Bidirectional Path Tracing

Mikkel Adamsen <adamsen@diku.dk>

July 19, 2009

Abstract

In this report we present the bidirectional path tracing method. Bidirectional path tracing combines the ideas of shooting and gathering light to create photorealistic images. We discuss, how the well known path tracing algorithm and the light tracing algortihm are subsets of the bidirectional path tracing algorithm. Finally we will show tests that shows how bidirectional path tracing can decrease the noise, in rendered images, in curtain scenes.

Contents

1 Introduction

The goal of global illumination algorithms is to render realistic looking images. A popular method for solving the global illumination problem, is path tracing. By shooting out rays, from the eye, into the scene, we can find out how much light reaches the eye. Unfortunately some scenes, are really slow to render with path tracing. This is especially, scenes that contains caustics. Another method called light tracing shoots out rays from the light source. As it turns out, light tracing is very good at rendering caustics, but not at much else. In bidirectional path tracing we combine the two methods, by shooting out rays from both the eye and the light source at the same time.

Since we have a limited space, we will assume that the reader has some of the basic knowledge about path tracing. We will therefore only dicuss the theory needed to create the most basic version of a bidirectional path tracer. We will therefore not discuss subjects such as stratified sampling, importance sampling, russian roullette, subsurface scattering, paticipating media and dispersion, or discuss how bidirectional path tracing can be used with those subjects.

Futhermore will limit ourself to scenes that only contains a single light source, and we model the eye by using a pin hole camera.

2 Related Work

At the base of any advanced global illumination algorithm, is the raytracing method. By shooting out rays from the eye into the scene we can get the color of the object, nearest to the eye. In [Whitted, 1980] Whitted extended raytracing with a recursive raytracing method. This was an exiting new method that could easily do advanced effect such as reflective and refractive materials. However, it relies on a local illumination model, meaning that it can't do global illumination effects such as indirect illumination.

In [Goral et al., 1984] the radiosity method was introduced. With this method, they could do indirect illumination by dividing the scene into patches and distributing energy between the patches. Since they save the energy in each path, the method is view independent. The radiosity method requires that the surfaces has diffuse materials and it can therefore not do the reflection effects that can be archived with raytracing.

In [Cook et al., 1984] they extended the raytracing method with a stocastic sampling technique. By shooting out multiple rays per pixel they could archive effects such as motion blur, depth of field and soft shadows. Path tracing was introduced in [Kajiya, 1986], where Kajiya realised that the stocastic sampling technique could be used to solve the rendering equation.

New methods started to pop up that, instead of shooting rays out of the eye, would shoot out rays from the light source. One of the first methods was introduced in [Arvo, 1986] where rays are shoot out of the light source and stoored in an illumination map. A second pass uses the illumination map to create the final image. A similar method was introduced in [Chen et al., 1991] where they start out by creating a caustic map and uses this in the second pass combined with path tracing. The most popular two-pass method today is photon mapping [jen, 2001]. In the first pass a photon map is created by shooting out a number of photons from the light source. The photons then bounces through zero or more specular reflections. Whenever the photon hits a diffuse surface it's energy is stoored in the photon map. The photon map is the used in the second pass with path tracing. Theese two-pass methods are however biased.

In [Dutre et al., 1993], they created an unbiased version of light tracing using the potential equation introduced in [Pattanaik and Mudur, 1992]. This led to an unbiased version of the bidirectional path tracing algorithm [Lafortune and Willems, 1993], that traces rays from both the eye and the light source.

In [Veach and Guibas, 1994] they independently created bidirectional path tracing and in [Veach and Guibas, 1995] they improved it with multiple importance sampling. A way to futher improve the bidirectional path tracing method, is to use metropolis light transport, which was introduced in [Veach and Guibas, 1997]. Instead of constantly creating new paths from scratch, paths are mutated into new ones. If the mutation is bad it is thrown away. In this way the renderer will spend most of the time on interesting paths.

3 A Mathematical Framework for Global Illumination

In this section, we will describe the basic terms and definitions used in global illumination. Since we are trying to simulate how light behaves in the real world, we need radiometry to describe the units of light. In the last part we will derive some mathematical equations that can describe how light is transported from the light source to the eye.

3.1 Radiometry

Global illumination is the simulation of how light behave in the real world. We will however use some limitations. We will assume that the light moves in straight lines and that it moves at infite speed. Even with these limitations we can still simulate almost all the lighting effects that is visible to the eye.

3.1.1 Flux

Flux, often denoted as Φ, is the flow of energy and is measured in Watt. We can think of flux as the amount of energy that passes though an area in a given time. We can write flux as:

$$\Phi = \frac{dQ}{dt},$$

where dQ is a unit of enerqy and dt is a unit of time. Since flux is independent of time, flux is the basic unit of many global illumination algorithms.

3.1.2 Irradiance and Radiosity

Another measure that is useful, is the amount of flux that is arriving or departing per unit of surface area, dA. When the flux is arriving at the surface area, it is called irradiance, E. When the flux is leaving the surface, it is called radiosity, B. Radiosity is sometimes also know as radiant exitance, M. These can be written as:

$$E = \frac{d\Phi}{dA}, \qquad B = M = \frac{d\Phi}{dA}.$$

3.1.3 Radiant Intensity

We can also measure radiant flux per unit solid angle, $d\omega$. To describe this measurement, we use the term intensity, I. It can be written as:

$$I = \frac{d\Phi}{d\omega}.$$

3.1.4 Radiance

Radiance is the primary way of describing the light in global illumination algorithms. This is because radiance is the value that mostly ressembles color. Another important fact is that radiance does not change over distant, which makes it easy to work with. We can say that radiance equals the amount of radiant flux found in a single light ray. To find the flux of a single light ray, we have to find the amount of flux that exits or enters a single point on a

surface, from or in a single direction. This equals the flux per area per solid angle and can be written as:

$$L(\mathbf{x}, \omega) = \frac{d^2\Phi}{d\omega \; dA \; |\omega \cdot N_{\mathbf{x}}|}$$

Where $L(\mathbf{x}, \omega)$ is the radiance of point x in the direction ω and $N_{\mathbf{x}}$ is the normal at point $\mathbf{x}$. We can also go the other way and find the flux, if we know the radiance, by integrating over the surface area A and all the directions,Ω, in the hemisphere above the point $\mathbf{x}$:

$$\Phi = \int_A \int_\Omega L(\mathbf{x}, \omega) |\omega \cdot N_{\mathbf{x}}| \; d\omega \; dx.$$

3.1.5 Potential

The potential captures the flow of flux. If we have two patches i and j, then we can describe the amount of flux going from i to j as the potential. We write the potential as W. Potential does not have a unit of measure such as radiance has. We should instead, think of potential more as a value that scales how much light goes from i to j. This scale value should be between 0 and 1. The most simple example is the emitted potential, W_e. For example if we want to find out how much of the light emitted from a lightsource hits a pixel on the viewport, of a camera, we can use $W_e(\mathbf{x}, \omega)$, where $\mathbf{x}$ is an point on the viewport and ω is the direction from $\mathbf{x}$ to a point on the lightsource. If the ray hits the pixel, then W_e should return 1 since, all

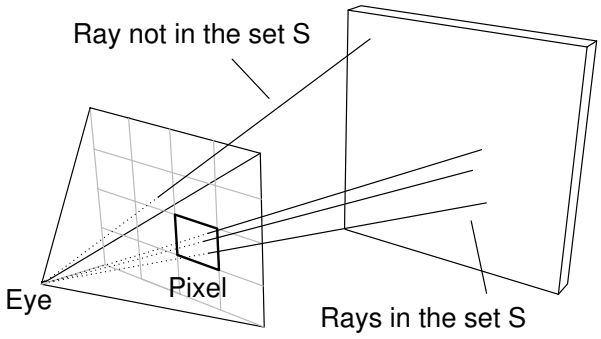


Figure 1: S is a set of rays that hit the eye through a pixel.

the light will contribute to the measurement of the pixel. W_e should return 0 otherwise. We can therefore define W_e as:

$$W_e(\mathbf{x}, \omega) = \begin{cases} 1 & \text{if } (\mathbf{x}, \omega) \in S \\ 0 & \text{if } (\mathbf{x}, \omega) \notin S \end{cases},$$

Where S is the set of rays that hits the pixel (see figure 1).

3.2 The Rendering Equation

The rendering equation can be used to compute the outgoing radiance from any surface point in the scene. The amount of outgoing radiance, $L_o(\mathbf{x}, \omega_o)$, from the point $\mathbf{x}$ in the direction ω_o, can be computed as the emitted radiance plus the reflected radiance:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + L_r(\mathbf{x}, \omega_o). \tag{1}$$

$L_e(\mathbf{x}, \omega_o)$ is the emitted radiance from point $\mathbf{x}$ in the direction ω_o. Only light sources has any emitted light, for any other surface the emitted radiance will be zero. $L_r(\mathbf{x}, \omega_o)$ is all of the incomming light, in a point $\mathbf{x}$, reflected in the direction ω_o. We can find the incomming radiance for a single direction, ω_i, as

$$L_i(\mathbf{x}, \omega_i).$$

To find out how much of the incomming radiance that is reflected, we multiply L_i by a factor that is always between 0 and 1. This factor is called a Bidirectional reflectance distribution function (BRDF). The BRDF determines the material of the surface point and is defined as:

$$f_r(\mathbf{x}, \omega_i, \omega_o),$$

where $\mathbf{x}$ is the surface point, ω_i is the incomming direction of the radiance and ω_o is the outgoing direction. By multiplying the BRDF with L_i we can compute how much of the

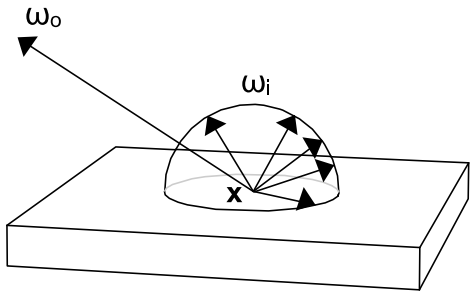


Figure 2: The light reflected on the point $\mathbf{x}$, in the direction ω_o, can come from all the directions of the hemisphere above the point $\mathbf{x}$. Only a small fraction of the directions have been shown by arrows here.

radiance from the direction, ω_i is reflected in the direction, ω_o. To find the combined amount of radiance that is reflected, we have to look at the radiance that comes from all directions, (see figure 2). We can do that by integration over all directions on the hemisphere above the point $\mathbf{x}$. This gives us:

$$L_r(\mathbf{x}, \omega_o) = \int_\Omega L_i(\mathbf{x}, \omega_i) f_r(\mathbf{x}, \omega_i, \omega_o) |N_\mathbf{x} \cdot \omega_i| \, d\omega_i \tag{2}$$

If we combine equation (1) and (2) we get the rendering equation:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega L_o(\mathbf{x}', -\omega_i) f_r(\mathbf{x}, \omega_i, \omega_o) |N_\mathbf{x} \cdot \omega_i| \, d\omega_i.$$

We have changed it from an integration over L_i to an integration over L_o, where $\mathbf{x}'$ is the point we find when we trace in the direction ω_i. Since both radiance terms are now L_o, we drop the subscript from now on.

3.3 The Potential Equation

The potential equation tells us how much potential, the light emitted from a light source, has of being received by a single pixel on the viewport of the camera. We can write the potential as the emitted potential, plus the reflected potential:

$$W(\mathbf{x}, \omega_o) = W_e(\mathbf{x}, \omega_o) + W_r(\mathbf{x}, \omega_o). \tag{3}$$

$W_e(\mathbf{x}, \omega_o)$ is the emitted potential, or the potential of a light ray going directly from the light source to the pixel. If the light ray does not hit the pixel directly, it can hit another surface point in the scene. Since light can be reflected on a surface, we can say that the potential also can be reflected, we denote this as $W_r(\mathbf{x}, \omega_o)$. The reflected ray could hit the pixel, and it would increase the potential of hitting the pixel.

When a ray hits a surface the ray can be reflected in all directions. To find out how much of the potential that is going in the direction ω_o we have to scale the potential, with the BRDF, as we did for radiance in the rendering equation. To find the combined potential in a point we have to integrate over all directions and we get that W_r is defined as:

$$W_r(\mathbf{x}, \omega_o) = \int_\Omega f_r(\mathbf{x}, \omega_o, \omega_i) W(\mathbf{x}, \omega_i) |N_\mathbf{x} \cdot \omega_i| \, d\omega. \tag{4}$$

Equation (3) and (4) can then be combined into the potential equation, as follows:

$$W(\mathbf{x}, \omega_o) = W_e(\mathbf{x}, \omega_o) + \int_\Omega f_r(\mathbf{x}, \omega_o, \omega_i) W(\mathbf{x}', -\omega_i) |N_\mathbf{x} \cdot \omega_i| \, d\omega. \tag{5}$$

4 Path Integral

The light transport problem is usually expressed in the form of an integral equation. In the integral equation method we begin in a point, such as the eye point. We then recursivly find the next point, by shooting out a ray into the scene. This means that we can only find the next point by locally evaluating our current point. Here we will express the light transport problem as an integral over all light paths in a scene. A path is defined as the eye point, a series of reflections and a point on a light source. The path integral method gives a more global view of the problem compared to the integral equation method. By using the path integral we will see that as long as we have two points in the scene we can connect those two points by creating an edge between them. It is however necessary to do a visibilty check to see if the two points are mutually visible to each other.

4.1 The Three Point Form

We start by transforming the rendering equation, which uses an integral over solid angle, into the light transport equation which uses an integral over surface area. In the rendering equation we find the incomming radiance, on a point $\mathbf{x}'$, by integrating over all the incomming directions. Another way of finding the incomming radiance is to integrate over all the surface points in the scene. To do this we have to make the rendering equation independent of directions ω_i and ω_o by making it take points instead of directions, as arguments. We start by finding the outgoing radiance from point $\mathbf{x}'$ to point $\mathbf{x}$ (see figure 3), by rewriting the radiance term as:

$$L(\mathbf{x}' \to \mathbf{x}) = L(\mathbf{x}', \omega_o),$$

where $\mathbf{x}, \mathbf{x}'$ are points in the scene and $\omega_o = \overrightarrow{\mathbf{x}'\mathbf{x}}$. Here we use the overhead arrow to describe a direction going from $\mathbf{x}'$ to $\mathbf{x}$. Next we look at the BRDF, which can simply be written as:

$$f_r(\mathbf{x}'' \to \mathbf{x}' \to \mathbf{x}) = f_r(\mathbf{x}', \omega_i, \omega_o),$$

where $\omega_i = \overrightarrow{\mathbf{x}'\mathbf{x}''}$ and $\omega_o = \overrightarrow{\mathbf{x}'\mathbf{x}}$. To complete the transformation, we have to integrate over the

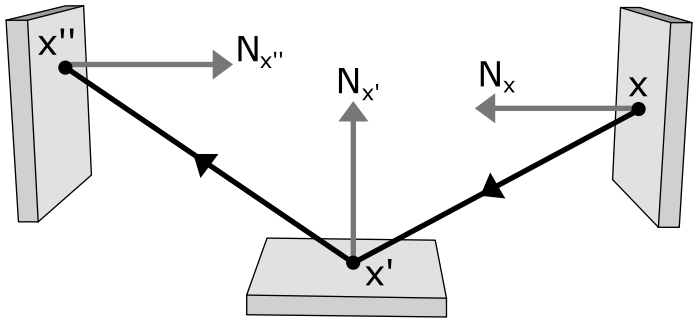


Figure 3: The light comes from $\mathbf{x}''$ and is reflected in point $\mathbf{x}'$, towards point $\mathbf{x}$.

combined area of all the surfaces, A, instead of the solid angle, Ω. The relationship between unit area and the unit solid angle is given as:

$$d\omega = \frac{dA \ |N_{\mathbf{x}''} \cdot \overrightarrow{\mathbf{x}''\mathbf{x}'}|}{\|\mathbf{x}'' - \mathbf{x}'\|^2}.$$

We can now write the light transport equation as:

$$L(\mathbf{x}' \to \mathbf{x}) = L_e(\mathbf{x}' \to \mathbf{x}) + \int_A L(\mathbf{x}'' \to \mathbf{x}') f_r(\mathbf{x}'' \to \mathbf{x}' \to \mathbf{x}) G(\mathbf{x}'' \leftrightarrow \mathbf{x}') dA(\mathbf{x}''). \tag{6}$$

To make the equation a little more compact we use the geometric coupling term, $G(x \leftrightarrow x')$, which is defined as:

$$G(\mathbf{x}'' \leftrightarrow \mathbf{x}') = V(\mathbf{x}'' \leftrightarrow \mathbf{x}') \frac{|N_{\mathbf{x}''} \cdot \overrightarrow{\mathbf{x}''\mathbf{x}'}| \ |N_{\mathbf{x}'} \cdot \overrightarrow{\mathbf{x}'\mathbf{x}''}|}{\|\mathbf{x}'' - \mathbf{x}'\|^2}.$$

$V(\mathbf{x}'' \leftrightarrow \mathbf{x}')$ is the visibility function, $V(\mathbf{x}'' \leftrightarrow \mathbf{x}') = 1$ if we can shoot a ray from $\mathbf{x}''$ to $\mathbf{x}'$ without hitting anything along the ray, and $V(\mathbf{x}'' \leftrightarrow \mathbf{x}') = 0$ otherwise.

4.2 The Neumann Serie Expansion

An equation with the same value on both sides of the equation cannot be directly evaluated. An example is the following equation:

$$L = f(x) + \int_A L \ dA. \tag{7}$$

We can evaluate the equation by recursively replacing the value on the right with itself. If we do this to equation (7) we get:

$$\begin{aligned} L &= f(x) + \int_A L \ dA \\ &= f(x) + \int_A f(x) dA + \int_A \int_A L \ dA dA \\ &= f(x) + \int_A f(x) dA + \int_A \int_A f(x) dA dA + \int_A \int_A \int_A L \ dA dA dA \\ &= ... \end{aligned}$$

This is called the Neumann Series Expansion. We can write it more compactly as:

$$L = \sum_{m=0}^{\infty} \underbrace{\int_A ... \int_A}_{m} f(x) \underbrace{dA ... dA}_{m}. \tag{8}$$

4.3 The Path Integral

We can now expand the light transport equation (6), using the Neumann series expansion (8). The result is an infinte sum. Here we will only show a few of the first terms:

$$\begin{aligned} L(\mathbf{x}_1 \to \mathbf{x}_0) = \quad & L_e(\mathbf{x}_1 \to \mathbf{x}_0) + \textstyle\int_A f_r(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0) L_e(\mathbf{x}_2 \to \mathbf{x}_1) G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_1) dA(\mathbf{x}_2) \\ & + \textstyle\int_A \int_A L_e(\mathbf{x}_3 \to \mathbf{x}_2) f_r(\mathbf{x}_3 \to \mathbf{x}_2 \to \mathbf{x}_1) G(\mathbf{x}_3 \leftrightarrow \mathbf{x}_2) \\ & f_r(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0) G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_1) dA(\mathbf{x}_3) dA(\mathbf{x}_2) + ... \end{aligned} \tag{9}$$

Intuitivly we can see it as the summation of the radiance found in all the paths of length 1, 2, 3 and so on. As an example of a path, we can see a path with length 3 in figure 4. Each

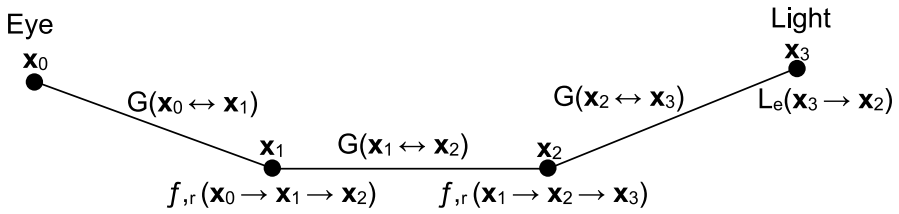


Figure 4: A path with a path length of three. The path starts at the eye point, $\mathbf{x}_0$, it is the reflected through point $\mathbf{x}_1$ and $\mathbf{x}_2$ up to a point on the light source, $\mathbf{x}_3$.

path contains a point on the eye and a point on the light source and a series of points where the light is reflected. We can write equation (9) more compactly as:

$$L(\mathbf{x}_1 \rightarrow \mathbf{x}_0) = \sum_{i=1}^{\infty} P(\overline{p}_i).$$

Where $\overline{p}_i$ is a path:

$$\overline{p}_i = \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_i,$$

where $\mathbf{x}_0$ is the eye and $\mathbf{x}_i$ is a point on the light source. $P(\overline{p}_i)$ gives us the amount of radiance on path $\overline{p}_i$:

$$P(\overline{p}_i) = \underbrace{\int_A ... \int_A}_{i-1} L_e(\mathbf{x}_i \rightarrow \mathbf{x}_{i-1}) \left(\prod_{j=1}^{i-1} f_r(\mathbf{x}_{j+1} \rightarrow \mathbf{x}_j \rightarrow \mathbf{x}_{j-1}) G(x_{j+1} \leftrightarrow \mathbf{x}_j) \right) dA(\mathbf{x}_2)...dA(\mathbf{x}_i). \tag{10}$$

4.4 Monte Carlo Methods Applied to the Path Integral

The overall problem we are trying to solve, is to find the radiance, L_p, that flows through a pixel:

$$L_p = \int_D f_j(\overline{p}) dD \tag{11}$$

Where D is all the possible light paths in the scene. $\overline{p}$ is a single ligth path and f_j is the measurement contribution function, that gives us the measurement of a single path. f_j can be found by using the integrand of equation (10). For example if we have the path $\overline{p} = x_0, x_1, x_2, x_3$, we get:

$$\begin{aligned} f_j = {} & L_e(\mathbf{x}_3 \rightarrow \mathbf{x}_2) G(\mathbf{x}_3 \leftrightarrow \mathbf{x}_2) fr(\mathbf{x}_3 \rightarrow \mathbf{x}_2 \rightarrow \mathbf{x}_1) G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_1) \\ & fr(\mathbf{x}_2 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_0) G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_0) W_e(\mathbf{x}_1 \rightarrow \mathbf{x}_0) \end{aligned}$$

With monte carlo integration we can approximate equation (11), by taking the average of N randomly sampled paths, giving us:

$$L_p = \frac{1}{N} \sum_{i=1}^{N} \frac{f_j(\overline{p_i})}{p(\overline{p_i})}$$

Where $p(\overline{p_i})$ is the probability distribution function (PDF) of sampling the path $\overline{p_i}$. To find the PDF of a path we have look at the way the path was created. If we use path tracing, then we start at eye and selects the next points by a random process until we select a point on the light source. The PDF depends on all the choises made to create the path.

When we get a PDF, it is usually given in respect to solid angle, $p(\omega)$, this is often the case when we use a BRDF to sample a new direction. To find the PDF with respect to surface area, $p(\mathbf{x})$, we can use a conversion given in [Veach, 1997]:

$$p(\mathbf{x}) = p(\omega_o) \left(\frac{|N_{\mathbf{x}'} \cdot \overrightarrow{\mathbf{x}'\mathbf{x}''}|}{\|\mathbf{x}'' - \mathbf{x}'\|^2} \right)$$

4.5 Path Sampling

To evaluate a path, we must first create the path. To create a path there are several differrent methods. We will here look a familiy of methods for creating paths, called local path sampling algorithms. The methods are called local since they generate the next point, based on local information in the current point.

- The first method is to sample a point on a surface. The surface could be a light source, the lens of the eye or it could be a point on any surface in the scene.
- The second method is to sample a direction according to a PDF, for example using the local BRDF of the current point. We then cast a ray in the found direction to find the next point. This is the same method as used in classical path tracing.
- The last method is to combine two already existing points to create an edge between them. To find out if this is an valid edge, there need to be visibilty check to see if the edge intesects any other objects in the scene.

By combining these techniques, we can create paths in a large number of scenes. Starting either at the light source or at the eye. Then using the second method to create new sub paths. We could also start a path in both ends and use the third method to combine the two subpaths.

5 Path Tracing

In path tracing, we solve the rendering equation by using monte carlo integration. This means that instead of integrating over the entire hemisphere of directions, we sample the hemisphere to get a single direction, ω_i. We then let the reflected radiance, from ω_i, approximate the entire amount of reflected radiance by dividing it with the pdf of sampling ω_i. For example if we have a pdf that is sampled according to a uniform distribution over the hemisphere, the pdf equals $1/2\pi$. This actually corresponds to multiplying the reflected radiance with 2π which is the solid angle of the hemisphere. The rendering equation, estimated with monte carlo integration is given as:

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \frac{L(\mathbf{x}', -\omega_i) f_r(\mathbf{x}, \omega_i, \omega_o) |N_\mathbf{x} \cdot \omega_i|}{p(\omega_i)} \tag{12}$$

We can evaluate equation (12) in a recursive manner. By starting at the eye, we trace a ray in the scene to find the nearest point, $\mathbf{x}$ (see figure 5). We then sample a new direction, ω_i on the hemisphere above $\mathbf{x}$. We use the direction ω_i to trace a new ray and we find the point $\mathbf{x}'$. We can now evaluate $L(\mathbf{x}', \omega_i)$ in a recursive manner using equation (12). We continue this until we hit some maximum path length or we could use russian roulette.

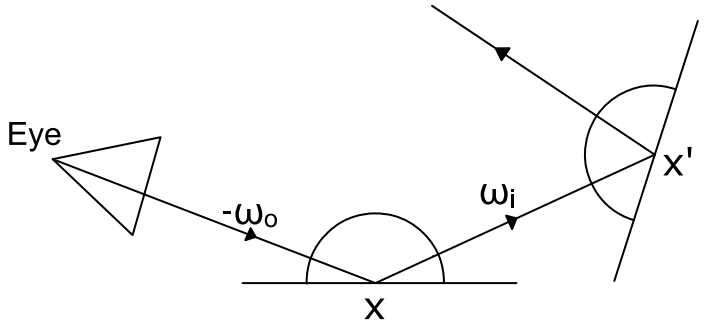


Figure 5: This shows an example of path tracing. The arrows show the direction we are tracing in. We start by finding the point $\mathbf{x}$, by tracing a ray from the eye point through a pixel. We then sample a new direction ω_i from the hemisphere above $\mathbf{x}$. By tracing in the direction ω_i, we find the next point $\mathbf{x}'$.

5.1 Next Event Estimation

The problem with our path tracing method, is that many of the paths will never hit any light source. Their contribution will therfore be zero. This is particularly a problem, if we have a small light source, compared to the scene, since it will be harder to hit and it will create a lot of noise in the final image. If we use a point light source the problem is even worse, since the point light is an infinitly small point. The chance of a ray hitting a point light source, is so small that we say it is zero. Which makes it impossible to use light point sources with our current path tracing method. Instead we can use next event estimation. We start with the outgoing radiance, L_o, which is the sum of the emitted light, L_e and the reflected light, L_r:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + L_r(\mathbf{x}, \omega_o). \tag{13}$$

The reflected light is given as:

$$L_r(\mathbf{x}, \omega_o) = \int_\Omega L(\mathbf{x}', -\omega_i) f_r(\mathbf{x}, \omega_i, \omega_o) |N_\mathbf{x} \cdot \omega_i| \; d\omega_i \tag{14}$$

The radiance comming from x' can be found by $L(\mathbf{x}', -\omega_i)$. We can estimate the next event by inserting equation (13) for $(\mathbf{x}', -\omega_i)$ into equation (14). This gives us:

$$\begin{aligned}
L_r(\mathbf{x}, \omega_o) &= \int_\Omega \left[L_e(\mathbf{x}', -\omega_i) + L_r(\mathbf{x}', -\omega_i) \right] f_r(\mathbf{x}, \omega_i, \omega_o) |N_\mathbf{x} \cdot \omega_i| \; d\omega_i \\
&= \int_\Omega L_e(\mathbf{x}', -\omega_i) f_r(\mathbf{x}, \omega_i, \omega_o) |N_\mathbf{x} \cdot \omega_i| \; d\omega_i + \int_\Omega L_r(\mathbf{x}', -\omega_i) f_r(\mathbf{x}, \omega_i, \omega_o) |N_\mathbf{x} \cdot \omega_i| \; d\omega_i \\
&= \int_A L_e(\mathbf{x}' \to \mathbf{x}) f_r(\mathbf{x}, \overrightarrow{\mathbf{x}'\mathbf{x}}, \omega_o) G(\mathbf{x} \leftrightarrow \mathbf{x}') \; dA_i + \int_\Omega L_r(\mathbf{x}', -\omega_i) f_r(\mathbf{x}, \omega_i, \omega_o) |N_\mathbf{x} \cdot \omega_i| \; d\omega_i \\
&= L_{direct} + L_{indirect}
\end{aligned} \tag{15}$$

We have now divided the reflected radiance into the direct radiance and the indirect radiance. This method is also know as direct lighting. To get a more intuitive view of how direc lighting works, we can look at figure 6. We can see how some of the directions, the ones going through the grey area, goes towards the lightsource. We divide the hemisphere up, so that we have the direct radiance, that we get from the directions that goes towards the light source, and the indirect radiance, that we get from the entire hemi sphere. To find the direct radiance we

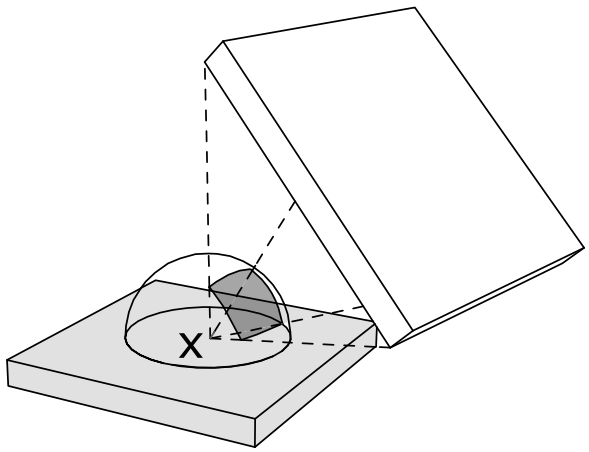


Figure 6: The hemisphere represent all the directions above a point on the surface. The dark grey area is the lightsource projected down, onto the hemisphere, represents all the directions that will hit the lightsource.

have to integrate over all the directions that goes through the grey area. This is actually a very hard problem, but it turns out that we can get the same result if we integrate over the surface of the light source, which is alot easier. The change from an integral over directions to an integral over surface area, is done in the same way as in section 4.1.

We find the indirect radiance by integrating over the entire hemisphere. This means that directions going towards the light source will contribute twice. To awoid this, we have to be carefull not to include the emitted light when we are evaluating the indirect light.

5.2 Applying the Path Integral

We start by solving equation (15) with monte carlo integration, and we get:

$$L_p = \frac{1}{N}\sum_{i=0}^{N} L_{p_i},$$

where L_p is the radiance measurement of a pixel. N is the amount of samples taken for each pixel. L_{p_i} is the radiance of a single sample. By using the path integral on L_{p_i} we get:

$$L_{p_i} = \sum_{s=0}^{N_s} C_s,$$

Where C_s is the contribution of a single path with the length s and N_s is the maximum path length. It is important to notice that, to get a single sample, we have to sum together the paths with different path length. We will apply monte carlo integration to the contribution, C_s, this give us:

$$\begin{aligned} C_s &= \frac{L_e(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{x}_s})}{p(\mathbf{y}_0)} f_r(\mathbf{x}_s, \overrightarrow{\mathbf{x}_s\mathbf{y}_0}, \overrightarrow{\mathbf{x}_s\mathbf{x}_{s-1}}) G(\mathbf{y}_0 \leftrightarrow \mathbf{x}_s) \\ &\left(\prod_{i=1}^{s-1} \frac{f_r(\mathbf{x}_i, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}) |N_{\mathbf{x}_i} \cdot \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}|}{p(\overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}})}\right). \end{aligned} \tag{16}$$

Here we estimate a path $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_s, \mathbf{y}_0$, where $\mathbf{y}_0$ is the point we have selected on the light source. $p(\mathbf{y}_0)$ is the PDF of selecting $\mathbf{y}_0$, since we can select any point on the area of the light source, A_{light}, we get that

$$p(\mathbf{y}_0) = \frac{1}{A_{light}}$$

The first part of equation (16) is the direct lighting from $\mathbf{y}_0$ that we receive in $\mathbf{x}_s$. $L_e(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{x}_s})$ is the amount of radiance from point $\mathbf{y}_0$ going in the direction towards $\mathbf{x}_s$. This is multiplied to the geometric term, that scales how much of the light is received at point $\mathbf{x}_s$. We use the geometric term since we integrate over the area of the light source. The geometric term is then multiplied to the BRDF, that scales how much of the received light, that goes toward $\mathbf{x}_{s-1}$.

The final part of equation (16) corresponds to the indirect lighting. This is computed, at each point on the path, as the product of the BRDF and a dot product, divided by the PDF from the BRDF. This gives us the scale factor of how much of the light received in $\mathbf{x}_i$ will be sent towards $\mathbf{x}_{i-1}$. According to equation (10) we should have used the geometric term instead of the dot product, but this would also mean that we should use a PDF given in respect to surface area. Since we get the the PDF from the BRDF using importance sampling, we get the PDF in respect to solid angle and we would have to transform it, this gives us:

$$\frac{G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})}{p_A(\mathbf{x}_{i+1})} = \frac{\frac{|N_{\mathbf{x}''} \cdot \overrightarrow{\mathbf{x}''\mathbf{x}'}| \ |N_{\mathbf{x}'} \cdot \overrightarrow{\mathbf{x}'\mathbf{x}''}|}{\|\mathbf{x}'' - \mathbf{x}'\|^2}}{p(\omega_o)\left(\frac{|N_{\mathbf{x}'} \cdot \overrightarrow{\mathbf{x}'\mathbf{x}''}|}{\|\mathbf{x}'' - \mathbf{x}'\|^2}\right)} = \frac{|N_{\mathbf{x}''} \cdot \overrightarrow{\mathbf{x}''\mathbf{x}'}|}{p(\omega_o)}.$$

6 Light Tracing

Another way to solve the global illumination problem, is the light tracing method. Instead of sending rays out from the eye that will eventually hit a light source, as in path tracing, we send out rays from the light source that eventually will hit the eye. More precisely, we want to see how much of the light, that is emitted from the light source, is seen by the eye.

We will start by finding out how much flux is emitted from the light source. In our implementation we only use light sources that emits equal amounts of light in all directions, from each point on the surface, but other types of light sources such as spotlights, which only emits light in certain directions, could also be used. The function, $L_e(\mathbf{y}, \omega_o)$ describes how much radiance is emitted at point $\mathbf{y}$ in the direction ω_o. By using $L_e(\mathbf{y}, \omega_o)$, we can find the amount of flux, that is emitted from a light source, by integrating over all surface points and directions. The total amount of flux leaving a light source is found as:

$$\Phi = \int_A \int_\Omega L_e(\mathbf{y}, \omega_o) |N_\mathbf{y} \cdot \omega_o| \; d\omega \; dA. \tag{17}$$

If we want to find out how much of the flux, that is emitted from the light source, actually hits a pixel, P_j, we can insert the potential equation, from equation (5) into equation (17). This gives us the amount of flux that hits the set S:

$$\Phi(S) = \int_A \int_\Omega L_e(\mathbf{y}, \omega_o) W(\mathbf{y}, \omega_o) |N_\mathbf{y} \cdot \omega_o| \; d\omega \; dA,$$

where S is the set of paths that goes from a point, $\mathbf{y}$, on the light source, through a series of reflections and hits the eye, $\mathbf{x}_0$, through the pixel P_j (see figure 7). The emitted potential

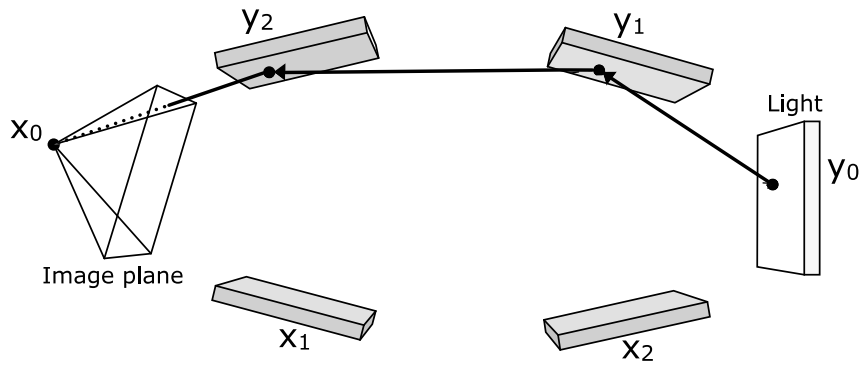


Figure 7: An example of a path starting at the point $\mathbf{y}_0$ on the light source, going through the reflections at $\mathbf{y}_1$ and $\mathbf{y}_2$. The path ends at the eye, $\mathbf{x}_0$, going through pixel P_j

from the potential equation, is given as $W_e(\mathbf{x}, \omega_o)$ where $\mathbf{x}$, is a point on the viewport. Since we are using a pinhole camera it means that for each point $\mathbf{x}$ there is only one direction where $W_e(\mathbf{x}, \omega_o)$ is 1. This is the direction going from the point $\mathbf{x}$ to the eye point, this means that only very few or none of the rays we send out from the light source will contribute to the final image. We will look at this problem in section 6.1. Another problem is that right now we can only find the amount of flux that flows through a pixel, but what we want is the amount of radiance flowing through a pixel. We will show a way to convert the flux into radiance in section 6.2. Lastly, in section 6.3 we will show how the path integral can be applied to light tracing.

6.1 Next Event Estimation

As it is impossible to use a point light source in the path tracing algorithm without direct lighting, we have a similar problem with using light tracing and a pin hole camera. If we use a pin hole camera, the light rays have to hit the eye position, which is an infinitly small point, it will be so unlikely that we say that it is impossible. This makes it impossible to use light tracing with a pin hole camera. We can solve the problem, using next event estimation, in a similar way to what we do in direct lighting. We start with the potential equation, that can be written as:

$$W(\mathbf{y}, \omega_o) = W_e(\mathbf{y}, \omega_o) + W_r(\mathbf{y}, \omega_o), \tag{18}$$

where $W_e(\mathbf{y}, \omega_o)$ is the emitted potential and $W_r(\mathbf{y}, \omega_o)$ is the reflected potential. The reflected potential can be written as:

$$W_r(\mathbf{y}, \omega_o) = \int_\Omega W(\mathbf{y}', -\omega_i) f_r(\mathbf{y}, \omega_i, \omega_o) |\omega_o \cdot N_\mathbf{y}| \; d\omega_i \tag{19}$$

We can now use next event estimation by inserting equation (18) into equation (19) and we get:

$$\begin{aligned}
W_r(\mathbf{y}, \omega_o) &= \int_\Omega [W_e(\mathbf{y}', -\omega_i) + W_r(\mathbf{y}', -\omega_i)] f_r(\mathbf{y}, \omega_i, \omega_o) |\omega_o \cdot N_\mathbf{y}| \; d\omega_i \\
&= \int_\Omega W_e(\mathbf{y}', -\omega_i) f_r(\mathbf{y}, \omega_i, \omega_o) |\omega_o \cdot N_\mathbf{y}| \; d\omega_i + \int_\Omega W_r(\mathbf{y}', -\omega_i) f_r(\mathbf{y}, \omega_i, \omega_o) |\omega_o \cdot N_\mathbf{y}| \; d\omega_i \\
&= \int_A W_e(\mathbf{y}' \to \mathbf{y}) f_r(\mathbf{y}, \overrightarrow{\mathbf{y}'\mathbf{y}}, \omega_o) G(\mathbf{y}' \leftrightarrow \mathbf{y}) \; dA + \int_\Omega W_r(\mathbf{y}', -\omega_i) f_r(\mathbf{y}, \omega_i, \omega_o) |\omega_o \cdot N_\mathbf{y}| \; d\omega_i \\
&= W_{direct} + W_{indirect}
\end{aligned} \tag{20}$$

The first term is the direct contribution of the flux from the light source. We integrate over the area, A, of the viewport but since we are using a pinhole camera, only the direction going towards the eye point will contribute. We can therefore just send a single ray towards the eye, to get the direct potential. The second term is the potential going through reflections.

6.2 Converting Flux to Radiance

The result we get from the potential equation is the combined flux, $d\Phi$, that flows through a pixel set, but we are interested in finding the intensity. To do this we must convert the flux of each pixel set into a radiance value. Radiance is defined as flux per solid angle per projected area:

$$L(\mathbf{x}, \omega) = \frac{d^2\Phi}{d\omega \; dA^\perp}$$

The projected area, $dA^\perp$, is the projected area of the eye (see figure 8) and can be found as:

$$dA^\perp = cos(\theta) d\mu_{\text{eye}}$$

Remembering that the relationship between the differential solid angle, $d\omega$, and the differential surface area, dA, is given as:

$$d\omega = \frac{dA \; |N_{\mathbf{x}''} \cdot \overrightarrow{\mathbf{x}''\mathbf{x}'}|}{\|\mathbf{x}'' - \mathbf{x}'\|^2}.$$

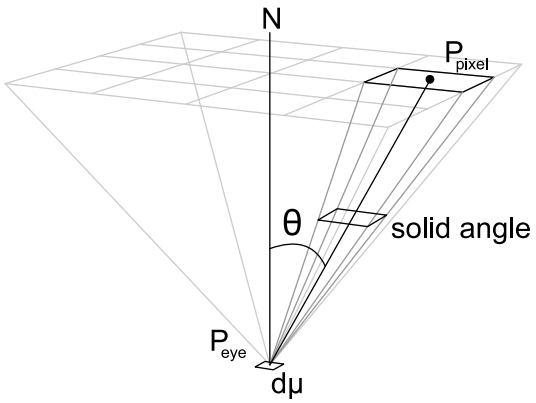


Figure 8: We convert the amount of flux that is in point P_{pixel}, into the amount of radiance that hits the area of the eye, $d\mu_{\text{eye}}$, at point P_{eye}. N is the normal. θ is the angle between the normal and the direction from P_{eye} to P_{pixel}.

We can find the solid angle of the pixel seen from the eye:

$$d\omega = \frac{A_{\text{pixel}}\ cos(\theta)}{\|P_{\text{eye}} - P_{\text{pixel}}\|^2}$$

We can now find the radiance of a pixel set as:

$$L_{\text{pixel}} = \frac{d\Phi}{cos(\theta)\ d\mu_{\text{eye}}\ \frac{A_{\text{pixel}}\ cos(\theta)}{\|P_{\text{pixel}} - P_{\text{eye}}\|^2}}$$

and we can rewrite this, such that it can be multiplied onto the flux to convert it to radiance. We approximate the area of the eye as a infinitly small point, and thus the $d\mu_{eye}$ goes away and we get:

$$L_{\text{pixel}} = d\Phi \frac{\|P_{\text{pixel}} - P_{\text{eye}}\|^2}{cos(\theta)^2\ A_{pixel}}.$$

6.3 Applying the Path Integral

To find the radiance measurement of a pixel, L_p we use:

$$L_p = \frac{1}{N} \sum_{i=0}^{N} L_{p_i},$$

L_{p_i} is the radiance of a single sample. In path tracing, N would be the number of samples taken for each pixel. In light tracing, N is the number of rays emitted from the light source. This is because each ray that is emitted approximates the entire amount of flux that is emitted from the light source and so each ray emitted could potentially contribute to each pixel. For example if we emit 10 rays from the light source, and only one of them hit pixel j, then we estimate that only 10% of the combined flux emitted by the light is received at pixel j.

By using the path integral on L_{p_i} we get:

$$L_p = \sum_{t=0}^{N_t} C_t\ \frac{\|P_{\text{pixel}} - P_{\text{eye}}\|^2}{cos(\theta)^2\ A_{pixel}}.$$

Here we sum the contributions, C_t, with path length t, up to the maximum path length N_t. This is just like we do in path tracing, the only difference, is that we get the result in flux. To get the result in radiance we multiply each path with the flux-to-radiance function. C_t, can be computed as:

$$C_t = \frac{L_e(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{y}_1})}{p(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{y}_1})} \left(\prod_{i=1}^{t-1} \frac{f_r(\mathbf{y}_i, \overrightarrow{\mathbf{y}_i\mathbf{y}_{i-1}}, \overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}}) |N_{\mathbf{y}_i} \cdot \overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}}|}{p(\overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}})} \right)$$
$$f_r(\mathbf{y}_t, \overrightarrow{\mathbf{y}_t y_{t-1}}, \overrightarrow{\mathbf{y}_t\mathbf{x}_0}) G(\mathbf{y}_t \leftrightarrow \mathbf{x}_0) \frac{W_e(\mathbf{x}_0, \overrightarrow{\mathbf{y}_t\mathbf{x}_0})}{p(\mathbf{x}_0)}.$$

The first part is $L_e(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{y}_1})$, which is the emitted light from the light source at point $\mathbf{y}_0$. The PDF, $p(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{y}_1})$, give us the probability distribution of selecting a ray from $\mathbf{y}_0$ in the direction $\overrightarrow{\mathbf{y}_0\mathbf{y}_1}$. For a lightsource that has a uniform distribution of the light, the PDF is:

$$p(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{y}_1}) = \frac{1}{A_{\text{light}}\ 2\pi},$$

since $\mathbf{y}_0$ can be sampled from the surface area A_{light} and can go in all the directions of the hemisphere above the point $\mathbf{y}_0$ and the solid angle for a hemisphere is 2π. The next part is the product of the BRDF multiplied with the dot product. This is the indirect part of the path. The PDF, $p(\overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}})$ can be found by using the pdf from the BRDF.

The third part is $f_r(\mathbf{y}_t, \overrightarrow{\mathbf{y}_t y_{t-1}}, \overrightarrow{\mathbf{y}_t\mathbf{x}_0}) G(\mathbf{y}_t \leftrightarrow \mathbf{x}_0)$. This is the direct part of the path, where we create a ray from $\mathbf{y}_t$ directly to the eye, $\mathbf{x}_0$.

In the last part $W_e(\mathbf{x}_0, \overrightarrow{\mathbf{y}_t\mathbf{x}_0})$ will always give 1, since we are using a pinhole camera, unless the ray between $\mathbf{x}_0$ and $\mathbf{y}_t$ does not hit the viewport or if the ray comes from behind the camera. $p(\mathbf{x}_0)$ will always give 1 because there is only one point we can choose as $\mathbf{x}_0$.

7 Bidirectional Path Tracing

Both the path tracing and light tracing method can be used to solve the global illumination problem. Some paths are however more easily traced with one method, than the other. Path tracing have problems with paths that include caustics. A path, containg caustics, must first hit a diffuse material, then go through a series of specular bounces and finally hit a lightsource. This has a low probability but a high contribution, which creates a lot of noise in the image. In general we can say that path tracing performs poorly when it can't use direct lighting.

Light tracing is very good at tracing paths containing caustics, but for most other paths it creates a very noisy image. Luckily it turns out that both method can be combined into one method, that uses the best from path tracing and light tracing. This method is called bidirectional path tracing.

7.1 Sampling the Paths

In bidirectional path tracing we start from both ends of the path. We create a subpath, starting from the eye: $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_s$ and another subpath, starting at a lightsource: $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_t$. The length of each subpath can be determined by either setting a max path length or by using a method such as russian roulette, that randomly stops the path after som criteria. When we have both subpaths, we can combine them by creating an edge between both ends of each subpath, to create a single path going from the eye to the light source: $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_s, \mathbf{y}_t, ..., \mathbf{y}_2, \mathbf{y}_1, \mathbf{y}_0$. To make sure it is a legal path we have to do a visibility check using a shadow ray, between the points $\mathbf{x}_s$ and $\mathbf{y}_t$.

A good optimization strategy is to connect all the points on the eye path with all the vertices on the light path, (see figure 9). This way we can create multiple paths by reuseing all the points we have traced in the scene. The only cost is that we have to do a visibility check between the end points, using a shadow ray.

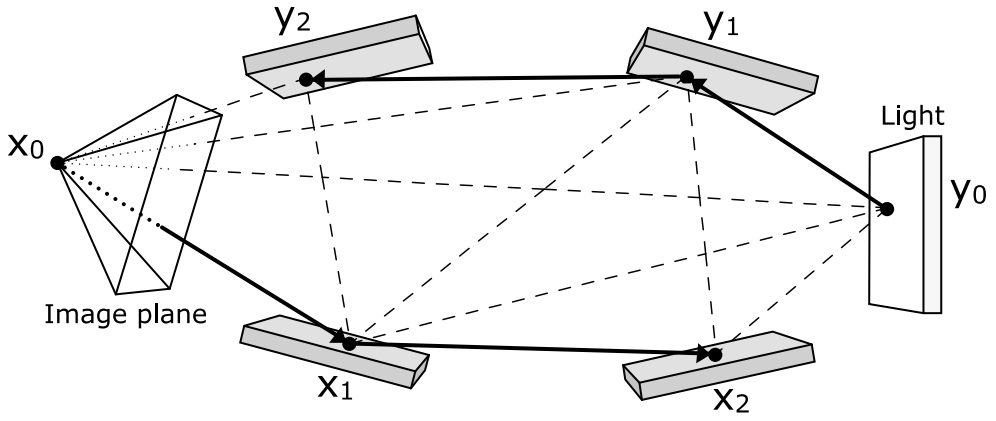


Figure 9: The different ways to combine the eye path and the light path.

7.2 A Bidirectional Estimator

We can write the estimate for the radiance as

$$L_p = \sum_{s=0}^{N_E} \sum_{t=0}^{N_L} w_{s,t} V(s,t) C_{s,t}. \tag{21}$$

Where $V(s,t)$ is the visibility check between the s'th point on the eye path and the t'th point on the light path. $w_{s,t}$ is a weight function. $C_{s,t}$ is the contribution of a path with s vertices on the eye path and t vertices on the light path. There are four distinct cases for which we have to evaluate $C_{s,t}$:

- $s = 0, t = 0$: This is the light that is directly visible to the eye. It can be evaluated by creating a path between $\mathbf{y}_0$ and the eyepoint $\mathbf{x}_0$.

$$C_{0,0} = L_e(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{x}_0}) G(\mathbf{x}_0, \mathbf{y}_0)$$

- $s > 0, t = 0$: This is the light we get when we go through an eye path and create a point on the light source. This term corresponds to classic path tracing with direct lighting. Instead of using $\mathbf{y}_0$ as the point on the light source we can reduce the noise in the image by sampling a new point on the light source. In this way it is also possible to use better sampling methods such as the ones described in [Shirley et al., 1996].

$$C_s = \frac{L_e(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{x}_s})}{p(\mathbf{y}_0)} f_r(\mathbf{x}_s, \overrightarrow{\mathbf{x}_s\mathbf{y}_0}, \overrightarrow{\mathbf{x}_s\mathbf{x}_{s-1}}) G(\mathbf{y}_0 \leftrightarrow \mathbf{x}_s) \left(\prod_{i=1}^{s-1} \frac{f_r(\mathbf{x}_i, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}) |N_{\mathbf{x}_i} \cdot \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}|}{p(\overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}})} \right)$$

- $s = 0, t > 0$: This term corresponds to the light tracing algorithm. Note, that this will be estimates for other pixels in the image.

$$C_{t,0} = \frac{L_e(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{y}_1})}{p(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{y}_1})} \left(\prod_{i=1}^{t-1} \frac{f_r(\mathbf{y}_i, \overrightarrow{\mathbf{y}_i\mathbf{y}_{i-1}}, \overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}}) |N_{\mathbf{y}_i} \cdot \overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}}|}{p(\overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}})} \right) f_r(\mathbf{y}_t, \overrightarrow{\mathbf{y}_t y_{t-1}}, \overrightarrow{\mathbf{y}_t\mathbf{x}_0}) G(\mathbf{y}_t \leftrightarrow \mathbf{x}_0) \frac{W_e(\mathbf{x}_0, \overrightarrow{\mathbf{y}_t\mathbf{x}_0})}{p(\mathbf{x}_0)}.$$

- $s > 0, t > 0$: This term is the estimation of the radiance reaching the eye, on a path connected by s verticed on the eye path and t vertices on the light path.

$$C_{t,s} = \frac{L_e(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{y}_1})}{p(\mathbf{y}_0, \overrightarrow{\mathbf{y}_0\mathbf{y}_1})} \left(\prod_{i=1}^{t-1} \frac{f_r(\mathbf{y}_i, \overrightarrow{\mathbf{y}_i\mathbf{y}_{i-1}}, \overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}}) |N_{\mathbf{y}_i} \cdot \overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}}|}{p(\overrightarrow{\mathbf{y}_i\mathbf{y}_{i+1}})} \right) f_r(\mathbf{y}_t, \overrightarrow{\mathbf{y}_t\mathbf{y}_{t-1}}, \overrightarrow{\mathbf{y}_t\mathbf{x}_s}) G(\mathbf{y}_t, \mathbf{x}_s) f_r(\mathbf{x}_s, \overrightarrow{\mathbf{x}_s\mathbf{y}_t}, \overrightarrow{\mathbf{x}_s\mathbf{x}_{s-1}}) \left(\prod_{i=1}^{s-1} \frac{f_r(\mathbf{x}_i, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}) |N_{\mathbf{x}_i} \cdot \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}|}{p(\overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}})} \right)$$

7.3 The Weight Function

We will now look at how we can create a weight function, $w_{s,t}$, that we can use in equation (21).

If we look at figure 10, then we can see the contributions of paths with s points on the eye path and t points on the light path. If we define our weight function to be

$$w_{s,t} = \begin{cases} 1 & \text{if } s = 0 \\ 0 & \text{if } s \neq 0 \end{cases},$$

we would only use the sum of the contributions of the left most column in figure 10. This corresponds to using the path tracing method. Similarly we could use a weight function that would only use the top row of figure 10. This would correspond to using the light tracing method. Both of these weight functions are valid, but they are also very wastefull, since we throw away a lot of perfectly good samples.

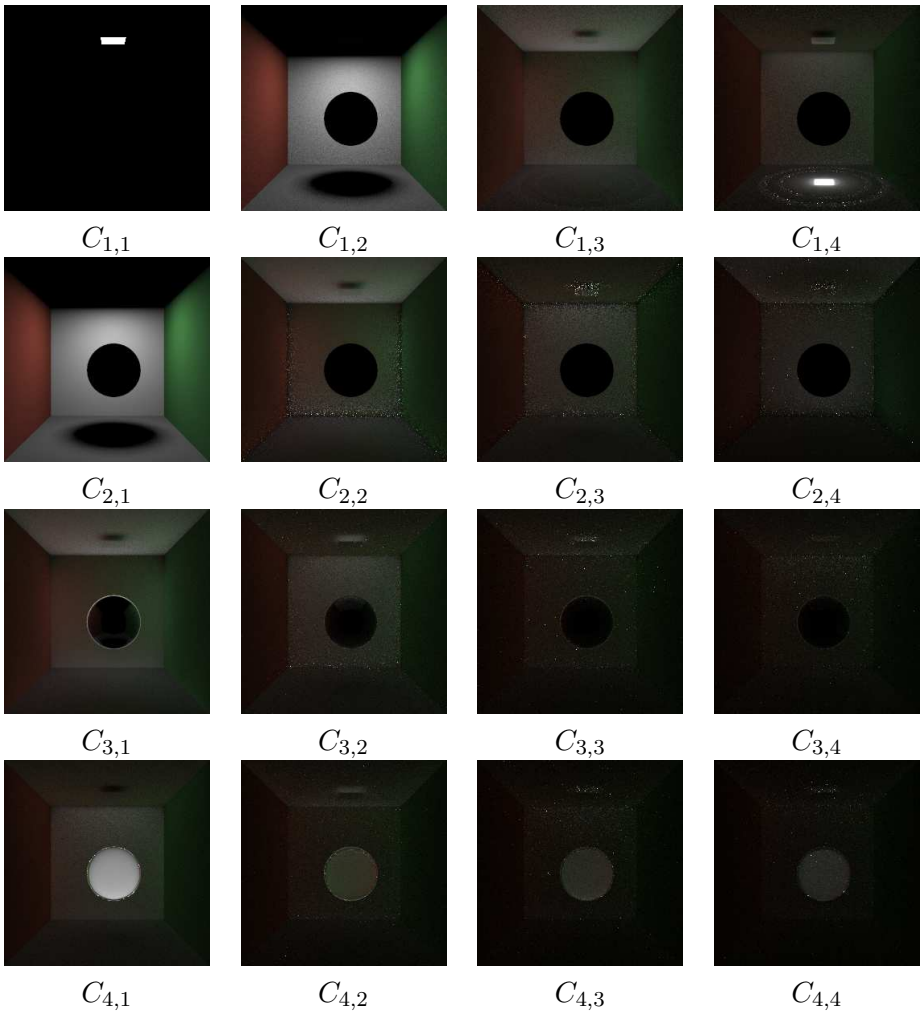


Figure 10: The unweighted contribution, $C_{s,t}$ for different paths, with an eye path with s points and a light path with t points.

If we, on figure 10, look at the images for $C_{2,1}$ and $C_{1,2}$ then both images are the con-

tribution of the light going from the lightsource, through a single reflection and into the eye. The only difference is that we have started at opposite ends of the paths. We can combine, these two contributions into a single result. The naive way of doing it, would be to take the average of the two contributions. We can do the same with a path length of three, by taking the average of $C_{3,1}$, $C_{2,2}$, and $C_{1,3}$. The general rule for any weight function is that the weight of all paths with the same length must sum up to one:

$$\sum_{s=0}^{N} w_{s,N-s} = 1(N = 0, 1, ...),$$

where N is the length of the path.

7.4 The Weight Function With Multiple Importance Sampling

A better way to create the weight function, is to use multiple importance sampling. In this method we look at, how the same path can be sampled in different ways. If we look at figure 11, we can see how the same path, with $s + t$ points, can be sampled in $s + t - 1$ different ways. Since these three different samples are the same path, their weights must sum to one.

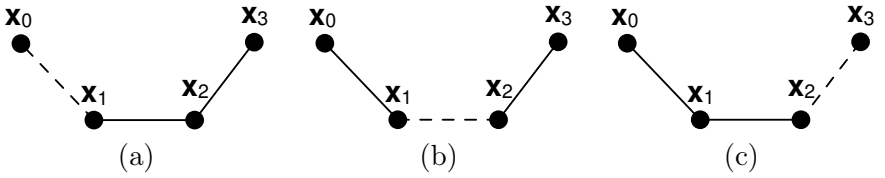


Figure 11: $\mathbf{x}_0$ is the eye point, $\mathbf{x}_1$ is point on the eye path, $\mathbf{y}_0$ is a point on the light source, and $\mathbf{y}_1$ is a point on the light path. We can see how the same path can be sampled in three different ways. (a) would correspond to light tracing, (b) would be bidirectional path tracing and (c) corresponds to path tracing.

In [Veach, 1997] he argues, that the best way to use multiple importance sampling, to create a weight function, is to use the power heuristic:

$$w_{s,t} = \frac{p_j^\beta}{\sum_{i=0}^{(s+t-1)} p_i^\beta}. \tag{22}$$

He further argues that a good value for the power, β, is 2. p_j is the PDF, in respect to surface area, for the path we are currently evaluating, while $p_0...p_{j-1}$ and $p_{j+1}...p_{j+t}$ represents the PDFs of the paths we could have created using the same points. We can easily see that this weight function will sum up to one. Furthermore we can simplify equation (22), by observing that the values only matter up to a scale value.

$$w_{s,t} = \frac{p_j^2}{\sum_{i=0}^{(s+t-1)} p^2} = \frac{1}{\sum_{i=0}^{(s+t-1)} (p_i/p_j)^2} \tag{23}$$

To find the PDF, p_j, for the path we are evaluating we have we have to find the PDF, p^L, for the light path and the PDF, p^E, for the eye path. p^L and p^E can both be found by multiplying the PDFs from each point on their subpath. Since we get the PDFs from

the BRDFs, in respect to solid angle, we have to remember to convert them into PDFs with respect to surface area. The combined PDF for the entire path is given as:

$$p_j = p^L p^E$$

Finding the PDFs for the other possible paths, is a little bit harder, since now we have to find the PDF for paths we didn't sample. But if we look at equation (23), we can see that we only have to find the ratio p_i/p_j.

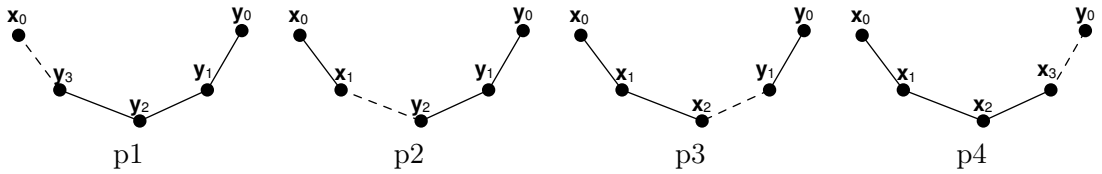


Figure 12: The same path, sampled in differrent ways. Even though a name for a point changes, it is still the same point.

For example if we know the PDF $p2$ (see figure 12), then we can find the ratio p_3/p_2 as:

$$\begin{aligned}
\frac{p_3}{p_2} &= \frac{p_3^L\ p_3^E}{p_2^L\ p_2^E} \\
&= \frac{p(\mathbf{x}_0 \to \mathbf{x}_1)p(\mathbf{x}_1 \to \mathbf{x}_2)p(\mathbf{y}_0 \to \mathbf{y}_1)}{p(\mathbf{x}_0 \to \mathbf{x}_1)p(\mathbf{y}_0 \to \mathbf{y}_1)p(\mathbf{y}_1 \to \mathbf{y}_2)} \\
&= \frac{p(\mathbf{x}_1 \to \mathbf{x}_2)}{p(\mathbf{y}_1 \to \mathbf{y}_2)}
\end{aligned}$$

To find $p(\mathbf{y}_1 \to \mathbf{y}_2)$ in $p3$, we have to extend our BRDF to give us the PDF, from finding the direction: $\overrightarrow{\mathbf{x}_1\mathbf{x}_2}$, given the direction: $\overrightarrow{\mathbf{x}_1\mathbf{x}_0}$. The general rule for finding the ratio p_{i+1}/p_i is given as:

$$\frac{p_{i+1}}{p_i} = \frac{P(\mathbf{x}_{i-1} \to \mathbf{x}_i)}{P(\mathbf{y}_{i-1} \to \mathbf{y}_i)}.$$

If we want to go the other way, we can find the ratio for p_{i-1}/p_i as:

$$\frac{p_{i-1}}{p_i} = \frac{P(\mathbf{y}_{i-1} \to \mathbf{y}_i)}{P(\mathbf{x}_{i-1} \to \mathbf{x}_i)}.$$

Given that we know the ratio $\frac{p_{j+1}}{p_j}$, we can find the next ratio by multiplying it with the former ratio:

$$\frac{p_{j+2}}{p_{j+1}}\frac{p_{j+1}}{p_j} = \frac{p_{j+2}}{p_j}$$

We can then use the ratio $\frac{p_{j+2}}{p_j}$ to find $\frac{p_{j+3}}{p_j}$, which can then be used to find $\frac{p_{j+4}}{p_j}$ and so on:

$$\begin{aligned}
\frac{p_{j+3}}{p_{j+2}}\frac{p_{j+2}}{p_j} &= \frac{p_{j+3}}{p_j} \\
\frac{p_{j+4}}{p_{j+3}}\frac{p_{j+3}}{p_j} &= \frac{p_{j+4}}{p_j}
\end{aligned}$$

If we go in the other direction we can, in the same way, find the ratios for $\frac{p_{i-1}}{p_i}$ as:

$$\frac{p_{j-2}}{p_{j-1}}\frac{p_{j-1}}{p_j} = \frac{p_{j-2}}{p_j}$$
$$\frac{p_{j-3}}{p_{j-2}}\frac{p_{j-2}}{p_j} = \frac{p_{j-3}}{p_j}$$
$$\frac{p_{j-4}}{p_{j-3}}\frac{p_{j-3}}{p_j} = \frac{p_{j-4}}{p_j}$$

7.5 Handeling Specular Surfaces

When we are connecting the two end points $\mathbf{x}_s$ and $\mathbf{y}_t$ (see figure 13), by creating an edge between them, we want to find out how much of the light is reflected from $\mathbf{y}_t$ to $\mathbf{x}_s$. As a part of finding the reflection, we use the BRDF of point $\mathbf{y}_t$ to find out how much of the light comming from direction $\overrightarrow{\mathbf{y}_{t-1}\mathbf{y}_t}$ goes out in the direction $\overrightarrow{\mathbf{y}_t\mathbf{x}_s}$. There is however a problem if the BRDF uses a dirac delta distribution function, that is defined such that $\int \delta(x)dx = 1$ and for all $x \neq 0$, $\delta(x) = 0$. An example of this is a perfect mirror, where given the direction $\overrightarrow{\mathbf{y}_{t-1}\mathbf{y}_t}$ the light will only be reflected in the direction $\overrightarrow{\mathbf{y}_t\mathbf{x}'_s}$ and the probability that $\overrightarrow{\mathbf{y}_t\mathbf{x}'_s}$ is equal to $\overrightarrow{\mathbf{y}_t\mathbf{x}_s}$ is so low that we say it's zero. This is a problem for all paths where one of the

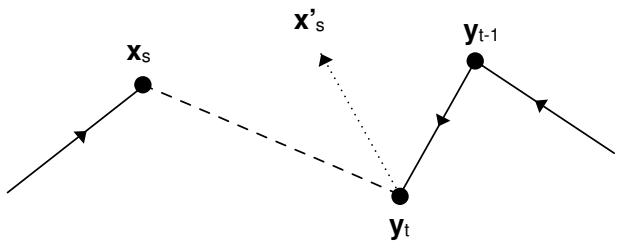


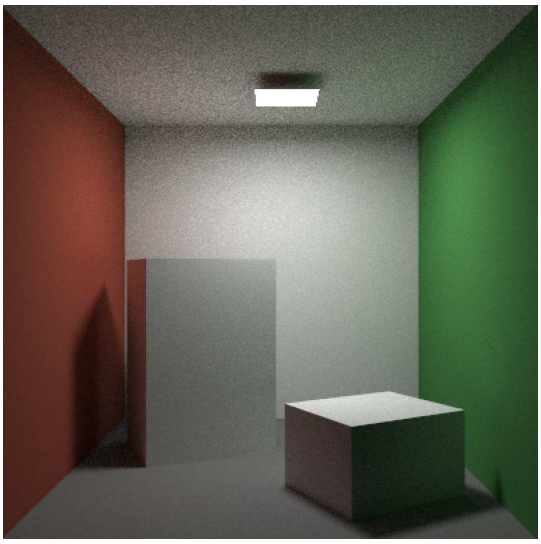
Figure 13: Two subpaths that has been connected with an edge, between point $\mathbf{x}_s$ and $\mathbf{y}_t$. The BRDF in point $\mathbf{y}_t$ only reflects light in the direction towards $\mathbf{x}'_s$, this means that no light will go towards $\mathbf{x}_s$.

points is on a surface with a material that use a dirac distribution. We can solve the problem by making sure that the weight of the path will be zero for such paths and therefore they will not contribute to the final image. We do this when the weight is computed. If $\mathbf{x}_s$ or $\mathbf{y}_t$ is a point on a surface with a material that uses a dirac delta distribution then the PDF for the entire path is set to zero.

8 Results

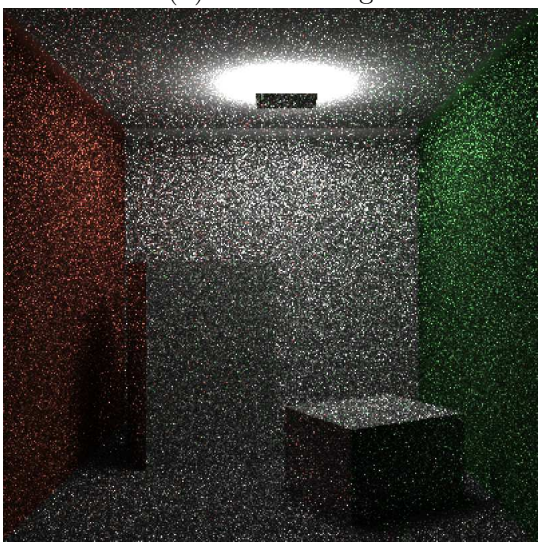
We will here present a few test images, that has been rendered with our implementation of a path tracer and a bidirectional path tracer. As can be seen in some of the images, there is still a few bugs left in the code, but none that should affect the results of these test, in any major way.

(a) Bidirectional Path Tracing



(b) Path tracing

(c) Bidirectional Path Tracing



(d) Path tracing

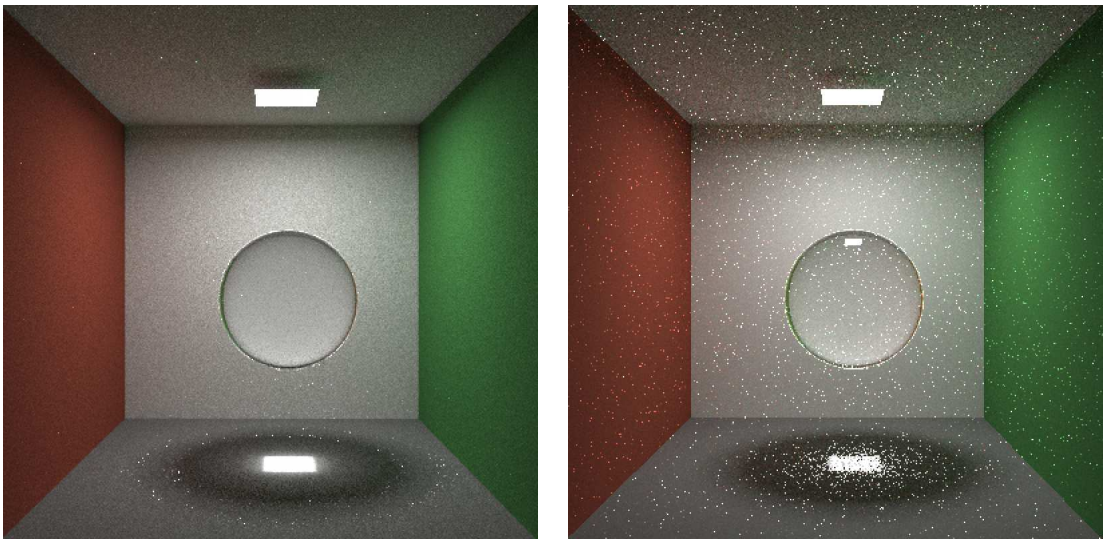
Figure 14: A cornell box scene with diffuse surfaces. In (a) and (b) the light source is pointing downwards. In (c) and (d) the light source is point upwards.

In the first test (see figure 14) we have a scene based on the classic cornell box. Inside the box are two smaller boxes and a light source in the ceiling. Every object in this scene has a diffuse material. Each image has an image size of 400×400 pixels and has been given the same render time. In the first image (14.a) the scene is rendered with a bidirectional path

tracer, with a maximum path length of four. For comparison, the same scene is rendered with a path tracer in (14.b), also with a maximum path length of four. We can see that the image (14.a) rendered with bidirectional path tracing, actually has a bit more noise than the one using only path tracing.

In (14.c), the almost, same scene is rendered. The difference is, that the lightsource has been flipped, so that it now points up into the ceiling. If we compare with (14.a), we can see that, it has not really affected the quality of the image. In (14.d) we have the same setup as in (14.c). If we compare (14.d) to (14.b), we can easily see that, there is a major difference in the quality. The noise is due to the fact that, only few of the points in the scene receives any direct lighting. In this case the path tracer performs very poorly, compared to the bidirectional path tracer.

In the second test (see figure 15), we have a scene that contains the same cornell box, as in the previous test. Inside the box is a sphere, with a glass material. The images are rendered in 400×400 pixels and rendered with the same render time. In (15.a) the scene is rendered with a bidirectional path tracer, using a maximum path length of four. For comparison the same scene is rendered in (15.b) using a path tracer, with a maximum path length of four.



(a) Bidirectional Path Tracing (b) Path tracing

Figure 15: The same scene renderer with both bidirectional path tracing in (a) and path tracing in (b). It is clear that the image rendered with bidirectional path tracing contains a lot less noise.

In (15.b) the caustics creates a lot of noise in the scene. The refracted caustic below the glass sphere is just a lot of noise. The reflected caustics are spread out into scene and creates an overall noisy image. With our implementation it will take many hours before the noise is completly gone. In the image rendered with bidirectional path tracing, (15.a), we have a beatiful caustic below the glass sphere and there is almost none of the reflected noise we can see in (15.b). There is however some noise on the floor, that looks like two circles of noise. We belive that this is due to a bug in the implementation of the raytracer and not a fault of the bidirectional path tracer.

In figure 16, we have the different contributions, that was combined, to create (15.a). Each row of the pyramid contains contributions, from paths with the same path length.

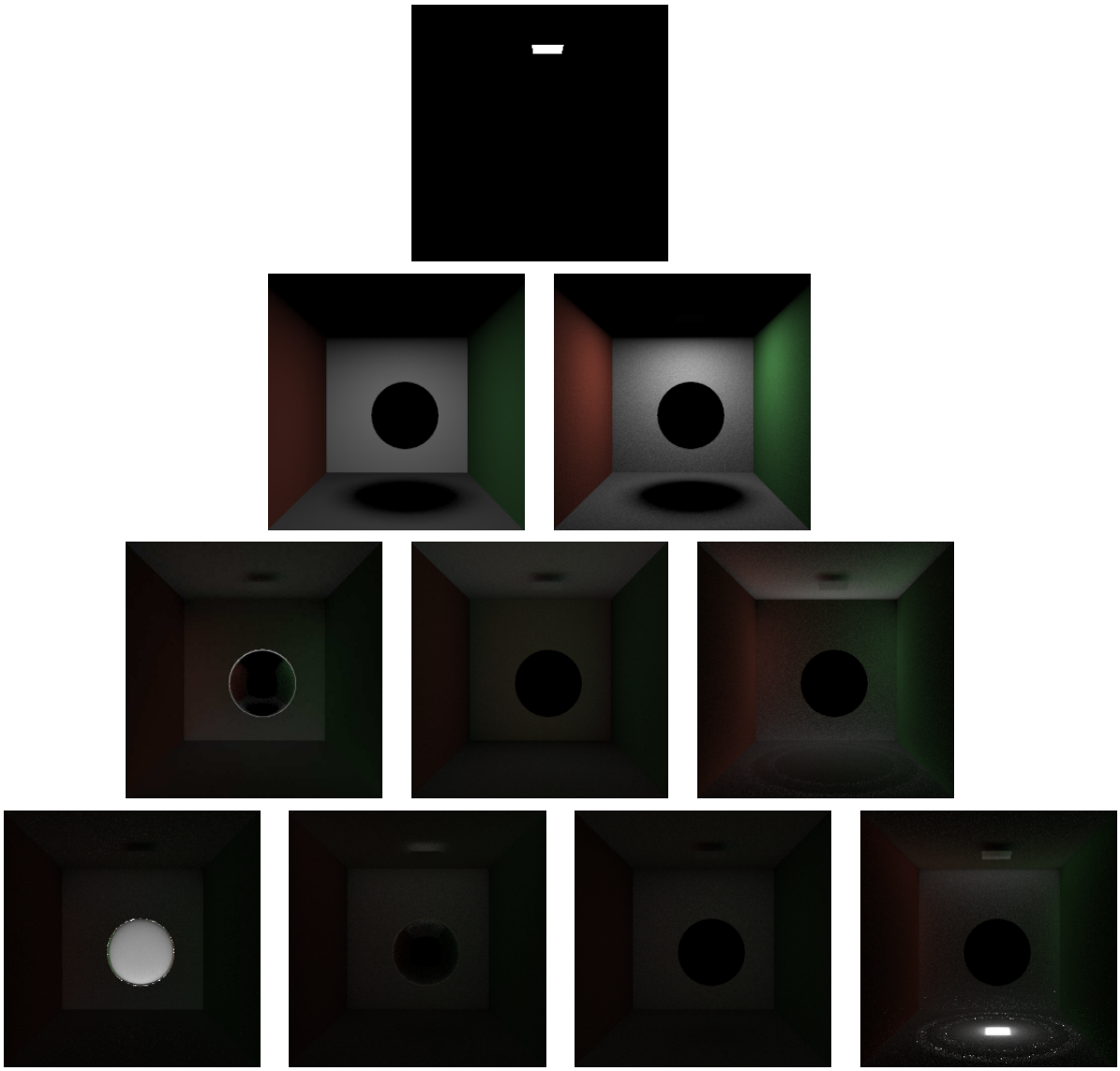


Figure 16: Each row of the pyramid contains contributions, from paths with the same path length. The contributions that we see are weighted with multiple importance sampling. In the top image we have the light that is directly visible to eye, this is kind of boring since we can only see the light source. In the second row we can see the direct lighting contribution. The image on the left is due to path tracing and the image on the right is due to light tracing. The images are very similar. We would have preferred that the light tracing contribution was zero, since path tracing is alot better at direct lighting. The last row contains some more interresting things. On the image to the left we can see that it is almost completly dark, except for the refraction we can see inside the sphere. In the other images the sphere is completly black. This is an example of how the multiple importance sampling has choosen tp weight the others with zero, since they do not contribute to the final image.

9 Conclusion

We have now presented the necessary theory, that is needed, to implement a basic bidirectional path tracer. There is, however still some improvements that can be made. The speed can be improved, by reducing the amount of visibility checks, between the subpaths, by using a method such as the one found in [Veach and Guibas, 1997].

A problem in our implementation, is that we do not consider paths with zero points on the light path. This means that we can't render caustics that has been reflected in a perfect mirror. This i due to the fact that the path contains two or more materials that uses a dirac delta distribution.

If we compare bidirectional path tracing to simple path tracing, we saw in section 8 that for some scenes, bidirectional path tracing performs a lot better. This is especially in scenes where there is very little direct lighting and in scenes containing caustics.

Bidirectional path tracing also has some cases where it peforms badly. Bidirectional path tracing is not very efficient for outdoor scenes, since shooting out rays from the sun, will have a very small probability of hitting the scene. Another example is, where we have a building with multiple rooms and each room has it's own light source. If the eye is placed in one of the rooms and the doors between the rooms are closed, then bidirectional path tracing will spend a lot of time creating light paths, that can never be connected to the eye paths. If one of the doors is opened up, the probability of tracing through the door opening, can still be very low.

A method such as metropolis light transport is better at handeling scenes, that is lit through small openings such as doors. If the metropolis light transport method find a good path through the door it will mutate into similar paths that goes through the door. This gives us a lot of samples, of the light found in the other room. Metropolis light transport uses bidirectional path tracing, to create the initial path it can mutate from. Metropolis light transport is therefore a popular extension to bidirectional path tracing.

Another method that compares to bidirectional path tracing is photon mapping. Just as in bidirectional path tracing, photon mapping shoot out rays from both the eye and the light source, except that when it shoots out from the light source, it is called photons instead of rays. The difference, from bidirectional path tracing, is that it uses two passes. In the first pass, it shoots out photons from the light source. When the photons hit a diffuse surface, the photon is saved in the photon map. In the second pass we shoot out rays from the eye. For each point we hit, we do a lookup into the photon map to find the n nearest photons. We use the nearest photons to estimate the reflected radiance of the point.

The benefit of photon mapping is that it is very fast and if enough photons are shoot into the scene, it can create images as good as the ones created with bidirectional path tracing, but usually a lot faster. The problem with photon mapping is that it is biased towards how many photons are used. If we use to few photons, it will create artifacts, no matter how long the render time is for the second pass. Bidirectional path tracing is unbiased, if we do not use a maximum path length, but instead uses a method such as russian roulette to determine the path length. This means that while bidirectional path tracing has a slow convergence, it is guranteed that given enough time it will not have any artifacts.

The conclusion must be that if we want to render images in a fast way, but risk artifacts, we should use photon mapping. If we instead do not want to tinker, with getting the right amount of photons and have unlimited time, we should use metropolis light transport. All in all bidirectional path tracing seems, to be just a speed bump, towards implementing a metropolis light transport renderer.

References

[jen, 2001] (2001). *Realistic Image Synthesis Using Photon Mapping*. A K Peters, Ltd.

[Arvo, 1986] Arvo, J. (1986). Backward ray tracing. In *In ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*.

[Chen et al., 1991] Chen, S. E., Rushmeier, H. E., Miller, G., and Turner, D. (1991). A progressive multi-pass method for global illumination. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 165–174, New York, NY, USA. ACM.

[Cook et al., 1984] Cook, R. L., Porter, T., and Carpenter, L. (1984). Distributed ray tracing. *SIGGRAPH Comput. Graph.*, 18(3):137–145.

[Dutre et al., 1993] Dutre, P., Lafortune, E., and Willems, Y. D. (1993). Monte Carlo Light Tracing with Direct Pixel Contributions. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 128–137, Alvor, Portugal.

[Goral et al., 1984] Goral, C. M., Torrance, K. E., Greenberg, D. P., and Battaile, B. (1984). Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.*, 18(3):213–222.

[Kajiya, 1986] Kajiya, J. T. (1986). The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150.

[Lafortune and Willems, 1993] Lafortune, E. P. and Willems, Y. D. (1993). Bi-directional Path Tracing. In Santo, H. P., editor, *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal.

[Pattanaik and Mudur, 1992] Pattanaik, S. N. and Mudur, S. P. (1992). Computation of Global Illumination by Monte Carlo Simulation of the Particle Model of Light. In *Third Eurographics Workshop on Rendering*, pages 71–83, Bristol, UK.

[Shirley et al., 1996] Shirley, P., Wang, C., and Zimmerman, K. (1996). Monte carlo techniques for direct lighting calculations. *ACM Trans. Graph.*, 15(1):1–36.

[Veach, 1997] Veach, E. (1997). *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis.

[Veach and Guibas, 1994] Veach, E. and Guibas, L. (1994). Bidirectional Estimators for Light Transport. In *Fifth Eurographics Workshop on Rendering*, pages 147–162, Darmstadt, Germany.

[Veach and Guibas, 1995] Veach, E. and Guibas, L. J. (1995). Optimally combining sampling techniques for monte carlo rendering. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428, New York, NY, USA. ACM.

[Veach and Guibas, 1997] Veach, E. and Guibas, L. J. (1997). Metropolis light transport. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 65–76.

[Whitted, 1980] Whitted, T. (1980). An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349.

A Lights

A.1 Point Lights

A.1.1 Sampling point

Since we only have one point we always use the position of the point light. This gives us only one possibibility to sample the point and we therefore get:

$$PDF = 1$$

A.1.2 Sampling ray

The rays from a point ligth can go in every direction. It is therefore necessary to make a uniform sample on a sphere.

$$PDF = \frac{1}{4\pi}$$

A.2 Spot Lights

A.3 Area Lights

For et area light source skal man når man tracer mod lyset bruge og man slår op tilfældigt på lyskilden. pdf = $1/A$ Det er givet in respect to surface area. To compute the PDF with respect to solid angle we can convert it by:

$$\frac{r^2}{cos\theta}$$

hvor A er lyskildens samlet areal.

A.3.1 Sample Hemisphere

Når man går den anden vej og det er en lyskilde der sender lige meget lys i alle retninger, så skal man

$$\text{pdf} = \frac{1}{2\pi}\frac{1}{A}$$

A.3.2 Sample Cosine Hemisphere

$$\text{pdf} = \frac{cos\theta}{\pi}\frac{1}{A}$$

A.4 Infinite Area Lights

B Camera

B.1 Pinhole Camera

B.1.1 Sampling point

Since we only have one point. Although we have make sure the ray going through the hit point and the sampled point hits the viewport. going in both directions the

$$PDF = 1$$

B.1.2 Sampling ray

This is simple. Find a point on the viewport create a ray starting in the eye point, and ray.dir = Normalize(E - P). ray.origin = E.

$$PDF = 1$$