Monte Carlo and Importance Sampling for Rendering

Taehwan Kim

January 29, 2025

Abstract

Monte Carlo methods and importance sampling are fundamental techniques in rendering, enabling the simulation of light transport with high accuracy. Monte Carlo integration provides a robust framework for solving complex rendering equations by sampling light paths stochastically, while importance sampling optimizes this process by prioritizing samples that contribute more significantly to the final result. This article explores the principles and practical applications of these methods in rendering.

1 Introduction

Rendering is the process of creating realistic images by simulating how light interacts with objects in a scene. At its core lies the rendering equation, a mathematical model that captures the complexity of light transport. However, solving this equation directly is incredibly challenging due to its complexity. Monte Carlo methods offer a way forward by using random sampling to approximate the rendering equation. While powerful, these methods can introduce noise and inefficiencies. Importance sampling addresses this by focusing on the parts of the equation that matter most, improving accuracy and efficiency. In this article, we assume that the reader is already familiar with basic rendering concepts and the mathematical principles behind the rendering equation. With this foundation in mind, we will explore the application of Monte Carlo methods and importance sampling in rendering.

2 Rendering Equation

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i)\, d\omega_i \tag{1}$$

$L_o(\mathbf{x}, \omega_o)$: Radiance leaving point x in direction ω_o
$L_e(\mathbf{x}, \omega_o)$: Radiance emitted from x in direction ω_o
$\int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i)\, d\omega_i$: Radiance reflected from x in direction ω_o
The rendering equation is difficult to solve analytically. Instead, we approach it from a different perspective using Monte Carlo methods. Later, we will revisit the equation to apply the Monte Carlo approach.

3 Monte Carlo Method

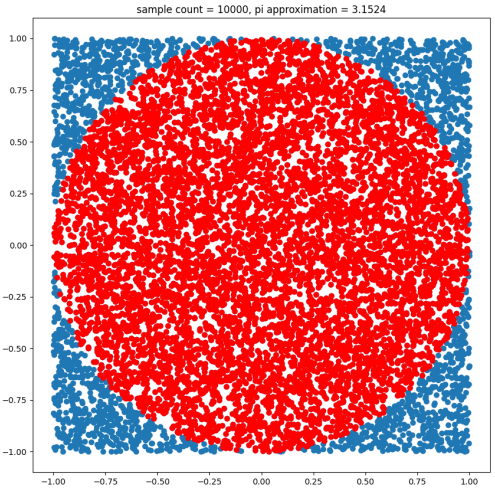
Lets first start with simple example.

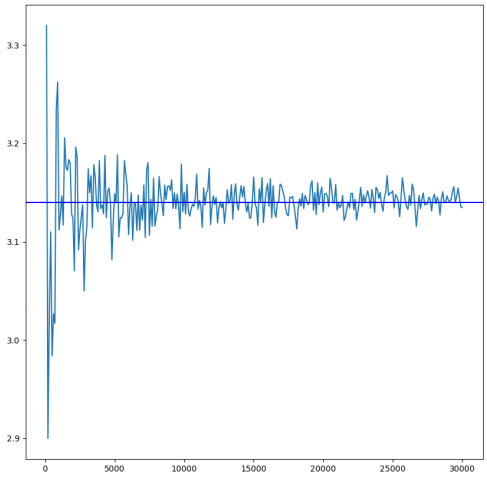
```
int samplingCount = 10000000;
int inCircleCount = 0;
int inSquareCount = samplingCount;
int radius = 50;

for (int i = 0; i < samplingCount; ++i)
{
    ... Random Uniform Sample x, y
    if (x * x + y * y <= radius * radius)
```

```
    {
        ++inCircleCount;
    }
}
float result = 4 * (inCircleCount / static_cast<float>(inSquareCount));
```

We find that if the sampling count gets larger, the result approaches to π.





When A_c is area of circle and A_s is area of square,

$$
\begin{aligned}
A_c &= \pi r^2 \\
A_s &= (2r)^2 \\
\frac{A_c}{A_s} &= \frac{\pi r^2}{4r^2} = \frac{\pi}{4} \\
\pi &= 4 \cdot \frac{A_c}{A_s}
\end{aligned}
$$

This means that the probability of a randomly sampled point (x, y) falling inside the circle is approximately $\frac{\pi}{4}$. By multiplying this probability by 4, we can estimate the value of π.

This approach demonstrates the essence of a Monte Carlo simulation. It is not designed to provide an exact or perfectly accurate result but rather to deliver an approximation that is sufficiently accurate while being computationally efficient.

3.1 Expected value

Then now, we have to use some mathematics to deeply use this idea. The Expected Value $E_{x\sim P}[f(x)]$ of a function $f(x)$ is defined as a average value of the function over some distribution of values $P(x)$ over it's domain D. It is defined as

$$E_{x\sim P}[f(x)] = \int_D f(x)P(x)dx$$

Suppose that we want to evaluate a 1D integral $\int_a^b f(x)dx$ by uniform random variables $X_i \in [a, b]$, the expected value is

$$F_n = \frac{b-a}{n}\sum_{i=1}^{n} f(X_i)$$

$E[F_n]$. When uniform PDF of $[a, b]$ is $P(x) = \frac{1}{b-a}$ (We will talk about this in next chapter),

$$\begin{aligned}
E[F_n] &= E\big[\frac{b-a}{n}\sum_{i=1}^{n} f(X_i)\big] \\
&= \frac{b-a}{n}\sum_{i=1}^{n} E\big[f(X_i)\big] \\
&= \frac{b-a}{n}\sum_{i=1}^{n}\int_a^b f(x)P(x)dx \\
&= \frac{b-a}{n}\sum_{i=1}^{n}\int_a^b f(x)\frac{1}{b-a}dx \\
&= \frac{1}{n}\sum_{i=1}^{n}\int_a^b f(x)dx \\
&= \int_a^b f(x)dx
\end{aligned}$$

We can generalize from uniform PDF to more unrestricted PDF by

$$\begin{aligned}
X_i &\sim P \\
F_n &= \frac{1}{n}\sum_{i=1}^{n}\frac{f(X_i)}{P(X_i)} \\
E[F_n] &= E\big[\frac{f(X_i)}{P(X_i)}\big] \\
&= \frac{1}{n}\sum_{i=1}^{n} E\big[\frac{f(X_i)}{P(X_i)}\big] \\
&= \frac{1}{n}\sum_{i=1}^{n}\int_a^b \frac{f(x)}{P(x)}P(x)dx \\
&= \frac{1}{n}\sum_{i=1}^{n}\int_a^b f(x)dx \\
&= \int_a^b f(x)dx
\end{aligned}$$

4 Importance Sampling

First, it is important to clarify the purpose of "Importance Sampling." The core idea is to sample not uniformly across the interval $[a, b]$, but rather with a weighted distribution that reflects the importance of different regions. This approach is useful because some areas contribute very little to the approximation of the average and can be sampled less frequently to improve efficiency.

4.1 Probability Density Funciton (PDF)

The probability density function (PDF) is a continuous function that represents the density of probabilities. By integrating the PDF over a given interval, we can determine the probability for that interval. Since it is a density function, the total area under the curve is always equal to 1.

To build intuition, let's start with a discrete function $D(x)$. Imagine dividing data into bins, where each bin has a data count b_i and a width w_i. The density for each bin can be approximated by dividing the count b_i by its width w_i.

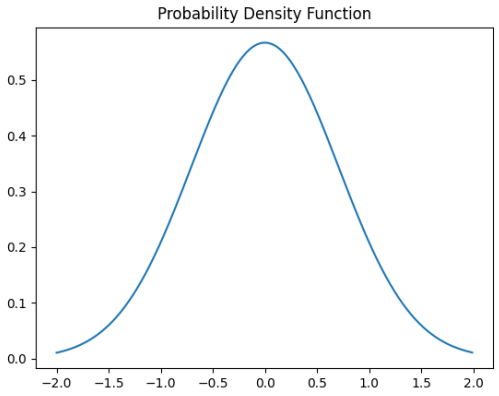
$$\textbf{total } t = \sum b_i$$
$$\textbf{density of } b_i = d_i = \frac{b_i}{t}$$
$$\textbf{bin density } = \frac{d_i}{w_i}$$
$$P(x) = \lim_{N \to \infty} D(x)$$

As the number of bins N approaches infinity and their widths approach zero, the discrete function transitions into a continuous probability density function, $P(x)$. This explains why dividing by w_i is necessary—it ensures we are calculating density per unit width.

$$\textbf{Probability}(x | a \leq x \leq b) = \int_a^b P(x)dx$$

It's important to note that calculating the probability at an exact point x is not meaningful because, with infinite bins, the probability at a single point becomes zero. Instead, probabilities derived from the PDF are always calculated over a finite interval.

For a more rigorous definition of the probability density function and its related concepts, there are many detailed articles available. However, this intuitive explanation should suffice for now.



4.2 Cumulative Density Funciton (CDF)

Cumulative Density Function also follows similar properties with PDF.

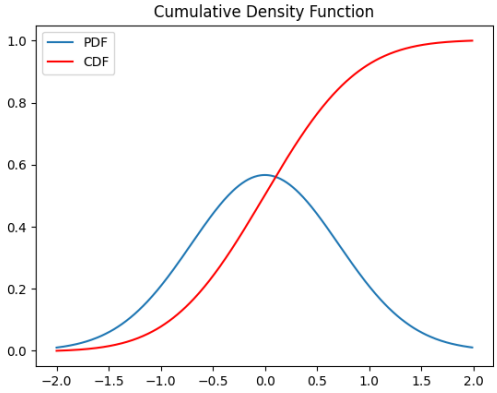
$$C(x) = \int_{-\infty}^{x} P(x')dx'$$
$$C(\infty) = 1$$
$$C(-\infty) = 0$$

So we can define PDF as this way.

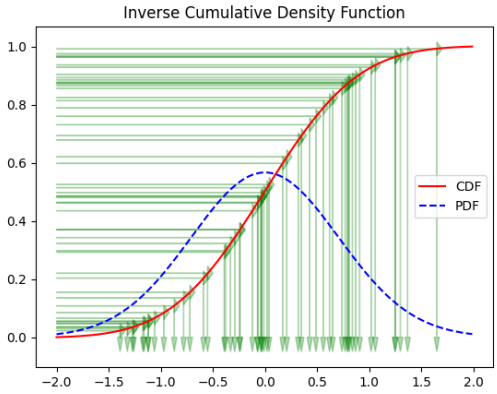
$$P(x) = \frac{dC(x)}{dx}$$

4.3 Inverse Cumulative Density Function

Inverse of Cumulative Density Function can be used to get random variable under probability of PDF. Basic step will be,



1. Get CDF $C(x)$ from PDF $P(x)$
2. Get Inverse CDF $I(x)$ from $C(x)$
3. Get random variable from $I(x)$ under probability of $P(x)$



4.4 Connecting With Monte Carlo

For example, if we have function $f(x) = 2x$

```
import random

def f(x):
    return 2 * x

N = 100000
sum = 0
a = 0
b = 5
for i in range(N):
    rand = random.uniform(a, b)
    sum += f(rand)

print((b - a) * (sum / N))
```

and it appears to be $25.01290928329435 \approx 25 = \int_0^5 2x dx$. We just sampled with uniform distribution but it's implicit, so let's make it explicit. When PDF $P(x)$ is uniform under $[0, 5]$

$$\begin{aligned} P(x) &= c \\ \int_0^5 c dx &= 1 \\ 5c &= 1 \\ c &= \frac{1}{5} \end{aligned}$$

Then CDF should be

$$\begin{aligned} C(x) &= \int_{-\infty}^{x} P(x')dx' \\ &= \int_0^x \frac{1}{5} dx' \\ &= \frac{1}{5}\Big|_0^x \\ &= \frac{x}{5} \end{aligned}$$

So Inverse CDF is

$$\begin{aligned} C(x) &= \frac{x}{5} \\ y &= \frac{x}{5} \\ 5y &= x = I(y) \end{aligned}$$

And the monte carlo method was

$$F_n = \frac{1}{n} \sum_{i=1}^{n} \frac{f(X_i)}{P(X_i)}$$

$$E[F_n] = \int_a^b f(x)dx$$

Let's apply this information to our code.

```
import random

a = 0
b = 5

def f(x):
    return 2 * x
def I(x):
    return 5 * x
def P(x):
    return 1 / (b - a)

N = 100000
sum = 0

for i in range(N):
    px = I(random.uniform(0, 1))
    sum += f(px) / P(px)

print(sum / N)
```

The result is 25.00189492830597. This structure also can be used to integrate other functions. The $P(x)$ and $f(x)$ similar it gets the accuracy and efficiency will go higher.

5 Applying to Rendering

We use the Monte Carlo method to approximate integration and refine the probability density function (PDF) for faster and more efficient computations. However, in rendering, our goal is to apply this approach in a 3D setting. Let's explore how we extend Monte Carlo sampling from one dimension to higher dimensions.

5.1 Monte Carlo on Unit Sphere

Suppose we want to integrate the function $f(\theta, \phi)$ over the surface of the unit sphere, where θ and ϕ represent both angles and directions. To apply the Monte Carlo method, we need to define PDF $P(x)$, that describes the distribution of sampled directions.

For a uniform random variable, the PDF must be $\frac{1}{\textbf{area}}$. On a unit sphere, this corresponds to $\frac{1}{4\pi}$. Other PDFs will be introduced in the following sections.

5.2 Monte Carlo on Rendering Equation

Let's revisit 1, the rendering equation.
$\int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i)\, d\omega_i$: Total integrated value over hemisphere Ω will be the radiance reflected from x in direction ω_o.

$f_r(x, \omega_i, \omega_o)$ has the reflection distribution, called *Bidirectional Reflectance Distribution Function* (BRDF).

Applying Monte Carlo method on the rendering equation is easier than expected. When $\omega_i \sim P$,

$$\begin{aligned} L_o(\mathbf{x}, \omega_o) &= L_e(\mathbf{x}, \omega_o) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i)\, d\omega_i \\ &= L_e(x, \omega_o) + E[f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i) \frac{1}{P(\omega_i)}] \end{aligned}$$

An actual applied equation in implementation will change based on the BRDF and PDF we choose.

5.3 Importance Sampling in Rendering

We use importance sampling to focus on more significant directions, such as small light sources or the most influential directions for the current surface.

If we sample uniformly across all hemispherical directions, the probability of selecting a small light source is low. As a result, even if the light is strong, its contribution to the scene may be diminished, leading to unintended results.

Now, let's compare uniform sampling and importance sampling in the context of a Lambertian BRDF. $f_r(\mathbf{x}, \omega_i, \omega_o) = \frac{p_d}{\pi}$ and PDF $P(\omega_i) = \frac{1}{2\pi}$.

$$\begin{aligned} E[f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i) \frac{1}{P(\omega_i)}] &= E[L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i) \frac{p_d}{\pi} 2\pi] \\ &= 2p_d E[L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i)] \end{aligned}$$

Lambertian BRDF with importance sampling will be, PDF $P(\omega_i) = \frac{(\mathbf{n} \cdot \omega_i)}{\pi} = \frac{\cos\theta}{\pi}$ for cos weighted hemisphere random sampling.

$$E[L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i) \frac{p_d}{\pi} \frac{\pi}{(\mathbf{n} \cdot \omega_i)}] = p_d E[L_i(\mathbf{x}, \omega_i)]$$

5.4 Direct Light Sampling

We saw how to use uniform and cos weighted PDF for importance sampling. But we just "steered" the direction to "usually" effective angles, not the exact light direction.

We will still use our $\frac{f(x)}{P(x)}$ structure, so we need to get the PDF of the light.

To sample uniformly on the light with an area of A, the PDF $P(x)$ on that surface will be $\frac{1}{A}$. But we are working on the hemisphere area, so let's get $P(\omega)$.

When p is on the surface and q is on the light,

$$d\omega = \frac{dA \cdot \cos\theta}{\mathbf{distance}^2(p,q)}$$

Since $P(q) \cdot dA$ and $P(\omega) \cdot d\omega$ is same because $d\omega$ is just dA projected on hemisphere,

$$\begin{aligned}
P(\omega) \cdot d\omega &= P(q) \cdot dA \\
P(\omega) \cdot \frac{dA \cdot \cos\theta}{\mathbf{distance}^2(p,q)} &= P(q) \cdot dA \\
P(q) &= \frac{1}{A} \\
P(\omega) \cdot \frac{dA \cdot \cos\theta}{\mathbf{distance}^2(p,q)} &= \frac{dA}{A} \\
P(\omega) &= \frac{\mathbf{distance}^2(p,q)}{A \cdot \cos\theta}
\end{aligned}$$

Of course, it's not always the best approach to sample light directly because we also need to account for ambient light—reflection from other surfaces that aren't light sources themselves.

5.5 Mixing PDF

Adding PDF with another PDF with weights will result a new Mixed PDF.

$$\begin{aligned}
&P_m(x) = a_0 P_0(x) + a_1 P_1(x) + a_2 P_2(x) + \cdots + a_{N-1} P_{N-1}(x) \\
&\sum_{i=0}^{N} a_i = 1
\end{aligned}$$

We can use this to mix cos weighted PDF and Direct Light PDF, and more.

6 Conclusion

We've explored the Monte Carlo method and importance sampling as techniques to approximate the integration in the rendering equation. Once the fundamental concepts are understood, the process becomes quite intuitive.

I hope there will be an opportunity to discuss topics like Multiple Importance Sampling and ReSTIR in the future as well.

Basic ideas was from [PS24], and [PJH16]

References

[PJH16] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2016.

[PS24] Steve Hollasch Peter Shirley, Trevor David Black. Ray tracing: The rest of your life, August 2024. `https://raytracing.github.io/books/RayTracingTheRestOfYourLife.html`.