A Survey of Importance Sampling Applications in Unbiased Physically Based Rendering

Dietger van Antwerpen

April 30, 2011

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

In this survey, we will discuss the application of importance sampling to the field of unbiased physically based rendering. Unbiased rendering algorithms are based on the Monte Carlo method. In this context, importance sampling techniques are used to reduce the variance of the Monte Carlo estimate. Importance sampling has played an important role in the development of the field of unbiased rendering and has let to three major rendering algorithms: Path Tracing (PT), Bidirectional Path Tracing (BDPT) and Metropolis Light Transport (MLT). In our discussion, we will focus on these algorithms, discussing them in detail. Furthermore, we will more briefly discuss important extensions to these algorithms, based on importance sampling.

In chapter 2, we introduce the subject of physically based rendering, followed by an introduction of the Monte Carlo method and the notion of importance sampling in chapter 3. Furthermore, we discuss the general application of the Monte Carlo method to physically based rendering. This leads to the three major unbiased rendering algorithms, discussed in the following chapters. In chapter 4, we discuss the PT algorithm. BDPT is discussed in chapter 5 and in chapter 6, we discuss the MLT algorithm. Finally, we finish with a conclusion in chapter 7.

Chapter 2

Physically Based Rendering

In this chapter, we discuss the subject of physically based rendering. We start with an introduction into physically based rendering in section 2.1. We than introduce the rendering equation for physically based light transport in free space as developed by Kajiya [10] in section 2.2. Following, in section 2.3 we discuss an alternative rendering equation developed by Veach [22]. This formulation is based on the rendering equation by Kajiya but is defined over the space of all light transport paths from light source to eye. Finally, in section 2.4 we discuss the possible existence of singularities in the rendering equation and a classification of light transport paths based on singularities, as introduced by Heckbert [7].

2.1 Physically Based Rendering

Generally speaking, physically based rendering is the synthesis of images according to the laws of physics. Given a mathematical description of a scene, a virtual photo of the scene is synthesized according to the laws of physics describing light transport in the real world. In practice, a simplified model of light transport is used in physically based rendering in order to make rendering computationally feasible. For example, most renderers only consider one or a few wavelengths instead of a whole band of frequencies. Light effects such as polarization, interference and redshift that are seldom perceived directly by humans are only considered in very specific scientific applications. Also, although light has a finite velocity, practically every physically based renderer assumes the scene illumination to always be in an equilibrium state. Furthermore, many renderers assume that light travels through vacuum, only interacting with the scene at the surface of the scene geometry. This excludes effects due to participating media (such as fog) and subsurface scattering within objects (such as a wax candle). In this document, we will not discuss these effects either and only consider the interaction between light and scene at the geometry surfaces.

In order to render an image of a scene, a description of the scene is required. Because of the assumption that light only interacts with the scene at geometry surfaces, the geometry is defined by its surfaces. In general, any representation of surface geometry suffices, as long as it is possible to compute the intersection point of a ray with the geometry. In practice, scene geometry is usually represented as a polygon mesh (see figure 2.1). Many polygons are required to accurately describe curved surfaces, therefore alternative representations such as NURBS, allowing certain forms of curvature, are also used. However, because the intersec-

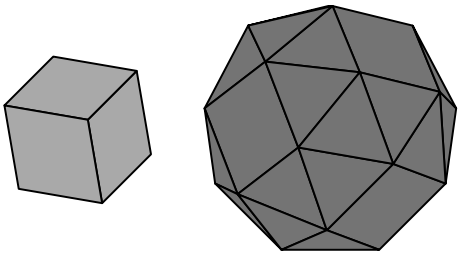

Figure 2.1: Scene geometry, modeled by polygons.

tion between a ray and a curved representation is often much harder to compute than the intersection with a polygon, polygons are still the representation of choice for most renderers.

For each surface point, it is also necessary to describe the material properties concerning light transport. Nicodemus proposed to describe the material of a surface by its *Bidirectional Reflection Distribution Function*(BRDF) $f_r(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i)$ [17]. This function gives the fraction of incoming radiance $\vec{\omega}_i$ at surface point $\mathbf{x}$ that is reflected in direction $\vec{\omega}_o$. For most materials, part of the incoming radiance is absorbed while the remaining radiance is reflected in different directions (Figure 2.2). For translucent materials, part of the light may be refracted and enter

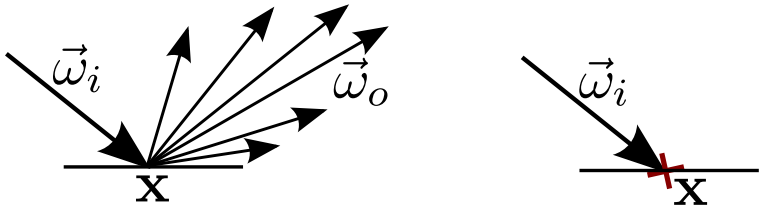

Figure 2.2: Part of the radiance is absorbed while the remaining radiance is reflected in different directions.

the material. If this is the case, the BRDF is also defined for incoming and outgoing directions on opposite sides of the surface (Figure 2.3). Such BRDF's must account for refraction due to varying material densities on opposite sides of the surface.

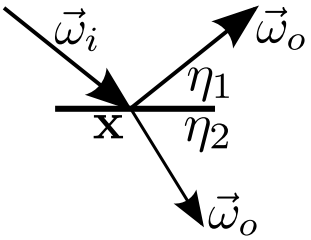

Figure 2.3: For translucent materials, part of the light is scattered to the other side of the surface.

In practice, a BRDF for a material is often approximated by a combination of multiple simpler BRDF's. Figure 2.4 gives such an example. The material has a pure diffuse component f_d, scattering incoming radiance equally in all directions, and a highly specular component f_s, scattering light around the direction of pure reflection.

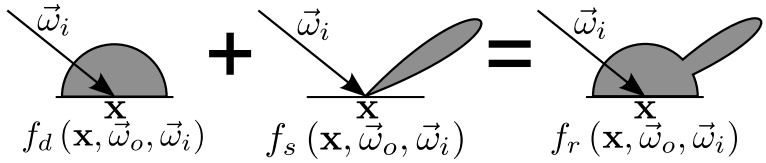

Figure 2.4: A complex BRDF is often approximated by combination multiple simpler BRDF's

Another common method for modeling physically accurate materials is by measuring the BRDF in real life for various combinations of incoming and outgoing directions and using these sample points to approximate the BRDF during rendering.

Finally, the light sources in the scene must be modeled as well. Usually, the light sources are a part of the scene geometry. Each surface point may or may not emit radiance into the scene. The function $L_e(\mathbf{x}, \vec{\omega}_o)$ describes the radiance emitted along $\vec{\omega}_o$ by the surface at surface point $\mathbf{x}$. If L_e is non-zero, the surface point $\mathbf{x}$ lies on a light source with L_e giving the emitted radiance distribution of the light source over the unit hemisphere(see figure 2.5).

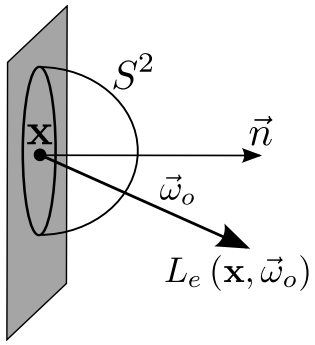

Figure 2.5: L_e describes the emitted radiance per surface point over the whole scene geometry.

For the rest of this document, the scene is assumed a given. We will not further discuss the scene representation or the possible BRDF and emitter models.

2.2 Rendering Equation

Kajiya developed a general rendering equation for physically based light transport in free space [10]. This equation describes how the radiance $L_o(\mathbf{x}, \vec{\omega}_o)$ leaving a surface point $\mathbf{x}$ in direction $\vec{\omega}_o$ relates to the incoming radiance arriving at $\mathbf{x}$ (figure 2.6). Because human perception is relatively slow compared to the speed of light, light transport is usually perceived in equilibrium. The rendering equation describes this equilibrium state:

$$L_o(\mathbf{x}, \vec{\omega}_o, \lambda) = L_e(\mathbf{x}, \vec{\omega}_o, \lambda) + \int_{S^2} f_r(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i, \lambda) L_i(\mathbf{x}, \vec{\omega}_i, \lambda) d\sigma^{\perp}(\vec{\omega}_i) \tag{2.1}$$

In this equation, $L_o(\mathbf{x}, \vec{\omega}_o, \lambda)$ is the radiance of wavelength λ leaving $\mathbf{x}$ in direction $\vec{\omega}_o$. The radiance leaving $\mathbf{x}$ relates to the amount of emitted radiance $L_e(\mathbf{x}, \vec{\omega}_o, \lambda)$ and the amount of incoming radiance that is scattered in direction $\vec{\omega}_o$.

The amount of incoming radiance that is scattered in direction $\vec{\omega}_o$ is described by the second term in equation (2.1). For each incoming direction $\vec{\omega}_i$, $L_i(\mathbf{x}, \vec{\omega}_i, \lambda)$ equals the incoming

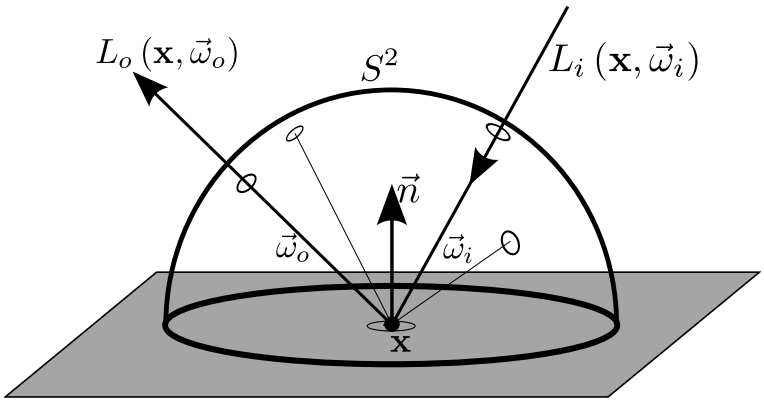

Figure 2.6: The light transport equation relates the outgoing radiance to the incoming radiance at a surface point.

radiance at $\mathbf{x}$ and the BRDF $f_r(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i, \lambda)$ describes the fraction of this incoming radiance that is scattered towards $\vec{\omega}_o$. The total amount of radiance scattered along $\vec{\omega}_o$ is obtained by integrating over all incoming directions on the hemisphere S^2 around $\mathbf{x}$.

The use of λ in the rendering equation indicates that light transport depends on the wavelength of the light involved. From now on, we will drop the λ in our notation for convenience. Note however that for accurate rendering of some light effects, it is important to consider an entire band of wavelengths instead of only a single wavelength. Accounting for multiple wavelengths during rendering is called spectral rendering.

In the context of rendering, the transport equation is often called the *rendering equation*. Veach showed that when rendering an image, each image pixel can be modeled as a radiance sensor [22]. Each sensor j measures radiance over surface area and has a sensor sensitivity function $W_j(\mathbf{x}, \vec{\omega}_i)$ describing the sensors sensitivity to radiance arriving at $\mathbf{x}$ along direction $\vec{\omega}_i$ (Figure 2.7). The total radiance I_j measured by a sensor j is described by the *measurement*

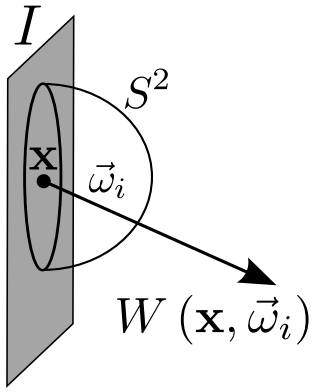

Figure 2.7: The sensor is described by the sensor sensitivity function on its surface.

equation:

$$I_j = \int_{I \times S^2} W_i(\mathbf{x}, \vec{\omega}_i)\, L_i(\mathbf{x}, \vec{\omega}_i)\, dA_I(\mathbf{x})\, d\sigma^{\perp}(\vec{\omega}_i) \tag{2.2}$$

In this equation, I is the surface area of the image plane. The sensor sensitivity function W_j is defined over the surface of the image plane. The measured radiance along direction $\vec{\omega}_i$ equals the incoming radiance $L_i(\mathbf{x}, \vec{\omega}_i)$ along this direction multiplied by the sensors sensitivity $W_j(\mathbf{x}, \vec{\omega}_i)$. Integrating over both the image plane and all incoming directions on the hemisphere S^2 gives the total radiance measured by the sensor.

The render equation (2.1) and measurement equation (2.2) integrate over the unit hemisphere S^2 around a surface point $\mathbf{x}$ and are expressed per unit projected solid angle $d\sigma^{\perp}(\vec{\omega})$. In practice, it is however often more convenient to express these equations per unit solid angle $d\sigma(\vec{\omega})$ or per unit area $dA(\mathbf{x}')$. When the integral is expressed per unit area, the incoming radiance is integrated over the scene's surface geometry M instead of the unit hemisphere S^2. The relation between the units *solid angle*, *projected solid angle* and *surface area* is depicted in figure 2.8 and are given by

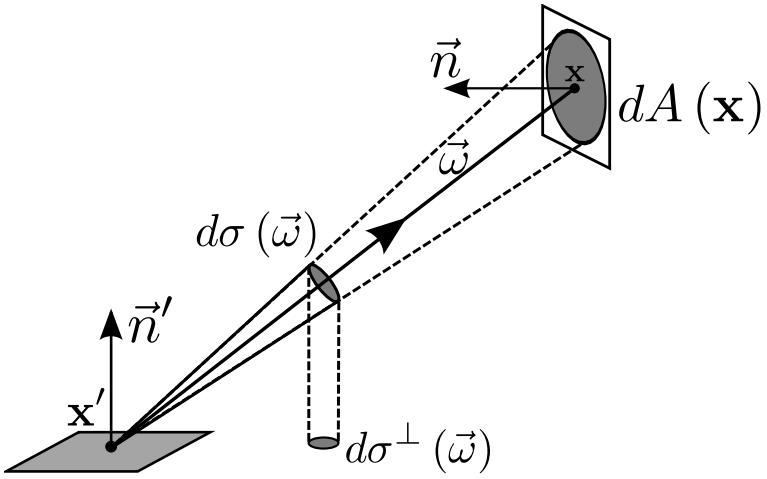

Figure 2.8: The geometric relation between the units projected solid angle, solid angle and surface area

$$d\sigma^{\perp}(\vec{\omega}) = \left|\vec{\omega} \cdot \vec{n}'\right| d\sigma(\vec{\omega}) = G\left(\mathbf{x} \leftrightarrow \mathbf{x}'\right) dA(\mathbf{x}) \tag{2.3}$$

In these transformations, $G(\mathbf{x} \leftrightarrow \mathbf{x}')$ is called the *geometric factor* and equals

$$G\left(\mathbf{x} \leftrightarrow \mathbf{x}'\right) = V\left(\mathbf{x} \leftrightarrow \mathbf{x}'\right) \frac{\left|\vec{\omega} \cdot \vec{n}\right| \left|\vec{\omega} \cdot \vec{n}'\right|}{\left|\mathbf{x} - \mathbf{x}'\right|^2} \tag{2.4}$$

where $V(\mathbf{x} \leftrightarrow \mathbf{x}')$ is called the *visibility factor*, being 1 iff the two surface points $\mathbf{x}$ and $\mathbf{x}'$ are visible from one another and 0 otherwise.

Using these transformations, the rendering equation can easily be expressed over unit area instead of projected solid angle, resulting in the *three-point form* of the rendering equation. This equation describes how radiance arriving at surface point $\mathbf{x}''$ from surface point $\mathbf{x}'$ relates to radiance arriving at $\mathbf{x}'$ (Figure 2.9). The equation is similar to equation (2.1) but is expressed over surface area:

$$L_o\left(\mathbf{x}' \rightarrow \mathbf{x}''\right) = L_e\left(\mathbf{x}' \rightarrow \mathbf{x}''\right) + \int_M f_r\left(\mathbf{x} \rightarrow \mathbf{x}' \rightarrow \mathbf{x}''\right) L_i\left(\mathbf{x} \rightarrow \mathbf{x}'\right) G\left(\mathbf{x} \leftrightarrow \mathbf{x}'\right) dA(\mathbf{x}) \tag{2.5}$$

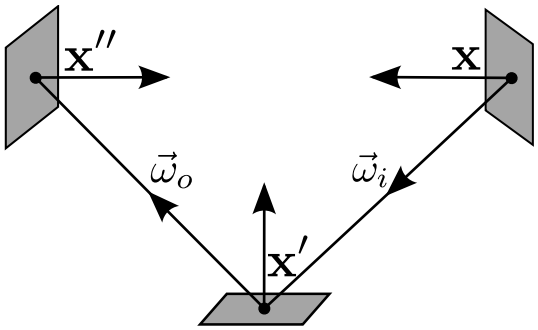

Figure 2.9: Three-point form of light transport

In this equation, $\mathbf{x}' \rightarrow \mathbf{x}''$ indicates the flow of radiance from $\mathbf{x}'$ to $\mathbf{x}''$ and $\mathbf{x} \rightarrow \mathbf{x}' \rightarrow \mathbf{x}''$ indicates the flow of radiance from $\mathbf{x}$ arriving at $\mathbf{x}'$ and then being scattered towards $\mathbf{x}''$. Using this notation, the equation becomes more intuitive: $f_r\left(\mathbf{x} \rightarrow \mathbf{x}' \rightarrow \mathbf{x}''\right)$ describes the fraction of incoming radiance from $\mathbf{x}$ arriving at $\mathbf{x}'$ that is scattered towards $\mathbf{x}''$. Similar, $L\left(\mathbf{x}' \rightarrow \mathbf{x}''\right)$ equals the radiance from $\mathbf{x}'$ arriving at $\mathbf{x}''$.

Similar to the three-point rendering equation, it is possible to express the measurement equation over unit surface area instead of projected solid angle. Substituting equation (2.3) into equation (2.2) gives

$$I_i = \int_{I \times M} W_i\left(\mathbf{x}' \rightarrow \mathbf{x}\right) L_i\left(\mathbf{x}' \rightarrow \mathbf{x}\right) G\left(\mathbf{x} \leftrightarrow \mathbf{x}'\right) dA\left(\mathbf{x}\right) dA\left(\mathbf{x}'\right) \tag{2.6}$$

Or, more conveniently, the measurement equation is written as the integral over a single measurement function f_i

$$f_i\left(\mathbf{x}_0, \mathbf{x}_1\right) = W_i\left(\mathbf{x}_1 \rightarrow \mathbf{x}_0\right) L_i\left(\mathbf{x}_1 \rightarrow \mathbf{x}_0\right) G\left(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1\right) \tag{2.7}$$

Resulting in the measurement equation

$$I_i = \int_{I \times M} f_i\left(\mathbf{x}_0, \mathbf{x}_1\right) dA\left(\mathbf{x}_0\right) dA\left(\mathbf{x}_1\right) \tag{2.8}$$

Physically based rendering concerns itself with estimating this measurement equation for each image pixel.

2.3 Path Space

The rendering equation as developed by Kajiya describes the light transport as a local equilibrium at any point in the scene. Starting from this rendering equation, Veach developed an alternative rendering equation, defined over the space of all light transport paths from light source to eye [22].

In vacuum, the radiance leaving $\mathbf{x}$ towards $\mathbf{x}'$ equals the radiance reaching $\mathbf{x}'$ from $\mathbf{x}$, that is, the radiance is constant over the light path $\mathbf{x} \leftrightarrow \mathbf{x}'$. Hence,

$$L_i\left(\mathbf{x} \rightarrow \mathbf{x}'\right) = L_o\left(\mathbf{x}' \rightarrow \mathbf{x}\right) \tag{2.9}$$

In other words, the rendering equation is recursive as the incoming radiance at one surface point is related to the outgoing radiance at another. This corresponds to the fact that a photon can bounce through the scene for an arbitrary number of bounces before reaching some surface point $\mathbf{x}$. Each bounce corresponds to a scattering event at a surface point. For example, when unrolling one level of recursion in equation (2.5), we get

$$\begin{aligned} L_o(\mathbf{x}_1 \to \mathbf{x}_0) = {} & L_e(\mathbf{x}_1 \to \mathbf{x}_0) \\ & + \int_M f_r(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0)\, L_e(\mathbf{x}_2 \to \mathbf{x}_1)\, G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_1)\, dA(\mathbf{x}_2) \\ & + \int_M f_r(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0) \int_M f_r(\mathbf{x}_3 \to \mathbf{x}_2 \to \mathbf{x}_1)\, L_i(\mathbf{x}_3 \to \mathbf{x}_2)\, G(\mathbf{x}_3 \leftrightarrow \mathbf{x}_2)\, dA(\mathbf{x}_3) \\ & \cdot G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_1)\, dA(\mathbf{x}_2) \end{aligned} \tag{2.10}$$

In this equation, the radiance reaching $\mathbf{x}_0$ directly after being emitted at $\mathbf{x}_1$ is expressed by the first term. The radiance reaching $\mathbf{x}_0$ through a single bounce after being emitted is expressed by the second, non-recursive term. The remaining incoming radiance, arriving at $\mathbf{x}_0$ after at least two bounces, is expressed by the last term. In a similar manner, equation (2.5) can be unrolled completely, resulting in the expression

$$\begin{aligned} L_o(\mathbf{x}_1 \to \mathbf{x}_0) = {} & \\ & L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \\ & \sum_{k=2}^{\infty} \int_{M^k} l(x_0 x_1 x_2 \cdots x_k)\, dA(x_2) \cdots dA(x_k) \end{aligned} \tag{2.11}$$

Where $l : \bigcup_{i=2}^{\infty} M^{i+1} \to \mathbb{R}$ is the *radiance flow function* defined as

$$l(\mathbf{x}_0 \mathbf{x}_1 \mathbf{x}_2 \cdots \mathbf{x}_k) = \prod_{i=1}^{k-1} f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})\, G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})\, L_e(\mathbf{x}_k \to \mathbf{x}_{k-1}) \tag{2.12}$$

This expression is no longer recursive. Instead, it separately evaluates the radiance reaching $\mathbf{x}_0$ through i bounces after being emitted at $\mathbf{x}_i$. The sequence of surface points $\mathbf{x}_i \cdots \mathbf{x}_0$ forms a light transport path of length i over which the emitted radiance flows from $\mathbf{x}_i$ to $\mathbf{x}_0$. By integrating over all such paths, the radiance arriving at $\mathbf{x}_0$ over paths of length i is obtained. Summing over all possible path lengths gives the total radiance $L_o(\mathbf{x}_1 \to \mathbf{x}_0)$ reaching $\mathbf{x}_0$.

Using this formulation, it is possible to rewrite the measurement equation as an integral over the space of all light transport paths. A light transport path $\mathbf{X}^k$ is any sequence of surface points $\mathbf{x}_0 \cdots \mathbf{x}_{k+1} \in I \times M^k$ of which the first point $\mathbf{x}_0$ lies on the image plane I and the remaining lie on the scene's surface geometry M. Figure 2.10 shows an example of a light transport path of length 5. The set of all light transport paths of length k is defined as $\Omega_k = I \times M^k$ and unified path space is defined as the union of all paths of all lengths and equals $\Omega = \bigcup_{i=1}^{\infty} \Omega i$.

Combining the radiance flow function from equation (2.12) and the measurement function from equation (2.7) gives rise to an alternative measurement function $f_i : \Omega \to \mathbb{R}$, defined

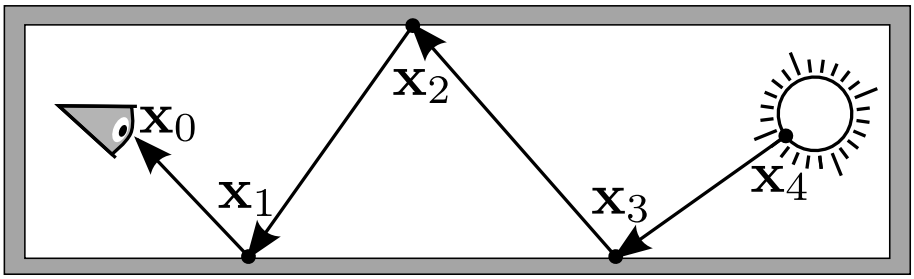

Figure 2.10: A Light Transport Path from light to eye of length 5.

over unified path space.

$$\begin{aligned} f_i\left(x_0 \cdots x_k\right) &= W_i\left(\mathbf{x}_1 \rightarrow \mathbf{x}_0\right) G\left(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1\right) \prod_{j=1}^{k-1} f_r\left(\mathbf{x}_{j+1} \rightarrow \mathbf{x}_j \rightarrow \mathbf{x}_{j-1}\right) G\left(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}\right) \cdot L_e\left(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1}\right) \\ &= W_i\left(\mathbf{x}_1 \rightarrow \mathbf{x}_0\right) \prod_{j=0}^{k-1} G\left(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1}\right) \cdot \prod_{j=1}^{k-1} f_r\left(\mathbf{x}_{j+1} \rightarrow \mathbf{x}_j \rightarrow \mathbf{x}_{j-1}\right) \cdot L_e\left(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1}\right) \end{aligned} \tag{2.13}$$

Figure 2.11 shows how the factors in this measurement function correspond to the example light transport path from figure 2.10. Using this measurement function, the measurement

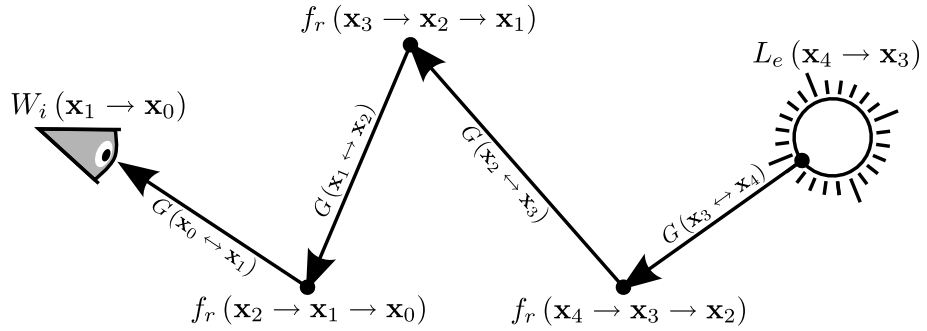

Figure 2.11: Correspondence between measurement function and an example path in unified path space

equation (2.8) can now be expressed over unified path space as

$$I_i = \int_{\Omega} f_i(\mathbf{X}) d\Omega\left(\mathbf{X}\right) \tag{2.14}$$

In this formulation, light transport is expressed with respect to unit area, so path space is integrated with respect to the area-product measure. Hence, unit path space $d\Omega\left(\mathbf{X}^k\right)$ corresponds to

$$d\Omega\left(\mathbf{x}_0 \cdots \mathbf{x}_k\right) = dA\left(\mathbf{x}_0\right) \times dA\left(\mathbf{x}_1\right) \times \cdots \times dA\left(\mathbf{x}_k\right) \tag{2.15}$$

The remainder of this document will concern itself with estimating the measurement equation over unified path space for each image pixel.

2.4 Singularities

Many of the aforementioned functions may contain singularities. For example, the BSDF of perfect reflective and refractive materials is modeled using Dirac functions. Furthermore, L_e may exhibit singularities to model purely directional light sources. Finally, common camera models such as the pinhole camera are modeled using Dirac functions in the sensor sensitivity functions W_j. Special care must be taken to support these singularities.

Heckbert introduced a notation to classify light transport paths using regular expressions of the form $E\,(S|D)^*\,L$ [7]. Each symbol represents a path vertex. E represents the eye and is the first path vertex. This vertex is followed by any number of path vertices, classified as specular(S) or diffuse(D) vertices. A path vertex is said to be *specular* whenever the scattering event follows a singularity in the local BSDF, also called a specular bounce. Finally, the last vertex is classified as a light source(L). We refer to the classification of a path as its signature.

Chapter 3

Monte Carlo Rendering

In this chapter we discuss the use of Monte Carlo sampling to estimate the measurement equation. In section 3.1, we discuss Monte Carlo integration, attributed to Neumann and Ulam [16]. In section 3.2, we discuss Importance Sampling as a variance reduction technique in Monte Carlo estimators. In section 3.3, we discuss an optimal method for combining multiple Importance Sampling strategies in the context of rendering, as presented by Veach [23]. Finally, in section 3.4 we discuss in some detail common methods for sampling light transport path vertices and how to transform the corresponding probabilities between important geometric units [22].

3.1 Monte Carlo Method

The goal of physically based rendering is to evaluate the measurement equation from last chapter for all image pixels. The measurement equation is an integral of the possibly non-continuous measurement function over the high-dimensional path space. It is therefore infeasible to evaluate the measurement equation analytically. Instead, it is estimated using numerical methods. Deterministic methods for numerical integration usually operate by taking a number of evenly spaced samples from a function. These methods however suffer from the *curse of dimensionality*; They do not apply well to integrals over high-dimensional spaces. Estimating the measurement equation using deterministic methods is therefor rather inefficient. Instead, the measurement equation is usually estimated using the Monte Carlo method, which relies on repeated random sampling to compute an estimate of an integral. The development of the Monte-Carlo method has been attributed to Neumann and Ulam [16]. In its basic form, the Monte Carlo method samples N points $\mathbf{X}_1, \cdots, \mathbf{X}_N$ in path space according to some convenient density function p, and then computes the estimate as

$$\hat{I}_i = \frac{1}{N} \sum_{j=1}^{N} \frac{f_i(\mathbf{X}_j)}{p(\mathbf{X}_j)} \tag{3.1}$$

It is straightforward to show the estimator $\hat{I}_i$ gives the correct result on average, provided that $\frac{f_i(\mathbf{X})}{p(\mathbf{X})}$ is finite whenever $f_i(\mathbf{X}) \neq 0$

$$\begin{aligned} E\left[\hat{I}_i\right] &= E\left[\frac{1}{N}\sum_{j=1}^{N}\frac{f_i(\mathbf{X}_j)}{p(\mathbf{X}_j)}\right] \\ &= \frac{1}{N}\sum_{j=1}^{N}\int_{\Omega}\frac{f_i(\mathbf{X})}{p(\mathbf{X})}p(\mathbf{X})\,d\Omega(\mathbf{X}) \\ &= \int_{\Omega} f_i(\mathbf{X})\,d\Omega(\mathbf{X}) \\ &= I_i \end{aligned} \tag{3.2}$$

An estimator that gives the correct result on average is called an *unbiased* estimator. The quality of the estimator is determined by its variance; the square of the expected error. Because the image pixels are estimated independently, the errors in these estimates are also independent. These errors therefore show up as high frequency noise in the final rendering. The advantage of using Monte Carlo integration is that the expected error in the estimate decreases at a rate $O\left(\sqrt{N}\right)$, regardless of the dimensionality of the sample space. Hence, the estimate can be made arbitrarily accurate by increasing the number of samples.

3.2 Importance Sampling

The Monte Carlo method relies on samples from path space. As explained in the last section, these samples may be sampled according to any convenient density function p, provided that $\frac{f(\mathbf{X})}{p(\mathbf{X})}$ is finite whenever $f(\mathbf{X}) \neq 0$. This however does not mean that the particular choice of p is irrelevant. The choice of p determines to a large extend the quality of the estimator. While a good choice of p results in a low variance estimator, a particularly bad choice of p can result in high variance in the estimator. Figure 3.1 shows the difference between a good and a bad

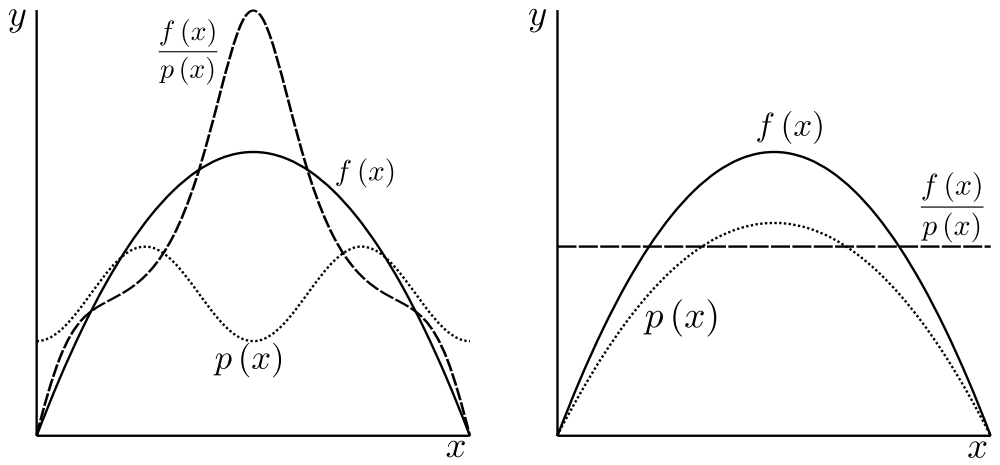

Figure 3.1: Sampling proportional to f results in a low variance estimator. This is called importance sampling.

choice of density function. Each sample $\mathbf{X}_j$ contributes $\frac{f(\mathbf{X}_j)}{p(\mathbf{X}_j)}$ to the estimate. When $\frac{f(\mathbf{X}_j)}{p(\mathbf{X}_j)}$ is roughly constant, each sample will contribute equally and the corresponding estimator will have relatively low variance. On the other hand, when $\frac{f(\mathbf{X}_j)}{p(\mathbf{X}_j)}$ various significantly over path space, the contributions of samples to the estimate will vary equally, resulting in high variance in the estimator. Ideally, when the probability density function p is proportional to f, the estimator will have zero variance. Sampling according to a density function p that is roughly proportional to f is called *Importance Sampling*. Importance Sampling is arguably the most important variance reduction technique in the context of unbiased physically based rendering. The remainder of this document will show how importance sampling is applied to physically based rendering.

3.3 Multiple Importance Sampling

When applying importance sampling to the Monte Carlo method, the density function p must be chosen to be roughly proportional to f. At the same time, it must be relatively easy to generate samples according to this chosen density function. In practice, it is difficult to find one such density function. Often, there are multiple possible importance sampling strategies resulting in different density functions, each accurately resembling f only on a part of the sample space while being less accurate on the rest of the sample space.

In the context of unbiased rendering, Veach proposed a method for combining multiple importance sampling strategies into a single estimate, called *Multiple Importance Sampling* (MIS) [23]. Instead of choosing only one of these strategies and generating all samples according to this strategy, MIS seeks to combine different importance sampling strategies into one optimal, unbiased estimate. Thus, instead of sampling all N samples according to the same sampling density function p, each sample $\mathbf{X}_i$ is sampled according to a different density function p_i. These samples are then combined into a single Monte Carlo estimate

$$\hat{I}_i = \sum_{j=1}^{N} w_j\left(\mathbf{X}_j\right) \frac{f\left(\mathbf{X}_j\right)}{p_j\left(\mathbf{X}_j\right)} \tag{3.3}$$

In this estimate, $w_j\left(\mathbf{X}_j\right)$ is a weight function used to combine the contributions of all samples into a single estimate. For the estimate to remain unbiased, the weight function must satisfy two conditions

$$\begin{aligned} w_i(\mathbf{X}) \neq 0 &\Longrightarrow p_i\left(\mathbf{X}\right) \neq 0 \\ f\left(\mathbf{X}\right) &\Longrightarrow \sum_{i=1}^{i=N} w_i(\mathbf{X}) = 1 \end{aligned} \tag{3.4}$$

In other words, there must be at least one strategy to sample each contributing sample and when some samples may be sampled with multiple strategies, the weights for these strategies must sum to one.

There are various possible combination strategies used in practice. An obvious strategy is to assign a constant weight w_i to each sampling strategy. However, this combination strategy does not work very well. Whenever one of the sampling strategies has a high variance, this high variance will also occur in the combination.

One common technique is to partition the sampling space among the sampling techniques, which is a form of stratification. The sample space Ω is partitioned in N parts $\Omega_1 \cdots \Omega_N$.

The integral is then estimated in each region Ω_i separately, using samples from strategy p_i only. The corresponding weight function equals:

$$w_i(\mathbf{X}) = \begin{cases} 1, & \text{if } \mathbf{X} \in \Omega_i \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

This technique is used extensively in physically based rendering, but it has the disadvantage that it is often difficult to determine how the space should be partitioned and which strategy should be applied to which part of the sample space. Using a fixed partition can lead to much higher variance than necessary.

A related combination strategy is the *maximum heuristic*. It partitions the sample space Ω into N regions according to which of the N sampling strategies has the highest probability of generating samples at each point in sample space. This corresponds to the following weight function:

$$w_i(\mathbf{X}) = \begin{cases} 0, & \text{if } p_i(\mathbf{X}) < p_j(\mathbf{X}) \vee i > j \text{ for all } j \\ 1, & \text{otherwise} \end{cases} \tag{3.6}$$

Note that whenever multiple sampling strategies have the same probability, the sampling strategy with the lowest index is chosen. The maximum heuristic chooses the sampling strategy with the lowest variance on each part of the sample space. This results in a relatively good combination. Although this method uses a fixed partition, the partition adapts to the particular scene, choosing the optimal sampling strategy over the sample space.

Another fairly straightforward weight function satisfying the weighting conditions is

$$w_i(\mathbf{X}) = \frac{p_i(\mathbf{X})}{\sum_{j=0}^{N} p_j(\mathbf{X})} \tag{3.7}$$

This weight function is called the *balance heuristic*. Veach showed that the balance heuristic is the best possible combination in the absence of further information [22]. In particular, he proved that no other combination strategy can significantly improve over the balance heuristic. The general form of the balance heuristic, called the *power heuristic*, is given by

$$w_i(\mathbf{X}) = \frac{p_i(\mathbf{X})^\beta}{\sum_{j=0}^{N} p_j(\mathbf{X})^\beta} \tag{3.8}$$

Note that the power heuristic becomes the maximum heuristic as $\beta \to \infty$.

3.4 Sampling Probability Conversion

When applying the Monte Carlo method to the measurement equation, one must somehow be able to sample a random light transport path $\mathbf{X}_k = x_0 \cdots x_k$ from path space Ω according to some known probability density function p. This probability must be evaluated with respect to the area-product measure $d\Omega(\mathbf{X}_k)$. One possible method would be to choose a path length k according to some density $p_l(k)$ and then independently sample one vertex $x_0 \in I$ on the image plane with probability $p_I(\mathbf{x}_0)$ and k independent vertices's $\mathbf{x}_1 \cdots \mathbf{x}_k \in M^k$ directly on the surface geometry according to some density $p_M(\mathbf{x}_i)$ over M (See figure 3.2). Although this method does generate valid light transport paths according to a known probability density

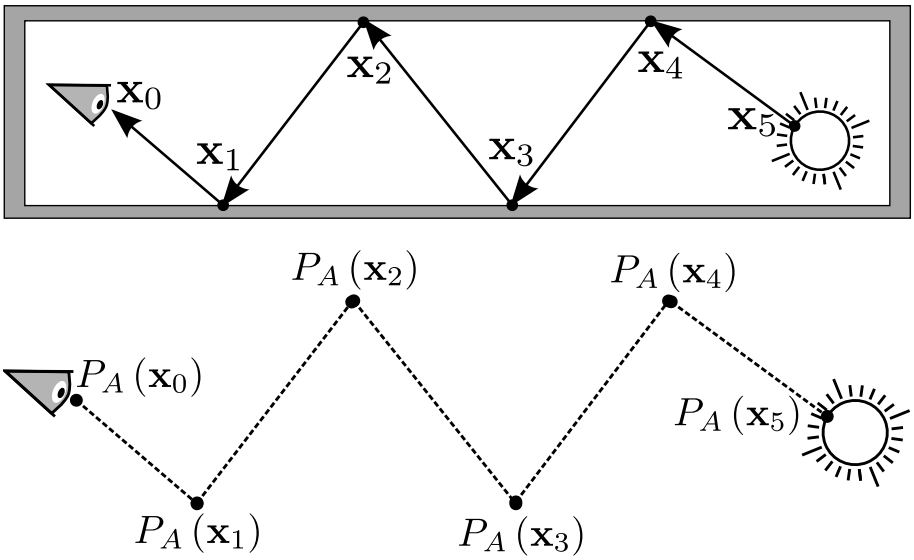

Figure 3.2: Light transport paths can be sampled by independently sampling all path vertices.

function, it is seldom an effective sampling strategy. Among others, an important reason for this is that in many scenes, two consecutive, independently sampled path vertices $x_i \to x_{i-1}$ will be occluded from one another with high probability. Such samples have no contribution to the estimator and are therefore of little value[1]. Remember that according to importance sampling, samples with zero contribution should be sampled with zero probability. Therefore, instead of sampling all path vertices independently, path vertices are usually sampled one after the other, where the sampling probability of each vertex $\mathbf{x}_i$ may depend on all vertices sampled so far.

Figure 3.3 shows three common methods for sampling path vertex $\mathbf{x}_i$, given the path vertices $\mathbf{x}_{i-1}$ and $\mathbf{x}_{i+1}$. The first method simply samples $\mathbf{x}_i$ directly on the surface geometry, independent of the other two path vertices(upper right). The second method samples $\mathbf{x}_i$ forward from $\mathbf{x}_{i-1}$. It does this by sampling an outgoing direction from $\mathbf{x}_{i-1}$ and tracing a ray into the scene to find $\mathbf{x}_i$ (lower left). Similar, the third method samples $\mathbf{x}_i$ backwards from $\mathbf{x}_{i+1}$ (lower right). All path sampling methods discussed in later chapters are based on these three methods for sampling individual path vertices.

Usually, when sampling a path vertex $\mathbf{x}_i$ using backward or forward sampling, the outgoing direction is sampled per unit (projected) solid angle. The probability of sampling $\mathbf{x}_i$ per projected solid angle through forward sampling is expressed as $P_{\vec{\sigma}^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i)$. Similar, the probability of sampling $\mathbf{x}_i$ backward per projected solid angle is expressed as $P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1})$. In order to compute the probability $P_M(\mathbf{x}_i)$ of sampling $\mathbf{x}_i$ per unit surface area, these probabilities need to be converted from unit projected solid angle to unit surface area. Using equation (2.8), these probabilities relate according to

$$G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})\, P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) = P_M(\mathbf{x}_i) = G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1})\, P_{\vec{\sigma}^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i) \tag{3.9}$$

See figure 3.3 for a visual representation of the sampling probabilities per unit surface area for forward and backward sampling.

[1]For occluded vertices, the visibility term $V(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1})$ is zero.

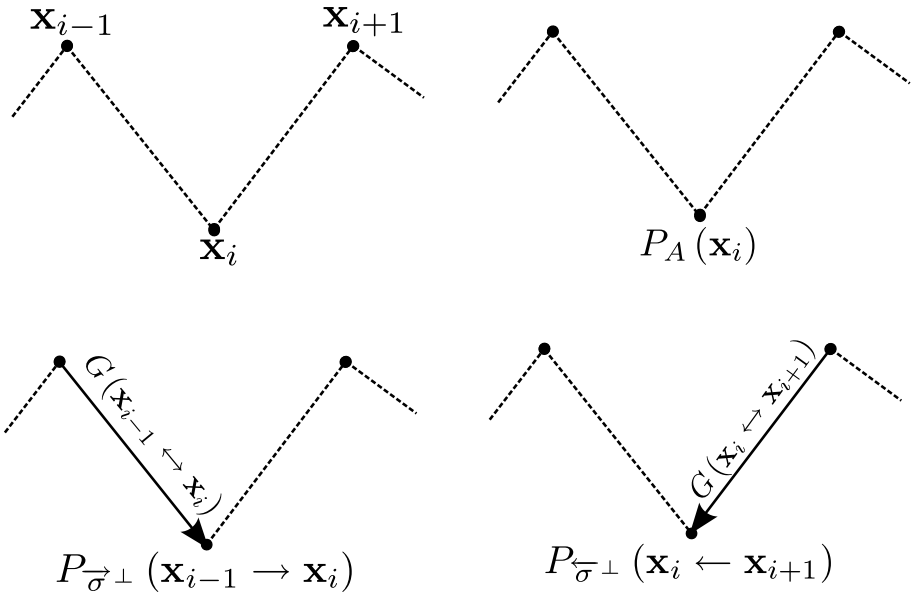

Figure 3.3: Three methods for sampling path vertex $\mathbf{x}_i$: Directly on the surface geometry, through forward tracing from vertex $\mathbf{x}_{i-1}$ and through backward tracing from vertex $\mathbf{x}_{i+1}$

Often a sequence of consecutive path vertices $\mathbf{x}_s \cdots \mathbf{x}_t$ is sampled using either forward or backward sampling. Using the conversion factors from equation 3.9, the probability of sampling such a sequence per unit area can be expressed in terms of the probabilities for sampling the individual vertices per unit projected solid angle. In case of forward sampling, given path vertex $\mathbf{x}_{s-1}$, the sampling probability per unit area for $\mathbf{x}_s \cdots \mathbf{x}_t$ equals

$$\prod_{i=s}^{t} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1}) \prod_{i=s}^{t} P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_{i-1} \rightarrow \mathbf{x}_i) = P_M(\mathbf{x}_i) \tag{3.10}$$

Similar, for backward sampling, given vertex $\mathbf{x}_{t+1}$, the sampling probability per unit area equals

$$\prod_{i=s}^{t} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \prod_{i=s}^{t} P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) = P_M(\mathbf{x}_i) \tag{3.11}$$

Using these two equations, it is also possible to relate the backward sampling probability for a sequence $\mathbf{x}_s \cdots \mathbf{x}_t$ to the corresponding forward sampling probability using

$$\begin{aligned} \prod_{i=s}^{t} P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) &= \prod_{i=s}^{t} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1}) \prod_{i=s}^{t} \frac{1}{G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})} \prod_{i=s}^{t} P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_{i-1} \rightarrow \mathbf{x}_i) \\ &= \frac{G(\mathbf{x}_s \leftrightarrow \mathbf{x}_{s-1})}{G(\mathbf{x}_t \leftrightarrow \mathbf{x}_{t+1})} \prod_{i=s}^{t} P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_{i-1} \rightarrow \mathbf{x}_i) \end{aligned} \tag{3.12}$$

It is possible to sample some path vertices using backward sampling while sampling other vertices using forward sampling or surface area sampling. For example, the sequence $\mathbf{x}_2 \cdots \mathbf{x}_t$ can be constructed by sampling the vertices $\mathbf{x}_2 \cdots \mathbf{x}_s$ using forward sampling and sampling

$\mathbf{x}_t \cdots \mathbf{x}_{s+1}$ using backward sampling. The corresponding probability conversion factor becomes

$$\prod_{i=2}^{s} G\left(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1}\right) \prod_{i=s+1}^{t} G\left(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}\right) \prod_{i=2}^{s} P_{\overrightarrow{\sigma}^{\perp}}\left(\mathbf{x}_{i-1} \rightarrow \mathbf{x}_i\right) \prod_{i=s+1}^{k} P_{\overleftarrow{\sigma}^{\perp}}\left(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}\right) = \prod_{i=2}^{k} P_M\left(\mathbf{x}_i\right) \tag{3.13}$$

Sampling a sequence by combining forward and backward sampling is called *bidirectional sampling*. Figure 3.4 shows a bidirectionally sampled light transport path. The vertices $\mathbf{x}_0$,$\mathbf{x}_1$ and $\mathbf{x}_5$ are sampled using surface area sampling, $\mathbf{x}_2$ is sampled using forward sampling and vertices $\mathbf{x}_3$ and $\mathbf{x}_4$ are sampled using backward sampling. The figure also shows the sampling probabilities, including conversion factors to convert from projected solid angle to unit area.

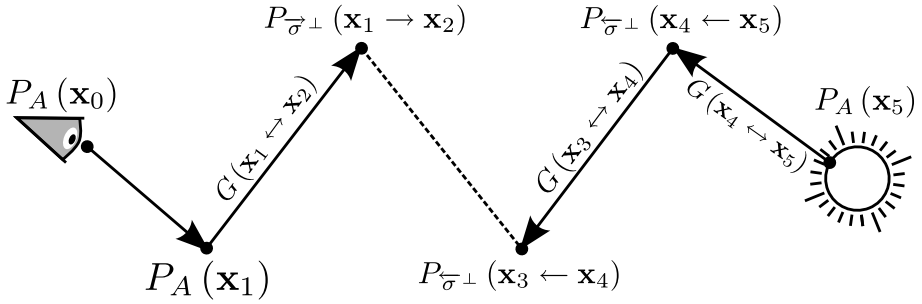

Figure 3.4: Bidirectional sampled light transport path, including probability conversion factors.

Camera Model

Usually, the first two path vertices $\mathbf{x}_0$ and $\mathbf{x}_1$ are sampled using special sampling methods that depend on the sensor sensitivity function $W_i\left(\mathbf{x}_1 \rightarrow \mathbf{x}_0\right)$. The sensor sensitivity function describes how the image is constructed from incoming radiance and corresponds to a *camera model*. For a given camera model and corresponding sampling methods for $\mathbf{x}_0$ and $\mathbf{x}_1$, the sensor sensitivity function is usually not specified explicitly but the following expression is specified instead

$$\hat{W}_i\left(\mathbf{x}_1 \rightarrow \mathbf{x}_0\right) = W_i\left(\mathbf{x}_1 \rightarrow \mathbf{x}_0\right) G\left(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1\right) \frac{1}{P_I\left(\mathbf{x}_0\right) P_M\left(\mathbf{x}_1\right)} \tag{3.14}$$

Where $P_I\left(\mathbf{x}_0\right)$ and $P_M\left(\mathbf{x}_1\right)$ are the probability of sampling respectively $\mathbf{x}_0$ and $\mathbf{x}_1$ using the sampling methods corresponding to the camera model. In this document we will take a simple finite aperture lens as an example, but other, more complex camera models would give similar results. Figure 3.5 visualizes the camera model used. The first vertex $\mathbf{x}_0$ is sampled on the camera lens with probability $P_I\left(\mathbf{x}_0\right)$. Vertex $\mathbf{x}_1$ is found by sampling a point $\mathbf{v}$ on the view plane V and tracing a ray from $\mathbf{x}_0$ through $\mathbf{v}$ into the scene. The first intersection between the ray and the scene geometry gives $\mathbf{x}_1$. $P_V\left(\mathbf{x}_1\right)$ is the probability of sampling $\mathbf{x}_1$ per unit view plane area, or in other words, the probability of sampling $\mathbf{v}$ on the view plane. When applying the Monte Carlo method to the measurement equation, it is necessary to convert this probability from unit view plane area to unit surface area. Figure 3.5 shows the relation

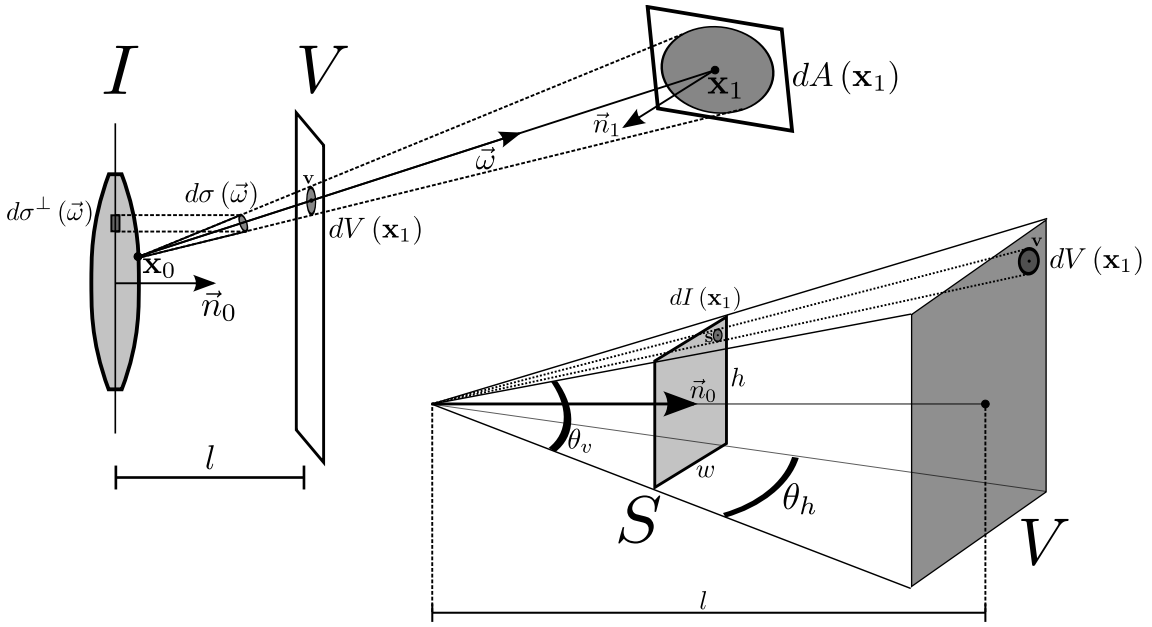

Figure 3.5: Camera model for a simple camera with finite aperture lens.

between these measures and the corresponding probabilities relate according to

$$\frac{P_V(\mathbf{x}_1)}{(\vec{n}_0 \cdot (\mathbf{x}_1 - \mathbf{x}_0))^3} = \frac{P_{\vec{\sigma}}(\mathbf{x}_0 \to \mathbf{x}_1)}{l^2 |\mathbf{x}_1 - \mathbf{x}_0|^3} = \frac{P_M(\mathbf{x}_1)}{l^2 \vec{n}_1 \cdot (\mathbf{x}_1 - \mathbf{x}_0)} \tag{3.15}$$

In this equation, l is the distance from lens to view plane and $\vec{n}_0$ is the view direction. Figure 3.5 also shows how the view plane V corresponds to screen space S, depending on l, the horizontal and vertical field of view angles θ_h and θ_v, and the image width and height w and h. The point $\mathbf{v}$ on the view plane is generated by sampling a point $\mathbf{s}$ in screen space S and constructing the corresponding point on the view plane. If $P_S(\mathbf{x}_1)$ is the probability of sampling $\mathbf{x}_1$ per unit screen space area (probability of sampling $\mathbf{s}$ in screen space), another conversion factor is required to convert from unit screen space area to unit view plane area

$$\frac{4l^2 \tan(\theta_h) \tan(\theta_v)}{wh} P_V(\mathbf{x}_1) = P_S(\mathbf{x}_1) \tag{3.16}$$

Or, using $\rho = \frac{4l^2 \tan(\theta_h) \tan(\theta_v)}{wh}$, shortly

$$\rho P_W(\mathbf{x}_1) = P_S(\mathbf{x}_1) \tag{3.17}$$

Putting everything together, the probability of sampling $\mathbf{x}_1$ per unit screen space area can be converted to the corresponding probability per unit surface area using

$$P_M(\mathbf{x}_1) = \frac{\rho (\vec{n}_0 \cdot (\mathbf{x}_1 - \mathbf{x}_0))^3}{l^2 \vec{n}_1 \cdot (\mathbf{x}_1 - \mathbf{x}_0)} P_S(\mathbf{x}_1) \tag{3.18}$$

Hence, for the finite aperture camera model, equation (3.14) becomes

$$\hat{W}_i(\mathbf{x}_1 \to \mathbf{x}_0) = W_i(\mathbf{x}_1 \to \mathbf{x}_0) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \frac{l^2 \vec{n}_1 \cdot (\mathbf{x}_1 - \mathbf{x}_0)}{\rho (\vec{n}_0 \cdot (\mathbf{x}_1 - \mathbf{x}_0))^3} \frac{1}{P_I(\mathbf{x}_0) P_S(\mathbf{x}_1)} \tag{3.19}$$

When $\mathbf{x}_0$ is sampled uniformly on the lens and $\mathbf{s}$ is sampled uniformly in screen space, $\hat{W}_i(\mathbf{x}_1 \to \mathbf{x}_0) = c$ for the simple finite aperture camera where c is a constant that determines the total sensor sensitivity.

Chapter 4

Path Tracing

In this chapter, we discuss the Path Tracing algorithm, based on the work of Cook and Kajiya [4, 26]. In sections 4.1 and 4.2, we give a detailed discussion of the algorithm. In section 4.3, we discuss the application of MIS to path tracing, as proposed by Veach [22]. In section 4.4, we present pseudo code for the Path Tracing algorithm. In the remaining sections, we discuss important extensions of the Path Tracing algorithm. In section 4.5, we discuss the work of Jensen. Jensen proposed a method to approximate the incoming radiance at a path vertex using a photon map and use this approximation to guide importance sampling during path extension [9]. In section 4.6 we discuss the work of Sherly and Talbot on importance resampling. Sherly proposed to reduce the impact of sampling proportional to an approximation of the actual importance through importance resampling. He applied this to the computation of direct lighting [19]. Talbot further generalized the method and applied it to path extension direction sampling [21, 20]. Finally, in section 4.7, we discuss the work of Bekaert. Bekaert proposed to combine paths contributing to different pixels to cheaply increase the number of samples per pixel [1].

4.1 Sampling

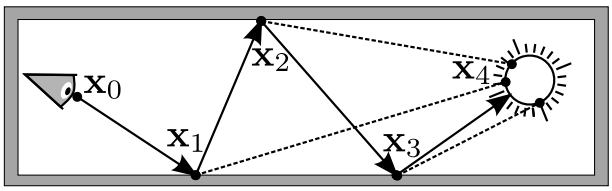

Figure 4.1: Path tracing sample.

The oldest unbiased light transport sampling algorithm, now called path tracing, was developed by Cook [4], based on whitted ray tracing as developed by Whitted [26]. Cooks deferred ray tracing algorithm has later been modified to solve the rendering equation from Kajiya [10]. In path tracing, a collection of paths is sampled forward, starting at the eye. Figure 4.1 shows a single path tracing sample. A sampler starts by tracing a path $\mathbf{y}_0 \cdots \mathbf{y}_k$ forward from the eye, through the image plane into the scene. The path is repeatedly *extended* with another path vertex until it is *terminated*. At each path vertex, the path is terminated

with a certain probability using *Russian roulette*. Each path vertex $\mathbf{y}_i$ is explicitly *connected* to a random point $\mathbf{z}$ on a light source to form the complete light transport path $\mathbf{y}_0 \cdots \mathbf{y}_i \mathbf{z}$ from the eye to the light source. These complete paths are called *explicit* paths. However, when the path 'accidentally' hits a light source at vertex $\mathbf{y}_k$, this path $\mathbf{y}_0 \cdots \mathbf{y}_k$ also forms a valid light transport path. Such paths are called *implicit* paths. Hence, light transport paths may be generated as either an explicit or implicit path. For simplicity, we assume that light sources do not reflect any light, so the path may be terminated whenever it hits a light source.

When a path is generated using path tracing, part of the path is sampled using forward sampling. In order to compute the Monte Carlo contribution of such a path, the probability of sampling these vertices per projected solid angle needs to be converted to unit surface area as explained in section 3.4. Using equations 3.10 and 3.18, the probability of generating an implicit path $\mathbf{x}_0 \cdots \mathbf{x}_k$ per unit path space can be expressed in terms of the sampling probabilities for the individual vertices as

$$\begin{aligned} p_I(\mathbf{x}_0 \cdots \mathbf{x}_k) =& \frac{l^2 \vec{n}_1 \cdot (\mathbf{x}_1 - \mathbf{x}_0)}{\rho(\vec{n}_0 \cdot (\mathbf{x}_1 - \mathbf{x}_0))^3} \prod_{i=1}^{k-1} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \cdot \\ & P_I(\mathbf{x}_0) P_S(\mathbf{x}_1) \prod_{i=2}^{k} P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i) \\ =& P_I(\mathbf{x}_0) \prod_{i=1}^{k} P_M(\mathbf{x}_i) \end{aligned} \tag{4.1}$$

The probability of sampling explicit paths is the same, except for the last path vertex which does not require any conversion factors because it is sampled directly per unit surface area

$$\begin{aligned} p_E(\mathbf{x}_0 \cdots \mathbf{x}_k) =& \frac{l^2 \vec{n}_1 \cdot (\mathbf{x}_1 - \mathbf{x}_0)}{\rho(\vec{n}_0 \cdot (\mathbf{x}_1 - \mathbf{x}_0))^3} \prod_{i=1}^{k-2} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \cdot \\ & P_I(\mathbf{x}_0) P_S(\mathbf{x}_1) \prod_{i=2}^{k-1} P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i) P_M(\mathbf{x}_k) \\ =& P_I(\mathbf{x}_0) \prod_{i=1}^{k} P_M(\mathbf{x}_i) \end{aligned} \tag{4.2}$$

Note that during path tracing, the inverse probability for Russian roulette termination at vertex $\mathbf{x}_{i-1}$ is assumed to be incorporated in the forward sampling probability $P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i)$.

As explained in section 3.2, the variance in the estimate can be reduced significantly by applying importance sampling, thus by generating light transport paths proportional to their measurement function f. In general it is not possible to sample proportional to f without knowing f beforehand [1]. It is however possible to reduce the variance of the estimator by sampling according to an approximation of f. In path tracing, this is usually done locally during path generation. At each path vertex $\mathbf{x}_i$, the importance of the next path vertex $\mathbf{x}_{i+1}$ is proportional to the reflected radiance $L(\mathbf{x}_{i+1} \to \mathbf{x}_i) f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$. This leads to two importance sampling strategies:

[1] In chapter 6, the Metropolis-Hastings algorithm is used to generate a sequence of correlated samples proportional to f.

1. Try to locally estimate $L(\mathbf{x}_{i+1} \to \mathbf{x}_i)$ or $L(\mathbf{x}_{i+1} \to \mathbf{x}_i)G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$ and sample $\mathbf{x}_{i+1}$ accordingly

2. Sample $\mathbf{x}_{i+1}$ proportional to $f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$

The strategies are visualized in figure 4.2. The gray shape represents the local BSDF. In the left image, the outgoing direction is sampled proportional to the BSDF, while in the right image, the outgoing direction is sampled according to the estimated incoming radiance L. Both strategies are used during path tracing. Most radiance is expected to come directly

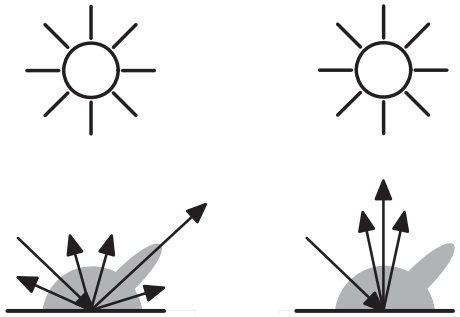

Figure 4.2: Importance sampling along to the local BSDF and towards the light source.

from light sources, so explicit connections are a form of importance sampling according to L. The second method of importance sampling is used during path extension, where the outgoing direction is usually chosen proportional to f_r. Note that these strategies are very rough, as explicit connections only account for direct light while extensions only account for local reflection.

Importance sampling is also combined with Russian roulette to terminate paths. During sampling, the termination probability at path vertex x_i is chosen proportional to the estimated fraction of incoming radiance that is absorbed. Usually, the absorption estimate is based on the local BSDF.

4.2 Path contribution

The contribution of a sampled path $\mathbf{X} = \mathbf{x}_0 \cdots \mathbf{x}_k$ to the Monte Carlo estimate equals $\frac{f(\mathbf{X})}{p(\mathbf{X})}$ (see section 3.1). The measurement contribution function $f(\mathbf{X})$ shares several factors with the sampling probability $p(\mathbf{X})$ for sampling explicit and implicit paths from last section, which will cancel out in the Monte Carlo contribution $\frac{f(\mathbf{X})}{p(\mathbf{X})}$.

After canceling out the common factors, the Monte Carlo contribution of an implicit path equals

$$\frac{f_j(\mathbf{x}_0 \cdots \mathbf{x}_k)}{p_I(\mathbf{x}_0 \cdots \mathbf{x}_k)} = \hat{W}_j(\mathbf{x}_1 \to \mathbf{x}_0) \prod_{i=1}^{k-1} \frac{f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})}{P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})} \cdot L_e(\mathbf{x}_k \to \mathbf{x}_{k-1}) \tag{4.3}$$

All geometric factors cancel out because they appear in both equations 4.2 and 2.7. Note that we used the modified sensor sensitivity function $\hat{W}_j$ from section 3.4.

Furthermore, after canceling out the common factors, the contribution for an explicit path becomes

$$\begin{aligned}\frac{f_j(\mathbf{x}_0 \cdots \mathbf{x}_k)}{p_E(\mathbf{x}_0 \cdots \mathbf{x}_k)} =& \hat{W}_j(\mathbf{x}_1 \to \mathbf{x}_0) \prod_{i=1}^{k-2} \frac{f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})}{P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})} \\ & \cdot f_r(\mathbf{x}_k \to \mathbf{x}_{k-1} \to \mathbf{x}_{k-2}) G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k-1}) \frac{L_e(\mathbf{x}_k \to \mathbf{x}_{k-1})}{P_M(\mathbf{x}_k)}\end{aligned} \tag{4.4}$$

Because the last vertex is not sampled using forward sampling but using surface area sampling, the geometric factor in f_j between the last two vertices does not cancel out and therefore appears in the final contribution.

Figures 4.3 and 4.4 show an implicit and explicit sampled light transport path, visualizing the factors in their contributions. The figures give an indication of the order in which these

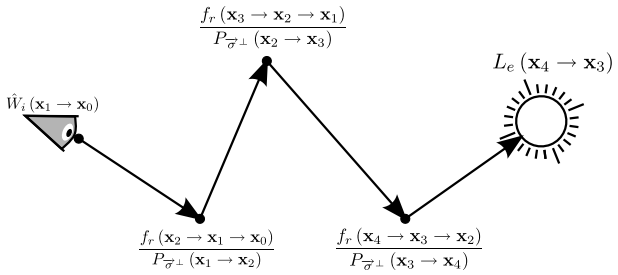

Figure 4.3: Monte Carlo contribution of implicit path.

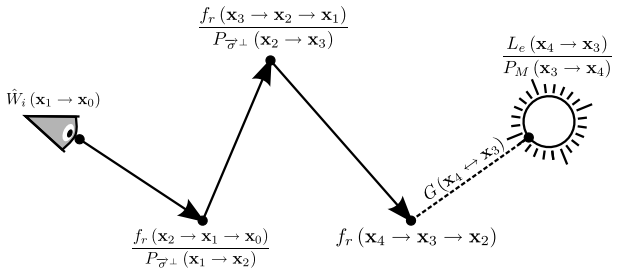

Figure 4.4: Monte Carlo contribution of explicit path.

contributions are computed during sample construction, starting with evaluating the sensor sensitivity function and iteratively extending the path, evaluating the BSDF and sample probabilities and finally evaluating the light source emittance.

4.3 MIS weights

As explained in section 3.3, when light transport paths may be sampled through multiple sampling strategies, the samples need to be combined using Multiple Importance Sampling as proposed by Veach [23]. For optimally combining implicit and explicit samples, we need to compute their importance weights according to the power heuristic. All but the last path vertex are sampled according to the same probability distributions for both explicit and

implicit paths. Therefore, when computing the weights for MIS, the common factors cancel out and only the probabilities for sampling the last path vertex using forward sampling and surface area sampling are used. For implicit paths, the power heuristic equals

$$\begin{aligned} w_I(\mathbf{x}_0 \cdots \mathbf{x}_k) &= \frac{p_I(\mathbf{x}_0 \cdots \mathbf{x}_k)^\beta}{p_E(\mathbf{x}_0 \cdots \mathbf{x}_k)^\beta + p_I(\mathbf{x}_0 \cdots \mathbf{x}_k)^\beta} \\ &= \frac{\left(P_{\vec{\sigma}^\perp}(\mathbf{x}_{k-1} \to \mathbf{x}_k) G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k-1})\right)^\beta}{P_M(\mathbf{x}_k)^\beta + \left(P_{\vec{\sigma}^\perp}(\mathbf{x}_{k-1} \to \mathbf{x}_k) G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k-1})\right)^\beta} \end{aligned} \tag{4.5}$$

For explicit paths, the power heuristic equals

$$\begin{aligned} w_E(\mathbf{x}_0 \cdots \mathbf{x}_k) &= \frac{p_E(\mathbf{x}_0 \cdots \mathbf{x}_k)^\beta}{p_E(\mathbf{x}_0 \cdots \mathbf{x}_k)^\beta + p_I(\mathbf{x}_0 \cdots \mathbf{x}_k)^\beta} \\ &= \frac{P_M(\mathbf{x}_k)^\beta}{P_M(\mathbf{x}_k)^\beta + \left(P_{\vec{\sigma}^\perp}(\mathbf{x}_{k-1} \to \mathbf{x}_k) G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k-1})\right)^\beta} \end{aligned} \tag{4.6}$$

4.4 Algorithm

Using the contributions and MIS weights from previous sections, we can construct a path tracing sample and compute its total contribution. Algorithm 1 gives the pseudo code for generating a PT sample. The algorithm returns the combined contribution of all explicit and implicit paths in the generated sample. The algorithm starts by generating vertex $\mathbf{y}_0$ on the lens. Then, $\mathbf{y}_1$ is sampled by tracing a ray through the view plane according to the sampling probability $P_S(\mathbf{y}_1)$ per unit screen space area (see section 3.4). If the scene is not closed, the ray may miss all geometry, failing to generate vertex $\mathbf{y}_1$. In this case, the path immediately terminates. Otherwise, the sample is extended iteratively.

During extension, it is first checked to see if the path so far is a valid implicit path. If so, the corresponding importance weight and contribution are computed and added to the total sample contribution. In the computation of the importance weight, $P_M(\mathbf{y}_i)$ is the probability with which $\mathbf{y}_i$ would have been sampled as the light vertex if this path was constructed as an explicit path instead. Note that paths of length 1 can only be sampled using implicit paths and thus their importance weight equals 1.

Then, an explicit connection is made by generating a vertex $\mathbf{z}$ on the light source. If the current vertex $\mathbf{y}_i$ and the light vertex $\mathbf{z}$ are not visible from one another, the connection failed. Otherwise, the importance and contribution of the explicit path are computed and added to the total sample contribution. In the computation of the importance weight, $P_{\vec{\sigma}^\perp}(\mathbf{y}_{i-1} \to \mathbf{y}_i \to \mathbf{z})$ is the probability with which $\mathbf{z}$ would have been sampled as the next path vertex if this path was constructed as an implicit path instead.

Finally, the next path vertex is sampled through forward sampling. This again requires ray tracing. The path may be terminated for two reasons; either it is terminated through Russian roulette or the generated output ray misses all geometry, effectively terminating the path. If the path is not terminated, the path is extended by another vertex during the next iteration. Note that the Russian roulette probability is assumed to be incorporated in the sampling probability $P_{\vec{\sigma}^\perp}(\mathbf{y}_i \to \mathbf{y}_{i+1})$. Hence, with non-zero probability, no next vertex may be generated. When the path has terminated, the total sample contribution is returned.

Algorithm 1 : PT()

$color \leftarrow$ black

sample $\mathbf{y}_0 \sim P_I(\mathbf{y}_0)$
sample $\mathbf{y}_1 \sim P_S(\mathbf{y}_1)$

$f_X \leftarrow \hat{W}(\mathbf{y}_1 \rightarrow \mathbf{y}_0)$

$i \leftarrow 0$
while path is not terminated **do**
  $i \leftarrow i + 1$

  {*Check implicit path*}
  if $L_e(\mathbf{y}_i \rightarrow \mathbf{y}_{i-1}) > 0$ **then**
    if $i > 1$ **then**
      $P_M(\mathbf{y}_{i-1} \rightarrow \mathbf{y}_i) \leftarrow P_{\vec{\sigma}^\perp}(\mathbf{y}_{i-1} \rightarrow \mathbf{y}_i) G(\mathbf{y}_{i-1} \leftrightarrow \mathbf{y}_i)$
      $w_I \leftarrow \frac{P_M(\mathbf{y}_{i-1} \rightarrow \mathbf{y}_i)^\beta}{P_M(\mathbf{y}_i)^\beta + P_M(\mathbf{y}_{i-1} \rightarrow \mathbf{y}_i)^\beta}$
    else
      $w_I \leftarrow 1$
    end if
    $color \leftarrow color + w_I f_X L_e(\mathbf{y}_i \rightarrow \mathbf{y}_{i-1})$
  end if

  {*Make explicit connection*}
  sample $\mathbf{z} \sim P_M(\mathbf{z})$
  if $V(\mathbf{y}_i \leftrightarrow \mathbf{z}) = 0$ **then**
    $w_E \leftarrow \frac{P_M(\mathbf{z})^\beta}{P_M(\mathbf{z})^\beta + \left(P_{\vec{\sigma}^\perp}(\mathbf{y}_{i-1} \rightarrow \mathbf{y}_i \rightarrow \mathbf{z}) G(\mathbf{y}_i \leftrightarrow \mathbf{z})\right)^\beta}$
    $color \leftarrow color + w_E f_X f_r(\mathbf{z} \rightarrow \mathbf{y}_i \rightarrow \mathbf{y}_{i-1}) G(\mathbf{z} \leftrightarrow \mathbf{y}_i) \frac{L_e(\mathbf{z} \rightarrow \mathbf{y}_i)}{P_M(\mathbf{z})}$
  end if

  {*Extend path*}
  sample $\mathbf{y}_{i+1} \sim P_{\vec{\sigma}^\perp}(\mathbf{y}_i \rightarrow \mathbf{y}_{i+1})$
  $f_X \leftarrow f_X \cdot \frac{f_r(\mathbf{y}_{i+1} \rightarrow \mathbf{y}_i \rightarrow \mathbf{y}_{i-1})}{P_{\vec{\sigma}^\perp}(\mathbf{y}_i \rightarrow \mathbf{y}_{i+1})}$
end while

return *color*

4.5 Photon Map Importance Sampling

Remember from section 4.1 that during extension at some path vertex, the next path vertex $\mathbf{x}_{i+1}$ should optimally be sampled proportional to $L(\mathbf{x}_{i+1} \to \mathbf{x}_i) f_r(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1}) G(\mathbf{x}_{i+1} \leftrightarrow \mathbf{x}_i)$. Hence, the projected outgoing direction $\vec{w}_o$ should be sampled proportional to $L(\mathbf{x}, \vec{w}_o) f_r(\mathbf{x}, \vec{w}_o, \vec{w}_i)$. So far, we only used a local approximation based on the BRDF. Using such an approximation does not account for $L(\mathbf{x}, \vec{w}_o)$ and therefore it often is very inaccurate.

Jensen proposed a method to approximate the incoming radiance $L(\mathbf{x}, \vec{w}_o)$ with a photon map and use this to guide importance sampling during extension [9]. Initially, many photons are traced, starting from the light source and bouncing through the scene. Each time a photon interacts with a surface, the photon is stored in a photon map. During path tracing, this photon map is used to compute an approximation $\hat{L}(\mathbf{x}, \vec{w}_o)$ of the incoming radiance at $\mathbf{x}$. Using this approximation, the outgoing extension direction $\vec{w}_o$ can now be sampled proportional to $\hat{L}(\mathbf{x}, \vec{w}_o) f_r(\mathbf{x}, \vec{w}_o, \vec{w}_i)$

Before extending the path, the unit hemisphere around $\mathbf{x}$ is mapped on a unit square and discretized. Then, the N nearest photons are located in the photon map. These photons are used to compute $\hat{L}(\mathbf{x}, \vec{w}_o)$. Each photon represents an amount of incoming radiance along some incoming direction $\vec{v}_i$. Therefore, for each photon, the corresponding reflected radiance $L(x, \vec{v}_i) f_r(\mathbf{x}, \vec{v}_i, \vec{w}_i)$ is computed and accumulated in the corresponding region on the unit square. Finally, a point on the unit square is sampled proportional to the accumulated radiance. The corresponding outgoing direction is found by an inverse mapping from the unit square to the unit hemisphere.

This method samples outgoing directions roughly proportional to reflected incoming radiance. Note however that the additional sampling cost is relatively high. A photon map needs to be generated, the N nearest photons need to be found and their contributions need to be computed. For this method to be efficient, the evaluation of the BRDF is assumed to be relatively cheap, as it must be evaluated for all N photons. Furthermore, the incoming radiance $L(\mathbf{x}, \vec{w}_o)$ is assumed to vary significantly. If not, sampling proportional to $f_r(\mathbf{x}, \vec{w}_o, \vec{w}_i)$ would suffice.

4.6 Importance Resampling

When applying importance sampling to path tracing, one often needs to sample proportional to some known function f. For example, during extension, the outgoing direction is preferably sampled proportional to the BRDF f_r. Furthermore, during an explicit connection, the light vertex x is preferable sampled proportional to $f_r(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') G(\mathbf{x} \leftrightarrow \mathbf{x}') L_e(\mathbf{x} \to \mathbf{x}')$. Although most of these functions are known in advance, it often remains hard to generate samples proportional to them. Therefore, instead of sampling proportional to f, samples are generated proportional to a more convenient approximation $\hat{f} \approx f$. However, using an approximation usually increases the overall variance in the estimator; The further off the approximation, the worse the variance.

Sherly proposed to reduce the impact of using an approximation through importance resampling. He applied this to the computation of direct lighting [19]. Talbot further generalized the method and applied it to path extension direction sampling [21, 20].

Instead of generating a single sample X proportional to f, N candidate samples $X_1 \cdots X_N$ are generated, proportional to the approximation $\hat{f}$. To correct for the use of an approxima-

tion, a final sample is selected (resampled) from the candidate samples. All candidate samples are weighted with $w_i = \frac{f(X_i)}{\hat{f}(X_i)}$. If the weight of a candidate sample is high, it indicates that the corresponding value is undersampled. If the weight is low, the corresponding value is oversampled. To correct for this under/oversampling, the final sample is resampled from the candidate samples, proportional to their weights.

For $N = 1$, the method deteriorates to importance sampling proportional to $\hat{f}$. As $N \to \infty$, the method samples more proportional to the intended function f, at the cost of requiring the generation of more candidate samples. The value N can be used to trade between sampling quality and sampling cost.

This method is easily applied to path tracing. Whenever it is impossible to sample outgoing directions proportional to f_r, importance resampling can be combined with a more convenient approximation to improve the sampling quality. Similar, when generating light vertices for explicit connections, importance resampling can be used to improve the quality of the samples, sampling roughly proportional to both the BRDF f_r and the light source emittance L_e.

4.7 Path Re-Using

The PT algorithm described so far generates a new independent path for each pixel. Because generating a single path is costly, it can be worthwhile to reuse part of the path to cheaply generate new, correlated samples. Bekaert proposed to combine paths contributing to different pixels to cheaply increase the number of samples per pixel [1]. First, N independent paths $X_1 \cdots X_N$ are generated, with $X_i = \mathbf{x}_0^i \mathbf{x}_1^i \mathbf{x}_2^i \cdots$, each path contributing to a different pixel. Then, these paths are reused by connecting the primary ray $\mathbf{x}_0^i \mathbf{x}_1^i$ of each path i to the remainder $\mathbf{x}_2^j \cdots$ of the other $N-1$ paths $j \neq i$ to produce the paths $\mathbf{x}_0^i \mathbf{x}_1^i \mathbf{x}_2^j \cdots$ (see figure 4.5). The number of samples per pixel is increased N'fold at the cost of only a single extra

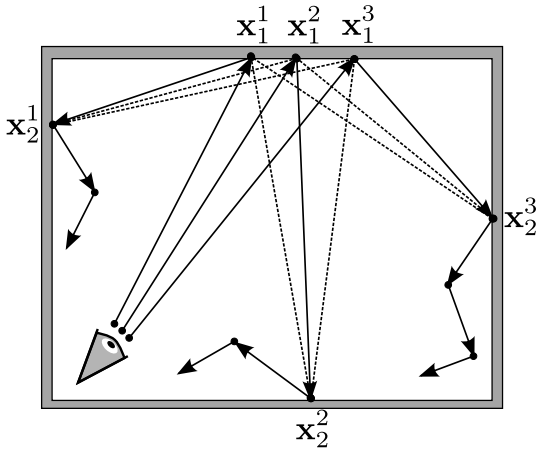

Figure 4.5: Path Re-Use

shadow ray per newly generated sample. Because the reused samples share part of their paths, this method increases the correlation between image pixels. By using slightly interspaced pixels for a single group, visual artifacts due to correlation are minimized. The value N can be used to trade correlation for efficiency.

Chapter 5

Bidirectional Path Tracing

In this chapter, we discuss the BiDirectional Path Tracing algorithm, independently developed by both Veach and Lafortune [22, 12]. In sections 5.1, 5.2 and 5.4, we describe the BDPT algorithm in detail. In section 5.3, we present a novel computation schema for constructing the balance heuristic weights for BDPT samples. In section 5.5, we discuss the work of Lafortune, who extended the BDPT algorithm for rendering participating media [13].

5.1 Sampling

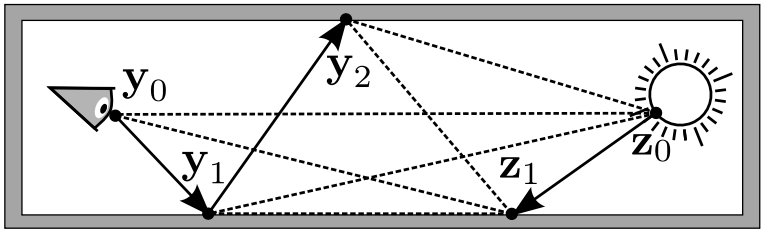

Figure 5.1: BiDirectional Path Tracing Sample generates light transport paths by connecting an eye path an a light path

In the PT sampler from last chapter, all but the last path vertex are sampled by tracing a path forward from the eye into the scene. This is not always the most effective sampling strategy. In scenes with mostly indirect light, it is often hard to find valid paths by sampling backwards from the eye. Sampling a part of the path backward, starting at a light source and tracing backward into the scene, can solve this problem.

This is exactly what the BiDirectional Path Tracing (BDPT) sampler does. BDPT was independently developed by Veach and Lafortune [22, 12]. It samples an eye path and a light path and connects these to form complete light transport paths. The eye path starts at the eye and is traced forward into the scene, like in the PT sampler. The light path starts at a light source and is traced backward into the scene. Connecting the endpoints of any eye and light path using an explicit connection results in a complete light transport path from light source to eye.

Like the PT sampler, for efficiency, instead of generating a single path per sample, the BDPT sampler generates a collection of correlated paths per sample. Figure 5.1 shows a

complete BDPT sample. When constructing a sample, an eye path and a light path are sampled independently, where after all the vertices of the eye path are explicitly connected to all the vertices of the light path to construct valid light transport paths. Each connection results in a light transport path.

Let $\mathbf{X} \in \Omega$ be a light transport path of length k with vertices $\mathbf{x}_0 \cdots \mathbf{x}_k$, that is part of a BDPT sample. This path can be constructed using one of $k+1$ different bidirectional sampling strategies by connecting an eye path $\mathbf{Y}^s = \mathbf{y}_0 \cdots \mathbf{y}_s$ of length $s \geq 0$ with a light path $\mathbf{Z}^t = \mathbf{z}_1 \dots \mathbf{z}_t$ of length $t \geq 0$, where $k = s + t$ ($\mathbf{y}_i = \mathbf{x}_i$ and $\mathbf{z}_i = \mathbf{x}_{k-i+1}$). Note that both the eye and light path may be of length zero. In case of a zero length light path, the eye path is an implicit path and should end at a light source. In case of an eye path of length 0, the light path is directly connected to the eye. We will not deal with light paths directly hitting the eye[1].

The probability of bidirectionally generating a complete path $\mathbf{X}$ by connecting an eye path of length s with a light path of length $t = k - s$ equals $p_s(\mathbf{X})$. Each bidirectional sampling strategy represents a different importance sampling strategy and has a different probability distribution over path space. Hence, by combining all sampled paths using MIS, the strengths of these different sampling strategies are combined in a single unbiased estimator. By incorporating the termination probabilities into the path sampling probability $p_s(\mathbf{X})$ (similar to the PT sampler), the number of samples per strategy n_i becomes 1. Applying the power heuristic to the BDPT sampler results in a MIS weight function of

$$w_s(\mathbf{X}) = \frac{p_s(\mathbf{X})^\beta}{\sum_{i=0}^{k} p_i(\mathbf{X})^\beta} \tag{5.1}$$

Now let us turn to the computation of the probability $p_i(\mathbf{X})$ for any $0 \leq i \leq k$. This probability equals the probability of sampling all the individual vertices on the path per unit area. Each vertex is either sampled as part of the eye path or the light path. In this case, the first i vertices are sampled as part of the eye path $\mathbf{Y}^s$ ($s = i$). The remaining $t = k - s$ vertices are sampled as part of the light path $\mathbf{Z}^t$. The case with $s = k$ corresponds to implicit paths as sampled by a path tracer, hence their probability remains

$$\begin{aligned} p_k(\mathbf{x}_0 \cdots \mathbf{x}_k) =& \frac{l^2 \vec{n}_1 \cdot (\mathbf{x}_1 - \mathbf{x}_0)}{\rho \left(\vec{n}_0 \cdot (\mathbf{x}_1 - \mathbf{x}_0)\right)^3} \prod_{i=2}^{k} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1}) \cdot P_I(\mathbf{x}_0) P_S(\mathbf{x}_1) \prod_{i=2}^{k} P_{\vec{\sigma}^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i) \\ =& P_I(\mathbf{x}_0) \prod_{i=1}^{k} P_M(\mathbf{x}_i) \end{aligned} \tag{5.2}$$

Similar, for the case where all but vertex $\mathbf{x}_0$ is sampled backward from the light($s = 0$), the probability becomes

$$\begin{aligned} p_0(\mathbf{x}_0 \cdots \mathbf{x}_k) =& \prod_{i=1}^{k-1} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \cdot P_I(\mathbf{x}_0) \prod_{i=1}^{k-1} P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) P_M(\mathbf{x}_k) \\ =& P_I(\mathbf{x}_0) \prod_{i=1}^{k} P_M(\mathbf{x}_i) \end{aligned} \tag{5.3}$$

[1] A light path can only directly hit the eye when a camera model with finite aperture lens is used. The contribution of such paths is usually insignificant.

Note the absence of the extra conversion term to convert $P_S(\mathbf{x}_1)$ from unit screen space to unit surface area because $\mathbf{x}_1$ is sampled as part of the light path. Finally, Using equation 3.13 from section 3.4 for converting bidirectional sampling probabilities to unit surface area, the sampling probability for all remaining paths becomes

$$\begin{aligned} p_s(\mathbf{x}_0 \cdots \mathbf{x}_k) =& \frac{l^2 \vec{n}_1 \cdot (\mathbf{x}_1 - \mathbf{x}_0)}{\rho(\vec{n}_0 \cdot (\mathbf{x}_1 - \mathbf{x}_0))^3} \prod_{i=2}^{s} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1}) \prod_{i=s+1}^{k-1} G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \cdot \\ & P_I(\mathbf{x}_0) P_S(\mathbf{x}_1) \prod_{i=2}^{s} P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_{i-1} \rightarrow \mathbf{x}_i) \prod_{i=s+1}^{k-1} P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) P_M(\mathbf{x}_k) \qquad (5.4) \\ =& P_I(\mathbf{x}_0) \prod_{i=1}^{k} P_M(\mathbf{x}_i) \end{aligned}$$

BDPT requires many correlated connection rays to be traced. It is possible to reduce the number of connection rays by applying Russian roulette to the connections. Instead of making all connections, each connection is made with a certain probability, based on the contribution this connection would make to the estimator if it would succeed. By correcting for this probability, the estimate remains unbiased. Note that this optimization is also a form of importance sampling; Connections are sampled according to their estimated contribution. The estimate is based on the assumption that the connection will actually succeed.

5.2 Path contribution

The contribution of a sampled path $\mathbf{X} = \mathbf{x}_0 \cdots \mathbf{x}_k$ to the Monte Carlo estimate equals $\frac{f(\mathbf{X})}{p(\mathbf{X})}$ (see section 3.1). Just like paths sampled through path tracing, the measurement contribution function $f(\mathbf{X})$ shares several factors with the probability $p(\mathbf{X})$ for bidirectionally sampling paths per unit path space, which again cancel out in the Monte Carlo contribution $\frac{f(\mathbf{X})}{p(\mathbf{X})}$.

In the last section, we saw that a path $\mathbf{X} = \mathbf{x}_0 \cdots \mathbf{x}_k$ of length k can be sampled using either one of $k+1$ sampling strategies. Furthermore, we defined $p_s(\mathbf{X})$ as the probability of bidirectionally sampling $\mathbf{X}$ by connecting the eye path $\mathbf{x}_0 \cdots \mathbf{x}_s$ of length[2] s with the light path $\mathbf{x}_{s+1} \cdots \mathbf{x}_k$ of length $t = k - s$. Paths sampled using light paths of length $t = 0$ and $t = 1$ correspond to implicit and explicit paths as sampled by a path tracer. We already showed how common factors cancel out in the contribution of implicit and explicit paths in section 4.2. What is left are the contributions for paths sampled using any of the remaining bidirectional sampling strategies.

For paths sampled with a zero length eye path ($s = 0$), all vertices except $\mathbf{x}_0$ are sampled as part of the light path. After combining equations 5.3, 3.18 and 2.7 and canceling out the

[2]Note that because $\mathbf{x}_0$ is also part of the eye path, an eye path of length s contains $s + 1$ vertices.

common factors, the Monte Carlo contribution of these paths equals

$$\begin{aligned}\frac{f_j\left(\mathbf{x}_0 \cdots \mathbf{x}_k\right)}{p_0\left(\mathbf{x}_0 \cdots \mathbf{x}_k\right)} =& \hat{W}_j\left(\mathbf{x}_1 \rightarrow \mathbf{x}_0\right) \frac{l^2 \vec{n}_1 \cdot\left(\mathbf{x}_0-\mathbf{x}_1\right)}{\rho\left(\vec{n}_0 \cdot\left(\mathbf{x}_1-\mathbf{x}_0\right)\right)^3} P_S\left(\mathbf{x}_1\right) \\ & f_r\left(\mathbf{x}_2 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_0\right) \\ & \prod_{j=2}^{k-1} \frac{f_r\left(\mathbf{x}_{j+1} \rightarrow \mathbf{x}_j \rightarrow \mathbf{x}_{j-1}\right)}{P_{\overleftarrow{\sigma}^{\perp}}\left(\mathbf{x}_{j-1} \leftarrow \mathbf{x}_j\right)} \\ & \frac{L_e\left(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1}\right)}{P_M\left(\mathbf{x}_k\right) P_{\overleftarrow{\sigma}^{\perp}}\left(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k\right)}\end{aligned} \tag{5.5}$$

Note that we again used the modified sensor sensitivity function $\hat{W}_j$ 3.4. For all other bidirectional strategies, using $\hat{W}_j$ removes the conversion factor from unit screen space to unit surface area resulting from sampling $\mathbf{x}_1$ per unit screen space. However, when $s = 0$, $\mathbf{x}_1$ is sampled as part of the light path and is therefore not sampled per unit screen space. Hence, no conversion is required. Consequently, using $\hat{W}_j$ introduces an extra correction factor $\frac{l^2 \vec{n}_1 \cdot\left(\mathbf{x}_0-\mathbf{x}_1\right)}{\rho\left(\vec{n}_0 \cdot\left(\mathbf{x}_1-\mathbf{x}_0\right)\right)^3}$.

Finally for the remaining sampling strategies with $0 < s < k - 1$, after canceling out the common factors, the Monte Carlo contribution of a path equals

$$\begin{aligned}\frac{f_j\left(\mathbf{x}_0 \cdots \mathbf{x}_k\right)}{p_s\left(\mathbf{x}_0 \cdots \mathbf{x}_k\right)} =& \hat{W}_j\left(\mathbf{x}_1 \rightarrow \mathbf{x}_0\right) \\ & \prod_{j=1}^{s-1} \frac{f_r\left(\mathbf{x}_{j+1} \rightarrow \mathbf{x}_j \rightarrow \mathbf{x}_{j-1}\right)}{P_{\overrightarrow{\sigma}^{\perp}}\left(\mathbf{x}_j \rightarrow \mathbf{x}_{j+1}\right)} \\ & f_r\left(\mathbf{x}_{s+1} \rightarrow \mathbf{x}_s \rightarrow \mathbf{x}_{s-1}\right) G\left(\mathbf{x}_s \leftrightarrow \mathbf{x}_{s+1}\right) f_r\left(\mathbf{x}_{s+2} \rightarrow \mathbf{x}_{s+1} \rightarrow \mathbf{x}_s\right) \\ & \prod_{j=s+2}^{k-1} \frac{f_r\left(\mathbf{x}_{j+1} \rightarrow \mathbf{x}_j \rightarrow \mathbf{x}_{j-1}\right)}{P_{\overleftarrow{\sigma}^{\perp}}\left(\mathbf{x}_{j-1} \leftarrow \mathbf{x}_j\right)} \\ & \frac{L_e\left(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1}\right)}{P_M\left(\mathbf{x}_k\right) P_{\overleftarrow{\sigma}^{\perp}}\left(\mathbf{x}_{k-1} \leftarrow \mathbf{x}_k\right)}\end{aligned} \tag{5.6}$$

All geometric factors on the eye and light path cancel out because they appear in both equation 5.4 and 2.7. The only remaining geometric factor corresponds to the connection edge between $\mathbf{x}_s$ and $\mathbf{x}_{s+1}$.

Figures 5.2 and 5.3 show light transport paths sampled with bidirectional sampling strategies corresponding to $s = 0$ and $s = 2$. The figures visualize the factors in these paths' Monte Carlo contributions. The figures also give an indication of the order in which these contributions are computed during sample construction. For the eye path, construction starts with evaluating the sensor sensitivity function and iteratively extending the eye path, evaluating the BSDF and sample probabilities. For the light path, construction starts with evaluating the light source emittance and iteratively extending the light path, evaluating the BSDF and sample probabilities. Finally, the eye and light path are explicitly connected and the connection is evaluated.

5.3 MIS

Besides computing the contribution for each bidirectional path, a MIS weight needs to be computed. Veach proposed a method for efficiently constructing balance heuristic weights. Their

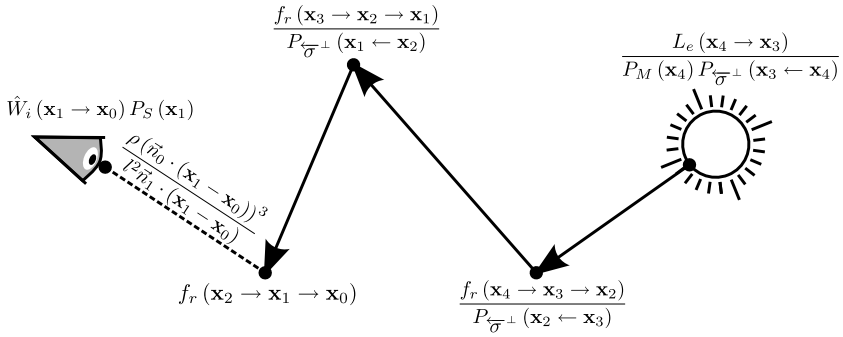

Figure 5.2: Monte Carlo contribution of path sampled by connecting a light path directly to the eye.

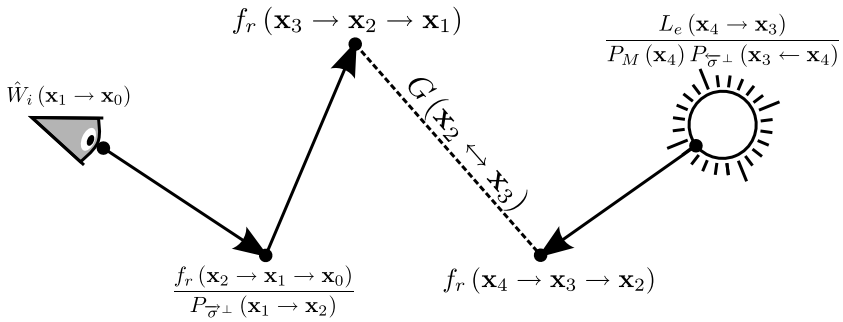

Figure 5.3: Monte Carlo contribution of bidirectional sampled path.

method computes the ratio $\frac{p_{i+1}(\mathbf{X})}{p_i(\mathbf{X})}$ for all $i = 0 \cdots k-1$. Then, these ratios are repeatedly applied to $p_s(\mathbf{X})$ to find the remaining probabilities $p_{s+1}(\mathbf{X}) \cdots p_k(\mathbf{X})$ and $p_{s-1}(\mathbf{X}) \cdots p_0(\mathbf{X})$. The balance heuristic is then computed by combining these probabilities. Although this computation schema is very generic and fairly efficient in terms of computations, the number of operations to construct the weight depends on the path length, requiring data from all vertices on the path. This prevents efficient parallelization of both computations and memory access. We propose a different schema for computing the power heuristic weights. During the construction of the eye and light paths, we recursively compute one extra quantity in each path vertex. Using this quantity, the weights for each connection can be constructed locally, requiring only data from the eye and light vertex directly involved in the connection.

In the remainder of this section, we use the following abbreviations for the forward and backward sampling probabilities of individual path vertices

$$\overleftarrow{p}_i = P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i) G(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)$$

$$\overrightarrow{p}_i = P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$$

$$\overleftarrow{p}_{k+1} = P(\mathbf{x}_k)$$

Thus, the bidirectional sampling probability 3.13 becomes

$$p_s(\mathbf{x}_0 \cdots \mathbf{x}_k) = P_I(\mathbf{x}_0) \prod_{i=0}^{s-1} \overrightarrow{p}_i \prod_{i=s+2}^{k+1} \overleftarrow{p}_i$$

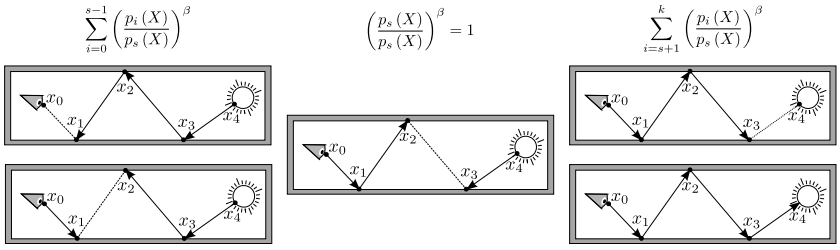

Figure 5.4: The power heuristic weights can be split up in three parts.

Now, instead of directly computing the weights for each connection, it is easier to compute the inverse weights

$$\frac{1}{w_s(\mathbf{X})} = \sum_{i=0}^{k} \frac{p_i(\mathbf{X})}{p_s(\mathbf{X})}$$

Based on common factors in the terms, this equation is split into three parts, as shown in figure 5.4.

$$\frac{1}{w_s(\mathbf{X})} = \sum_{i=0}^{s-1} \frac{p_i(\mathbf{X})}{p_s(\mathbf{X})} + 1 + \sum_{i=s+1}^{k} \frac{p_i(\mathbf{X})}{p_s(\mathbf{X})} \tag{5.7}$$

The first term in equation 5.7 represents all sampling strategies with less than s eye vertices. These strategies all sample at least the vertices $\mathbf{x}_k \cdots \mathbf{x}_s$ as part of the light path. Hence, the probability of sampling these vertices is a common factor in both numerator and denominator and cancel out. Furthermore, all forward sampling probabilities appearing in the numerator also appear in the denominator and thus cancel out as well, resulting in

$$\sum_{i=0}^{s-1} \frac{p_i(\mathbf{X})}{p_s(\mathbf{X})} = \sum_{i=0}^{s-1} \frac{\prod_{j=i+2}^{s+1} \overleftarrow{p}_j}{\prod_{j=i}^{s-1} \overrightarrow{p}_j} = \overleftarrow{p}_{s+1} \sum_{i=0}^{s-1} \left(\frac{1}{\overrightarrow{p}_i} \prod_{j=i+1}^{s-1} \frac{\overleftarrow{p}_{j+1}}{\overrightarrow{p}_j} \right)$$

By expending the last term in this summation, the sought after recursion appears

$$\sum_{i=0}^{s-1} \frac{p_i(\mathbf{X})}{p_s(\mathbf{X})} = \overleftarrow{p}_{s+1} \left(\frac{1}{\overrightarrow{p}_{s-1}} + \frac{\overleftarrow{p}_s}{\overrightarrow{p}_{s-1}} \sum_{i=0}^{s-2} \left(\frac{1}{\overrightarrow{p}_i} \prod_{j=i+1}^{s-2} \frac{\overleftarrow{p}_{j+1}}{\overrightarrow{p}_j} \right) \right)$$

We define the quantity d_s^E as

$$d_s^E = \sum_{i=0}^{s-1} \frac{1}{\overrightarrow{p}_i} \prod_{j=i+1}^{s-1} \frac{\overleftarrow{p}_{j+1}}{\overrightarrow{p}_j} = \frac{1 + \overleftarrow{p}_s d_{s-1}^E}{\overrightarrow{p}_{s-1}}$$

The base case for d_s^E depends on the particular camera model used. During the construction of the eye path, this quantity is recursively computed for each eye vertex. Using this quantity, the first term in equation 5.7 reduces to

$$\sum_{i=0}^{s-1} \frac{p_i(\mathbf{X})}{p_s(\mathbf{X})} = d_s^E \overleftarrow{p}_{s+1} \tag{5.8}$$

Hence, by using the precomputed quantity d_s^E when making a connection, the first term can be computed locally, requiring only data from the eye vertex involved in the connection and a fixed amount of operations, independent of the path lengths.

The second term in equation 5.7 corresponds to the strategy with exactly s eye vertices and is the strategy actually used to sample this path. Because both numerator and denominator are the same, this term reduces to 1.

Finally, the last term in equation 5.7 is similar to the first term and represents all sampling strategies with more than s eye vertices. After eliminating all common factors, this sum become

$$\sum_{i=s+1}^{k} \frac{p_i(\mathbf{X})}{p_s(\mathbf{X})} = \overrightarrow{p}_s \sum_{i=s+1}^{k} \left(\frac{1}{\overleftarrow{p}_{i+1}} \prod_{j=s+1}^{i-1} \frac{\overrightarrow{p}_j}{\overleftarrow{p}_{j+1}} \right)$$

The corresponding quantity d_{s+1}^L is defined as

$$d_{s+1}^L = \sum_{i=s+1}^{k} \left(\frac{1}{\overleftarrow{p}_{i+1}} \prod_{j=s+1}^{i-1} \frac{\overrightarrow{p}_j}{\overleftarrow{p}_{j+1}} \right) = \frac{1 + d_{s+2}^L \overrightarrow{p}_{s+1}}{\overleftarrow{p}_{s+2}}$$

Again, the base case for d_{s+1}^L depends on the particular emission model used. During the construction of the light path, this quantity is recursively computed for each light vertex. Hence, the last term in equation 5.7 reduces to

$$\sum_{i=s+1}^{k} \frac{p_i(\mathbf{X})}{p_s(\mathbf{X})} = d_{s+1}^L \overrightarrow{p}_s \tag{5.9}$$

By subsituting equations 5.8 and 5.9 into equation 5.7, the equation reduces to

$$\frac{1}{w_s(\mathbf{X})} = \begin{cases} 1 + d_1^L \overrightarrow{p}_0 & \text{if } s = 0 \\ d_k^E \overleftarrow{p}_{k+1} + 1 & \text{if } s = k \\ d_s^E \overleftarrow{p}_{s+1} + 1 + d_{s+1}^L \overrightarrow{p}_s & \text{otherwise} \end{cases}$$

Note that the value $\overleftarrow{p}_s$ in the quantity d_s^E depends on both the eye an light vertex involved in the connection. Because the light vertex is not known during the construction of the eye path, d_s^E can only be computed from d_{s-1}^E when the actual connection is made. The same holds for d_{s+1}^L.

Finally, a word on computing the probabilities $\overleftarrow{p}_i$ and $\overrightarrow{p}_i$. Due to the geometric factor in these probabilities, two path vertices are required to compute these. However, by carefully splitting up these computations, data from only a single vertex is required for computing each part. For example, $\overleftarrow{p}_i$ can be split into

$$\overleftarrow{p}'_i = |\vec{\omega}_{i-1} \cdot \vec{n}_{i-1}|$$

$$\overleftarrow{p}_i = \overleftarrow{p}'_i P(\mathbf{x}_{i-1} \leftarrow \mathbf{x}_i \leftarrow \mathbf{x}_{i+1}) \frac{|\vec{\omega}_{i-1} \cdot \vec{n}_i|}{||\mathbf{x}_i - \mathbf{x}_{i-1}||^2}$$

With $\vec{\omega}_i$ the outgoing extension direction at vertex $\mathbf{x}_i$. During eye path construction, $\overleftarrow{p}'_i$ is computed in vertex $\mathbf{x}_{i-1}$ and $\overleftarrow{p}_i$ is computed from $\overleftarrow{p}'_i$ in vertex $\mathbf{x}_i$. The computation of $\overrightarrow{p}_i$ during light path construction can be split up in a similar way.

Remember that the balance heuristic is not the only valid MIS weight schema. A recursive computation scheme for the more generic power heuristic can easily be obtained by raising all probabilities $\overleftarrow{p}$ and $\overrightarrow{p}$ to the power of β. Similar, the maximum heuristic can be obtained from the balance heuristic by substituting all addition operators by maximum operators.

Path termination

In the recursive formulation of the eye path quantity $d_i^{\mathbf{E}}$, we need to evaluate the reverse sampling probability $\overleftarrow{p}_i$. At the time of evaluation, no information about the light path is available. Therefore, this evaluation must be independent of the light path. Because of these restrictions, any sampling decisions at a vertex may only depend on local information such as the incoming and outgoing direction and the surface properties. However, it may not depend on global path information, such as the vertex index on the path, because the light path length is not known while evaluating these reverse probabilities. Note that the probability to terminate the path at a vertex during construction is part of its sampling probability. Usually, Russian roulette is used to decide termination. Because of the former restriction, the Russian roulette method may also only depend on local vertex information and not on the global vertex index in the path. The same holds for computing $\overrightarrow{p}_i$ on the light path.

5.4 Algorithm

Using the contributions from the previous section and the recursive MIS weights from section 5.3, we can now construct a BDPT sample. The pseudo code for generating such a sample is given by algorithms 2, 3 and 4.

Algorithm 2 generates an eye path $\mathbf{y}_0 \cdots \mathbf{y}_{N_E}$ and is somewhat similar to the path tracing algorithm (see algorithm 1), but without the evaluation of explicit and implicit paths. Instead, the recursive MIS quantities for the path vertices are computed. The algorithm samples the first two vertices according to the camera model and iteratively extends the path with extra vertices until the path terminates. As with the path tracing algorithm from chapter 4, the eye path is terminated either because the extension ray misses all scene geometry or due to Russian roulette. Again, the Russian roulette probability is assumed to be incorporated in the forward sampling probability. Note the special case which occurs while extending vertex $\mathbf{x}_1$. This special case allows us to use the modified sensor sensitivity function $\hat{W}_j$ and relieves us from evaluating $P_{\overleftarrow{\sigma}^\perp}(\mathbf{y}_0 \leftarrow \mathbf{y}_1)^\beta$.

Algorithm 3 looks a lot like algorithm 2 and generates a light path $\mathbf{z}_1 \cdots \mathbf{z}_{N_L}$. The main difference is that the first vertex $\mathbf{z}_1$ is sampled on the light source instead of on the lens and that the second vertex $\mathbf{z}_2$ is sampled using backward sampling from the light source. No special case is required when extending $\mathbf{z}_2$.

After both the eye and light path are sampled, algorithm 4 evaluates all connections and computes the total sample contribution. The algorithm iterates over all $N_E + 1$ eye vertices, checking for implicit paths. If an implicit path is encountered, the path contribution and importance are computed and added to the image along the first path edge. For each eye vertex, connections are made to all N_L light vertices on the light path. For each pair, a connection is established. If the vertices are not visible from one another, the connection failed. Otherwise, the importance and contribution of the path are computed and added to the image along the first path edge. Note that special care must be taken when the vertices $\mathbf{y}_0$ and $\mathbf{z}_1$ are involved in the connection. Also, instead of accumulating the total sample

Algorithm 2 : SampleEyePath

sample $\mathbf{y}_0 \sim P_I(\mathbf{y}_0)$
sample $\mathbf{y}_1 \sim P_S(\mathbf{y}_1)$

$F_0^Y \leftarrow \hat{W}(\mathbf{y}_1 \rightarrow \mathbf{y}_0)$
$D_0^Y \leftarrow 0$
$P_0^Y \leftarrow \left(\frac{\rho(\vec{n}_0 \cdot (\mathbf{y}_1 - \mathbf{y}_0))^3}{l^2 \vec{n}_1 \cdot (\mathbf{y}_1 - \mathbf{y}_0)} P_S(\mathbf{y}_1)\right)^\beta$

$i \leftarrow 0$
while path is not terminated **do**
  $i \leftarrow i + 1$
  sample $\mathbf{y}_{i+1} \sim P_{\overrightarrow{\sigma}^\perp}(\mathbf{y}_i \rightarrow \mathbf{y}_{i+1})$
  $F_i^Y \leftarrow F_{i-1}^Y \cdot \frac{f_r(\mathbf{y}_{i+1} \rightarrow \mathbf{y}_i \rightarrow \mathbf{y}_{i-1})}{P_{\overrightarrow{\sigma}^\perp}(\mathbf{y}_i \rightarrow \mathbf{y}_{i+1})}$
  $\overrightarrow{p}_{i-1} = P_{i-1}^Y$
  if i = 1 **then**
    $\overleftarrow{p}_i \leftarrow 0$
  else
    $\overleftarrow{p}_i \leftarrow \left(P_{\overleftarrow{\sigma}^\perp}(\mathbf{y}_{i-1} \leftarrow \mathbf{y}_i) G(\mathbf{y}_{i-1} \leftrightarrow \mathbf{y}_i)\right)^\beta$
  end if
  $D_i^Y \leftarrow \frac{1 + \overleftarrow{p}_i D_{i-1}^Y}{\overrightarrow{p}_{i-1}}$
  $P_i^Y \leftarrow \left(P_{\overrightarrow{\sigma}^\perp}(\mathbf{y}_i \rightarrow \mathbf{y}_{i+1}) G(\mathbf{y}_{i-1} \leftrightarrow \mathbf{y}_i)\right)^\beta$
end while
$N_E \leftarrow i$

contribution as done in PT algorithm 1, all contributions are directly accumulated on the image. This is necessary because not all paths in the sample contribute to the same image pixel.

5.5 Participating Media

Up until now, we assumed that light only interacts with the scene at surface geometry. In reality however, light may also interact with participating media present in the scene. Lafortune extended the BDPT algorithm for rendering participating media [13]. The rendering equation 2.1 is only defined at point on the surface geometry. However, when rendering participating media, light may be scattered and absorbed at any point within the volume of participating media. Therefore, the rendering equation must be extended to incorporate these events. The rendering equation at any point within a medium corresponds to

$$\frac{\delta L(\mathbf{x}, \vec{\omega}_o, \lambda)}{\delta s} = \sigma(\mathbf{x}) \int_{S^2} f_s(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i, \lambda) L(\mathbf{x}, \vec{\omega}_i, \lambda) \, d\sigma(\vec{\omega}_i) - \kappa(\mathbf{x}) L(\mathbf{x}, \vec{\omega}_o, \lambda) \tag{5.10}$$

In this equation, $\sigma(\mathbf{x})$ is the scattering coefficient and $\kappa(\mathbf{x})$ the extinction coefficient at a point $\mathbf{x}$ in space. The function $f_s(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i, \lambda)$ is the Bidirectional Scattering Distribution Function, describing the fraction of incoming light along $\vec{\omega}_i$ that is scattered towards $\vec{\omega}_o$.

Algorithm 3 : SampleLightPath

sample $\mathbf{z}_1 \sim P_M(\mathbf{z}_1)$
sample $\mathbf{z}_2 \sim P_{\overrightarrow{\sigma}^\perp}(\mathbf{z}_1 \to \mathbf{z}_2)$

$F_1^Z \leftarrow \frac{L_e(\mathbf{z}_2 \to \mathbf{z}_1)}{P_M(\mathbf{z}_1) P_{\overrightarrow{\sigma}^\perp}(\mathbf{z}_1 \to \mathbf{z}_2)}$
$D_1^Z \leftarrow \frac{1}{P_M(\mathbf{z}_1)}$
$P_1^Z \leftarrow \left(P_{\overrightarrow{\sigma}^\perp}(\mathbf{z}_1 \to \mathbf{z}_2) G(\mathbf{z}_1 \leftrightarrow \mathbf{z}_2)\right)^\beta$

$i \leftarrow 1$
while path is not terminated **do**
  $i \leftarrow i + 1$
  sample $\mathbf{z}_{i+1} \sim P_{\overrightarrow{\sigma}^\perp}(\mathbf{z}_i \to \mathbf{z}_{i+1})$
  $F_i^Z \leftarrow F_{i-1}^Z \cdot \frac{f_r(\mathbf{z}_{i-1} \to \mathbf{z}_i \to \mathbf{z}_{i+1})}{P_{\overrightarrow{\sigma}^\perp}(\mathbf{z}_i \to \mathbf{z}_{i+1})}$
  $\overleftarrow{p}_i \left(\leftarrow P_{\overleftarrow{\sigma}^\perp}(\mathbf{z}_{i-1} \leftarrow \mathbf{z}_i) G(\mathbf{z}_{i-1} \leftrightarrow \mathbf{z}_i)\right)^\beta$
  $\overrightarrow{p}_{i-1} \leftarrow P_{i-1}^Z$
  $D_i^Z \leftarrow \frac{1 + \overleftarrow{p}_i D_{i-1}^Z}{\overrightarrow{p}_{i-1}}$
  $P_i^Z \leftarrow \left(P_{\overrightarrow{\sigma}^\perp}(\mathbf{z}_i \to \mathbf{z}_{i+1}) G(\mathbf{z}_{i-1} \leftrightarrow \mathbf{z}_i)\right)^\beta$
end while
$N_L \leftarrow i$

The extended rendering equation describes how the change in radiance at point $\mathbf{x}$ along $\vec{\omega}_o$ depends on both the damping of incoming radiance along $\vec{\omega}_o$ and the scattering of radiance coming from any other incoming directions $\vec{\omega}_i$ towards $\vec{\omega}_o$.

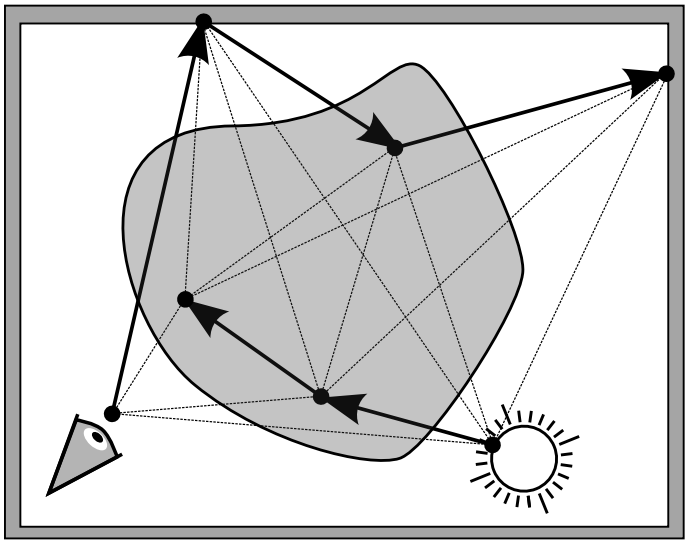

Figure 5.5: Bidirectional sampling with participating media.

To simulate this equation, light transport path may now contain scattering vertices within participating media. In the context of BDPT, this means that while generating an eye/light

Algorithm 4 : Connect

for $i = 0$ to N_E **do**
  {Check implicit path}
  if $i > 0$ and $L_e\left(\mathbf{y}_i \to \mathbf{y}_{i-1}\right)$ **then**
    $\overleftarrow{p}_Y \leftarrow \left(P_{\overleftarrow{\sigma}^\perp}\left(\mathbf{y}_{i-1} \leftarrow \mathbf{y}_i\right) G\left(\mathbf{y}_{i-1} \leftrightarrow \mathbf{y}_i\right)\right)^\beta$
    $\overrightarrow{p}_Z \leftarrow P_M\left(\mathbf{y}_i\right)^\beta$
    $D \leftarrow 1 + \frac{1+\overleftarrow{p}_Y D^Y_{i-1}}{P^Y_{i-1}} \overrightarrow{p}_Z$
    $F \leftarrow F^Y_{i-1} L_e\left(\mathbf{y}_i \to \mathbf{y}_{i-1}\right)$
    $w \leftarrow \frac{1}{D}$
    contribute wF along edge $\mathbf{y}_0\mathbf{y}_1$
  end if

  {Connect to light path}
  for $j = 1$ to N_L **do**
    if $\mathbf{y}_i$ is visible from $\mathbf{z}_j$ **then**
      $D \leftarrow 1$
      if $i > 0$ **then**
        $\overleftarrow{p}_Y \leftarrow \left(P_{\overleftarrow{\sigma}^\perp}\left(\mathbf{y}_{i-1} \leftarrow \mathbf{y}_i\right) G\left(\mathbf{y}_{i-1} \leftrightarrow \mathbf{y}_i\right)\right)^\beta$
        $\overrightarrow{p}_Z \leftarrow \left(P_{\overrightarrow{\sigma}^\perp}\left(\mathbf{z}_j \to \mathbf{y}_i\right) G\left(\mathbf{y}_i \leftrightarrow \mathbf{z}_j\right)\right)^\beta$
        $D \leftarrow D + \frac{1+\overleftarrow{p}_Y D^Y_{i-1}}{P^Y_{i-1}} \overrightarrow{p}_Z$
        $F \leftarrow F^E_{i-1} f_r\left(\mathbf{z}_j \to \mathbf{y}_i \to \mathbf{y}_{i-1}\right) G\left(\mathbf{y}_i \leftrightarrow \mathbf{z}_j\right)$
      else
        $F \leftarrow \hat{W}\left(\mathbf{z}_j \to \mathbf{y}_0\right) \frac{l^2 \vec{n}_{z_j} \cdot (\mathbf{y}_0 - \mathbf{z}_j)}{\rho\left(\vec{n}_{y_0} \cdot (\mathbf{z}_j - \mathbf{y}_0)\right)^3} P_S\left(\mathbf{z}_j\right)$
      end if
      $\overrightarrow{p}_Y \leftarrow \left(P_{\overrightarrow{\sigma}^\perp}\left(\mathbf{y}_i \to \mathbf{z}_j\right) G\left(\mathbf{y}_i \leftrightarrow \mathbf{z}_j\right)\right)^\beta$
      if $j > 1$ **then**
        $\overleftarrow{p}_Z \leftarrow \left(P_{\overleftarrow{\sigma}^\perp}\left(\mathbf{z}_{j-1} \leftarrow \mathbf{z}_j\right) G\left(\mathbf{z}_{j-1} \leftrightarrow \mathbf{z}_j\right)\right)^\beta$
        $D \leftarrow D + \frac{1+\overleftarrow{p}_Z D^Z_{j-1}}{P^Z_{j-1}} \overrightarrow{p}_Y$
        $F \leftarrow F \cdot F^L_{j-1} f_r\left(\mathbf{z}_{j+1} \to \mathbf{z}_j \to \mathbf{y}_i\right)$
      else
        $D \leftarrow D + \frac{\overrightarrow{p}_Y}{P_M(\mathbf{z}_1)}$
        $F \leftarrow F \cdot L_e\left(\mathbf{z}_1 \to \mathbf{y}_i\right)$
      end if
      $w \leftarrow \frac{1}{D}$
      if i = 0 **then**
        contribute wF along edge $\mathbf{y}_0\mathbf{z}_j$
      else
        contribute wF along edge $\mathbf{y}_0\mathbf{y}_1$
      end if
    end if
  end for
end for

path, the path scatters in mid-air with non-zero probability. Figure 5.5 shows an example BDPT sample accounting for scattering events. Again, during connection all vertices from the light path are connected to vertices on the eye path to generate complete light transport paths, scattering vertices are included in these connections. The path contribution and weights must account for the scatter and extinction probabilities along the connections.

During eye/light path generation, the next path vertex may be either a vertex on the surface geometry or within participating media. After each extension, the nearest surface intersection point with some distance d is computed. Then, a random scattering distance s is selected, based on the media's scattering and absorption characteristics. This distance indicates at what point along the ray the path will scatter. If this distance is further than the nearest intersection point ($s \geq d$), the intersection points is handled as usual. Otherwise ($s < d$), the scattering point is added as the next path vertex. Note that the selection of the scatter distance s based on the media's scattering and absorption characteristics is a form of importance sampling. This algorithm is capable of rendering scenes containing heterogeneous participating media.

Chapter 6

Metropolis Light Transport

In this chapter, we discuss the Metropolis Light Transport algorithm. In section 6.1, we start with a discussion of the Metropolis-Hastings algorithm, proposed by Metropolis and further generalized by Hastings [15, 6]. Following, we discuss the application of Metropolis-Hastings to light transport in detail in sections 6.2 and 6.3, resulting in the Metropolis Light Transport algorithm as presented by Veach [22]. In the remaining sections, we will discuss several important extensions to the MLT algorithm; We discuss the work of Pauly in section 6.4. Pauly extended MLT to support participating media, proposing two extra mutation strategies, the scattering and propagation mutation, specially designed for rendering participating media [18]. In section 6.5, we discuss the work of Szirmay-Kalos. Szirmay-Kalos presented an analysis of the start-up bias problem in MLT, analyzing its impact on the estimate [14]. In section 6.6, we discuss an alternative mutation strategy, proposed by Kelemen. This mutation strategy works on random number instead of path vertices [11]. In section 6.7, we discuss Hoberock's work on the use of different importance functions in MLT, resulting in a noise aware MLT sampler [8]. Cline developed the Energy Redistribution Path Tracing (ERPT) algorithm, discussed in secion 6.8. ERPT solves some of MLT's problems, as ERPT does not suffer from startup bias and does not require an accurate and unbiased estimate of a normalization constant [3]. Finally, in section 6.9 we discuss how Lai and Fan used the population Monte Carlo framework to adapt MLT mutation strategies during sampling, based on information gathered so far during sampling [28, 5].

6.1 Metropolis Sampling

As explained in section 3.2, perfect importance sampling requires sampling proportional to the function f. We stated that this was not generally possible without knowing f beforehand. However, in the context of unbiased rendering, f is known beforehand but because it is so complex and irregular, it is hard to sample proportional to f. Metropolis proposed an algorithm that is capable of sampling proportional to any function as long as it can be evaluated [15]. Hastings further generalized the algorithm for a broader class of applications [6], therefore, the algorithm is now known as the *Metropolis-Hastings* algorithm. The algorithm generates a sequence of samples $\mathbf{X}_0 \cdots \mathbf{X}_i \cdots$ with $p_i : \Omega \to \mathbb{R}$ being the probability density function for $\mathbf{X}_i$. The sequence is constructed as a Markov chain, so each sample only depends on its predecessor in the sequence. $\mathbf{X}_{i+1}$ is constructed from $\mathbf{X}_i$ by randomly mutating $\mathbf{X}_i$ according to some *mutation strategy*. The algorithm may use any convenient mutation strat-

egy, as long as it satisfies ergodicity. Some mutation strategies are however more effective than others. In section 6.3, we will discuss specific mutation strategies in the context of light transport. We will discuss ergodicity shortly. A mutation strategy is described by its *tentative transition function* $T(\mathbf{Y}|\mathbf{X})$, which is the probability density function for constructing $\mathbf{Y}$ as a mutation of $\mathbf{X}$.

The desired sample distribution is obtained by accepting or rejecting proposed mutations according to a carefully chosen acceptance probability. If a mutation is rejected the next sample will remain the same as the current sample ($\mathbf{X}_{i+1} = \mathbf{X}_i$). Let $a(\mathbf{Y}|\mathbf{X})$ be the acceptance probability for accepting mutation $\mathbf{Y}$ as $\mathbf{X}_{i+1}$, given $\mathbf{X} = \mathbf{X}_i$. The acceptance probability is chosen so that when $p_i \propto f$, so will p_{i+1}. Hence, the equilibrium distribution for the sample distribution sequence $p_0 \cdots p_i \cdots$ is proportional to f. This is achieved by letting $a(\mathbf{Y}|\mathbf{X})$ satisfy the *detailed balance* condition

$$f(\mathbf{Y})\, T(\mathbf{X}|\mathbf{Y})\, a(\mathbf{X}|\mathbf{Y}) = f(\mathbf{X})\, T(\mathbf{Y}|\mathbf{X})\, a(\mathbf{Y}|\mathbf{X}) \tag{6.1}$$

When the acceptance probability satisfies the detailed balance and the mutation strategy satisfies ergodicity, the probability density sequence will converge to the desired equilibrium distribution. In order to reach the equilibrium distribution as quickly as possible, the best strategy is to make the acceptance probability as large as possible. This results in the following acceptance probability:

$$a(\mathbf{X}|\mathbf{Y}) = \min\left(1, \frac{f(\mathbf{X})\, T(\mathbf{Y}|\mathbf{X})}{f(\mathbf{Y})\, T(\mathbf{X}|\mathbf{Y})}\right) \tag{6.2}$$

As mentioned earlier, ergodicity must be satisfied in order for the sequence to reach the equilibrium distribution. Ergodicity means that the sequence converges to the same distribution, no matter how $\mathbf{X}_0$ was chosen. In practice, it is sufficient that $T(\mathbf{Y}|\mathbf{X}) > 0$ for any $\mathbf{X}, \mathbf{Y} \in \Omega$ with $f(\mathbf{X}) > 0$ and $f(\mathbf{Y}) > 0$. In other words, all paths are reachable from all other paths through a single mutation. This is to prevent the sequence from getting stuck in a part of path space, unable to reach another part.

The sample sequence produced by the Metropolis algorithm is used for *perfect* importance sampling, proportional to f. Each sample contributes $\frac{f(\mathbf{X}_i)}{p_i(\mathbf{X}_i)}$ to the Monte Carlo estimator. As p_i is assumed to be proportional to f, $\frac{f(\mathbf{X}_i)}{p_i(\mathbf{X}_i)} = c$, however, it is usually not possible to analytically determine c. Integrating both sides of the equation results in

$$c = \int_\Omega f(\mathbf{X}) d\Omega(\mathbf{X}) \tag{6.3}$$

This equation can be used to estimate c. In the context of unbiased rendering, c is usually estimated using a small number of PT or BDPT samples.

Note that if $\mathbf{X}_0$ is not sampled according to f, that is $p_0 \not\propto f$, then $\mathbf{X}_i$ will have the desired distribution only in the limit as $i \to \infty$. The bias introduced by the difference between p_i and $\frac{f}{c}$ is called startup bias and will result in bias in the Monte Carlo estimate. The startup bias is often reduced by discarding the first k samples, but it is difficult to choose an appropriate k. In section 6.8, we discuss an approach to eliminate startup bias altogether.

6.2 Metropolis Light Transport

Veach applied the Metropolis-Hastings algorithm to the light transport problem to reduce the variance in the Monte Carlo estimate for each pixel [24, 22]. The Metropolis algorithm is used

to generate a sequence of light transport paths, sampling proportional to some function f. Veach proposed to not apply the Metropolis algorithm directly to estimate the measurement equation of an individual pixel. Instead, he proposed to generate samples proportional to the combined measurement contribution function for all m pixel sensors

$$f(\mathbf{X}) = \sum_{j=1}^{m} f_j(\mathbf{X}) \tag{6.4}$$

These samples are then shared to estimate the individual integrals I_j. This has several advantages; First of all, because the measurement functions for nearby pixels are often very similar, applying small mutations to hard-to-find paths often results in similar paths that contribute to nearby pixels. Second, as there is only one Metropolis sequence instead of m, the impact of startup bias is reduced and the normalization constant c has to be estimated only once. A disadvantage is that the number of samples contributing to some pixel j becomes proportional to I_j. When there are large variations in the brightness over the image plane, dark areas in the image become relatively undersampled compared to brighter areas.

When estimating multiple integrals at once, the Metropolis Monte Carlo estimators may be further improved by not only letting accepted mutations contribute, but the rejected mutations as well. Assume $\mathbf{X}_i = \mathbf{X}$ and let $\mathbf{Y}$ be the proposed mutation. Then, $\mathbf{Y}$ is accepted as $\mathbf{X}_{i+1}$ with probability $a(\mathbf{Y}|\mathbf{X})$ and $\mathbf{X}$ is accepted ($\mathbf{Y}$ is rejected) as $\mathbf{X}_{i+1}$ with probability $1 - a(\mathbf{Y}|\mathbf{X})$. Hence, the expected contributions of $\mathbf{X}$ and $\mathbf{Y}$ equal $c\,(1 - a(\mathbf{Y}|\mathbf{X}))$ resp. $ca(\mathbf{Y}|\mathbf{X})$. Instead of only letting $\mathbf{X}_{i+1}$ contribute c to the estimates, it is possible to let $\mathbf{X}$ and $\mathbf{Y}$ both contribute their expected contributions instead. Because different paths may contribute to different pixels, the average number of samples contributing to a pixel increases, especially for relatively dark pixels having on average lower acceptance probabilities.

Algorithm 5 shows a general Metropolis Light Transport (MLT) sampler. The sampler generates a sequence of N samples using the Metropolis-Hastings algorithm. Of these, the first k are discarded to eliminate startup bias. The algorithm's initial path $\mathbf{X}$ and estimated normalization constant c are passed as parameters to the sampler. For an estimator with low bias, N should be significantly large.

Algorithm 5 : Metropolis($\mathbf{X}, N, k, c$)

```
X_0 ← X
for i = 1 to N + k do
    Y ←mutate(X_{i−1}) ∝ T(Y|X_{i−1})
    a ← a(Y|X_{i−1})
    if i ≥ k then
        contribute a c/N to image along X_{i−1}
        contribute (1 − a) c/N to image along Y
    end if
    if a ≥ U(0, 1) then
        X_i ← Y
    else
        X_i ← X_{i−1}
    end if
end for
```

What is left to discuss is the mutation strategy used to construct the sample sequence. As mentioned earlier, the mutation strategy should satisfy ergodicity. This is achieved by sampling a *fresh* path using either PT or BDPT with non-zero probability. For the remainder of mutations, the path vertices are perturbed in order to locally explore path space. In the next section, we discuss the most important mutations in path space.

6.3 Path Space Mutation

Paths are mutated in path space by applying small mutations to path vertices and edges. Veach proposed two simple perturbation methods, the lens and caustic mutations [24]. Caustic mutations are applied to paths of the form $EDS(S|D)^*L$, these paths are responsible for caustic effects, hence the name. Figure 6.3 shows an example of a caustic mutation. Lens mutations are applied to all remaining paths[1]. Figure 6.1 shows an example of a lens mutation. Both mutation types only perturb the first i vertices of the path $\mathbf{x}_0 \cdots \mathbf{x}_k$ with $0 < i \leq k$. When $i < k$, the mutated path is formed by concatenating the perturbed subpath $\mathbf{x}'_0 \cdots \mathbf{x}'_i$ and the original subpath $\mathbf{x}_{i+1} \cdots \mathbf{x}_k$ to form the complete path $\mathbf{x}'_0 \cdots \mathbf{x}'_i \mathbf{x}_{i+1} \cdots \mathbf{x}_k$. If $i = k$, then $\mathbf{x}'_0 \cdots \mathbf{x}'_i$ already forms a complete path. The signature and length of the path are unaffected by these mutations. When the signature of the path does change during mutation, the mutation is immediately rejected. In the next section we will discuss in more detail how lens and caustic mutations are sampled, including a variation on the lens mutation proposed by Lai.

Sampling

Lens mutation: The lens mutation creates a new path from an existing path, beginning at the eye. The mutation is started by perturbing the outgoing eye direction $\mathbf{x}_0 \rightarrow \mathbf{x}_1$, which usually means the mutation will contribute to a different pixel. The new eye subpath is propagated forward through the same number of specular bounces as the original path, until the first diffuse vertex $\mathbf{x}_j$ is reached. If the next vertex $\mathbf{x}_{j+1}$ on the original path is diffuse, the mutated vertex $\mathbf{x}'_j$ is explicitly connected to $\mathbf{x}_{j+1}$ to form the complete mutation $\mathbf{x}'_0 \cdots \mathbf{x}'_j \mathbf{x}_{j+1} \cdots \mathbf{x}_k$ [2]. However, if the next vertex $\mathbf{x}_{j+1}$ is specular, the outgoing direction $\mathbf{x}_j \rightarrow \mathbf{x}_{j+1}$ is perturbed by a small angle and the eye subpath is further extended through another chain of specular vertices until the next diffuse vertex is reached. This process is repeated until either two consecutive diffuse vertices are connected, or the light source is reached. Figure 6.1 shows the mutation of a path of the form $ESDSDL$. First, the outgoing direction from the eye is perturbed. The mutation is then extended through the first specular bounce. The outgoing direction of the first diffuse vertex is also perturbed and the mutation is again extended through another specular bounce. Finally, the path is explicitly connected to the light source to complete the mutation.

The lens mutation mutates the first $m + 1$ path vertices of a path through forward sampling. As explained, part of the mutation is sampled forward. Therefore when computing the mutation probability $p(\mathbf{Y}|\mathbf{X})$ some mutation probabilities must be converted from unit projected solid angle to unit surface area. Let $P_L^m(\mathbf{Y}|\mathbf{X})$ be the probability of generating $\mathbf{Y}$

[1] Veach differentiates between lens and multi-chain perturbations, we do not make this distinction [24].

[2] For an explicit connection to make sense, the two vertices involved both need to be diffuse. Therefore, the next vertex must be diffuse.

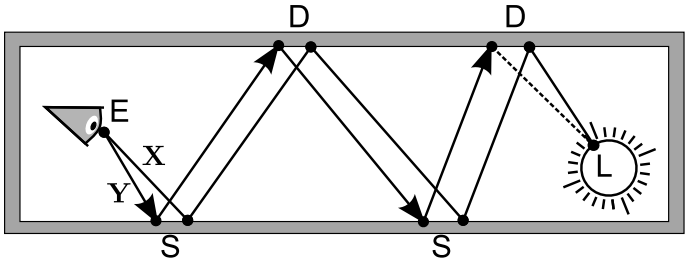

Figure 6.1: Lens mutation of a path of the form $ESDSDL$. The mutation starts at the eye. Further path vertices are mutated forward until an explicit connection can be made.

from $\mathbf{X}$ per unit path space by mutating the first $m+1$ vertices using the lens mutation. Then, using the conversion factors from equations 3.10 and 3.18, we get

$$\begin{aligned}\frac{P_L^m\left(\mathbf{X}|\mathbf{Y}\right)}{p_L^m\left(\mathbf{Y}|\mathbf{X}\right)} =& \frac{l^2\vec{n}_1\cdot\left(\mathbf{x}_1-\mathbf{x}_0\right)}{l^2\vec{n}_1\cdot\left(\mathbf{y}_1-\mathbf{y}_0\right)}\frac{\rho\left(\vec{n}_0\cdot\left(\mathbf{y}_1-\mathbf{y}_0\right)\right)^3}{\rho\left(\vec{n}_0\cdot\left(\mathbf{x}_1-\mathbf{x}_0\right)\right)^3}\prod_{i=1}^{m-1}\frac{G\left(\mathbf{x}_i\leftrightarrow\mathbf{x}_{i+1}\right)}{G\left(\mathbf{y}_i\leftrightarrow\mathbf{y}_{i+1}\right)} \\ &\cdot\frac{P_I\left(\mathbf{x}_0|\mathbf{Y}\right)P_S\left(\mathbf{x}_1|\mathbf{Y}\right)}{P_I\left(\mathbf{y}_0|\mathbf{X}\right)P_S\left(\mathbf{y}_1|\mathbf{X}\right)}\prod_{i=1}^{m-1}\frac{P_{\vec{\sigma}^\perp}\left(\mathbf{x}_i\rightarrow\mathbf{x}_{i+1}|\mathbf{Y}\right)}{P_{\vec{\sigma}^\perp}\left(\mathbf{y}_i\rightarrow\mathbf{y}_{i+1}|\mathbf{X}\right)}\end{aligned} \tag{6.5}$$

Lai Lens mutation: Lai proposed an alternative lens mutation that requires only perturbing the outgoing eye direction $\mathbf{x}_0\rightarrow\mathbf{x}_1$ [27]. Just like the original lens mutation, the new eye subpath is propagated forward through the same number of specular bounces as the original path, until the first diffuse vertex $\mathbf{x}_j$ is reached. However, if the next vertex $\mathbf{x}_{j+1}$ is specular, instead of perturbing the outgoing direction $\mathbf{x}_j\rightarrow\mathbf{x}_{j+1}$ as in the original lens mutation, an explicit connection is made to the specular vertex $\mathbf{x}_{j+1}$ on the original path. If this connection succeeds, the mutation is extended through the remaining specular vertices until the next diffuse vertex is reached. Again, this process is repeated until either two consecutive diffuse vertices are connected, or the light source is reached. Figure 6.2 shows the mutation of a path of the form $ESDSDL$. First, the outgoing direction from the eye is perturbed. The mutation is then extended through the first specular bounce. Instead of perturbing the outgoing direction, the first diffuse vertex is connected to the next specular vertex and the mutation is again extended through another specular bounce. Finally, the path is explicitly connected to the light source to complete the mutation.

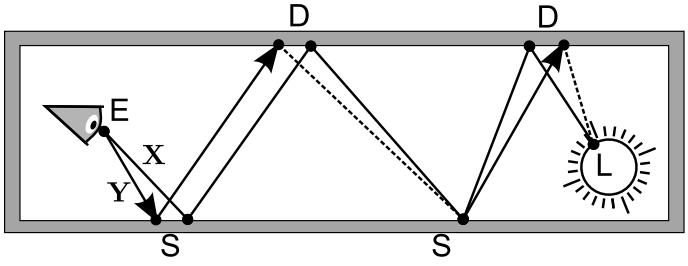

Figure 6.2: Lai Lens mutation of a path of the form $ESDSDL$. The mutation starts at the eye. Further path vertices are mutated forward until an explicit connection can be made. Diffuse vertices are connected directly to specular vertices while extending the path.

Caustic mutation: Caustic mutations are very similar to lens mutations, but perturb the path backward towards the eye, instead of starting at the eye and working forward to the light source. The caustic perturbation creates a new path from an existing path, beginning at the second diffuse vertex $\mathbf{x}_i$ from the eye. The mutation starts by perturbing the outgoing direction $\mathbf{x}_i \rightarrow \mathbf{x}_{i-1}$. The new subpath is propagated backward through $i-2$ specular bounces until the first diffuse vertex $\mathbf{x}_1$ is reached. This vertex is then explicitly connected to the eye vertex $\mathbf{x}_0$ to complete the mutation. Figure 6.3 shows the mutation of a path of the form $EDSDL$. The mutation starts by mutating the outgoing direction at the second diffuse vertex. The mutation is then extended through one specular bounce. Finally, the first diffuse vertex is explicitly connected to the eye to complete the mutation.

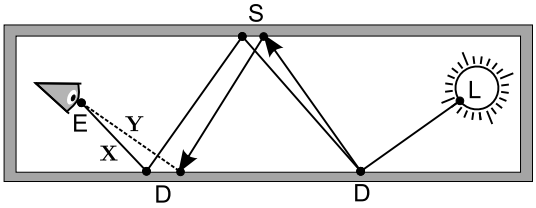

Figure 6.3: Caustic mutation of a path of the form $EDSDL$. The mutation starts at the second diffuse vertex from the eye. All preceding vertices are perturbed backward and an explicit connection to the eye is made.

Similarly to lens mutations, the caustic mutation mutates the first $m+1$ path vertices of a path, however this time through backward sampling. Thus again, when computing $P(\mathbf{Y}|\mathbf{X})$ some mutation probabilities must be converted from unit projected solid angle to unit surface area. Let $P_C^m(\mathbf{Y}|\mathbf{X})$ be the probability of generating $\mathbf{Y}$ from $\mathbf{X}$ per unit path space by mutating the first $m+1$ path vertices using the caustic mutation. Then, using the conversion factors from equation 3.11, we get

$$\frac{P_C^m(\mathbf{X}|\mathbf{Y})}{p_C^m(\mathbf{Y}|\mathbf{X})} = \prod_{i=1}^{m} \frac{G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})}{G(\mathbf{y}_i \leftrightarrow \mathbf{y}_{i+1})} \cdot \frac{P_I(\mathbf{x}_0|\mathbf{Y})}{P_I(\mathbf{y}_0|\mathbf{X})} \prod_{i=1}^{m-1} \frac{P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}|\mathbf{Y})}{P_{\overleftarrow{\sigma}^\perp}(\mathbf{y}_i \leftarrow \mathbf{y}_{i+1}|\mathbf{X})} \cdot \frac{P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_m \leftarrow y_{m+1}|\mathbf{Y})}{P_{\overleftarrow{\sigma}^\perp}(\mathbf{y}_m \leftarrow \mathbf{x}_{m+1}|\mathbf{X})} \tag{6.6}$$

When using more advanced camera models with a finite aperture lens, both mutations should also perturb the eye vertex $\mathbf{x}_0$ itself. For simple models such as the pinhole camera, there is only one valid $\mathbf{x}_0$, so perturbing $\mathbf{x}_0$ is not necessary. Aside from these perturbation mutations, Veach also proposed mutations that substitute any subpath by a completely new subpath of possibly different length and signature, using bidirectional mutations [24]. We will not discuss these mutations here, and direct the interested reader to the original paper on Metropolis light transport [24]. Because the original paper is found to be difficult to understand by many people trying to implement the Metropolis Light Transport algorithm, Cline presented a comprehensive tutorial on Metropolis Light Transport and its implementation [2].

Acceptance probability

In this section, we show how to compute the acceptance probability for the lens and caustic mutations. Remember from section 6.1 that for some path $\mathbf{Y}$ that is generated from path $\mathbf{X}$

through mutations, the acceptance probability in the MLT algorithm is defined as

$$a = \min\left(1, \frac{f(\mathbf{Y})\, p(\mathbf{X}|\mathbf{Y})}{f(\mathbf{X})\, p(\mathbf{Y}|\mathbf{X})}\right) \tag{6.7}$$

Because usually only a part of the path is mutated, many of the factors in this equation cancel out.

Lets start with the measurement contribution factor $\frac{f(\mathbf{Y})}{f(\mathbf{X})}$ in the acceptance probability. When $0 < m < k$, that is, only the first $m+1$ vertices on the path are mutated and the remaining vertices $\mathbf{x}_{m+1} \cdots \mathbf{x}_k = \mathbf{y}_{m+1} \cdots \mathbf{y}_k$ stay the same, some of the factors in both measurement contribution functions cancel out, resulting in

$$\frac{f(\mathbf{y}_0 \cdots \mathbf{y}_k)}{f(\mathbf{x}_0 \cdots \mathbf{x}_k)} = \frac{W_i(\mathbf{y}_1 \to \mathbf{y}_0)}{W_i(\mathbf{x}_1 \to \mathbf{x}_0)} \prod_{j=0}^{m} \frac{G(\mathbf{y}_j \leftrightarrow \mathbf{y}_{j+1})}{G(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1})} \prod_{j=1}^{m+1} \frac{f_r(\mathbf{y}_{j+1} \to \mathbf{y}_j \to \mathbf{y}_{j-1})}{f_r(\mathbf{x}_{j+1} \to \mathbf{x}_j \to \mathbf{x}_{j-1})} \tag{6.8}$$

In case the entire path is mutated by a lens mutation ($m = k$), no common factors appear and the the measurement contribution factor becomes

$$\frac{f(\mathbf{y}_0 \cdots \mathbf{y}_k)}{f(\mathbf{x}_0 \cdots \mathbf{x}_k)} = \frac{W_i(\mathbf{y}_1 \to \mathbf{y}_0)}{W_i(\mathbf{x}_1 \to \mathbf{x}_0)} \prod_{j=0}^{k-1} \frac{G(\mathbf{y}_j \leftrightarrow \mathbf{y}_{j+1})}{G(\mathbf{x}_j \leftrightarrow \mathbf{x}_{j+1})} \prod_{j=1}^{k} \frac{f_r(\mathbf{y}_{j+1} \to \mathbf{y}_j \to \mathbf{y}_{j-1})}{f_r(\mathbf{x}_{j+1} \to \mathbf{x}_j \to \mathbf{x}_{j-1})} \tag{6.9}$$

Note that in these equations, we used, for convenience, the special notation $L_e(\mathbf{x}_k \to \mathbf{x}_{k-1}) = f_r(\mathbf{x}_{k+1} \to \mathbf{x}_k \to \mathbf{x}_{k-1})$. In this case, $\mathbf{x}_{k+1}$ can be thought of as an implicit vertex on all light transport paths being the source of all light in the scene [22].

What is left is the probability factor $\frac{p(\mathbf{X}|\mathbf{Y})}{p(\mathbf{Y}|\mathbf{X})}$ in the acceptance probability. Because the lens and caustic mutations are sampled differently, this factor differs for both mutations strategies. Let's start with the lens mutation. Using equation 6.5, we get

$$\begin{aligned}\frac{P_L^m(\mathbf{X}|\mathbf{Y})}{p_L^m(\mathbf{Y}|\mathbf{X})} =& \frac{l^2 \vec{n}_1 \cdot (\mathbf{x}_1 - \mathbf{x}_0)}{l^2 \vec{n}_1 \cdot (\mathbf{y}_1 - \mathbf{y}_0)} \frac{\rho(\vec{n}_0 \cdot (\mathbf{y}_1 - \mathbf{y}_0))^3}{\rho(\vec{n}_0 \cdot (\mathbf{x}_1 - \mathbf{x}_0))^3} \prod_{i=1}^{m-1} \frac{G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})}{G(\mathbf{y}_i \leftrightarrow \mathbf{y}_{i+1})} \\ & \cdot \frac{P_I(\mathbf{x}_0|\mathbf{Y})\, P_S(\mathbf{x}_1|\mathbf{Y})}{P_I(\mathbf{y}_0|\mathbf{X})\, P_S(\mathbf{y}_1|\mathbf{X})} \prod_{i=1}^{m-1} \frac{P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1}|\mathbf{Y})}{P_{\overrightarrow{\sigma}^\perp}(\mathbf{y}_i \to \mathbf{y}_{i+1}|\mathbf{X})}\end{aligned} \tag{6.10}$$

Similarly for the caustic mutation, using equation 6.6, we get

$$\frac{P_C^m(\mathbf{X}|\mathbf{Y})}{p_C^m(\mathbf{Y}|\mathbf{X})} = \prod_{i=1}^{m} \frac{G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})}{G(\mathbf{y}_i \leftrightarrow \mathbf{y}_{i+1})} \cdot \frac{P_I(\mathbf{x}_0|\mathbf{Y})}{P_I(\mathbf{y}_0|\mathbf{X})} \prod_{i=1}^{m-1} \frac{P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}|\mathbf{Y})}{P_{\overleftarrow{\sigma}^\perp}(\mathbf{y}_i \leftarrow \mathbf{y}_{i+1}|\mathbf{X})} \cdot \frac{P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_m \leftarrow y_{m+1}|\mathbf{Y})}{P_{\overleftarrow{\sigma}^\perp}(\mathbf{y}_m \leftarrow \mathbf{x}_{m+1}|\mathbf{X})} \tag{6.11}$$

The acceptance probability term $\frac{f(\mathbf{Y})p(\mathbf{X}|\mathbf{Y})}{f(\mathbf{X})p(\mathbf{Y}|\mathbf{X})}$ can now be obtained by combining the above equations. Note that geometric factors appear in both the measurement contribution factor $\frac{f(\mathbf{Y})}{f(\mathbf{X})}$ and the probability factor $\frac{p(\mathbf{X}|\mathbf{Y})}{p(\mathbf{Y}|\mathbf{X})}$. Many of these common factors will cancel out in the final acceptance probability.

Figures 6.4 and 6.5 show how to compute the acceptance probability term for the lens mutation after all common factors are canceled out. Figures 6.4 shows a partially mutated path, requiring an explicit connection, while figure 6.5 shows a fully mutated path. The

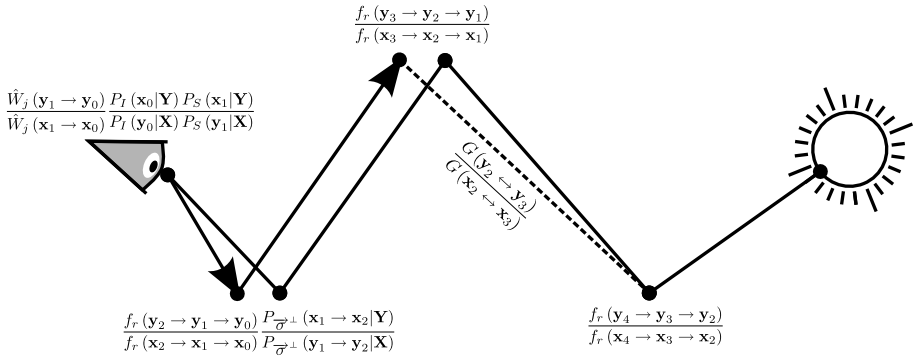

Figure 6.4: Acceptance probability for partial lens mutation.

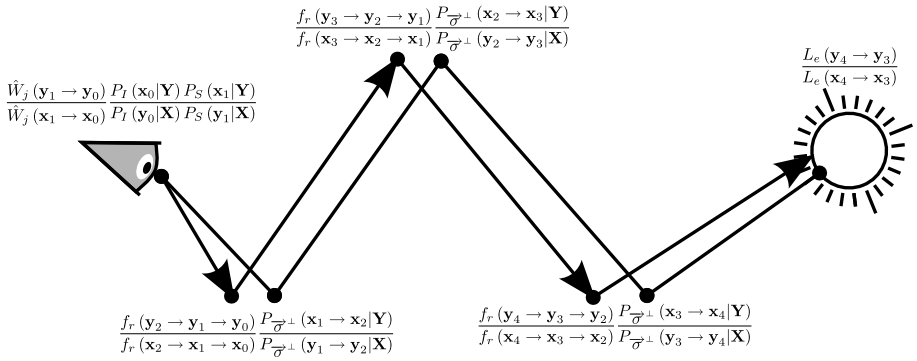

Figure 6.5: Acceptance probability for full lens mutation.

figures give an indication of the order in which the acceptance probability term is computed during mutation construction, starting with evaluating the modified sensor sensitivity function and iteratively extending the mutation with one vertex, evaluating the BSDF and sample probabilities and finally evaluating the connection or light source emittance. Note that we used the modified sensor sensitivity function for the finite aperture lens from appendix 3.4.

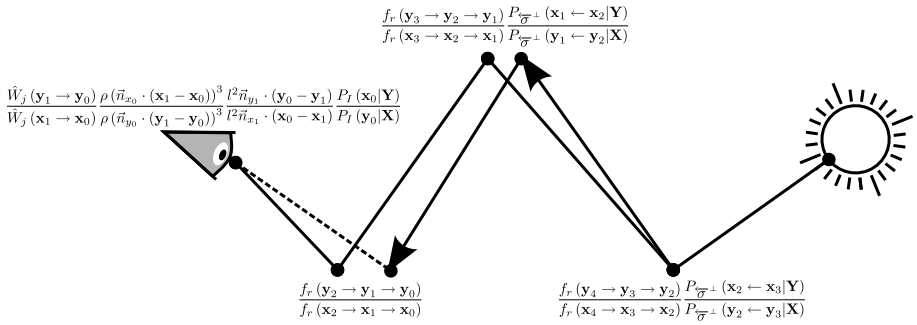

Figure 6.6: Acceptance probability for partial caustic mutation.

Figures 6.6 and 6.7 show how to compute the acceptance probability term for the caustic mutation. Figures 6.6 shows a partially mutated path, starting somewhere in the middle of the path, while figure 6.7 shows a fully mutated path, starting at the light source. Again, the

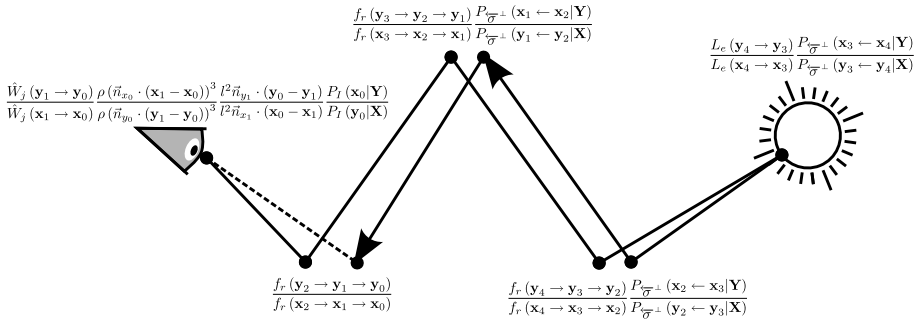

Figure 6.7: Acceptance probability for full caustic mutation.

figures give an indication of the order in which the acceptance probability term is computed during mutation construction, starting at the light source or an intermediate vertex and iteratively extending the mutation backwards with one vertex, evaluating the BSDF and sample probabilities and finally making an explicit connection to the eye and evaluating the sensor sensitivity function. Similar to the bidirectional sample in figure 5.2, the use of the modified sensor sensitivity function in the caustic mutation requires an extra correction factor $\frac{l^2 \vec{n}_1 \cdot (\mathbf{x}_0 - \mathbf{x}_1)}{\rho(\vec{n}_0 \cdot (\mathbf{x}_1 - \mathbf{x}_0))^3}$ because vertex $\mathbf{x}_1$ is sampled backward.

In practice, most mutations are symmetrical, so the fraction $\frac{P_I(\mathbf{x}_0|\mathbf{Y})}{P_I(\mathbf{y}_0|\mathbf{X})}$, $\frac{P_S(\mathbf{x}_1|\mathbf{Y})}{P_S(\mathbf{y}_1|\mathbf{X})}$, $\frac{P_{\overleftarrow{\sigma}^\perp}(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}|\mathbf{Y})}{P_{\overleftarrow{\sigma}^\perp}(\mathbf{y}_i \leftarrow \mathbf{y}_{i+1}|\mathbf{X})}$ and $\frac{P_{\overrightarrow{\sigma}^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1}|\mathbf{Y})}{P_{\overrightarrow{\sigma}^\perp}(\mathbf{y}_i \to \mathbf{y}_{i+1}|\mathbf{X})}$ all cancel out to 1, significantly simplifying the computations.

Algorithm

Using the acceptance probability formulations from the previous section, we now give pseudo code for the lens and caustic mutations. Algorithm 6 and 7 give the caustic and lens mutation algorithms in pseudo code. The algorithms take an initial path $\mathbf{X}_k$ of length k and the number m, indicating the number of vertices to mutate, as input and return the mutated path $\mathbf{y}_0 \cdots \mathbf{y}_k$ and the corresponding acceptance probability.

Both algorithms start with mutating the eye vertex $\mathbf{x}_0$. Note that this is only relevant when using a finite aperture lens. The caustic mutation proceeds with mutating the path vertices backward and finally making an explicit connection to the mutated eye vertex. The lens mutation first mutates path vertex $\mathbf{x}_1$, according to the camera model (see appendix 3.4), before mutating the remaining vertices forward from the eye. If not all vertices are mutated, an explicit connection is established.

Whenever one of the algorithms terminate before completing the mutation, for example because an explicit connection failed or an extension ray misses all scene geometry, effectively terminating the mutation, a zero acceptance probability is returned. No mutated path is returned as it will be rejected anyway.

6.4 Participating Media

The lens and caustic mutations as presented by Veach are not designed to handle participating media [22]. Pauly proposed two extra mutation strategies, the scattering and propagation mutation, specially designed for rendering participating media [18]. Both mutations apply

Algorithm 6 : CausticMutation($\mathbf{X}_k, m$)

{*Mutate the first m+1 vertices*}
sample $\mathbf{y}_0 \sim P_I(\mathbf{y}_0|\mathbf{x}_0)$
$\mathbf{y}_{m+1} \cdots \mathbf{y}_k = \mathbf{x}_{m+1} \cdots \mathbf{x}_k$
$T \leftarrow \frac{P_I(\mathbf{x}_0|\mathbf{y}_0)}{P_I(\mathbf{y}_0|\mathbf{x}_0)}$
for $i = m$ to 1 **do**
  sample $\mathbf{y}_i \sim P_{\overleftarrow{\sigma}^\perp}\left(\mathbf{y}_i \leftarrow \mathbf{y}_{i+1}|\mathbf{x}_i\mathbf{x}_{i+1}\right)$
  if path is terminated **then**
    return 0
  end if
  $T \leftarrow T \cdot \frac{P_{\overleftarrow{\sigma}^\perp}\left(\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}|\mathbf{y}_i\mathbf{y}_{i+1}\right)}{P_{\overleftarrow{\sigma}^\perp}\left(\mathbf{y}_i \leftarrow \mathbf{y}_{i+1}|\mathbf{x}_i\mathbf{x}_{i+1}\right)}$
  if $i + 1 = k$ **then**
    $T \leftarrow T \cdot \frac{L_e\left(\mathbf{y}_k \rightarrow \mathbf{y}_{k-1}\right)}{L_e(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1})}$
  else
    $T \leftarrow T \cdot \frac{f_r\left(\mathbf{y}_{i+2} \rightarrow \mathbf{y}_{i+1} \rightarrow \mathbf{y}_i\right)}{f_r(\mathbf{x}_{i+2} \rightarrow \mathbf{x}_{i+1} \rightarrow \mathbf{x}_i)}$
  end if
end for
if $V(\mathbf{y}_0 \leftrightarrow \mathbf{y}_1) = 0$ **then**
  return 0
end if
$T \leftarrow T \cdot \frac{\hat{W}_i(\mathbf{y}_1 \rightarrow \mathbf{y}_0)}{\hat{W}_i(\mathbf{x}_1 \rightarrow \mathbf{x}_0)} \frac{\rho\left(\vec{n}_{x_0} \cdot (\mathbf{x}_1 - \mathbf{x}_0)\right)^3}{\rho\left(\vec{n}_{y_0} \cdot (\mathbf{y}_1 - \mathbf{y}_0)\right)^3} \frac{l^2 \vec{n}_{y_1} \cdot (\mathbf{y}_0 - \mathbf{y}_1)}{l^2 \vec{n}_{x_1} \cdot (\mathbf{x}_0 - \mathbf{x}_1)}$
$T \leftarrow T \cdot \frac{f_r(\mathbf{y}_2 \rightarrow \mathbf{y}_1 \rightarrow \mathbf{y}_0)}{f_r(\mathbf{x}_2 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_0)}$

{*Return acceptance probability and mutated path*}
return $\langle \min(1, T), y_0 \cdots y_k \rangle$

small perturbations to a scattering vertex on an existing light path. The scattering mutation perturbs the outgoing scattering direction at a scattering vertex, similar to the way lens and caustic mutations perturb outgoing directions at regular path vertices. The propagation

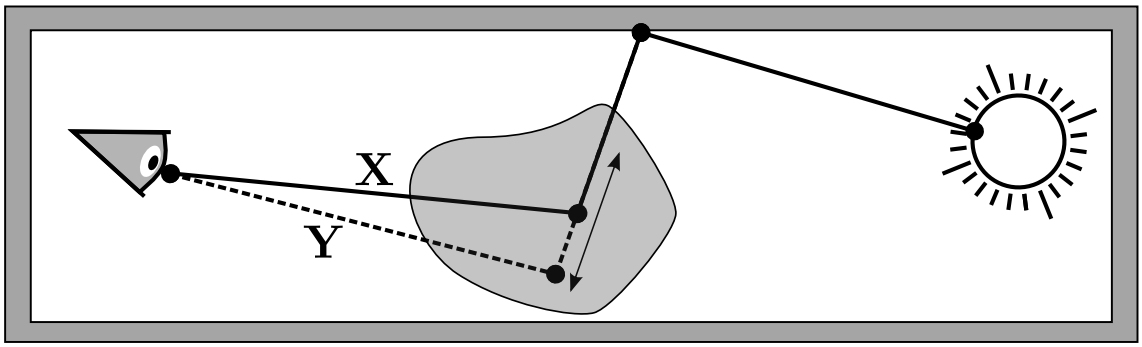

Figure 6.8: The propagation mutation perturbs a scattering vertex by moving it along its incoming or outgoing path edge.

Algorithm 7 : LensMutation($\mathbf{X}_k, m$)

{*Mutate the first m+1 vertices*}
sample $\mathbf{y}_0 \sim P_I\left(\mathbf{y}_0|\mathbf{x}_0\right)$
sample $\mathbf{y}_1 \sim P_S\left(\mathbf{y}_1|\mathbf{x}_1\right)$
if path is terminated **then**
  return 0
end if
$T \leftarrow \frac{\hat{W}_i(\mathbf{y}_1 \to \mathbf{y}_0)}{\hat{W}_i(\mathbf{x}_1 \to \mathbf{x}_0)} \frac{P_I(\mathbf{x}_0|\mathbf{y}_0) P_S(\mathbf{x}_1|\mathbf{y}_1)}{P_I(\mathbf{y}_0|\mathbf{x}_0) P_S(\mathbf{y}_1|\mathbf{x}_1)}$
for $i = 2$ to m **do**
  sample $\mathbf{y}_i \sim P_{\overrightarrow{\sigma}^\perp}\left(\mathbf{y}_{i-1} \to \mathbf{y}_i|\mathbf{x}_{i-1}\mathbf{x}_i\right)$
  if path is terminated **then**
    return 0
  end if
  $T \leftarrow T \cdot \frac{f_r\left(\mathbf{y}_i \to \mathbf{y}_{i-1} \to \mathbf{y}_{i-2}\right)}{f_r(\mathbf{x}_i \to \mathbf{x}_{i-1} \to \mathbf{x}_{i-2})} \frac{P_{\overrightarrow{\sigma}^\perp}\left(\mathbf{x}_{i-1} \to \mathbf{x}_i|\mathbf{y}_{i-1}\mathbf{y}_i\right)}{P_{\overrightarrow{\sigma}^\perp}\left(\mathbf{y}_{i-1} \to \mathbf{y}_i|\mathbf{x}_{i-1}\mathbf{x}_i\right)}$
end for

{*Make an explicit connection*}
if $m < k$ **then**
  if $V\left(\mathbf{y}_m \leftrightarrow \mathbf{y}_{m+1}\right) = 0$ **then**
    return 0
  end if
  $T \leftarrow T \cdot \frac{G\left(\mathbf{y}_m \leftrightarrow \mathbf{y}_{m+1}\right)}{G(\mathbf{x}_m \leftrightarrow \mathbf{x}_{m+1})} \frac{f_r\left(\mathbf{y}_{m+1} \to \mathbf{y}_m \to \mathbf{y}_{m-1}\right)}{f_r(\mathbf{x}_{m+1} \to \mathbf{x}_m \to \mathbf{x}_{m-1})}$
  $\mathbf{y}_{m+1} \cdots \mathbf{y}_k = \mathbf{x}_{m+1} \cdots \mathbf{x}_k$
end if

if $m \geq k - 1$ **then**
  $T \leftarrow T \cdot \frac{L_e\left(\mathbf{y}_k \to \mathbf{y}_{k-1}\right)}{L_e(\mathbf{x}_k \to \mathbf{x}_{k-1})}$
else
  $T \leftarrow T \cdot \frac{f_r\left(\mathbf{y}_{m+2} \to \mathbf{y}_{m+1} \to \mathbf{y}_m\right)}{f_r(\mathbf{x}_{m+2} \to \mathbf{x}_{m+1} \to \mathbf{x}_m)}$
end if

{*Return acceptance probability and mutated path*}
return $\langle \min\left(1, T\right), y_0 \cdots y_k \rangle$

mutation displaces the scattering vertex along its incoming or outgoing path edge, moving the scattering vector closer or further along the edge. The perturbed vertex is connected to the remainder of the path in order to form a complete light transport path. Figure 6.8 shows an example of the propagation mutation. Both mutation strategies explore path space locally, allowing it to efficiently render difficult effects involving participating media, such as volumetric caustics.

6.5 Startup Bias

As explained in section 6.1, the MLT algorithm suffers from startup bias because the distribution of the initial sample is different from the desired equilibrium distribution. As more samples are generated, the impact of start-up bias on the final estimate reduces. Szirmay-Kalos presented an analysis of this start-up bias problem in MLT, analyzing its impact on the estimate [14]. He performed a theoretical analyzes on a simplified model using Fourier analyzes. He furthermore validated the results using simulations on an actual MLT implementation, using BDPT to generate initial samples. He concluded that the impact of start-up bias can be quite significant, especially for relatively uniform integrands. Therefore, MLT becomes efficient only for difficult integrand.

6.6 Primary Sample Space Mutation

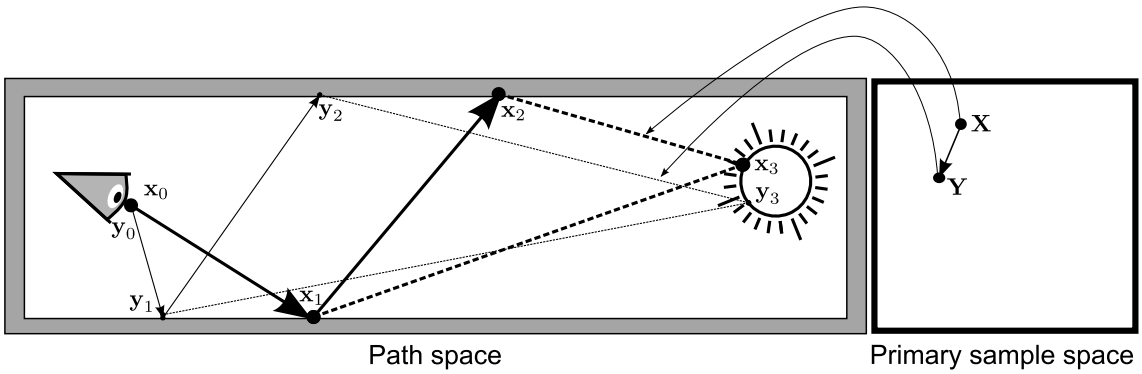

Figure 6.9: Mutation in primary sample space correspond to mutations in path space.

In practice, the actual implementation of the mutations discussed so far has been found to be quite difficult to do correctly. Kelemen proposed an alternative, much simpler mutation strategy, working on random number instead of path vertices [11]. He made the observation that, whether using PT, BDPT or some variation of these, paths are always generated from a list of random numbers. Therefore, each sample can be described as a list of random numbers and a mapping from random numbers to the actual path in path space. This gave way to primary sample space, the space of these random number lists. In essence, samples are uniformly drawn from primary sample space and mapped to their corresponding path in path space. Kelemen proposed to apply mutations directly to the samples in primary sample space, instead of mutating the path vertices in path space. By applying small perturbations to all random numbers comprising a sample, a slightly perturbed path is constructed. Figure

6.9 shows the relation between mutations in primary sample space and path space. When applying the same perturbations to all random numbers in the list, the tentative transition function becomes symmetric, hence $\frac{p(\mathbf{X}|\mathbf{Y})}{p(\mathbf{Y}|\mathbf{X})} = 1$. This significantly simplifies the computation of the acceptance probability. Therefore, this mutation strategy can easily be applied to either PT or BDPT. The algorithm only requires a sampler that converts random number lists to paths in path space and evaluates their total contribution. This makes implementing features such as participating media relatively easy, as the mutation strategy itself does not require any changes.

Because paths may be of any length, each sample theoretically corresponds to an infinite list of numbers. To keep the handling of samples feasible, random values for the sample are generated lazy. That is, random values are only generated when necessary to construct the corresponding sample in path space. Hence, if a path is terminated early, only few random values are generated. Similar, the random values are mutated lazy, allowing for the storage of a finite number of random values per sample.

The Kelemen mutation only applies one of two mutation strategies: a large step mutation, which corresponds to generating a fresh list of random numbers, and a small step mutation, slightly perturbing each value in the list. The large step corresponds to generating an new independent path and therefore satisfies ergodicity. The small step allows for local exploration of path space. Kelemen showed that his mutation strategy often results in a higher average acceptance probability which leads to less variance in the estimator.

6.7 Noise Aware Metropolis Sampling

One of the characteristics of MLT is that samples are generated proportional to their contribution, corresponding to perfect importance sampling (see section 6.1). As a consequence, more samples are dedicated to bright pixels than to dark pixels. This results in a relatively high noise to signal ratio for dark pixels. Preferably, all samples contributing to the same pixel should be sampled proportional to their contribution, while the average number of samples per pixel remains the same for all pixels. Going even further, one would prefer to dedicate more samples to image regions with much noise.

The distribution of samples in MLT is determined by the importance function used in computing the acceptance probability from equation 6.7. Normally, the contribution $f(X)$ is used as the importance of a sample X. However, it is perfectly possible to use alternative importance functions instead. Hoberock explored the use of different importance functions, resulting in a noise aware MLT sampler [8]. In order to dedicate more samples to noisy image regions, the importance function is modified to favor these regions. To do so, the distribution of noise over the image plane needs to be estimated. The algorithm works in multiple stages, repeatedly adapting the importance function based on a progressive estimate of the noise distribution. Initially, all image pixels have equal importance, corresponding to normal MLT. Then a number of samples is generated and used to estimate the variance distribution for all pixels in a low resolution image. This estimate is then used to adapt the importance function, favoring pixels with higher variance. This process is repeated, each time increasing the resolution of the variance estimation image until it reaches the intended image resolution.

The use of this method result in a much better stratification of samples over the image plane, reducing visible noise. Note however that because the importance function is repeatedly

updated, new startup bias is introduced after each adaption. No attempts have been made to asses or quantify the impact of this increase in startup bias.

6.8 ERPT

In this section, we discuss the Energy Redistribution Path Tracing (ERPT) algorithm as proposed by Cline [3]. ERPT is an extension of MLT, solving some of its problems. First of all, ERPT does not suffer from startup bias. Also, ERPT does not require an accurate and unbiased estimate of the normalization constant c, although it can increase its performance. Furthermore, the mutation strategy used in ERPT does not need to satisfy the ergodicity constraint.

MLT suffers from startup bias because the initial path $\mathbf{X}_0$ for the mutation chain is not sampled proportional to f. The ERPT algorithm solves this problem and *does* sample $\mathbf{X}_0$ proportional to f. This is done by sampling multiple mutation chains per sample, all with equal length N. Path tracing is used to generate the initial path $\mathbf{X}_0$ for these mutation chains. Using PT, a path $\mathbf{X}_0$ is sampled with probability $p(\mathbf{X}_0)$, thus the probability of having $\mathbf{X}_0$ as the initial path of a chain is proportional to $p(\mathbf{X}_0)$. However, this number should be proportional to $f(\mathbf{X}_0)$. Hence, the number of mutation chains starting at $\mathbf{X}_0$ is off by a factor $\frac{f(\mathbf{X}_0)}{p(\mathbf{X}_0)}$. This is resolved by starting multiple chains per initial path $\mathbf{X}_0$. When on average, the number of chains for $\mathbf{X}_0$ is proportional to $\frac{f(\mathbf{X}_0)}{p(\mathbf{X}_0)}$, the initial mutation chain paths $\mathbf{X}_0$ are sampled proportional to f and startup bias is eliminated. This is realized by making the number of mutation chains $numChains(\mathbf{X}_0)$ per path $\mathbf{X}_0$ equal to:

$$numChains(\mathbf{X}_0) = \left\lfloor U(0,1) + \frac{1}{N \times e_d} \frac{f(\mathbf{X}_0)}{p(\mathbf{X}_0)} \right\rfloor \tag{6.12}$$

In this equation, $U(0,1)$ is a uniform random number between 0 and 1 and e_d is the amount of energy that is contributed to the image by each mutation in a chain, also called the algorithm's energy quantum.

To see why ERPT does not require an unbiased estimate of c, let us compute the expected number of contributions to the image plane for MLT and ERPT. For MLT, the total number of contributions to the image plane always equals N. For an ERPT sample with initial path $\mathbf{X}_0$, the average number of contributions equals $\frac{1}{e_d}\frac{f(\mathbf{X}_0)}{p(\mathbf{X}_0)}$. Therefore, the expected number of contributions per ERPT sample equals $\int_\Omega \frac{1}{e_d}\frac{f(\mathbf{X})}{p(\mathbf{X})} p(\mathbf{X}) d\Omega(\mathbf{X}) = \frac{1}{e_d}\int_\Omega f(\mathbf{X}) d\Omega(\mathbf{X}) = \frac{c}{e_d}$. Compared to MLT, the expected number of contributions per ERPT sample is off by a factor of $\frac{c}{e_d}\frac{1}{N}$. So, to keep the ERPT estimator unbiased, instead of contributing $\frac{c}{N}$ per mutation, as in MLT, an ERPT mutation should contribute $\frac{c}{N}\frac{e_d}{c}\frac{N}{1} = e_d$. Hence, the ERPT algorithm does not require an explicit estimate of c.

The energy quantum e_d may be chosen freely without introducing bias. However, it does influence its performance. The number of mutation chains per ERPT sample is inversely proportional to both N and e_d. The expected number M of mutation chains per ERPT sample can be regulated by using $e_d = \frac{c}{N \times M}$. This again requires an estimate of c. However, this estimate does not need to be very accurate and may even be biased, as it only influences the performance of the ERPT algorithm. In practice, the ERPT algorithm is not very sensitive to the value of e_d.

In ERPT, generating mutation chains for a PT sample is called energy redistribution. Algorithm 8 shows the energy redistribution part of an ERPT sampler. As input, it takes an initial PT path $\mathbf{X}_0$, the mutation chain length N and the energy quantum e_d.

Algorithm 8 : EnergyRedistribution($\mathbf{X}_0, N, e_d$)

$numChains \leftarrow \left\lfloor U(0,1) + \frac{f(\mathbf{X}_0)}{p(\mathbf{X}_0)} \frac{1}{N \times e_d} \right\rfloor$
for $i = 1$ to $numChains$ **do**
  $\mathbf{Y} \leftarrow \mathbf{X}_0$
  for j = 1 to N **do**
    $\mathbf{Z} \leftarrow$mutate($\mathbf{Y}$)
    $a \leftarrow a(\mathbf{Y} \rightarrow \mathbf{Z})$
    deposit ae_d energy at $\mathbf{Z}$
    deposit $(1-a)e_d$ energy at $\mathbf{Y}$
    if $a \geq U(0,1)$ **then**
      $\mathbf{Y} \leftarrow \mathbf{Z}$
    end if
  end for
end for

Finally, a useful property of the ERPT algorithm is that its mutation strategy does not necessarily need to satisfy ergodicity for the outcome to be unbiased. The reason is that mutation chains already have their initial samples distributed proportional to f. Note however that although the outcome will remain unbiased when using only a few ERPT samples, the error may become unacceptably large. Using many ERPT samples with relatively short mutation chains solves this problem.

Although N should not be chosen too large, N should not be chosen too small either. In ERPT the value of N determines the amount of energy redistribution. For effective energy redistribution, N should be reasonably large. If N is too small, ERPT effectively deteriorates into a quantized PT algorithm. To see why, let us look at the extreme case without any energy redistribution; each mutation chain only contains its initial sample. The number of chains per initial sample is proportional to f, so each sample will contribute on average $e_d \left\lfloor U(0,1) + \frac{1}{e_d} \frac{f(\mathbf{X}_0)}{p(\mathbf{X}_0)} \right\rfloor = \frac{f(\mathbf{X}_0)}{p(\mathbf{X}_0)}$. All this energy is contributed to a single pixel; the pixel corresponding to $\mathbf{X}_0$. Hence, because each chain contributes a single energy quantum, the remaining algorithm is a quantized PT algorithm, performing worse than normal PT.

The ERPT algorithm presented in [3] uses only the lens and caustic mutation types from the MLT algorithm and an N in the order of 100. Note that because lens and caustic mutations do not change the length or signature of a path, all paths in a mutation chain have the same length and signature.

6.9 Population Monte Carlo

Usually, mutation strategies in MLT and ERPT require magic numbers to determine the perturbation size distribution. It is however not obvious which numbers work best. Lai and Fan proposed to use the population Monte Carlo framework to adapt these magic numbers, based on information gathered so far during sampling [28, 5]. Instead of using only a single perturbation size distribution, they used a mixture of multiple perturbation size distributions

of various sizes. Each time a mutation is applied, one of these mixture components is chosen at random and perturbation sizes are sampled accordingly. Initially, all mixture components are selected with equal probability, but as more samples are generated, the probability for selecting each mixture component for a certain image pixel is adapting according to its success so far.

Initially, a large population of initial samples is generated. All samples in the population are mutated some fixed number of time. During each mutation, a mixture component is selected for each sample and the mutation is applied. Within each pixel, the acceptance probabilities for all mutations contributing to this pixel are accumulated for each mixture component. After all mutations are complete, the probability of selecting a mixture component is adapted, favoring mixture components with high accumulated acceptance. Finally, low contribution samples are eliminated from the population and new initial samples are regenerated to keep the population size constant. This process of sampling, adapting, elimination and regeneration is repeated until the image has converged.

Population Monte Carlo automatically favors perturbation sizes with higher acceptance probabilities. This removes the need for very accurate magic numbers selection.

Chapter 7

Conclusion

In this survey, we studied the applications of importance sampling in the field of unbiased physically based rendering. Importance sampling has played an important role in the development of this field, resulting in efficient and robust rendering algorithms. We discussed the three major unbiased rendering algorithms and how they relate to importance sampling. We further discussed several important extensions to these algorithms, based on importance sampling, giving an overview of the various applications of importance sampling in the context of unbiased physically based rendering.

The applications of importance sampling each have their own advantages and disadvantages. The variance in the final estimate of a method depends heavily on the particular scene that is being rendered. An important question that is left unanswered is how to select the right sampling method for a particular scene. Optimally combining different sampling strategies using MIS already partially solves this problem. Furthermore, from the importance sampling point of view, metropolis sampling seems to be the best choice as it achieves *perfect* importance sampling. However, although MLT samples proportional to f, the variance in the estimate can still be arbitrary high. The cause of this variance lies in the correlation between consecutive mutated samples in the Markov Chain. When the correlation between samples is high, the estimate will suffer from high variance as well. Because this correlation cannot easily be computed, it is hard to predict how well MLT will perform for particular scenes and light effects. For this reason, MLT does not necessarily lead to a better estimate than other sampling methods such as PT and BDPT. The quality of the MLT estimator for sampling certain light effects heavily depends on the particular mutation strategy used. Furthermore, because MLT samples are highly correlated, combining MLT with PT or BDPT using multiple importance sampling does not necessarily lead to an optimal combination either. Further research is required to better understand the quality of the MLT algorithm and how they relate to other sampling methods and the particular mutation strategy used in MLT.

Although importance sampling is a major variance reduction technique, other techniques such as stratification and quasi-Monte Carlo sampling have been shown to reduce variance in the Monte Carlo estimate [25]. However, it is often not obvious how to combine these techniques with advanced importance sampling methods such as BDPT or MLT. For example, while quasi-Monte Carlo sampling can be applied to path tracing, often significantly reducing the variance in the estimate, it is not so easily applied to MLT without introducing a large and unpredictable bias. Therefore, further research is required to combine alternative variance reduction techniques with importance sampling methods.

Bibliography

[1] Philippe Bekaert, Mateu Sbert, and John Halton. Accelerating path tracing by re-using paths. In *Proceedings of the 13th Eurographics workshop on Rendering*, EGRW '02, pages 125–134, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[2] David Cline. A practical introduction to metropolis light transport. Technical report, 2005.

[3] David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tracing. In *Proceedings of ACM SIGGRAPH 2005*, SIGGRAPH '05, pages 1186–1195, New York, NY, USA, 2005. ACM.

[4] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *SIGGRAPH Comput. Graph.*, 18:137–145, January 1984.

[5] Shaohua Fan. *Sequential Monte Carlo Methods for Physically Based Rendering*. PhD thesis, University of Wisconsin - Madison, Madison, WI, 2006.

[6] W.K. Hastings. Monte carlo samping methods using markov chains and their applications. *Biometrika*, pages 97–109, 1970.

[7] Paul S. Heckbert. Discontinuity meshing for radiosity. In *Third Eurographics Workshop on Rendering*, pages 203–226, 1992.

[8] Jared Hoberock and John C. Hart. Arbitrary importance functions for metropolis light transport. *Comput. Graph. Forum*, 29(6):1993–2003, 2010.

[9] Henrik Wann Jensen. Importance driven path tracing using the photon map. In *Proceedings of Eurographics Rendering Workshop*, pages 326–335. Springer-Verlag, 1995.

[10] James T. Kajiya. The rendering equation. In *Computer Graphics*, pages 143–150, 1986.

[11] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Proceedings of Computer Graphics Forum*, pages 531–540, 2002.

[12] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of third international conference on computational graphics and visualization techniques (COMPUGRAPHICS 93)*, pages 145–153, 1993.

[13] Eric P. Lafortune and Yves D. Willems. Rendering participating media with bidirectional path tracing. In *Proceedings of Eurographics Rendering Workshop*, pages 91–100. Springer-Verlag/Wien, 1996.

[14] Péter Dornbach László Szirmay-Kalos and Werner Purgathofer. On the start-up bias problem of metropolis sampling.

[15] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[16] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44:335–341, 1949.

[17] Fred E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Appl. Opt.*, 4(7):767–773, Jul 1965.

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

[19] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte carlo techniques for direct lighting calculations. *ACM Trans. Graph.*, 15:1–36, January 1996.

[20] Justin F. Talbot. Importance resampling for global illumination. Master's thesis, Brigham Young University, 2005.

[21] Justin F. Talbot, David Cline, and Parris K. Egbert. Importance resampling for global illumination. In Kavita Bala and Philip Dutré, editors, *Rendering Techniques 2005 Eurographics Symposium on Rendering*, pages 139–146, Aire-la-Ville, Switzerland, 2005. Eurographics Association.

[22] Eric Veach. Robust monte carlo methods for light transport simulation. 1998. Adviser-Guibas, Leonidas J.

[23] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering, 1995.

[24] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of ACM SIGGRAPH 1997*, pages 65–76. Addison Wesley, 1997.

[25] Carsten Alexander Wächter. *Quasi-Monte Carlo Light Transport Simulation by Efficient Ray Tracing*. PhD thesis, Institut für Medieninformatik, Ochsenhausen, Germany, 2007.

[26] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23:343–349, June 1980.

[27] L. Zhang Y.-C. Lai, F. Liu and C. R. Dyer. Efficient schemes for monte carlo markov chain algorithms in global illumination. In *Proceedings of 4th International Symposium on Visual Computing*, 2008.

[28] S. Chenney Y.-C. Lai, S. Fan and C. R. Dyer. Photorealistic image rendering with population monte carlo energy redistribution. In *Proceedings of Eurographics Symposium on Rendering*, pages 287–296, 2007.