Date of acceptance Grade

Instructor

Sequential Monte Carlo Instant Radiosity

Peter Hedman

Helsinki May 21, 2015

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Peter Hedman

Työn nimi — Arbetets titel — Title

Sequential Monte Carlo Instant Radiosity

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	May 21, 2015	95 pages + 6 appendices

Tiivistelmä — Referat — Abstract

The focus of this thesis is to accelerate the synthesis of physically accurate images using computers. Such images are generated by simulating how light flows in the scene using unbiased Monte Carlo algorithms. To date, the efficiency of these algorithms has been too low for real-time rendering of error-free images. This limits the applicability of physically accurate image synthesis in interactive contexts, such as pre-visualization or video games.

We focus on the well-known Instant Radiosity algorithm by Keller [1997], that approximates the indirect light field using *virtual point lights* (VPLs). This approximation is unbiased and has the characteristic that the error is spread out over large areas in the image. This low-frequency noise manifests as an unwanted "flickering" effect in image sequences if not kept temporally coherent. Currently, the limited VPL budget imposed by running the algorithm at interactive rates results in images which may noticeably differ from the ground-truth.

We introduce two new algorithms that alleviate these issues. The first, *clustered hierarchical importance sampling*, reduces the overall error by increasing the VPL budget without incurring a significant performance cost. It uses an unbiased Monte Carlo estimator to estimate the sensor response caused by all VPLs. We reduce the variance of this estimator with an efficient hierarchical importance sampling method. The second, *sequential Monte Carlo Instant Radiosity*, generates the VPLs using heuristic sampling and employs non-parametric density estimation to resolve their probability densities. As a result the algorithm is able to reduce the number of VPLs that move between frames, while also placing them in regions where they bring light to the image. This increases the quality of the individual frames while keeping the noise temporally coherent – and less noticeable – between frames.

When combined, the two algorithms form a rendering system that performs favourably against traditional path tracing methods, both in terms of performance and quality. Unlike prior VPL-based methods, our system does not suffer from the objectionable lack of temporal coherence in highly occluded scenes.

ACM Computing Classification System (CCS):

- **Computing methodologies∼Rendering**
- *Mathematics of computing∼Sequential Monte Carlo methods*
- *Mathematics of computing∼Density estimation*

Avainsanat — Nyckelord — Keywords
Computer Graphics, Instant Radiosity, Sequential Monte Carlo, Ray Tracing

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

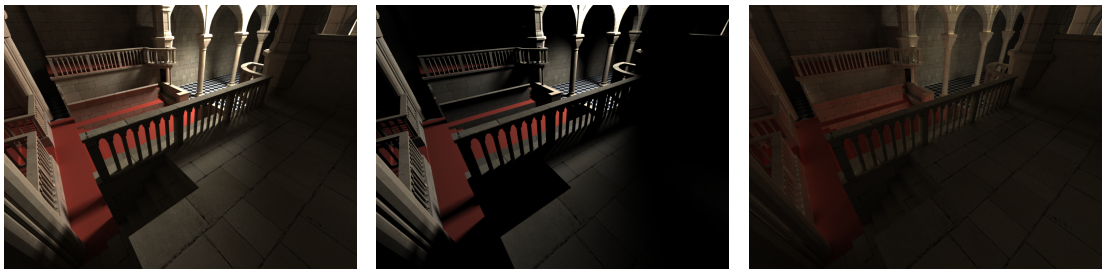
Acknowledgements

First and foremost I would like to thank Tero Karras and Jaakko Lehtinen for their invaluable guidance and insight during the research phase of this thesis. I would also like to thank Timo Aila and Samuli Laine for a multitude of productive brainstorming sessions.

I would also like to thank my father, Lars Hedman, who patiently helped me tie up loose ends and acted as a sounding board as I was writing the thesis. Thanks to Jeremias Berg for providing insight into the mathematical foundations of this thesis. Thanks to Teemu Roos who generously dedicated his time to review a thesis of this length.

Thanks to Marco Dabrovic for the Sibenik Cathedral Scene, Guillermo M. Leal Llaguno for the San Miguel Scene, Anat Grynberg and Greg Ward for the Conference Room Scene, Frank Meinl at Crytek for the Sponza Scene and the Walkthru project at EECS Berkeley for the Soda Hall Scene.

Contents



(a) Global illumination. (b) Direct illumination. (c) Indirect illumination.

Figure 1: A breakdown of the global illumination in an image into its direct and indirect components.

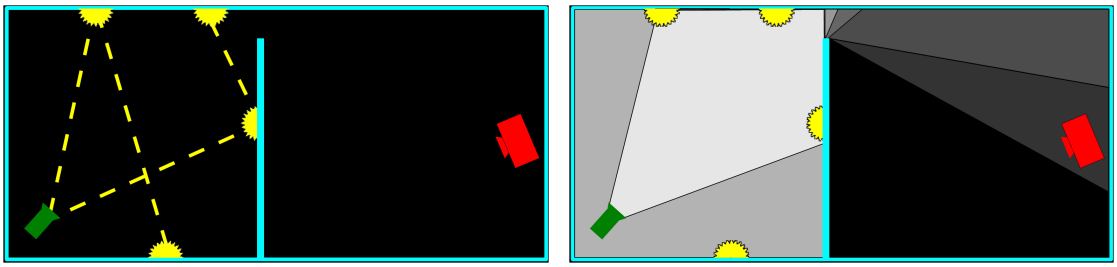
1 Introduction

Physically based computer graphics aims to generate images that are as realistic as possible. The film industry makes use of such graphics for special effects and animated full length films. They are also used for architectural presentations and product visualization, where accurately predicting appearance is important. Modern video games strive for more realistic graphics, but are constrained by the need to synthesize images in real-time. In other words, the delay between input and an updated image in the game needs to be small enough to prevent the player from perceiving individual images. [Dutre et al., 2006, Chapter 1.1.1]

To synthesize an image, we have to simulate the light which arrives at a virtual camera. This simulation needs to account how light emitted from the light sources interacts with the surfaces in the scene. When light reaches a surface, the material dictates how much of that light is reflected, transmitted or absorbed. The complexity of realistic image synthesis can be appreciated by considering the multitude of routes light can take from a light source to the camera. This is further exacerbated by the fact that light can reflect arbitrarily many times off the surfaces in the scene.

One way to describe the light that falls on a surface in the scene is to break it down into two components, *direct* light and *indirect* light. Direct light arrives straight from one of the light sources while indirect light has been reflected off the objects in the scenes at least once. To synthesize physically accurate images we have to resolve the *global illumination* in the scene. This means that not only do we have to account for the direct light arriving on the surfaces, but we also has to consider *all* of the indirect light, see Figure 1 for an example.

Formally, synthesizing an image is achieved by solving the *rendering problem*. That



(a) VPLs are created by tracing photons. (b) The resulting illumination in the scene.

Figure 2: Traditional Instant Radiosity generates VPLs (in yellow) by shooting photons from the light source (in green). The resulting VPLs do not sufficiently illuminate the surfaces seen by the camera (in red). In reality, light from the light source is able to reach the the right side of the cyan wall.

is, resolving how much light falls on one or more virtual sensors in the scene. Commonly, the scene is described as a collection of objects and light sources. We know the location and sizes of the objects and also know the reflective properties of their surfaces. The same also applies to the locations and emissive properties of the light sources. A *rendering algorithm* takes as input a description of the scene and the locations of the virtual sensors (a virtual camera) and then produces an collection of sensor readings (an image) by simulating the light flows in the scene.

In this thesis our goal is to create a rendering algorithm for interactive image sequences, that is flexible enough to synthesize both rough approximations on limited hardware and physically accurate images in real-time on future hardware. We focus on the well-known Instant Radiosity rendering algorithm by Keller [1997]. It approaches the problem of global illumination using *virtual point lights* (VPLs) to represent indirect light. If we interpret the light emitted by the VPLs as direct it enables us to synthesize images with global illumination by only simulating direct light. This is advantageous, since images containing only direct light can be synthesized at real-time rates using hardware accelerated rendering algorithms. The algorithm generates VPLs by simulating how *photons* (particles of light) emitted from the light sources bounce around in the scene, see Figure 2a. During the simulation a VPL is deposited every time a photon bounces off a surface in the scene.

Instant Radiosity is able to generate physically accurate images if it runs for long enough. On the other hand, it can also run at real-time rates by generating fewer VPLs. In this case the error it makes is often spread out over large areas of the image (*low frequency noise*) but also manifests as bright spots when VPLs have

been placed near corners in the scene. In fast image sequences, low frequency noise that changes between images is perceived as an unwanted "flickering"-effect.

One way to alleviate these issues is to devise methods that increase the number of VPLs that bring light to the image. This can be achieved by a brute-force increase of the total VPL budget [Dong et al., 2009, Hašan et al., 2007, Ritschel et al., 2008, Segovia et al., 2006b, Walter et al., 2005]. However, many of the VPLs generated by traditional Instant Radiosity do not bring light to the image, for example, when they cannot illuminate the surfaces visible in the image (Figure 2b) or when they land on a highly absorbent (black) material. These issues become more prominent in larger and heavily occluded scenes where only a few of the light sources actually bring indirect light to the image. This implies that methods that deposit VPLs in a more intelligent fashion will not only result in higher quality images without increasing the VPL budget, but also become necessary to cope with larger scenes [Georgiev and Slusallek, 2010, Ritschel et al., 2011, Segovia et al., 2006a, 2007].

The "flickering"-effect can be reduced using complementary methods. For example, by preventing VPLs from changing places between images [Laine et al., 2007] or by smoothing the illumination over multiple images [Knecht et al., 2010, Mara et al., 2013]. However, no prior algorithm has simultaneously dealt with large and heavily occluded scenes while also alleviating the "flickering"-effect. Either these algorithms completely disregard temporal coherence or they fail to account for the layout of the scene when placing the VPLs.

The main contributions of this thesis are two rendering algorithms that improve upon Instant Radiosity. Together they produce results that match well with ground truth and keep the error temporally coherent — and thus less noticeable — between frames.

One, clustered hierarchical importance sampling, is a novel rendering method that greatly increases the number of VPLs available. It achieves this by using intelligently chosen subsets of the VPLs when evaluating the light received by each sensor. As shown in our experiments, this directly increases the quality of the illumination in the images.

The other, sequential Monte Carlo Instant Radiosity, reduces the number of VPLs that move between images in a sequence while at the same time placing them in regions where they bring light to the image. This algorithm produces VPLs that move gracefully between frames to account for the new location of the camera as well as the light sources, also in large scenes of which only a small portion is visible. It is

the first to achieve this while supporting multiple bounces of indirect illumination.

1.1 Thesis structure

The rest of this thesis organized as follows:

In Chapter 2 we introduce Monte Carlo integration and describe a collection of variance reduction techniques. In Chapter 3 we interpret Sequential Monte Carlo methods in the context of integration. We also introduce a framework for heuristic sequential sampling based on non-parametric density estimation.

In Chapter 4 we describe a mathematical model for illumination and rendering. In Chapter 5 we introduce two well-known rendering methods, forward path tracing and Instant Radiosity. We also compare techniques that evaluate visibility. In Chapter 6 we present spatial data structures that we make use of in the remainder of the thesis. In Chapter 7 we review previous improvements to the Instant Radiosity algorithm that reduce low frequency noise.

In Chapter 8 we attack the many-lights rendering problem using Monte Carlo estimators for the sums over all light sources. We present a powerful importance sampling algorithm to reduce the variance of such estimators. In Chapter 9 we describe a heuristic sequential sampling algorithm for VPLs based on the framework presented in Chapter 3.

In Chapter 10 we discuss important implementation details of the methods presented in Chapters 8 and 9. We also test these methods experimentally and analyse the results. In Chapter 11 we reiterate the results from the previous chapters and discuss potential extensions for our algorithms.

In Appendix A we give a simplified introduction to measure theory and describe well-known concepts from probability theory using a measure-therotic approach.

2 Monte Carlo integration

Monte Carlo methods estimate values using random simulations. The earliest example of such a method is an experiment by Comte de Buffon in 1777. The experiment involved dropping needles on to a wooden floor. Buffon proved that the probability

of a needle landing on two floor boards was

$$\frac{2L}{\pi d},$$

where L is the length of the needle and d is the width of a floor board. A result that can be used to estimate the value of π [Kalos and Whitlock, 1986, Chapter 1.3]. Modern Monte Carlo integration was introduced in the 1940s by Stanislaw Ulam and John von Neumann when they both worked on the Manhattan project. Ulam first thought of idea when trying to estimate the probability of successfully laying out all the cards in a game of solitaire [Eckhardt, 1987].

Rendering physically based images involves evaluating difficult integrals with analytic solutions only for very restricted cases. This forces rendering algorithms to make use of numerical integration methods, such as deterministic quadrature rules (e.g. Simpson's rule) or Monte Carlo integration. Which method is more suitable depends on the dimensionality of the domain. In this chapter we show that the error Monte Carlo integration is independent of the dimensionality of the domain. On the other hand, the error of quadratures increases exponentially with the dimensionality of the integration domain [Niederreiter, 1992, Chapter 1.1]. As a consequence, Monte Carlo methods are more commonly employed in rendering algorithms where the integrals that need to be evaluated often have high dimensional domains.

The book *Monte Carlo Methods* by Kalos and Whitlock [1986] is a good reference for Monte Carlo methods in general. Chapter 2 of *Robust Monte Carlo Methods for Light Transport Simulation* by Veach [1998] is a good introduction to the Monte Carlo methods commonly employed in rendering.

2.1 Monte Carlo integration

We want to estimate an integral of the form

$$\int_A f(x)\,\mathrm{d}\mu(x),$$

where μ is a measure defined on the measurable space $\mathbb{X}$, $f : \mathbb{X} \to \mathbb{R}$ and A is a measurable subset of $\mathbb{X}$. The first step towards Monte Carlo integration is the realization that we can express the integral using an expected value. That is, given a random variable $\mathcal{X} : \Omega_{\mathcal{X}} \to A$ and the *probability density function* (PDF) $p_{\mathcal{X}} = \dfrac{\mathrm{d}P_{\mathcal{X}}}{\mathrm{d}\mu}$,

we see that

$$E\left[\frac{f(X)}{p_{\mathcal{X}}(\mathcal{X})}\right] = \int_A \frac{f(x)}{p_{\mathcal{X}}(x)} \,\mathrm{d}P_{\mathcal{X}}(x) = \int_A \frac{f(x)}{p_{\mathcal{X}}(x)} p_{\mathcal{X}}(x) \,\mathrm{d}\mu(x) = \int_A f(x) \,\mathrm{d}\mu(x),$$

as long as $f(x) = 0$ whenever $p_{\mathcal{X}}(x) = 0$.

Law of large numbers. We already have an intuition that the expected value tells us where we should find the average of many outcomes from a random variable. We can justify this intuition with the help of the *weak law of large numbers* [Athreya and Lahiri, 2006, Chapter 8.1]. Given the random variables $(x_1, \ldots, x_n)$ which are independent and identically distributed (i.i.d), the weak law of large numbers states that the average

$$a_N = \frac{1}{N} \sum_{i=1}^{N} x_i$$

converges in probability to the expected value. In other words, for any $\epsilon > 0$

$$\lim_{N \to \infty} P(|a_N - E[x]| \geq \epsilon) = 0.$$

Monte Carlo estimators. Based on the weak law of large numbers we see that for any error bound $\epsilon > 0$, the error of the *Monte Carlo estimator*

$$\frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p_{\mathcal{X}}(x_i)}$$

compared to the integral

$$\int_A f(x) \,\mathrm{d}\mu(x)$$

will be less than ϵ with a very high probability as N tends to infinity. In other words, we can now form an estimate for the integral provided that we find a way to sample outcomes $(x_1, \ldots, x_n)$ from $\mathcal{X}$.

2.2 Sampling outcomes

Here we introduce ways to sample outcomes from $\mathcal{X}$. First, it is useful to note that we cannot use a deterministic algorithm to generate truly random values. One way around this is to use special hardware that monitors some physical process. A more common approach is to make use of an algorithm that generates sequences of

pseudo random numbers, which have roughly the same properties as sequences of i.i.d. truly random numbers [von Neumann, 1951]. A common example of a pseudo random number generator is the *Mersenne twister* by Matsumoto and Nishimura [1998]. Commonly, pseudo random numbers approximate outcomes from a uniformly distributed random variable $\mathcal{U}_{[0,1]} \sim \mathcal{U}([0,1])$.

For Monte Carlo integration it is often useful to perform importance sampling (Section 2.4), where the outcomes are sampled from a distribution that reduces the error of the estimator. Assume that we want to generate outcomes from $\mathcal{X} : \Omega_{\mathcal{X}} \to A$, where $A \subset \mathbb{R}$. If the *cumulative distribution function* (CDF) $F_{\mathcal{X}}$ is invertible, we can directly use uniformly distributed outcomes $(u_1, \ldots, u_N)$ from $\mathcal{U}_{[0,1]}$ and transform them into samples from $\mathcal{X}$, where

$$x_i = F_{\mathcal{X}}^{-1}(u_i), \text{ for every i } \leq N.$$

This sampling method is called the *analytic inversion method* or the *transformation method* and is covered in detail by Kalos and Whitlock [1986, Chapter 3.1].

If we cannot analytically invert the CDF, an alternative method is the *rejection method* [Kalos and Whitlock, 1986, Chapter 3.5]. Assume that we can generate outcomes from a random variable $\mathcal{Z}$ such that for some $M > 0$,

$$p_{\mathcal{Z}}(x)M \geq p_{\mathcal{X}}(x) \text{ for each } x \in A.$$

We now apply Algorithm 1 to generate outcomes from $\mathcal{X}$.

Algorithm 1 The rejection method

```
1: loop
2:     z ← OUTCOMEFROM(Z)
3:     u ← OUTCOMEFROM(U_[0,1])
4:     if p_Z(z)Mu < p_X(z) then
5:         return z
```

We will cover more specific sampling methods for light transport quantities in Chapter 5. Kalos and Whitlock [1986] provide a good overview of general purpose sampling methods in Chapter 3 of their book *Monte Carlo Methods*.

2.3 Bias and variance

Consider the error of a Monte Carlo estimator. If we let I be the exact value of our integral and $\hat{I}_N$ be the value of our estimator. In other words,

$$I = \int_A f(x)\,\mathrm{d}\mu(x) \text{ and } \hat{I}_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p_{\mathcal{X}}(x_i)}.$$

The error is expressed as

$$Err[\hat{I}_N] = \hat{I}_N - I$$

and the estimator *unbiased* if

$$E[Err[\hat{I}_N]] = 0$$

holds true for every N. An equally important requirement is that the estimator converges in probability to the correct result. More precisely, for every $\epsilon > 0$

$$\lim_{N\to\infty} P(|Err[\hat{I}_N]| \geq \epsilon) = 0.$$

We call such an estimator *consistent*.

Error bounds. We analyse the error of an unbiased estimator using the error metric known as *root mean squared error (RMSE)*, which is defined as

$$\sigma(\hat{I}_N) = \sqrt{E[(\hat{I}_N - E[\hat{I}_N])^2]} = \sqrt{E[(\hat{I}_N - I)^2]}.$$

We provide error bounds for the RMSE using the unbiased variance estimate [Kalos and Whitlock, 1986, Chapter 2.7]. More precisely,

$$\begin{aligned} Var(\hat{I}_N) &= E[\frac{1}{N-1}(\frac{1}{N}\sum_{i=1}^{N}(\frac{f(x_i)}{p_{\mathcal{X}}(x_i)})^2 - (\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p_{\mathcal{X}}(x_i)})^2)] \\ &= \frac{1}{N-1}(E[(\frac{f(\mathcal{X})}{p_{\mathcal{X}}(\mathcal{X})})^2] - E[\frac{f(\mathcal{X})}{p_{\mathcal{X}}(\mathcal{X})}]^2) \\ &= \frac{Var(\hat{I}_1)}{N-1} = O(\frac{1}{N}) \end{aligned}$$

and consequently the RMSE is $O(\sqrt{N}^{-1})$.

2.4 Importance sampling

Since Monte Carlo simulations only run for a finite amount of time, an estimator never converges fully. Instead, simulations are run until the error is low enough for

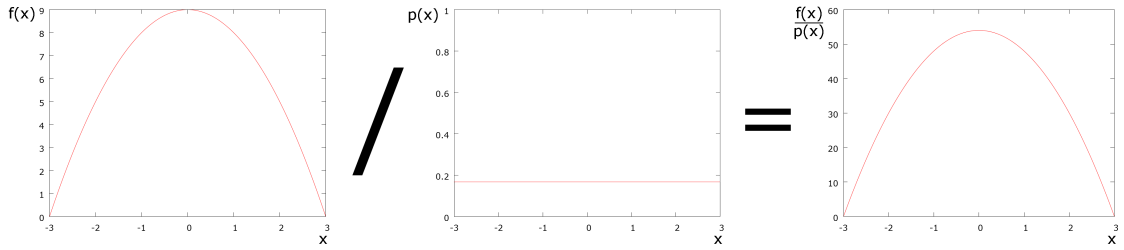


Figure 3: Sampling from a uniformly distributed random variable $\mathcal{X} \sim \mathcal{U}([-3, 3])$ not change the shape of the integrand at all. As a consequence, all outcomes in $[-3, 3]$ will have different contributions to the sum in a Monte Carlo estimator.

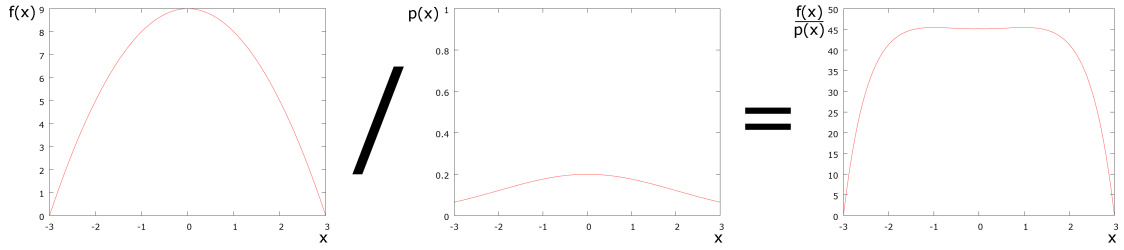


Figure 4: The PDF of a normally distributed random variable $\mathcal{X} \sim \mathcal{N}(0, 4)$ more closely follows the integrand and flattens out the function evaluated by each outcome. We see that any sample which lands in $[-2, 2]$ will evaluate to roughly the same value.

the application at hand. If we want to speed up the simulation time, we have to reduce variance of the Monte Carlo estimator. Importance sampling achieves this by sampling more outcomes in regions that are important to the integral.

Consider a positive function $f : \mathbb{X} \to [0, \infty]$ from a measurable space to the real numbers. Let A be a measurable subset of $\mathbb{X}$ and μ be a measure on $\mathbb{X}$. Our goal is to reduce the variance of the Monte Carlo estimator

$$\hat{I}_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p_{\mathcal{X}}(x_i)}$$

for the integral

$$I = \int_A f(x) \, \mathrm{d}\mu(x),$$

where $(x_1, \ldots, x_N)$ are i.i.d. outcomes from the random variable $\mathcal{X} : \Omega_{\mathcal{X}} \to A$.

Importance sampling is a variance reduction technique where we reduce the variance of $\hat{I}_1$ by changing the probability distribution our outcomes are sampled from. If we

rewrite the variance of $\hat{I}_1$, we see that

$$\begin{aligned} Var(\hat{I}_1) &= E[(\frac{f(\mathcal{X})}{p_{\mathcal{X}}(\mathcal{X})})^2] - E[\frac{f(\mathcal{X})}{p_{\mathcal{X}}(\mathcal{X})}]^2 \quad = E[(\frac{f(\mathcal{X})}{p_{\mathcal{X}}(\mathcal{X})})^2] - I^2 \\ &= \int_A (\frac{f(x)}{p_{\mathcal{X}}(x)})^2 p_{\mathcal{X}}(x)\,\mathrm{d}\mu(x) - I^2 = \int_A \frac{f(x)^2}{p_{\mathcal{X}}(x)}\,\mathrm{d}\mu(x) - I^2. \end{aligned}$$

Since is I^2 constant with respect to $p_{\mathcal{X}}$, the variance is minimized by choosing a probability distribution that minimizes

$$\int_A \frac{f(x)^2}{p_{\mathcal{X}}(x)}\,\mathrm{d}\mu(x).$$

Kalos and Whitlock [1986, Chapter 4.1] show that this expression minimized by a distribution whose PDF is proportional to f. In other words,

$$p_{\mathcal{X}}(x) = \frac{f(x)}{C}, \text{ where } C = \int_A f(x)\,\mathrm{d}\mu(x).$$

In fact, if we manage to sample from such a distribution, the variance of the estimator will be zero. That said, it is usually impossible to evaluate the PDF since the normalizing constant C is equal to I, the integral we want to estimate in the first place. However, we can still use this as guidance for what distribution we should use to generate our outcomes. In general, we want $p_{\mathcal{X}}(x)$ to be similar to $f(x)$. Visually this means that the graph of $\frac{f(x)}{p_{\mathcal{X}}(x)}$ should be as flat as possible. Intuitively, we want to flatten out the peaks in $f(x)$ with corresponding peaks in $p(x)$ without introducing new ones in places where $p(x)$ is significantly smaller than $f(x)$.

Figures 3 and 4 visually demonstrate the effect of importance sampling. The left graph shows the function $f(x) = 9 - x^2$ whose integral over the domain $[-3, 3]$ we're trying to estimate. The graph in the middle is the PDF $p(x)$ of the random variable we're sampling outcomes from and the right graph shows us the fraction $\frac{f(x)}{p(x)}$ which defines the terms in a Monte Carlo estimator for $\int_{-3}^{3} f(x)\,\mathrm{d}\mu(x) = 36$.

2.5 Markov chain Monte Carlo

A *Markov chain* is a sequence of random variables $(\mathcal{X}_1, \mathcal{X}_2, \dots)$, $\mathcal{X}_i : \Omega_{\mathcal{X}} \to X$, where $\mathcal{X}_i$ is conditionally independent of all the preceding variables given $\mathcal{X}_{i-1}$.

More precisely,

$$p_{\mathcal{X}_{1:i}}(x_i|x_1,\ldots,x_{i-1}) = p_{\mathcal{X}_{i-1:i}}(x_i|x_{i-1}) \text{ for all outcomes } (x_1,\ldots,x_i),$$

where $\mathcal{X}_{a:b}$ is the joint variable we get by combining the variables $(\mathcal{X}_a,\ldots,\mathcal{X}_b)$. A Markov chain is called *time homogeneous* if it satisfies

$$p_{\mathcal{X}_{i-1:i}}(x|y) = p_{\mathcal{X}_{j-1:j}}(x|y)$$

for all outcomes x, y and all indices $i > 1$, $j > 1$. For these types of Markov chains, we drop the indices and simply refer to the conditional PDF

$$p_{\mathcal{X}_{i-1:i}}(x|y) = p_{\mathcal{X}}(y \to x)$$

as the *transition density*.

Equilibrium distributions. Some time homogeneous Markov chains have a *equilibrium distribution*. Intuitively, for any outcomes x_1 and x_N the we expect the PDF of the equilibrium distribution to be such that the conditional PDF

$$p_{\mathcal{X}_{1:N}}(x_N|x_1) \to p_{\mathcal{X}}(x_N)$$

as N tends to infinity. Once a Markov chain has reached its equilibrium distributions, it also stays there. More precisely,

$$\int_{\mathbb{X}} p_{\mathcal{X}}(x)p_{\mathcal{X}}(x \to y)\,\mathrm{d}\mu(x) = p_{\mathcal{X}}(y)$$

for all outcomes y.

The Metropolis-Hastings method. *The Metropolis-Hastings method* enables us to sample outcomes from a variable $\mathcal{X} : \Omega_{\mathcal{X}} \to A$, such that the PDF $p_{\mathcal{X}}$ is proportional to any positive function $f : A \to [0,\infty]$. In other words, for every $x \in A$

$$p_{\mathcal{X}}(x) = \frac{f(x)}{C}$$

for some constant $C > 0$. The method was first introduced in the context of computational physics by Metropolis et al. [1953] and was later extended to the general case by Hastings [1970].

The idea is to construct a time homogeneous Markov chain whose equilibrium distribution is that of $\mathcal{X}$. A simple way to achieve this is to ensure that the chain is

both *irreducible* and that it is *reversible* for the distribution of $\mathcal{X}$ [Hastings, 1970, Section 2.2]. A sufficient condition for a Markov chain to be irreducible is that there is a non-zero probability density of reaching any outcome $x \in A$ from another $y \in A$ in a finite amount of transitions. A Markov chain is reversible for the distribution of $\mathcal{X}$ if it satisfies the detailed balance equation

$$p_{\mathcal{X}}(y)p_{\mathcal{X}}(y \to x) = p_{\mathcal{X}}(x)p_{\mathcal{X}}(x \to y)$$

for all outcomes x and y.

We then sample outcomes from $\mathcal{X}$ by performing a random walk using a chain that fulfils the requirements above. Note that such outcomes are not independent and the error bounds we derived in Chapter 2.3 do not hold. In fact, practical convergence bounds for Markov chain Monte Carlo algorithms are still under active research. Cowles and Carlin [1996] compare common heuristics that indicate whether a Markov chain Monte Carlo algorithm has converged or not.

The key insight of the sampling method is to factor the transition density into a proposal density $k(y \to x)$ and an acceptance probability $a(y \to x)$. Formally,

$$p_{\mathcal{X}}(y \to x) = k(y \to x)a(y \to x),$$

where $k(y \to x)$ is usually the PDF of a random variable $\mathbb{P}(y) : \Omega_{\mathbb{P}(y)} \to X$ that we are able to sample outcomes from. In order to guarantee that the equilibrium distribution of the Markov chain is that of $\mathcal{X}$, we see that

$$\begin{aligned} & p_{\mathcal{X}}(y)k(y \to x)a(y \to x) = p_{\mathcal{X}}(x)k(x \to y)a(x \to y) \\ \Leftrightarrow \quad & f(y)k(y \to x)a(y \to x) = f(x)k(x \to y)a(x \to y) \end{aligned}$$

must hold true for all x and y in A. A common way to achieve this is to define the acceptance probability as

$$a(y \to x) = \min\left(1, \frac{f(x)k(x \to y)}{f(y)k(y \to x)}\right)$$

[Kalos and Whitlock, 1986, Chapter 3.7]. We must still manually verify that the resulting Markov chain is irreducible. In practice this means that the support of the proposal density should be large enough to bridge any regions in A where f is zero.

We can use Algorithm 2 to sample outcomes from $\mathcal{X}$. The idea is to perform a random walk starting from any element $x \in A$. At each iteration we generate a candidate outcome x' from a random variable $\mathbb{P}(x)$ and use $a(x \to x')$ to decide whether we stay in place or move to x'. Note that we always record a new outcome at every iteration, even if the move was not accepted.

Algorithm 2 The Metropolis-Hastings method

$x \leftarrow$ any element in A
for i = 1 to total amount of samples **do**
  $x' \leftarrow \text{OUTCOMEFROM}(\mathcal{P}(x))$
  $u \leftarrow \text{OUTCOMEFROM}(\mathcal{U}_{[0,1]})$
  if $u \leq \min\left(1, \frac{f(x')p_{\mathbb{P}(x')}(x)}{f(x)p_{\mathbb{P}(x)}(x')}\right)$ **then**
    $x \leftarrow x'$
  $\text{RECORDOUTCOME}(x)$

3 Sequential Monte Carlo

Suppose that we want to sample outcomes from a sequence of random variables $(\mathcal{X}_1, \mathcal{X}_2, \dots)$, where $\mathcal{X}_i$ and $\mathcal{X}_j$ are not necessarily identically distributed if $i \neq j$. This is useful for modelling or simulating a process that changes over time, e.g. tracking a moving object or, in our case, rendering animations under changing illumination. The idea of *sequential Monte Carlo (SMC)* methods is to enable us to sample a set of initial outcomes from $\mathcal{X}_1$ and then at each iteration i evolve the outcomes so their distribution matches that of $\mathcal{X}_{i+1}$. These methods were initially developed in the context of *filtering*, where one tries to reconstruct a signal from noisy measurements. For this purpose, the outcomes $(x_{1i}, \dots, x_{Ni})$ from iteration i are commonly used to estimate expected values

$$E[f(\mathcal{X}_i)] \approx \frac{1}{N}\sum_{j=1}^{N} f(x_{ij})$$

for some function f. The PDF $p_{\mathcal{X}_i}$ seldom occurs in the expected values and doesn't need a closed-form expression.

We will focus on SMC methods in the context of estimating integrals. Suppose we want to form estimates for integrals of the form

$$I_i = \int_{A_i} f_i(x)\,\mathrm{d}\mu(x),$$

where μ is a measure on the measurable space $\mathbb{X}$, $f_i : \mathbb{X} \to [0, \infty]$ and $A_i \subset \mathbb{X}$ is measurable. Note that both f_i and A_i are subject to change for different i. In order to form Monte Carlo estimates

$$\hat{I}_i = \frac{1}{N}\sum_{j=1}^{N} \frac{f_i(x_{ij})}{p_{\mathcal{X}_i}(x_{ij})},$$

for the integrals $(I_1, I_2, \dots)$ we have to solve two problems. First, we need to be able to sample outcomes from a sequence of random variables $(\mathcal{X}_1, \mathcal{X}_2, \dots)$, such that $f_i(x) = 0$ whenever $p_{\mathcal{X}_i}(x) = 0$ for all $x \in A_i$. Second, we need to be able to evaluate the PDF $p_{\mathcal{X}_i}(x)$ for every outcome $x \in A_i$. We can freely choose how $(\mathcal{X}_1, \mathcal{X}_2, \dots)$ are distributed but recall from Chapter 2.4 that we ideally want $p_{\mathcal{X}_i}$ to be as close to f_i as possible.

Del Moral et al. [2006] present a general framework for SMC methods where the PDF is of interest and propose several solutions, some which we will cover in this chapter. *Sequential Monte Carlo Methods in Practice* by Doucet et al. [2001] is a good in-depth reference for SMC methods in the context of filtering.

3.1 Markov chain sequential Monte Carlo methods

The unbiased methods presented above are very restrictive and will not be suitable for importance sampling if the integrands f_i are complicated. Recall that we can use the Metropolis-Hastings method to generate outcomes from a variable whose PDF is proportional to f_i. Here we extend it to the sequential setting.

Let $(x_{11}, \dots, x_{1N})$ be outcomes that we've sampled from $\mathcal{X}_1$ at iteration 1. At every remaining iteration i, we want to evolve the samples using a Markov Chain Monte Carlo method such that the PDF $p_{\mathcal{X}_i}$ is as close to f_i as possible. In contrast to traditional Markov chain Monte Carlo methods, we use one Markov chain per outcome and only concern ourselves with the current position of the chain, not its entire history. In order to adapt the samples to changes in the integrand, we have to ensure that the transition density $k_i(x \to y)$ at iteration i is such that the PDF of its equilibrium distribution is proportional to f_i. If we use the Metropolis-Hastings method, this means that we use f_i to calculate the acceptance probability.

As f_i changes this sampling method is known to degenerate the quality of the set of outcomes [Doucet et al., 2001, Chapter 1.3.3]. In other words, as i increases more and more outcomes tend to be in regions where f_i is zero. This issue is solved by occasionally performing *weighted resampling with replacement* on the outcomes, a process where each outcome x_j is given the weight

$$w_j = \frac{f_i(x_j)}{p_{\mathcal{X}_i}(x_j)}.$$

The new set of outcomes $(x'_1, \dots, x'_N)$ is then generated by sampling N times from the old set of outcomes $(x_1, \dots, x_N)$, such that the probability mass of selecting an

old outcome x_j is

$$p_{\mathcal{X}'}(x_j) = \frac{w_j}{\sum_{k=1}^{N} w_k}.$$

In practice, this gets rid of all the outcomes where f_i is zero and replaces them with duplicates of outcomes that had larger weights. It can be shown that afterwards the probability density of an outcome x' is approximately proportional to $f_i(x')$ [Smith and Gelfand, 1992].

Density estimation. If f_i does not change with i, each Markov chain will eventually reach its equilibrium distribution. In other words, for large enough j the PDF of $\mathcal{X}_j$ will be proportional to f_j. However, if f_i changes with i we have to evaluate the PDF of $\mathcal{X}_i$ as the marginal of $\mathcal{X}_{1:i}$. More precisely, if we haven't performed resampling, we can express the PDF of $\mathcal{X}_i$ with the recursive formula

$$p_{\mathcal{X}_i}(x) = \int_{\mathbb{X}} p_{\mathcal{X}_{i-1}}(y) k_i(y \to x) \, \mathrm{d}\mu(y)$$

for all $x \in \mathbb{X}$. As a consequence, we can seldom find a closed-form expression for $p_{\mathcal{X}_i}$ and are instead forced to look for ways to estimate it.

3.2 Non-parametric density estimation

One way to estimate the PDF of $\mathcal{X}_i$ is to employ *non-parametric density estimation*. Unlike Monte Carlo integration-based density estimation [Del Moral et al., 2006], which requires the PDF to be expressed as an integral, these methods work directly on outcomes from $\mathcal{X}_i$ and need no other information about $p_{\mathcal{X}_i}$. These methods are commonly used to analyse experimental data but we employ them for Monte Carlo integration.

Consistent methods. A motivating example is the variance reduction technique developed by Yakowitz et al. [1978]. Given the integral

$$I = \int_{[0,1]^d} f(x) \, \mathrm{d}\mu(x)$$

where μ is the Lebesgue measure on $\mathbb{R}^d$ and $f : [0,1]^d \to \mathbb{R}$, they develop a Monte Carlo estimator

$$\hat{I} = \sum_{i=1}^{N} w_i f(x_i)$$

where the values of w_i only depend on the outcomes $(x_1, \ldots, x_N)$. If we interpret this technique as a density estimation method, the estimated PDF would be

$$\hat{p}_{\mathcal{X}}(x_i) = \frac{1}{Nw_i}$$

which allows us to express the estimator in the familiar form

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{\hat{p}_{\mathcal{X}}(x_i)}.$$

Yakowitz et al. show that $\hat{I}$ is consistent if f is continuous and has continuous partial second derivatives. If $d \leq 2$ $\hat{I}$ also attains asymptotically lower variance than a conventional Monte Carlo estimator. However, the opposite holds true for domains of higher dimensionality and they suggest that it will be difficult to design a weighting scheme that results in rapid convergence for such domains.

K-nearest neighbours. Consider a collection of i.i.d. outcomes $(x_1, \ldots, x_N)$ from a random variable $\mathcal{X} : \Omega_{\mathcal{X}} \to \mathbb{R}^n$. Many non-parametric density estimation rely on the insight that we can approximate the probability density in a measurable region $A \subset \mathbb{R}^n$ as

$$\Phi(A) = \frac{\text{number of } x_i \in A}{N} \mu(A)^{-1},$$

where μ is the Lebesgue measure on $\mathbb{R}^n$. The *k-nearest neighbour (KNN)* method estimates the probability density for a point $x' \in \mathbb{X}$ as

$$\hat{p}_{\mathcal{X}}(x') = \Phi(B_k(x')),$$

where $B_k(x')$ is the smallest n-ball centred on x' that contains k outcomes from $\mathcal{X}$. Loftsgaarden and Quesenberry [1965] prove that $\hat{p}_{\mathcal{X}}(x')$ is a consistent estimate for $p_{\mathcal{X}}(x')$ if

$$\lim_{N \to \infty} k = \infty,$$
$$\lim_{N \to \infty} \frac{k}{N} = \infty$$

and $p_{\mathcal{X}}$ is continuous at x'.

In practice, we compute $\hat{p}_{\mathcal{X}}(x')$ by finding the distance r to the k-th nearest outcome to x'. Now

$$B_k(x') = \{x \in \mathbb{R}^n \mid \|x - x'\| < r\}$$

and we can evaluate $\mu(B_k(x'))$ using the closed-form expression for the volume of an n-ball. In other words,

$$\mu(B_k(x')) = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} r^n,$$

where Γ is the gamma function. We now write the probability density estimate as

$$\hat{p}_{\mathcal{X}}(x') = \frac{k}{N} \frac{\Gamma(\frac{n}{2}+1)}{\pi^{\frac{n}{2}} r^n}.$$

While KNN density estimation can be efficiently implemented in low-dimensional spaces, it suffers from the *curse of dimensionality* where the either the time requirement or the space requirement of the algorithm is exponential in the dimensionality of the space [Meiser, 1993].

Heuristic sampling. These density estimation methods enable us to estimate the probability densities for any collection of values $(x_1, \ldots, x_N)$ as if they were outcomes from some random variable $\mathcal{X}$. This has led to SMC methods where the outcomes are generated using heuristics and the PDF is estimated [Hämäläinen et al., 2006, 2014, Laine et al., 2007]. To the extent of our knowledge, no error bounds or consistency proofs have been derived for such methods and we have to rely on experimental validation.

Algorithm 3 Heuristic sequential Monte Carlo integration

1: $(x_{0,1}, \ldots, x_{0,N}) \leftarrow$ initial outcomes
2: **for** i = 1 to ∞ **do**
3: $\quad (x_{i,1}, \ldots, x_{i,N}) \leftarrow \textsc{HeuristicSampling}(f_i, (x_{i-1,1}, \ldots, x_{i-1,N}))$
4: $\quad (\hat{p}(x_{i,1}), \ldots, \hat{p}(x_{i,N})) \leftarrow \textsc{EstimateDensities}((x_{i,1}, \ldots, x_{i,N}))$
5: $\quad \hat{I}_i \leftarrow \frac{1}{N} \sum_{j=1}^{N} \frac{f_i(x_{i,j})}{\hat{p}_{\mathcal{X}}(x_{i,j})}$

We introduce a general framework against which these heuristic SMC methods can be contrasted. Algorithm 3 outlines how to use heuristic sampling methods and non-parametric density estimation to approximate the integrals $(I_1, I_2, \ldots)$ presented earlier. The initial outcomes at line 1 can be generated using any method, for example from a low-discrepancy sequence. The heuristic sampling method at line 3 depends on the application, but may depend on both the outcomes from the last iteration and the target current integrand. On line 4 we can estimate the densities using any non-parametric density estimation method.

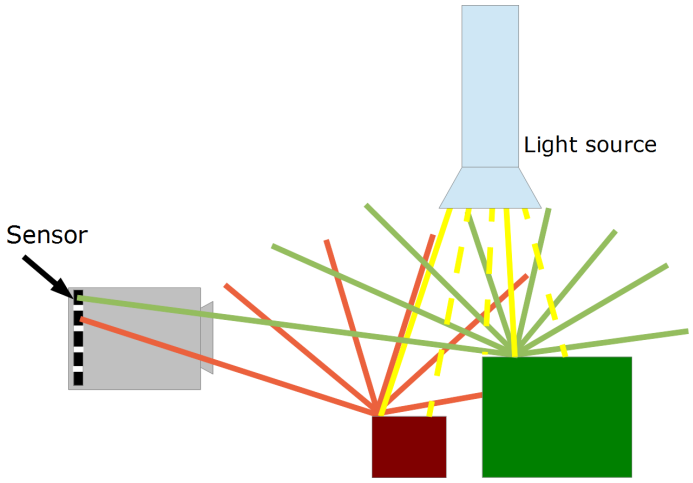


Figure 5: An example of how digital image is formed by light that falls on a grid sensors.

This algorithm can be used when rendering animations with indirect illumination. In this setting, integral I_i represent a frame in the animation. Since subtle errors in the illumination are easier to spot if they change between frames, we want the error to be as consistent as possible between frames. In order to achieve this, we need a sampling heuristic that moves as few outcomes as possible between iterations. In Chapter 7.4 we review such an algorithm by Laine et al. [2007]. We extend the algorithm to a more general case in Chapter 9.

Hämäläinen et al. [2014] make use of SMC with heuristic sampling to synthesize physically valid character motion in a dynamic environment. At its core, this problem involves tracking local maxima of a sequence of target functions $(f_1, \ldots, f_N)$. As the quality of character motion is subjective, Hämäläinen et al. do not have to find the global maximum of f_i and can use heuristic sampling methods to inject domain specific knowledge. They estimate the probability densities using a data structure that subdivides the domain. The same data structure is also used to drive the sampling process.

4 Light transport theory

At its core, rendering involves finding out how much light falls on a sensor. We can evaluate the reading of the sensor by simulating how light from the light sources interact with the scene before reaching the sensor, see Figure 5 for an example. We construct a digital image from the readings from a grid of sensors. More precisely,

each *pixel* in the image represents the reading from a sensor in the grid. In order to perform this simulation we need to be able to represent the scene and the light sources mathematically. We also need to have a model for how light interacts with the scene.

In order to achieve physically based rendering, we naturally turn to physical models of light. The most complete physical model for light and how interacts with matter, or in our case the scene, is given by *quantum electrodynamics*. This model is usually considered too detailed for rendering typical computer graphics scenes and simpler models are used in its stead. Here we use the *geometric optics* model, which can be summarized using the following assumptions:

1. Objects in the scene can emit, reflect, transmit and/or absorb light.
2. Light travels in straight lines in a medium with a constant index of refraction.
3. Light moves with infinite speed. In other words, light reaches its destination instantly.
4. Light is not affected by external factors such as magnetic fields.

Beyond the assumptions made by geometric optics, we also assume that light does not interact with the medium it travels through. As a consequence, emission, reflection, transmission and absorption can only happen at the surfaces of the objects in the scene.

Due to its simplicity, we can't use the geometrics optics model to simulate all the lighting effects we are able to perceive in the real world. A comparison of the different mathematical models for light and the effects they are able to simulate can be found in Chapter 1.5 of *Robust Monte Carlo Methods for Light Transport Simulation* by Veach [1998]. Refer to *QED: The Strange Theory of Light and Matter* by Feynman [2006] for an intuitive introduction to quantum electrodynamics.

This chapter is largely based on Chapters 3 and 8 in *Robust Monte Carlo Methods for Light Transport Simulation* by Veach [1998], which provide a measure-theoretic treatment of light transport theory. For a introduction to light transport theory, refer to Chapter 2 in the second edition of *Advanced Global Illumination* by Dutre et al. [2006].

4.1 Domains and measures

Before we can develop a model for how light interacts with the scene, we need to define what a scene is. We also need a formal way to describe a set of directions.

The scene. Similarly to Veach [1998, Chapter 3.1] we represent the scene $\mathcal{M} \subset \mathbb{R}^3$ as the finite union

$$\mathcal{M} = \bigcup_{i=1}^{N} \mathcal{S}_i,$$

where each $\mathcal{S}_i$ is a *surface*. Formally, a surface is a piecewise differentiable closed two-dimensional manifold. For our purposes it is sufficient to consider a surface as a mesh of polygons that form tight seals on any shared edges. Note that this definition does not permit any solid objects, as the scene always consists of two-dimensional surfaces. In practice, we achieve the illusion of solid objects by using a their shells when defining the scene.

We define the *surface area measure* A on $\mathbb{R}^3$ to be the two-dimensional Hausdorff measure $\mathcal{H}^2$ [Morgan, 2008, Chapter 2.3][1]. We can use A to measure the area of any surface patch in $\mathcal{M}$, but more importantly, we use it to express integrals of the form

$$\int_{\mathcal{M}} f(\mathbf{x}) \, \mathrm{d}A(\mathbf{x}) = \int_{\mathcal{M}} f(\mathbf{x}) \, \mathrm{d}\mathcal{H}^2(\mathbf{x})$$

that integrate a function f over all the surfaces in the scene.

Directions. We represent directions as unit vectors $\omega \in \mathbb{S}^2$, where $\mathbb{S}^2$ is the unit sphere $\mathbb{R}^3$. We express the direction of a point $\mathbf{x} \in \mathcal{M}$ as seen from $\mathbf{x}' \in \mathcal{M}$ using the notation

$$\omega(\mathbf{x}' \to \mathbf{x}) = \frac{\mathbf{x}' - \mathbf{x}}{\|\mathbf{x}' - \mathbf{x}\|}.$$

Since each surface is differentiable, we can derive a surface normal $N(\mathbf{x})$ for any point $\mathbf{x} \in \mathcal{M}$. We use this normal to define the set $H^+(\mathbf{x})$ of directions above the surface as

$$H^+(\mathbf{x}) = \{\omega \in \mathbb{S}^2 \mid \omega \cdot N(\mathbf{x}) > 0\}.$$

Similarly, the set of directions below the surface

$$H^-(\mathbf{x}) = \{\omega \in \mathbb{S}^2 \mid \omega \cdot N(\mathbf{x}) < 0\}.$$

[1] See Appendix A.1 for details.

Geometrically, these sets are the hemispheres resulting from slicing the unit sphere centred on $\mathbf{x}$ in half using the tangent plane defined by $N(\mathbf{x})$. We refer to $H^+(\mathbf{x})$ as the upper hemisphere and to $H^-(\mathbf{x})$ as the lower hemisphere.

We extend the concept of an angle to a set of directions $\Omega \subset \mathbb{S}^2$. The *solid angle* of Ω is expressed in *steradians (sr)* and is defined to be the area of Ω on the sphere. Formally, we measure solid angles using $\mathcal{H}^2$ on $\mathbb{S}^2$ and use the notation

$$\sigma(\Omega) = \mathcal{H}^2(\Omega).$$

As we can uniquely define a direction using two points, it is often useful to consider the solid angle *subtended* by a set $D \in \mathcal{M}$ as seen from a point $\mathbf{x} \in \mathbb{R}^3$. In this case, we measure the solid angle using σ on the projection Ω of D onto the unit sphere centred on $\mathbf{x}$. Provided that every point in D is visible to $\mathbf{x}$, Dutre et al. [2006, Appendix B.4] establish the relationship

$$\int_D f(\omega)\,\mathrm{d}\sigma(\omega) = \int_\Omega f(\omega(\mathbf{x}' \to \mathbf{x}))\frac{|N(\mathbf{x}) \cdot \omega(\mathbf{x}' \to \mathbf{x})|}{\|\mathbf{x}' - \mathbf{x}\|^2}\,\mathrm{d}A(\mathbf{x}'),$$

which allows us to convert between the two representations of the directions in Ω using the Radon-Nikodym derivative[2]

$$\frac{\mathrm{d}A(\mathbf{x}')}{\mathrm{d}\sigma(\omega)} = \frac{|N(\mathbf{x}) \cdot \omega(\mathbf{x}' \to \mathbf{x})|}{\|\mathbf{x}' - \mathbf{x}\|^2}.$$

Consider the case where Ω is a set of directions at a point $\mathbf{x} \in \mathcal{M}$, such that either $\Omega \subset H^+(\mathbf{x})$ or $\Omega \subset H^-(\mathbf{x})$, but not both. In this case, we often talk about the *projected solid angle* of Ω. Intuitively, the projected solid angle is the area we get if we project Ω onto the tangent plane. Formally, we define a measure for the projected solid angle of Ω as

$$\sigma^\perp_{\mathbf{x}}(\Omega) = \int_\Omega |\omega \cdot N(\mathbf{x})|\,\mathrm{d}\sigma(\omega).$$

From this we see that

$$\frac{\mathrm{d}\sigma(\omega)}{\mathrm{d}\sigma^\perp_{\mathbf{x}}(\omega)} = |\omega \cdot N(\mathbf{x})|.$$

Similarly, given a direction ω the projected area of a region $D \in \mathcal{M}$ is the area we get if we project D to a plane perpendicular to ω. Formally, we define a measure for the projected area as

$$A^\perp_\omega(D) = \int_D |\omega \cdot N(\mathbf{x})|\,\mathrm{d}A(\mathbf{x}),$$

[2] A measure-theoretic generalization of a derivative. See Appendix A.1 for details.

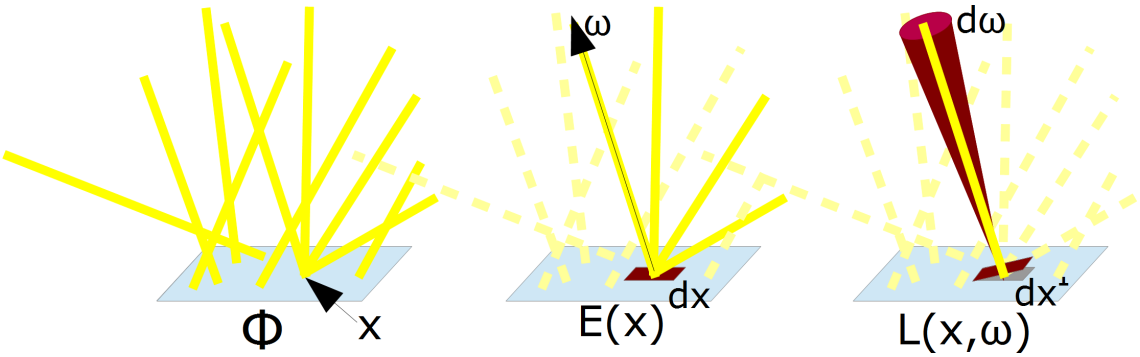


Figure 6: The relationship between radiometric quantities. Here, the Φ measures radiant power of the stream of photons that pass through the surface. The irradiance $E(x)$ measures how much power passes through an infinitesimal patch dx around the point x. The radiance $L(x, \omega)$ restricts this even further and measures the power coming from photons that pass through $dx^{\perp}$ (the projection of dx that is perpendicular to ω) and whose trajectories are contained within the infinitesimal cone $d\omega$ around the direction ω.

which implies

$$\frac{\mathrm{d}A(\mathbf{x})}{\mathrm{d}A^{\perp}_{\omega}(\mathbf{x})} = |\omega \cdot N(\mathbf{x})|.$$

4.2 Radiometric quantities

We express light transport theory in terms of *radiometric quantities*, which were developed to measure electromagnetic radiation. A special case of these quantities are the *photometric quantities*, which are specific to the visible spectrum of electromagnetic radiation. Here we only cover the more general radiometric quantities, see Figure 6 for an overview.

Radiant energy and power. In quantum electrodynamics light is made up of *photons*, tiny particles that carry a small amount of energy expressed in Joules. The energy of a photon depends on its wavelength, which also defines its colour. For a given region of the scene and a given time period, it is in theory possible to calculate how much *radiant energy* Q it has received by simply summing up the energies carried by the photons that intersected with it.

In geometric optics, however, we have no such concept as a photon. Instead we assume that light is continuous in nature and model them with *light rays*. A light ray is uniquely defined by a point of origin on a surface in the scene, and a direction

vector. In other words, a light ray is an element in *ray space*

$$\mathcal{R} = \mathcal{M} \times \mathbb{S}^2.$$

Veach [1998, Appendix 3.B.] shows that any function which plausibly models the energy carried by photons is also a well defined measure on

$$\mathbb{R} \times \mathcal{R},$$

where $\mathbb{R}$ represents time. As a consequence, we consider Q to be a measure.

Radiant power Φ is the flow rate of radiant energy over time which we express in Watts (J/s). Formally,

$$\Phi = \frac{\mathrm{d}Q(t)}{\mathrm{d}\mu(t)},$$

where we use the Lebesgue measure μ to measure time in $\mathbb{R}$. For example, we can use Φ express the strength of a light bulb. Note that this does not specify the size of the light bulb or the directionality of the light it emits.

Irradiance and radiosity. *Irradiance* E and *radiosity* B both express radiant power per unit area. Irradiance describes how much radiant power arrives at a point while radiosity describes how much radiant power leaves from a point.

Formally,

$$E(\mathbf{x}) = \frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}A(\mathbf{x})} = B(\mathbf{x})$$

which we express in Watts per square meter (W/m^2).

We often make use of E and B when the light is not directional. For example, if every point on the light bulb evenly emits light in every direction, then $B(\mathbf{x})$ tells us exactly how bright the point $\mathbf{x}$ on the light bulb appears.

Radiance. In order to represent light that is directional we need *radiance* $L(\mathbf{x}, \omega)$, where $\mathbf{x}$ is a point on a surface and ω is a direction. Informally, $L(\mathbf{x}, \omega)$ tells us how much radiant power a small surface patch centred on x receives from (or emits to) a small solid angle centred on ω. Formally, radiance is radiant power per unit solid angle per unit area perpendicular to ω. In other words,

$$L(\mathbf{x}, \omega) = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \omega)}{\mathrm{d}A^{\perp}_{\omega}(\mathbf{x})\,\mathrm{d}\sigma(\omega)}$$

and we express it in Watts per square meter per steradian ($W/(m^2 sr)$). Intuitively, the projected area allows us to account for the fact that the solid angle spreads out if it hits the surface at grazing angles.

It is often useful to rewrite radiance with respect to projected solid angle. By applying the Radon-Nikodym derivates mentioned above, we see that

$$L(\mathbf{x}, \omega) = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \omega)}{|\omega \cdot N(\mathbf{x})| \, \mathrm{d}A(\mathbf{x}) \, \mathrm{d}\sigma(\omega)} = \frac{\mathrm{d}^2\Phi(\mathbf{x}, \omega)}{\mathrm{d}A(\mathbf{x}) \, \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)}.$$

Similarly to irradiance and radiosity, we also make the distinction between *incident* and *exitant* radiance. The incident radiance $L^{\leftarrow}(\mathbf{x}, \omega)$ measures how much radiance is arriving at x from the direction ω. Conversely, exitant radiance $L^{\rightarrow}(\mathbf{x}, \omega)$ measures how much radiance leaves from $\mathbf{x}$ in the direction ω. In order to simplify computations we consider $\omega \in H^+(\mathbf{x})$ to be oriented away from $\mathbf{x}$ in both cases. For example, if $\mathbf{x}' \in \mathcal{M}$ is the point closest to $\mathbf{x} \in \mathcal{M}$ along the direction ω, we see that

$$L^{\leftarrow}(\mathbf{x}, \omega) = L^{\rightarrow}(\mathbf{x}', -\omega)$$

follows from our assumption that light does not interact with the medium it travels trough. In other words, *radiance remains constant along straight lines.*

Representing colours. The definitions above do not distinguish between photons of different wavelengths and can not model coloured light as such. This can be modelled using *spectral radiance* $L_\lambda(\mathbf{x}, \omega)$, which is radiance per unit wavelength. More precisely,

$$L_\lambda(\mathbf{x}, \omega) = \frac{\mathrm{d}L(\mathbf{x}, \omega)}{\mathrm{d}\mu(\lambda)}$$

where the wavelength λ is a positive real number and μ is the Lebesgue measure on $\mathbb{R}$.

In computer graphics it is sufficient if our model is able to represent the colours a human eye can perceive. This can be achieved by only considering the spectral radiance for a finite amount of wavelengths [Akenine-Möller et al., 2008, Chapter 7.3], although some effects such as dispersion through a prism cannot be modelled without considering the full spectrum. Commonly only three wavelengths are used which correspond to the colours red, green and blue (RGB). In the remainder of this thesis we often use the term radiance $L(x, \omega)$, when in fact we are referring to a vector

$$[L_{\lambda r}(\mathbf{x}, \omega), L_{\lambda g}(\mathbf{x}, \omega), L_{\lambda b}(\mathbf{x}, \omega)]^T$$

of RGB spectral radiances.

4.3 Surface interactions

Because of our assumptions, light interactions only happen at the surfaces of the scene. With the help of the radiometric quantities we now form a mathematical model for these interactions.

Emission. We say that a region in the scene emits light if generates radiant energy and we consider these regions to be our light sources. Formally, we model emission as a function [3]

$$L_e^{\rightarrow} : \mathcal{M} \times \mathbb{S}^2 \to [0, \infty],$$

which allows us to express how much radiance any point in the scene emits in any direction.

In this thesis we only work with *diffuse* emitters. A diffuse emitter is a region $D \in \mathcal{M}$ such that

$$L_e^{\rightarrow}(\mathbf{x}, \omega) = \begin{cases} L(\mathbf{x}) & \text{if } \omega \in H^+(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases}$$

for every point $\mathbf{x} \in D$, where $L(\mathbf{x})$ is some positive number. In other words, a diffuse emitter is a region of points that emit light evenly over their outgoing hemispheres.

In this thesis we use two different types of diffuse emitters to represent our light sources: *point lights* and *area lights*. A point light is a diffuse emitter where D is a singleton set consisting of a single point $\mathbf{x} \in \mathcal{M}$. An area light is a diffuse emitter which emits the same radiance from every point. That is, for every point $\mathbf{x}$ in D we let $L(\mathbf{x}) = L_D$ for some positive number L_D.

Reflectance and absorption. For opaque surfaces that do not transmit light it is useful to model reflection and absorption using the same mathematical concept. Nicodemus [1965] introduced the *bidirectional reflectance distribution function (BRDF)* $f_{\mathbf{x}}^r$, which represents these two phenomena for a point $\mathbf{x} \in \mathcal{M}$. Specifically, $f_{\mathbf{x}}^r$ is the ratio between the differential exitant radiance and the differential irradiance

$$\mathrm{d}E(\mathbf{x}, \omega_i) = L^{\leftarrow}(\mathbf{x}, \omega_i)\, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega_i)$$

caused by a differential projected solid angle centred on a direction ω_i. In other words, we can relate the incident radiance from ω_i and differential exitant radiance

[3] If we want to represent colours a more accurate definition would be $L_e^{\rightarrow} : \mathcal{M} \times \mathbb{S}^2 \to [0, \infty]^3$, which expresses emission in terms of RGB spectral radiances.

towards ω_o by

$$\mathrm{d}L^{\rightarrow}(x, \omega_o) = f^r_{\mathbf{x}}(\omega_i, \omega_o) L^{\leftarrow}(\mathbf{x}, \omega_i) \, \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega_i).$$

We form an expression for the total exitant radiance $L^{\rightarrow}(x, \omega_o)$ by integrating this expression over the hemisphere. This results in the *reflectance equation*

$$L^{\rightarrow}(\mathbf{x}, \omega_o) = \int_{H^+(\mathbf{x})} f^r_{\mathbf{x}}(\omega_i, \omega_o) L^{\leftarrow}(\mathbf{x}, \omega_i) \, \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega_i).$$

[Nicodemus, 1965]. Since the BRDF is used to model reflectances, both ω_i and ω_o are in the same hemisphere $H^+(\mathbf{x})$.

Transmittance. Transmission can be modelled the same way as reflectance using a *bidirectional transmission distribution function (BTDF)*. In this case the outgoing ω_o is always in the hemisphere which is opposite of the one ω_i is in. Sometimes it is useful to combine the BRDF and the BTDF into a single function that describes all the possible scattering events. This combined functions is often referred to as the *bidirectional scattering distribution function (BSDF)*. We will not discuss transmission in further detail here, but refer the interested reader to Chapter 3.6 in *Robust Monte Carlo Methods for Light Transport Simulation* by Veach [1998].

Properties of the BRDF. Physically plausible BRDFs should at least satisfy two basic properties. The first is *Helmholtz reciprocity*, which means that reversing the direction of the light does not change the value of the BRDF. More precisely

$$f^r_{\mathbf{x}}(\omega_a, \omega_b) = f^r_{\mathbf{x}}(\omega_b, \omega_a),$$

for every $\omega_a \in H^+(\mathbf{x})$ and $\omega_b \in H^+(\mathbf{x})$. The second is that the BRDF should be *energy conserving*. Intuitively this means that a surface should not be able to reflect more light than it receives. Formally we express this as

$$\int_{H^+(\mathbf{x})} f^r_{\mathbf{x}}(\omega_i, \omega_o) \, \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega_o) \leq 1,$$

for every $\omega_i \in H^+(x)$.

Diffuse BRDFs. A *diffuse* BRDF is used to model materials that reflect light evenly over the upper hemisphere, such as cloth or concrete. Informally, the appearance of these materials does not change when you view them from different

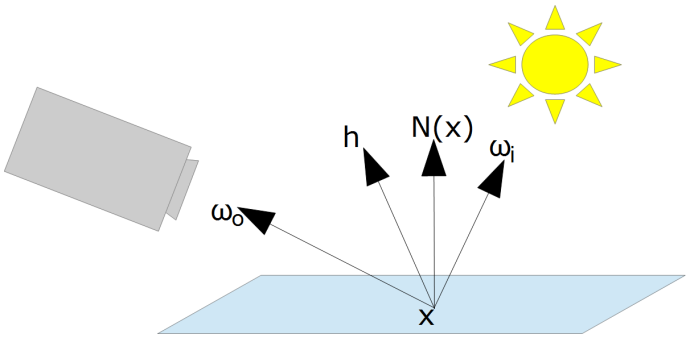


Figure 7: Overview of the vectors used in the Blinn-Phong BRDF model.

directions. Formally, a diffuse BRDF is defined as

$$f_{\mathbf{x}}^{r}(\omega_i, \omega_o) = \frac{c}{\pi},$$

for some constant $c \in [0, 1]$. Since

$$\int_{H^+(\mathbf{x})} \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega_o) = \sigma_{\mathbf{x}}^{\perp}(\mathcal{H}^+(\mathbf{x})) = \pi,$$

we divide by π to ensure that the BRDF is energy conserving.

Blinn-Phong BRDF. Many physical materials cannot be accurately modelled using a diffuse BRDF as their appearance does change when you view them from different directions. One important contributing factor for this phenomenon is that the highlights change with the viewing direction. This becomes evident when you consider the fact that a highlight is a distortion of a perfect reflection. Because of this glossy BRDFs, that are able simulate highlights, can model a wider range of physical materials than diffuse BRDFs.

The *Blinn-Phong* BRDF is a glossy BRDF that was first presented by Blinn [1977]. Here we use an energy conserving form of the BRDF which can be found in Chapter 7.6 of the third edition of *Real-Time Rendering* [Akenine-Möller et al., 2008]. Formally, we define the BRDF as

$$f_{\mathbf{x}}^{r}(\omega_i, \omega_o) = c_s \frac{s+8}{8\pi} \max(0, h \cdot N(\mathbf{x}))^s + \frac{c_d}{\pi}$$

where

$$h = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}$$

is the *half vector* that defines where the surface normal should be to perfectly reflect light between ω_o and ω_i, see Figure 7. We see that this BRDF is in fact a linear

combination of a diffuse component and a component that models the glossy highlight. The relative strengths of these components are determined by the constants $c_s \in [0,1]$ and $c_d \in [0,1]$ that satisfy $c_s + c_d \leq 1$.

We gain intuition for the glossy component by noting that the dot product determines where the highlights appear as it measures how similar the surface normal is to the half vector. In turn, the exponent s determines how large the highlights are.

Microfacet theory. A physically based model for surfaces is given by *microfacet theory*, which assumes that each surface consists of a collection of small mirrors. In this setting, the value of the BRDF is the fraction of the mirrors which reflect light perfectly from the incoming direction to the outgoing direction. Cook and Torrance [1982] formulate a BRDF based on microfacet theory that is able to account for more physical phenomena than the Blinn-Phong BRDF presented here.

4.4 The rendering equation

The lighting we are able experience is always in a steady-state configuration, or equilibrium. When we interact with the scene, for example by switching a light source on or off, the lighting almost immediately reaches equilibrium since photons travels so much faster than what we can perceive. Under our assumptions equilibrium is achieved immediately after a change has been made, since all rays of light reach their destinations instantly.

We represent the equilibrium configuration using exitant radiance $L^{\rightarrow}(\mathbf{x}, \omega)$ over all points $\mathbf{x} \in \mathcal{M}$ and all directions $\omega \in \mathbb{S}^2$. We can express incident radiance in terms of exitant radiance by

$$L^{\leftarrow}(\mathbf{x}, \omega) = L^{\rightarrow}(r(\mathbf{x}, \omega), -\omega)$$

where $r : \mathcal{M} \times \mathbb{S}^2 \to \mathcal{M}$,

$$r(\mathbf{x}, \omega) = \mathbf{x} + \omega \min\{t \in [0, \infty] \mid \mathbf{x} + t\omega \in \mathcal{M}\}$$

is the *ray casting* function that returns the point in $\mathcal{M}$ that is closest to $\mathbf{x}$ along the direction ω.

Hemispherical formulation. The first mathematical formulation for the rendering equation was simultaneously presented by Kajiya [1986] and Immel et al.

[1986]. This formulation expresses equilibrium radiance using the recursive integral equation

$$\begin{aligned} L^{\rightarrow}(\mathbf{x}, \omega_o) &= L_e^{\rightarrow}(\mathbf{x}, \omega_o) + \int_{H^+(\mathbf{x})} f_x^r(\omega_i, \omega_o) L^{\leftarrow}(\mathbf{x}, \omega_i) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega_i) \\ &= L_e^{\rightarrow}(\mathbf{x}, \omega_o) + \int_{H^+(\mathbf{x})} f_{\mathbf{x}}^r(\omega_i, \omega_o) L^{\rightarrow}(r(\mathbf{x}, \omega_i), -\omega_i) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega_i), \end{aligned}$$

which boils down to the statement that the exitant radiance is the sum of the reflected radiance and the emitted radiance. Note that we cannot evaluate this integral directly, as the equilibrium radiance $L^{\rightarrow}$ occurs on both the left and the right hand side.

Operator formulation. We use the reflectance equation to define the *light transport operator* $\mathcal{T}$ that maps a function $L^{\rightarrow} : \mathcal{M} \times \mathcal{S}^2 \rightarrow [0, \infty]$ describing exitant radiance in a scene to another function $\mathcal{T}L^{\rightarrow} : \mathcal{M} \times \mathcal{S}^2 \rightarrow [0, \infty]$ that models the exitant radiance caused by reflecting $L^{\rightarrow}$ off the surfaces in the scene [Veach, 1998, Chapter 4.4]. For example, if $L^{\rightarrow}$ models the direct light emitted from the light sources in the scene, then $\mathcal{T}L^{\rightarrow}$ models the resulting indirect light that reflects off the surfaces. More precisely,

$$\mathcal{T}L^{\rightarrow}(\mathbf{x}, \omega) = \int_{H^+(\mathbf{x})} f_{\mathbf{x}}^r(\omega_i, \omega) L^{\rightarrow}(r(\mathbf{x}, \omega_i), -\omega_i) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega_i).$$

Using this operator, we write the rendering equation in the compact form

$$L^{\rightarrow}(\mathbf{x}, \omega_o) = L_e^{\rightarrow}(\mathbf{x}, \omega_o) + \mathcal{T}L^{\rightarrow}(\mathbf{x}, \omega_o).$$

Area formulation. The rendering equation can also be expressed as an integral over the visible surfaces. More precisely, we express the integral above with respect to the surface area measure with the help of the Radon-Nikodym derivatives defined

in 4.1. In other words,

$$\begin{aligned} L^{\rightarrow}(x,\omega_o) &= L_e^{\rightarrow}(\mathbf{x},\omega_o) + \int\limits_{H^+(\mathbf{x})} f_x^r(\omega_i,\omega_o) L^{\leftarrow}(\mathbf{x},\omega_i)\,\mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega_i) \\ &= L_e^{\rightarrow}(\mathbf{x},\omega_o) + \int\limits_{H^+(\mathbf{x})} f_x^r(\omega_i,\omega_o) L^{\leftarrow}(\mathbf{x},\omega_i)|N(\mathbf{x})\cdot\omega_i|\,\mathrm{d}\sigma(\omega_i) \\ &= L_e^{\rightarrow}(\mathbf{x},\omega_o) + \int\limits_{\mathcal{V}_{\mathbf{x}}} f_{\mathbf{x}}^r(\omega_i,\omega_o) L^{\rightarrow}(\mathbf{x}',\omega(\mathbf{x}'\rightarrow\mathbf{x})) \\ &\qquad\qquad \frac{|N(\mathbf{x})\cdot\omega(\mathbf{x}\rightarrow\mathbf{x}')||N(\mathbf{x}')\cdot\omega(\mathbf{x}'\rightarrow\mathbf{x})|}{\|\mathbf{x}'-\mathbf{x}\|^2}\,\mathrm{d}A(\mathbf{x}') \end{aligned}$$

where $\mathcal{V}_{\mathbf{x}} \subset \mathcal{M}$ are the surfaces which are visible from x. We phrase this as an integral over all of $\mathcal{M}$ using a visibility function $V : \mathcal{M}\times\mathcal{M} \rightarrow \{0,1\}$, where

$$V(\mathbf{x}\leftrightarrow\mathbf{x}') = \begin{cases} 1 & \text{if } \mathbf{x} \text{ and } \mathbf{x}' \text{ are mutually visible} \\ 0 & \text{otherwise.} \end{cases}$$

If we combine the derivatives with the visibility function into a single *geometry term* $G(\mathbf{x}\leftrightarrow\mathbf{x}')$, where

$$G(\mathbf{x}\leftrightarrow\mathbf{x}') = V(\mathbf{x}\leftrightarrow\mathbf{x}')\frac{|N(\mathbf{x})\cdot\omega(\mathbf{x}\rightarrow\mathbf{x}')||N(\mathbf{x}')\cdot\omega(\mathbf{x}'\rightarrow\mathbf{x})|}{\|\mathbf{x}'-\mathbf{x}\|^2},$$

we are able to express the rendering equation in the more compact form

$$L^{\rightarrow}(\mathbf{x},\omega_o) = L_e^{\rightarrow}(\mathbf{x},\omega_o) + \int\limits_{\mathcal{M}} f_{\mathbf{x}}^r(\omega_i,\omega_o) L^{\rightarrow}(\mathbf{x}',\omega(\mathbf{x}'\rightarrow\mathbf{x}))G(\mathbf{x}\leftrightarrow\mathbf{x}')\,\mathrm{d}A(\mathbf{x}').$$

The measurement equation. When rendering a picture we are only interested in the equilibrium radiance that falls on one of our sensors. In fact, we only care about the sensor response to that radiance. We consider our sensors to be a part of the scene and for one of the sensors $C \subset \mathcal{M}$ we use the function

$$W : \mathcal{M}\times\mathbb{S}^2 \rightarrow [0,\infty]$$

to model how large the sensor response is to incident radiance from any direction at any point in $\mathcal{M}$. Clearly $W(\mathbf{x},\omega) = 0$ if $\mathbf{x}$ is not in C, although the exact definition of W depends on the mathematical model we choose for our camera.

We now express the sensor response to the equilibrium radiance using the *measurement equation*

$$\begin{aligned} I &= \int\limits_{\mathcal{M}\times\mathcal{S}^2} W(\mathbf{x},\omega) L^{\leftarrow}(\mathbf{x},\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x}) \\ &= \int\limits_{\mathcal{M}\times\mathcal{S}^2} W(\mathbf{x},\omega) L^{\rightarrow}(r(\mathbf{x},\omega),-\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x}). \end{aligned}$$

In fact, this is a mathematical formulation of the rendering problem. In order to render a picture we need to be able to evaluate or approximate this integral.

Neumann series. Note that we can recursively expand the measurement equation using the light transport operator. More precisely,

$$\begin{aligned} I &= \int\limits_{\mathcal{M}\times\mathcal{S}^2} W(\mathbf{x},\omega) L^{\rightarrow}(r(\mathbf{x},\omega),-\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x}) \\ &= \int\limits_{\mathcal{M}\times\mathcal{S}^2} W(\mathbf{x},\omega) L_e^{\rightarrow}(r(\mathbf{x},\omega),-\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x}) \\ &+ \int\limits_{\mathcal{M}\times\mathcal{S}^2} W(\mathbf{x},\omega) \mathcal{T} L^{\rightarrow}(r(\mathbf{x},\omega),-\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x}) \\ &= \int\limits_{\mathcal{M}\times\mathcal{S}^2} W(\mathbf{x},\omega) L_e^{\rightarrow}(r(\mathbf{x},\omega),-\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x}) \\ &+ \int\limits_{\mathcal{M}\times\mathcal{S}^2} W(\mathbf{x},\omega) \mathcal{T} L_e^{\rightarrow}(r(\mathbf{x},\omega),-\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x}) \\ &+ \int\limits_{\mathcal{M}\times\mathcal{S}^2} W(\mathbf{x},\omega) \mathcal{T}\mathcal{T} L^{\rightarrow}(r(\mathbf{x},\omega),-\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x}) \\ &= \ldots \end{aligned}$$

Veach [1998, Appendix 4.B] shows that for physical scenes, the last term in the expansion (that contains the equilibrium radiance $L^{\rightarrow}(r(\mathbf{x},\omega),-\omega)$ eventually diminishes after the light transport operator has been applied multiple times. Based on this, it can be shown that the *Neumann series* [Taylor and Lay, 1986, Chapter 4.11]

$$\sum_{i=0}^{\infty} \int\limits_{\mathcal{M}\times\mathcal{S}^2} W(\mathbf{x},\omega) \mathcal{T}^i L_e^{\rightarrow}(r(\mathbf{x},\omega),-\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x})$$

converges to the value I of the measurement equation. Intuitively, this means that you can form a physically correct picture by adding together the sensor response

to different kinds of light. That is, light that directly reaches the sensor from the light sources, light that reaches the sensor after reflecting once off the surfaces in the scene, light that reaches the sensor after reflecting twice off the surfaces, and so on.

4.5 Path integral formulation

Veach [1998, Chapter 8] presents an alternative formulation of the measurement equation that doesn't include the recursive integral from the rendering equation. Using this formulation we are able to express the response of a sensor $C \subset \mathcal{M}$ as

$$I = \int_{\mathcal{P}} f(\bar{\mathbf{x}}) \, \mathrm{d}\mathcal{A}(\bar{\mathbf{x}}),$$

where $\mathcal{P}$ is the space of *transport paths* of all lengths, $\mathcal{A}$ is a measure on $\mathcal{P}$ and f is the *measurement contribution function*. This formulation allows for a direct application of Monte Carlo integration. It also provides intuition for variance reduction techniques since it describes the light transport problem globally in contrast to the rendering equation, which only expresses it in terms of local reflections. The rest of this section follows Chapter 8 in *Robust Monte Carlo Methods for Light Transport Simulation* by Veach [1998] and derives expressions for f, $\mathcal{P}$ and $\mathcal{A}$.

Three-point formulation. The *three-point formulation* of the rendering equation does not make use of directions, instead all quantities are expressed using points in $\mathcal{M}$. In this formulation, we express radiance as

$$\begin{aligned} L(\mathbf{x} \to \mathbf{x}') &= L^{\rightarrow}(\mathbf{x}, \omega(\mathbf{x} \to \mathbf{x}')) \\ L(\mathbf{x} \leftarrow \mathbf{x}') &= L^{\leftarrow}(\mathbf{x}, \omega(\mathbf{x} \to \mathbf{x}')), \end{aligned}$$

which clearly distinguishes incident and exitant radiance using the direction of the arrow. Similarly, the BRDF can be rewritten as

$$f^r(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') = f^r_{x'}(\omega(\mathbf{x}' \to \mathbf{x}), \omega(\mathbf{x}' \to \mathbf{x}'')).$$

We express the rendering equation in the desired form by substituting the expressions above into the area formulation. In other words,

$$L(\mathbf{x}' \to \mathbf{x}'') = L_e(\mathbf{x}' \to \mathbf{x}'') + \int_{\mathcal{M}} f^r(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') L(\mathbf{x} \to \mathbf{x}') G(\mathbf{x} \leftrightarrow \mathbf{x}') \, \mathrm{d}A(\mathbf{x}).$$

We also write the measurement equation in the three-point formulation by expressing it with respect to the surface area measure. More precisely,

$$I = \int_{\mathcal{M}\times\mathcal{M}} W(\mathbf{x} \to \mathbf{x}')L(\mathbf{x} \leftarrow \mathbf{x}')G(\mathbf{x} \leftrightarrow \mathbf{x}')\,\mathrm{d}A(\mathbf{x}')\,\mathrm{d}A(\mathbf{x}),$$

where $W(\mathbf{x} \to \mathbf{x}') = W(\mathbf{x}, \omega(x \to \mathbf{x}'))$.

Path space. A transport path $\bar{\mathbf{x}}$ of length k is a k-tuple of points $(\mathbf{x}_1, \ldots, \mathbf{x}_k) \in \mathcal{M}^k$. We see that $\bar{\mathbf{x}}$ may cause a sensor response if $W(\mathbf{x}_k \to \mathbf{x}_{k-1}) > 0$, $L_e(\mathbf{x}_2 \leftarrow \mathbf{x}_1) > 0$ and all adjacent points in the path are mutually visible. Let $\mathcal{P}^k = \mathcal{M}^k$ be the set of all transport paths of length k. We define a measure $\mathcal{A}^k$ on $\mathcal{P}^k$ as the product measure

$$\mathcal{A}^k = \underbrace{A \times \cdots \times A}_{k \text{ times}}.$$

We define *path space* $\mathcal{P}$ as the countably infinite union

$$\mathcal{P} = \bigcup_{k=1}^{\infty} \mathcal{P}^k.$$

We also extend $\mathcal{A}^k$ to define the *product area measure* $\mathcal{A}$ on $\mathcal{P}$, where

$$\mathcal{A}(D) = \sum_{k=1}^{\infty} \mathcal{A}^k(\mathcal{P}^k \cap D)$$

for all measurable sets $D \subset \mathcal{P}$. Now consider the integral

$$J = \int_{\mathcal{P}} f(\bar{\mathbf{x}})\,\mathrm{d}\mathcal{A}(\bar{\mathbf{x}}).$$

Since we can express $\mathcal{P}$ as a union of disjoint sets $\mathcal{P}^k$, we see that

$$J = \int_{\bigcup_{k=1}^{\infty} \mathcal{P}^k} f(\bar{\mathbf{x}})\,\mathrm{d}\mathcal{A}(\bar{\mathbf{x}}) = \sum_{k=1}^{\infty} \int_{\mathcal{P}^k} f(\bar{\mathbf{x}})\,\mathrm{d}\mathcal{A}(\bar{\mathbf{x}}) = \sum_{k=1}^{\infty} \int_{\mathcal{P}^k} f(\bar{\mathbf{x}})\,\mathrm{d}\mathcal{A}^k(\bar{\mathbf{x}}).$$

The last step relies on the fact that $\mathcal{A}(D) = \mathcal{A}^k(D)$ for all measurable sets $D \subset \mathcal{P}^k$.

Measurement contribution function. We find an expression for the measurement contribution function by rewriting the Neumann series expansion of the measurement equation as an integral over $\mathcal{P}$ with respect to the product area measure

(a) A biased image missing indirect illumination.

(b) An unconverged image containing variance.

(c) A fully converged unbiased image.

Figure 8: A comparison of bias and variance in the context of rendering.

$\mathcal{A}$. In other words,

$$I = \sum_{k=2}^{\infty} \int_{P^k} L_e(\mathbf{x}_2 \leftarrow \mathbf{x}_1) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \prod_{i=2}^{k-1} f^r(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) W(x_k \to \mathbf{x}_{k-1}) \, \mathrm{d}\mathcal{A}^k(\mathbf{x}_1, \ldots, \mathbf{x}_k).$$

We now define the measurement contribution function f as

$$f(\bar{\mathbf{x}}) = \begin{cases} 0 & \text{if } \bar{\mathbf{x}} \in \mathcal{P}^1 \\ f^k(\bar{\mathbf{x}}) & \text{if } \bar{\mathbf{x}} \in \mathcal{P}^k \text{ for } k > 1, \end{cases}$$

where

$$f^k(\bar{\mathbf{x}}) = L_e(\mathbf{x}_2 \leftarrow \mathbf{x}_1) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \prod_{i=2}^{k-1} f^r(\mathbf{x}_{i-1} \to x_i \to \mathbf{x}_{i+1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) W(\mathbf{x}_k \to \mathbf{x}_{k-1}).$$

Using this definition of f, we now express the measurement equation in the desired form

$$I = \int_{\mathcal{P}} f(\bar{\mathbf{x}}) \, \mathrm{d}\mathcal{A}(\bar{\mathbf{x}}).$$

5 Rendering algorithms

As we mentioned in Chapter 4, rendering involves finding out how much light falls on a sensor. As the name implies, rendering algorithms simulate the light which

falls on the sensor. Mathematically, these algorithms estimate the measurement equation, which was also described in Chapter 4.

In this thesis, we focus on algorithms that employ Monte Carlo integration. In this context bias is systematic error in the image, which is perceived as missing effects such as missing indirect illumination. Variance is perceived as noise in the image. See Figure 8 for an example.

Here we introduce two well-known unbiased and consistent rendering algorithms[4]. Informally, given enough samples these algorithms produce physically correct images. For a treatment of a wider selection of unbiased rendering algorithms, refer to the second edition of *Advanced Global Illumination* by Dutre et al. [2006].

5.1 Evaluating visibility

All three formulations of the measurement equation (and consequently the rendering equation) contain a term that describes visibility. In the area and path formulations visibility is accounted for explicitly using the visibility function $V : \mathcal{M} \times \mathcal{M} \to \{0, 1\}$. The ray casting function $r : \mathcal{M} \times \mathbb{S}^2 \to \mathbb{M}$ implicitly accounts for visibility in the hemispherical formulation of the rendering equation, since

$$V(\mathbf{x} \leftrightarrow \mathbf{x}') = \begin{cases} 1 & \text{if } r(\mathbf{x}, \omega(\mathbf{x} \to \mathbf{x}')) = \mathbf{x}', \\ 0 & \text{otherwise.} \end{cases}$$

Here we describe two common approaches to algorithmically evaluate the ray casting function: *ray casting* [Appel, 1968] and *rasterisation* [Bouknight, 1970]. We limit our discussion to scenes which consist of polygonal meshes. Note that ray casting algorithms can be devised for other scene representations as well, for example implicit surfaces [Kalra and Barr, 1989] or fully volumetric representations [Levoy, 1990].

Ray casting. As the name implies, ray casting algorithms directly evaluate the ray casting function. A naive implementation the ray casting function $r(x, \omega)$ would intersect the ray against all polygons in the scene. More efficient algorithms exclude large groups of polygons from future processing using conservative intersection tests.

[4]Strictly speaking, the algorithms described in this chapter do not estimate the full measurement equation as they do not account for very direct light that travels directly from a light source to a sensor without any intermediate bounces. Even though the algorithms can be extended to account for this effect in a straightforward fashion, we choose to omit it in favour of clarity of presentation.

	Conference	San Miguel	Fairy	Sibenik
Ray casting	71.17	43.65	63.90	73.67
Rasterisation	786.74	101.23	856.16	855.86

Table 1: Comparison of hardware accelerated rasterisation and GPU-based ray casting in the context of resolving the surfaces which are visible to the camera. Both algorithms estimate this by evaluating the ray casting function at every pixel in a frame with the resolution 1920×1080. The table shows the number of frames each algorithm is able to deliver per second in four different scenes. The ray casting algorithm and scenes are both identical to the ones used by Aila et al. [2012] and the tests were performed on a NVIDIA GTX 680 GPU.

Refer to *Heuristic Ray Shooting Algorithms* by Havran [2000] for an in-depth treatment of ray casting algorithms.

As seen above, we can use a ray casting algorithm to implement the visibility function $V(\mathbf{x} \leftrightarrow \mathbf{x}')$. That said, $V(\mathbf{x} \leftrightarrow \mathbf{x}')$ is better implemented with a more efficient algorithm which is able to return 0 as soon as it finds any occluding point in $\mathcal{M}$ between x and x' along $\omega(\mathbf{x} \to \mathbf{x}')$. In contrast, if $V(\mathbf{x} \leftrightarrow \mathbf{x}')$ is implemented in terms of the ray casting function, then the resulting algorithm would unnecessarily find the closest point to $\mathbf{x}$ along $\omega(\mathbf{x} \to \mathbf{x}')$.

Rasterisation. Rasterisation is the process of converting a continuous picture into a grid of points, or pixels. Together with a hidden surface removal algorithm we can use rasterisation to estimate the ray casting function, an operation which is accelerated with tailor-made hardware in modern GPUs. See Chapter 2 and 3 of the third edition of *Real-Time Rendering* by Akenine-Möller et al. [2008] for an overview of this type of hardware-accelerated rasterisation.

Given a region of directions $\Omega \subset \mathbb{S}^2$ and a point of origin x, a rasterisation algorithm estimates the image $r(\mathbf{x}, \Omega) \subset \mathcal{M}$ of the ray casting function. This is achieved by splitting Ω into a grid of disjoint regions. All the directions within each region are assumed to map to the same point, which we find by evaluating the ray casting function for a single representative direction.

Performance and bias. Aila et al. [2012] analyse the performance characteristics of ray casting algorithms on modern GPUs. Under favourable conditions, for example when resolving the surfaces which are visible to a camera in the scene,

such algorithms can evaluate the ray casting function more than 400 million times per second. Under these conditions, ray casting is significantly slower than rasterisation. In Table 1 we compare the fastest ray casting algorithm described by Aila et al. against hardware accelerated rasterisation with the OpenGL application programming interface [Shreiner et al., 2013]. Note, however, that the performance of ray casting degrades gracefully when computing visibilities from multiple points of origin. In contrast, multiple points of origin have a more significant impact on rasterisation as the entire scene needs to be rasterised once for every point of origin.

In spite of its performance, the use of rasterisation introduces bias to rendering algorithms as it only approximates the ray casting function. As a consequence, ray casting algorithms are a better fit for unbiased rendering.

5.2 Sampling points and directions

We consider the problem of sampling the points that make up a transport path. Suppose we want to sample a point on a specific polygon $D \subset \mathcal{M}$, for example a sensor or a light source. If the polygon is a parallelogram, that is

$$D = \{\mathbf{o} + a\mathbf{e}_1 + b\mathbf{e}_2 \mid a \in [0,1], b \in [0,1]\}$$

for some vectors $\mathbf{o}, \mathbf{e}_1, \mathbf{e}_2 \in \mathbb{R}^3$, then

$$\mathbf{x} = \mathbf{o} + a\mathbf{e}_1 + b\mathbf{e}_2$$

is an uniformly distributed point on D if a and b are i.i.d. outcomes from $\mathcal{U}_{[0,1]}$. One way to generate outcomes on an arbitrary polygon is to first sample from a parallelogram that surrounds it and then reject the outcomes that are outside the polygon. The outcomes are uniformly distributed on D with respect to the surface area measure A. In other words, the PDF with respect to A satisfies

$$p(\mathbf{x}) = \frac{1}{A(D)}$$

for all outcomes $\mathbf{x} \in D$.

Consider the case where we want to sample points on the surfaces which are visible to some point $\mathbf{x} \in \mathcal{M}$. One way to achieve this is by sampling a direction $\omega \in H^+(\mathbf{x})$ and then using the ray casting function to find the point $r(\mathbf{x}, \omega)$. When constructing a transport path it is useful to importance sample the directions. As the incident radiance is still unknown to us, the best we can do is to sample from a distribution

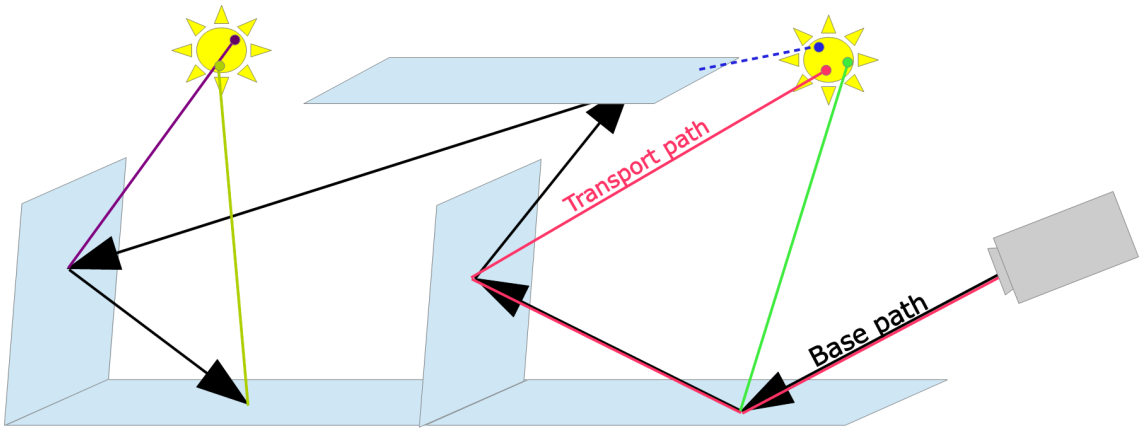


Figure 9: An example of a base path (shown in black) and a transport path (shown in red) generated by forward path tracing. Note that every point on the base path forms a transport path of it's own even though they are not explicitly highlighted.

which is as close to the BRDF as possible. If the BRDF is diffuse we want to sample directions from a distribution which is uniform with respect to the projected solid angle measure, in other words

$$p(\omega) = \frac{1}{\sigma^{\perp}_{\mathbf{x}}(H^{+}(\mathbf{x}))} = \frac{1}{\pi}$$

for all directions $\omega \in H^{+}(\mathbf{x})$. Shirley and Chiu [1997] present an algorithm that samples directions from this distribution by first sampling uniformly from the unit disk and then lifting the outcomes up to the hemisphere. The resulting directions need to be further rotated before they lie on $H^{+}(\mathbf{x})$.

5.3 Forward path tracing

The forward path tracing algorithm was introduced by Kajiya [1986] as an unbiased method for estimating the hemispherical formulation of the rendering equation. Here we present the algorithm as an unbiased Monte Carlo estimator for the path integral form of the measurement equation.

The algorithm creates transport paths from *base paths* that it generates starting from points on the sensors. Each base path is recursively extended by sampling its next point from the surfaces which are visible from the current point. The base path is then converted into multiple transport paths. These transport paths are created by connecting every point on the base path to a point sampled on the light sources. See Figure 9 for an overview.

If we only consider light that reaches the camera through one or more intermediate bounces, we can use the definitions from Chapter 4.5 to write the path integral formulation of the measurement equation as the infinite sum

$$\sum_{k=3}^{\infty} \int_{\mathcal{P}^k} f(\bar{\mathbf{x}}) \, \mathrm{d}\mathcal{A}^k(\bar{\mathbf{x}}).$$

First we show how the algorithm forms a Monte Carlo estimate for the finite sum

$$\sum_{k=3}^{M} \int_{\mathcal{P}^k} f(\bar{\mathbf{x}}) \, \mathrm{d}\mathcal{A}^k(\bar{\mathbf{x}}),$$

which we then extend to the infinite case with the help of a technique called *Russian roulette*.

Sampling paths. In order to form a Monte Carlo estimator

$$\hat{I}_N^k = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}_1, \ldots, \mathbf{x}_k)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_k)}$$

for one of the terms

$$\int_{\mathcal{P}^k} f(\mathbf{x}_1, \ldots, \mathbf{x}_k) \, \mathrm{d}\mathcal{A}^k(\mathbf{x}_1, \ldots, \mathbf{x}_k)$$

in the measurement equation, we need to sample transport paths $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ of length k and be able to evaluate the fraction

$$\frac{f(\mathbf{x}_1, \ldots, \mathbf{x}_k)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_k)}$$

for each transport path that we have sampled. The numerator is the measurement contribution function, which we know how to evaluate. The denominator is more problematic to evaluate as it requires us to express the probability density of the transport path with respect to the product area measure $\mathcal{A}^k$.

Recall that $\bar{\mathbf{x}}$ does not cause a sensor response unless $\mathbf{x}_1$ is on a light source, $\mathbf{x}_k$ is on the sensor and all adjacent points on the path are mutually visible. The forward path tracing algorithm tries to ensure this by sampling $\mathbf{x}_1$ on a light source and $\mathbf{x}_k$ on the sensor. The remaining points $\mathbf{x}_i$, where $1 < i < k$, are sampled from the surfaces which are visible to $\mathbf{x}_{i+1}$ [5].

[5] It is often beneficial to sample $\mathbf{x}_{k-1}$ differently to account for the camera model. Ideally, $\mathbf{x}_{k-1}$ should be sampled from a random variable whose PDF with respect to A is proportional to $W(\mathbf{x}_k \to \mathbf{x}_{k-1})$.

Using the sampling methods described in Chapter 5.2 above we have analytic expressions for the probability densities of $\mathbf{x}_1$ and $\mathbf{x}_k$ with respect to the surface area measure A. We also know the probability densities of the other points x_i with respect to the projected solid angle measure $\sigma^{\perp}_{\mathbf{x}_{i+1}}$. In order to express the joint probability density of the entire path with respect to the product area measure $\mathcal{A}^k$, we have to apply the Radon-Nikodym derivatives from Chapter 4. Formally, the probability density of $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ with respect to $\mathcal{A}^k$ is expressed

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_k) = p(\mathbf{x}_1)p(\mathbf{x}_k) \prod_{i=2}^{k-1} p(\omega(\mathbf{x}_{i+1} \to \mathbf{x}_i)) \frac{\mathrm{d}A(\mathbf{x}_i)}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}_{i+1}}(\omega(\mathbf{x}_{i+1} \to \mathbf{x}_i))},$$

where

$$\begin{aligned} \frac{\mathrm{d}A(\mathbf{x}_i)}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}_{i+1}}(\omega(\mathbf{x}_{i+1} \to \mathbf{x}_i))} &= \frac{\mathrm{d}A(\mathbf{x}_i)}{\mathrm{d}\sigma(\omega(\mathbf{x}_{i+1} \to \mathbf{x}_i))} \frac{\mathrm{d}\sigma(\omega(\mathbf{x}_{i+1} \to \mathbf{x}_i))}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}_{i+1}}(\omega(\mathbf{x}_{i+1} \to \mathbf{x}_i))} \\ &= \frac{|N(\mathbf{x}_{i+1}) \cdot \omega(\mathbf{x}_{i+1} \to \mathbf{x}_i)||N(\mathbf{x}_i) \cdot \omega(\mathbf{x}_i \to \mathbf{x}_{i+1})|}{\|\mathbf{x}_i - \mathbf{x}_{i+1}\|^2}. \end{aligned}$$

Recall that the numerator in the fraction is defined as

$$\begin{aligned} f(\mathbf{x}_1, \ldots, \mathbf{x}_k) &= f^k(\mathbf{x}_1, \ldots, \mathbf{x}_k) \\ &= L_e(\mathbf{x}_2 \leftarrow \mathbf{x}_1) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \\ &\quad \prod_{i=2}^{k-1} f^r(\mathbf{x}_{i-1} \to x_i \to \mathbf{x}_{i+1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \\ &\quad W(\mathbf{x}_k \to \mathbf{x}_{k-1}). \end{aligned}$$

If we inspect the geometry terms, we see that

$$G(\mathbf{x} \leftrightarrow \mathbf{y}) = V(\mathbf{x} \leftrightarrow \mathbf{y}) \frac{\mathrm{d}A(\mathbf{y})}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega(\mathbf{x} \to \mathbf{y}))}$$

where the Radon-Nikodym derivatives conveniently get cancelled out by identical factors in the denominator, with the notable exception of $G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_1)$. Due to the way we've sampled the points, we also know that the visibility term will evaluate to 1 between all point on the path, again with the exception of $V(\mathbf{x}_2 \leftrightarrow \mathbf{x}_1)$. These properties are often exploited when implementing the algorithm. In order to preserve clarity we choose not to apply such optimizations in our description of forward path tracing.

Finite sums. In the finite case, we estimate each of the integrals in the sum with separate Monte Carlo estimators. More precisely, we estimate the sum of integrals

$$\sum_{k=3}^{M} \int_{\mathcal{P}^k} f(\mathbf{x}_1, \ldots, \mathbf{x}_k) \, \mathrm{d}\mathcal{A}^k(\mathbf{x}_1, \ldots, \mathbf{x}_k)$$

using the sum of estimators

$$\sum_{k=3}^{M} \hat{I}_k^N = \sum_{k=3}^{M} \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}_1, \ldots, \mathbf{x}_k)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_k)}.$$

The forward path tracing algorithm reduces computation by reusing the same base path to generate one transport path per estimator. First it generates the base path $(\mathbf{x}_1, \ldots, \mathbf{x}_{M-1})$, where $\mathbf{x}_{M-1}$ is on the sensor and the other points $\mathbf{x}_i$ are sampled from the surfaces which are visible to $\mathbf{x}_{i+1}$. After that it converts the base path into transport paths by attaching each suffix, $(\mathbf{x}_j, \ldots, \mathbf{x}_{M-1})$ for some $1 < j < M$, from the base path to a point $\mathbf{y}_j$ that has been sampled on the light sources. After this we have $M - 1$ transport paths of the form $(\mathbf{y}_j, \mathbf{x}_j, \ldots, \mathbf{y}_{M-1})$. In fact, we have one path for each of the estimators in the sum.

Russian roulette. Russian roulette allows us to estimate infinite sums with a finite amount of computation without introducing bias. At its core, Russian roulette relies on the fact that

$$E[X] = E[R(q, X)]$$

if we define

$$R(q, X) = \begin{cases} \frac{1}{q} X \text{ with a probability of } q \\ 0 \text{ with a probability of } 1 - q. \end{cases}$$

Based on this, we see that

$$\sum_{k=3}^{\infty} R(2^{-k+3}, \hat{I}_k^N)$$

is an unbiased estimate of the path integral formulation of the measurement equation. The variance of this estimator can be reduced by tweaking the values of q as described by Veach [1998, Chapter 10.4.1]. Algorithm 4 describes how to evaluate this estimate with a single light source $D \subset \mathcal{M}$. The algorithm alternates between extending the base path and creating a new transport path, both of which are performed by the recursive function ESTIMATERADIANCE. The lines $4 - 8$ elucidate the term Russian roulette, as there is a 50% chance of terminating the procedure at

every recursion. This method ensures that the probability of surviving long enough to generate a path of length k is 2^{-k+3}.

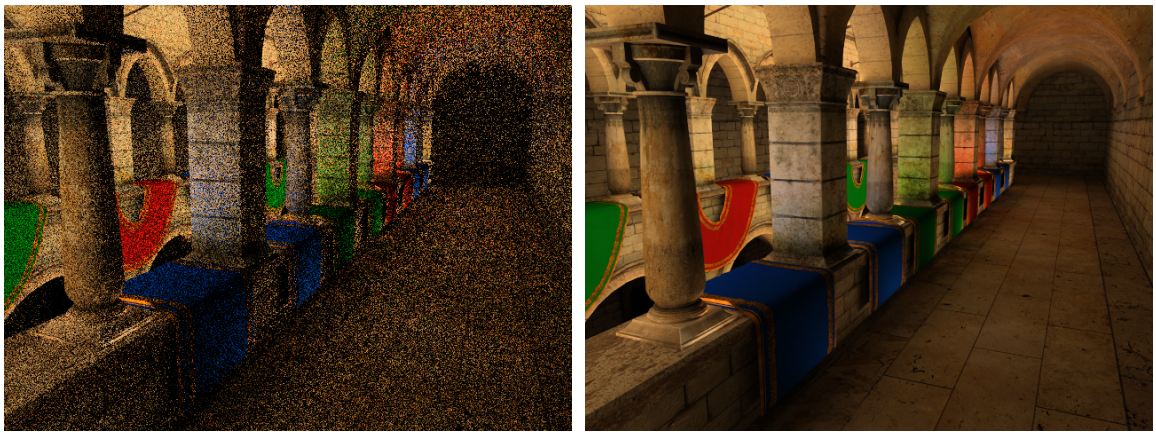
Algorithm 4 Forward path tracing

function ESTIMATERADIANCE($\mathbf{o}, \mathbf{x}$)
  $\mathbf{y} \leftarrow$ SAMPLEPOLYGON(D)
  $L \leftarrow \dfrac{L_e(\mathbf{y} \to \mathbf{x}) G(\mathbf{y} \leftrightarrow \mathbf{x}) f^r(\mathbf{y} \to \mathbf{x} \to \mathbf{o})}{\text{PDFAREA}(\mathbf{y}, D)}$
  $rr \leftarrow$ OUTCOMEFROM($\mathcal{U}_{[0,1]}$)
  if $rr \leq \frac{1}{2}$ **then**
    $\omega \leftarrow$ SAMPLEDIRECTION($H^+(\mathbf{x})$)
    $\mathbf{x}' \leftarrow r(\mathbf{x}, \omega)$
    $L \leftarrow L + 2 \dfrac{G(\mathbf{x} \leftrightarrow \mathbf{x}') f^r(\mathbf{x}' \to \mathbf{x} \to \mathbf{o}) \text{ESTIMATERADIANCE}(\mathbf{x}', \mathbf{x})}{\dfrac{\mathrm{d}A(\mathbf{x}')}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)} \text{PDFPROJECTEDSOLIDANGLE}(\omega, H^+(\mathbf{x}))}$
  return L
for each sensor S **do**
  for $i = 1$ to N **do**
    $\mathbf{x} \leftarrow$ SAMPLEPOLYGON(S)
    $\omega \leftarrow$ SAMPLEDIRECTION($H^+(\mathbf{x})$)
    $\mathbf{x}' \leftarrow r(\mathbf{x}, \omega)$
    $L \leftarrow G(\mathbf{x} \leftrightarrow \mathbf{x}') W(\mathbf{x} \to \mathbf{x}') \text{ESTIMATERADIANCE}(\mathbf{x}', \mathbf{x})$
    $p \leftarrow \dfrac{\mathrm{d}A(\mathbf{x}')}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)} \text{PDFPROJECTEDSOLIDANGLE}(\omega, H^+(\mathbf{x})) \text{PDFAREA}(\mathbf{x}, S)$
    $\text{measurement}[S] \leftarrow \text{measurement}[S] + \dfrac{1}{N} \dfrac{L}{p}$

Variance characteristics. Forward path tracing produces unbiased estimates of the measurement equation. This means that the error in the images it produces will be in the form of variance. As the sensor response for each pixel is estimated using independently generated transport paths, the variance manifests as high frequency noise. Perceptually this noise is similar to film grain, see Figure 10 for an example.

5.4 Instant Radiosity

Instant Radiosity was introduced by Keller [1997] as an algorithm for estimating the hemispherical formulation of the measurement equation. In the original formulation, the algorithm makes use of rasterisation to resolve visibility which results in faster



(a) Unconverged image containing high frequency noise.

(b) Fully converged image.

Figure 10: Demonstration of the variance characteristics of the forward path tracing algorithm. Note that the images above only contain indirect light.

computation times but also introduces bias. This formulation also relies on the *radiosity assumption*, that all the surfaces in the scene are diffuse. This assumption simplifies computation because the indirect light field can be simulated using diffuse emitters. Here we present Instant Radiosity as an unbiased estimator for the full measurement equation in the path integral formulation. In order to achieve this, we do not make use of the radiosity assumption and use ray casting to resolve visibility.

The central idea behind Instant Radiosity is to represent the illumination in the scene using *virtual point lights (VPLs)*. After the VPLs have been placed, the image is rendered using only their direct illumination. Without the radiosity assumption, we cannot represent the indirect light field using diffuse emitters and talk about *light paths* instead of VPLs. Formally, a light path is a collection of points $(\mathbf{y}_1, \ldots, \mathbf{y}_k)$, where all adjacent points are mutually visible and $\mathbf{y}_1$ is on a light source. The algorithm generates a transport path $(\bar{\mathbf{y}}, \bar{\mathbf{x}})$ by connecting a light path $\bar{\mathbf{y}}$, to a *camera path* $\bar{\mathbf{x}}$. Since the final image is rendered using only direct illumination from the light paths, each camera path contains exactly two points. Formally, a camera path is a pair of points $(\mathbf{x}_1, \mathbf{x}_2)$ that are mutually visible, where $\mathbf{x}_2$ is on a sensor.

Sampling paths. Instant Radiosity generates camera paths by sampling a pair of points $(\mathbf{x}_1, \mathbf{x}_2)$, such that $\mathbf{x}_2$ is on a sensor and $\mathbf{x}_1$ is on the surfaces visible to $\mathbf{x}_2$. Since the camera paths can only contain two points, the algorithm is only able to sample longer transport paths by increasing the length of the light paths. Similarly

to forward path tracing, it samples light paths by first generating base paths whose lengths are determined by Russian roulette. Unlike forward path tracing, each base path is generated by first sampling a point $\mathbf{y}_1$ on the light sources, after which the other points $\mathbf{y}_i$ on the path are sampled from the surfaces visible to $\mathbf{y}_{i-1}$. The resulting base path contains multiple light paths defined by each of its prefixes.

Suppose we have generated a transport path $(\bar{\mathbf{y}}, \bar{\mathbf{x}})$ by deterministically connecting a camera path and a light path. If we want to use this path in an estimator for the path integral form of the measurement equation, we need to be able to evaluate the fraction

$$\frac{f(\bar{\mathbf{y}}, \bar{\mathbf{x}})}{p(\bar{\mathbf{y}}, \bar{\mathbf{x}})},$$

where f is the familiar measurement contribution function. Like with forward path tracing we're faced with the problem of evaluating the probability density of the transport path with respect to the product area measure $\mathcal{A}$. If we already know the probability density of $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ with respect to $\mathcal{A}$, we use the fact that $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ have been independently sampled to express $p(\bar{\mathbf{y}}, \bar{\mathbf{x}})$ as the product $p(\bar{\mathbf{y}})p(\bar{\mathbf{x}})$.

Using the same reasoning as in the forward path tracing section, we see that the probability density of a light path of length k with respect to $\mathcal{A}$ is

$$p(\bar{\mathbf{y}}) = p(\mathbf{y}_1) \prod_{i=2}^{k} p(\omega(\mathbf{y}_{i-1} \to \mathbf{y}_i)) \frac{\mathrm{d}A(\mathbf{y}_i)}{\mathrm{d}\sigma^{\perp}_{\mathbf{y}_{i-1}}(\omega(\mathbf{y}_{i-1} \to \mathbf{y}_i))}$$

and the probability density of a camera path with respect to $\mathcal{A}$ is

$$p(\bar{\mathbf{x}}) = p(\mathbf{x}_2)p(\omega(\mathbf{x}_2 \to \mathbf{x}_1)) \frac{\mathrm{d}A(\mathbf{x}_1)}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}_2}(\omega(\mathbf{x}_2 \to \mathbf{x}_1))}.$$

In contrast to forward path tracing, where each sensor generates independent transport paths, Instant Radiosity saves computation by sharing the light path between all sensors. Algorithm 5 describes this process in detail for the case where the scene is illuminated by one light source $D \subset \mathcal{M}$. At a high level, the algorithm first creates a base path whose prefixes defines a set of light paths. It then generates one camera path per sensor from and constructs a set of transport paths by connecting each camera path to every light path. This process is repeated until enough light transport paths have been generated. The function GENERATELIGHTPATHS generates a base path whose length is determined using Russian roulette. If we analyse the function, we see that the probability of generating a base path of length k is 2^{-k+1} and that the radiance is also scaled appropriately.

Algorithm 5 Instant Radiosity

function vplRadiance($l, \mathbf{x}$)
  if $l.\mathbf{x} \in D$ **then**
    return $L_e(l.\mathbf{x} \to x)$
  return $f^r(l.\mathbf{x}' \to l.\mathbf{x} \to \mathbf{x})G(l.\mathbf{x}' \leftrightarrow l.\mathbf{x})l.L$
function generateLightPaths()
  $\mathbb{L} \leftarrow \emptyset$
  $l.\mathbf{x} \leftarrow$ samplePolygon(D)
  $l.p \leftarrow$ pdfArea($l.\mathbf{x}, D$)
  loop
    $\mathbb{L} \leftarrow \mathbb{L} \cup \{l\}$
    $rr \leftarrow$ outcomeFrom($\mathcal{U}_{[0,1]}$)
    if $rr > \frac{1}{2}$ **then**
      return $\mathbb{L}$
    $\omega \leftarrow$ sampleDirection($H^+(l.\mathbf{x})$)
    $l'.\mathbf{x} \leftarrow r(l.\mathbf{x}, \omega)$
    $l'.\mathbf{x}' \leftarrow l.\mathbf{x}$
    $l'.p \leftarrow$ pdfProjectedSolidAngle($\omega, H^+(l.\mathbf{x})$)$\dfrac{\mathrm{d}A(l'.\mathbf{x})}{\mathrm{d}\sigma^{\perp}_{l.\mathbf{x}}(\omega)} l.p$
    $l'.L \leftarrow 2$ vplRadiance($l, l'.\mathbf{x}$)
    $l \leftarrow l'$
for $i = 1$ to N **do**
  $\mathbb{L} \leftarrow$ generateLightPaths()
  for each sensor S **do**
    $\mathbf{x} \leftarrow$ samplePolygon(S)
    $p \leftarrow$ pdfArea($\mathbf{x}, S$)
    $\omega \leftarrow$ sampleDirection($H^+(\mathbf{x})$)
    $\mathbf{x}' \leftarrow r(\mathbf{x}, \omega)$
    $p' \leftarrow$ pdfProjectedSolidAngle($\omega, H^+(\mathbf{x})$)$\dfrac{\mathrm{d}A(\mathbf{x}')}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)} p$
    for each l in $\mathbb{L}$ **do**
      $L \leftarrow W(\mathbf{x} \to \mathbf{x}')G(\mathbf{x} \leftrightarrow \mathbf{x}')f^r(l.\mathbf{x} \to \mathbf{x}' \to \mathbf{x})$
        $< G(\mathbf{x}' \leftrightarrow l.\mathbf{x})$vplRadiance($l, \mathbf{x}'$)
      measurement[S] $\leftarrow$ measurement[S] $+ \dfrac{1}{N}\dfrac{L}{pl.p}$

(a) Unconverged image containing both low frequency noise and singularities.

(b) Fully converged image.

Figure 11: Demonstration of the variance characteristics of Instant Radiosity. Red circles mark singularities. Note that the images above only contain indirect light.

Variance characteristics. Similarly to forward path tracing, Instant Radiosity does not produce images with systematic bias and the error it produces is in the form of variance. In contrast to forward path tracing, Instant Radiosity uses heavily correlated paths for all the sensors and as a consequence the variance manifests as low frequency noise. In other words, the error is spread out over large areas of the image. However, low frequency noise is clearly noticeable when rendering image sequences, since it causes large regions of the image to change appearance between frames. Informally, this is perceived as a "flickering"-effect.

Another consequence of the sampling method occurs when the end of a light path is placed adjacent to a surface in the scene, for example in a corner. This manifests as bright spots in the image and is explained by a singularity in the terms of the Monte Carlo sum, see Figure 11 for examples of both low frequency noise and singularities.

The singularities occur because one of the Radon-Nikodym derivatives hidden by $G(\mathbf{x} \leftrightarrow \mathbf{y})$ in the numerator does not get compensated for by the probability density in the denominator. Specifically, this occurs between the last point on the light path $\mathbf{y}_k$ and the first point of the camera path $\mathbf{x}_1$. As a consequence, the numerator of each term in the Monte Carlo sum still contains

$$G(\mathbf{y}_k \leftrightarrow \mathbf{x}_1) = V(\mathbf{y}_k \leftrightarrow \mathbf{x}_1)\frac{|N(\mathbf{y}_k) \cdot \omega(\mathbf{y}_k \rightarrow \mathbf{x}_1)||N(\mathbf{x}_1) \cdot \omega(\mathbf{x}_1 \rightarrow \mathbf{y}_k)|}{\|\mathbf{x}_1 - \mathbf{y}_k\|^2}.$$

The bright spots appear because this function tends to infinity as the distance between $\mathbf{y}_k$ and $\mathbf{x}_1$ approaches zero.

One effective, although biased method of removing the singularities is to clamp the geometry term to a maximum value. In fact, this was employed in the original formulation of Instant Radiosity [Keller, 1997]. Later methods, such as the one described by Kollig and Keller [2006], have been developed that are able to remove this singularities in an unbiased fashion, although at the cost of rendering speed.

6 Spatial data structures

Spatial data is described by Samet [1990] as points, lines, rectangles, regions, surfaces and volumes. Spatial data structures organize such data in n-dimensional space. In this thesis, we concern ourselves with two spatial data structures: The *bounding volume hierarchy* (BVH), which organizes points in $\mathbb{R}^3$, and the *bounding cone hierarchy* (BCH), which organizes directions (points in $\mathbb{S}^2$).

The Design and Analysis of Spatial Data Structures by Samet [1990] is a good reference for spatial data structures. Refer to Chapter 14.1 of the third edition of *Real-Time Rendering* by Akenine-Möller et al. [2008] for an overview of spatial data structures in the context of computer graphics. *Heuristic Ray Shooting Algorithms* by Havran [2000] is a thorough treatment on how to accelerate the ray casting operation using spatial data structures. Walter et al. [2005] make use of bounding cone hierarchies in the context of representing the aggregate illumination from a collection of point lights.

6.1 Bounding volume hierarchy

Suppose that we want to organize a set of points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subset \mathbb{R}^3$. One way to achieve this is with the help of *bounding volumes*, simple geometric objects that contains these points. Objects such as spheres and boxes are commonly used as bounding volumes. Expensive calculations against the original set of points can be approximated by performing the calculations on the much simpler bounding volume.

If we build a BVH, that is a hierarchy that consists of bounding volumes within bounding volumes, then we are able to choose the precision of our approximations. This is achieved by descending the hierarchy until the approximation error between the bounding volume and the points it contains is small enough for our purposes. Formally, a bounding volume hierarchy is a tree of bounding volumes, where a parent node encompasses its children.

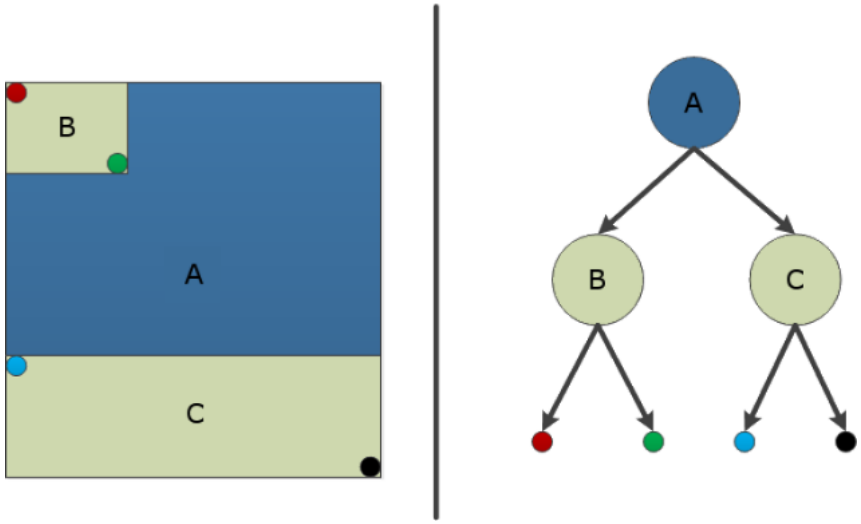


Figure 12: A 2D visualization of a bounding volume hierarchy. The left side shows how the points are grouped together in space, the right side shows the tree structure of the hierarchy.

In this thesis, we use *axis aligned bounding boxes* (AABBs) as our bounding volumes. Mathematically, the AABB $B(X)$ for the points X is defined as

$$\begin{aligned} B(X) = &[\min(pr_x(X)), \max(pr_x(X))]\times \\ &[\min(pr_y(X)), \max(pr_y(X))]\times \\ &[\min(pr_z(X)), \max(pr_z(X))], \end{aligned}$$

where $pr_x : \mathbb{R}^3 \to \mathbb{R}$ projects points in $\mathbb{R}^3$ onto the x-axis. pr_y and pr_z are similar functions for the y and z axes. Informally, $B(X)$ is the smallest box that contains all the points in X with the constraint that its faces are parallel to the coordinate axes. See Figure 12 for an illustration of a AABB-based BVH in 2D.

Construction methods. Different BVHs can be built from the same set of points by choosing to group different bounding volumes together at each level of the hierarchy. Here we describe two heuristics that are often employed when creating a bounding volume hierarchy: the *spatial mean* and the *object median* [Havran, 2000, Chapter 4.2.2].

The spatial mean heuristic strives to always split the spatial extent equally between the children of a node. This ensures that the sizes of the bounding volumes in the hierarchy shrink as quickly as possible. The rationale is that small nodes are less likely to targeted for processing.

The object median heuristic attempts to always split the amount of objects equally between the children of a node. Constructing spatial data structures using this technique leads to balanced (and consequently shallow) hierarchies. This directly affects the maximum root-to-leaf traversal time of the resulting BVH.

Two common methods of building these hierarchies are the *top-down* method and the *bottom-up* (or *agglomerative*) method. The top-down method starts by creating a bounding volume that contains all of the points. It then proceeds to generate the inner bounding volumes in the hierarchy by recursively splitting the nodes using one of the heuristics described above. The bottom-up method constructs a hierarchy by first merging two points into a bounding volume. It then proceeds to merge a bounding volume or a point with another point or bounding volume, both of which have been chosen using one of the heuristics above. This process repeats until there is only one bounding volume.

Surface area heuristic. In the context of ray casting algorithms MacDonald and Booth [1990] introduced the *surface area heuristic* (SAH), a method of estimating the ray casting performance of a spatial data structure. MacDonald and Booth show that the split which minimizes the SAH estimate lies between the spatial mean and the object median. However, the SAH estimate is not a perfect indicator of ray casting performance. Aila et al. [2013] propose a new metric as an extension to SAH, which better reflects ray casting performance.

6.2 Bounding cone hierarchy

Bounding cone hierarchies are an variant of bounding volume hierarchies that organize directions instead of points. Suppose we want to organize a set of directions $\Omega = \{\omega_1, \ldots, \omega_k\} \subset \mathbb{S}^2$. One way to bound these directions is with so called *bounding cones*, circular regions in $\mathbb{S}^2$. We represent a bounding cone $B(\Omega)$ using the pair (ω_c, θ), where $\omega_c \in \mathbb{S}^2$ is the direction at the centre of the cone and $\theta \in [-1, 1]$ is the lower limit for the cosine of the angle between ω_c and any direction $\omega \in \Omega$ [6]. Again, a BCH is a hierarchy that consists of bounding cones within bounding cones, see Figure 13 for an illustration.

Constructing an optimal bounding cone around Ω is solved by first constructing an

[6] Strictly speaking, this defienes a spherical sector, that is a cone with a rounded base. We use the term cone to agree with previous literature.

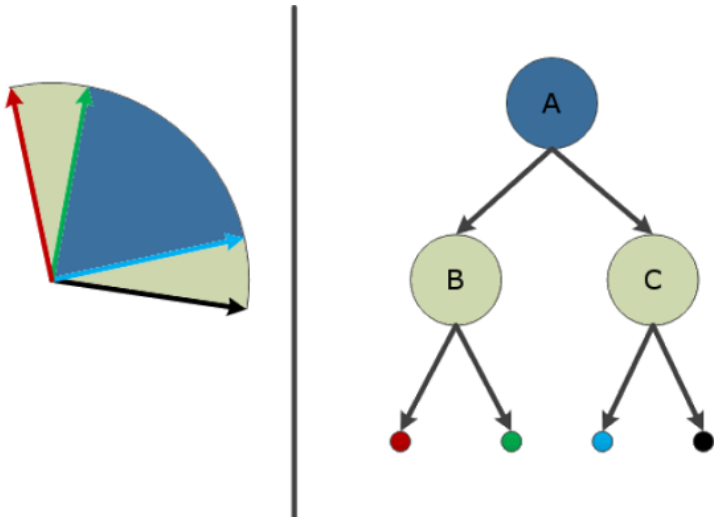


Figure 13: A 2D visualization of a bounding cone hierarchy. The left side shows how the directions are grouped together in space, the right side shows the tree structure of the hierarchy.

optimal bounding sphere around the points in $\Omega \subset \mathbb{S}^2 \subset \mathbb{R}^3$. See Figure 14 for an illustration of the resulting sphere, where $\mathbf{x}$ is the centre of the sphere and r is its radius. According to the law of cosines, we have

$$\begin{aligned} r^2 &= \|\mathbf{x}\|^2 + 1^2 - 2\|\mathbf{x}\| \cos(\alpha) \\ \Rightarrow \cos(\alpha) &= \frac{\|\mathbf{x}\|^2 + 1 - r^2}{2\|\mathbf{x}\|}, \end{aligned}$$

which allows us to express the resulting bounding cone $B(\Omega)$ as the pair $(\frac{\mathbf{x}}{\|\mathbf{x}\|}, \cos(\alpha))$. Barequet and Elber [2005] present a selection of algorithms that finds the optimal bounding cone for a set of directions. We make use of the conceptually simpler approach by Shirmun and Abi-Ezzi [1993] that initially treats the directions as points in $\mathbb{R}^3$ and forms an AABB around them. The final bounding cone is then constructed from the smallest sphere surrounding the AABB.

7 Previous work

In this chapter we review methods that simulate indirect light by extending the Instant Radiosity algorithm. Specifically, we look at how these methods reduce the

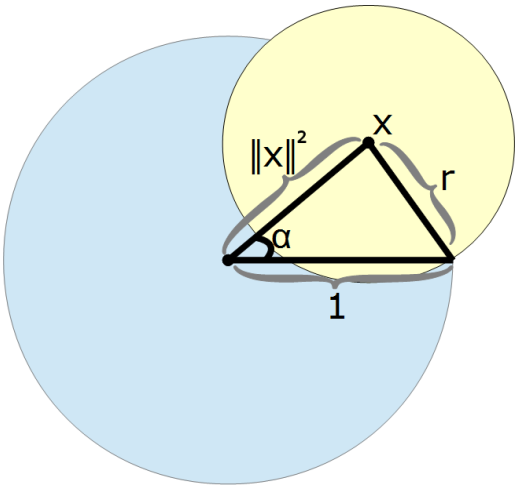


Figure 14: A diagram that helps us construct a bounding cone from a bounding sphere of directions.

"flickering"-effect[7]. From our perspective, the methods can be divided into three categories:

1. Methods that increase the VPL budget. In this case the effect is reduced as a consequence of brute force variance reduction in each individual frame.

2. Methods that importance sample the VPLs according to the sensor reponses they cause. Again, this reduces the effect through variance reduction.

3. Methods that move fewer VPLs between images in a sequence. These methods do not reduce the over all variance, but instead prevents the error from fluctuating between frames.

First we give an overview briefly describing methods from each category. We also shortly cover temporal filtering, a method unrelated to Instant Radiosity that also reduces "flickering". We then present one representative method in detail from each category: *Lightcuts* by Walter et al. [2005] from the first category, *Metropolis Instant Radiosity* by Segovia et al. [2007] from the second category and *Incremental Instant Radiosity* by Laine et al. [2007] from the third category.

[7]For a more general overview of extensions to Instant Radiosity, refer to the survey by Dachsbacher et al. [2014].

7.1 Overview

Increasing the VPL budget. Many lights rendering is the problem of rendering a picture under direct illumination from a large amount of point lights, which is the case in Instant Radiosity. Hašan et al. [2007] present a method that interprets *many lights rendering* as a low rank matrix. Using the matrix interpretation, they show that the entire image can be rendered using only a few representative point lights, which they find by exploring the surfaces that are indirectly visible to the camera using rasterisation. Hašan et al. [2008] extend this method to the temporal dimension, where the shaded results from a representative point light can be reused over multiple frames in an animated video.

Ritschel et al. [2008] develop another improvement that increases the VPL budget by settling for approximate visibility. More precisely, they rasterise simplified version of the scene when estimating the visibility between the camera paths and the VPLs. Dong et al. [2009] achieve the same goal by grouping the VPLs into clusters and compute the visibilities on a per-cluster basis. Ulbrich et al. [2013] present a method that increases the VPL budget by decoupling the visibility computation from the VPLs. Instead they adaptively subdivide the scene into patches and use shadow maps to estimate the hemispherical visibility for each patch.

The total VPL budget can be increased if only a subset of the VPLs are used to estimate the sensor response of a pixel. Segovia et al. [2006b] and Laine et al. [2007] achieve this by grouping the pixels into blocks. They then evenly divide the VPLs such that a single VPL is only used to evaluate the sensor response of one pixel per block. Georgiev et al. [2012] interpret the sensor response of a pixel as a Monte Carlo sum over all the VPLs. To reduce variance, they first evaluate the sensor response caused by the light from every VPL reflecting off a set of sparsely selected points on the visible surfaces. They then interpolate these responses to define importance sampling weights for the VPLs when estimating the sensor response of a pixel. Wu and Chuang [2013] also employ importance sampling to select VPLs when estimating the response of a pixel, but define their importance sampling weights differently. Once visibility is known, the sensor response caused by the light from a VPL reflecting off a visible point is easy to evaluate. They form the importance sampling weights by evaluating the sensor responses caused by each VPL using estimated visibilities.

While these techniques do reduce the overall error in Instant Radiosity renderings through brute-force variance reduction, they do not provide any means to keep the

error consistent between frames.

Importance sampling VPLs. Segovia et al. [2006a] present an improvement that importance samples the VPLs according to the sensor responses they cause. Their method performs importance sampling through weighted resampling with replacement from a collection of candidate VPLs. They generate the candidates by sampling light paths as in traditional Instant Radiosity as well as sampling from the surfaces that are indirectly visible to the sensors. They then set the resampling weights by estimating the total sensor response caused by each VPL. Georgiev and Slusallek [2010] also generate a collection of candidate VPLs, but instead use rejection sampling to generate VPLs that are distributed according to the sensor responses they cause. Ritschel et al. [2011] present similar method that works in real-time for single-bounce indirect illumination by ignoring the visibility term when computing the resampling weights. As above, these techniques cannot keep the rendering error consistent between frames.

Moving fewer VPLs. Both Prutkin et al. [2012] and Barák et al. [2013] extend the real-time importance sampling method by Ritschel et al. [2011] to minimize the movement of VPLs between frames.

Similarly to Dong et al. [2009], Prutkin et al. [2012] form the final set of VPLs by clustering together a large collection of candidate VPLs. They minimize the movement of VPLs by seeding the clustering algorithm using the VPLs from the last frame. Barák et al. [2013] use a different importance sampling algorithm to select VPLs, as the analytic inversion method used by Ritschel et al. [2011] is sensitive to local changes, even if the pseudo-random number generator is initialized the same way every frame. Instead they sample the VPLs using a variant of the Metropolis-Hastings method with independent proposals.

Although both of these methods demonstrably reduce the "flickering" effect, they only support single bounce indirect illumination and cannot enforce temporal coherence when the primary light sources move. In comparison, our method doesn't suffer from these limitations.

Temporal filtering. Another way to reduce the "flickering" effect is to filter the illumination over time. This is achieved by maintaining a separate image of the indirect illumination from the previous frames, which is then reprojected whenever

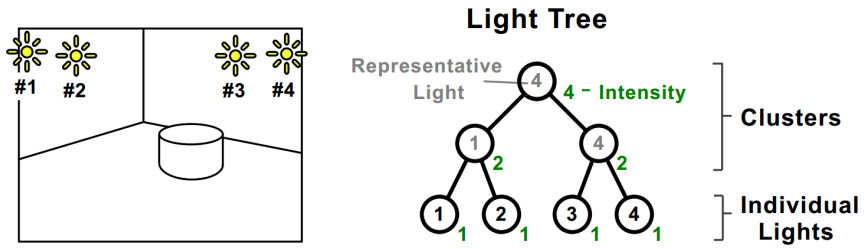


Figure 15: A diagram from the original Lightcuts paper by Walter et al. [2005] that shows the light tree constructed from a set of four point lights. The left side shows the point lights in the scene and the right side shows the resulting light tree. From this diagram we see how the inner nodes represent the aggregate illumination of their descendants by choosing a representative point light. We also see that the intensity of an inner node is the total intensity of the point lights it represents.

the camera moves. Knecht et al. [2010] present such a method that uses an exponentially weighted moving average to combine the the indirect illumination over a sliding window of frames. We employ a similar method by Mara et al. [2013] in our work.

7.2 Lightcuts

Lightcuts is a many lights rendering method developed by Walter et al. [2005]. It renders scenes directly illuminated by point lights such that the rendering time is only weakly dependent on the number of point lights. This can be used to render physically based images with indirect illumination if the point lights are generated using Instant Radiosity.

The main insight in Lightcuts is that although every camera path needs to account for the incident radiance from all point lights, it only needs accurate radiance from a small portion of them. To achieve this, the point lights which are not essential for an accurate sensor response are clustered together and the incident radiance from each cluster is evaluated by treating it as a single point light.

In order to accelerate the per-camera path clustering of the point lights Walter et al. make use of a *light tree*, a bounding volume hierarchy of the point lights in the scene which they construct in a bottom-up fashion based on custom heuristics. The light tree also doubles as a bounding cone hierarchy, since it bounds the normals of the point lights. Each inner node in the light tree also stores the sum of the intensities for all the point lights it represents. It also stores the position and normal of one of

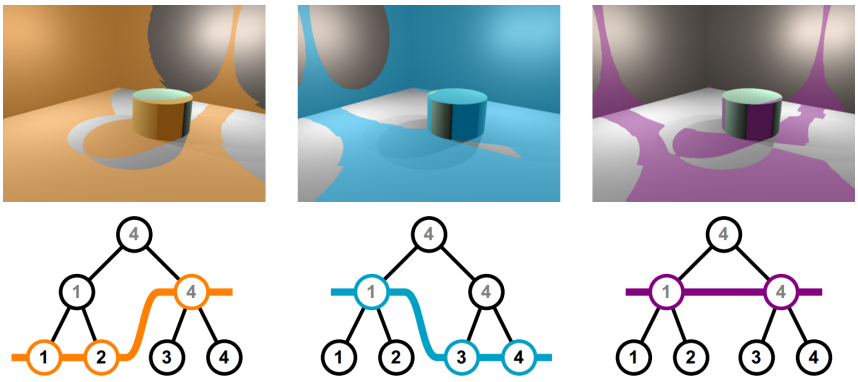


Figure 16: Another diagram from the original Lightcuts paper by Walter et al. [2005] that shows images rendered under the illumation from different cuts of the light tree. The highlighted regions in the pictures represent the areas where the illumination from the cuts agrees with the illumination from the original set of point lights.

the point lights which represents the cluster during rendering. See Figure 15 for an illustration of the light tree.

Tree cuts. In order for a camera path to account for the incident radiance from all the point lights in the scene, it needs to connect to all the nodes in a *cut* of the light tree. A cut of a tree is a subset of its nodes, such that all paths from the leaves to the root of the tree visit exactly one node in the cut. From Figure 16 we see that even though a cut of the light tree does not produce valid estimates of the incident radiance for all the camera paths in an image, these estimates are accurate enough for large regions of the image. This means that an accurate image can be formed by selecting different cuts for different camera paths.

The Lightcuts method finds a suitable cut of the light tree for a camera path with the help of an error metric defined on the nodes in the tree. In practice, it chooses a cut starting from the root node of the hierarchy and progressively refines the cut by replacing the node in the cut that has the largest error with its two descendants. This process is repeated until the maximum error of the nodes in the cut is below a predefined threshold.

Error metric. In order to bound the error of the sensor response caused by connecting a camera path $\bar{\mathbf{x}} = (\mathbf{x}_1, \mathbf{x}_2)$ to a node in the light tree, it is sufficient to bound the error of the exitant radiance from $\mathbf{x}_1$ towards $\mathbf{x}_2$ on the sensor. The error metric used by Walter et al. is an upper bound $\hat{S}_{\bar{\mathbf{x}}}(D)$ for the exitant radiance $\tilde{S}(D)$

caused by connecting $\bar{\mathbf{x}}$ to the representative point light of a node D in the light tree. $\hat{S}_{\bar{\mathbf{x}}}(D)$ is also an upper bound for the total exitant radiance $S(D)$ caused by connecting $\mathbf{x}$ to all point lights represented by D. From this we see that the error

$$\begin{aligned} Err_{\bar{\mathbf{x}}}(D) &= |\tilde{S}_{\bar{\mathbf{x}}}(D) - S_{\bar{\mathbf{x}}}(D)| &&= \max(\tilde{S}_{\bar{\mathbf{x}}}(D) - S_{\bar{\mathbf{x}}}(D), S(D) - \tilde{S}_{\bar{\mathbf{x}}}(D)) \\ &\leq \max(\hat{S}_{\bar{\mathbf{x}}}(D) - 0, \hat{S}_{\bar{\mathbf{x}}}(D) - 0) &&= \hat{S}_{\bar{\mathbf{x}}}(D), \end{aligned}$$

since $0 < S_{\bar{\mathbf{x}}}(D) < \hat{S}_{\bar{\mathbf{x}}}(D)$ and $0 < \tilde{S}_{\bar{\mathbf{x}}}(D) < \hat{S}_{\bar{\mathbf{x}}}(D)$.

Expressing the exitant radiance. We derive expressions for $S_{\bar{\mathbf{x}}}(D)$ and $\hat{S}_{\bar{\mathbf{x}}}(D)$ by assuming that the point lights are diffuse emitters and that all the materials in the scene are opaque. With the help of the notation from Chapter 5.4, we see that

$$\begin{aligned} S_{\bar{\mathbf{x}}}(D) &= \sum_{l \in D} L_e(l.\mathbf{x} \to \mathbf{x}_1) G(l.\mathbf{x} \leftrightarrow \mathbf{x}_1) f^s(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2) \\ &= \sum_{l \in D} L_e(l.\mathbf{x} \to \mathbf{x}_1) V(l.\mathbf{x} \leftrightarrow \mathbf{x}_1) \\ &\qquad \frac{|N(\mathbf{x}_1) \cdot \omega(\mathbf{x}_1 \to l.\mathbf{x})||N(l.\mathbf{x}) \cdot \omega(l.\mathbf{x} \to \mathbf{x}_1)|}{\|l.\mathbf{x} - \mathbf{x}_1\|^2} \\ &\qquad f^s(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2). \end{aligned}$$

Where f^s is the BSDF, which for opaque materials is defined as

$$f^s(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2) = \begin{cases} f^r(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2) & \text{if } \omega(\mathbf{x}_1 \to l.\mathbf{x}) \in H^+(\mathbf{x}_1) \\ 0 & \text{otherwise,} \end{cases}$$

where f^r is a BRDF. Since all of the point lights are assumed to be diffuse emitters, we see that

$$L_e(l.\mathbf{x} \to \mathbf{x}_1) = \begin{cases} L(l.\mathbf{x}) & \text{if } \omega(l.\mathbf{x} \to \mathbf{x}_1) \in H^+(l.\mathbf{x}) \\ 0 & \text{otherwise,} \end{cases}$$

where $L(l.\mathbf{x})$ is the intensity of the point light l.

Since $\omega \cdot N(\mathbf{x}) < 0 \Leftrightarrow \omega \notin H^+(\mathbf{x})$, we rewrite

$$\begin{aligned} S_{\bar{\mathbf{x}}}(D) = \sum_{l \in D} & L(l.\mathbf{x}) V(l.\mathbf{x} \leftrightarrow \mathbf{x}_1) \\ & \frac{\max(0, N(\mathbf{x}_1) \cdot \omega(\mathbf{x}_1 \to l.\mathbf{x})) \max(0, N(l.\mathbf{x}) \cdot \omega(l.\mathbf{x} \to \mathbf{x}_1))}{\|l.\mathbf{x} - \mathbf{x}_1\|^2} \\ & f^r(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2), \end{aligned}$$

where we make use of clamped cosines to substitute the opaque BSDF with its corresponding BRDF and substitute the emitted radiance with the intensity of the point light. With a similar transformation, we express

$$\begin{aligned}\tilde{S}(D) =& L(D)V(l_D.\mathbf{x} \leftrightarrow \mathbf{x}_1) \\ & \frac{\max(0, N(\mathbf{x}_1) \cdot \omega(\mathbf{x}_1 \rightarrow l_D.\mathbf{x}))\max(0, N(l_D.\mathbf{x}) \cdot \omega(l_D.\mathbf{x} \rightarrow \mathbf{x}_1))}{\|l_D.\mathbf{x} - \mathbf{x}_1\|^2} \\ & f^r(l_D.\mathbf{x} \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2),\end{aligned}$$

where $L(D) = \sum_{l \in D} L(l.\mathbf{x})$ and l_D is the representative point light for D.

Factoring the exitant radiance. Walter et al. derive an expression for $\hat{S}$ by decomposing the terms in S up into four factors:

- the material factor $M_l(\mathbf{x}_1) = \max(0, N(\mathbf{x}_1) \cdot \omega(\mathbf{x}_1 \rightarrow l.\mathbf{x}))f^s(l.\mathbf{x} \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2)$,
- the geometry factor $G_l(\mathbf{x}_1) = \dfrac{\max(0, N(l.\mathbf{x}) \cdot \omega(l.\mathbf{x} \rightarrow \mathbf{x}_1))}{\|l.\mathbf{x} - \mathbf{x}_1\|^2}$,
- the visibility factor $V_l(\mathbf{x}_1) = V(l.\mathbf{x} \leftrightarrow \mathbf{x}_1)$ and
- the intensity factor $I_l = L(l.\mathbf{x})$.

Assume that we have the upper bounds $M_l(\mathbf{x}_1) \leq \hat{M}_D(\mathbf{x}_1)$, $G_l(\mathbf{x}_1) \leq \hat{G}_D(\mathbf{x}_1)$ and $V_l(\mathbf{x}_1) \leq \hat{V}_D(\mathbf{x}_1)$. In this case,

$$\begin{aligned}S_{\bar{\mathbf{x}}}(D) = \sum_{l \in D} M_l(\mathbf{x}_1)G_l(\mathbf{x}_1)V_l(\mathbf{x}_1)I_l \quad & \leq \hat{M}_D(\mathbf{x}_1)\hat{G}_D(\mathbf{x}_1)\hat{V}_D(\mathbf{x}_1)\sum_{l \in D} I_l \\ = \hat{M}_D(\mathbf{x}_1)\hat{G}_D(\mathbf{x}_1)\hat{V}_D(\mathbf{x}_1)L(D) &= \hat{S}_{\bar{\mathbf{x}}}(D).\end{aligned}$$

It is easy to show that $\hat{S}_{\bar{\mathbf{x}}}(D)$ is an upper bound for $\tilde{S}_{\bar{\mathbf{x}}}(D)$ as well.

The visibility factor has the trivial upper bounds $\hat{V}_D(\mathbf{x}_1) = 1$. Walter et al. show how derive upper bounds for the geometry factor as well as the material factor for diffuse materials. They also derive upper bounds for the material factor for a couple of glossy BRDFs.

Comparison. We use ideas similar to Lightcuts in our clustered hierarchical importance sampling method. We also use a light hierarchy to approximate the total radiance arriving from a cluster of point lights represented by a node in the hierarchy. See Chapter 8 for more details. The main difference between our approaches

is that Lightcuts uses the hierarchy to generate error-bounded images, while our method uses it to importance sample from the point lights. This makes our method able to do incremental rendering, i.e. the ability to indefinitely improve an image. In contrast, the only way to improve the quality of a image rendered using Lightcuts is to re-render it using a smaller error bound.

7.3 Metropolis Instant Radiosity

Metropolis Instant Radiosity (MIR) by Segovia et al. [2007] is a sampling algorithm for light paths[8] that distributes them according to how much indirect light they bring to the camera. More precisely, the resulting probability density of a light path is proportional to the total response it causes for all the sensors in an image.

Segovia et al. sample light paths with the help of the *Metropolis Light Transport (MLT)* algorithm by Veach [1998]. MLT is an application of the Metropolis-Hastings sampling method to light transport paths. It generates light transport paths $(\bar{\mathbf{z}}_1, \ldots, \bar{\mathbf{z}}_k)$ such that their probability densities with respect to the product area measure are proportional to the image contribution function

$$g(\bar{\mathbf{z}}) = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} f_S(\bar{\mathbf{z}}),$$

where f_S is the measurement response function for the sensor S and $\mathcal{S}$ is the family of all sensors in the image. More precisely,

$$p(\bar{\mathbf{z}}_i) = \frac{g(\bar{\mathbf{z}}_i)}{\int\limits_{\mathcal{P}} g(\bar{\mathbf{z}}) \, \mathrm{d}\mathcal{A}(\bar{\mathbf{z}})}$$

for all $1 \leq i \leq n$.

Recall from chapter 5.4 that we can represent each transport path $\bar{\mathbf{z}}$ as camera path $\bar{\mathbf{x}}$ that is connected to a light path $\bar{\mathbf{y}}$. Using this, Segovia et al. express the probability density $p(\bar{\mathbf{z}})$ as the joint density $p(\bar{\mathbf{y}}, \bar{\mathbf{x}})$. The probability density of only the light path $\bar{\mathbf{y}}$ is the marginal

$$p(\bar{\mathbf{y}}) = \int\limits_{\mathcal{P}^2} p(\bar{\mathbf{y}}, \bar{\mathbf{x}}) \, \mathrm{d}\mathcal{A}^2(\bar{\mathbf{x}}) = \frac{1}{\int\limits_{\mathcal{P}} g(\bar{\mathbf{z}}) \, \mathrm{d}\mathcal{A}(\bar{\mathbf{z}})} \int\limits_{\mathcal{P}^2} g(\bar{\mathbf{y}}, \bar{\mathbf{x}}) \, \mathrm{d}\mathcal{A}^2(\bar{\mathbf{x}})$$

[8] In the original formulation of the algorithm, Segovia et al. employed the radiosity assumption and used VPLs instead of light paths. Here, we present the algorithm without assuming diffuse materials and therefore view it as a sampling method for light paths.

which in fact is proportional to the total response caused by the light path $\bar{\mathbf{y}}$. In other words, *light paths can be sampled from the desired distribution by sampling light transport paths using MLT and dropping the last two points that make up the camera path.*

These light paths cannot directly be used for Instant Radiosity rendering, as the PDF $p(\bar{\mathbf{y}})$ does not admit a closed-form expression. Instead, Segovia et al. estimate the probability densities by noting that the unnormalized PDF

$$g(\bar{\mathbf{y}}) = \int_{\mathcal{P}^2} g(\bar{\mathbf{y}}, \bar{\mathbf{x}}) \, \mathrm{d}\mathcal{A}^2(\bar{\mathbf{x}}) = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} \int_{\mathcal{P}^2} f_S(\bar{\mathbf{y}}, \bar{\mathbf{x}}) \, \mathrm{d}\mathcal{A}^2(\bar{\mathbf{x}})$$

is the average of the sensor responses in an image rendered using only the illumination from $\bar{\mathbf{y}}$. In order to estimate the full normalized PDF, they make use of an alternative rendering method to estimate the denominator of normalizing constant

$$C = \int_{\mathcal{P}} g(\bar{\mathbf{z}}) \, \mathrm{d}\mathcal{A}(\bar{\mathbf{z}}).$$

The final method is summarized by Algorithm 6. The lines $2 - 3$ sample light paths from the desired distribution using the method described above. The function SAMPLECAMERAPATH generates a camera path using the sampling method in Chapter 5.4 and $p(\bar{\mathbf{x}})$ expresses its probability density with respect to the product area measure. Computation is saved in the inner for loops by using the same camera paths to both estimate the probability density of a light path and render the image. In practice, the lines $11 - 12$ adds terms to the Monte Carlo estimators for each sensor by rescaling the contributions of the transport paths that were previously used to estimate the probability density of the light path.

Comparison. In Chapter 9 we introduce a method that also importance samples light paths according to the sensor responses they cause. While MIR samples light paths in a provably optimal fashion, it does not address the issue of temporal coherence. It cannot ensure that the distribution of light paths are similar between two images in a sequence. In contrast, our method uses heuristic sampling to minimize the number of light paths that change between images. While our method cannot perfectly match the quality of MIR for single images, we show in Chapter 10.3 the error of our method fluctuates significantly less than that of MIR.

Algorithm 6 Metropolis Instant Radiosity

$\tilde{C} \leftarrow$ estimate of $\int_{\mathcal{P}} g(\bar{\mathbf{z}})\,\mathrm{d}\mathcal{A}(\bar{\mathbf{z}})$.
$(\bar{\mathbf{z}}_1, \ldots, \bar{\mathbf{z}}_N) \leftarrow$ transport paths sampled using MLT.
$(\bar{\mathbf{y}}_1, \ldots, \bar{\mathbf{y}}_N) \leftarrow$ light paths formed from $(\bar{\mathbf{z}}_1, \ldots, \bar{\mathbf{z}}_N)$.
for each light path $\bar{\mathbf{y}}$ **do**
    $\tilde{p} \leftarrow 0$
    for each sensor S **do**
        $\bar{\mathbf{x}} \leftarrow$ SAMPLECAMERAPATH()
        $\text{unscaledMeasurement}[S] \leftarrow \dfrac{f_S(\bar{\mathbf{y}}, \bar{\mathbf{x}})}{p(\bar{\mathbf{x}})}$
        $\tilde{p} \leftarrow \tilde{p} + \dfrac{1}{M}\text{unscaledMeasurement}[S]$
    $\tilde{p} \leftarrow \dfrac{\tilde{p}}{\tilde{C}}$
    for each sensor S **do**
        $\text{measurement}[S] \leftarrow \dfrac{1}{N}\dfrac{\text{unscaledMeasurement}[S]}{\tilde{p}}$

7.4 Incremental Instant Radiosity

Incremental Instant Radiosity (IIR) by Laine et al. [2007] is a sequential Monte Carlo method that employs heuristic sampling and density estimation to move as few VPLs between images in a sequence as possible. Since the algorithm makes use of rastersation to resolve visibility, moving fewer VPLs not only reduces the "flickering"-effect but also increases rendering speed. This is because the visibility information acquired by rasterisation does not need to be re-evaluated for VPLs that haven't moved.

The algorithm makes use of the radiosity assumption and is designed to simulate single-bounce indirect illumination from a point-shaped light source. In other words, the light paths contain no more than two points and the primary light source is either a point light or an omni light (a point-shaped light source that emits the same radiance over the entire sphere of directions). Here we present a more general version of the algorithm that lifts the radiosity assumption by sampling light paths instead of VPLs. We restrict our treatment to the case where the primary light is a point light, but the resulting algorithm can be extended to the omni-light case in a straightforward fashion.

Under these conditions, each light path $\bar{\mathbf{y}}$ is a pair of points $(\mathbf{y}_1, \mathbf{y}_2)$ where $\mathbf{y}_1$ always coincides with the location of the primary light $\mathbf{z}$. As a result, a collection of light

paths

$$(\bar{\mathbf{y}}_1, \ldots, \bar{\mathbf{y}}_N) = ((\mathbf{z}, \mathbf{y}_1), \ldots, (\mathbf{z}, \mathbf{y}_N))$$

is uniquely defined by the points $(\mathbf{y}_1, \ldots, \mathbf{y}_N)$ on the surfaces visible to $\mathbf{z}$. The goal of the sampling method is to maintain a set of these points, such that the points we get by projecting the directions

$$\omega(\mathbf{z} \to \mathbf{y}_i) \, \forall \, 1 \leq i \leq N$$

onto the tangent plane of $\mathbf{z}$ should be as evenly placed as possible. Note that the projected points all lie on the unit disk $\mathcal{D}(\mathbf{z})$ in the local coordinate space of $\mathbf{z}$. Laine et al. ensure even placement by sampling the light paths in a fashion that minimizes the *dispersion metric*, which is the area of the largest circle in $\mathcal{D}(\mathbf{z})$ that doesn't contain any of the projected directions.

Let $(\mathbf{y}_1^\perp, \ldots, \mathbf{y}_N^\perp)$ be the resulting points in $\mathcal{D}(\mathbf{z})$ from projecting the directions $(\omega(\mathbf{z} \to \mathbf{y}_1), \ldots, \omega(\mathbf{z} \to \mathbf{y}_N))$ onto the tangent plane of $\mathbf{z}$. The method makes use of the *Voronoi diagram* of the points $\mathcal{D}(\mathbf{z})$ to both maintain the set of light paths and estimate their probability densities. The Voronoi diagram splits the unit disk into regions containing one point each, such that the point closest to any location in a region is the point inside the region. More precisely, the Voronoi region that contains the point $\mathbf{y}_i^\perp$ is defined as

$$V(\mathbf{y}_i^\perp) = \left\{ \mathbf{x} \in \mathcal{D}(\mathbf{z}) \mid \underset{1 \leq j \leq N}{\arg\min}(\|\mathbf{x} - \mathbf{y}_j^\perp\|) = i \right\}.$$

Intuitively, the points can be considered as post offices and the Voronoi regions can be thought of as post code areas. That is, the post code area you're in uniquely determines your nearest post office.

The algorithm minimizes the dispersion metric by progressively moving the point with the smallest Voronoi region in $\mathcal{D}(\mathbf{z})$ to the centre of the largest empty circle, which also can be efficiently located using the Voronoi diagram. The new location of $\mathbf{y}_i$ on a light path that has been moved is determined using the ray casting function and the function $\omega_\mathbf{z}^\perp : \mathcal{D}(\mathbf{z}) \to H^+(\mathbf{z})$ that maps points on the unit disk to their corresponding directions. More precisely,

$$\mathbf{y}_i = r(\mathbf{z}, \omega_\mathbf{z}^\perp(\mathbf{y}_i^\perp)).$$

Before rendering the image, the probability densities of the light paths with respect to the projected solid angle measure $\sigma_\mathbf{z}^\perp$ are estimated as

$$\hat{p}(\mathbf{y}_i) = \frac{1}{N} \sigma_\mathbf{z}^\perp(\omega_\mathbf{z}^\perp(V(\mathbf{y}_i^\perp))) = \frac{1}{N} A(V(\mathbf{y}_i^\perp)).$$

See Algorithm 7 for pseudo-code describing this method. In order to ensure temporal coherence, the algorithm doesn't allow more than $recalcMax$ light paths to change location between two frames. On the other hand, the algorithm also ensures that at least $recalcMin$ light paths move between two frames in order keep minimizing the dispersion measure. On line 19 we denote the measurement response for the sensor $S \subset \mathcal{M}$ by f_S.

Algorithm 7 Incremental Instant Radiosity

1: $N \leftarrow$ maximum light path budget
2: $\mathcal{Y} \leftarrow N$ points on the surfaces visible to $\mathbf{z}$.
3: **for** each frame **do**
4: Update the position and orientation of $\mathbf{z}$.
5: $\mathcal{V} \leftarrow$ the points in $\mathcal{Y}$ still visible to $\mathbf{z}$
6: **while** $|\mathcal{Y}| - |\mathcal{V}| < recalcMin$ **do**
7: $\mathbf{y} \leftarrow \arg\min_{\mathbf{y} \in \mathcal{V}} A(V(\mathbf{y}^{\perp}))$
8: $\mathcal{V} \leftarrow \mathcal{V} \setminus \{\mathbf{y}\}$
9: **for** $i = 1$ to $\min(N - |\mathcal{V}|, recalcMax)$ **do**
10: $\mathbf{y}^{\perp} \leftarrow$ centre of largest empty circle in $\mathcal{D}(\mathbf{z})$
11: $\mathbf{y} \leftarrow r(\mathbf{z}, \omega_{\mathbf{z}}^{\perp}(\mathbf{y}^{\perp}))$
12: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{y}\}$
13: $\mathcal{Y} \leftarrow \mathcal{V}$
14: **for** each $\mathbf{y}$ in $\mathcal{Y}$ **do**
15: $\bar{\mathbf{y}} \leftarrow (\mathbf{z}, \mathbf{y})$
16: $\tilde{p} \leftarrow \dfrac{\mathrm{d}A(\mathbf{y})}{\mathrm{d}\sigma_{\mathbf{z}}^{\perp}(\omega(\mathbf{z} \to \mathbf{y}))} \dfrac{1}{|\mathcal{Y}|} A(V(\mathbf{y}^{\perp}))$
17: **for** each sensor S **do**
18: $\bar{\mathbf{x}} \leftarrow$ sampleCameraPath()
19: measurement$[S] \leftarrow \dfrac{1}{|\mathcal{Y}|} \dfrac{f_S(\bar{\mathbf{y}}, \bar{\mathbf{x}})}{p(\bar{\mathbf{x}})\tilde{p}}$

If we interpret the algorithm using the heuristic sampling framework presented in Chapter 3, we see that lines 5-13 correspond to the HeuristicSampling function and line 16 corresponds to the EstimateDensities function. The estimate for the integral is formed by rendering the image on the lines 14-19.

Comparison. At a high level, IIR is very similar to sequential Monte Carlo Instant Radiosity which we introduce in Chapter 9. Both methods employ heuristic sampling

and density estimation to move fewer light paths between images in a sequence. However, while our method works on light paths with an arbitrary length, IIR requires them to contain exactly two points. This limits IIR to single-bounce indirect illumination, a restriction our method does not impose.

While IIR importance samples the light paths according to the radiance they receive from the light source, it does not account for the location of the camera. In contrast, our method importance samples light paths according to the sensor response they cause, which makes it able to cope with large and heavily occluded scenes, where only a few light sources significantly contribute to the image.

8 Clustered hierarchical importance sampling

In this chapter, we present a new many lights rendering method. It employs Monte Carlo estimators to render pictures without the need to explicitly account for the illumination received by all camera paths from each of the point lights in the scene.

Recall from Chapter 7.2 that we can express the exitant radiance along the camera path $\bar{\mathbf{x}} = (\mathbf{x}_1, \mathbf{x}_2)$ from $\mathbf{x}_1$ towards $\mathbf{x}_2$ on the sensor as the sum

$$S(D) = \sum_{l \in D} L_e(l.\mathbf{x} \to \mathbf{x}_1) G(l.\mathbf{x} \leftrightarrow \mathbf{x}_1) f^s(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2)$$

over all the point lights in the set D. We estimate $S(D)$ using the unbiased Monte Carlo estimator

$$\tilde{S}^N(D) = \frac{1}{N} \sum_{i=1}^{N} \frac{L_e(l_i.\mathbf{x} \to \mathbf{x}_1) G(l_i.\mathbf{x} \leftrightarrow \mathbf{x}_1) f^s(l_i.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2)}{p(l_i)},$$

where the outcomes $(l_1, \ldots, l_N)$ are sampled from a discrete random variable $\mathcal{L} : \Omega_{\mathcal{L}} \to D$. The remainder of this chapter is dedicated to reducing the variance of this estimator using importance sampling.

8.1 Ideal importance sampler

Recall from Chapter 2 that we achieve the lowest possible variance by sampling the point lights $(l_1, \ldots, l_N)$ from a discrete variable $\mathcal{L}_o : \Omega_{\mathcal{L}_o} \to D$ whose *probability mass function*)(PMF) is proportional to the exitant radiance caused by the point lights. That is,

$$p_{\mathcal{L}_o}(l) = \frac{L_e(l.\mathbf{x} \to \mathbf{x}_1) G(l.\mathbf{x} \leftrightarrow \mathbf{x}_1) f^s(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2)}{S(D)}$$

for all $l \in D$. The problem with this approach is that the denominator of the PMF is the sum $S(D)$ that we're attempting to estimate in the first place.

A less expensive method to is to sample the outcomes from a variable $\mathcal{L}_v : \Omega_{\mathcal{L}_v} \to D$ whose PMF is proportional to the exitant radiance caused by the point lights assuming that they are visible to $\mathbf{x}_1$. In this case,

$$p_{\mathcal{L}_v}(l) = \frac{L_e(l.\mathbf{x} \to \mathbf{x}_1)\dfrac{\mathrm{d}A(l.\mathbf{x})}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}_1}(\omega(\mathbf{x}_1 \to l.\mathbf{x}))} f^s(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2)}{\displaystyle\sum_{l' \in D} L_e(l'.\mathbf{x} \to \mathbf{x}_1)\frac{\mathrm{d}A(l'.\mathbf{x})}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}_1}(\omega(\mathbf{x}_1 \to l'.\mathbf{x}))} f^s(l'.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2)}$$

since $G(l.\mathbf{x} \leftrightarrow \mathbf{x}_1) = V(l.\mathbf{x} \leftrightarrow \mathbf{x}_1)\dfrac{\mathrm{d}A(l.\mathbf{x})}{\mathrm{d}\sigma^{\perp}_{\mathbf{x}_1}(\omega(\mathbf{x}_1 \to l.\mathbf{x}))}$ and our assumption implies $V(l.\mathbf{x} \leftrightarrow \mathbf{x}_1) = 1$.

Even though we save computation by not having to evaluate the ray casting function to compute the PMF, this sampling method is not feasible for real-time rendering as the resulting algorithm is linear in both number of point lights and number of camera paths. For example, to render a full HD image (with a resolution of 1920×1080) that is illuminated by 32000 point lights, the denominator of the PMF forces us to compute the exitant radiance caused by a point light over 60 billion times, even if we only sample one transport path per pixel.

8.2 Light hierarchy

We reduce the computation required for sampling the point lights with the help of a light hierarchy similar to the one used in Lightcuts [Walter et al., 2005]. In our case, the light hierarchy is a bounding volume hieararchy (BVH, see Chapter 6) of the point light positions as well as a bounding cone hierarchy (BCH, also in Chapter 6) of the point light normals. At each node in the hierarchy we also store other aggregate information, such as the total intensity of the point lights it represents.

Using this hierarchy, we design a sampling algorithm whose running time is logarithmic with respect to the number of point lights, provided that the hierarchy is balanced. In the example above, this would reduce number of times we need to compute exitant radiance by three orders of magnitude.

Hierarchical importance sampling. Our sampling method generates outcomes by performing a random traversal from the root of the hierarchy down to a point

Algorithm 8 Hierarchical importance sampling

```
1: p ← 1
2: D ← root of hierarchy
3: while D is not leaf do
4:     p_0 ← estimateRadiance(D[0])
5:     p_1 ← estimateRadiance(D[1])
6:     total ← p_0 + p_1
7:     p_0 ← p_0 / total
8:     p_1 ← p_1 / total
9:     u ← outcomeFrom(U_[0,1])
10:    if u ≤ p_0 then
11:        D ← D[0]
12:        p ← p_0 p
13:    else
14:        D ← D[1]
15:        p ← p_1 p
16: return (D.l, p)
```

light in one of the leaves. See Algorithm 8 for pseudo-code. At each node D in the hierarchy, the method makes the binary choice between the two sub-trees $D[0]$ and $D[1]$. These choices are made by sampling an outcome from a binary variable $B_D : \Omega_{B_D} \to 0, 1$, such that the probability of 0 or 1 is determined by the estimated exitant radiance caused by the corresponding sub-trees. Once a leaf D is reached, the algorithm returns the only point light $D.l$ represented by D and the probability p of choosing it. With this method, the probability of choosing a point light is the product of the probabilities for the choices made when descending the hierarchy.

In order to sample point lights using this algorithm, we need to implement the function estimateRadiance. That is, we need to estimate the exitant radiance caused by a node D in the hierarchy. More precisely, we need to efficiently estimate the sum $S(D)$ without explicitly computing exitant radiance caused by each point light represented by D.

Factoring the exitant radiance. Similarly to the upper bound employed in the Lightcuts method Walter et al. [2005], we also derive our estimate by decomposing the exitant radiance into factors. Recall from Chapter 7.2 that we can write the

exitant radiance caused by the point lights represented by the hierarchy node D as

$$S(D) = \sum_{l \in D} L(l.\mathbf{x}) V(l.\mathbf{x} \leftrightarrow \mathbf{x}_1) f^r(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2) \\ \frac{\max(0, N(\mathbf{x}_1) \cdot \omega(\mathbf{x}_1 \to l.\mathbf{x})) \max(0, N(l.\mathbf{x}) \cdot \omega(l.\mathbf{x} \to \mathbf{x}_1))}{\|l.\mathbf{x} - \mathbf{x}_1\|^2}$$

if we assume opaque materials. We decompose each term in this expression into five factors:

- the light cosine factor $C_{\bar{\mathbf{x}}}(l) = \max(0, N(l.\mathbf{x}) \cdot \omega(l.\mathbf{x} \to \mathbf{x}_1)),$,
- the material factor $M_{\bar{\mathbf{x}}}(l) = f^r(l.\mathbf{x} \to \mathbf{x}_1 \to \mathbf{x}_2) \max(0, N(\mathbf{x}_1) \cdot \omega(\mathbf{x}_1 \to l.\mathbf{x}))$,
- the distance factor $D_{\bar{\mathbf{x}}}(l) = \dfrac{1}{\|l.\mathbf{x} - \mathbf{x}_1\|^2}$,
- the visibility factor $V_{\bar{\mathbf{x}}}(l) = V(l.\mathbf{x} \leftrightarrow \mathbf{x}_1))$ and
- the intensity factor $I_{\bar{\mathbf{x}}}(l) = L(l.\mathbf{x})$.

The goal is to form an estimate ESTIMATERADIANCE(D) of $S(D)$ that can be evaluated without inspecting all the point lights represented by D. We achieve this by replacing $C_{\bar{\mathbf{x}}}(l), M_{\bar{\mathbf{x}}}(l), D_{\bar{\mathbf{x}}}(l), V_{\bar{\mathbf{x}}}(l)$ with estimates $\tilde{C}_{\bar{\mathbf{x}}}(D), \tilde{M}_{\bar{\mathbf{x}}}(D), \tilde{D}_{\bar{\mathbf{x}}}(D), \tilde{V}_{\bar{\mathbf{x}}}(D)$ that do not depend on any of the light sources in D, but instead represent averages of these factors for all lights in D. In this case,

$$\begin{aligned} \text{ESTIMATERADIANCE}(D) &= \sum_{l \in D} \tilde{C}_{\bar{\mathbf{x}}}(D) \tilde{M}_{\bar{\mathbf{x}}}(D) \tilde{D}_{\bar{\mathbf{x}}}(D) \tilde{V}_{\bar{\mathbf{x}}}(D) I_{\bar{\mathbf{x}}}(l) \\ &= \tilde{C}_{\bar{\mathbf{x}}}(D) \tilde{M}_{\bar{\mathbf{x}}}(D) \tilde{D}_{\bar{\mathbf{x}}}(D) \tilde{V}_{\bar{\mathbf{x}}}(D) \sum_{l \in D} I_{\bar{\mathbf{x}}}(l) \\ &= \tilde{C}_{\bar{\mathbf{x}}}(D) \tilde{M}_{\bar{\mathbf{x}}}(D) \tilde{D}_{\bar{\mathbf{x}}}(D) \tilde{V}_{\bar{\mathbf{x}}}(D) \hat{I}_{\bar{\mathbf{x}}}(D), \end{aligned}$$

which can be evaluated without summing over the point lights represented by D since we can precompute the sum over all the intensity factors $\hat{I}_{\bar{\mathbf{x}}}(D) = \sum_{l \in D} I_{\bar{\mathbf{x}}}(l)$.

In contrast to the Lightcuts method, that requires an upper bound for $S(D)$, our method can work under more lenient restrictions, since we're using the estimate of $S(D)$ to importance sample point lights instead of providing a strict error bound. In order for our hierarchical sampling method to be unbiased, the only restriction for the estimate is for it to be non-zero whenever $S(D) > 0$.

Ideally, our estimates should be defined such that ESTIMATERADIANCE equals $S(D)$. However, if this was the case, we would already know the total exitant radiance caused by the root R of the hierarchy, that represents all the point lights, and wouldn't need to estimate it using $\tilde{S}^N(R)$. Instead, we settle for a easier goal that $S(D)$ should equal ESTIMATERADIANCE whenever $D = \{l\}$ is a singleton set and that the approximation $\tilde{S}_F(D)$ degrades gracefully when the number of point lights represented by D increases.

Distance factor. The distance factor has the largest impact on the exitant radiance as it is the only factor which is not bounded from above. In fact, as the distance between the point light $l.\mathbf{x}$ and the point $\mathbf{x}_1$ on the camera path grows smaller the distance factor will tend to infinity.

One way to approximate the distance factor is to make use of the bounding volume $BV(D)$ to form an upper bound. More precisely,

$$\tilde{D}_{\bar{\mathbf{x}}}(D) = \frac{1}{\min\limits_{\mathbf{y} \in BV(D)} (\|\mathbf{y} - \mathbf{x}_1\|^2)}.$$

However, this approximation does not consider the positions of the point lights inside the bounding volume. Since the approximation tends to infinity as the distance between the bounding volume and $\mathbf{x}_1$ approaches zero, it might very well overestimate the contribution from D to such extents that Algorithm 8 deterministically chooses D over its neighbouring sub-tree. Even worse, this approximation is undefined when $\mathbf{x}_1$ is inside $BV(D)$, as it leads to $\min\limits_{\mathbf{y} \in BV(D)} (\|\mathbf{y} - \mathbf{x}_1\|^2) = 0$.

These problems can be avoided if we define $\tilde{D}_{\bar{\mathbf{x}}}(D)$ as the *harmonic mean* of distance term. In other words,

$$\tilde{D}_{\bar{\mathbf{x}}}(D) = \frac{|D|}{\sum\limits_{l \in D} D_{\bar{\mathbf{x}}}(l)^{-1}} = \frac{|D|}{\sum\limits_{l \in D} \|l.\mathbf{x} - \mathbf{x}_1\|^2}.$$

This approximation does consider the positions of all the point lights in D and is well defined even when $\mathbf{x}_1$ is inside the bounding volume. In fact, the only possibility for the denominator to be zero is if $\mathbf{x}_1$ coincides with the position of all the point lights in D.

We can compute the harmonic mean efficiently by noting that its inverse can be decomposed as follows

$$\tilde{D}^{-1}_{\bar{\mathbf{x}}}(D) = \frac{1}{|D|} \sum_{l \in D} (\|l.\mathbf{x}\|^2) - 2(\frac{1}{|D|} \sum_{l \in D} l.\mathbf{x}^T)\mathbf{x}_1 + \|\mathbf{x}_1\|^2.$$

We exploit this by precomputing $A_D = \frac{1}{|D|}\sum_{l\in D}(\|l.\mathbf{x}\|^2)$ and $B_D = \frac{1}{|D|}\sum_{l\in D} l.\mathbf{x}^T$ for each node D in the hierarchy. Now, for any camera path $\bar{\mathbf{x}}$, we can evaluate

$$\tilde{D}_{\bar{\mathbf{x}}}(D) = (A_D - 2B_D\mathbf{x}_1 + \|\mathbf{x}_1\|^2)^{-1}$$

without having iterate over the point lights represented by D.

Dot products. We form an estimate for the material factor using the radiosity assumption. Under this assumption the material factor is of the form

$$M_{\bar{\mathbf{x}}}(l) = \frac{c_{\mathbf{x}_1}}{\pi}\max(0, N(\mathbf{x}_1)\cdot\omega(\mathbf{x}_1 \to l.\mathbf{x})),$$

for some constant $c_{\mathbf{x}_1} \in [0,1]$. That said, it is possible to extend the importance sampling method described here to arbitrary BRDF models by deriving appropriate approximations for the corresponding material factors.

Since the light cosine factor is also a clamped dot product, we estimate both of these factors using the the same mechanism. As the value of these clamped dot products only assume values in $[0,1]$, the impact they make on the final estimate is not as large as the distance factor. However, these factors play an important role since they inform the sampling algorithm about scenarios where the exitant radiance caused by D has to be zero. This can occur when D is fully behind the surface where $\mathbf{x}_1$ is located, in other words $M_{\bar{\mathbf{x}}}(l) = 0$ for all $l \in D$. Another such scenario is when all the point lights in D are oriented away from $\mathbf{x}_1$, more precisely $C_{\bar{\mathbf{x}}}(l) = 0$ for all $l \in D$.

A representative estimate for these factors should be zero in the scenarios described above. However, for the sampling algorithm to remain unbiased, we also have to make sure that the estimate is never zero when the actual factor is non-zero. Keeping this in mind, we base our estimate on an upper bound of the clamped dot product.

In order to form an upper bound for a dot product of the form

$$N(\mathbf{x}_1)\cdot\omega(\mathbf{x}_1 \to l.\mathbf{x})$$

we exploit the fact that the possible positions for $l.\mathbf{x}$ are in the bounding volume $BV(D)$. Recall from Chapter 6 that we use AABBs as our bounding volumes. As a consequence, we can express any point $\mathbf{y} \in BV(D)$ in terms of its centre $BV(D)_c$ and diagonal vector $BV(D)_d$. In other words,

$$\mathbf{y} = BV(D)_c + a \odot BV(D)_d$$

where $a \in [-1,1]^3$ and $\odot$ represents element-wise vector multiplication. Given an arbitrary point $\mathbf{y} \in BV(D)$, we see that the dot product

$$\begin{aligned} N(\mathbf{x}_1) \cdot \omega(\mathbf{x}_1 \to \mathbf{y}) &= \frac{N(\mathbf{x}_1) \cdot (BV(D)_c + a \odot BV(D)_d - \mathbf{x}_1)}{|\mathbf{y} - \mathbf{x}_1|} \\ &\leq \frac{N(\mathbf{x}_1) \cdot (BV(D)_c - \mathbf{x}_1) + |N(\mathbf{x}_1)| \cdot BV(D)_d}{\min\limits_{\mathbf{y}' \in BV(D))} |\mathbf{y}' - \mathbf{x}_1|}. \end{aligned}$$

The last step forms an upper bound by setting the denominator to the minimum distance over all potential points within the bounding volume. This causes similar problems as bounding volume based upper bound for distance factor we described above. Since the denominator does not affect whether our approximation is zero or not, we are free to estimate it using other methods without introducing bias. As we have already have a reasonable estimate $\tilde{D}_{\bar{\mathbf{x}}}(D)$ for the distance term, and

$$\sqrt{D_{\bar{\mathbf{x}}}(l)} = \frac{1}{|\mathbf{l}.\mathbf{x} - \mathbf{x}_1|}$$

we use $\sqrt{\tilde{D}_{\bar{\mathbf{x}}}(D)}$ to estimate the denominator.

We directly apply the results from above to form the estimate

$$\tilde{M}_{\bar{\mathbf{x}}}(l) = \frac{c_{\mathbf{x}_1}}{\pi} \sqrt{\tilde{D}_{\bar{\mathbf{x}}}(D)} \max(0, N(\mathbf{x}_1) \cdot (BV(D)_c - \mathbf{x}_1) + |N(\mathbf{x}_1)| \cdot BV(D)_d)$$

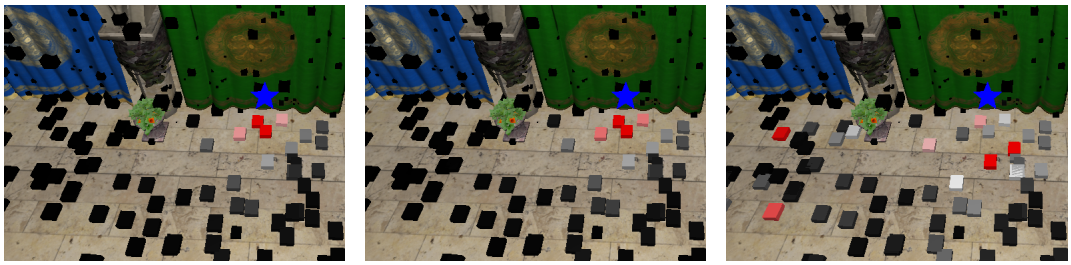
for the material term. We use the results to also form the estimate

$$\Phi_{\bar{\mathbf{x}}}(D) = \sqrt{\tilde{D}_{\bar{\mathbf{x}}}(D)}(\omega_D \cdot (\mathbf{x}_1 - BV(D)_c) + |\omega_D| \cdot BV(D)_d)$$

for the dot product between the centre vector ω_D of the bounding cone for D and the direction $\omega(l.\mathbf{x} \to \mathbf{x}_1)$. In order to estimate the light cosine factor, we also need to account for the bounding cone of D. We do this by estimating the cosine of the angle between the bounding cone and the direction in the same fashion as the Lightcuts method [Walter et al., 2005]. In other words, if θ_D is the cosine threshold for the bounding cone of D, then

$$\tilde{C}_{\bar{\mathbf{x}}}(D) = \begin{cases} 1 & \text{if } \Phi_{\bar{\mathbf{x}}}(D) \geq \theta_D \\ \max(0, \cos(\cos^{-1}(\Phi_{\bar{\mathbf{x}}}(D)) - \cos^{-1}(\theta_D))) & \text{otherwise.} \end{cases}$$

Visibility factor. As approximating the visibility is computationally expensive, we lift the restriction that ESTIMATERADIANCE$(D) = S(D)$ when D is a singleton set to ensure that the estimate $\tilde{S}^N(D)$ is unbiased. We use the same approximation $\tilde{V}_{\mathbf{x}}(D) = 1$ also employed in the Lightcuts method, which clearly does not introduce bias as it ensures that $\tilde{V}_{\mathbf{x}}(D)$ is non-zero when $S(D) > 0$.



(a) Ideal importance sampler.

(b) Hierarchical importance sampler with camera path clusters.

(c) Hierarchical importance sampler.

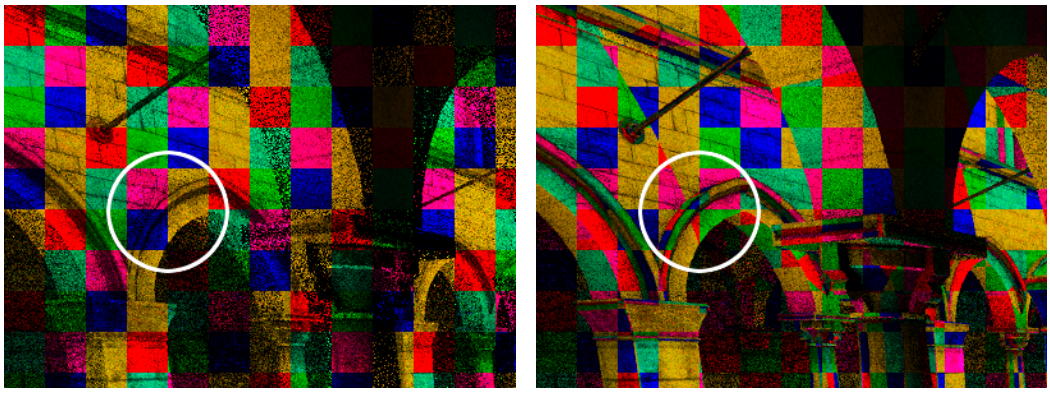
Figure 17: A comparison of the probabilities the sampling techniques assign to the point lights. The goal is to sample point lights (represented by boxes) for the camera path represented by the blue star. Ideally, point lights that bring more light to the star should have a higher probability of being sampled. The probabilities are colour coded, where red represents a high probability and black represents a low probability. The intensity of a point light is visualized by the size of its box. As we can see, introducing camera path clusters significantly improves upon the probabilities of the hierarchical importance sampler.

8.3 Camera path clusters

Even though the sampling algorithm described above allows us to rapidly importance sample point lights, as we can see from Figure 17 it still has room for improvement. Recall that the ideal probability of sampling a point light is proportional to the exitant radiance it causes. Ignoring visibility, the estimate $\tilde{S}_F(D)$ is correct when D is singleton set and degrades as amount of elements in D increases. As such, the first choice made by Algorithm 8 is made with the lowest quality estimate of $S(D)$. However, this choice is often the most important one, as it excludes the largest amount of point lights from consideration.

Here we present a solution to this problem that doesn't make use of $\tilde{S}_F$ in the upper levels of the light hierarchy. Instead, we aim to start Algorithm 8 from a node deeper down in the hierarchy instead of the root. We achieve this by sampling the first node D from a cut of the hierarchy.

Forming clusters. Note that both the ideal importance sampling algorithm and Algorithm 8 can be described this way. The ideal importance sampling algorithm starts Algorithm 8 from a cut of the hierarchy that contains all of the leaf nodes.



(a) Clusters that only consider screen-space positions.

(b) Clusters that also account for normals and depth.

Figure 18: Visualisation of how different implementations of the identifier I forms clusters of the camera paths. The colour tint of a pixel indicates which cluster it belongs to. Note how accounting for normals and depth appropriately separates the pixels belonging to different parts of the arch.

Conversely, when the cut contains only the root of the hierarchy, the method described above simplifies to Algorithm 8 alone. This implies a trade-off between computational complexity and the quality of the probabilities for sampling the point lights. We circumvent this trade-off by sharing the same cut between similar camera paths.

In practice, we achieve this by forming clusters from similar camera paths and computing a cut for each cluster rather than each of the paths. We form these clusters using a method similar to the one described by Olsson et al. [2012]. In other words, we map each camera path $\bar{\mathbf{x}} = (\mathbf{x}_1, \mathbf{x}_2)$ to a identifier $I(\bar{\mathbf{x}})$ by quantizing the position and normal of $\mathbf{x}_1$ into a binary representation. Each identifier represents a cluster and we form cuts for the clusters that contain at least one camera path.

Clearly, the function I determines how the clusters are shaped, see Figure 18. Ideally, we'd like the clusters to contain camera paths that react to incoming light in a similar fashion. We define I in terms of two functions $F^N : \mathbb{R}^3 \to \{0,1\}^N$, which quantizes positions into a binary representation with N bits, and $G^M : \mathbb{S}^2 \to \{0,1\}^M$, that quantizes normals into a binary representation with M bits. We then define $I(\bar{\mathbf{x}})$ as the concatenation of $F^N(\mathbf{x}_1)$ and $G^M(N(\mathbf{x}_1))$.

We define F^N using the pixel coordinates of the camera path and the depth of $\mathbf{x}_1$ relative to the screen. The number of bits we use to represent the pixel coordinates

is a tunable constant, whereas we consistently use 5 bits to represent the depth in our experiments. In order to account for perspective, we quantize the cube root of the depth in favour of a linear quantization.

We define G^M by linearly quantizing the coordinates of the normal into a K-bit representation. More precisely, $G^M(N(\mathbf{x}_1))$ is formed by concatenating the K-bit binary representations for the coordinates of $N(\mathbf{x}_1)$. In our experiments we use two bits per coordinate, that is $K = 2$ and $M = 6$.

Forming the cuts. We construct the cuts in a top-down fashion. We start the process with the minimal cut that only contains the root of the hierarchy. We then continue in breadth-first order to replace nodes in the cut with their two descendants until the maximum node count is met. We guide this process using a heuristic to form different cuts for each camera path cluster. This allows us to refine the nodes that are close to the clusters at the expense of those who are further away.

Given a node D in the cut, our heuristic estimates the approximation error made by $\tilde{S}_F(D)$ which we use to determine whether to replace D with its descendants or not. Let $\mathbf{x}_C$ be the centre of the camera path cluster and $\mathbf{x}_D$ be the centre of D. Furthermore, let r_C be the radius of the camera path cluster and r_D be the radius of D. Similarly to the approach used by Rokhlin [1985], we estimate the approximation error using the ratio

$$\frac{\|\mathbf{x}_C - \mathbf{x}_D\|}{\max(r_C, r_D)}.$$

If this ratio is below a tunable threshold, we keep D in the cut and do not replace it with its children. In all of our experiments, we set this threshold to 5.

After forming a cut $C = \{D_1, \ldots, D_N\}$, we precompute a PMF p_C and a CDF F_C to be able to efficiently sample nodes from C. Keeping in mind that we're only going to sample nodes from C for camera paths from the same cluster X, we define

$$p_C(D) = \frac{\textsc{estimateRadiance}(X, D)}{\sum_{D' \in C} \textsc{estimateRadiance}(X, D')}$$

where $\textsc{estimateRadiance}(X, D)$ estimates the average exitant radiance along all camera paths in X caused by all the point lights represented by D.

Estimating exitant radiance. In order to form the CDF and PMF above, we need to define ESTIMATERADIANCE. Using the same reasoning as in the previous

section, we split each term in the sum

$$S(X,D) = \frac{1}{|X|}\sum_{\bar{\mathbf{x}}\in X}\sum_{l\in D} L(l.\mathbf{x})V(l.\mathbf{x}\leftrightarrow\mathbf{x}_1)f^r(l.\mathbf{x}\to\mathbf{x}_1\to\mathbf{x}_2) \\ \frac{\max(0,N(\mathbf{x}_1)\cdot\omega(\mathbf{x}_1\to l.\mathbf{x}))\max(0,N(l.\mathbf{x})\cdot\omega(l.\mathbf{x}\to\mathbf{x}_1))}{\|l.\mathbf{x}-\mathbf{x}_1\|^2}$$

into the factors $C_X(l), M_X(l), D_X(l), V_X(l)$ and $I_X(l)$. We directly use the estimates from the previous section for the intensity factor and the visibility factor. In other words,

$$\hat{I}_X(D) = \sum_{l\in D} I_X(l)$$

and $\tilde{V}_X(D) = 1$.

We estimate the distance factor by precomputing the aggregates

$$A_X = \frac{1}{|X|}\sum_{\bar{\mathbf{x}}\in X}(\|\mathbf{x}_1\|^2) \text{ and } B_X = \frac{1}{|X|}\sum_{\bar{\mathbf{x}}\in X}\mathbf{x}_1$$

that we use to form the estimate

$$\tilde{D}_X(D) = (A_D - 2B_DB_X + A_X)^{-1}$$

which indeed is the harmonic mean of the distance factor in $D\times X$.

We construct an AABB $BV(X)$ and a bounding cone (ω_X,θ_X) for X which we use to form estimates for $C_X(l)$ and $M_X(l)$ by extending the results from the previous section. Again, we employ the radiosity assumption to estimate the material factor and precompute the average $\frac{c_X}{\pi}$ of the BRDFs for all camera paths in the cluster. We then estimate $M_X(D)$ by

$$\tilde{M}_X(D) = \frac{c_X}{\pi}\begin{cases}1 & \text{if } \Phi_X^M(D)\geq\theta_X\\ \max(0,\cos(\cos^{-1}(\Phi_X^M(D))-\cos^{-1}(\theta_X))) & \text{otherwise.}\end{cases}$$

where

$$\Phi_X^M(D) = \sqrt{\tilde{D}_X(D)}(\omega_X\cdot(BV(X)_c - BV(D)_c) + |\omega_X|\cdot(BV(X)_d + BV(D)_d))$$

is our estimate for the material factor. Similarly, if (ω_D,θ_D) is the bounding cone for D, then

$$\tilde{C}_X(D) = \begin{cases}1 & \text{if } \Phi_X^C(D)\geq\theta_D\\ \max(0,\cos(\cos^{-1}(\Phi_X^C(D))-\cos^{-1}(\theta_D))) & \text{otherwise.}\end{cases}$$

where

$$\Phi_X^C(D) = \sqrt{\tilde{D}_X(D)}(\omega_D \cdot (BV(D)_c - BV(X)_c) + |\omega_D| \cdot (BV(D)_d + BV(X)_d))$$

is our estimate for the light cosine factor.

Putting it all together, we now define ESTIMATERADIANCE as the product of these estimates. In other words,

$$\text{ESTIMATERADIANCE}(X, D) = \tilde{C}_X(D)\tilde{M}_X(D)\tilde{D}_X(D)\tilde{V}_X(D)\hat{I}_X(D).$$

Sampling algorithm. Once we have grouped the camera paths into clusters and formed a cut for each of the clusters, we apply Algorithm 9 to sample a point light for each camera path. The algorithm determines which cluster a camera path belongs by evaluating the identifier function I. This also determines which cut $C \in \mathcal{C}$ to sample from. On lines 4 and 5 the algorithm samples the first node D from C, such that the probability of selecting D is determined by the PMF $p_C(D)$. In practice, this is implemented using the precomputed CDF F_C and a binary search. Finally, on line 6, it invokes Algorithm 8 to sample a light source from D.

Algorithm 9 Clustered Importance Sampling

1: $I_{\bar{\mathbf{x}}} \leftarrow I(\bar{\mathbf{x}})$
2: $C \leftarrow \mathcal{C}[I_{\bar{\mathbf{x}}}]$
3: $u \leftarrow \text{OUTCOMEFROM}(\mathcal{U}_{[0,1]})$
4: $C' \leftarrow \{D \in C \mid F_C(D) \geq u\}$
5: $D \leftarrow C'[0]$
6: $(l, p) \leftarrow \text{HIERARCHICALIMPORTANCESAMPLE}(D)$
7: **return** $(l, p_C(D)p)$

9 Sequential Monte Carlo Instant Radiosity

In this chapter we present an importance sampling method for Instant Radiosity rendering algorithm that minimizes the number of VPLs that move between images in a sequence. Similarly to Incremental Instant Radiosity by Laine et al. [2007], our method is also a sequential Monte Carlo method that employs heuristic sampling and density estimation. However, unlike any prior method, *ours is able to simulate multi-bounce indirect illumination as well as distribute the VPLs according to the impact they have on the image.*

We make use of the radiosity assumption and talk about VPLs instead of light paths. This allows us to express the sampling algorithm in a simpler form, although it can easily be extended to support arbitrary BRDFs.

The main insight for the algorithm is that VPLs only contribute to the image if they are *indirectly visible* to the camera. In other words, that they can be reached by reflecting off the surfaces that are visible to the camera. Formally, a point $\mathbf{y}$ is indirectly visible if

$$\int_{\mathcal{P}^2} W(\mathbf{x}_2 \to \mathbf{x}_1) G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_1) G(\mathbf{x}_1 \leftrightarrow \mathbf{y}) \, \mathrm{d}\mathcal{A}(\mathbf{x}_2) \, \mathrm{d}\mathcal{A}(\mathbf{x}_1) > 0.$$

We build upon this insight and design our algorithm to maintain set of intelligently placed VPLs on the indirectly visible surfaces. As our sampling algorithm employs heuristic sampling, the probability densities of the resulting VPLs cannot be evaluated analytically and must instead be estimated using KNN[9] density estimation as described in Section 9.1.

Refer to Algorithm 10 for a high level description of our method. When a new frame begins (line 4), we have a set of old VPLs $(\mathbf{z}_{i-1,1}, \ldots, \mathbf{z}_{i-1,N})$ from the previous frame. As described in Section 9.2 we generate VPLs for the current frame by first sampling a collection of points $(\hat{\mathbf{z}}_{i,1}, \ldots, \hat{\mathbf{z}}_{i-1,N})$ on the indirectly visible surfaces by tracing paths from the camera (line 5). Then on line 6, we evaluate how suitable these points would be as VPLs, see Section 9.3. As described in Section 9.4 we remove the least suitable VPLs from the last frame (line 7) — most importantly ones that are not indirectly visible anymore — and replace them with the most suitable new candidates (line 8). We then estimate the exitant radiosity $(B_{i,1}, \ldots, B_{i,N})$ of each VPL (line 9). This estimation can be performed using any path sampling algorithm, such as forward path tracing. Finally, before rendering the frame (lines 11 – 14), we estimate the probability density of each VPL (line 10).

9.1 KNN density estimation

To use the VPLs to estimate the measurement equation in the path integral form, we need to estimate their probability densities with respect to the surface area measure A. We achieve this using KNN density estimation under the same assumptions made by the Photon mapping algorithm [Wann Jensen, 2001]. That is, we assume that the

[9] K-nearest neighbours. See Section 3.2.

Algorithm 10 Sequential Monte Carlo Instant Radiosity

$(\mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) \leftarrow$ sampleIndirectlyVisiblePoints()
$(\hat{p}_{0,1}, \ldots, \hat{p}_{0,N}) \leftarrow$ estimateSurfaceAreaDensity$(\mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N})$
$(B_{0,1}, \ldots, B_{0,N}) \leftarrow$ estimateRadiosity$(\mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N})$
for each new frame $i = 1$ to ∞ **do**
    $(\hat{\mathbf{z}}_{i,1}, \ldots, \hat{\mathbf{z}}_{i,N}) \leftarrow$ sampleIndirectlyVisiblePoints()
    $(s_{i,1}, \ldots, s_{i,N}) \leftarrow$ computeSuitability$(\hat{\mathbf{z}}_{i,1}, \ldots, \hat{\mathbf{z}}_{i,N})$
    $(\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,M}) \leftarrow$ removeUnsuitableVPLs$(\mathbf{z}_{i-1,1}, \ldots, \mathbf{z}_{i-1,N})$
    $(\mathbf{z}_{i,M}, \ldots, \mathbf{z}_{i,N}) \leftarrow$ chooseSuitablePoints$((\hat{\mathbf{z}}_{i,1}, \ldots, \hat{\mathbf{z}}_{i,N}), (s_{i,1}, \ldots, s_{i,N}))$
    $(B_{i,1}, \ldots, B_{i,N}) \leftarrow$ estimateRadiosity$(\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,N})$
    $(\hat{p}_{i,1}, \ldots, \hat{p}_{i,N}) \leftarrow$ estimateSurfaceAreaDensity$(\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,N})$
    for $j = 0$ to N **do**
        for each sensor S **do**
            $(\mathbf{x}_2, \mathbf{x}_1) \leftarrow$ sampleCameraPath()
            $\text{measurement}[S] \leftarrow \frac{1}{N} \frac{W(\mathbf{x}_2 \rightarrow \mathbf{x}_1) f^r(\mathbf{z}_{i,j} \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2) G(\mathbf{x}_1 \leftrightarrow \mathbf{z}_{i,j}) B_{i,j}}{p(\mathbf{x}_1, \mathbf{x}_2) \hat{p}_{i,j}}$

k nearest neighbours of any VPL are all on the same, planar surface[10]. This allows us to perform an unconstrained KNN search in the scene $\mathcal{M}$ using the algorithm presented by Roussopoulos et al. [1995]. We then estimate the probability density of a VPL $\mathbf{x}$ as

$$\hat{p}(\mathbf{x}) = \frac{k}{N} \frac{1}{\pi r_k^2},$$

where r_k is the distance to the kth nearest neighbour of $\mathbf{x}$.

9.2 Generating candidates

As depicted above, we generate new VPLs by selecting them from a set of candidates. We generate these candidates by first sampling a collection of points $Z = (\bar{\mathbf{z}}_1, \ldots, \bar{\mathbf{z}}_N)$ on the indirectly visible surfaces. In practice, each point $\bar{\mathbf{z}}$ is generated by sampling a path $(\mathbf{z}_3, \mathbf{z}_2, \mathbf{z}_1)$, where $\mathbf{z}_1$ is the resulting VPL candidate. The path is generated similarly to how the base paths are sampled in Chapter 5.3. That is, $\mathbf{z}_3$ is sampled on the sensor, the so called *primary hit* $\mathbf{z}_2$ is sampled from the surfaces visible to $\mathbf{z}_2$ and $\mathbf{z}_1$ is sampled from the surfaces visible to $\mathbf{z}_2$.

[10]Note that these assumptions are consistent if the scene is a polygonal mesh. As the density of the VPLs increases there will inevitably be a point where all the k nearest neighbours of $\mathbf{x}$ in $\mathbb{R}^3$ are located on the same polygon as $\mathbf{x}$.

We then estimate the exitant radiosity at the VPLs by sampling light paths from them. By sampling more than one light path from each VPL (and also varying the length of these paths), the illumination from a single VPL is in fact a sample of the multi-bounce indirect illumination. In practice, we achieve this by using forward path tracing to sample a fixed number of new light paths from each point in Z. We maintain these light paths between iterations by regenerating their first vertices (that lie on the light sources) when the illumination in the scene changes.

Since we assume diffuse materials, we only need to estimate the exitant radiosity at each VPL. Without this assumption, we would have to treat each light path separately if they have different directions for the radiance incident on the VPL.

9.3 Suitability

To importance sample the VPLs according to how much indirect light they bring to the image, the suitability score of a VPL should indicate how well its probability density corresponds to the light it brings to the image. If we assume that indirect visibility is binary, the amount of indirect light a VPL $\mathbf{z}$ brings to the image only depends on its radiosity $B(\mathbf{z})$, since we already know that all of our VPL candidates are indirectly visible. We account for radiosity by defining the suitability of a VPL candidate as its strength as it would occur in a Monte Carlo sum. In other words,

$$\text{SUITABILITY}(\mathbf{z}) = \frac{B(\mathbf{z})}{\hat{p}(\mathbf{z})},$$

where $\hat{p}(\mathbf{z})$ is a KNN density estimate for the point $\mathbf{z}$. We evaluate $\hat{p}(\mathbf{z})$ by only searching for neighbors among the VPLs from the last frame. To cover the indirectly visible surfaces as evenly as possible, we use single nearest neighbour search when evaluating suitability. The reason for this is that k nearest neighbour search is not able to distinguish evenly spaced VPLs from isolated groups of $k-1$ VPLs.

Informally, each VPL is used during rendering to approximate the radiosity of its nearest-neighbour region. In order to avoid systematic bias we have to ensure that the VPLs are representative samples of these regions. The sampling heuristics we employ effectively repel new candidates from landing near VPLs from the last frame because we estimate the probability density using nearest-neighbour search.

If we directly define the suitability in terms of the radiosity for the candidate alone, the sampling method tends to favour VPLs that have a comparatively high radiosity for their regions. This results in a bias towards brighter VPLs. We circumvent this

issue by not using the radiosity of VPL candidate when defining the suitability, but instead approximate the average radiosity within its nearest neighbour region. In practice, we compute this approximation as the average of radiosities of the neighbouring VPLs from the last frame.

9.4 Removing and replacing VPLs

Recall that we want the VPLs to be indirectly visible to the camera. As a consequence, for every frame we remove any VPLs that are not indirectly visible. A VPL $\mathbf{z}$ is indirectly visible if we can find a point $\mathbf{x}$ which is both visible to the camera and $\mathbf{z}$. We find such points using two different methods. One that generates points that are definitely visible to the camera, in that case we have to check if they are visible to the VPLs. The other generates points that are definitely visible to a VPL and instead requires us to check that they are visible to the camera.

We generate points that are definitely visible to the camera simply by sampling camera paths of the form $(\mathbf{z}_2, \mathbf{z}_1)$, where $\mathbf{z}_1$ are on the surfaces visible to the camera. For each such camera path we choose a VPL $\mathbf{z}$ with the help of the hierarchical importance sampling algorithm from the Chapter 8. We then query the indirect visibility of $\mathbf{z}$ by checking if it is visible to the primary hit $\mathbf{z}_1$.

Recall that each VPL was generated by sampling three vertex paths $(\mathbf{z}_3, \mathbf{z}_2, \mathbf{z}_1)$ from the camera. Even if the camera moves, the primary hit $\mathbf{z}_2$ of such a path is definitely visible to the corresponding VPL. This allows us to query the indirect visibility of a VPL by checking if $\mathbf{z}_2$ is still visible to the camera.

Coping with undersampling. Removing VPLs based on indirect visibility alone can lead to severe undersampling of new parts of the scene. It's possible for the indirectly visible surfaces from the previous frame to be a strict subset of the indirectly visible surfaces of the current frame. In other words, a new camera position might uncover new indirectly visible surfaces without occluding any surfaces from the last frame. For example, when the camera is moving backwards.

We propose another heuristic to complement indirect visibility. The general idea is to forcefully remove VPLs from the last frame when there are regions that haven't been sufficiently sampled. Again we make use of the VPL candidates generated for this frame. Any candidate VPL that has landed in an undersampled region will have an exceptionally high suitability score.

We decide how many VPLs should be removed by comparing the suitability of the new candidates against some representative aggregate for the suitabilities of the VPLs from the last frame. In our implementation we count how many of the new candidates are more suitable than the 90th percentile of last frame's VPLs. If the indirect visibility heuristic hasn't removed that many VPLs yet, we remove more VPLs until we reach the budget. In an attempt to reduce temporal noise we deterministically remove the oldest VPLs in favor of resampling without replacement.

Replacing VPLs. We now replace the removed VPLs with suitable candidates from Y. To ensure that there's always a non-zero probability of placing a new VPL in any indirectly visible region, we do not deterministically choose the most suitable replacements from Y. Instead, we sample the replacements such that suitable VPLs have higher probabilities of being chosen. In a sequential program, this is best achieved by resampling without replacement. In order to cater for the parallel programming model of modern GPUs we instead use Algorithm 11 to sample from Y.

Algorithm 11 Parallel VPL Sampling

1: $M \leftarrow$ number VPLs to be replaced
2: **for** each $\bar{\mathbf{y}} \in Y$ **in parallel do**
3: weights$[\bar{\mathbf{y}}] \leftarrow$ SUITABILITY$(\bar{\mathbf{y}})$OUTCOMEFROM$(\mathcal{U}_{[0,1]})$
4: PARALLELSORT$(Y, weights)$
5: **return** $(\bar{\mathbf{y}}_N, \ldots, \bar{\mathbf{y}}_{N-M})$

10 Implementation and results

In this chapter present the technologies used to implement the clustered hirearchical importance sampling algorithm from Chapter 8 and the sequential Monte Carlo Instant Radiosity algorithm from Chapter 9. We experimentally evaluate the quality of these algorithms and analyse the results. We also measure the performance of a system that makes use of both of these algorithms to render images with indirect illumination at interactive rates.

Here we choose not to focus on direct illumination, and instead render images containing only indirect light. We also choose to omit Russian roulette and limit the

(a) Overview. (b) Ground truth. (c) Ideal importance sampling. (d) Our method. (e) Forward path tracing.

Figure 19: A comparison of the high frequency noise exhibited by different rendering algorithms when they are restricted to 8 camera paths per pixel. We see that our method is closely resembles the ideal case and attains a considerably lower error than forward path tracing.

length of a transport path to six points. In other words, we simulate three bounce indirect illumination.

10.1 Implementation

Both algorithms presented here and the comparison methods are implemented to run on modern GPUs. The algorithms are implemented in C++ [Stroustrup, 2000] with the help of the CUDA framework for general purpose GPU (GPGPU) computing [Farber, 2012].

In our implementation, we use the following third party components:

- The Optix Prime library for a GPGPU implementation of a ray casting algorithm [OptiX Prime].
- The temporal reprojection filter developed by Mara et al. [2013] to smooth out the remaining high frequency noise.
- The GPGPU tree builder presented by Karras [2012] to rapidly construct the light hierarchy using a top-down algorithm that uses the spatial mean heuristic.
- The CUB library by Merrill [2014] for GPGPU sorting.

10.2 Single image quality

We measure the error made by our many lights rendering algorithms as estimators for the measurement equation by creating the point lights using sequential Monte Carlo

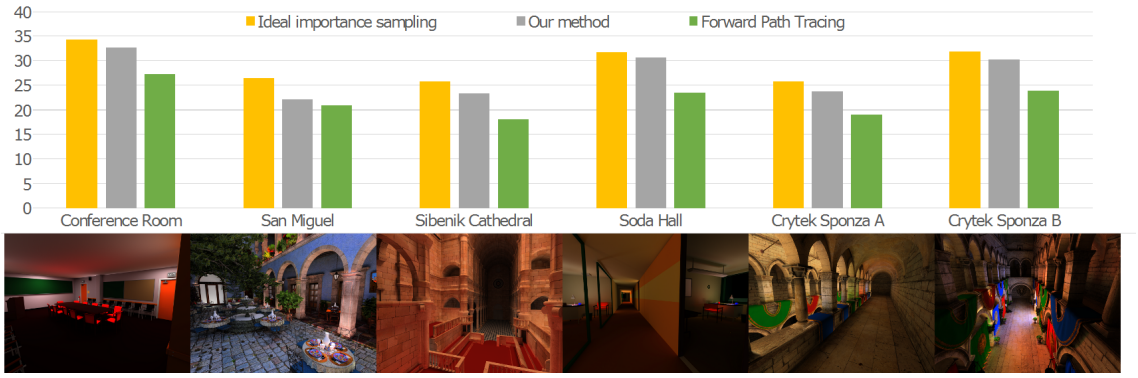


Figure 20: Diagrams depicting the PSNR achieved by our method for a sample of six different scenes. We see that our method consistently outperforms forward path tracing and comes close to the quality achieved by the ideal importance sampler.

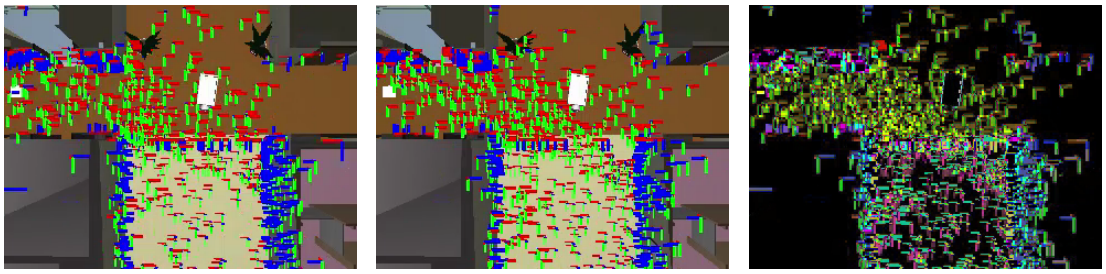
Instant Radiosity. We achieve this by rendering scenes illuminated by 32768 VPLs and comparing the results against the converged results of forward path tracing. For the clustered hierarchical importance sampler, we use a on-screen cluster size of 16×16 pixels in all scenes except for San Miguel, where we reduce the amount of clusters by increasing the cluster size to 32×32 pixels due to memory budget constraints.

See Figure 19 for a visual comparison of the high frequency noise for these algorithms in comparison to forward path tracing. We include the ideal importance sampler from Chapter 8.1 as a theoretical upper bound for the quality which can be achieved without estimating visibility. From this figure it's obvious that the highest quality is achieved by the ideal importance sampler and the clustered hierarchical importance sampler.

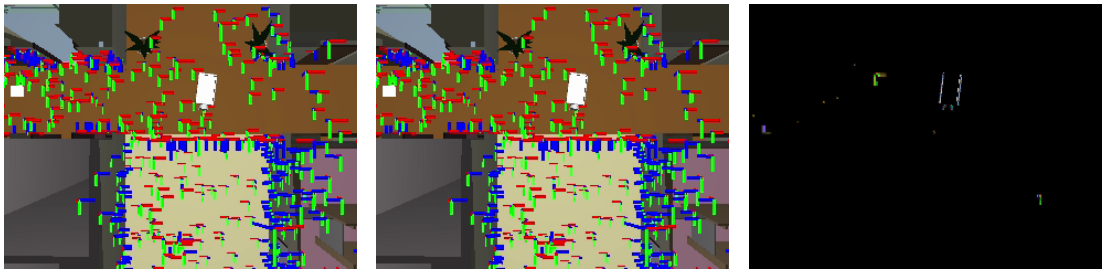
We measure the quality of the rendered images using their *peak signal-to-noise ratio (PSNR)* [Salomon, 2006, Chapter 4.2.2]. PSNR is the logarithm of the ratio between the brightest pixel in the image and the mean squared error (MSE). Informally, an image with a high PSNR value contains less noise. From Figure 20 we see that the clustered hierarchical importance sampling algorithm also performs well in a wider selection of scenes.

10.3 Image sequence quality

Here we evaluate the ability of sequential Monte Carlo Instant Radiosity to generate accurate, yet stable VPL sets. We compare against Instant Radiosity Keller [1997]



(a) Metropolis Instant Radiosity.



(b) Sequential Monte Carlo Instant Radiosity.

Figure 21: The scene viewed from above, where the camera is represented by the white box. The leftmost column visualizes the VPLs in the kth image in a sequence by drawing their local coordinate axes. The middle column shows the VPLs in the $(k+1)$th image, where the camera has moved slightly to the left. The column on the right shows the absolute difference between the other two. Note that our method keeps the distribution of VPLs temporally stable. See the video at
`http://cs.helsinki.fi/u/phedman/smcir/vpl_distributions.mp4`
for a more in-depth presentation.

and Metropolis Instant Radiosity Segovia et al. [2007][11]. In these experiments, our method generates 24 light paths from each VPL, 8 for each indirect bounce.

First, Figure 21 demonstrates how our method incrementally adapts the distribution of VPLs to a moving camera. In contrast, Metropolis Instant Radiosity changes the location of all VPLs between frames.

Second, to assess the temporal smoothness of the error, we give both our algorithm and the two comparison methods a budget of 2048 VPLs, deterministically connect all of them to every pixel, and track the intensities of several pixels over the sequence. Figure 22 plots these sensor responses as the camera moves through a scene. We

[11]Instead of using MLT to generate transport paths whose probability densities are proportional to the sensor response they cause, we instead generate a collection of candidate paths using forward path tracing. We then employ weighted resampling without replacement to generate a set of transport paths approximately distributed proportionally to the sensor response they cause.

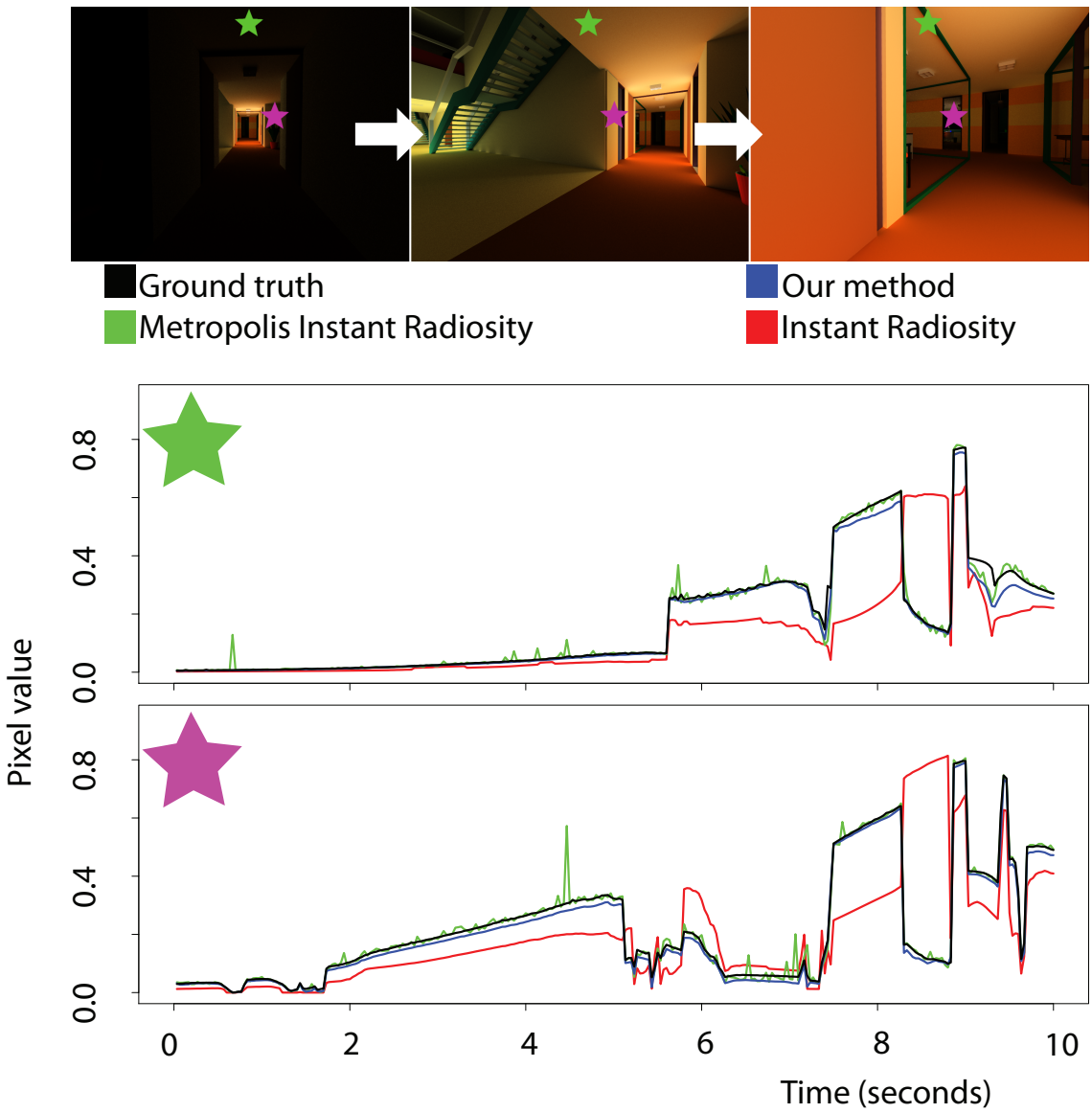


Figure 22: The sensor response over time for the highlighted pixels in a sequence of images. In this sequence, the camera moves through a heavily occluded scene, illuminated by 10 light sources. Unlike the comparison methods, we see that our method both follows the ground truth closely and is temporally stable. All methods use 2048 VPLs and deterministic connections to all pixels to highlight the temporal behaviour of the error.

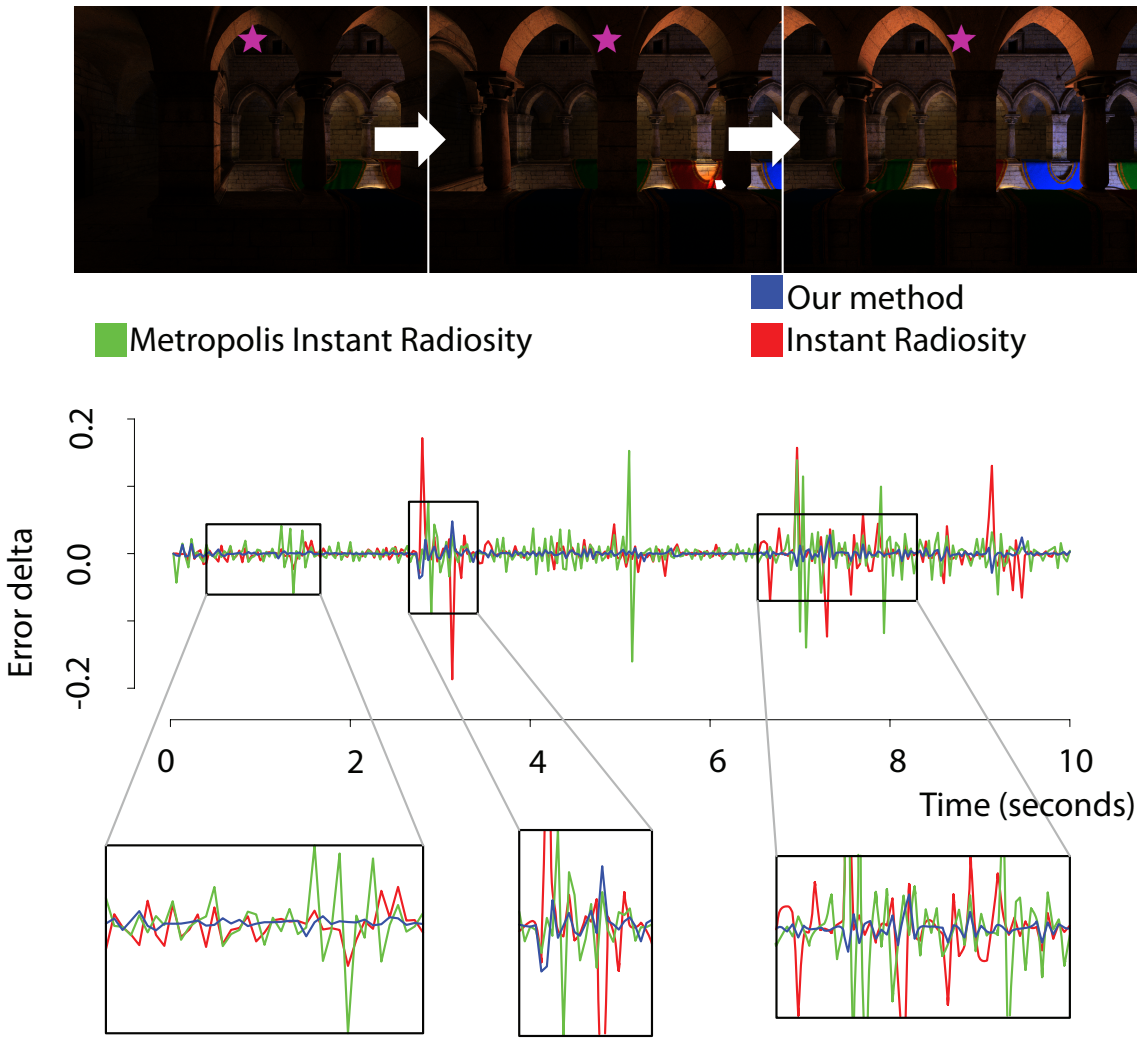


Figure 23: The differential error with respect to time for the highlighted pixel. In this case, the camera is slowly moving left in a scene which is illuminated by two light sources. This shows that the error of our method fluctuates significantly less than that of the comparison methods.

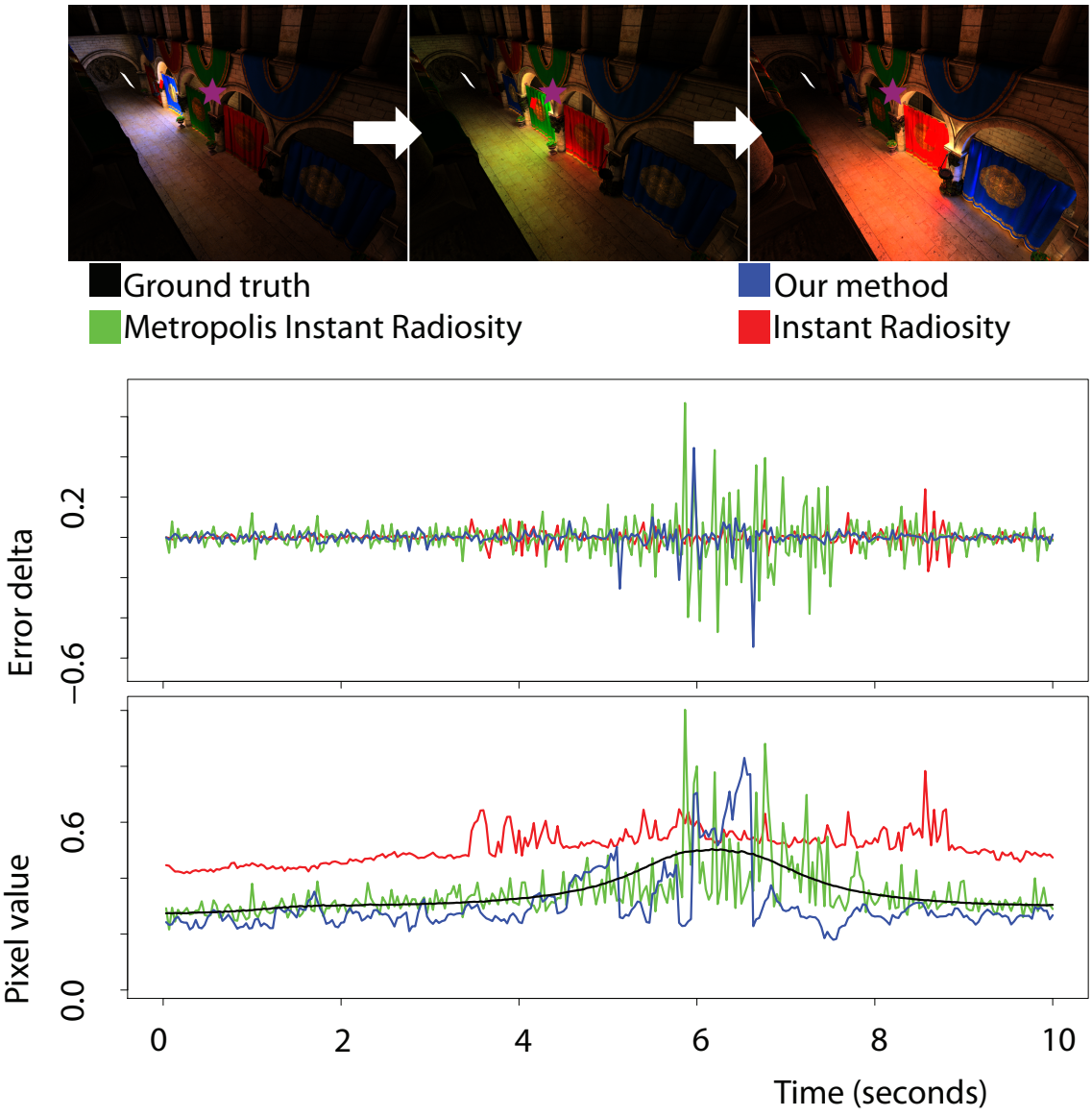


Figure 24: The sensor response and error differential for the case where the light source is moving instead of the camera. We see that our method is closer to the ground truth than Instant Radiosity and fluctuates less than Metropolis Instant Radiosity.

compute the ground truth for these pixels by running forward path tracing until convergence. We observe that while the estimates provided by Metropolis Instant Radiosity closely follow the ground truth, they are not temporally stable and an image sequence rendered using this method would suffer from flickering. In contrast, the estimates produced by our method, while slightly biased, do not fluctuate with time. This is further evidenced by Figure 23, which displays the differential of the error with respect to time. Instant Radiosity consistently produces large errors, as it does not adapt to the view-light configuration.

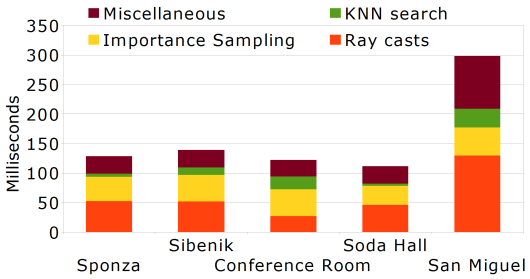
Figure 24 shows a similar graph for a case where the light source is moving instead of the camera. This results in lower accuracy for all methods. However, our method remains closer to ground truth than Instant Radiosity, while at the same time being more temporally stable than Metropolis Instant Radiosity.

We also compare these methods by rendering videos. The videos at
`http://cs.helsinki.fi/u/phedman/smcir/moving_light.mp4` and
`http://cs.helsinki.fi/u/phedman/smcir/moving_camera.mp4`
demonstrate the error characteristics of our method and the comparison methods when rendering images at interactive rates. From the videos, we see that our method much higher temporal stability than the comparison methods. Forward path tracing and our method use 8 camera paths per pixel and employ the temporal reprojection filter by Mara et al. [2013] to alleviate the high frequency noise. Instant Radiosity and Metropolis Instant Radiosity have a VPL budget of 2048 VPLs and employ 8×8 interleaved sampling as described by Laine et al. [2007] to reduce the number of transport paths they generate per pixel from 2048 to 32.

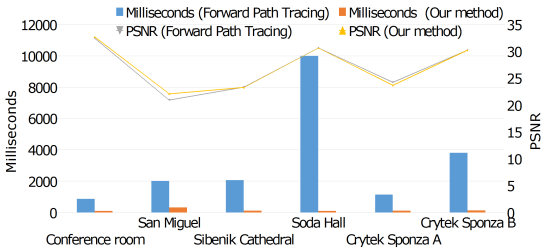
10.4 Rendering performance

We analyse the rendering performance of the sequential Monte Carlo Instant Radiosity algorithm on the following hardware: A desktop PC equipped with an Intel i5 2500 CPU, 16 GB RAM and a NVIDIA GTX Titan GPU. In Figure 25a we display the time spent rendering an image for a collection of scenes. The rendering time is split into four categories: ray tracing, density estimation, importance sampling and miscellaneous. The miscellaneous category contains operations such as evaluating BRDFs, sampling new directions in which to perform ray casts and evaluating the measurement contribution function.

In Figure 25b we compare the performance of a forward path tracer against sequen-



(a) Rendering time in milliseconds for six different scenes.



(b) Rendering time in milliseconds required for forward path tracing to attain the same quality (PSNR) as our method with a budget of 8 camera paths per pixel.

Figure 25: Rendering performance of our method in contrast to forward path tracing.

tial Monte Carlo Instant Radiosity. The figure displays the running time required for the forward path tracing algorithm to reach the same PSNR as sequential Monte Carlo Instant Radiosity achieves with only 8 camera paths per pixel. We see that sequential Monte Carlo Instant Radiosity is always an order of magnitude faster than forward path tracing.

11 Conclusion

We set out to create a rendering method for interactive image sequences, which is flexible enough to synthesize both rough approximations for limited hardware and physically accurate images in real-time for future hardware. The result is a method that faithfully renders images with multi-bounce indirect illumination at interactive rates. The method consists of two novel rendering algorithms that both extend Instant Radiosity [Keller, 1997].

The first, clustered hierarchical importance sampling, is a many-lights rendering method that increases the point light budget without incurring a significant performance cost. As a whole, our method compares favourably against forward Path Tracing, both in terms of quality and rendering speed.

The second, sequential Monte Carlo Instant Radiosity, generates light paths using a heuristic sampling method that importance samples them according to much light they bring to the image, but also reduces the amount of light paths that move between images in a sequence. Previous work has only focused on one of these

Figure 26: Images containing glossy materials rendered using Instant Radiosity with 32768 VPLs.

aspects, and no prior algorithm has been able to achieve both at the same time. Our experiments show that our algorithm produces images whose error is on par with Metropolis Instant Radiosity, while keeping the illumination temporally stable in sequences.

11.1 Future work

In this thesis, we employ ray casting to evaluate visiblity, however, this is not a necessary assumption for sequential Monte Carlo Instant Radiosity. An interesting venue for future research is to explore the behaviour of this algorithm under hard real-time time constraints where rasterisation is used to resolve visibility.

The clustered hierarchical importance sampler employs the naive assumption that all VPLs are visible. Designing better estimate for the visibility factor would reduce variance even further.

The camera path clusters take first steps towards a dually hierarchical importance sampler which computes the sampling probabilities using not only a light hierarchy but also a hierarchy built from the camera paths. This is similar in spirit to *Multidimensional Lightcuts* by Walter et al. [2006].

We've employed the radiosity assumption when implementing our algorithms. However, neither one of the algorithms rely on the assumption. Traditionally, Instant Radiosity has not been used with glossy materials due to the limited VPL budget. However, as Figure 26 implies, the increased VPL budget caused by the clustered hierarchical importance sampler allows for plausible glossy reflections. An interesting venue of research would be to add support for glossy BRDFs to our rendering system by deriving an estimate for the material factor for these BRDFs.

References

T. Aila, S. Laine, and T. Karras. Understanding the efficiency of ray traversal on GPUs – Kepler and Fermi addendum. NVIDIA Technical Report NVR-2012-02, NVIDIA Corporation, June 2012.

T. Aila, T. Karras, and S. Laine. On quality metrics of bounding volume hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference*, HPG '13, pages 101–107, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2135-8.

T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008. ISBN 987-1-56881-424-7.

A. Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), pages 37–45, New York, NY, USA, 1968. ACM.

K. B. Athreya and S. N. Lahiri. *Measure Theory and Probability Theory*. Springer Texts in Statistics. Springer, New York, NY, USA, 2006. ISBN 0-387-35434-7.

T. Barák, J. Bittner, and V. Havran. Temporally coherent adaptive sampling for imperfect shadow maps. *Computer Graphics Forum*, 32(4):87–96, 2013.

G. Barequet and G. Elber. Optimal bounding cones of vectors in three dimensions. *Information Processing Letters*, 93(2):83–89, January 2005.

J. F. Blinn. Models of light reflection for computer synthesized pictures. *ACM SIGGRAPH Computer Graphics*, 11(2):192–198, July 1977.

W. J. Bouknight. A procedure for generation of three-dimensional half-toned computer graphics presentations. *Communications of the ACM*, 13(9):527–536, September 1970.

R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1):7–24, January 1982.

M. K. Cowles and B. P. Carlin. Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91(434): 883–904, 1996.

C. Dachsbacher, J. Křivànek, M. Hašan, A. Arbree, B. Walter, and J. Novák. Scalable realistic rendering with many-light methods. *Computer Graphics Forum*, 33 (1):88–104, 2014.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

Z. Dong, T. Grosch, T. Ritschel, J. Kautz, and H.-P. Seidel. Real-time indirect illumination with clustered visibility. In *Proceedings of the Vision, Modeling, and Visualization Workshop*, pages 187–196, 2009.

A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Statistics for Engineering and Information Science. Springer, New York, NY, USA, 2001. ISBN 0-387-95146-6.

P. Dutre, K. Bala, and P. Bekaert. *Advanced Global Illumination 2nd Edition*. A. K. Peters, Ltd., Wellesley, MA, USA, 2006. ISBN 1-56881-307-3.

R. Eckhardt. Stan Ulam, John von Neumann, and the Monte Carlo method. *Los Alamos Science*, 15 (Special Issue, Stanisław Ulam 1909–1984):131–136, 1987.

R. Farber. *CUDA Application Design and Development*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012. ISBN 978-0-12388-426-8.

R. P. Feynman. *QED: The Strange Theory of Light and Matter*. Princeton University Press, Princeton, NJ, USA, 2006. ISBN 0-691-12575-9.

I. Georgiev and P. Slusallek. Simple and robust iterative importance sampling of virtual point lights. In H. P. A. Lensch and S. Seipel, editors, *Proceedings of Eurographics 2010 (short papers)*, pages 57–60, Norrkoping, Sweden, 2010. Eurographics Association.

I. Georgiev, J. Křivànek, S. Popov, and P. Slusallek. Importance caching for complex illumination. *Computer Graphics Forum*, 31(23):701–710, 2012.

P. R. Halmos. *Measure Theory*. Number v. 9 in Graduate Texts in Mathematics. Springer, New York, NY, USA, 1974. ISBN 0-387-90088-8.

P. Hämäläinen, T. Aila, T. Takala, and J. Alander. Mutated kd-tree importance sampling. In *Proceedings of the The Ninth Scandinavian Conference on Artificial Intelligence*, pages 39–45, Helsinki, Finland, 2006. Finnish Artificial Intelligence Society.

P. Hämäläinen, S. Eriksson, E. Tanskanen, V. Kyrki, and J. Lehtinen. Online motion synthesis using sequential Monte Carlo. *ACM Transactions on Graphics*, 33(4): 51:1–51:12, July 2014.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

V. Havran. *Heuristic Ray Shooting Algorithms.* PhD thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Prague, Czech, November 2000.

M. Hašan, F. Pellacini, and K. Bala. Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics*, 26(3), July 2007.

M. Hašan, E. Velázquez-Armendariz, F. Pellacini, and K. Bala. Tensor clustering for rendering many-light animations. In *Proceedings of the Nineteenth Eurographics Conference on Rendering*, EGSR '08, pages 1105–1114, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

D. S. Immel, M. F. Cohen, and D. P. Greenberg. A radiosity method for non-diffuse environments. *ACM SIGGRAPH Computer Graphics*, 20(4):133–142, August 1986.

J. T. Kajiya. The rendering equation. *ACM SIGGRAPH Computer Graphics*, 20 (4):143–150, August 1986.

M. H. Kalos and P. A. Whitlock. *Monte Carlo Methods. Vol. 1: Basics.* Wiley-Interscience, New York, NY, USA, 1986. ISBN 0-471-89839-2.

D. Kalra and A. H. Barr. Guaranteed ray intersections with implicit surfaces. *ACM SIGGRAPH Computer Graphics*, 23(3):297–306, July 1989.

T. Karras. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics conference on High-Performance Graphics*, EGGH-HPG'12, pages 33–37, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association. ISBN 978-3-905674-41-5.

A. Keller. Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7.

M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer. Differential instant radiosity for mixed reality. In *9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 99–107. IEEE Computer Society, Oct 2010.

T. Kollig and A. Keller. Illumination in the presence of weak singularities. In H. Niederreiter and D. Talay, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 245–257. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-25541-3.

S. Laine, H. Saransaari, J. Kontkanen, J. Lehtinen, and T. Aila. Incremental instant radiosity for real-time indirect illumination. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 277–286, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. ISBN 978-3-905673-52-4.

M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9 (3):245–261, July 1990.

D. O. Loftsgaarden and C. P. Quesenberry. A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3):1049–1051, 1965.

D. J. MacDonald and K. S. Booth. Heuristics for ray tracing using space subdivision. *The Visual Computer: International Journal of Computer Graphics*, 6(3):153–166, May 1990.

M. Mara, M. McGuire, and D. Luebke. Lighting deep g-buffers: Single-pass, layered depth images with minimum separation applied to indirect illumination. Technical Report NVR-2013-004, NVIDIA Corporation, December 2013. URL `http://graphics.cs.williams.edu/papers/DeepGBuffer13`.

M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, January 1998.

S. Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106(2):286 – 303, 1993.

D. Merrill. CUB. [Online, fetched September 2014] `http://nvlabs.github.io/cub/`, 2014.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

F. Morgan. *Geometric Measure Theory: A Beginner's Guide*. Geometric Measure Theory Series. Elsevier Science, Burlington, MA, USA, 2008. ISBN 9780080922409.

Fred E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied Optics*, 4(7):767–775, Jul 1965.

H. Niederreiter. *Random Number Generation and quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992. ISBN 0-89871-295-5.

O. Olsson, M. Billeter, and U. Assarsson. Clustered deferred and forward shading. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*, EGGH-HPG'12, pages 87–96, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association. ISBN 978-3-905674-41-5.

OptiX Prime. OptiX. [Online, fetched September 2014] `https://developer.nvidia.com/optix`, 2014.

J. Pitman. *Probability*. Springer Texts in Statistics. Springer, New York, NY, USA, 1993. ISBN 0-387-97974-3.

R. Prutkin, A. Kaplanyan, and C. Dachsbacher. Reflective Shadow Map Clustering for Real-Time Global Illumination. In Carlos Andujar and Enrico Puppo, editors, *Eurographics 2012 - Short Papers*, Aire-la-Ville, Switzerland, Switzerland, 2012. The Eurographics Association.

T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics*, 27(5):129:1–129:8, December 2008.

T. Ritschel, E. Eisemann, I. Ha, J. D. K. Kim, and H.-P. Seidel. Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. *Computer Graphics Forum*, 30(8):2258–2269, 2011.

V. Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60(2):187–207, 1985.

N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. *ACM SIGMOD Record*, 24(2):71–79, May 1995.

D. Salomon. *Data Compression: The Complete Reference, 4th edition.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 1-84628-602-6.

H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990. ISBN 0-201-50255-0.

B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. Bidirectional instant radiosity. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, EGSR'06, pages 389–397, Aire-la-Ville, Switzerland, Switzerland, 2006a. Eurographics Association. ISBN 3-905673-35-5.

B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. Non-interleaved deferred shading of interleaved sample patterns. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '06, pages 53–60, New York, NY, USA, 2006b. ACM. ISBN 3-905673-37-1.

B. Segovia, J.C. Iehl, and B. Péroche. Metropolis instant radiosity. *Computer Graphics Forum*, 26(3):425–434, 2007.

P. Shirley and K. Chiu. A low distortion map between disk and square. *Journal of Graphics Tools*, 2(3):45–52, December 1997.

L. A. Shirmun and S. S. Abi-Ezzi. The cone of normals technique for fast processing of curved patches. *Computer Graphics Forum*, 12(3):261–272, 1993.

D. Shreiner, G. Sellers, J. M. Kessenich, and B. M. Licea-Kane. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3 (8th Edition).* Addison-Wesley Professional, Boston, MA, USA, 2013. ISBN 978-0-32177-303-6.

A. F. M. Smith and A. E. Gelfand. Bayesian statistics without tears: A sampling-resampling perspective. *The American Statistician*, 46(2):pp. 84–88, 1992.

B. Stroustrup. *The C++ Programming Language.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 2000. ISBN 0-201-70073-5.

A. E. Taylor and D. C. Lay. *Introduction to Functional Analysis, 2nd Edition.* Krieger Publishing Co., Inc., Melbourne, FL, USA, 1986. ISBN 0898749514.

J. Ulbrich, J. Novák, H. Rehfeld, and C. Dachsbacher. Progressive visibility caching for fast indirect illumination. In Michael M. Bronstein, Jean Favre, and Kai Hormann, editors, *Proceedings of the Vision, Modeling, and Visualization Workshop*, pages 203–210, Aire-la-Ville, Switzerland, Switzerland, 2013. Eurographics Association.

E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation.* PhD thesis, Stanford University, Stanford, CA, USA, 1998.

J. von Neumann. Various techniques used in connection with random digits. *National Bureau of Standards Applied Mathematics Series*, 12:36–38, 1951.

B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. Lightcuts: A scalable approach to illumination. *ACM Transactions on Graphics*, 24(3):1098–1107, July 2005.

B. Walter, A. Arbree, K. Bala, and D. P. Greenberg. Multidimensional lightcuts. *ACM Transactions on Graphics*, 25(3):1081–1088, July 2006.

H. Wann Jensen. *Realistic Image Synthesis Using Photon Mapping.* A. K. Peters, Ltd., Natick, MA, USA, 2001. ISBN 1-56881-147-0.

Y.-T. Wu and Y.-Y. Chuang. VisibilityCluster: Average directional visibility for many-light rendering. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1566–1578, Sept 2013.

S. Yakowitz, J. Krimmel, and F. Szidarovszky. Weighted Monte Carlo integration. *SIAM Journal on Numerical Analysis*, 15(6):1289–1300, 1978.

A Measures and probability theory

Here we review the relevant concepts from probability theory which we need to describe Monte Carlo integration and sequential Monte Carlo. Since the light transport problem cannot be accurately modelled using naive probability theory, these concepts are best described using the measure-theoretic approach to probability theory. As a formal treatment of measure-theoretic approach is out of the scope for this thesis we make use of simplifications to preserve clarity of presentation.

For an introduction to naive probability theory, refer to *Probability* by Pitman [1993]. *Measure Theory* by Halmos [1974] is a good introduction to measure theory. For a formal treatment of measure-theoretic probability, *Measure Theory and Probability Theory* by Athreya and Lahiri [2006] is a good reference.

A.1 Measures

A *measure* is a generalization of quantities such as length, area and volume. A measure is always defined on a *measurable space* $\mathbb{X} = (X, \mathcal{F})$, where X is a set and $\mathcal{F} \subset \mathcal{P}(X)$ is a σ-algebra that contains the measurable subsets of X. For our purposes, we can consider $\mathcal{F}$ as a family of sets which contains at least $\emptyset$ and X. A measure μ on $\mathbb{X}$ is a function $\mu : \mathcal{F} \to [0, \infty]$ which allows us to express the sizes of the elements in $\mathcal{F}$. Aside from always being non-negative, a measure must also satisfy two other properties we commonly associate with the concept of size. Namely,

$$\mu(\emptyset) = 0 \text{ and } \mu(\bigcup_i A_i) = \sum_i \mu(A_i)$$

for up to countably many disjoint sets $A_i \in \mathcal{F}$.

Measurable functions and integrals. We can extend the concept of measurability to functions as well. A function $f : (X, \mathcal{F}_X) \to (Y, \mathcal{F}_Y)$ between two measurable spaces is *measurable* if the preimage of any measurable set is also measurable. In other words, for any set $A \subset Y$,

$$A \in \mathcal{F}_Y \Rightarrow f^{-1}(A) \in \mathcal{F}_X.$$

Any measurable function can be integrated. Given the measurable function $f : X \to \mathbb{R}$, we denote the Lebesgue integral of f over a set $A \in \mathcal{F}$ with respect to the

measure μ as

$$\int_A f(a)\,\mathrm{d}\mu(a).$$

Common measures. An important example of a measurable space and a measure the *Lebesgue measure* on $\mathbb{R}^n$, which coincides with our intuitive notion of volume. As one might assume, integrals taken with respect to the Lebesgue measure behave like the familiar Riemann integral in the cases where both exist.

Consider the case where we want to measure the size of a lower-dimensional manifold in $\mathbb{R}^n$. For example, we want to measure the area of a surface in $\mathbb{R}^3$. We cannot use the three-dimensional Lebesgue measure for this purpose as it measures volume. As a consequence, the measure of any set D on the surface would be zero. Instead, we make use of a *Hausdorff measure* to express this area [Morgan, 2008, Chapter 2.3]. Informally, a k-dimensional Hausdorff measure $\mathcal{H}^k$ on $\mathbb{R}^n$ coincides with our intuition of k-dimensional volumes (or areas) embedded in $\mathbb{R}^n$. As one might assume, $\mathcal{H}^k$ coincides with the Lebesgue measure on $\mathbb{R}^n$ if $k = n$

Radon-Nikodym derivatives. We make heavy use of a concept called the *Radon-Nikodym derivative*. Consider two measures μ_1 and μ_2 on $\mathbb{X}$ such that $\mu_1(A) = 0$ whenever $\mu_2(A) = 0$ for all $A \in \mathcal{F}$. In this case, the Radon-Nikodym derivative

$$\frac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2} : \mathbb{X} \to \mathbb{R}$$

is the function that satisfies

$$\int_A f(a)\,\mathrm{d}\mu_1(a) = \int_A f(a)\frac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}(a)\,\mathrm{d}\mu_2(a).$$

Product measures. A *product measure* allows us to define a measure on the Cartesian product of measurable spaces. Consider two measurable spaces $\mathbb{X} = (X, \mathcal{F}_X)$ and $\mathbb{Y} = (Y, \mathcal{F}_Y)$ equipped with the measures μ_X and μ_Y. We extend the Cartesian product $X \times Y$ into a measure space by equipping it with the σ-algebra $\mathcal{F}_X \times \mathcal{F}_Y$. The product measure $\mu_X \times \mu_Y$ is a measure on $(X \times Y, \mathcal{F}_X \times \mathcal{F}_Y)$ and is defined as

$$\begin{aligned}(\mu_X \times \mu_Y)(D) &= \int_D \mu_Y(D_x)\,\mathrm{d}\mu_X(x) \\ &= \int_D \mu_X(D_y)\,\mathrm{d}\mu_Y(y) \quad \forall\, D \in \mathcal{F}_X \times \mathcal{F}_Y,\end{aligned}$$

where $D_x = \{y \in Y \mid (x, y) \in D\}$ and $D_y = \{x \in X \mid (x, y) \in D\}$.

Given three measures μ_1, μ_2 and μ_3 on $\mathbb{X}$, we use the shorthand

$$\frac{\mathrm{d}^2\mu_1}{\mathrm{d}\mu_2\,\mathrm{d}\mu_3}$$

to express Radon-Nikodym derivative between μ_1 and the product measure $\mu_2 \times \mu_3$.

A.2 Probability measures and random variables

We use the concept of a measure to express probabilities. We extend a measurable space $(\Omega, \mathcal{F})$ to a *probability space* by combining it with a measure $P : \mathcal{F} \to [0, 1]$ which also satisfies $P(\Omega) = 1$. Such a measure is called a *probability measure*. We use a probability space $(\Omega, \mathcal{F}, P)$ to model a random event. In which case the outcomes of the event are in Ω, which is called the sample space. We now express the probability of a set of outcomes $A \in \mathcal{F}$ as $P(A)$.

Random variables. In order to model more complicated outcomes we use the concept of a *random variable*. Given a probability space $(\Omega, \mathcal{F}, P)$ and a measurable space $\mathbb{X} = (X, \mathcal{G})$, a random variable X is a measurable function from Ω to X. The variable allows us to express the probability of any set $A \in \mathcal{G}$ as

$$P(\mathcal{X} \in A) = P(X^{-1}(A)).$$

This naturally induces a measure $P_\mathcal{X}$ on $\mathbb{X}$, where for any $A \in \mathcal{G}$,

$$P_\mathcal{X}(A) = P(\mathcal{X} \in A).$$

$P_\mathcal{X}$ is also referred to as a *probability distribution* and we say that $\mathcal{X}$ is distributed according to $P_\mathcal{X}$. This can also be expressed with the shorthand $\mathcal{X} \sim P_\mathcal{X}$. Random variables are commonly used to model measurements, where $\mathbb{X} = \mathbb{R}$ and $\mathcal{G}$ is the family of Lebesgue-measurable subsets of $\mathbb{R}$.

Probability density functions. The *probability density function* (PDF), $p_\mathcal{X} : X \to [0, \infty]$, for $\mathcal{X}$ describes the probability of the elements in X and is defined using the Radon-Nikodym derivative

$$p_\mathcal{X} = \frac{\mathrm{d}P_\mathcal{X}}{\mathrm{d}\mu}$$

for some reference measure μ. When $\mathbb{X} \subset \mathbb{R}$, μ is commonly the Lebesgue measure. However, many light transport algorithms make heavy use of different reference measures.

Let $\mathcal{X} : \Omega_{\mathcal{X}} \to (\mathbb{X}, \mathcal{G}_{\mathcal{X}})$ be a random variable. With the help of its PDF, $p_{\mathcal{X}}$, we can evaluate the probability for the set of outcomes $A \in \mathcal{G}_{\mathcal{X}}$ with the integral

$$P(\mathcal{X} \in A) = \int_A p_{\mathcal{X}}(x) \, \mathrm{d}\mu(x),$$

where μ is the reference measure for $p_{\mathcal{X}}$.

Discrete random variables. With a suitable choice of measure the concepts presented in this chapter also extend to the case where the sample space is discrete. In this case we speak of *probability mass functions (PMF)* instead of PDFs and the integrals are naturally replaced by sums. For example, for a discrete random variable $\mathcal{X} : \Omega_{\mathcal{X}} \to (X, \mathcal{G}_{\mathcal{X}})$, we evaluate the probability for the set of outcomes $A \in \mathcal{G}_{\mathcal{X}}$ using the sum

$$P(\mathcal{X} \in A) = \sum_{x' \in A} p_{\mathcal{X}}(x').$$

A.3 Joint probability measures

A joint probability measure informs us of the interplay between two or more random variables. Here we consider pairs of random variables, but the concept can easily be extended to account for more. Given two probability spaces $(\Omega_1, \mathcal{F}_1, P_1)$ and $(\Omega_2, \mathcal{F}_2, P_2)$, we refer to a measure P defined on $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2$ as a joint probability measure.

Now consider two continuous random variables,

$$\mathcal{X}_1 : \Omega_1 \to (X_1, \mathcal{G}_1) \text{ and } \mathcal{X}_2 : \Omega_2 \to (X_2, \mathcal{G}_2).$$

which we combine into a single random variable

$$\mathcal{X} : \Omega_1 \times \Omega_2 \to (X_1, \mathcal{G}_1) \times (X_2, \mathcal{G}_2).$$

In order to evaluate the joint probability for a set of outcomes $A \in X_1 \times X_2$ from $\mathcal{X}$, we use the joint probability measure P and get

$$P(\mathcal{X} \in A) = P(\mathcal{X}^{-1}(A)).$$

We refer to the PDF $p_{\mathcal{X}} : X_1 \times X_2 \to [0, \infty]$ as the joint PDF of $\mathcal{X}_1$ and $\mathcal{X}_2$. Note that a joint probability measure has to satisfy

$$p_{\mathcal{X}_1}(x_1) = \int_{\mathbb{X}_2} p_{\mathcal{X}}(x_1, x_2) \, \mathrm{d}\mu_2(x_2) \text{ and } p_{\mathcal{X}_2}(x_2) = \int_{\mathbb{X}_1} p_{\mathcal{X}}(x_1, x_2) \, \mathrm{d}\mu_1(x_1),$$

where μ_1 and μ_2 are the reference measures for $p_{\mathcal{X}_1}$ and $p_{\mathcal{X}_2}$.

We call $\mathcal{X}_1$ and $\mathcal{X}_2$ *independent* if the PDF of the joint variable can be expressed as the product of the PDFs of its components. In other words,

$$p_{\mathcal{X}}(x_1, x_2) = p_{\mathcal{X}_1}(x_1) p_{\mathcal{X}_2}(x_2)$$

for all $x_1 \in \mathbb{X}_1, x_2 \in \mathbb{X}_2$. Intuitively, two random variables are independent if the outcome of one does not have an impact on the probabilities of the other.

We call $p_{\mathcal{X}_1}$ and $p_{\mathcal{X}_2}$ *marginal PDFs*. Intuitively, the marginal PDF of $\mathcal{X}_1$ describes its probability if we completely disregard the outcomes of $\mathcal{X}_2$. In contrast, the *conditional PDF* $p_{\mathcal{X}}(x_1|x_2)$ describes the probability of $\mathcal{X}_1$ if we fix the value of $\mathcal{X}_2$ to x_2. In other words,

$$p_{\mathcal{X}}(x_1|x_2) = \frac{p_{\mathcal{X}}(x_1, x_2)}{p_{\mathcal{X}_2}(x_2)} \text{ and } p_{\mathcal{X}}(x_2|x_1) = \frac{p_{\mathcal{X}}(x_1, x_2)}{p_{\mathcal{X}_1}(x_1)}.$$

A.4 Expected value and variance

One way to better understand random variables is to compute various summarizing quantities. One of the most descriptive of these quantities is the mean or the expected value, which tells us where we can expect to find the average of multiple outcomes from the event random variable. For a random variable $\mathcal{X} : \Omega_{\mathcal{X}} \to (X, \mathcal{G}_{\mathcal{X}})$ the expected value $E[\mathcal{X}]$ is

$$E[\mathcal{X}] = \int_X x \, \mathrm{d}P_{\mathcal{X}}(x) = \int_X x p_{\mathcal{X}}(x) \, \mathrm{d}\mu(x)$$

where μ is the reference measure for $p_{\mathcal{X}}$. We can also compute expected values for functions of random variables. For example, given a function $f : X \to \mathbb{R}^N$ we have

$$E[f(\mathcal{X})] = \int_X f(x) \, \mathrm{d}P_{\mathcal{X}}(x) = \int_X f(x) p_{\mathcal{X}}(x) \, \mathrm{d}\mu(x)$$

where the reference measure μ for $p_{\mathcal{X}}$ commonly is the Lebesgue measure. The linearity of integration extends to expected values as well. Given two random variables $\mathcal{X}$ and $\mathcal{Y}$ and the real numbers a and b, we have

$$E[a\mathcal{X} + b\mathcal{Y}] = aE[\mathcal{X}] + bE[\mathcal{Y}].$$

Another useful quantity is the *variance* of a real-valued random variable. The variance for $\mathcal{X} : \Omega_{\mathcal{X}} \to \mathbb{R}$ is

$$Var(\mathcal{X}) = E[(\mathcal{X} - E[\mathcal{X}])^2].$$

We see that for any real number a

$$Var(a\mathcal{X}) = a^2 Var(\mathcal{X})$$

because of the linearity of expected values. This also allows us to express the variance in the simpler form

$$Var(\mathcal{X}) = E[\mathcal{X}^2] - E[\mathcal{X}]^2.$$

The square root of the variance, $\sigma(\mathcal{X}) = \sqrt{Var(\mathcal{X})}$ is called the *standard deviation* and tells us how outspread we expect the outcomes of $\mathcal{X}$ to be.

A.5 Cumulative distribution functions and quantiles

We can define a *cumulative distribution function* (CDF) for any real valued random variable. Given the outcome $x' \in \mathbb{R}$ from the random variable $\mathcal{X} : \Omega_{\mathcal{X}} \to \mathbb{R}$, the CDF $F_{\mathcal{X}} : \mathbb{X} \to [0, 1]$ gives us the probability of all the outcomes less than or equal to x. In other words,

$$F_{\mathcal{X}}(x') = \int_{-\infty}^{x'} \mathrm{d}P_{\mathcal{X}}(x) = \int_{-\infty}^{x'} p_{\mathcal{X}}(x)\,\mathrm{d}\mu(x)$$

where μ is the Lebesgue measure on $\mathbb{R}$.

With the help of the CDF, we can introduce the concept of a *quantile*. Consider the random variable $\mathcal{X} : \Omega_{\mathcal{X}} \to \mathbb{R}$ and a value $a \in [0, 1]$. The a-quantile of $\mathcal{X}$ is the smallest number x, such that

$$F_{\mathcal{X}}(x) \geq a.$$

A common example is the 0.5-quantile, or the *median* which splits the outcomes of the random variable into two equally likely sets.

A.6 Common probability distributions

The uniform distribution. A random variable is uniformly distributed if its PDF is constant. In other words, given some measurable set $A \subset \mathbb{X}$ the variable

$\mathcal{X} : \Omega_{\mathcal{X}} \to A$ is uniformly distributed if

$$p_{\mathcal{X}}(x) = \frac{1}{\mu(A)}$$

for every $x \in A$. We denote this by $\mathcal{X} \sim \mathcal{U}(A)$.

For real-valued random variables we commonly define the uniform distribution on intervals. For example, the PDF of a real-valued random variable $\mathcal{X} \sim \mathcal{U}([a, b])$ is

$$p_{\mathcal{X}}(x) = \frac{1}{b-a} \text{ for all } x \in [a, b]$$

and we can easily show that

$$E[\mathcal{X}] = \frac{b-a}{2}$$

and

$$Var[\mathcal{X}] = \frac{(b-a)^2}{12}.$$

The normal distribution. The *normal distribution* is a distribution often used to model events where the actual distribution is unknown. This distribution is defined on $\mathbb{R}^n$ but we will only introduce it for real-valued random variables. In this case the normal distribution $\mathcal{N}(m, V)$ is uniquely defined by two parameters, the mean m and the variance V.

The PDF of a real-valued random variable $\mathcal{X} \sim \mathcal{N}(m, V)$ is

$$p_{\mathcal{X}}(x) = \frac{1}{\sqrt{2\pi V}} e^{-\frac{(x-m)^2}{2V}}$$

for every $x \in \mathbb{R}$. As one might assume, it can be shown that $E[\mathcal{X}] = m$ and $Var(\mathcal{X}) = V$.