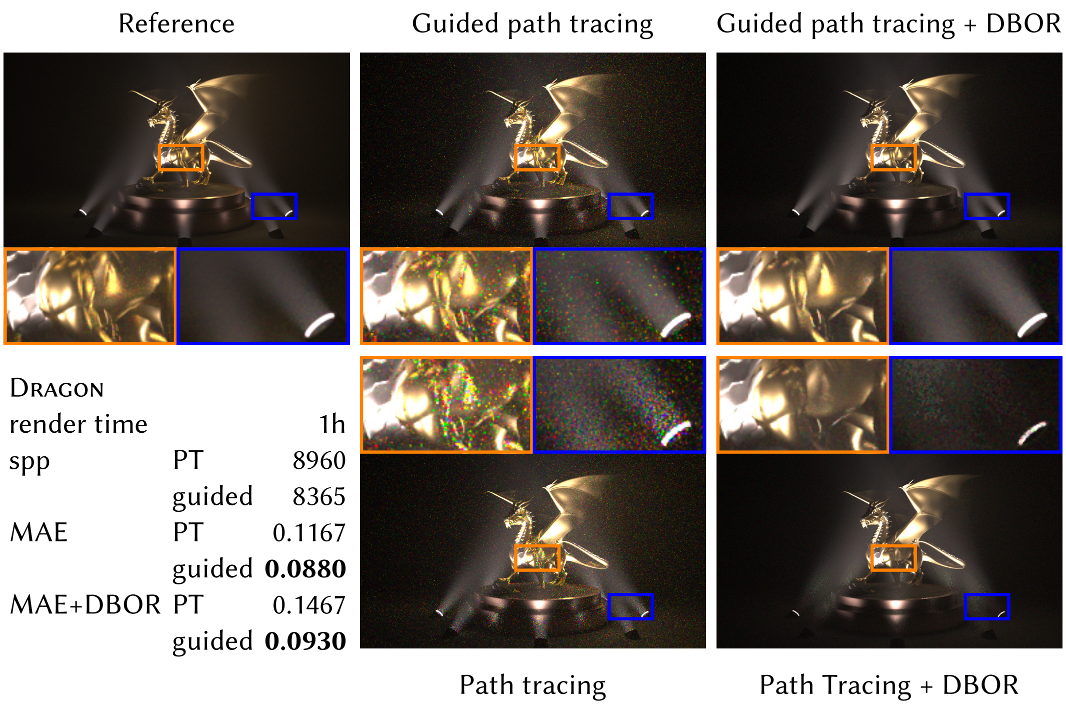
Pool		
Render time		30min
spp	PT	2184
	guided	517
MAE	PT	0.1599
	guided	**0.0868**
MAE+DBOR	PT	0.1215
	guided	**0.0558**



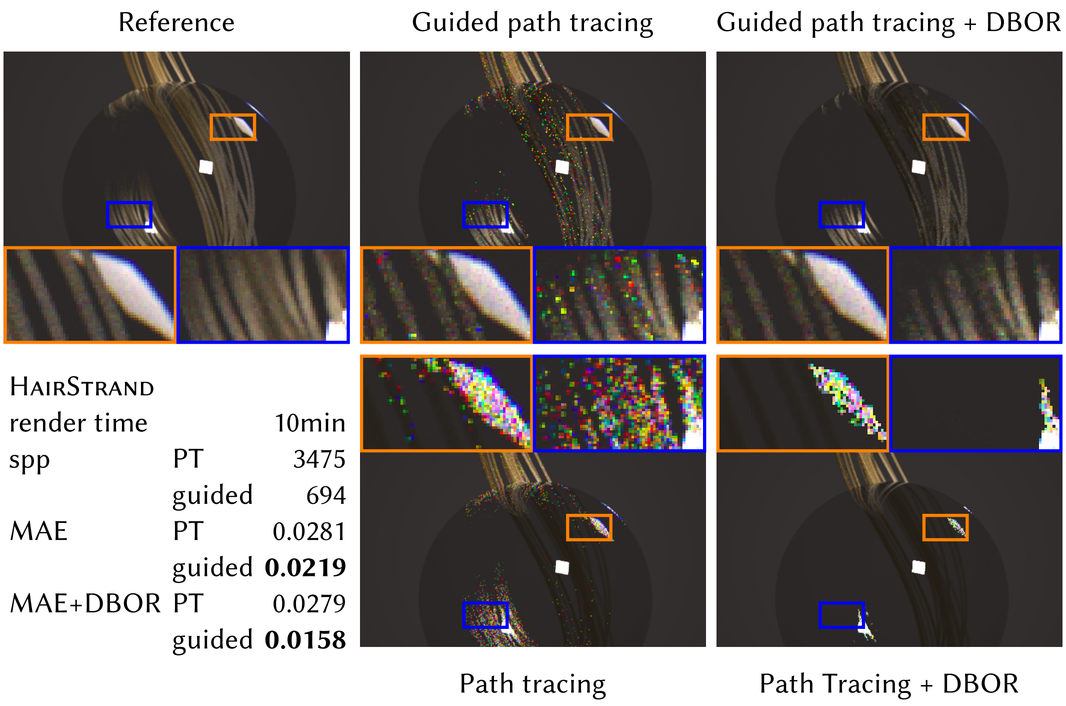
TUMBLER		
Render time		30min
spp	PT	3902
	guided	1448
MAE	PT	0.1532
	guided	**0.0767**
MAE+DBOR	PT	0.1498
	guided	**0.0473**



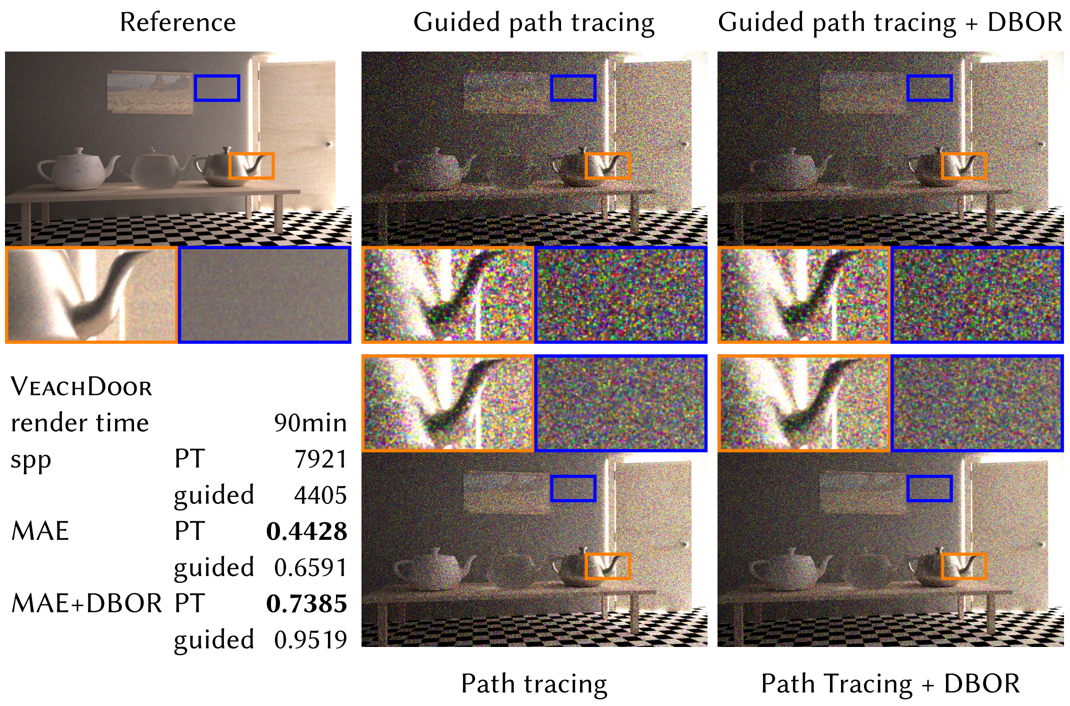
LIVINGROOM		
Render time		3h
spp	PT	3840
	guided	2749
MAE	PT	0.1063
	guided	**0.0789**
MAE+DBOR	PT	0.2459
	guided	**0.0824**



Dragon		
render time		1h
spp	PT	8960
	guided	8365
MAE	PT	0.1167
	guided	**0.0880**
MAE+DBOR	PT	0.1467
	guided	**0.0930**



HairStrand		
render time		10min
spp	PT	3475
	guided	694
MAE	PT	0.0281
	guided	**0.0219**
MAE+DBOR	PT	0.0279
	guided	**0.0158**



VEACHDOOR		
render time		90min
spp	PT	7921
	guided	4405
MAE	PT	**0.4428**
	guided	0.6591
MAE+DBOR	PT	**0.7385**
	guided	0.9519

Figure 6.10: Equal-time comparisons of path tracing and our guided path tracing. The insets show results with and without outlier removal. PT misses lighting effects it can not render efficiently (e.g., caustics). In all scenes guided path tracing robustly identifies the part of light transport that can not be handled by path tracing efficiently and improves upon this using our guided sampling for these parts. As a side effect, we obtain an interesting separation into a smooth/low contrast image (unguided contribution) and a sharp/high contrast image (guided contribution).

unguided sampler. After outlier removal no glints (highlights) remain in the path traced images whereas our method resolves these small bright features much better.

LivingRoom

The fourth scene in figure 6.10 shows a living room which is illuminated from the outside with environment lighting and two lamps on the inside. The lamps are spherical emitters surrounded by a dielectric cylinder that prevents next event estimation and causes a high variance for PT. Guided PT focuses on this difficult illumination which results in less noise especially in the region of the picture above the lamps. After outlier removal, PT misses much more energy in this region compared to guided PT.

Dragon

The fifth scene figure 6.10 shows a glossy dragon in slightly forward-scattering fog (mean cosine is 0.5) which is illuminated by several spotlights. The surrounding participating medium complicates the light transport simulation significantly.

The bottom inset shows the volume directly in front of one of the spot lights. Usually a specialised technique, such as equi-angular sampling [Kulla and Fajardo 2011], is required to sample such regions efficiently. Our guided sampling improves the rendering quality in these regions significantly and automatically.

HairStrand

The sixt scene in figure 6.10 shows a hair strand that is partly inside a glass sphere. This scene is difficult for many rendering algorithms: BDPT fails at sampling important paths as the glass sphere prevents next event estimation to the light source as well as to the camera. Photon mapping struggles in this scene because photon density estimation on hair is difficult: both the approximation of the area as a disk (often used for surfaces) and the volume as a sphere will result in visible errors. Our guiding approach works on hair fibers out of the box, and improves the image quality compared to the unguided version of PT significantly.

VeachDoor

The last scene in figure 6.10 is an example for a scene that is unsuited for our guided sampling in its current form. In this scene all paths are roughly equally hard to sample and there is no isolated lighting effect that guiding could focus on. This means that guiding will try to compute the whole multibounce diffuse light transport which it struggles with due to the curse of dimensionality. In this scene our guided PT therefore degenerates to PT, however, with an increased overhead which results in only half of the samples per pixel per time compared to regular PT.

We can observe that the unguided sampler computes more of the diffusive multiple scattering illumination. The guided sampler focuses on bright and sharp features such as the indirect glossy reflection of the directly illuminated part of the dragon's wing and the volumetric illumination from the spot lights. This multi-dimensional feature is potentially very hard to grasp with 2D guide records. Note that we cannot show a direct comparison to previous guiding techniques as these do not support participating media.

6.5.2 Comparison to related work

We compare to the previous work of Vorba et al. [Vorba et al. 2014] in figure 6.11 where we render water droplets on a leaf. We used the author's source code available in the rendering framework Mitsuba. This comparison has to be taken with a grain of salt because of the different rendering environment. We perform roughly an equal time comparison (4–5 min) by normalizing the performance of the two different rendering systems: we render simple path tracing with next event estimation at 1024 samples per pixel for both, and give the guiding algorithms in the corresponding frameworks the same time. What can be observed from these images is that the 2D caches struggle with separating the illumination from the constant environment map from the diffuse and very small light source present in this scene. Our guided sampling improves this situation: first, keeping complete transport paths can easily distinguish between well-separated light sources. Second, we can sample along guide paths maintaining the path configuration (reflect or transmit). Third, we detect that samples landing on a constant environment map have very little variance and can be handled by the path tracer. Thus the result is smoother with about half the number of samples per pixel while using about three orders of magnitude fewer guide records.

In figure 6.12 we compare to the work of Müller et al. [Müller et al. 2017] in the Pool scene. We used the author's source code available in the rendering framework Mitsuba. Here again, this comparison has to be taken with a grain of salt because of the different rendering environments. Considering the additional noise introduced by our spectral renderer the qualitative improvement over path tracing with next event estimation is similar for both methods.

In figure 6.13 we show a closeup of the DiningRoom scene focusing on the caustic and reflected caustic. Our implementation can be used with any path sampler for which we can compute the PDF in closed form. While path tracing is a reasonable choice, we also demonstrate guiding in combination with bidirectional path tracing (BDPT) [E. Lafortune and Y. Willems 1993; Veach and Guibas 1994] as the unguided sampler. As expected, BDPT

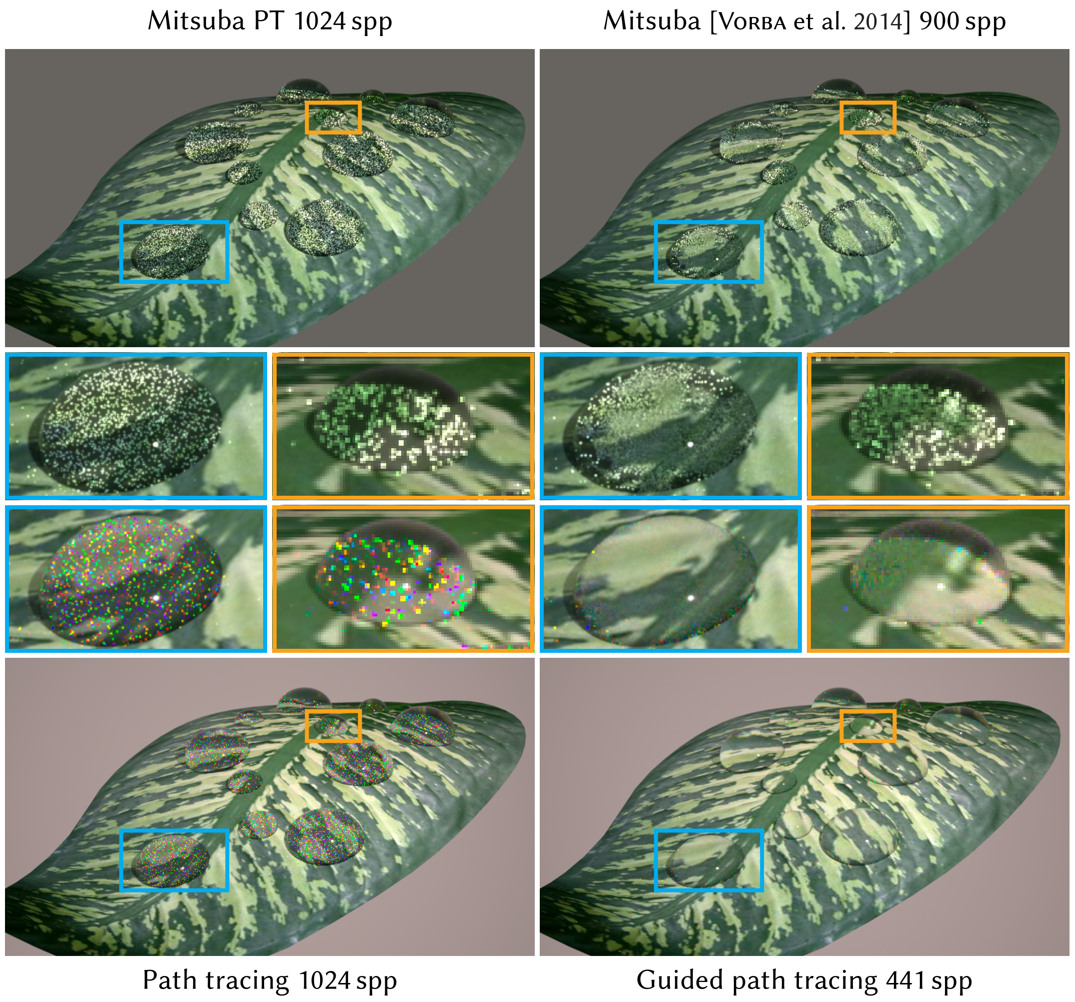

Mitsuba PT 1024 spp
Mitsuba [Vorba et al. 2014] 900 spp
Path tracing 1024 spp
Guided path tracing 441 spp

Figure 6.11: Roughly equal time comparison to 2D marginalized guide records [Vorba et al. 2014]. The top row has been rendered in Mitsuba, and the bottom row in our renderer (on different machines with different operating systems, one RGB and one spectral). We normalized run time to 1024 spp for next event estimation (left). The scene is lit by a key light to the right and a constant environment map. Here, 2D records should work just fine because the problem is essentially 2D. Since our method focuses its efforts on difficult regions (the water droplets), it performs much better using only 950 guide paths (bottom right) even compared to Vorba et al.'s method (top right) using roughly 700k guide records, which has to learn the whole scene.

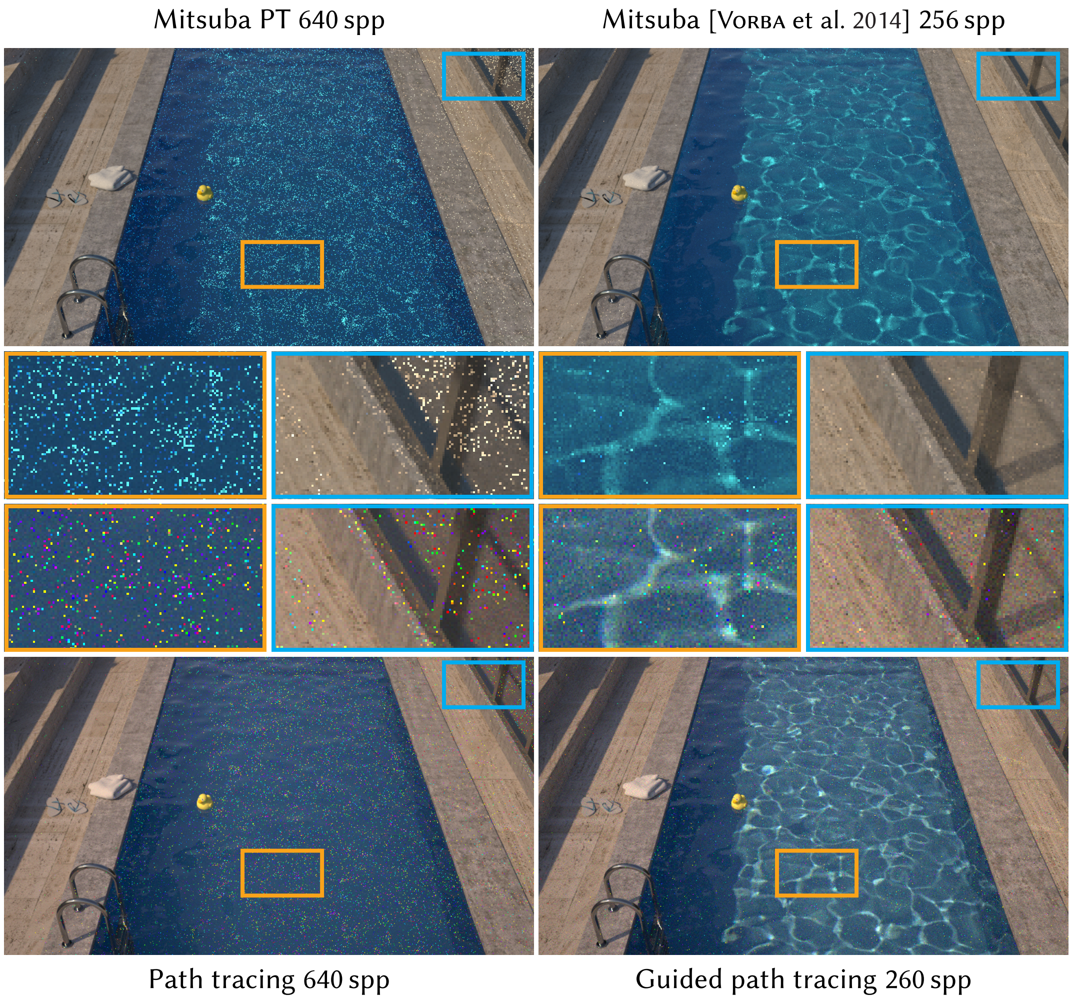

Mitsuba PT 640 spp
Mitsuba [Vorba et al. 2014] 256 spp
Path tracing 640 spp
Guided path tracing 260 spp

Figure 6.12: Roughly equal time comparison to 2D marginalized guide records [Müller et al. 2017]. The top row has been rendered in Mitsuba, and the bottom row in our renderer (on the same machine, one RGB and one spectral). We normalized run time to 640 spp/5 min for next event estimation (left). In this case the difficult parts of transport are spread all over the scene, so our method cannot show its full strength by focusing efforts to small areas in screen space. Factoring in the color noise from spectral rendering, it yields comparable benefits over path tracing.

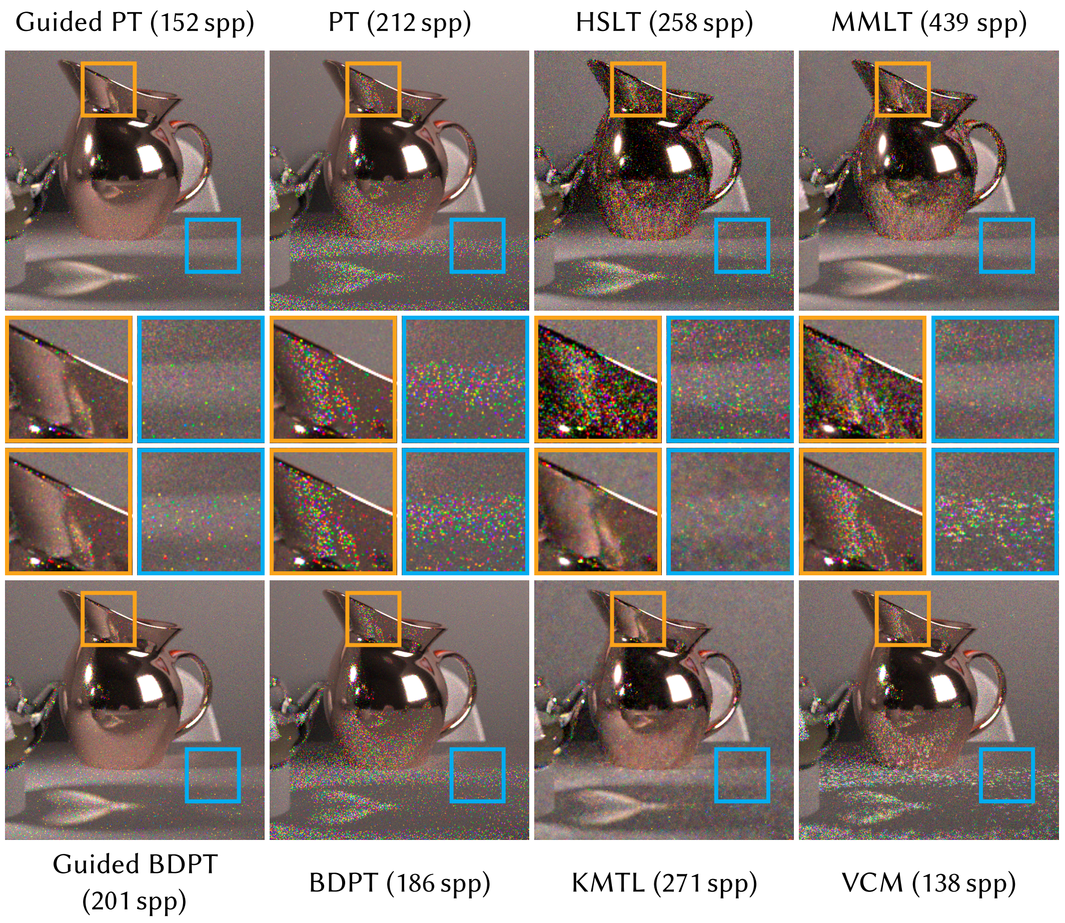


Figure 6.13: An equal-time comparison for a closeup view of the DiningRoom. The leftmost column shows guided sampling in conjunction with PT (top) and BDPT (bottom). BDPT and light tracing in general perform very poorly in this scene because the bright area light outside the scene attracts most samples and leaves very few for the prominent spot light which casts the caustic. Vertex connection and merging (VCM) [Georgiev et al. 2012a] suffers from the same problem. The selection phase of our guided sampling is able to focus on the caustic and especially the reflected caustic much better. Markov chain-based methods help here, we show three versions: Half vector space light transport (HSLT) [A. Kaplanyan et al. 2014] with a relatively large step size results in evenly spaced samples but relatively high noise level. Kelemen Metropolis (KMLT) [Kelemen et al. 2002] uses a small step size leading to blotchy appearance, especially in the blue inset or the back wall. Multiplexed Metropolis (MMLT) [Hachisuka et al. 2014] looks more even since it uses larger steps and takes advantage of light tracing, but shows scratch-like structures in the orange inset. We derive step sizes from neighboring samples instead, leading to smoother results.

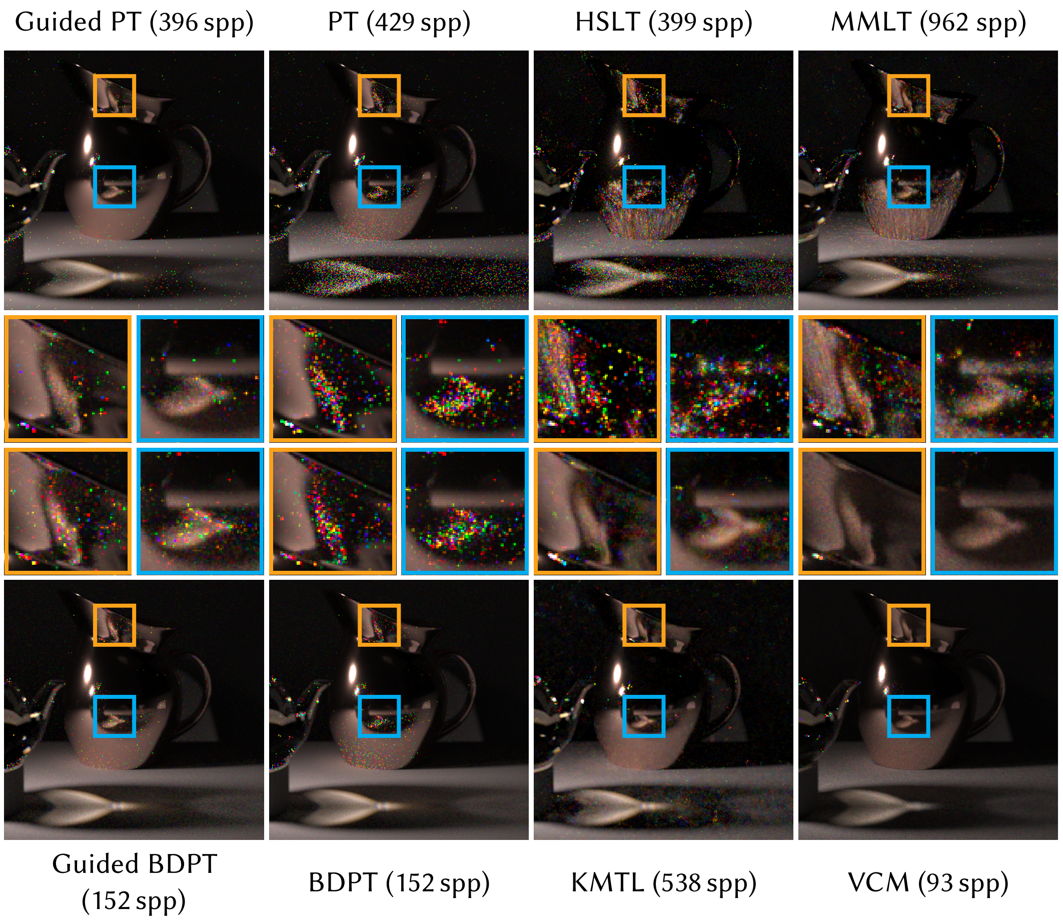


Figure 6.14: The same comparison as in figure 6.13 but with the exterior light removed, only the spot light remains. This makes light tracing a lot more efficient as can be seen for BDPT and VCM. Note that the Markov-chain based methods (HSLT, KMLT, MMLT) do not perform much differently.

fails to resolve the reflected caustics. In addition, the relatively large and bright exterior light source attracts most of the samples for the light tracing pass, making this technique inefficient. The guided sampling versions both used a learning phase of 30s out of 5min total render time (in both cases about 17 iterations, resulting in approx. 5700 guide paths). We also compare to Markov chain Monte Carlo methods which can be used to render such (reflected) caustics efficiently. However, they introduce correlated sampling artifacts throughout the image, not only in the difficult regions. Note that both HSLT [A. Kaplanyan et al. 2014; Hanika et al. 2015] and KMLT [Kelemen et al. 2002] in this figure use unidirectional path tracing only. In figure 6.14 we show the same experiment but without the area light (spotlight only). Methods using light tracing work much better in this setting because the area light from the scene does not take up many unimportant samples anymore.

6.5.3 Path length

We visualized light transport paths appearing in a volume caustic from a specular sphere in an unbounded homogeneous medium in figure 6.15. In this case, guided sampling saves a lot of render time because it does not generate overly long scattering paths in uninteresting regions of the path space, as opposed to unguided sampling. This shows how the guide cache with full paths can make Russian roulette for path termination unnecessary.

6.6 Discussion

Discarded ideas

In the course of this project, we explored various, as it turned out, wrong tracks, but we still want to share these. Replacing Gaussians during learning by constant disk kernels or cubic b-splines strongly limits the ability of local exploration due to the short or non-existent tails of these kernels.

To speed up the search for neighboring guide paths, we tried to use a repeated application of a lower dimensional 3D nearest neighbor search using a BVH, once per path vertex instead of a monolithic search for the whole path. In the simplistic example in figure 6.4 this might work, but may leave gaps between the guide paths when these are overlaid by many more undirected diffuse paths. 3D neighbor search will underestimate the distances, since many vertices seem close together but belong to paths that are far apart in other bounces. This will result in more residual outliers caused by the unguided contributions.

We experimented with photon relaxation-inspired guide path vertex moving [Spencer and M. W. Jones 2013]. However, doing so as a padded replication of 2D/3D operations breaks the correlation between bounces and the resulting paths do not transport much energy any more. We also experimented with high-dimensional move steps (as often found in sequential Monte Carlo) and resampled guide paths by using a Metropolis-Hastings update step similar to PMCMC [Andrieu et al. 2010]. Unfortunately this led to clumping and was working against the stratification in our selection stage. These experiences suggest that expectation maximisation-based fitting of Gaussians [Jakob et al. 2011] also probably does not work well in this context.

Learning

Our experiments showed that selecting too many new guide paths in one iteration hinders the exploration: the sampler requires a few iterations to find the regions of lowest probability density between existing Gaussians for good stratification. Moreover, irrelevant samples should never be chosen as guide paths and should be handled by the unguided sampler. In interesting regions which have already been discovered, adding too many overlapping Gaussians can lead to poor performance when evaluating the PDF. All our attempts to "repair" such scenarios with a resampling step failed. Randomly removing samples from the cache often resulted in oscillation instead of convergence.

Outlier removal

Any kind of adaptive sampling (such as guiding) takes away probability mass from areas that are deemed unimportant, to favour already discovered interesting areas. This is dangerous as it can increase variance in important but undiscovered areas. To balance this effect, we focus on the surroundings of samples which cause high variance and thus high mean squared error (MSE). Consequently, the areas we take probability mass away from have only minor effect on MSE. We do not introduce worse outliers since we combine with the unguided sampler via MIS. However, in the worst case this leads to fewer samples per pixel in the same time for regions marked as unimportant, increasing variance there. In such cases simply removing outlier samples often results in lower MSE (see figure 6.10). Note that this effect is found in all adaptive sampling schemes, also in previous low dimensional marginal guiding systems. This is why [Vorba et al. 2014] and similar methods always apply MIS with BSDF sampling to counteract the effect. For final renders, we always use outlier removal. Since it is run as a post-process [Zirr et al. 2018], we can always reconstruct the unbiased results which are equal to the filtered result if there are no outliers. Figure 6.10 shows results with and without outlier removal for all our test scenes, except the simple scenes in figures 6.6, 6.11 and 6.12 which do not use any outlier removal.

Animations

Though not part of our original scope, we implemented a straightforward adaptation of our method with explicit inter-frame reuse of guide paths for dynamic scenes as discribed in section 6.3.6. We evaluated temporal stability by rendering an animation of the Tumbler scene where the small lightsource moves around the glas. We found that reusing caches between frames substantially improves temporal stability and learning speed. We leave it for future work to thoroughly analyse this approach in a variety of scenes and optimize it for reduced flickering, fast learning, and balanced exploration of path space.

Complete paths vs. 2D marginals

We explored the possibilities of caching complete transport paths with all dimensions and their correlations (incoming lighting, BSDF, free flight distance in volumes). In cases where no such correlation exists (e.g., a diffuse Cornell box with no complex visibility which is correlated with path length), our method creates unnecessary overhead. Instead, low-dimensional caches storing 2D marginals [Vorba et al. 2014] may perform better. Also we assume that the unguided sampler works well in a significant portion of the sampling domain. In the VeachDoor scene this is not true and all transport would profit from guiding.

However, 2D marginal cache records do not consider BSDF or volumes. We are interested in storing reflectance fields, not light fields. Thus, in general the transport operator is 6D up to 9D. [Guo et al. 2018] recently showed that guiding with spatial data structures can be challenging with this number of dimensions already.

We argue that storing the full paths is a well-suited data structure since it allows us to clearly distinguish between types of scattering events (reflect, transmit, volume, specular), it encodes potentially long specular chains, and it sidesteps the clustering problem of multi-modal GMM: The path space helps discern different types of incident illumination, so we can use simpler uni-modal Gaussians and thus never run out of lobes as e.g., [Vorba et al. 2014].

We showed we can afford to store important paths and we hope to inspire future work making even better use of the data (e.g., by jumping between paths using them as low-dimensional marginal distributions to improve the performance in the VeachDoor scene).

Curse of dimensionality

Usually, the remaining hard-to-sample part consists of highly directed transport and is effectively low dimensional. Areas with many undirected bounces can cause problems for our guiding, as Gaussian kernels need to become large to close gaps in the sampling domain.

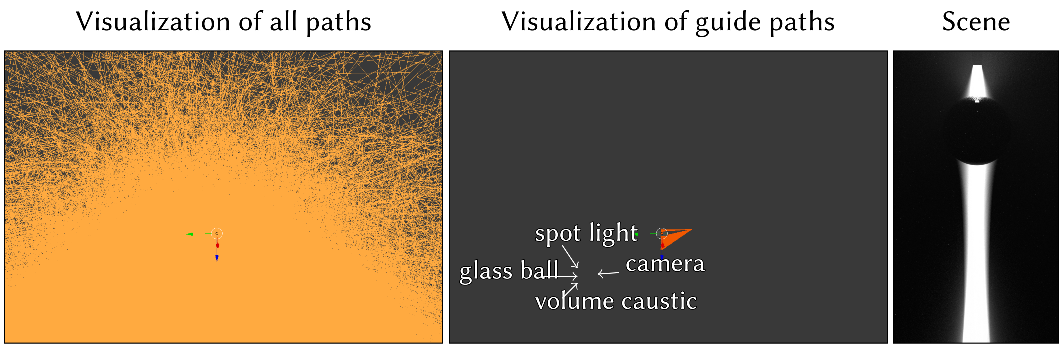


Figure 6.15: Path visualisation for a volume caustic scene where a specular sphere casts a caustic from a spot light (right). The left two pictures show screen grabs from a 3D visualisation of traced paths, at same scale. Guiding by complete paths efficiently selects only the relevant paths (middle) out of the complete set of paths a path tracer (left) would trace. Most of these paths will carry very little throughput and could be culled by using Russian roulette, because energy is transported here by paths with five vertices or less. However, without adjoint information Russian roulette can increase variance substantially [Vorba and Křivánek 2016].

Fortunately, the unguided samplers typically perform well in these regions. However, we did encounter situations with very large Gaussians (and thus slow PDF evaluation), or gaps which are filled by high-variance unguided paths (as in any dense, highly scattering medium or in the VeachDoor scene).

Deriving a PDF from many neighbors

One interesting aspect of our guiding is that, in contrast to MC or MCMC, we can make data-driven decisions about the sampling spread around a sample. We found the conditional distribution derived from a 6D covariance matrix very effective in generating accurate samples from very few guide paths. In theory, the transport operator should depend on three path vertices, i.e., the maximum dimension of the covariance matrix should be 9×9. Our implementation assumes that incoming rays have about the same direction since they are coupled to the guide path, but extending this may lead to even better results. Ideally such an approach would incorporate statistics about half vectors, which may avoid the need to sample the BSDF at all. Our fallback BSDF sampling uses independent sampling. At least it would potentially be better to use smaller steps in primary sample space by first deriving the random numbers using an inverse mapping [Otsu et al. 2017; Bitterli et al. 2017; Pantaleoni 2017].

6.7 Conclusion and future work

We introduced a new guided sampling technique for light transport simulation which employs complete transport paths instead of 2D marginals. This captures diverse kinds of lighting effects and their correlations, including the BSDF for glossy interactions and free flight sampling for volume scattering as well as path configuration (reflect vs. transmit and path length instead of Russian roulette). This technique is selective, in that it is used only in the regions of the path space where an existing Monte Carlo sampler is inefficient. We automatically detect this sub domain.

This separation into highly directed/high contrast contribution and diffusive/low contrast parts could be exploited in future work to derive specialised variance reduction techniques. For instance the low contrast part could work well with aggressive denoising or gradient domain path tracing [Kettunen et al. 2015].

The method conceptually differs from Markov chain-based methods: it derives sampling of a new path from many input paths instead of just one current state. This enables us to accurately estimate the spread of the distribution. Furthermore, using complete paths as guide records allows us to use simple unimodal Gaussian distributions instead of complicated clustering, while using significantly fewer guide records than previous work.

We would like to point out that a newly proposed guide path does not strictly require a well-defined PDF. Our technique would re-evaluate a PDF based on the current state of the cache. In other words, the guide paths need not be generated by an unbiased Monte Carlo sampler of the path integral. Instead, the cache could be filled by Markov chains violating the detailed balance condition or by hand-chosen paths. The latter could be useful to explicitly steer importance to a key light that is currently being look developed. Conversely, the guide path cache can be used to visualise the hardest transport paths in a scene to reveal potential modelling issues.

This new approach comes with new challenges. We propose a first set of techniques to approach these. In particular, we propose an iterative path selection strategy akin to particle filters which finds difficult regions in the sampling domain and keeps rare trajectories in a cache. From this cache, we derive a continuous, high-dimensional PDF similar to a Gaussian mixture model. We also present a sampling technique which creates full path samples from a set of neighboring guide paths. We expect each of these steps to improve in the future, such as faster PDF evaluation, or more accurate sampling from fewer guide paths.

7 Conclusion

In this thesis, we introduced several local and global importance sampling techniques for improving the efficiency of Monte Carlo-based rendering algorithms.

Chapter 4 introduced a data-driven global importance sampling approach for distributing VPLs in many-light rendering. We used the generated global information to enhance the information content of each VPL, which increases their efficiency in rendering scenes with highly glossy materials. The resulting Rich-VPLs have a more sophisticated emission profile better suited for glossy materials. In [Tokuyoshi 2015], a similar approach has since been used for real-time rendering of glossy materials with VPLs by encoding the emission profile as a small set of spherical Gaussians similar to our memory reduction scheme using von Mises-Fisher distributions.

Many-light methods rely on RGB light transport for efficiency compared to path tracing, which can also efficiently use spectral rendering. Since we use both RGB and spectral rendering in this thesis, we discussed their differences and the non-trivial intricacies of RGB rendering for global illumination in chapter 3.

In chapter 5, we introduced a method to render heterogeneous emissive volumes efficiently. The key idea of this method was to integrate emission for whole path segments and not only path vertices, which is much more efficient for thin media such as flames. However, to efficiently use this line integration in a Monte Carlo-based rendering algorithm, we need to combine it with the other sampling strategies in a meaningful way. To achieve this, we introduced a novel next event estimation technique that combines positional and directional sampling in a two-step approach. The proposed method robustly renders the whole range of heterogeneous emissive volumes, from thin flames to explosions with dense smoke. This method was also adopted by the VFX company Framestore in its production rendering software [Fascione et al. 2017].

Finally, the data-driven global importance sampling technique introduced in chapter 6 was able to improve a given Monte Carlo-based path sampling algorithm by identifying and successively creating paths, which the sampling algorithm struggles to generate. This method is closely related to other path guiding approaches, which overall receive increasing attention from researchers and practitioners alike [Vorba et al. 2019]. Path guiding based on marginalized local information was recently extended to handle volumetric light transport [Herholz et al. 2019]. Sequential Monte Carlo approaches inspired the learning scheme of our technique, and recently, an increasing number of machine learning approaches are applied to process and optimally use the collected guiding data [Vévoda et al. 2018; Müller et al. 2019]. Our technique is also selective in the sense that it tries only to compute the part of light transport with which a given base sampler struggles. This selectiveness reduces the overhead of our method by focusing its effort only to where it is needed. This idea was recently applied to Markov chain Monte Carlo [Bitterli and Jarosz 2019]. In this approach, only the parts of light transport where a given Monte Carlo sampler struggles, are computed using the Markov chain.

All methods introduced in this thesis improve the comparatively simple light transport algorithms path tracing and the many-light method. Of course, more sophisticated algorithms, such as MCMC, VCM, or gradient-domain methods, are also able to solve many of the problems discussed in this thesis. However, despite the ongoing effort to make these algorithms more practical [Šik and Křivánek 2019; Bitterli et al. 2017; Van de Woestijne et al. 2017; Manzi et al. 2016], they have not yet found broad adoption in practice due to their problems with temporal stability or their significant computational overhead.

It is uncertain if these algorithms can ever replace the relatively simple and comparatively low-maintenance path tracing algorithm, which can often be easily improved with custom modifications for particular and frequently one-time applications in practice

Also, the recent resurgence of Monte Carlo-based real-time rendering, fueled by long-awaited ray tracing hardware and advancements in image denoising [Schied et al. 2017; Chaitanya et al. 2017], will be limited to path tracing for the foreseeable future. In real-time settings, only a handful of paths can be traced per frame. Therefore, we expect an increased demand for low overhead (memory and compute) global and local importance sampling algorithms to ensure that these few paths are sampled as optimally as possible.

Bibliography

Agland, S. (2014). "*CG rendering and ACES*". http://nbviewer.ipython.org/gist/sagland/3c791e79353673fd24fa. Animal Logic.

Amanatides, J. and Woo, A. (1987). "*A fast voxel traversal algorithm for ray tracing*". Eurographics 87:3.

Ament, M., Bergmann, C., and Weiskopf, D. (2014). "*Refractive radiative transfer equation*". ACM Transactions on Graphics 33:2.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). "*Particle Markov chain Monte Carlo methods*". Journal of the Royal Statistical Society 72:3.

Arvo, J. (1986). "*Backward ray tracing*". SIGGRAPH Course Notes.

Barsky, B. A., Horn, D. R., Klein, S. A., Pang, J. A., and Yu, M. (2003). "*Camera models and optical systems used in computer graphics: part I, object-based techniques*". Proceedings of the International Conference on Computational Science and Its Applications.

Bitterli, B., Jakob, W., Novák, J., and Jarosz, W. (2017). "*Reversible jump Metropolis light transport using inverse mappings*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 37:1.

Bitterli, B. and Jarosz, W. (2017). "*Beyond points and beams: higher-dimensional photon samples for volumetric light transport*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36:4.

Bitterli, B. and Jarosz, W. (2019). "*Selectively Metropolised Monte Carlo light transport simulation*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 38:6.

Bitterli, B., Ravichandran, S., Müller, T., Wrenninge, M., Novák, J., Marschner, S., and Jarosz, W. (2018). "*A radiative transfer framework for non-exponential media*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 37:6.

Cappé, O., Douc, R., Guillin, A., Marin, J.-M., and Robert, C. P. (2008). "*Adaptive importance sampling in general mixture classes*". Statistics and Computing 18:4.

Cappé, O., Guillin, A., Marin, J.-M., and Robert, C. (2004). "*Population Monte Carlo*". Journal of Computational and Graphical Statistics 13:4.

Chaitanya, C. R. A., Kaplanyan, A. S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., and Aila, T. (2017). "*Interactive reconstruction of Monte Carlo image sequences using* *a recurrent denoising autoencoder*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36:4.

Chandrasekar, S. (1960). "*Radiative transfer*". Dover Publications Inc.

Chen, J., Paris, S., and Durand, F. (2007). "*Real-time edge-aware image processing with the bilateral grid*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 26:3.

Chopin, N., Jacob, P. E., and Papaspiliopoulos, O. (2013). "*SMC2: an efficient algorithm for sequential analysis of state space models*". Journal of the Royal Statistical Society: Series B (Statistical Methodology) 75:3.

CIE (2004). "*Colorimetry*". Technical report. Commision Internationale de l'Eclairage.

CIE (2008). "*Colorimetry - part 4: 1976 l*a*b* colour space*". Technical report. Commision Internationale de l'Eclairage.

Cline, D., Talbot, J., and Egbert, P. K. (2005). "*Energy redistribution path tracing*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 24:3.

Dachsbacher, C., Křivánek, J., Hašan, M., Arbree, A., Walter, B., and Novák, J. (2013). "*Scalable realistic rendering with many-light methods*". Computer Graphics Forum 33:1.

Dahm, K. and Keller, A. (2017a). "*Learning light transport the reinforced way*". ArXiv e-prints. arXiv: 1701.07403.

Dahm, K. and Keller, A. (2017b). "*Machine learning and integral equations*". ArXiv e-prints. arXiv: 1712.06115.

Dammertz, H., Hanika, J., and Keller, A. (2008). "*Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays*". Computer Graphics Forum (Proceedings of Eurographics) 27:4.

Davidovič, T., Křivánek, J., Hašan, M., Slusallek, P., and Bala, K. (2010). "*Combining global and local virtual lights for detailed glossy illumination*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 29:6.

DeCoro, C., Weyrich, T., and Rusinkiewicz, S. (2010). "*Density-based outlier rejection in Monte Carlo rendering*". Computer Graphics Forum (Proceedings of Pacific Graphics) 29:7.

Del Moral, P. (1996). "*Non linear filtering: interacting particle solution*". Markov Processes and Related Fields 2:4.

Devroye, L. (1986). "*Non-uniform random variate generation*". Springer.

Douc, R., Guillin, A., Marin, J., and Robert, C. (2007). "*Convergence of adaptive mixtures of importance sampling schemes*". ArXiv e-prints. arXiv: 0708.0711.

Dutre, P., Bala, K., Bekaert, P., and Shirley, P. (2006). "*Advanced global illumination*". AK Peters Ltd.

Ebert, D. S. and Musgrave, F. K. (2003). "*Texturing and modeling: a procedural approach*". Morgan Kaufmann.

Engelhardt, T. and Dachsbacher, C. (2008). "*Octahedron environment maps*". Proceedings of Vision, Modeling and Visualization.

Engelhardt, T., Novák, J., Schmidt, T.-W., and Dachsbacher, C. (2012). "*Approximate bias compensation for rendering scenes with heterogeneous participating media*". Computer Graphics Forum 31:7.

Epanechnikov, V. A. (1969). "*Non-parametric estimation of a multivariate probability density*". Theory of Probability & Its Applications 14:1.

Fan, S. (2006). "*Sequential Monte Carlo methods for physically based rendering*". PhD thesis. University of Wisconsin-Madison.

Fascione, L., Hanika, J., Fajardo, M., Christensen, P., Burley, B., and Green, B. (2017). "*Path tracing in production – part 1: writing production renderers*". SIGGRAPH Courses.

Fasiolo, M., Melo, F. E. de, and Maskell, S. (2018). "*Langevin incremental mixture importance sampling*". Statistics and Computing 28:3.

Georgiev, I. and Slusallek, P. (2010). "*Simple and robust iterative importance sampling of virtual point lights*". Eurographics short papers.

Georgiev, I., Křivánek, J., Davidovič, T., and Slusallek, P. (2012a). "*Light transport simulation with vertex connection and merging*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 31:6.

Georgiev, I., Křivánek, J., Hachisuka, T., Nowrouzezahrai, D., and Jarosz, W. (2013). "*Joint importance sampling of low-order volumetric scattering*". ACM Transactions on Graphics 32:6.

Georgiev, I., Křivánek, J., Popov, S., and Slusallek, P. (2012b). "*Importance caching for complex illumination*". Computer Graphics Forum (Proceedings of Eurographics) 31:3.

Gilks, W. and Berzuini, C. (2001). "*Following a moving target – Monte Carlo inference for dynamic Bayesian models*". Journal of the Royal Statistical Society 63:1.

Glassner, A. S. (1984). "*Space subdivision for fast ray tracing*". IEEE Computer Graphics and Applications 4:10.

Grimmett, G. and Welsh, D. (2014). "*Probability: an introduction*". 2nd. Oxford University Press.

Gruson, A., Ribardière, M., Šik, M., Vorba, J., Cozot, R., Bouatouch, K., and Křivánek, J. (2016). "*A spatial target function for Metropolis photon tracing*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36:1.

Guo, J. J., Bauszat, P., Bikker, J., and Eisemann, E. (2018). "*Primary sample space path guiding*". Eurographics Symposium on Rendering - Experimental Ideas & Implementations. The Eurographics Association.

Hachisuka, T., Jarosz, W., Weistroffer, R. P., Dale, K., Humphreys, G., Zwicker, M., and Jensen, H. W. (2008a). "*Multidimensional adaptive sampling and reconstruction for ray tracing*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 27:3.

Hachisuka, T. and Jensen, H. W. (2009). "*Stochastic progressive photon mapping*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 28:5.

Hachisuka, T. and Jensen, H. W. (2011). "*Robust adaptive photon tracing using photon path visibility*". ACM Transactions on Graphics 30:5.

Hachisuka, T., Kaplanyan, A. S., and Dachsbacher, C. (2014). "*Multiplexed Metropolis light transport*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 33:4.

Hachisuka, T., Ogaki, S., and Jensen, H. W. (2008b). "*Progressive photon mapping*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 27:5.

Hachisuka, T., Pantaleoni, J., and Jensen, H. W. (2012). "*A path space extension for robust light transport simulation*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 31:6.

Hanika, J. and Dachsbacher, C. (2014). "*Efficient Monte Carlo rendering with realistic lenses*". Computer Graphics Forum (Proceedings of Eurographics) 33:2.

Hanika, J., Kaplanyan, A., and Dachsbacher, C. (2015). "*Improved half vector space light transport*". Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 34:4.

Hašan, M., Křivánek, J., Walter, B., and Bala, K. (2009). "*Virtual spherical lights for many-light rendering of glossy scenes*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 28:5.

Hašan, M., Pellacini, F., and Bala, K. (2007). "*Matrix row-column sampling for the many-light problem*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 26:3.

Hastings, W. K. (1970). "*Monte Carlo sampling methods using Markov chains and their applications*". Biometrika 57:1.

Havran, V. (2000). "*Heuristic ray shooting algorithms*". PhD thesis. Technical University in Prague.

Hedman, P., Karras, T., and Lehtinen, J. (2016). "*Sequential Monte Carlo instant radiosity*". Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.

Hege, H.-C., Höllerer, T., and Stalling, D. (1996). "*Volume rendering: mathematical models and algorithmic aspects*". Technical report. Zuse Institute Berlin.

Heitz, E. (2014). “*Understanding the masking-shadowing function in microfacet-based BRDFs*”. Journal of Computer Graphics Techniques 3:2.

Henyey, L. G. and Greenstein, J. L. (1941). “*Diffuse radiation in the galaxy*”. The Astrophysical Jornal 93.

Herholz, S., Elek, O., Vorba, J., Lensch, H., and Křivánek, J. (2016). “*Product importance sampling for light transport path guiding*”. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 35:4.

Herholz, S., Zhao, Y., Elek, O., Nowrouzezahrai, D., Lensch, H. P. A., and Křivánek, J. (2019). “*Volume path guiding based on zero-variance random walk theory*”. ACM Transactions on Graphics 38:3.

Hey, H. and Purgathofer, W. (2002). “*Importance sampling with hemi-spherical particle footprints*”. Proceedings of the Spring Conference on Computer Graphics.

Hullin, M., Hanika, J., and Heidrich, W. (2012). “*Polynomial Optics: a construction kit for efficient ray-tracing of lens systems*”. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 31:4.

ITU (2002). “*Recommendation ITU-R BT.709-5: parameter values for the HDTV standards for production and international programme exchange*”. Technical report. International Telecommunication Union.

Jakob, W. (2013). “*Light transport on path-space manifolds*”. PhD thesis. Cornell University.

Jakob, W., Arbree, A., Moon, J. T., Bala, K., and Marschner, S. (2010). “*A radiative transfer framework for rendering materials with anisotropic structure*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 29:4.

Jakob, W. and Hanika, J. (2019). “*A low-dimensional function space for efficient spectral upsampling*”. Computer Graphics Forum (Proceedings of Eurographics) 38:2.

Jakob, W. and Marschner, S. (2012). “*Manifold exploration: a Markov chain Monte Carlo technique for rendering scenes with difficult specular transport*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 31:4.

Jakob, W., Regg, C., and Jarosz, W. (2011). “*Progressive expectation-maximization for hierarchical volumetric photon mapping*”. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 3:4.

Jarabo, A., Aliaga, C., and Gutierrez, D. (2018). “*A radiative transfer framework for spatially-correlated materials*”. ACM Transactions on Graphics 37:4.

Jarosz, W., Zwicker, M., and Jensen, H. W. (2008). “*The beam radiance estimate for volumetric photon mapping*”. Computer Graphics Forum (Proceedings of Eurographics) 27:2.

Jensen, H. W. (1995). "*Importance driven path tracing using the photon map*". Proceedings of the Eurographics Workshop on Rendering.

Jensen, H. W. (1996). "*Global illumination using photon maps*". Proceedings of the Eurographics Workshop on Rendering.

Jensen, H. W. and Christensen, P. H. (1998). "*Efficient simulation of light transport in scenes with participating media using photon maps*". Proceedings of SIGGRAPH.

Kajiya, J. T. (1986). "*The rendering equation*". Computer Graphics (Proceedings of SIGGRAPH) 20:4.

Kajiya, J. T. and Von Herzen, B. P. (1984). "*Ray tracing volume densities*". Computer Graphics (Proceedings of SIGGRAPH).

Kaplanyan, A. S. and Dachsbacher, C. (2013). "*Adaptive progressive photon mapping*". ACM Transactions on Graphics 32:2.

Kaplanyan, A. and Dachsbacher, C. (2013). "*Path space regularization for holistic and robust light transport*". Computer Graphics Forum (Proceedings of Eurographics) 32:3.

Kaplanyan, A., Hanika, J., and Dachsbacher, C. (2014). "*The natural-constraint representation of the path space for efficient light transport simulation*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 33:4.

Kelemen, C., Szirmay-Kalos, L., Antal, G., and Csonka, F. (2002). "*A simple and robust mutation strategy for the Metropolis light transport algorithm*". Computer Graphics Forum 21:3.

Keller, A. (1997). "*Instant radiosity*". SIGGRAPH '97.

Kettunen, M., Manzi, M., Aittala, M., Lehtinen, J., Durand, F., and Zwicker, M. (2015). "*Gradient-domain path tracing*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 34:4.

Kirk, D. and Arvo, J. (1991). "*Unbiased sampling techniques for images synthesis*". Computer Graphics (Proceedings of SIGGRAPH) 25:4.

Knaus, C. and Zwicker, M. (2011). "*Progressive photon mapping: a probabilistic approach*". ACM Transactions on Graphics 30:3.

Kollig, T. and Keller, A. (2002a). "*Efficient bidirectional path tracing by randomized quasi-Monte Carlo integration*". Monte Carlo and Quasi-Monte Carlo Methods.

Kollig, T. and Keller, A. (2002b). "*Efficient multidimensional sampling*". Computer Graphics Forum (Proceedings of Eurographics) 21:3.

Kollig, T. and Keller, A. (2004). "*Illumination in the presence of weak singularities*". Monte Carlo and Quasi-Monte Carlo Methods.

Křivánek, J. and d'Eon, E. (2014). "*A zero-variance-based sampling scheme for Monte Carlo subsurface scattering*". SIGGRAPH Talks.

Křivánek, J., Ferwerda, J. A., and Bala, K. (2010). "*Effects of global illumination approximations on material appearance*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 29:4.

Křivánek, J., Georgiev, I., Hachisuka, T., Vévoda, P., Šik, M., Nowrouzezahrai, D., and Jarosz, W. (2014). "*Unifying points, beams, and paths in volumetric light transport simulation*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 33:4.

Kulla, C. and Fajardo, M. (2011). "*Importance sampling of area lights in participating media*". SIGGRAPH Talks.

Kulla, C. and Fajardo, M. (2012). "*Importance sampling techniques for path tracing in participating media*". Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering).

Kutz, P., Habel, R., Li, Y. K., and Novák, J. (2017). "*Spectral and decomposition tracking for rendering heterogeneous volumes*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36:4.

Lafortune, E. P. and Willems, Y. D. (1996). "*Rendering participating media with bidirectional path tracing*". Proceedings of the Eurographics Workshop on Rendering.

Lafortune, E. and Willems, Y. (1993). "*Bi-directional path tracing*". Proceedings of the International Conference on Computational Graphics and Visualization Techniques.

Lai, Y.-C., Fan, S. H., Chenney, S., and Dyer, C. (2007). "*Photorealistic image rendering with population Monte Carlo energy redistribution*". Proceedings of the Eurographics Workshop on Rendering. The Eurographics Association.

Li, T.-M., Lehtinen, J., Ramamoorthi, R., Jakob, W., and Durand, F. (2015). "*Anisotropic Gaussian mutations for Metropolis light transport through Hessian-Hamiltonian dynamics*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 34:6.

MacAdam, D. L. (1935a). "*Maximum visual efficiency of colored materials*". Journal of the Optical Society of America 25:11.

MacAdam, D. L. (1935b). "*The theory of the maximum visual efficiency of colored materials*". Journal of the Optical Society of America 25:8.

Maloney, L. T. (1986). "*Evaluation of linear models of surface spectral reflectance with small numbers of parameters*". Journal of the Optical Society of America 3:10.

Mansencal, T., Mauderer, M., and Canavan, L. (2015). https://colour-science.org/posts/about-rgb-colourspace-models-performance.

Manzi, M., Kettunen, M., Durand, F., Zwicker, M., and Lehtinen, J. (2016). "*Temporal gradient-domain path tracing*". ACM Transactions on Graphics 35:6.

Marschner, S. and Shirley, P. (2015). "*Fundamentals of computer graphics*". 4th. CRC Press.

Matsumoto, M. and Nishimura, T. (1998). “*Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*”. ACM Transactions on Modeling and Computer Simulation 8:1.

McCluney, W. R. (2014). “*Introduction to radiometry and photometry*”. 2nd. Artech House.

Meng, J., Hanika, J., and Dachsbacher, C. (2016). “*Improving the Dwivedi sampling scheme*”. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 35:4.

Meng, J., Simon, F., Hanika, J., and Dachsbacher, C. (2015). “*Physically meaningful rendering using tristimulus colours*”. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 34:4.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). “*Equation of state calculations by fast computing machines*”. The Journal of Chemical Physics 21:6.

Metropolis, N. and Ulam, S. (1949). “*The Monte Carlo method*”. Journal of the American statistical association 44:247.

Meyer, G. W. (1988). “*Wavelength selection for synthetic image generation*”. Computer Vision, Graphics and Image Processing 41:1.

Mojzík, M., Fichet, A., and Wilkie, A. (2018). “*Handling fluorescence in a uni-directional spectral path tracer*”. Computer Graphics Forum 37:4.

Morley, R. K., Boulos, S., Johnson, J., Edwards, D., Shirley, P., Ashikhmin, M., and Premože, S. (2006). “*Image synthesis using adjoint photons*”. Proceedings of Graphics Interface.

Müller, T., Gross, M., and Novák, J. (2017). “*Practical path guiding for efficient light-transport simulation*”. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 36:4.

Müller, T., Mcwilliams, B., Rousselle, F., Gross, M., and Novák, J. (2019). “*Neural importance sampling*”. ACM Transactions on Graphics 38:5.

Murray-Coleman, J. and Smith, A. (1990). “*The automated measurement of BRDFs and their application to luminaire modeling*”. Journal of the Illuminating Engineering Society 19:1.

Museth, K. (2013). “*VDB: high-resolution sparse volumes with dynamic topology*”. ACM Transactions on Graphics 32:3.

Nicodemus, F. E., Richmond, J. C., Hsia, J. J., Ginsberg, I. W., and Limperis, T. (1977). “*Geometrical considerations and nomenclature for reflectance*”. Final Report National Bureau of Standards, Washington, DC. Inst. for Basic Standards.

Novák, J., Engelhardt, T., and Dachsbacher, C. (2011). "*Screen-space bias compensation for interactive high-quality global illumination with virtual point lights*". Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.

Novák, J., Nowrouzezahrai, D., Dachsbacher, C., and Jarosz, W. (2012a). "*Progressive virtual beam lights*". Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 31:4.

Novák, J., Nowrouzezahrai, D., Dachsbacher, C., and Jarosz, W. (2012b). "*Virtual ray lights for rendering scenes with participating media*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 31:4.

Novák, J., Selle, A., and Jarosz, W. (2014). "*Residual ratio tracking for estimating attenuation in participating media*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 33:6.

O'Neill, B. (2006). "*Elementary differential geometry*". 2nd. Elsevier.

O'Neill, M. E. (2014). "*PCG: a family of simple fast space-efficient statistically good algorithms for random number generation*". Technical report. Harvey Mudd College.

Otsu, H., Kaplanyan, A. S., Hanika, J., Dachsbacher, C., and Hachisuka, T. (2017). "*Fusing state spaces for Markov chain Monte Carlo rendering*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36:4.

Otsu, H., Masafumi, Y., and Hachisuka, T. (2018). "*Reproducing spectral reflectances from tristimulus colours*". Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 37:6.

Ou, J. and Pellacini, F. (2011). "*Lightslice: matrix slice sampling for the many-lights problem*". ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia).

Pantaleoni, J. (2017). "*Charted Metropolis light transport*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36:4.

Pauly, M., Kollig, T., and Keller, A. (2000). "*Metropolis light transport for participating media*". Proceedings of the Eurographics Workshop on Rendering.

Peercy, M. S. (1993). "*Linear color representations for full speed spectral rendering*". Proceedings of Computer Graphics and Interactive Techniques. SIGGRAPH '93. New York, NY, USA.

Perlin, K. and Hoffert, E. M. (1989). "*Hypertexture*". Proceedings of Computer Graphics and Interactive Techniques.

Perlin, K. (1985). "*An image synthesizer*". ACM Siggraph Computer Graphics 19:3.

Peter, I. and Pietrek, G. (1998). "*Importance driven construction of photon maps*". Proceedings of the Eurographics Workshop on Rendering.

Peters, C., Merzbach, S., Hanika, J., and Dachsbacher, C. (2019). “*Using moments to represent bounded signals for spectral rendering*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 38:4.

Pharr, M., Jakob, W., and Humphreys, G. (2017). “*Physically based rendering: from theory to implementation*”. 3rd. Morgan Kaufmann.

Qin, H., Sun, X., Hou, Q., Guo, B., and Zhou, K. (2015). “*Unbiased photon gathering for light transport simulation*”. ACM Trans. Graph. 34:6.

Raab, M., Seibert, D., and Keller, A. (2008). “*Unbiased global illumination with participating media*”. Monte Carlo and Quasi-Monte Carlo Methods.

Reibold, F., Hanika, J., Jung, A., and Dachsbacher, C. (2018). “*Selective guided sampling with complete light transport paths*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 37:6.

Royden, H. and Fitzpatrick, P. (2010). “*Real analysis*”. 4th. Prentice Hall.

Rubin, S. M. and Whitted, T. (1980). “*A 3-dimensional representation for fast rendering of complex scenes*”. Proceedings of Computer Graphics and Interactive Techniques.

Schied, C. et al. (2017). “*Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination*”. Proceedings of High Performance Graphics.

Schrade, E., Hanika, J., and Dachsbacher, C. (2016). “*Sparse high-degree polynomials for wide-angle lenses*”. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 35:4.

Schrödinger, E. (1919). “*Theorie der Pigmente größter Leuchtkraft*”. Annalen der Physik 367:15.

Segovia, B., Iehl, J. C., Mitanchey, R., and Péroche, B. (2006). “*Bidirectional instant radiosity.*” Proceedings of the Eurographics Conference on Rendering Techniques.

Segovia, B., Iehl, J. C., and Péroche, B. (2007). “*Metropolis instant radiosity*”. Computer Graphics Forum 26:3.

Šik, M. and Křivánek, J. (2019). “*Implementing one-click caustics in Corona renderer*”. Eurographics Symposium on Rendering - Industry Track.

Šik, M., Otsu, H., Hachisuka, T., and Křivánek, J. (2016). “*Robust light transport simulation via Metropolised bidirectional estimators*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 35:6.

Simon, F., Hanika, J., and Dachsbacher, C. (2015). “*Rich-VPLs for improving the versatility of many-light methods*”. Computer Graphics Forum (Proceedings of Eurographics) 34:2.

Simon, F., Hanika, J., Zirr, T., and Dachsbacher, C. (2017). “*Line integration for rendering heterogeneous emissive volumes*”. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 36:4.

Širca, S. (2016). "*Probability for physicists*". Springer.

Smith, T. and Guild, J. (1931). "*The CIE colorimetric standards and their use*". Transactions of the Optical Society 33:3.

Smits, B. (1999). "*An RGB-to-spectrum conversion for reflectances*". Journal of Graphics Tools 4:4.

Sobol, I. (1994). "*A primer for the Monte Carlo method*". CRC Press.

Spanier, J. and Gelbard, E. M. (1969). "*Monte Carlo principles and neutron transport problems*". Addison-Wesley.

Spencer, B. and Jones, M. (2009). "*Into the blue: better caustics through photon relaxation*". Computer Graphics Forum 28:2.

Spencer, B. and Jones, M. W. (2013). "*Progressive photon relaxation*". ACM Transactions on Graphics 32:1.

Stone, M. C., Cowan, W. B., and Beatty, J. C. (1988). "*Color gamut mapping and the printing of digital color images*". ACM Transactions on Graphics 7:4.

Stroud, A. H. and Secrest, D. (1966). "*Gaussian quadrature formulas*". Prentice-Hall.

Sun, Y., Fracchia, F. D., Calvert, T. W., and Drew, M. S. (1999). "*Deriving spectra from colors and rendering light interference*". IEEE Computer Graphics and Applications 19:4.

Szirmay-Kalos, L., Tóth, B., and Magdics, M. (2011). "*Free path sampling in high resolution inhomogeneous participating media*". Computer Graphics Forum.

Tokuyoshi, Y. (2015). "*Virtual spherical Gaussian lights for real-time glossy indirect illumination*". Computer Graphics Forum 34:7.

Turk, G. (1991). "*Generating textures on arbitrary surfaces using reaction-diffusion*". Computer Graphics (Proceedings of SIGGRAPH) 25:4.

Van de Woestijne, J., Frederickx, R., Billen, N., and Dutré, P. (2017). "*Temporal coherence for Metropolis light transport*". Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations.

Veach, E. (1998). "*Robust Monte Carlo methods for light transport simulation*". PhD thesis. Stanford University.

Veach, E. and Guibas, L. J. (1994). "*Bidirectional estimators for light transport*". Proceedings of the Eurographics Workshop on Rendering.

Veach, E. and Guibas, L. J. (1995). "*Optimally combining sampling techniques for Monte Carlo rendering*". Proceedings of SIGGRAPH.

Veach, E. and Guibas, L. J. (1997). "*Metropolis light transport*". Proceedings of SIGGRAPH.

Vévoda, P., Kondapaneni, I., and Křivánek, J. (2018). "*Bayesian online regression for adaptive direct illumination sampling*". ACM Transactions on Graphics 37:4.

Villemin, R. and Hery, C. (2013). “*Practical illumination from flames*”. Journal of Computer Graphics Techniques 2:2.

Vorba, J., Karlík, O., Šik, M., Ritschel, T., and Křivánek, J. (2014). “*On-line learning of parametric mixture models for light transport simulation*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 33:4.

Vorba, J. and Křivánek, J. (2016). “*Adjoint-driven Russian roulette and splitting in light transport simulation*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 35:4.

Vorba, J., Hanika, J., Herholz, S., Müller, T., Křivánek, J., and Keller, A. (2019). “*Path guiding in production*”. ACM SIGGRAPH 2019 Courses. SIGGRAPH ’19.

Wald, I. (2004). “*Realtime ray tracing and interactive global illumination*”. PhD thesis. Saarland University.

Wald, I., Woop, S., Benthin, C., Johnson, G. S., and Ernst, M. (2014). “*Embree: a kernel framework for efficient CPU ray tracing*”. ACM Transactions on Graphics 33:4.

Walter, B., Arbree, A., Bala, K., and Greenberg, D. P. (2006). “*Multidimensional lightcuts*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 25:3.

Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., and Greenberg, D. P. (2005). “*Lightcuts: a scalable approach to illumination*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 24:3.

Walter, B., Khungurn, P., and Bala, K. (2012). “*Bidirectional lightcuts*”. ACM Transactions on Graphics (Proceedings of SIGGRAPH) 31:4.

Walter, B., Marschner, S., Li, H., and Torrance, K. (2007). “*Microfacet models for refraction through rough surfaces*”. Proceedings of the Eurographics Symposium on Rendering.

Ward, G. J. et al. (1992). “*Measuring and modeling anisotropic reflection*”. Computer Graphics 26:2.

Weber, P., Hanika, J., and Dachsbacher, C. (2017). “*Multiple vertex next event estimation for lighting in dense, forward-scattering media*”. Computer Graphics Forum (Proceedings of Eurographics) 36:2.

Wilkie, A., Nawaz, S., Droske, M., Weidlich, A., and Hanika, J. (2014). “*Hero wavelength spectral sampling*”. Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 33:4.

Woodcock, E., Murphy, T., Hemmings, P., and Longworth, S. (1965). “*Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry*”. Proceedings of the Conference on Applications of Computing Methods to Reactor Problems.

Wrenninge, M. (2016). “*Efficient rendering of volumetric motion blur using temporally unstructured volumes*”. Journal of Computer Graphics Techniques 5:1.

WRIGHT, W. D. (1928). "*A re-determination of the trichromatic coefficients of the spectral colours*". Transactions of the Optical Society 30:4.

YUKSEL, C. and YUKSEL, C. (2017). "*Lighting grid hierarchy for self-illuminating explosions*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36:4.

ZHAO, S., RAMAMOORTHI, R., and BALA, K. (2014). "*High-order similarity relations in radiative transfer*". ACM Transactions on Graphics (Proceedings of SIGGRAPH) 33:4.

ZIRR, T., HANIKA, J., and DACHSBACHER, C. (2018). "*Reweighting firefly samples for improved finite-sample Monte Carlo estimates*". Computer Graphics Forum 37:6.

ZWICKER, M., JAROSZ, W., LEHTINEN, J., MOON, B., RAMAMOORTHI, R., ROUSSELLE, F., SEN, P., SOLER, C., and YOON, S.-E. (2015). "*Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering*". Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports) 34:2.