Vanden Heede Pauline

A study of variance reduction techniques in the Monte Carlo integration for simulating global illumination

Graduation work 2021-2022

Digital Arts and Entertainment
Howest.be

Contents

Abbreviations

Abbreviation	Full name
BSSRDF	Bidirectional Surface Scattering Reflectance Distribution Function
BRDF	Bidirectional Reflectance Distribution Function
BTDF	Bidirectional Transmission Distribution Function
BSDF	Bidirectional Scattering Distribution Function
CRV	Continuous Random Variable
DRV	Discrete Random Variable
PDF	Probability Density Function || Probability Distribution Function
UDF	Uniform Density Function || Uniform Distribution Function
CDF	Cumulative Density Function || Cumulative Distribution Function
ICDF	Inverse Cumulative Density Function || Inverse Cumulative Distribution Function
2D	Two-dimensional
3D	Three-dimensional
4D	Four-dimensional
i.d.d.	independent and identically distributed
aka	also known as
SPP	Samples Per Pixel
cw	clockwise
ccw	counter clockwise
MSE	Mean-Squared Error
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index
MS-SSIM	Multi-Scale Structural Similarity Index

Table 1: Abbreviations.

Quantities

Quantity name	symbol	Units
Radiant flux or power	Φ	Watt [W]
Angle	θ	Radians
Solid angle	ω	Steradians(sr)
Irradiance	E	$\left[\frac{W}{m^2}\right]$
Radiosity or radiant existance	B or M	$\left[\frac{W}{m^2}\right]$
Radiant Intensity	I	$\left[\frac{W}{sr}\right]$
Radiance	L	$\left[\frac{W}{sr\,m^2}\right]$
BSSRDF (table 1 Abbreviations.)	$S(p_i, \omega_i, p_o, \omega_o)$	sr^{-1}
BRDF (table 1 Abbreviations.)	$f_r(p, \omega_i, \omega_o)$	sr^{-1}
BTDF (table 1 Abbreviations.)	$f_t(p, \omega_i, \omega_o)$	sr^{-1}
BSDF (table 1 Abbreviations.)	$f(p, \omega_i, \omega_o)$	sr^{-1}
PDF (table 1 Abbreviations.)	$p(x)$	Dimensionless
Expected value	E	Takes on unit of parameter
Variance	σ^2	Takes on unit of parameter
Standard deviation	σ	Takes on unit of parameter
Mean squared error	MSE	Dimensionless
Peak Signal-to-Noise Ratio	PSNR	dB
Signal-to-Noise Ratio	SNR	dB
Structual Similarity Index	SSIM	Dimensionless

Table 2: Quantities.

List of Figures

List of Tables

List of Algorithms

1 Abstract

Producing an image that simulates the effect of global illumination with a relative small amount of noise is a difficult task when it comes to having relatively short render times. It states the need for knowing the effectiveness of variance reduction techniques. Variance reduction techniques try to reduce the noise in the image without having to increase the amount of samples the rendering algorithm needs to produce an image. The technique that will be used for the rendering algorithm is a combination of the Monte Carlo integration and path tracing. Then the variance reduction techniques, stratified and importance sampling, will be introduced. At the end it will be proven that these variance reduction techniques are effective in rendering images with less noise without needing extra samples and with a similar render time.

2 Introduction

Yielding images that are physically plausible and visually pleasing with respect to global illumination is the ultimate goal of this research. For this there is a need to take a deeper dive into the concepts of light and the techniques behind the rendering algorithm. It is of great importance to understand the concepts behind the reality. It is needed to have a better understanding of the following topics:

- A model to capture light to be able to display it on an image.
- A model for materials with which the light interacts.
- A model to capture the objects in the scene.

To be able to have a visually pleasing image without having a substantial render time, we need a bigger understanding of these variance reduction techniques which uses sample theory. It is stated that global illumination is not difficult to achieve and that the algorithm is relatively simple in it's complexity, in spite of being challenging to work efficiently with respect to the time it takes to effectively render an image.

The goal of this research is to obtain an optimal sampling technique so that the noise in the image is significantly less with similar render times, thus creating the research questions.

"Is it possible to create a less noisy image with similar render times, using different variance reduction techniques?"
"How effective are these variance reduction techniques?"

To compare images with each other, there are certain techniques that can be used. They all fall under the same group of image quality metrics. But to be able to tell something about the effectiveness of these reduction techniques we also need a way to express how much time it takes on average to render an image. For this, bench marking will be used. Combining these two gives a well defined answer to the research questions stated above.

3 Background - Literature Study

To the literature study, there are three big components to understand before we can start discussing the case study. The first big component is global illumination, which will briefly go over the core concept of illumination and the properties it needs to be calculated correctly. The second big component of this paper is the Monte Carlo integration; this will explain the basis of statistics and the integration into the rendering equation. The last component is sampling; this will brush up on the basics of simple sampling theory and will discuss the variance reduction techniques.

3.1 Global illumination

In order for objects to be visible in images, there must be a source of light so that light can be reflected from them to the sensor(e.g. the eyes, camera, ...). Illumination describes how the world gets lit up in it's varying degrees of colour. More specific to computer science; illumination is a set of algorithms that tries to describe how lighting in the real world works, or at least how sensors perceive it.
When we see light, we see it being emitted by light sources. That light will be bounced off a surface to reach our eye. When that light bounces only once before reaching the eye, we speak of direct illumination or local illumination (Fig. 3.1, left) [7]. But if that light bounces multiple times before reaching the eye, then we have the effect of indirect illumination (Fig. 3.1, right). To simulate global illumination we need the effect of both illumination models [10].

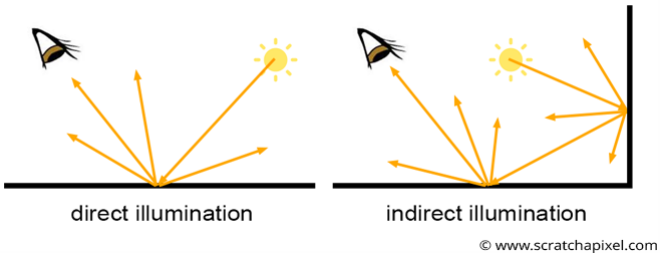

Figure 3.1: Illumination models [10].

Understanding some of the physical processes involved is helpful for accurately modelling light sources for rendering. So to be able to describe the natural phenomenon of light in code, a more thorough explanation about light and materials is needed. A more in-depth explanation of light and how it is expressed is given, but also how it travels through a scene and what influence materials have over the light scattering in the scene.

3.1.1 The basics of light

In the physical world we have many different types of light sources that have been invented to convert energy into emitted electromagnetic radiation. The rendering algorithm will display this radiation. The quanta of electric radiation are photons, by the means of which the radiation is propagated through a medium. These photons bounce around in the world and at each collision they are either absorbed, reflected or refracted [2]. But as sensors (e.g. the eye, camera) can only see the visible light, the rendering algorithm will only support a portion of the electromagnetic spectrum (Fig. 3.2).

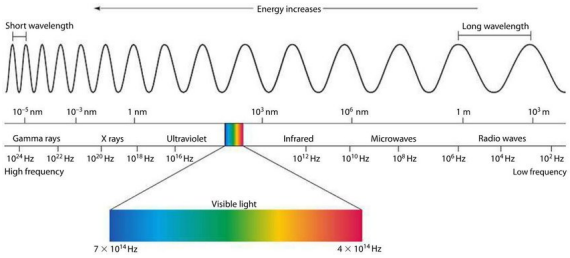

Figure 3.2: Schematic representation of light.

Photons can be seen as both particles and waves depending on how they are observed, this establishes the dual nature of light. Because our simulation is an approximation, we need models to describe light and how it behaves. Due to the dual nature there are many models available, with the main ones being physical optics, geometric optics and quantum optics [2].

3.1.2 Optics

The wave and quantum part of light are often ignored in rendering algorithms. This is the reason why geometric optics are the choice for this research. The model builds on three core assumptions; light is emitted, reflected and transmitted [8]. Outside of these three core assumptions, three other conditions are made about the behaviour of light [8].

- **Light travels in straight lines.** Effects like diffraction and interference are not taken into account. When light diffracts it means that the light bends along the path it travels. When this is ignored, it implies that light can easily be modeled as rays, as seen in section 4.1 The rendering algorithm.
- **Light travels instantaneously through a medium.** Light travels at infinite speed. This is a practical assumption to make global illumination algorithms easier, as it means that the rendered image can be captured immediately.
- **Light is not influenced by external factors.** The energy is conserved along its travel path.

3.1.3 Radiometry

Radiometry is the field of study that is involved in the measurement of light. Radiometric quantities exist for measuring the aspects of electromagnetic radiation; overall energy, power and power density with respect to area, direction or both [2]. To be complete, a brief summary is given of the most important quantities concerning this research (this information being an aggregate of [2], [4], [8], [19], [23], [25] sorted on contribution).
First of all we have flux or radiant power (Φ, table 2 Quantities.) . This is the flow or radiant energy over time.

$$\Phi = \frac{E}{dt} \tag{3.1}$$

Before any further explanation can be given, a small introduction to area measurements is needed. Firstly we have the angle (Fig. 3.3, left, table 2 Quantities.), which is the size of a continuous set of directions in a plane. Radians are defined as the length of the arc this set of direction intersects on, on an enclosing circle with radius 1. Secondly, we have the solid angle (Fig. 3.3, right, table 2 Quantities.), being the three-dimensional (3D, table 1 Abbreviations.) extension of the concept of an angle. Which is the size of a continuous set of directions in a 3D space. Steradians are defined by the area of the intersecting patch on an enclosing sphere with radius 1.

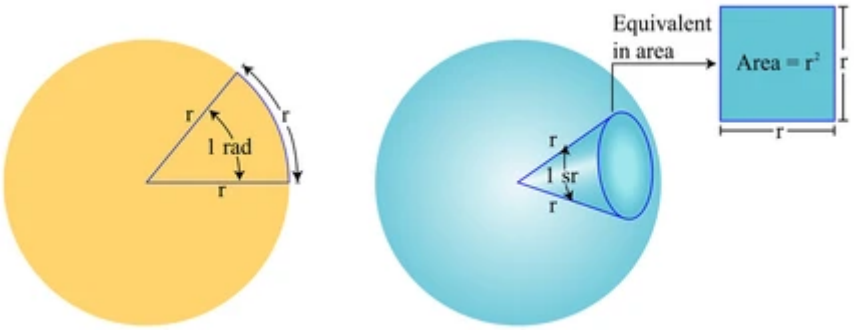

Figure 3.3: Radian and steradian.

Now that the area measurements are known, flux can be observed with respect to the surface area (Eq. 3.2) or with respect to the direction (solid angle) (Eq. 3.3).

- Irradiance (E, table 2 Quantities.), the indicent flux on a surface per unit surface area.
- Radiosity (B, table 2 Quantities.) or radiant existance (M), the exitant flux on a surface per unit surface area.

$$E = M = B = \frac{d\Phi}{dA} \tag{3.2}$$

- Radiant intensity (I, table 2 Quantities.), the flux density per unit solid angle.

$$I = \frac{d\Phi}{d\omega} \tag{3.3}$$

When both are combined together, we get Radiance (L, table 2 Quantities.). Radiance is the density or radiant flux with respect to both area and solid angle, otherwise called the measure of electromagnetic radiation in a single ray [2]. The area is measured in a plane perpendicular to the ray (Eq. 3.4). If this isn't the case the cosine factor is used (Eq. 3.5).

$$L(p,\omega) = \frac{d^2\Phi}{d\omega dA^{\perp}} \tag{3.4}$$

$$L(p,\omega) = \frac{d^2\Phi}{d\omega dAcos\theta} \tag{3.5}$$

The radiance leaving point p in direction ω is often written as $L_o(p,\omega)$, and the radiance coming into a point p in direction ω is often written as $L_i(p,\omega)$. The ray direction always points away from x as it is convenient for the calculations in section 4.3 Ray intersection at p. 32.
Because sensors are sensitive to energy (see section 3.1.1 The basics of light at p.13) they are also sensitive to radiance. This being the exact reason why the purpose of evaluating the shading equation is to compute the radiance along a given ray, which will be explained in more detail in the next paragraph.

3.1.4 Light transport

The goal of the rendering algorithm is to simulate the physics of the real world and to generate a realistic and accurate image [25]. In the rendering algorithm radiance is the final quantity computed. It ultimately describes the equilibrium or steady-state distribution of radiance in the scene [8]. First the flow of light particles or photons need to be described, for which transport theory is used [2]. It described how to formulate radiometric quantities discussed in section 3.1.3 Radiometry in terms of light flow.
Because the same assumptions that were made as in section 3.1.3 Radiometry and section 3.1.2 Optics about light are followed, light transport can be described as follows: when light travels, it follows a path from the sun to a surface (e.g. a rock, floor, tree) and it will bounce off that surface towards another surface or towards the observer or sensor [7]. The ray is a useful abstraction for approximating of the path along which light travels (due to these assumptions)[25]. The radiance along this ray is invariant and propagates instantaneously (section 3.1.2 Optics), which can only be achieved if the conservation of energy is enforced. These two properties make it possible to write the incident radiance as the outgoing radiance from another point p' in the opposite direction (Eq. 3.7) [19].

$$L_i(p,\omega) = L_o(p',-\omega) \tag{3.6}$$

$$L_i(p,\omega) = L_o(t(p,\omega),-\omega) \tag{3.7}$$

With $t(p,\omega)$ being a ray-cast function that computes the first surface point p' intersected by a ray from p in direction ω. If this function doesn't find an intersection, it returns a value for which the right hand side $L_o(t(p,\omega),-\omega)$ returns zero. If the conservation of energy is enforced at the surface, the total reflected radiance can be described by it's three major components.

- Radiance emitted at the point on the surface;
- The bidirectional scattering distribution function (BSDF, table 1 Abbreviations., section 3.1.5 Interaction of light with materials);
- The distribution of incident radiance arriving at the point.

Mathematically, the total reflected or exitant radiance can be written as the sum of the emitted radiance and the fraction of incident radiance (Eq. 3.8) [19]. The fraction of incident radiance is affected by its geometry and scattering properties, therefore the usage of the BSDF $f(p,\omega_o,\omega_i)$ is needed. It is also affected by how the light is oriented towards the surface, for which the cosine term discussed in section 3.1.3 Radiometry is used. Due to Eq. 3.7 we can simplify Eq. 3.8 to Eq. 3.9.

$$L_o(p,\omega_o) = L_e(p,\omega_o) + \int_{\delta^2} f(p,\omega_o,\omega_i) L_i(p,\omega_i) |cos\theta_i| \, d\omega_i \tag{3.8}$$

$$L(p,\omega_o) = L_e(p,\omega_o) + \int_{\delta^2} f(p,\omega_o,\omega_i) L(t(p,\omega_i),-\omega_i) |cos\theta_i| \, d\omega_i \tag{3.9}$$

This equation however, is impossible to solve analytically, as the radiance comes back on both the left hand side and right hand side of the equation. Nonetheless, with the combination of ray-tracing algorithms and the Monte Carlo integration, there is a powerful tool in existence to still arrive at a good estimation. This due to the fact that the transport equation naturally leads to recursive solutions, of which the Monte Carlo integration is one [25].

3.1.5 Interaction of light with materials

When light hits the surface, this surface reflects, transmits or emits light back into the environment [19]. The light energy and outgoing radiance interacts differently depending on the surface. Not only does the light act differently depending on the material or surface, the material also acts differently according to the light conditions [8]. The outgoing radiance is a property of the surface and thus of its microscopic configuration [18].
One of the first to describe this phenomenon with a mathematical equation was Nicodemus. He found that this interaction could be described by the Bidirectional Surface Scattering Reflectance

Distribution Function, also called the BSSRDF (table 1 Abbreviations.). The BSSRDF is general in its assumptions; it only takes into account the assumptions made in section 3.1.2 Optics together with ignoring some spatial parameters [17]. The BSSRDF describes the relation between the incident and reflected light [8]. Specifically, it expresses the connection between the radiance leaving p_o along the direction ω_o and the incoming or incident flux arriving at p_i in the direction ω_i [17].

$$S(p_o, \omega_o, p_i, \omega_i) = \frac{dL_o(p_o, \omega_o)}{d\Phi(p_i, \omega_i)} \tag{3.10}$$

To know the outgoing radiance, Eq. 3.10 can be written as:

$$dL_o(p_o, \omega_o) = S(p_o, \omega_o, p_i, \omega_i) d\Phi(p_i, \omega_i) \tag{3.11}$$

To be able to calculate the outgoing radiance, Eq. 3.11 needs to be integrated both over the surface area A and over all possible directions, in this case; the upper hemisphere Ω.

$$L_o(p_o, \omega_o) = \int_A \int_\Omega S(p_o, \omega_o, p_i, \omega_i) L_i(p_i, \omega_i) |cos\theta_i| \, d\omega_i \, dA \tag{3.12}$$

The BSSRDF takes subsurface scattering into account, this phenomenon happens when light gets reflected by translucent surfaces [19]. Because this is outside of the scope, the BSSRDF will be simplified to a Bidirectional Scattering Distribution Function (BSDF, table 1 Abbreviations.). This simplification is done by making the assumption that the incoming light leaves at the same point [8]. The BSDF $f(p, \omega_o, \omega_i)$ is composed by two other distribution functions, namely the Bidirectional Reflectance Distribution Function (BRDF, table 1 Abbreviations.) and the Bidirectional Transmittance Distribution Function (BTDF, table 1 Abbreviations.) [19].

Bidirectional Reflectance Distribution Function (BRDF)

The BRDF describes how much radiance is leaving the surface $L_o(p, \omega_o)$ towards the viewer as a result from the incoming radiance $L_i(p, \omega_i)$ (Eq. 3.13, see Fig. 3.4) [19].

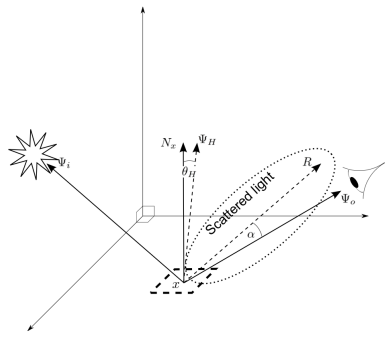

Figure 3.4: Representation of the BRDF [9].

The BRDF is a concentration of reflectance per steradians, so it can take any value in the interval $[0, \infty]$ [17].

$$f_r(p, \omega_o, \omega_i) = \frac{dL_o(p, \omega_o)}{dE(p, \omega_i)} \tag{3.13}$$

From section 3.1.3 Radiometry it is known that the irradiance can be written as the radiance integrated over the solid angle.

$$dE(p, \omega_i) = L_i(p, \omega_i) cos\theta_i \, d\omega_i \tag{3.14}$$

Substituting Eq. 3.14 in Eq. 3.13 results in Eq. 3.15, thus establishing the relation between the incoming and outgoing radiance.

$$f_r(p, \omega_o, \omega_i) = \frac{dL_o(p, \omega_o)}{L_i(p, \omega_i) cos\theta_i \, d\omega_i} \tag{3.15}$$

To get the radiance due to the incident illumination at p from all directions, Eq. 3.15 needs to be integrated, resulting in a version of the rendering equation described by Kajiya.

$$L_o(p, \omega_o) = \int_\Omega f(p, \omega_o, \omega_i) L_i(p, \omega_i) |cos\theta_i| \, d\omega_i \tag{3.16}$$

To be physically correct, the BRDF has a number of properties where it needs to hold itself to [8]. This has been tested by Duvenhage [9] for the three conditions positivity, symmetry or reciprocity and conservation of energy.

1. **Range.** The BRDF returns a positive value that can vary with wavelength.
2. **Dimension.** The BRDF is a 4D function that is defined at each point. Two dimensions correspond to the incoming direction and the other two dimensions correspond to the outgoing direction.
3. **Reciprocity (HelmHoltz reciprocity) or Symmetry.** This property states that the value remains unchanged if the incident and exitant directions are switched (Eq. 3.17). This means that reversing the direction of light does not change the amount of light that gets reflected.

$$\forall \omega_o, \omega_i : f_r(p, \omega_o, \omega_i) = fr(p, \omega_o, \omega_i) \tag{3.17}$$

4. **Relation between incident and reflected radiance.** The value of the BRDF is not dependent on the possible presence of irradiance along other incident angles. This means that the function is linear with respect to all incident directions.
5. **Energy conservation.** The law of conservation of energy requires that the total amount of power reflected over all directions must be less than or equal to the total amount of power

incident on the surface (Eq. 3.18). The BRDF only satisfies this constraint if the total energy reflected ≤ energy of incident light ($M \leq E$) (see section 3.1.2 Optics) [19].

$$\forall \omega_i : \int_\Omega f_r(p, \omega_i, \omega_o) cos\theta_i \, d\omega_o \leq 1 \tag{3.18}$$

Bidirectional Transmittance Distribution Function (BTDF)
The BTDF describes the distribution of transmitted light. The BTDF is very similar to the BRDF and gets written as $f_t(p, \omega_o, \omega_i)$. The only difference to the BRDF is that the BTDF does not obey the reciprocity property. This due to the fact that emitted light gets generated by the surface and thus will not be true to the conservation of energy.

Empirical models
To characterise the BRDF, empirical models are used. For simplicity the focus will be on the diffuse example of the BRDF (due to scope limits and time constraints).

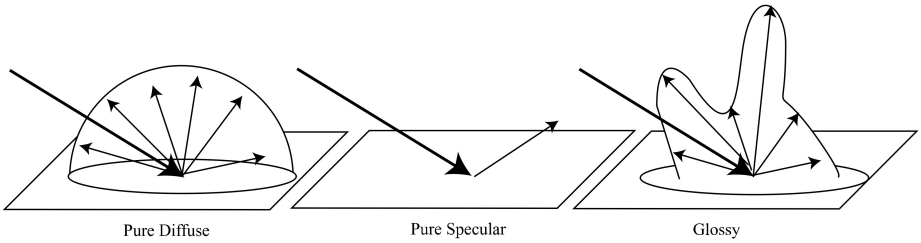

Figure 3.5: Different BRDF models [8].

For diffuse materials the light is reflected uniformly, which denotes that the reflected radiance is independent of the exitant direction [8]. So the BRDF has a constant value for all values of ω_o and ω_i.

$$f_r(p, \omega_o, \omega_i) = \frac{\rho_d}{\pi} \tag{3.19}$$

The reflectance ρ_d is the fraction of incident energy that is reflected at a surface. For physically based materials, this varies from 0 to 1 [8]. Fig. 3.5 shows multiple models; diffuse, pure specular and glossy materials. Specular materials only reflect or refract light in one specific direction and glossy materials are a combination of both the diffuse model and the specular model.

3.2 Randomised algorithms

Randomised algorithms can be sorted into two categories; Las Vegas and Monte Carlo. Las Vegas is an algorithm category that is always correct compared to Monte Carlo, which is correct on average [19]. Due to the nature of our problem we will focus on the Monte Carlo algorithm. Before going into the specifics of the Monte Carlo technique, a brief overview of some statistical definitions is given.

3.2.1 The basics of statistics

There is a difference between a continuous random variable and a discrete random variable. A random variable is called discrete when it can take on a finite number of positive values. The counterpart being a continuous random variable is when the variable can take on any value in $\mathbb{R}$ on the real line with a default domain $[-\infty, \infty]$ [14], [19].
The behaviour of this variable x is described by the distribution of the value [4]. This distribution is also called the Probability Density Function (PDF, table 1 Abbreviations.). The PDF returns the relative probability of a random variable taking on a particular variable [4]. The result of this function $p(x)$ lies in the domain $[0, 1]$ and it can be defined as:

$$F(x) \equiv P\{a\ random\ selection\ of\ X\ gives\ a\ value\ less\ than\ x\} = P\{X \geq x\} \tag{3.20}$$

When the distribution is differentiated, the pdf can be written as:

$$p(x) \equiv \frac{dP(x)}{dx}. \tag{3.21}$$

For a canonical random variable x [19],

$$p(x) = \begin{cases} 1 & x \in [0, 1) \\ 0 & otherwise \end{cases}, \tag{3.22}$$

the PDF has the normalisation property;

$$\int p(x)\, dx = P(\infty) = 1. \tag{3.23}$$

For a uniform random variable, the PDF becomes a constant value [4], [19]. With the PDF defined like above, the expected value of this PDF can be calculated. The expected value or mean is the average value over some distribution of values $p(x)$ over it's domain [14]. It is the value that a function $f(x)$ with underlying pdf $p(x)$ will take on [4].

$$E[f(x)] = \int_{-\infty}^{\infty} f(x)p(x)\, dx = \int f(x)p(x)\, dx \tag{3.24}$$

When there is only a one dimensional variable, the Eq. 3.24 can be simplified by substituting $f(x)$ for x.

$$E[x] = \int xp(x)\, dx \tag{3.25}$$

Now that we know the expected value, the variance can be calculated. The variance will quantify the error in a value estimated by the Monte Carlo algorithm (section 3.2.2 The Monte Carlo method) [14]. More mathematically, it is the expected value of the difference between x and $E[x]$ squared [4].

$$\sigma^2 = E\left[(x - E[x])^2\right] = \int (x - E[x])^2 p(x)\, dx \tag{3.26}$$

Which can be written as Eq. 3.27 after simplification

$$\sigma^2 = E[x^2] - (E[x])^2 = \int x^2 p(x)\, dx - \left(\int x p(x)\, dx\right)^2 \tag{3.27}$$

Variance has multiple properties:

- The variance of a constant random variable is zero;
- For a constant variable and a random variable,

$$\sigma^2(cx) = c^2\sigma^2(x); \tag{3.28}$$

- For independent variables x and y the variance of the sum is equal to the sum of the variance.

$$\sigma^2(x + y) = \sigma^2(x) + \sigma^2(y) \tag{3.29}$$

This last property being the stepping stone to higher dimensional problems (e.g. the rendering equation).
Now that everything about a PDF relevant to the research questions is mentioned, there is only one thing left to do and that is choosing a PDF.

3.2.2 The Monte Carlo method

"Monte Carlo methods represent the solution of a problem as a parameter of a hypothetical population, with using a random sequence of numbers to sample of the population from which statistical estimates can be accessed of the parameter."

Halton [11]

In short, it will define a random variable so that the expected value of that variable should be the solution to that problem. They become useful because they define a random variable x such that the expected value $E[x]$ would be the solution the the problem (Fig. 3.6).
Instead of directly computing the value of the integral in each shading point via quadrature rules, the integrand is evaluated at a number of random points from the domain [2].

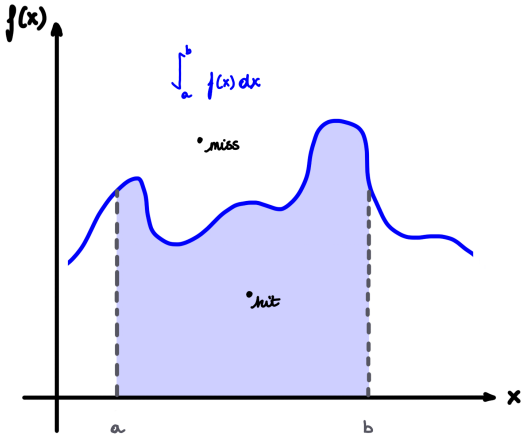

Figure 3.6: Graphical depiction of the Monte Carlo integration.

The Monte Carlo method can be split up in three general steps [8]:

1. The method will sample a variable according to the PDF;
2. It will then evaluate the function with the sample chosen in 1. ;
3. At the end it will average the results appropriately with the weight of the sampled value from 1.

Just like every method, this one also has it's advantages [14] and disadvantages [8], [19]:

+ This method has a conceptual simplicity to it. You can use a sampled random variable to get a somewhat correct answer on average.
+ The method can be applied to a wide set of problems, it scales easily to higher-dimensional problems.
- It has a rather slow convergence rate. This indicates that a lot of extra samples are needed for reducing the error or variance on the esitmation to a minimum. To ameliorate this, some variance reduction techniques will be discussed later (section 3.3.2 Stratified sampling and section 3.3.3 Importance sampling).

When we calculate the Monte Carlo integration, it will always be an approximation to the solution of the problem.

3.2.3 The Monte Carlo estimator

Say that a function is defined over the domain $[-\infty, \infty]$, then the integral of this function can be described as.

$$I = \int_{-\infty}^{\infty} f(x)\,dx = \int f(x)\,dx \tag{3.30}$$

Now, the central idea of the Monte Carlo integration is that the integral can be estimated by the sum [14].

$$\langle I \rangle = E\left[\int_{-\infty}^{\infty} g(x)f(x)\,dx\right] = \frac{1}{n}\sum_{i=1}^{m} f(x_i) \tag{3.31}$$

When these values are randomly selected over the domain with PDF $p(x)$, we can write Eq. 3.31 as.

$$\langle I \rangle = \frac{1}{n}\sum_{i=1}^{m} \frac{f(x_i)}{p(x_i)} \tag{3.32}$$

Thus, the Monte Carlo estimator tries to estimate or approximate the value of an arbitrary integral. When the samples are independent and identically distributed (i.d.d. table 1 Abbreviations.) we can write the estimator as

$$E[\langle I \rangle] = E\left[\frac{1}{n}\sum_{i=1}^{m} \frac{f(x_i)}{p(x_i)}\right] \tag{3.33}$$

$$= \frac{1}{n} n \int \frac{f(x)}{p(x)} p(x)\,dx \tag{3.34}$$

$$= \int f(x)\,dx \tag{3.35}$$

$$= I \tag{3.36}$$

With this derivation of the Eq. 3.33, it is proven that the Monte Carlo integration gives the correct answer on average. Hence the variance can be calculated accordingly.

$$\sigma^2 = \frac{1}{n}\int \left(\frac{f(x)}{p(x)} - I\right)^2 p(x)\,dx \tag{3.37}$$

$$\sigma^2 = \frac{1}{n}\int \frac{f(x)^2}{p(x)}\,dx - I^2 \tag{3.38}$$

Estimators can be categorised in two different categories: biased and unbiased estimators. The bias is the difference between the expected value of the estimator and the value of the integral. The bias for biased estimators is non-zero, and that for unbiased estimators is zero.

$$B[\langle I \rangle] = E[\langle I \rangle] - I \tag{3.39}$$

For increasing n, our bias vanishes.

$$\lim_{x \to \infty} B[\langle I \rangle] = 0 \tag{3.40}$$

3.2.4 The bridge to the global illumination algorithm

The Monte Carlo integration can be a solution to the integral that describes light scattering, aka (table 1 Abbreviations.) the rendering equation (section 3.1.4 Light transport) [19], despite the fact that the rendering equation is analytically impossible (even when the incident radiance function were available in closed form)[19]. An estimate of the reflected radiance by choosing a set of direction over the sphere is needed. Because of the Monte Carlo method's advantages (mentioned in section 3.2.2 The Monte Carlo method), the method makes for a prime solution to solve the rendering equation, as this is a higher-dimensional integral. Keeping in mind that the Monte Carlo integration will always be an approximation, it can be the most accurate answer that can be obtained given a certain investment of computer time, in spite of the random character of the answer [14].

3.3 Sampling

3.3.1 The basics of sampling theory

To be able to utilize the Monte Carlo technique, samples need to be computed from a PDF (described in section 3.2.1 The basics of statistics). To do this a PDF needs to be chosen to sample from. Starting with the most simple one; the uniform distribution function (UDF table 1 Abbreviations.). A distribution is uniform when every value in the domain has an equal probability of getting chosen (Fig. 3.7). It is not that because it is the simplest PDF that it also is the most efficient one [11].

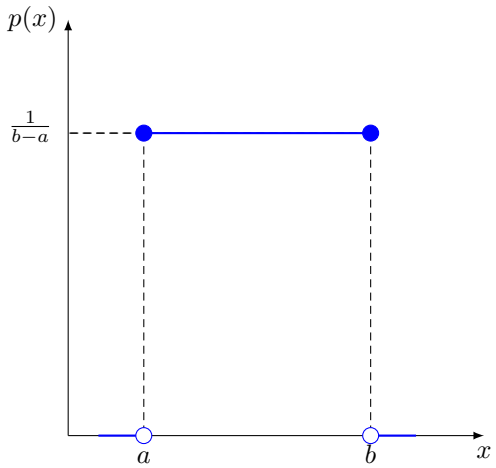

Figure 3.7: Uniform distribution function.

When sampling from a UDF, the Monte Carlo estimator will need a high number of samples to have an acceptable variance. To be able to reduce this variance, there are multiple methods designed, often called the variance reduction techniques. These techniques can be grouped in several categories [25].

- Analytically integrating a function that is similar to the integrand. This technique takes advantage of knowing what needs to be estimated. Due to the scope of this project, importance sampling will be the only technique that is discussed.
- Uniformly placing sample points across the integration domain. This somewhat ensures that samples are evenly distributed over the domain. The main technique is called stratified sampling.
- Adaptively controlling the sample density based on information gathered during the sampling. This technique places more samples where they are most useful.
- Combining the samples from two or more estimators whose values are correlated. Based on the assumption that one can find two or more estimators of which the values are correlated.

3.3.2 Stratified sampling

When the samples are unevenly distributed over the integrals domain, the estimated value will be a poor approximation of the integral. This uneven distribution of samples is otherwise called clumping of the samples [8]. This can happen due to an irrespective of the PDF, because the PDF describes the expected number of samples in a part of the domain. It also reduces the effectiveness of the Monte Carlo estimator considerably.
Stratified sampling will subdivide this domain into several non-overlapping regions, also called strata (singular; stratum). The method will be evaluating the integral over every stratum. In how many

regions the integral needs to be split is not defined, however it is beneficiary to take a number of regions so that there is only one sample needed from every subdivision. The simplest form would be that all the regions are of equal size [15].

$$\int_0^1 f(x)\,dx = \int_0^{\alpha_1} f(x)\,dx + \int_{\alpha_1}^{\alpha_2} f(x)\,dx + \cdots + \int_{\alpha_{n-2}}^{\alpha_{n-1}} f(x)\,dx + \int_{\alpha_{n-1}}^1 f(x)\,dx \tag{3.41}$$

Then we can define the variance as:

$$\sigma^2 = \sum_{i=1}^{n} \frac{\alpha_i - \alpha_{i-1}}{m_i} \int_{\alpha_{i-1}}^{\alpha_i} f(x)^2\,dx - \sum_{i=1}^{n} \frac{1}{m_i} \left(\int_{\alpha_{i-1}}^{\alpha_i} f(x)\,dx \right)^2 \tag{3.42}$$

When the assumptions that were discussed earlier are true, we can write Eq. 3.42 as:

$$\sigma^2 = \sum_{i=1}^{n} \frac{1}{N} \int_{\alpha_{i-1}}^{\alpha_i} f(x)^2\,dx - \sum_{i=1}^{n} \left(\int_{\alpha_{i-1}}^{\alpha_i} f(x)\,dx \right)^2 \tag{3.43}$$

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{n} \int_0^1 f(x)^2\,dx - \sum_{i=1}^{n} \left(\int_{\alpha_{i-1}}^{\alpha_i} f(x)\,dx \right)^2 \tag{3.44}$$

Because the latter term is always larger or equal to the I^2-term in Eq. 3.38, we can conclude that the variance is always less than or equal to a pure Monte Carlo sampling scheme [8], [15].

$$\sigma^2_{strat} \leq \sigma^2_{normal} \tag{3.45}$$

This is only true if the strata have different means (e.g. non-overlapping regions).
The cost of distributing the samples over the strata is negligible compared to the advantages it has, making stratified sampling a useful, inexpensive variance reduction technique [25] and with extension a fundamental optimisation [15].

3.3.3 Importance sampling

Choosing $p(x)$ intelligently is called importance sampling. When samples are generated without any knowledge of the context, there might be a lot of samples that don't or barely contribute to the estimated value. To solve this problem, importance sampling takes into account that some regions might be more important than others, because they have higher function values [15]. If $p(x)$ is large where g is large, there will be more samples in important regions. This somewhat solves the fundamental problem of Monte Carlo integration, which is diminishing returns [4].
To achieve this, the non uniform PDF is used to generate samples. Therefore a PDF that closely resembles the formula that needs to be integrated is chosen (Fig. 3.8) [8]. Due to this, it is logical that the graphs are ordered in decreasing variance from left to right.

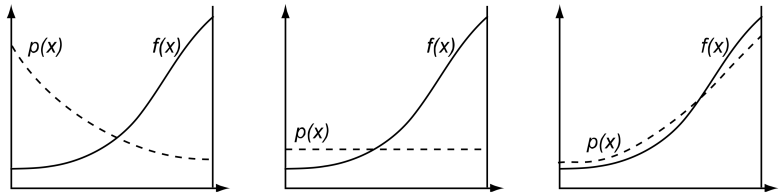

Figure 3.8: Three examples of importance functions [8].

The difficulty in this method lies in the choice of the PDF for which the variance is minimised and thus almost creates a perfect estimator.
The best choice of PDF is where;

$$p(x) = cf(x) \tag{3.46}$$

With c being the constant of proportionality, creating a perfect estimator.

$$E(x) = \int \left(\frac{f(x)}{p(x)}\right)^2 p(x)\,dx + \lambda \int p(x)\,dx \tag{3.47}$$

Due to the Euler-Lagrange differential, Eq. 3.47 can be written as

$$L(p) = \int \left(\frac{f(x)^2}{p(x)} + \lambda p(x)\right) dx \tag{3.48}$$

To minimise this formula it needs to be differentiated with respect to $p(x)$ and solved for $p(x) = 0$.

$$0 = \frac{\partial}{\partial p}\left(\frac{f(x)^2}{p(x)} + \lambda p(x)\right) \tag{3.49}$$

$$0 = -\frac{f^2(x)}{p^2(x)} + \lambda \tag{3.50}$$

$$p(x) = \frac{1}{\sqrt{\lambda}}|f(x)| \tag{3.51}$$

Then the optimal PDF can be written as [8].

$$p(x) = \frac{|f(x)|}{\int f(x)\,dx} \tag{3.52}$$

Depending on how the PDF is chosen, the variance can be reduced considerably, making importance sampling one of the most useful and powerful techniques of the Monte Carlo integration.

4 Case Study

The goal of this research is to simulate global illumination in a produced image. Therefore there is a need for a rendering algorithm. After this there s a study of a few variance reduction techniques.

4.1 The rendering algorithm

There are two common methods to solve the rendering equation (Eq. 3.8 at p. 16). Finite elements and Monte Carlo methods with corresponding techniques; the radiosity algorithm and the ray tracing algorithm with the path tracer as extension.
In our case study, the focus will be on the second method as this handles light transport efficiently [2]. The general idea of ray tracing with respect to lighting the scene is that light travels along a ray (section 3.1.2 Optics) from the light source into the scene, then it bounces around until finally it reaches the sensor that captures the image. This kind of ray tracing is called forward tracing [3]. When forward tracing would be implemented, there will be a lot of rays traced from the light that doesn't affect the result of the image because they will never arrive at the sensor. This is why backwards tracing is a more suitable solution to the problem. With backwards tracing, we follow the light from the sensor to the scene to the light source. The rays are cast from the camera (sensor) tracing the pixel grid into the scene[2]. This direction can be switched around due to the property of energy conservation expressed in Eq. 3.7 in section 3.1.3 Radiometry.
Such approaches are also called the light gathering approaches. They estimate (section 3.2 Randomised algorithms) the contribution of flux every time the ray bounces through the scene [15]. As these rays bounce across the scene they form a path. Thus, another name for this algorithm is path tracing [2]. We can split up this algorithm in some simple steps [23].

1. **Ray generation.** Together with the camera and the pixel offset, a ray is generated towards the scene. It computes the origin and direction of this ray (see Eq. 4.1) based on the pixel's viewing direction and the camera geometry. This is more discussed in section 4.2 Camera and section 4.3 Ray intersection

2. **Ray intersection.** This step finds the closest intersection point with the scene (further explained in section 4.3 Ray intersection).

3. **Shading.** To know the radiance value of the corresponding pixel, shading models are used to represent the micro-facet theory of materials. A more detailed explanation follows later in section 4.4 Shading models.

These simple steps can be written in pseudo code as follows:

Algorithm 1 Render algorithm

```
procedure RENDER const
    for (pixel in image) do
        for (n in nrOfSamples) do
            Ray r = camera.getRay(pixel + offset sample);
            pixel.colour += TracePath(r, 0);
            pixel.colour /= nrOfSamples;
        end for
    end for
end procedure
```

This algorithm is the first general-purpose Monte Carlo light transport algorithm [13]. With this algorithm we can create various effects, such as soft shadows (indirect lighting), transparent objects with caustics and many more. Due to the scope of this project and the defined research questions (section 2 Introduction) the focus will be on creating these soft shadows. When looking at the algorithms (algorithm 1 Render algorithm and algorithm 2 Path tracing algorithm) more closely, it is seen that the structure of these algorithms by default provide this indirect lighting (e.g. global illumination) and thus by default also creates the soft shadows [2].

Algorithm 2 Path tracing algorithm

```
procedure TracePath(Ray r, float depth)
    if depth >= maxDepth then
        return background colour;
    end if
    HitRecord record;
    if scene.Hit(r, record) then
        Material material = record.material;
        Colour emitted = material.Emit();
        Colour attenuation = material.BRDF();
        PDF pdf = renderer.usedPDF;
        Ray scattered(record.hitpoint, pdf.Generate());
        float value = pdf.value(scattered);
        float scatter = material.ScatterPDF(r, record, scattered);
        Colour incoming = TracePath(scattered, scene, depth + 1);
        Colour colour = emitted + (attenuation * scatter * incoming) / value;
        colour.MaxToOne();
        return colour;
    else
        return background colour;
    end if
end procedure
```

In the rendering algorithm there are a lot of general variables, worth mentioning are the ones called number of samples and the depth.

- **Number of samples (SPP).** These are the number of samples per pixel (SPP, table 1 Abbreviations. that are used to generate the image. A characteristic of the path tracing algorithm is that the estimate that is calculated with the Monte Carlo integration, has a higher variance with decreasing values of SPP. Thus, the estimate will become more imprecise for lower values of SPP [2].

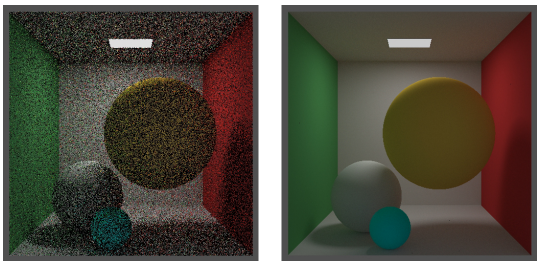

Figure 4.1: Renders of the path tracer with respectively 1SPP, 100SPP and 1000SPP.
(The images with greater detail can be found in A Images Appendix).

- **Depth (D).** This says something about how much you want to sample from the environment.

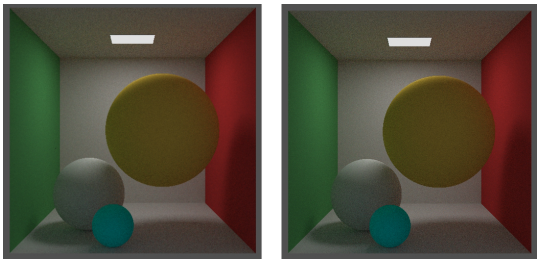

Figure 4.2: Renders of the path tracer with respectively 25D and 100D.
(The images with greater detail can be found in A Images Appendix).

Because we use a probabilistic method for incoming radiance and the Monte Carlo integration for sampling techniques, it is possible to utilize an already existing ray-tracer by modifying it [16].

4.2 Camera

When we want to generate an image, we need to project a 3D scene onto a 2D plane (e.g. the image plane). For this we need the concept of a camera.
The most standard approach to achieve this, is linear perspective[23]. With linear perspective, 3D objects are projected on a 2D plane in such a way that straight lines in the scene become straight lines in the image. There are 2 types of parallel projection; orthographic projection and perspective projection [23].

- **Orthographic projection.** When lines are parallel projected onto a plane. This gives a skewed view of the image compared to how the eye perceives the 3D scene.

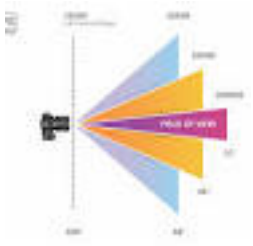

Figure 4.5: Effects of different focal lengths.

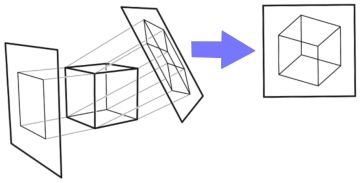

Figure 4.3: Orthographic projection of a 3D scene to a 2D scene.

- **Perspective projection.** Project along lines that gather at a point (e.g. viewpoint, origin of the camera). This method lies closer to how we perceive objects in the world, as the eye or camera (e.g. sensor) does not collect light from a single view direction.

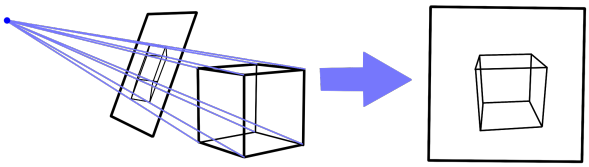

Figure 4.4: Perspective projection of a 3D scene to a 2D scene.

We can represent the camera as follows:

$$Camera = \begin{cases} POINT & origin \\ VECTOR[3] & uvw \\ SCALAR & focallength \end{cases} \quad \text{Ortho-normal basis of the camera.}$$

4.3 Ray intersection

Before we can generate rays, we need a mathematical representation for a ray. The properties of a ray can be represented as follows:

$$Ray = \begin{cases} POINT & origin \\ VECTOR & direction \end{cases}$$

The ray can be written as a line in the parametric form with an origin and direction [1], [12], [23]. Then a point along the ray can be calculated.

$$p(t) = Ray.origin + t * Ray.direction \tag{4.1}$$

The ray generation method starts from an ortho normal base (camera uvw see the representation of Camera). The image plane is some distance away from the camera. If we change this distance, we change the focal length of the camera (Fig. 4.5).

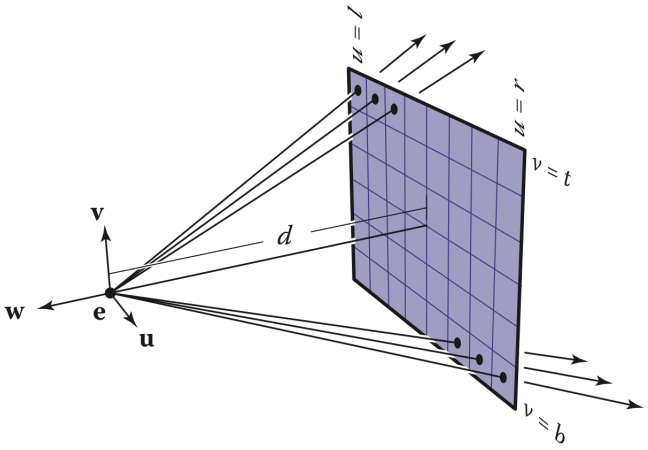

Figure 4.6: Perspective sampling of pixels [23].

For the rendering algorithm we need to be able to find the first intersection point with the objects of the scene. This means that t in Eq. 4.1 will fall in the interval $[0, \infty]$. The ray is already defined, thus only a description of these objects in the scene is still needed. The way of defining these is through implicit equations with a general formulation as was done in Eq. 4.2 [23].

$$f(x, y, z) = 0 \tag{4.2}$$

The calculation of this point is always done in the same order.

1. Substitute the ray equation Eq. 4.1 in the general implicit equation Eq. 4.2, giving

$$f(p(t)) = 0 \tag{4.3}$$

$$f(Ray.origin + t * Ray.direction) = 0 \tag{4.4}$$

2. Solve step 1 for t.

3. Fill in the value of t in the ray equation Eq. 4.1.

To keep the implementation simple, only a few primitives will be implemented and discussed. First the ray-plane intersection (section 4.3.1 Ray-Plane intersection) is discussed, secondly the ray-sphere intersection (section 4.3.2 Ray-Sphere intersection), and as last the ray-rectangle intersection (section 4.3.3 Ray-Rectangle intersection).

4.3.1 Ray-Plane intersection

The representation can be described as follows,

$$Plane = \begin{cases} POINT & origin \\ VECTOR & normal \end{cases}$$

The implicit equation of an infinite plane is:

$$(p - Plane.origin) \cdot Plane.normal = 0 \tag{4.5}$$

According to the first step in section 4.3 Ray intersection, substituting Eq. 4.1 in Eq. 4.5, gives

$$(Ray.origin + t * Ray.direction - Plane.origin) \cdot Plane.normal = 0 \tag{4.6}$$

Solving this for t gives (step 2 discussed in section 4.3 Ray intersection).

$$t = \frac{(Plane.origin - Ray.origin) \cdot Plane.normal}{Ray.direction \cdot Plane.normal} \tag{4.7}$$

As the last step we need to substitute t back into Eq. 4.1 (step 3 in section 4.3 Ray intersection),

$$p(t) = Ray.origin + \left(\frac{(Plane.origin - Ray.origin) \cdot Plane.normal}{Ray.direction \cdot Plane.normal} \right) * Ray.direction \tag{4.8}$$

4.3.2 Ray-Sphere intersection

We will define a sphere as follows:

$$Sphere = \begin{cases} POINT & origin \\ SCALAR & radius \end{cases}$$

with its according implicit function [23],

$$(p - Sphere.origin) \cdot (p - Sphere.origin) = Sphere.radius^2 \tag{4.9}$$

substituting p again with the definition of the ray:

$$\begin{aligned}&(Ray.origin + t * Ray.direction - Sphere.origin)\\&\cdot (Ray.origin + t * Ray.direction - Sphere.origin) - Sphere.radius^2 = 0\end{aligned} \tag{4.10}$$

Solving this for t gives us

$$\begin{aligned}&(Ray.direction \cdot Ray.direction)t^2\\&+ (2Ray.direction \cdot (Ray.origin - Sphere.origin))t\\&+ ((Ray.origin - Sphere.origin) \cdot (Ray.origin - Sphere.origin)) - Sphere.radius^2 = 0\end{aligned} \tag{4.11}$$

In this formula, the form of a quadratic equation $At^2 + Bt + C = 0$ can be identified with

$$\begin{aligned}A &= Ray.direction \cdot Ray.direction\\B &= 2Ray.direction \cdot (Ray.origin - Sphere.origin)\\C &= ((Ray.origin - Sphere.origin) \cdot (Ray.origin - Sphere.origin)) - Sphere.radius^2\end{aligned}$$

Because of this, the discriminant can be recognised to filter out intersections. The properties of the discriminant are:

$D < 0$ The ray does not intersect the sphere

$D = 0$ The ray is tangent on the sphere, there is one intersection point

$D > 0$ The ray intersects the sphere, there are two intersection points

With the only interesting case being $D > 0$, so the solutions for t are:

$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \tag{4.12}$$

The addition will always be tried before the subtraction, and the subtraction will only get checked if the addition returns an invalid value for t.

4.3.3 Ray-Rectangle intersection

The rectangle is represented with

$$Rectangle = \begin{cases} POINT[4] & corners \\ VECTOR & normal \end{cases}$$

How to solve the intersection is an extension of the ray-plane intersection that is explained in detail in section 4.3.1 Ray-Plane intersection. If there is a valid solution for this intersection, it is sure that

the ray intersects the plane on which the rectangle is defined. A way to find out if the point that this intersection found is in the rectangle(or polygon), is called inside-outside testing [12].
For this testing, there are some assumptions that are made; The testing is heavily dependent on how the vertices are defined in the array (see the representation of a rectangle). It is assumed that these are in counter clockwise order (ccw, see table 1 Abbreviations.) with respect to the normal of the rectangle.
Suppose P_0, P_1, P_2, P_3 form a rectangle with normal n, then the edges are E_0, E_1, E_2, E_3 and the corresponding edge are made from the point P that is returned with the intersection.

$$E_0 = P_1 - P_0 \quad \Rightarrow \quad V_0 = P - P_0 \tag{4.13}$$
$$E_0 = P_2 - P_1 \quad \Rightarrow \quad V_0 = P - P_1 \tag{4.14}$$
$$E_0 = P_3 - P_2 \quad \Rightarrow \quad V_0 = P - P_2 \tag{4.15}$$
$$E_0 = P_0 - P_3 \quad \Rightarrow \quad V_0 = P - P_3 \tag{4.16}$$
$$\tag{4.17}$$

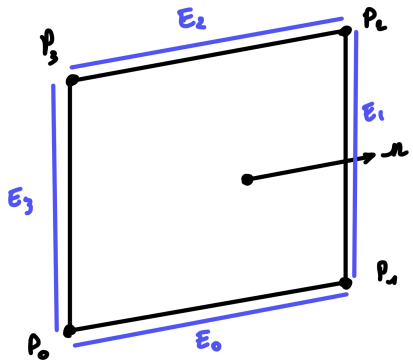

Figure 4.7: Drawing of a rectangle.

If we take the cross product of E and V, we will create a vector perpendicular to the plane formed by the two sides. When the point is left of the edge, the vector created by the cross product will somewhat align to the normal (Fig. 4.8), and thus creating a positive dot product. If the point passes all the edges, we can be certain that the point is lying in the rectangle and on the same plane as the rectangle.

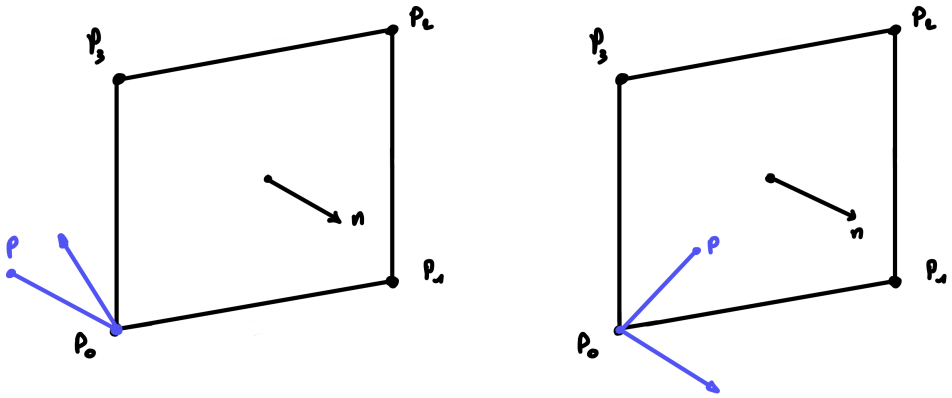

Figure 4.8: Creating vector with point P and the result cross product.

4.4 Shading models

To be able to tell if global illumination is achieved, it suffices to implement a simple diffuse material. The BRDF (section 3.1.5 Interaction of light with materials) has certain assumptions [19]. The incident light direction ω_i and the outgoing viewing direction ω_0 are normalized and the BRDF should thus not concern itself if these two are lying in the same hemisphere. The diffuse material that is implemented follows the Lambertian model. It is called Lambertian because it obeys Lambert's Law [24], which states that the BRDF is constant.

$$p(k_i) = \frac{cos\theta_i}{\pi} \tag{4.18}$$

4.5 Sampling

Sampling will be based on the local reflectance property of the surface [15], i.e. according to the cosine PDF. This will insure that more rays are sent in the direction that will have larger contributions, compared to the uniform PDF (section 3.2.1 The basics of statistics), according to these local reflective properties [15].

$$p(x) = \frac{1}{2A}\left[1 + cos\left(\frac{x - P}{A}\pi\right)\right] \tag{4.19}$$

With $A = Amplitude$ and $P = frequency$.

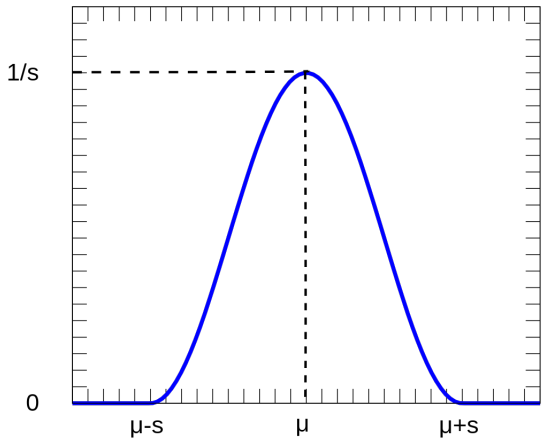

Figure 4.9: Cosine probability density function.

Because our sampling-space is not necessarily a square, we want to find a way to sample from other spaces while still holding true to the chosen distribution. One way to do this is via the rejection method (algorithm 3 Random point in unit disk.).

Algorithm 3 Random point in unit disk.

```
procedure RANDOM POINT IN UNIT DISK
    Vector3 point = max;
    while SqrMagnitude(point) >= 1 do
        float x = RandCanonical();
        float y = RandCanonical();
        point = 2 * (x,y,0) * (1,1,0);
    end while
    return point;
end procedure
```

This algorithm can easily be extended to 3D. As we would then create a random point in a unit sphere. This is done by creating another random variable z for the added dimension. The same algorithm can also be written in polar coordinates.

Algorithm 4 Random point in unit disk with polar coordinates.

```
procedure RANDOM POINT IN UNIT DISK
    float r = RandCanonical();
    float theta = RandCanonical() * 2 * π;
    Vector3 point(r * cos(theta), r * sin(theta));
    return point;
end procedure
```

This algorithm can also easily be extended to 3D. When describing a point in a sphere spherical

coordinates are used. This algorithm changes slightly more.

Algorithm 5 Random point in unit sphere with spherical coordinates.

```
procedure RANDOM POINT IN UNIT SPHERE
   float r = RandCanonical();
   float theta = RandCanonical() * 2 * π;
   float phi = RandCanonical() * 2 * π;
   Vector3 point(r * cos(phi) * sin(theta), r * sin(phi) * cos(theta),
r * cos(theta) );
   return point;
end procedure
```

Some other methods to achieve this, are Metropolis and the inverse function [23].
Rendering times are only loosely coupled to the scene representation, Monte Carlo algorithms can sample the scene to determine the information they actually need. They place very weak restrictions on the "randomness" of the numbers they use. [25]
To utilise the stratified sampling, we will divide a pixel in as many sub-pixels that there are samples. This is done as the most common example, as it helps jittering [4], [6].

4.5.1 Random variables

When a random number needs to be generated in the field of computer science, almost always a function is used in the order of $rand(min, max)$. Awareness needs be raised that these random variables are pseudo random variables [14], [19]. This is very valuable as debugging is an absolute requirement, which means that the code written needs to have the ability to repeat a particular run.

4.6 From theory to implementation

For the implementation, the ray tracing books from Shirley ([20]–[22]) were integrated in the raytracer implementation that was already available. These books provide a simple explanation and implementation of a Monte Carlo path tracer. The design of this implementation was then converted to fit in the Elite render engine, that had a few already existing base classes, such as the renderer, and math representations of a vector, point and matrices. Later on, other implementations were integrated to become a foremost outcome.
To describe the effectiveness of the variance reduction techniques, 2 abstract values are needed to be able to compare and verify the statements made in section 3 Background - Literature Study and section 4 Case Study. The methodology needs to be able to say something about the noise that is in an image. Therefore, image quality metrics are used.
There are two big categories in image quality metrics; full-reference quality metrics and no-reference quality metrics. For our research question, the full-reference quality metrics will be used.

- **Mean Squared Error(MSE, table 1 Abbreviations.).** The MSE measures the average squared difference between an actual and ideal pixel value.
- **Peak Signal-to-Noise Ratio(PSNR, table 1 Abbreviations.).** The PSNR is derived from the MSE, it indicates the ratio of maximum pixel intensity to the power of the distortion. One disadvantage of this method is that it might not align with the perceived quality of an image.
- **Structural Similarity Index (SSIM, table 1 Abbreviations.).** This metric combines both local image structure, luminance and contrast into a single local quality score.
- **Multi-Scale Structural Similarity Index (MS-SSIM, table 1 Abbreviations.).** This metric expands on the SSIM by combining the luminance information at the highest resolution level with structure and contrast information at several down-sampled resolutions.

The first two metrics are used to compare the image compression quality [5]. With other words, they look at how much noise there is in an image. In general, the higher the PSNR value, the better the quality of the compressed or reconstructed image. In this case, the images with less samples or a different variance reduction technique. For MSE, the lower the value, the better. Zero means that the image is identical to the reference image.
A big drawback of these metrics is that a reference image is needed. Herefore we will use an image with 1000 SPP to have the least amount of noise possible with a render time that is adequate.
To tell something about how equal the images are across sampling methods, MSE and SSIM will be used. The SSIM returns a value between zero and one. One indicates that the images are identical sets.

4.6.1 Matlab

To generate the image quality metrics, the trial version of Matlab is used. In mathlab itself, there are functions for the image quality metrics and the rest is self explanatory.

4.6.2 Benchmarking

To be able to tell if the variance reduction techniques are fully effective, the render function needs to be benchmarked. Therefore an own implementation will be used that renders the images an x amount of time, saves all the durations and also returns an average duration of all the iterations. To have a successful benchmark, the iteration with the lowest and highest value needs to be discarded.

4.7 Cornell Box

To render an image that can be compared with others, the same objects need to be at the same place with the same values. The cornell-box is a test that is suited to determine the precision of rendering software by comparing rendered scenes to a picture of the same scene (in this case a high quality render will be used). The experiments and results will be discussed in the next section.

5 Experiments and results

5.1 The setup

All the simulations are run on an Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21GHz with a RAM of 16GB. Generating images with respectively 1 SPP, 100 SPP and 1000 SPP; for 1D, 50D and 100D; for all combinations of sampling.

5.2 Generating the images

To be able to compare and contrast the different variance reduction techniques, images will be generated for all possible combinations that can be made. All images can be found in A Images Appendix.

5.3 Generating redering times

We will be benchmarking the time to render an image for 100SPP and 50D. All benchmark timings can be found in B Runtimes Appendix.

5.4 Comparing images

Comparing images will be done by a small script in Matlab, the results can be found in C Image quality metrics Appendix.

6 Discussion

Bigger pictures with the same properties can be found in A Images Appendix (to look at them in more detail).

6.1 Rendered images

The best image of all the results is the image Fig. A.11. This image is rendered with 1000 SPP and 100D, and it combines the different variance reduction techniques as well.

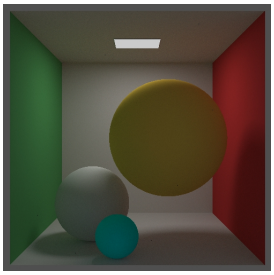

Figure 6.1: Best result (1000SPP and 100D).

Global illumination is simulated because of the light reflected on the walls (left render in Fig. 6.2), but also on the objects in the scene (middle render) and soft shadows are visible (right render). It can be concluded that the render algorithm works for simulating global illumination.

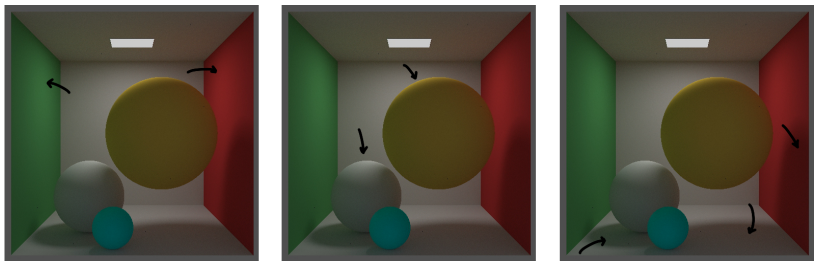

Figure 6.2: Global illumination properties in the image.

6.2 High SPP versus low SPP

As stated in section 4.1 The rendering algorithm the variance, also called the error of the estimator, which results in a noisy image. It is also mentioned that the variance increases when the SPP are lower. This is visible in the row of renders in Fig. 6.3. It is logical that the left render took the least amount of time to render and the right the most.

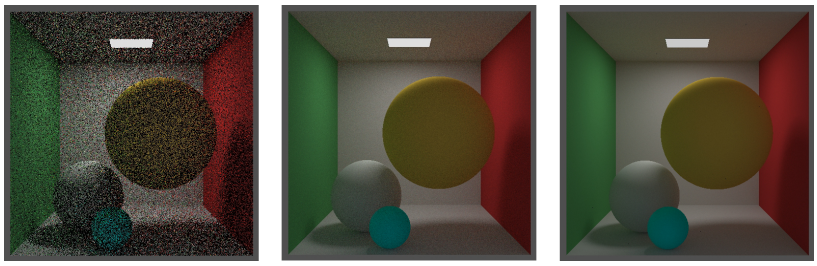

Figure 6.3: Renders with respectively 1SPP, 100SPP and 1000SPP.

6.3 High depth versus low depth

When we change the depth variable (see section 4.1 The rendering algorithm), the longevity of the maximum path that gets traversed by the rendering algorithm is changed. The higher the depth, the clearer our image becomes and the more light is simulated in the image. The image on the right, almost has no depth precision, because there is almost no shading present.

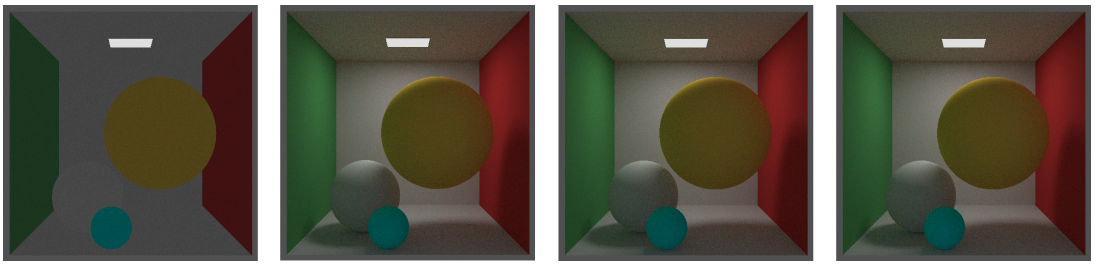

Figure 6.4: Renders with respectively 1D, 25D, 50D and 100D.

6.4 Normal versus importance sampling

The render on the right in Fig. 6.5 displays no global illumination anymore. This is due to the fact that when only integrating importance sampling, the algorithm only samples towards the light, creating the ray tracing algorithm for hard shadows. The most noticeable difference is the fact that the ceiling is almost completely black, which is because the light only emits light from one side.

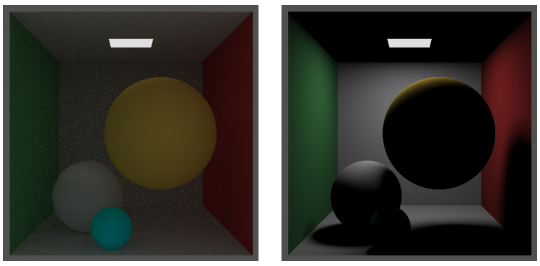

Figure 6.5: Normal versus stratified sampling.

6.5 Normal versus combination sampling

There is is overall more light in the image. This is due to the fact that we sample more towards the light, thus the camera is more exposed to the light in the scene (Fig. 6.6).

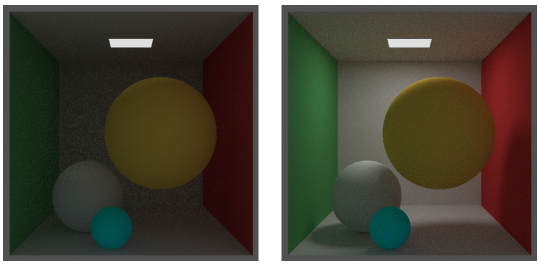

Figure 6.6: Normal versus stratified sampling.

6.6 Normal versus stratified sampling

The images discussed are based on the combination of normal and importance sampling, so that the full global illumination effect is visible. With the black arrows on the right render of Fig. 6.7, we can see that edges are sharper and the transition between a highlight and the shade are a little bit smoother.

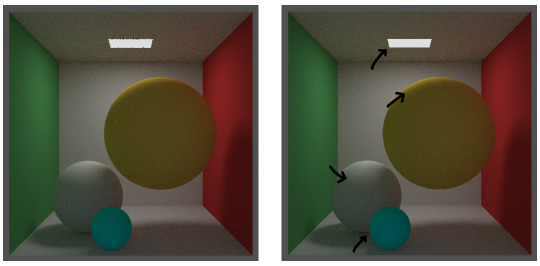

Figure 6.7: Normal versus stratified sampling.

6.7 Rendering times

When looking at the average timing results from the benchmark it can be concluded that these variance reduction techniques are even better in time (Fig. 6.8). It takes on average less time than the normal sampling. It can also be seen that stratified sampling on average takes a little bit less time than its counterpart.

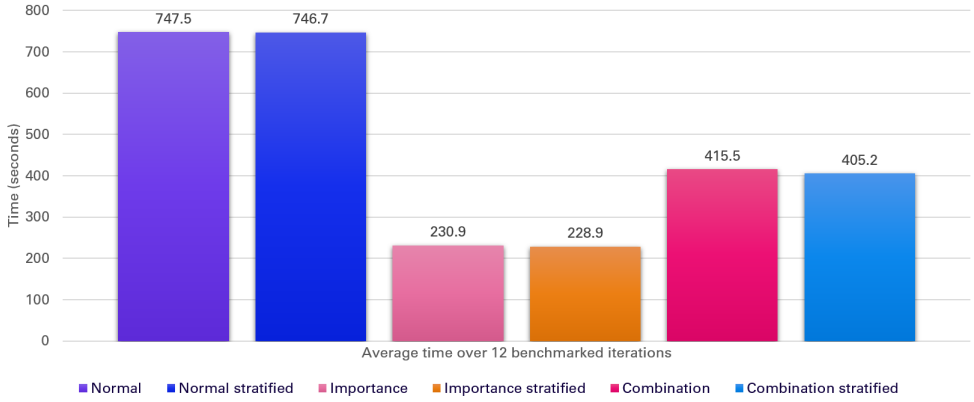

Figure 6.8: Average timings for the renders with 50D 100SPP.

6.8 Image quality results

Fig. 6.9 gives an overview of all quality metric results.

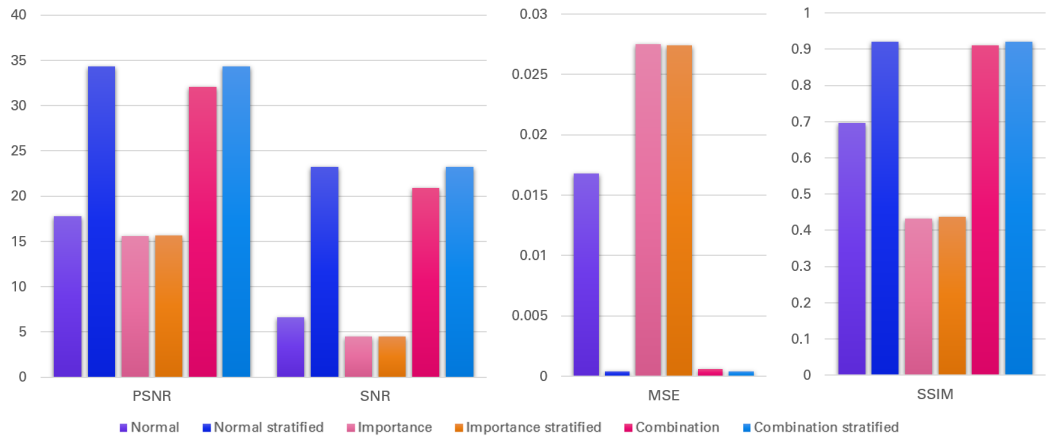

Figure 6.9: Image quality metrics results.

6.8.1 Mean Squared Error

The MSE values for importance sampling and the importance with stratified sampling are the biggest, which are due to the hard shadows we can see in the image Fig. A.2 and Fig. A.3. For the Combination of both importance and normal sampling, the values for the MSE are significantly smaller than the values of the MSE for the normal sampling. Thus confirming that these variance reduction techniques give a closer simulation to the real world than the pure Monte Carlo integration Fig. 6.9.

6.8.2 Peak Signal-to-Noise Ratio and Signal-to-Noise Ratio

The PSNR and SNR indicate how good the quality of an image is compared to the reference image. In Fig. 6.9 for both the importance sampling and the combination of importance and stratified sampling there are smaller values, meaning that the image has a lower quality compared to other sampling methods. It is a good sign that for both the combination of normal sampling and importance sampling gives us the highest values and thus the best results.

6.8.3 Structural Similarity Index

The SSIM value tells something about the structure of the image compared to the reference image. Like in the discussion of MSE and PSNR, it can be concluded that importance and importance-stratified sampling gives us the worst result, as they are closer to zero than the rest. With the combination of normal and importance sampling and with stratified sampling giving again the best results as they are close to 1, which means that there is a high structural similarity in the images compared to the reference picture.
To conclude, all these image quality metrics, give us the same answer. The stratified methods are better compared to their non stratified counterparts. The importance sampling is the least effective

in simulating global illumination as it does not take it into account and last but not least, the combination of all sampling methods gives us the best overall result.

7 Conclusion

The experiments together with the results that were reproduced shows that the variance reduction techniques are indeed efficient ways to reduce noise in a rendered image. It can even be said that these techniques are not only more efficient in reducing the noise, but also more efficient in time.

8 Future Work

There are a lot of other variance reduction techniques that are out there (section 3.3.1 The basics of sampling theory), that could be integrated into the project. Other materials can be made and included, such as the specular and glossy material (section 3.1.5 Interaction of light with materials). The path tracer could even be made to support subsurface scattering that happens when light gets reflected onto translucent objects. To reduce the rendering time even more, we can make the path tracer multi threaded.

References

[1] "A minimal ray-tracer: Rendering simple shapes (sphere, cube, disk, plane, etc.)" (), [Online]. Available: https://www.scratchapixel.com/lessons/3d-basic-rendering/minimal-ray-tracer-rendering-simple-shapes/ray-sphere-intersection.

[2] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, and S. Hillaire, *Real-Time Rendering 4th edition*, 4th ed. 2018, pp. 437–512, ISBN: 9781138627000.

[3] J. Arvo, R. L. Cook, A. S. Glassner, *et al.*, *An Introduction to Ray Tracing*, 1st, A. S. Glassner, Ed. Academic Press Limited, 1989.

[4] J. Arvo, M. Fajardo, P. Hanrahan, *et al.*, "State of the art in monte carlo ray tracing for realistic image synthesis," 2001.

[5] "Compute peak signal-to-noise ratio (psnr) between images - simulink - mathworks benelux." (), [Online]. Available: https://nl.mathworks.com/help/vision/ref/psnr.html.

[6] R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," *ACM SIGGRAPH Computer Graphics*, vol. 18, 3 Jul. 1984, ISSN: 0097-8930. DOI: 10.1145/964965.808590.

[7] "Disney's practical guide to path tracing - youtube." (), [Online]. Available: https://www.youtube.com/watch?v=frLwRLS_ZR0&ab_channel=WaltDisneyAnimationStudios.

[8] P. Dutre, P. Bekaert, and K. Bala, *Advanced Global Illumination*, 2nd ed. 2020.

[9] B. Duvenhage, K. Bouatouch, and D. G. Kourie, "Numerical verification of bidirectional reflectance distribution functions for physical plausibility," *ACM International Conference Proceeding Series*, pp. 200–208, 2013. DOI: 10.1145/2513456.2513499.

[10] "Global illumination and path tracing (an intuitive introduction to global illumination and path tracing)." (), [Online]. Available: https://www.scratchapixel.com/lessons/3d-basic-rendering/global-illumination-path-tracing.

[11] J. H. Halton, "A retrospective and prospective survey of the monte carlo method," *SIAM Review*, vol. 12, no. 1, pp. 1–63, 1970. DOI: 10.1137/1012001. eprint: https://doi.org/10.1137/1012001. [Online]. Available: https://doi.org/10.1137/1012001.

[12] J. F. Hughes, A. V. Dam, M. McGuire, *et al.*, *Computer Graphics: Principles and Practice*, 3rd. Addison-Wesley, Aug. 2013. [Online]. Available: https://cgpp.net/about.xml.

[13] J. T. Kajiya, "The rendering equation," ACM Press, 1986, ISBN: 0897911962. DOI: 10.1145/15922.15902.

[14] M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods*, 2nd ed. Wiley-Blackwell, 2008, ISBN: 978-3-527-40760-6.

[15] E. P. Lafortune, "Mathematical models and monte carlo algorithms for physically based rendering," Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium, Feb. 1995, p. 167. [Online]. Available: https://lirias.kuleuven.be/handle/123456789/134595.

[16] B. Lange, "The simulation of radiant light transfer with stochastic ray-tracing," in *Photorealistic Rendering in Computer Graphics*, P. Brunet and F. W. Jansen, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 30–44, ISBN: 978-3-642-57963-9.

[17] F. Nicodemus, J. Richmond, J. Hsia, I. Ginsberg, and T. Limperis, "Geometrical considerations and nomenclature for reflectance," *NBS Monograph 160*, 1977.

[18] F. E. Nicodemus, "Directional reflectance and emissivity of an opaque surface," *Applied Optics*, vol. 4, pp. 767–773, 7 Jul. 1965, ISSN: 0003-6935. DOI: 10.1364/AO.4.000767.

[19] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd edition, M. Phar, W. Jakob, and G. Humphreys, Eds. 2017, ISBN: 978-0-12-800645-0.

[20] P. Shirley, "Ray tracing in one weekend," 2018. [Online]. Available: https://github.com/petershirley/raytracinginoneweekend.

[21] ——, "Ray tracing: The next week," 2018. [Online]. Available: www.in1weekend.com.

[22] ——, "Ray tracing: The rest of your life," 2018. [Online]. Available: www.in1weekend.com.

[23] P. Shirley and S. Marschner, *Fundamentals of Computer Graphics*, 4th. CRC Press, 2016.

[24] P. S. Shirley, "Physically based lighting calculations for computer graphics," University of Illinois at Urbana-Champaign, 1991.

[25] E. Veach, "Robust monte carlo methods for light transport simulation," 1997, ISBN: 978-0-591-90780-3.

A Images Appendix

A.1 Normal sampling

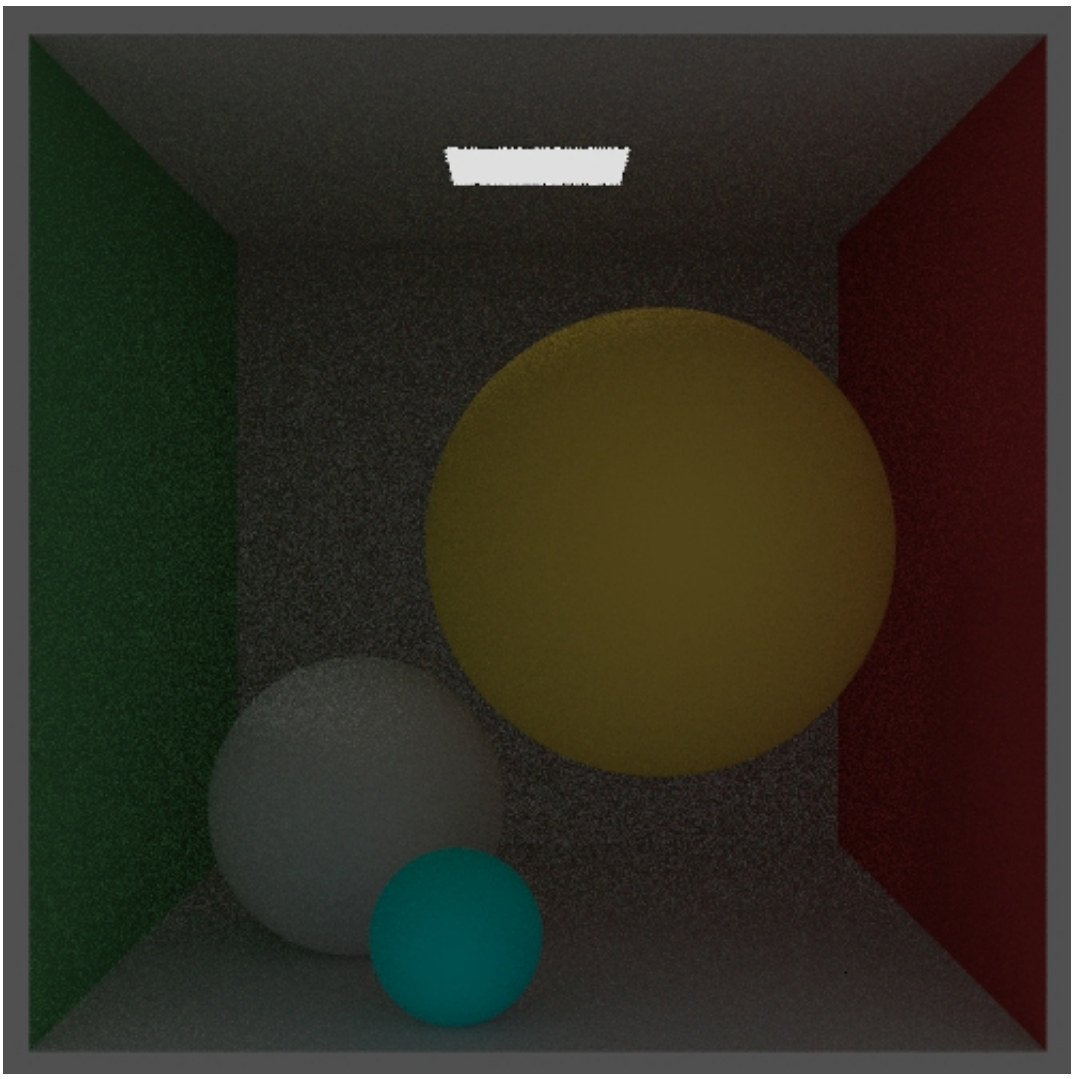

Figure A.1: Render with 100SPP, 50D.

A.2 Normal stratified sampling

A.3 Importance sampling

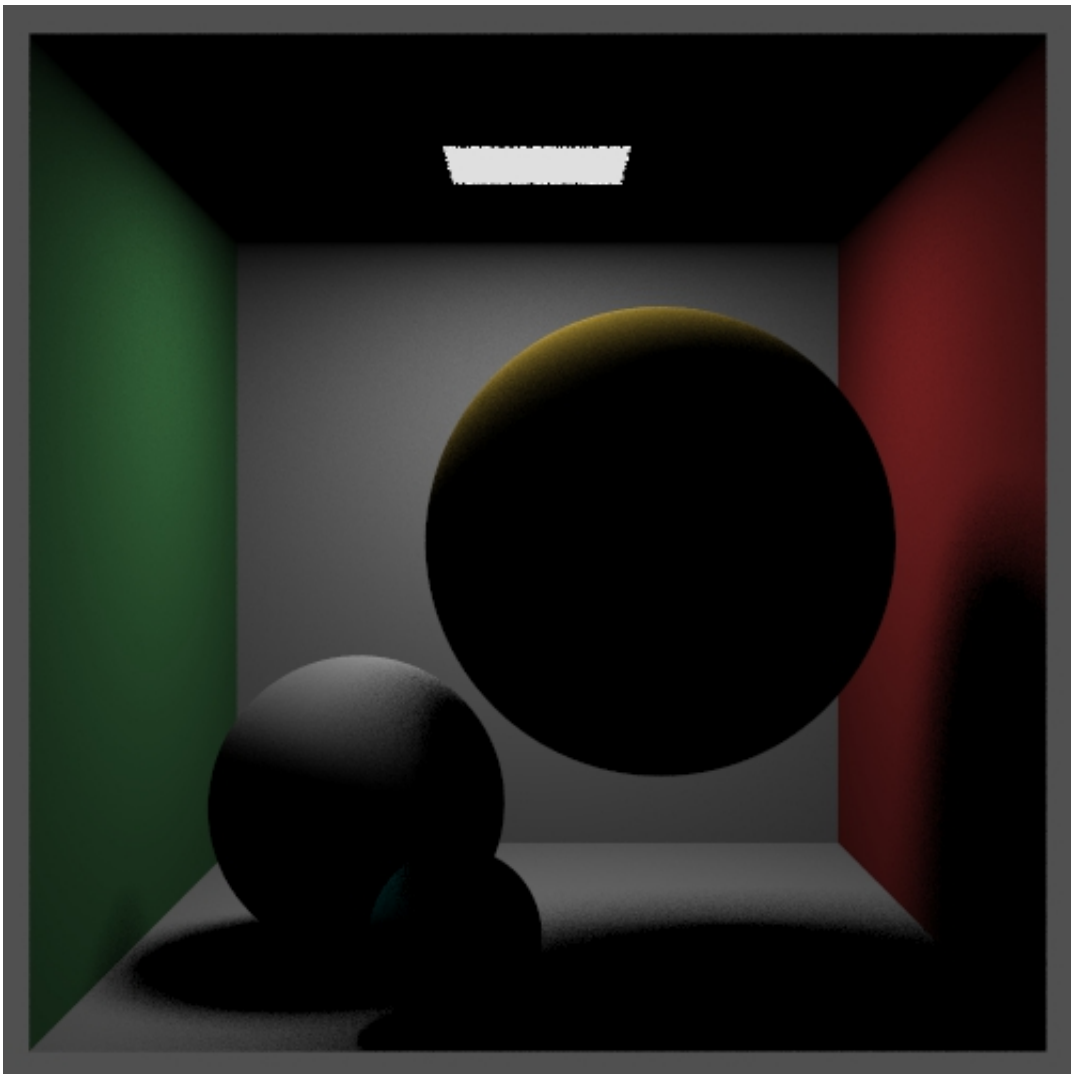

Figure A.2: Render with 100SPP, 50D.

A.4 Importance stratified sampling

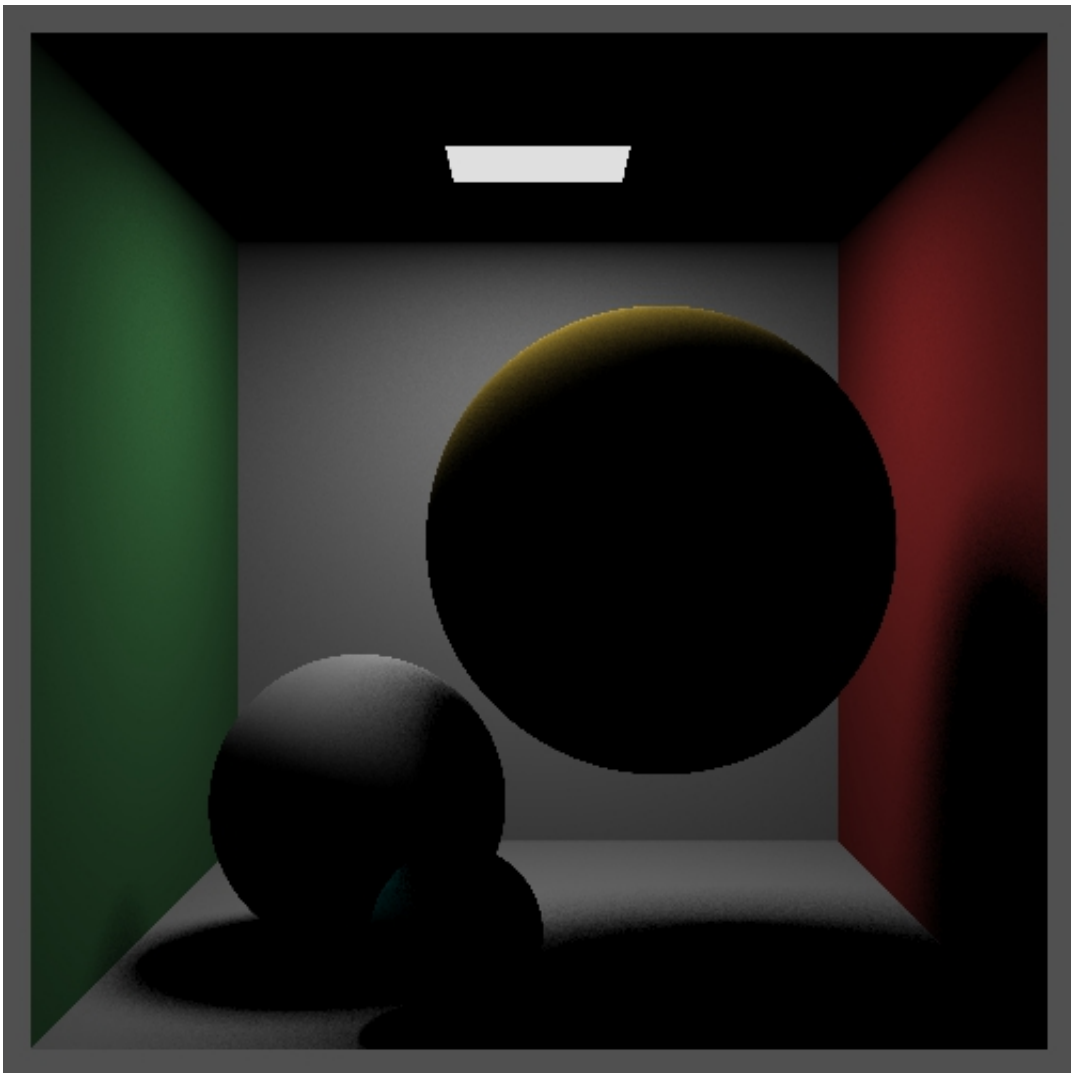

Figure A.3: Render with 100SPP, 50D.

A.5 Combined sampling

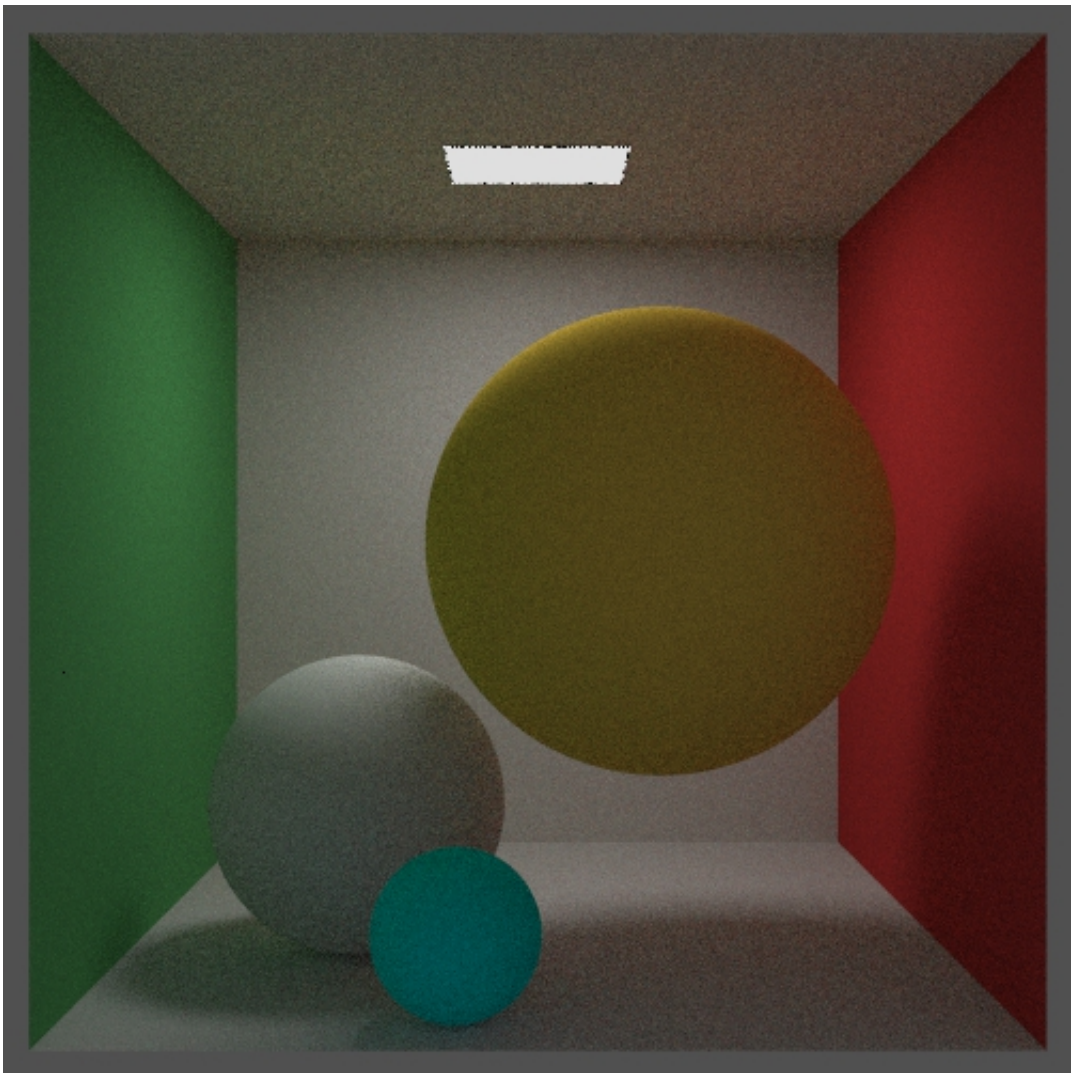

Figure A.4: Render with 100SPP, 50D.

A.6 Combined stratified sampling

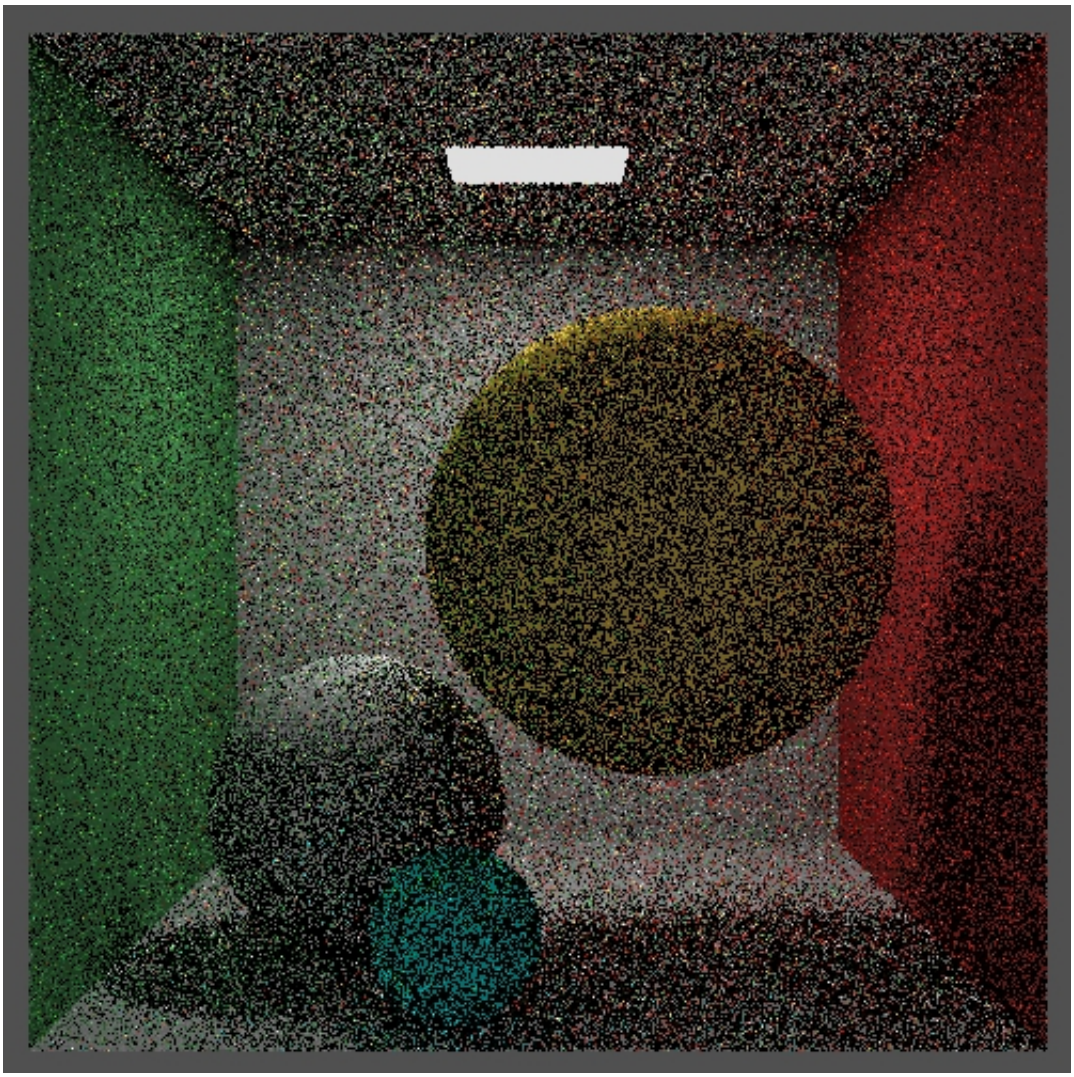

Figure A.5: Render with 1SPP, 50D.

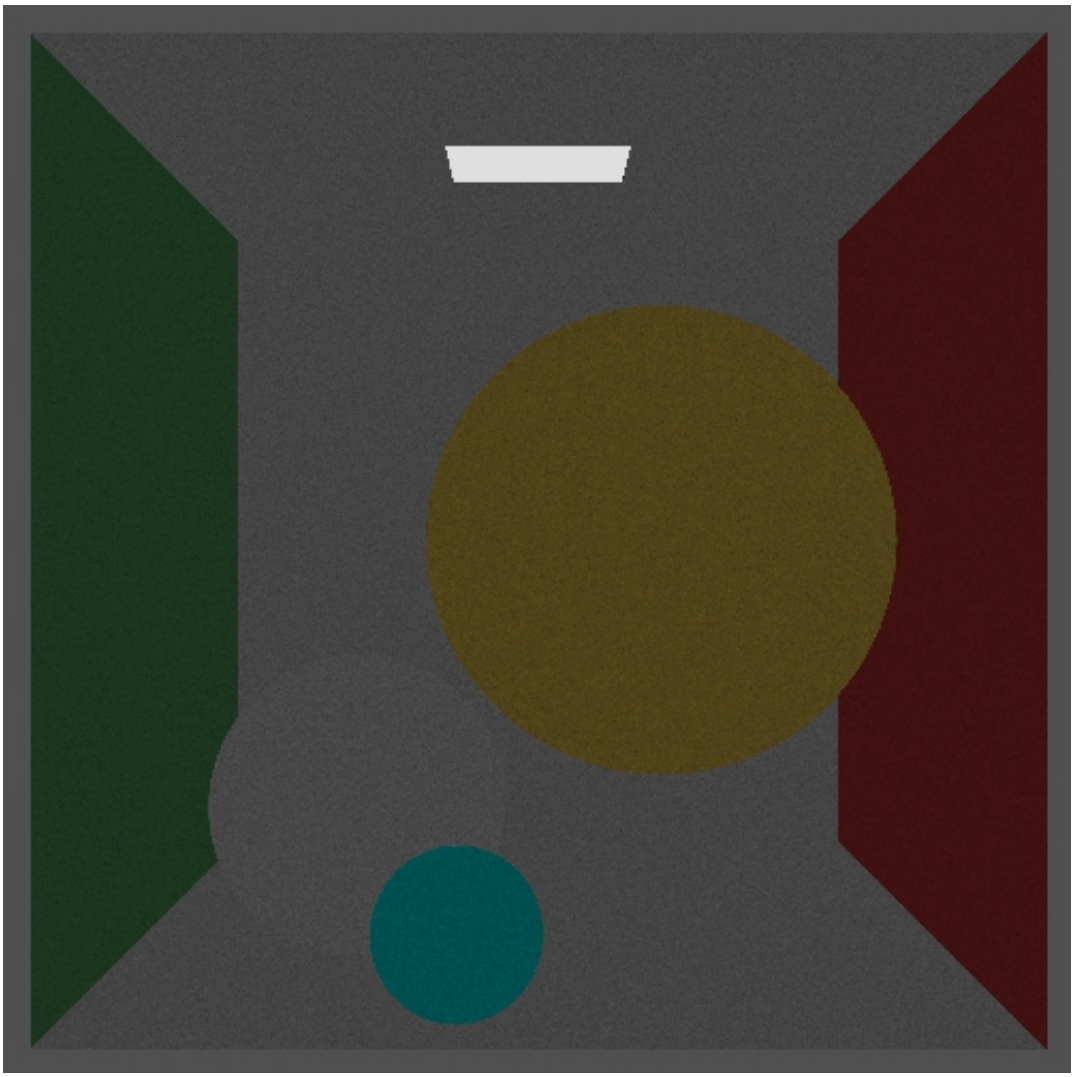

Figure A.6: Render with 100SPP, 1D.

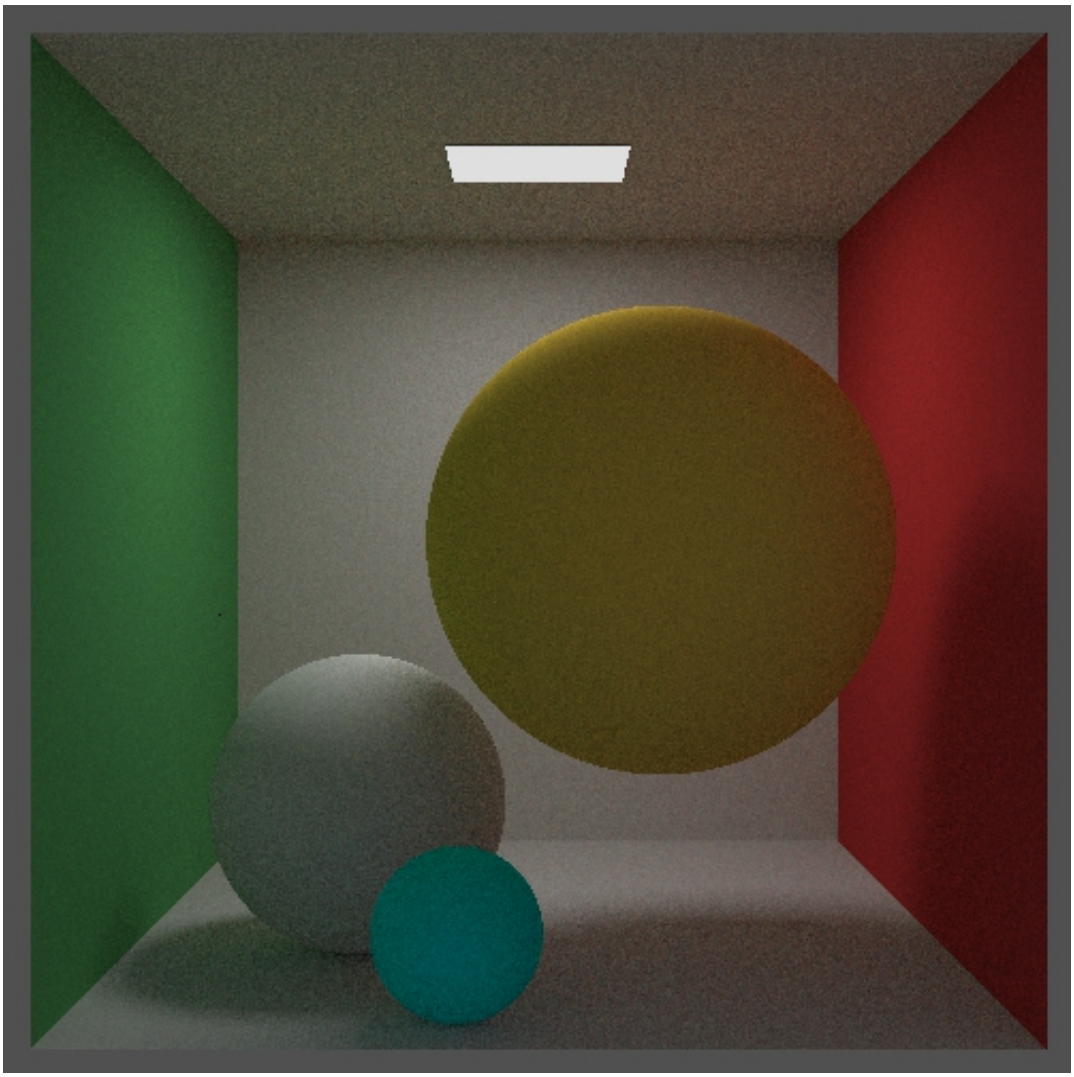

Figure A.7: Render with 100SPP, 25D.

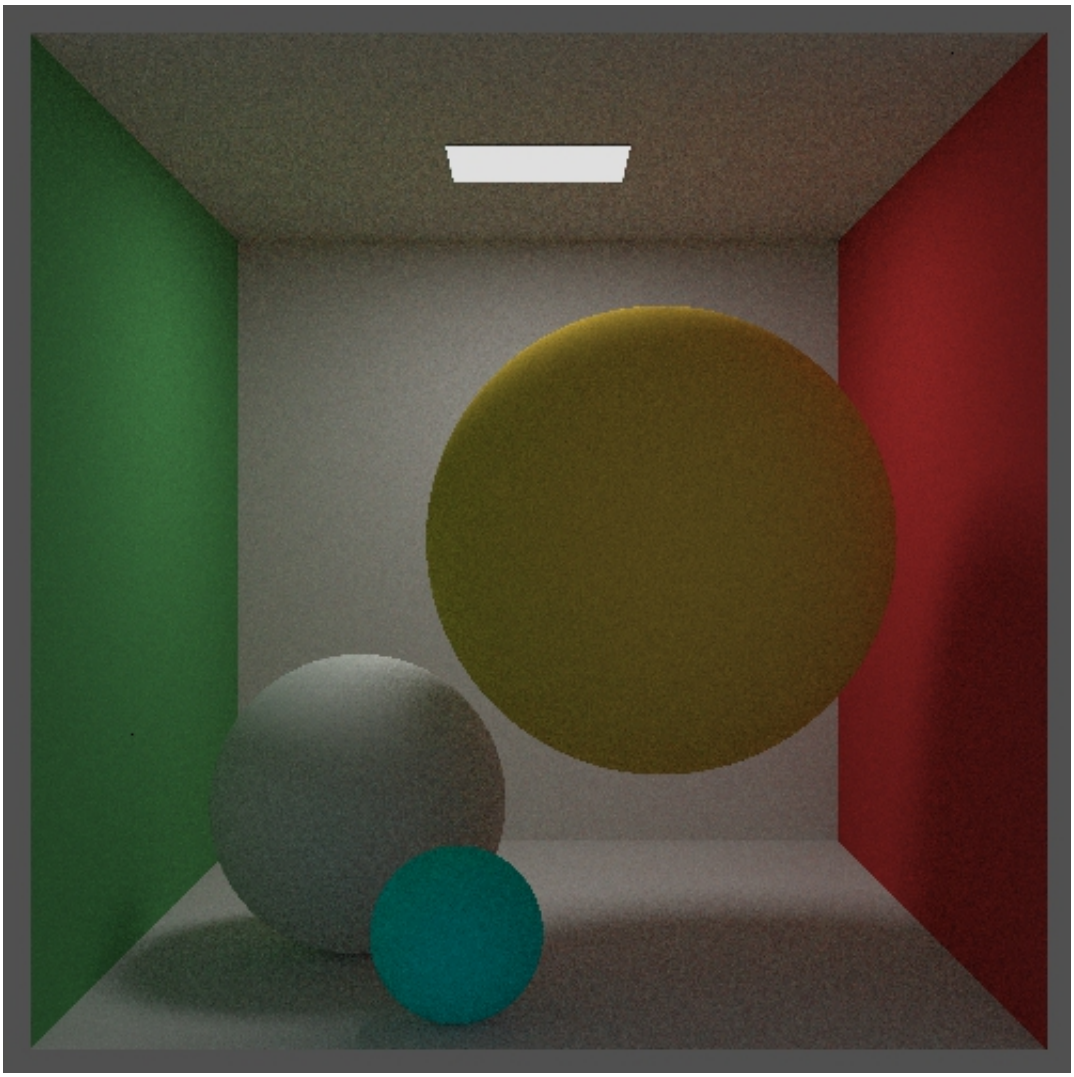

Figure A.8: Render with 100SPP, 50D.

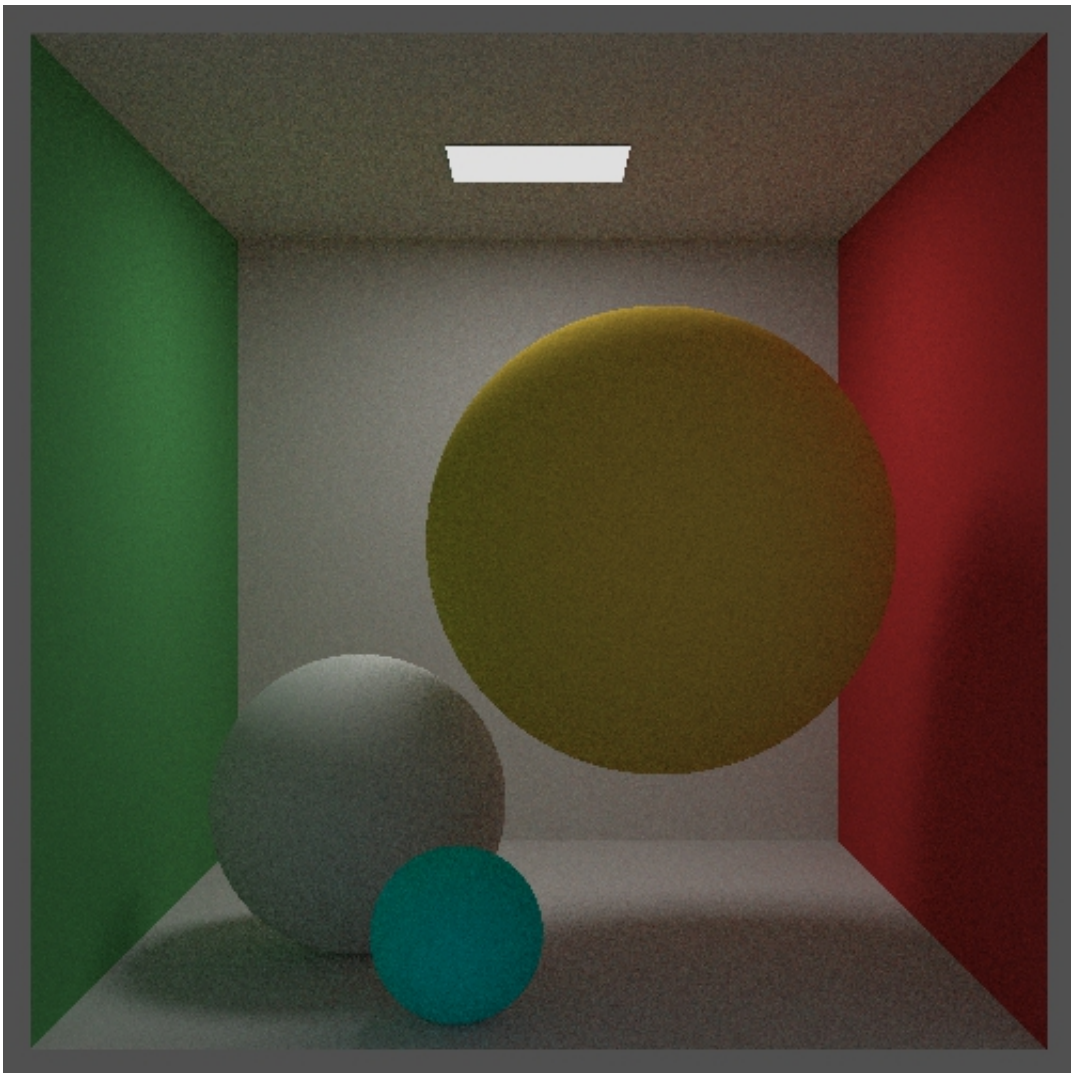

Figure A.9: Render with 100SPP, 100D.

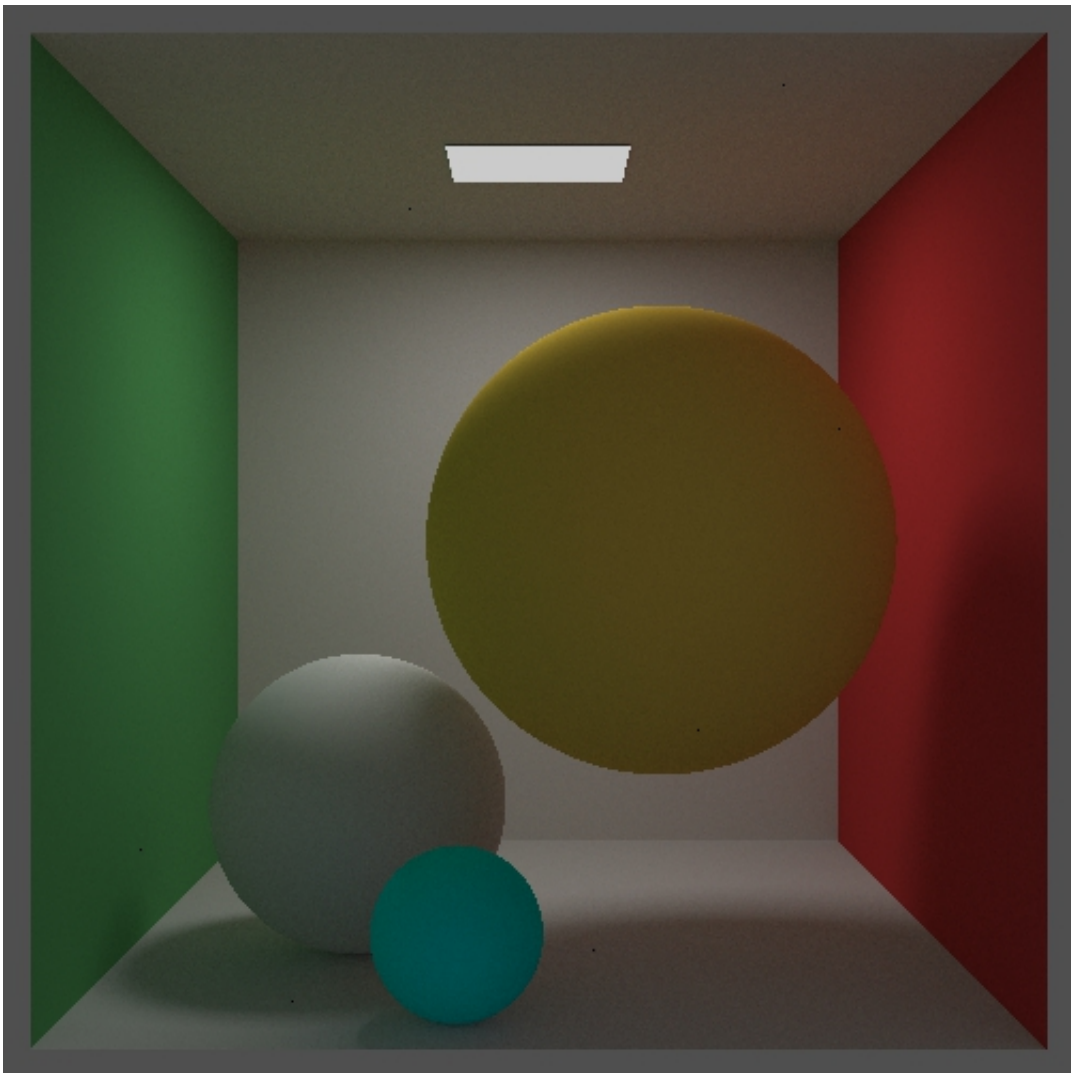

Figure A.10: Render with 1000SPP, 50D.

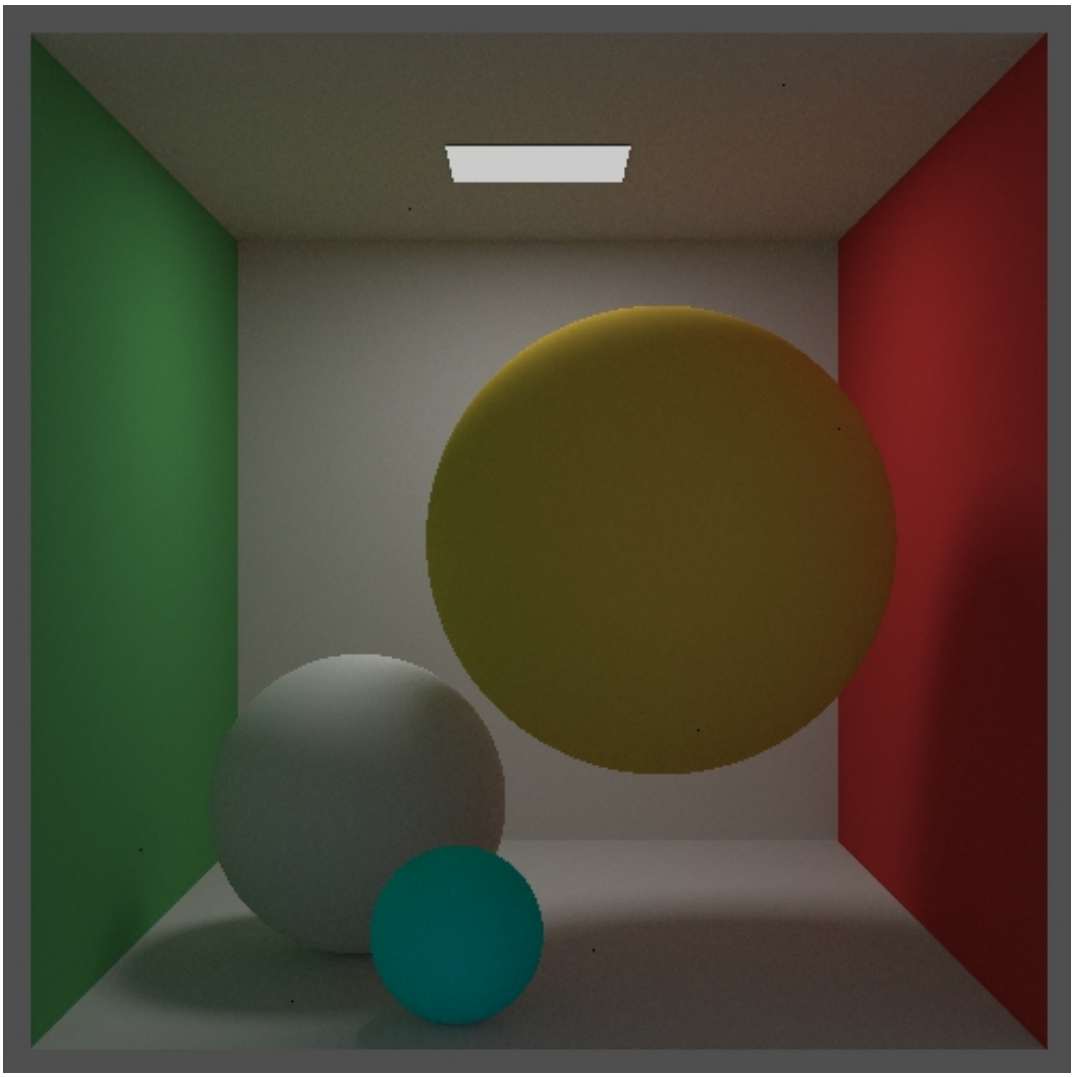

Figure A.11: Render with 1000SPP, 100D.

B Runtimes Appendix

Table 3: Benchmarking results with time expressed in seconds.

Technique Name	Time 0	Time 1	Time 2	Time 3	Time 4	Time 5	Time 6	Time 7	Time 8	Time 9	Average Time
Normal	736	720	746	734	738	757	743	734	799	768	747.5
Normal Stratified	752	736	748	713	709	757	758	736	775	783	746.7
Importance sampling	235	238	237	230	237	244	241	221	213	213	230.9
Importance sampling stratified	216	265	231	227	277	222	224	227	221	229	228.9
combined sampling	416	416	387	396	387	413	402	434	488	416	415.5
combined sampling stratified	396	383	398	412	401	405	416	423	410	408	405.2

Note: Images rendered with 100 SPP and a depth of 50.

C Image quality metrics Appendix

Table 4: Benchmarking results with time expressed in seconds.

Technique Name	MSE	PSNR	SNR	SSIM
Normal	17.7605	6.6247	0.6955	0.0168
Normal Stratified	34.3548	23.2210	0.9199	0.0004
Importance sampling	15.6098	4.4760	0.4315	0.0275
Importance sampling stratified	15.6266	4.4928	0.4370	0.40274
combined sampling	32.0932	20.9048	0.9099	.0006
combined sampling stratified	34.3313	23.1968	0.9193	0.0004

Note: Images quality metrics of renders with 100 SPP and a depth of 50 compared to Fig. A.11.