FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Quantifying the Convergence of Light-Transport Algorithms

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Adam Celarek, BSc.
Matrikelnummer 0926881

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Associate Prof. Jaakko Lehtinen, Aalto University
Assistant Prof. Wenzel Jakob, École Polytechnique Fédérale de Lausanne (EPFL)

Wien, 12. Oktober 2017

Adam Celarek

Michael Wimmer

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Quantifying the Convergence of Light-Transport Algorithms

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media Informatics and Visual Computing

by

Adam Celarek, BSc.
Registration Number 0926881

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Associate Prof. Jaakko Lehtinen, Aalto University
Assistant Prof. Wenzel Jakob, École Polytechnique Fédérale de Lausanne (EPFL)

Vienna, 12th October, 2017

Adam Celarek | Michael Wimmer

Erklärung zur Verfassung der Arbeit

Adam Celarek, BSc.
Talererstraße 4, 6322-Kirchbichl, Österreich

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 12. Oktober 2017

Adam Celarek

Acknowledgements

First of all, I thank Jaakko Lehtinen for giving me the opportunity to work in his group at Aalto and for introducing me to Wenzel Jakob. Thanks to both of them for supervising the bulk of the project, for the patience in countless teleconferences and for the valuable advice.

I thank Michael Wimmer for his support at my home university, for his motivation, enthusiasm and honest feedback.

Some of the calculations for this thesis were performed using computer resources within the Aalto University School of Science "Science-IT" project.

My special thanks go to my sister Anna, who proofread the thesis and helped with streamlining the text. Thanks go to Markus Kettunen and the other office mates for discussions, debugging code, eating lunches, drinking coffees and everything else. Thanks also go to my friends in Finland and Austria, I would not have survived the Finnish winter without them.

Last but not least, I want to thank my parents for their constant support throughout my studies, for the free housing and meals during the final phase of the thesis and for the lovely advice (even though they do not really understand what I am doing). I can count on them, whatever happens, and I am very grateful for that.

Kurzfassung

Das Ziel dieser Arbeit ist es, bessere Methoden zur Fehlermessung bei erwartungstreuen und physikalisch basierten Lichttransportalgorithmen einzuführen. Der Stand der Technik ist die Messung via mittlerer quadratischer Abweichung (MSE, Mean Square Error) oder optischem Vergleich von Renderings mit gleich langer Rechenzeit. Diese Methoden sind unzuverlässig, weil MSE anfällig für Ausreißer ist und optische Vergleiche inhärent subjektiv sind.

Wir führen einen simplen *Stellvertreteralgorithmus* ein: *Reine* Algorithmen produzieren ein einziges Bild mithilfe von N Rechenressourcen. Der Stellvertreter hingegen berechnet N unabhängige Bilder mithilfe von jeweils 1 Rechenressource. Das ermöglicht die Anwendung des Zentralen Grenzwertsatzes, dessen Konsequenz eine Konvergenz von $\Theta(1/N)$ ist. Aus dem Stellvertreter ergibt sich außerdem die Möglichkeit, routinemäßig Bilder mit Standardabweichung pro Pixel zu berechnen.

Mit dem Stellvertreteralgorithmus ist es einfach, den Erwartungswert des MSEs zu schätzen. Dieser Schätzer ist zuverlässiger als ein einzelner MSE-Wert und kann für verschiedene Rechenzeiten mittels Division durch N skaliert werden. Ein weiterer Vorteil ist die Möglichkeit, Konfidenzintervalle und Standardabweichung für den MSE-Wert zu berechnen.

Wir schlagen das Fehler Spektrum Ensemble (ESE, Error Spectrum Ensemble) als neues Werkzeug zur Evaluierung von Lichttransportalgorithmen vor. Es visualisiert den zu erwarteten Fehler und Ausreißer in Abhängigkeit von räumlichen Frequenzen. ESE wird mit den Daten aus den Stellvertreteralgorithmen generiert. Mittels einer Referenz werden N Fehlerbilder generiert, daraus werden Fourierquadratspektren berechnet, welche mittels radialem Mittelwert komprimiert werden. Der Deskriptor ist schließlich eine Zusammenfassung dieser Mittelwerte.

In den Ergebnissen zeigen wir, dass Standardabweichungsbilder, Bilder mit kleinem N, ESE und MSE Erwartungswert nützliche Werkzeuge zur Bewertung von Renderingalgorithmen sind.

Abstract

This work aims at improving methods for measuring the error of unbiased, physically based light-transport algorithms. State-of-the-art papers show algorithmic improvements via error measures like Mean Square Error (MSE) or visual comparison of equal-time renderings. These methods are unreliable since outliers can cause MSE variance and visual comparison is inherently subjective.

We introduce a simple *proxy* algorithm: *pure* algorithms produce one image corresponding to the computation budget N. The proxy, on the other hand, averages N independent images with a computation budget of 1. The proxy algorithm fulfils the preconditions for the Central Limit Theorem (CLT), and hence, we know that its convergence rate is $\Theta(1/N)$. Since this same convergence rate applies for all methods executed using the proxy algorithm, comparisons using variance- or standard-deviation-per-pixel images are possible. These per-pixel error images can be routinely computed and allow comparing the render quality of different lighting effects. Additionally, the average of pixel variances is more robust against outliers compared to the traditional MSE or comparable metrics computed for the pure algorithm.

We further propose the Error Spectrum Ensemble (ESE) as a new tool for evaluating light-transport algorithms. It summarizes expected error and outliers over spatial frequencies. ESE is generated using the data from the proxy algorithm: N error images are computed using a reference, transformed into Fourier power spectra and compressed using radial averages. The descriptor is a summary of those radial averages.

In the results, we show that standard-deviation images, short equal-time renderings, ESE and expected MSE are valuable tools for assessing light-transport algorithms.

Contents

CHAPTER 1

Introduction

Light-transport algorithms are used to compute realistic images from scene data, such as geometry, materials, light sources and a sensor. They do this by simulating the physics of light, therefore the process is also called physically based rendering.

Applications are film production, industrial visualisation and art. Since this kind of rendering is computationally expensive, large film productions use render farms with several thousand CPUs. For the same reason, those algorithms are not used extensively for games and other real-time applications. Quicker approximations are used for those, trading efficiency for systematic errors.

Consider how a physical environment is modelled: Each light source emits particles, called photons. They interact with the world taking different paths, are scattered or absorbed by objects, until finally a small portion arrives at the sensor. The interactions depend on the shape of objects and their material properties. Photons collected at the sensor, for instance a camera or an eye, constitute the image. Measured or perceived colour depends on the amount and wavelength of the arriving photons.

In light transport algorithms, the colour of a pixel is computed by measuring the amount of light following random paths between light sources and sensors.

Here are a few examples of such paths, sorted roughly in increasing difficulty of simulation:

- direct connection between light source and sensor.
- from light source onto a surface and then to the sensor, also called *direct light*.
- bouncing many times between scene surfaces. This is still relatively easy to simulate if only few non-specular materials are involved. An example is light scattered by walls, illuminating shadows that would be completely black otherwise.

- it includes many specular bounces, for instance glass objects with a lot of refractions or caustics (bundled light).
- the path includes participating media (volumetrics), for instance translucent skin, fog, smoke, cloudy fluids like milk, wax etc.

Monte Carlo (MC) and Markov Chain Monte Carlo (MCMC) methods are two classes of algorithms used for simulating light transport. Both of them generate random samples – light paths directed through the image pixels. The colour of each pixel is computed as the-per pixel average of the light intensities of those paths. The estimate becomes more precise when taking more and more samples. In other words, the algorithms converge to the solution.

MC and MCMC methods differ in the way samples are generated, which has implications on the speed and type of convergence.

- MC methods generate independent samples. The accuracy of the solution can be improved by concentrating samples on important paths (those transporting the majority of light).
- MCMC methods, on the other hand, mutate existing samples, generating correlated chains. Small mutations allow the chain to stay in the *vicinity* of important paths, while large mutations are used for exploration. Accuracy depends on the quality of the mutation set.

Both methods are *unbiased*, meaning that there is no systematic error, only noise due to variance. There are also several *biased* approximation methods, like for instance *photon mapping*. In this thesis, we will focus on unbiased algorithms, and we neglect participating media like fog, smoke, milk, wax etc.

1.1 Problem statement

In order to improve rendering algorithms, it is advisable to gain an understanding of their shortcomings and to have a reliable metric to quantify differences. Researchers have an understanding of the shortcomings of competing light-transport algorithms, but some of the knowledge is anecdotal, especially in the case of MCMC. Currently used techniques for assessing errors are simplistic and often inadequate for understanding their sources, making it difficult to evaluate algorithmic changes.

The asymptotic convergence rate of MC and MCMC algorithms is $\Theta(1/N)$ (N being the number of samples) [Dut96][APSS01], but this is a general and coarse statement inadequate for measurements. Furthermore, MC algorithms are generally known to converge steadily, meaning there are no jumps of the error over time ($\sim$ sample count). MCMC algorithms, on the other hand, can converge in an unpredictable manner. For

instance, certain lighting effects can be completely missing until they "pop up" when a certain batch of samples is added to the solution. This is the type of anecdotal knowledge we mentioned. We found no study on those effects.

Often the convergence rate is compared simply by visual evaluation or using basic error metrics, such as the Mean Square Error (MSE) after a fixed rendering time [VG97][CTE05][MKA+15]. Visual methods are imprecise and subjective. Simple metrics tell nothing about error properties like spatial frequency content. Having a fixed rendering time prevents measurement of temporal uniformity.

We believe that there is a need for a better way of assessing rendering error and its properties. Building on that, an in-depth study of convergence properties of light transport algorithms would benefit the field by formalising anecdotal knowledge.

1.2 Contribution

Our main contribution is an error descriptor which tackles the problem of measuring and visualising differences of MC and MCMC rendering algorithms. In particular, it summarises the overall convergence rate over different frequencies, or in other words, the constant factor of the $\Theta(1/N)$ rate in different frequencies. Additionally, it visualises the intensity of outliers, which are responsible for said jumps and "pop-ups".

The descriptor can be computed for any algorithm for which it is possible to separate the solution into batches of independent samples. Adapting any unbiased algorithm into such a form is trivial but might change the performance. We study the performance change in case of a MCMC algorithm (Metropolis Light Transport (MLT)) and offer a possible workaround for algorithms not fitting the scheme.

As a second contribution, we discuss statistical error properties of rendering algorithms, both in the spatial and the frequency domain. We investigate error distributions based on the Central Limit Theorem (CLT), show how they are transformed when computing the power spectrum and study the influence of outliers. These results document and formalise some knowledge that was only anecdotal previously and help to understand and apply the descriptor.

Finally, we run a number of test cases with the following goals:

- verify the descriptor
- explain how the "pop-up" effects mentioned in Section 1.1 are reflected in the descriptor
- highlight different properties of MC and MCMC algorithms
- investigate the influence of various parameters on MCMC

In particular, we run the measure for several well-known scenes with different challenges, one scene with parametrized light size and surface roughness, one test to visualise the behaviour of photon mapping, varying mutation sizes of Primary Sample Space MLT (PSSMLT), various MLT seeding methods and a comparison between Energy Redistribution Path Tracing (ERPT) and MLT with parameters set to mimic certain ERPT properties.

1.3 Structure of the work

We will explain important basics of light transport algorithms, MC and MCMC integration as well as the Fourier frequency transform in Chapter 2.

In Chapter 3 we will summarise the state of the art and other related research.

Chapter 4 surveys statistical properties of the 2D error signal in the spatial and the frequency domain and gives insights into the temporal behaviour when taking more and more samples. It is also useful for understanding the proposed descriptor, which we introduce in Chapter 5.

Chapter 6 contains insights and findings that we got using the descriptor. We will conclude in Chapter 7.

Descriptions and full results for the scenes are in Appendix A, some of the pitfalls we encountered during the work in Appendix D, a brief reference of tested rendering algorithms in Appendix E, and a table of symbols in Appendix F.

CHAPTER 2

Background

In this chapter we want to provide some background knowledge that is needed to understand other chapters in this work. Rather than delve into mathematical details, we will try to give intuitive understanding and references to literature.

2.1 Light transport

As mentioned in Chapter 1, light transport algorithms are used to compute an image of a virtual scene using a physically based model. We explain the path integral formulation of light transport, because it is easier to use with some of the rendering methods [VG94]. It is an equivalent alternative to the recursive formulation [Kaj86].

2.1.1 The path

We cover the physical model more in depth, but we start with one of the centrepieces: the path (Figure 2.1).

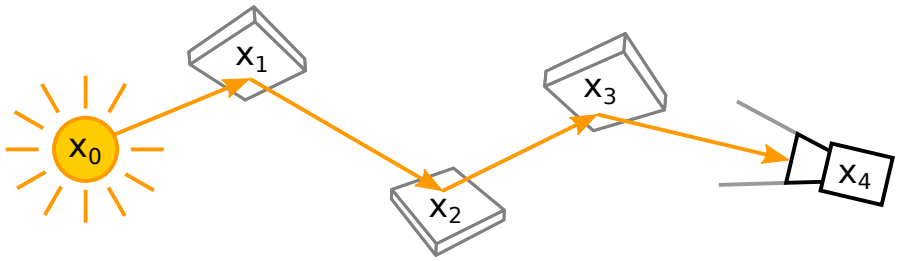


Figure 2.1: Example path from light source to camera. Edges between vertices ($x_0 \dots x_4$) are straight lines. We use the convention that path indices start at the light source.

A path denotes a list of vertices $x_0 \dots x_{D-1}$, which connect light source (x_0) and sensor (x_{D-1}) using straight edges. The vertices are positions in 3D space, augmented with information like the material and surface normal. The shortest possible path is $D = 2$, when the light source is visible in the camera. Direct lighting involves a third vertex on a surface, and each bounce adds more. In theory the number of vertices can be infinite, but in practice the number is limited by computational constrains.

When there is no participating media, photons go straight until stopped by an object – all vertices lie on surfaces. Therefore paths can be created by ray-tracing: A ray has a starting point and a direction. *Casting* it means finding its destination, the closest-hit point. The scene itself is modelled using triangle meshes. Hence, the closest hit point can be found by analytical ray-triangle intersections. The ray tracing process starts at an existing vertex, for instance the camera. The algorithm determines a new direction and casts a ray. The hit point becomes a new vertex and we repeat the process until a connection between sensor and light source is found.

2.1.2 The light transport function

Now that we know what a path is, and we have a basic understanding on how it is created, let us discuss how to use it for measuring light. The number and wavelength of photons travelling through path x to a certain sensor element is called the *flux*. It is given by the *light transport function* $f(x)$. All physical effects are packed into that function, defined as a product:

$$f(x) = L(x_0 \to x_1)G(x_0 \leftrightarrow x_1) \prod_{p=1}^{D-2} BSDF(x_{p-1} \to x_p \to x_{p+1})G(x_p \leftrightarrow x_{p+1}), \quad (2.1)$$

where L is the light intensity emitted from the light source, $BSDF$ the factors representing material properties and G geometry factors (Figure 2.2).

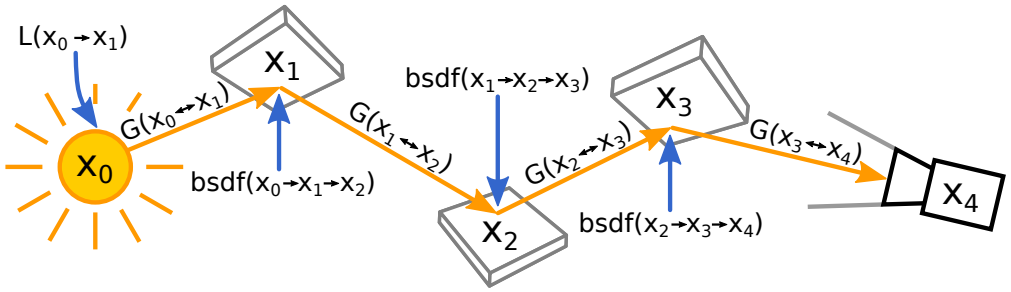
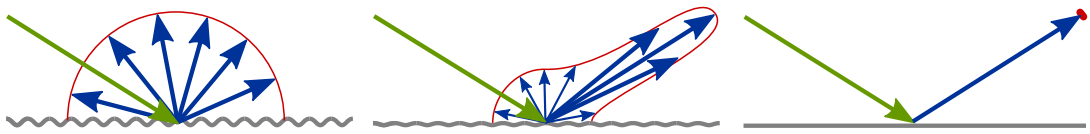


Figure 2.2: An example path of length $P = 5$ with light transport factors: Light intensity (L), geometry (G) and reflectance properties of the material ($bsdf$) are multiplied, resulting in the flux.



(a) Diffuse *(b) Glossy* *(c) Mirror (Dirac impulse)*

Figure 2.3: Examples of BSDFs: arrows symbolise photons and the envelopes the value returned for a specific outgoing direction.

The Bidirectional Scattering Distribution Function (BSDF) models the material of a surface, i.e., what happens, when a photon of a certain wavelength hits the surface.

There are several options:

- reflection
- absorption (energy is turned into heat, the photon disappears)
- refraction (the photon penetrates the object)

The function (Equation 2.2) takes the incoming ($\vec{\omega}_{in}$) and outgoing ($\vec{\omega}_{out}$) directions and returns the ratio of light transported per wavelength (colour channel). Often, the position $\vec{x}$ on the object is needed for spatially varying parameters like textures. This interface allows to model complex materials independently from the light transport algorithm. However, for this explanation it is more convenient to pass the position of previous, current and next vertices:

$$BSDF(\vec{x}, \vec{\omega}_{in}, \vec{\omega}_{out}) \equiv BSDF(x_{from} \to x \to x_{to}). \tag{2.2}$$

The notations are equivalent as incoming and outgoing directions can be computed – and positions can be ray traced. In a typical rendering program, BSDFs are often accompanied by functions that generate (sample) outgoing path directions. The sampling probability is proportional to the BSDF value, which helps to find paths that transport a lot of light.

As an example, the BSDF of a diffuse white wall paint is constant (Figure 2.3a), for plastic the value is larger in reflection direction (Figure 2.3b) and for a perfect mirror it is zero in all but the reflecting direction (Figure 2.3c). In accordance to the energy conservation law the integral over all directions is less or equal one. More details about the physical background and mathematical models can be found for instance in an article by Hoffman [Hof13].

We continue with light sources and sensors. The problem of following photons until they hit a sensor is reciprocal to seeking incoming radiation starting from the camera. The dual to photons is called *importons* [Vea97]. They are virtual particles emitted from the sensor, which contribute to the measurement when hitting a light source.

Physically, both, light sources and sensors, have a non-zero area and they emit or receive particles from a range of directions. The amount and wavelengths of light, that flows from position $\vec{x}$ on the light source into direction $\vec{\omega}_{out}$, is modelled by

$$L(\vec{x}, \vec{\omega}_{out}) \equiv L(x \to y). \tag{2.3}$$

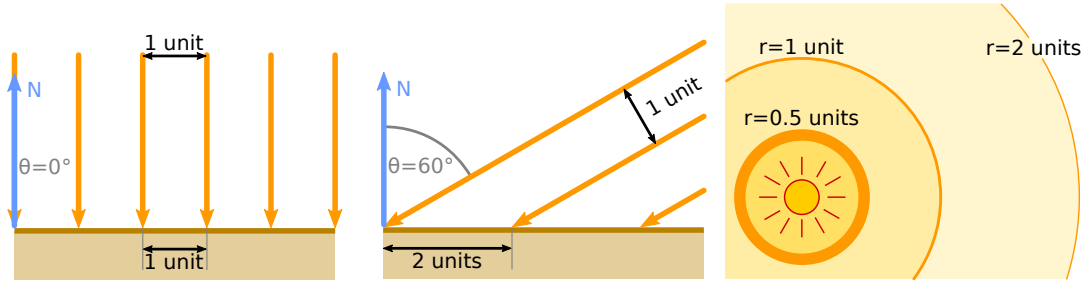
Again, the notation using vertices is equivalent. A similar function exists for the sensor, measuring the amount of importance flowing $x_{D-1} \to x_{D-2}$. We omit it, since it is 1 for all admissible paths (those that go through a pixel).

In computer graphics, light sources and sensors can be modelled as the limit of being infinitely small (a point) or having an infinitely small range of directions (singular direction). These simplifications make the computation easier at the expense of realism and in some cases a biased solution.

The last factor is G, the geometry factor. It consists of the visibility V, Lambert's law (see Figure 2.4a) and inverse-square law (see Figure 2.4b):

$$G(x \to y) = V(x \leftrightarrow y) \frac{\cos(\theta_{x \to y}) \cos(\theta_{y \to x})}{|\vec{x} - \vec{y}|^2}, \tag{2.4}$$

where path vertex x has the position $\vec{x}$ and $\theta_{x \to y}$ is the angle between the normal of x and a vector pointing from $\vec{x}$ to $\vec{y}$. Sometimes a visibility term (V) is needed (0 if there is a triangle between x and y and 1 otherwise). G is symmetric and it applies equally to photons and importons.



(a) $\cos(\theta)$, *cos of angle between incoming and normal vector* (b) *Distance*

Figure 2.4: Geometry factors in light transport: (a) Lambert's law, light or importance intensity on a surface is proportional to $\cos(\theta)$*. (b) The density of photons and importons in a spherical wave front is proportional to the inverse-square distance.*

2.1.3 Sampling

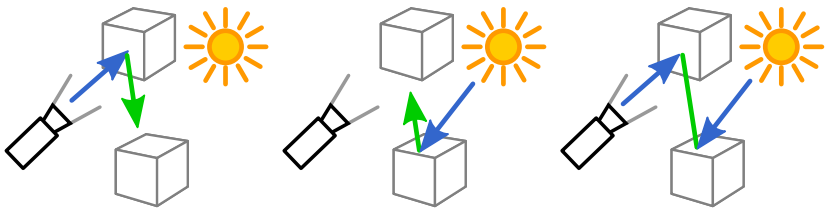
At this point we want to revisit the process of creating a path – sampling. It is desirable to find *important* paths that transport a lot of light.

Sampling usually starts at the camera or a light source (Figures 2.5a and b). At every step, the existing path is extended by sampling an outgoing direction at the last vertex. Important paths are found with a higher probability, if the BSDF and Lambert's law (Figure 2.4a) are taken into consideration while selecting the direction. Ray casts inherently create samples with an intensity proportional to the inverse-square law and Labert's law at the destination vertex.

Some algorithms have the ability to connect existing sub paths (Figure 2.5c). It is also possible to employ global information. For instance the light source is often sampled directly. In those case the visibility term of G comes into play.

We want to conclude with some examples of hard or impossible cases for the sampling process. The smaller the light source (or sensor), the harder it is to sample locally (without information about its position and size), see Figure 2.6. The range of important directions shrinks, which manifests as larger error in the resulting estimate. In the limiting case of a point light source, local sampling is not even possible. Something similar happens, when the distance to the light source is increased.

Here, direct sampling would be a solution. But that is not always possible, for instance, if the light source is obscured by a transparent object (Figure 2.7). The law of refraction



(a) Light direction *(b) Eye direction* *(c) Connection*

Figure 2.5: It is possible to extend (sample) a path in light (a) or eye direction (b). Some algorithms have the ability to connect sub paths (c).

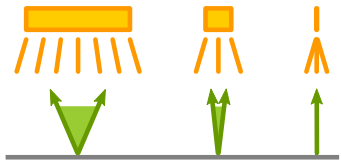


Figure 2.6: A light source of decreasing size (left to right) demonstrates how sampling an important direction randomly becomes more difficult for smaller ranges. In the limit, when only a singular direction is important (point light source), it is impossible to sample it randomly.

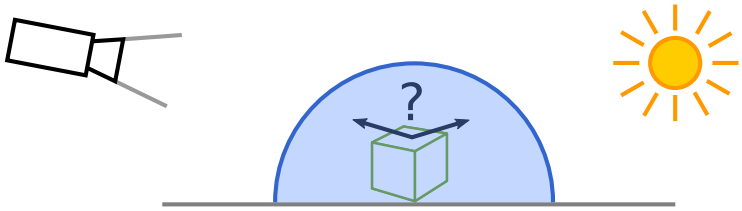


Figure 2.7: An object inside a perfectly specular glass bowl: This scene is impossible to render with a traditional MC or MCMC renderer if using a pinhole camera and a point light source.

must be respected, often changing the direction of the path.

In practice, error is often high when the light source is small or far away and only visible through specular bounces.

2.1.4 The path integral formulation of light transport

Veach and Guibas [VG94] formulated rendering as the solution to the following integral:

$$I[j] = \int_{\Omega} f(x) h_j(x) \, \mathrm{d}\mu(x), \tag{2.5}$$

where

- $I[j]$ is the brightness and colour of pixel j.
- Ω is the space of all transport paths, i.e., the union of all paths of length 2, 3 $\ldots \infty$.
- Accordingly x is a transport path of variable length and $f(x)$ its flux, computed by the light transport function.
- Since f has no notion of pixels, we need an additional pixel filter h_j. It is zero for most pixels, and only allows a contribution if the path x goes through the area assigned to pixel j. This separation is useful, for instance because a path can contribute to more than one pixel (anti-aliasing when the path is close to the pixel edge).
- μ is a measure on Ω, needed for integration.

The integration domain Ω is infinitely dimensional although it is embedded in the 3D scene. Every vertex in x adds a degree of freedom (a new dimension), and the vertex count is not limited.

In theory, any numerical method can be used to solve the integral. However, in practice simple, quadrature rules are prohibitive with higher dimensional integrals. Hence, MC (Section 2.3) or MCMC methods (Section 2.4) are employed.

2.2 Statistics

Next we will define a few statistical terms that we use later and remind about some important statistical properties.

A random variable is one that takes up a value by chance. Calculations including random variables result in transformed random variables. Observations of random variables have a non zero value in the random variable's Probability Density Function (PDF). The integral of the PDF over the whole domain is 1. In our context, we often have random variable and observation vectors, for instance a 2D image or spectrum.

Random variables can have an expectation denoted by $\mathrm{E}[X]$. The expectation operator is linear:

$$\mathrm{E}[aX] = a\,\mathrm{E}[X]$$

and

$$\mathrm{E}[X+Y] = \mathrm{E}[X] + \mathrm{E}[Y],$$

which is even true if X and Y are correlated.

We notate the mean of N observations of X as

$$\hat{X}_N = \frac{1}{N}\sum_{i=1}^{N} X_i.$$

There are many ways to measure dispersion, that is how much a distribution is spread out. Examples are standard deviation, variance, ratios of quantiles, etc.

Variance is defined as

$$\mathrm{Var}(X) = \mathrm{E}\left[(X - \mathrm{E}[X])^2\right] = \mathrm{E}\left[X^2\right] - \mathrm{E}[X]^2$$

and Kurtosis as

$$\mathrm{Kurt}[X] = \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2} \quad \text{with } \mu = \mathrm{E}[X].$$

The following rules apply to variance:

$$\mathrm{Var}(aX+b) = a^2\,\mathrm{Var}(X), \tag{2.6}$$

$$\mathrm{Var}\left(\sum_{i=1}^{N} X_i\right) = \sum_{i,j=1}^{N} \mathrm{Cov}(X_i, X_j) = \sum_{i}^{N} \mathrm{Var}(X_i) + \sum_{i\neq j} \mathrm{Cov}(X_i, X_j), \tag{2.7}$$

where Cov is the covariance.

We use the Central Limit Theorem (CLT) several times. It states that when taking the mean of $N \to \infty$ independent and identically distributed (i.i.d.) random variables (X)

with well defined expectation μ and finite variance σ^2, the resulting random variable $(\hat{X}_N)$ will tend towards a normal distribution:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} X_i = \hat{X}_N \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{N}\right).$$

The overall convergence rate of $\Theta(1/N)$ for MC methods is a direct consequence of it.

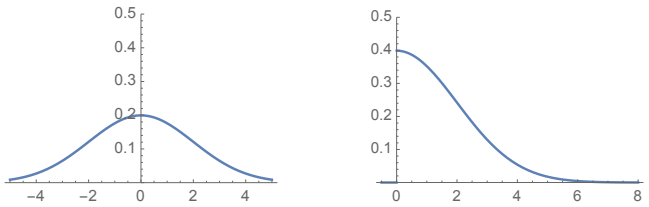
2.2.1 Probability distributions

There are several probability distributions based on the normal distribution ($\mathcal{N}$). We use the half normal distribution (Figure 2.8), which arises when taking the absolute value of a variable that is normal distributed:

$$Y = |X|, \quad X \sim \mathcal{N}(0, \sigma).$$

Its expectation and variance can be computed analytically, if the original variance is known:

$$\begin{aligned} \mathrm{E}[Y] &= \sigma\sqrt{2/\pi} \\ \mathrm{Var}(Y) &= \sigma^2(1 - 2/\pi). \end{aligned}$$



(a) Normal *(b) Half normal*

Figure 2.8: Probability density functions with $\mu = 0$ and $\sigma = 2$. See text for definitions.

2.3 Integration using Monte Carlo methods (MC)

MC is a numerical integration technique applicable for any function f, though for low dimensional functions there are methods with a better asymptotic convergence rate. In its simplest form N samples (x_i) are drawn uniformly in the integration domain Ω and then combined to the solution using [NB99][Liu01]:

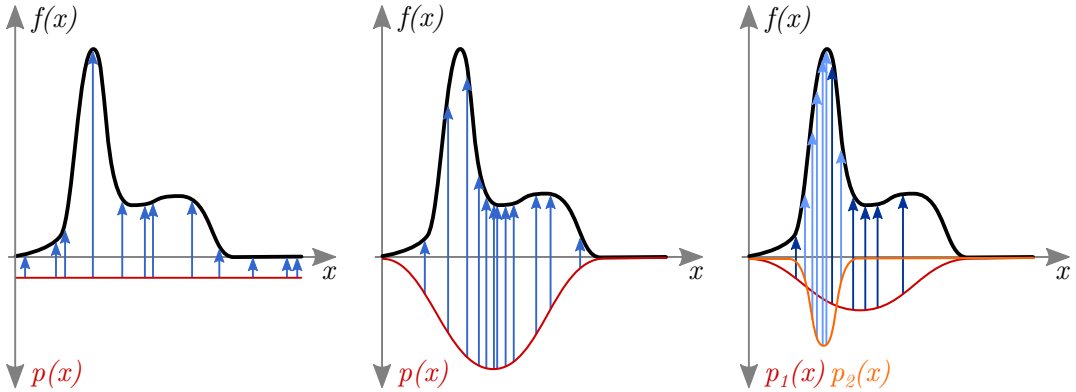
$$I = \int_\Omega f(x)\,\mathrm{d}x \approx V\frac{1}{N}\sum_{i=1}^{N} f(x_i) = \hat{I}_N \tag{2.8}$$

$$\text{with } V = \int_\Omega 1\,\mathrm{d}x. \tag{2.9}$$

This algorithm is unbiased, meaning that $I = \mathrm{E}\left[\hat{I}_N\right]$, the expectation is equal to the correct result. However, samples can fall into regions with $f(x_i) \approx 0$, the contribution to $\hat{I}_N$ is minimal (Figure 2.9a). Even worse, fewer samples fall into the important part, increasing variance of the estimate. It would be better to take more samples in areas where f is large and weight accordingly. This so-called *importance sampling* (Figure 2.9b) takes samples using a probability distribution that follows f as closely as possible. Equation

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} \tag{2.10}$$

is used to calculate the importance sampled estimate (we omit V from now on) [NB99]. Note that if probability p is exactly proportional to f then there is no variance at all, and the result is just a scaling factor.



(a) Uniform (b) Importance (c) Multiple importance

Figure 2.9: Difference between naive uniform sampling and importance sampling methods

One drawback of importance sampling is that it is often hard to come up with a single sample-generating function that follows f everywhere. Variance can increase tremendously when there are areas with a small p and a large f. Multiple Importance Sampling (MIS) can help, as several different sample-generation strategies with a combined probability are used. Figure 2.9c gives an idea how the samples can be distributed using two normal distributions. There are several ways on how the samples can be weighted, see Veach and Guibas [VG95]. Equation

$$\hat{I}_N = \frac{1}{N} \sum_{i=1}^{N} \frac{2f(x_i)}{p_1(x_i) + p_2(x_i)} \tag{2.11}$$

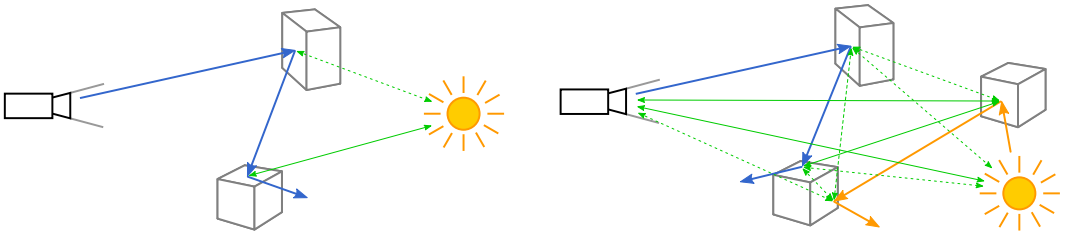
is just an example. Two sampling functions with sample probability p_1 and p_2 are used, we take 50% of the samples from strategy 1 and 50% from 2. The probability functions are added up below the fraction line, which reduces the aforementioned problems with a small p and large f.

$\hat{I}_N$ is a random variable with expected value equal to the solution of the integral. The variance of this random variable is a measure of the expected error and depends on the shape of f, the quality of the sampling function, and it decreases to zero asymptotically with the number of samples as $\Theta(1/N)$ [NB99].

2.3.1 Application in rendering

We use two MC algorithms in this thesis: Path Tracing (PT) and BiDirectional Path Tracing (BDPT). Both are visualised in Figure 2.10.

In PT all paths start at the camera, which means that the rays trace importons. At every surface hit point, the BSDF is importance sampled and the path terminates when a light source is hit. Alternatively, shadow rays are cast from every vertex (thin visibility test lines in Figure 2.10a). This increases the number of samples while being relatively cheap,



(a) Path tracing *(b) Bidirectional path tracing*

Figure 2.10: Thin lines represent visibility tests, dotted ones failed visibility tests and thick ones ray casts. Every positive visibility test is a connection between camera and light, i.e., a path sample *that contributes to the result. BDPT (b) results in more path samples per pixel sample (thick lines starting at the camera), hence one pixel sample is more expensive than in PT (a).*

as every connection to a light source is a separate sample. It is possible to perform MIS between the shadow rays and the BSDF rays cast. PT is biased if the scene contains perfect glass and a point light source, because the caustic cannot be sampled (compare with Figure 2.7). Accordingly, it has high variance with a small area light [PH10].

BDPT is more complex. First, camera and light paths are created, importance sampling the BSDFs. Then, each vertex of the camera path is connected to every visible light vertex (Figure 2.10b). This creates a number of camera-light connecting paths, the samples. Each path was created using a different sampling strategy (e.g. 1 camera vertex, D light vertices, 2 camera vertices, $D-1$ light vertices, and so on). When looking at one particular path, in theory it could have been also created by any of the other strategies. Hence, we can perform MIS between them [Vea97].

2.4 Integration using Markov Chain Monte Carlo methods (MCMC)

In MC, all samples are independent from each other. This means that if a sample with large f is found, the next sample cannot benefit from it. In MCMC on the other hand, every sample is derived from the previous one – we say that a sample is mutated. All samples together form a Markov chain of samples. The mutations are designed in such a way, that the distribution of samples is proportional to $f(x)$ in the limit, also called equilibrium distribution. This means that we can generate samples proportional to f, something very desirable when looking back to importance sampling.

We start with the mutations and find out later how everything fits together.

2.4.1 Mutations

The mutations are implemented by modifying an existing sample, generating a candidate mutation. This candidate is then either accepted or rejected as the new state, guaranteeing that the sample states are proportional to $f(x)$. Because of the guarantee we can use any type of modification – as long as we can compute the acceptance probability.

There are many variants for calculating the acceptance probability of a mutation $x \to x'$ [BGJM11]. One of the important variants, also used in Metropolis Light Transport (MLT), is that of Metropolis-Hasting [Gey11]. Its acceptance probability is calculated from old and new f as well as the probabilities T of generating x' from x and vice versa:

$$\mathrm{a}(x \to x') = \min\left(1, \frac{f(x')\,\mathrm{T}(x' \to x)}{f(x)\,\mathrm{T}(x \to x')}\right) \tag{2.12}$$

$$x_{i+1} = \begin{cases} x'_i, & \text{if uniform\_rand}[0,1] < \mathrm{a}(x_i \to x'_i) \\ x_i, & \text{else} \end{cases}. \tag{2.13}$$

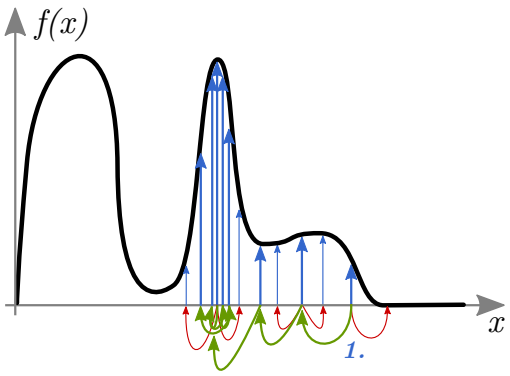


Figure 2.11: Example of a Markov Chain. Candidates are generated by $x' = x +$ uniform_rand$[-1, 1]$. *Accepted candidates are thick, rejected thin.*

Let us take a look at the example in Figure 2.11. Candidates are generated by modifying the old state by $x' = x +$ uniform_rand$[-1, 1]$ – resulting in transition probabilities of $\mathrm{T}(x' \to x) = \mathrm{T}(x \to x') = 0.5$. Hence we have an acceptance probability of $\mathrm{a}(x \to x') = \min(1, f(x')/f(x))$: Samples with a lower contribution are rejected probabilistically and with a higher one always accepted. That is all that is needed for our samples to be distributed proportional to f in the limit.

In the example, the chain does not reach the left part, because it is stuck in the high-contribution part in the middle. The sample count is too small to overcome the ditch. This is one of the drawbacks of MCMC, it is impossible to say whether the equilibrium distribution was reached or not [BGJM11].

The mutations used for generating new samples should fulfil certain desirable properties [VG97]:

- high acceptance probability, otherwise the chain will remain in the same state for a long time, producing outliers.
- large enough changes, otherwise the chain moves too slowly around the integration domain, possibly not reaching equilibrium within the sampling budget and producing correlated noise.
- ergodicity, the chain must be able to reach all of the integration domain.
- stratification, so that all of the integration domain is explored equally.
- low cost

Often one single mutation strategy cannot fulfil all of those requirements, therefore algorithms choose probabilistically between several strategies. Even then, designing a strategy is very challenging [KSKAC02][Jak13][HKD15].

2.4.2 Integration

The probability of sample x is by design

$$p(x) = \frac{f(x)}{b}, \tag{2.14}$$

with an unknown b that scales f in such a way that p integrates to 1. Let us try to plug that into Equation 2.10 from MC importance sampling:

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} = \frac{1}{N}\sum_{i=1}^{N}\frac{bf(x_i)}{f(x_i)} = b.$$

That does not work. b is the value that we are trying to estimate, we have gained nothing.

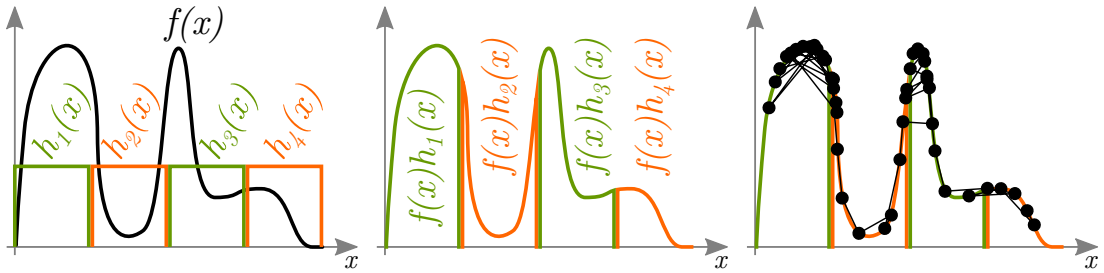
It is not possible to use Metropolis sampling to integrate the function that is being sampled. Instead, we sample one function (f) and integrate another ($f \times h$). In fact, we integrate many at once, all pixels at once.

Figure 2.12 shows how that works for the toy example. As shown in Section 2.1.4, the integral of one pixel is

$$I[j] = \int_\Omega f(x)h_j(x)\,\mathrm{d}\mu(x).$$

We can generate samples for $f(x)$, which are then used for all integrals (pixels) $I[j]$, $j = 1 \dots M$. The integration rule is

$$I[j] = \int_\Omega h_j(x)f(x)\,\mathrm{d}x = \mathrm{E}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{h_j(x)f(x_i)}{p(x_i)}\right] = \mathrm{E}\left[\frac{b}{N}\sum_{i=1}^{N}h_j(x_i)\right]. \tag{2.15}$$



(a) f and h_j pixel filter (b) Multiplied (c) MCMC sampling

Figure 2.12: We estimate several integrals at once: The Markov chain produces samples proportional to f (a and c) in the whole domain (pixel plane). Each integral (pixel) has a corresponding filter h_j (a) and the actual integrands are fh_j (b). The number of Markov states in every integral is proportional to f (c). Counting them gives a relative estimate.

Samples x_i are generated using the Markov chain. It is then counted how often the chain visits pixel j (assuming a rectangular filter h). Finally the value is scaled by b/N.

We still do not know the value of b. However, we know that it is the integral of f – since it scales f to a PDF. Remember that we defined f to be zero if the path does not end at the sensor. We see that b is the total energy falling on the sensor and that it can be estimated using an alternative Monte Carlo strategy with probability p_o:

$$b = \int_\Omega f(x)\,\mathrm{d}x = \mathrm{E}\left[\frac{1}{L}\sum_{i=1}^{L}\frac{f(x_i)}{p_0(x_i)}\right] = \mathrm{E}\left[\hat{W}_{(L)}\right]. \tag{2.16}$$

The final integration rule is

$$\hat{I}_N[j] = \frac{\hat{W}}{N}\sum_{i=1}^{N} h_j(x_i). \tag{2.17}$$

We conclude, that the Markov chain merely computes the relative contributions of the pixels while the overall brightness is computed in a separate step.

2.4.3 Seeding and start-up bias

We still do not know how to start the chain. The acceptance rule in Equation 2.12 only guarantees that the sample probability will be proportional to f in the limit, not at the start. Clearly, in Figure 2.11 the limit was not reached yet. One solution for this problem of start-up bias is to run the chain for a long enough time while throwing away the samples, a so called burn-in period. Geyer says this method is "fishy" [Gey11], because it is impossible to tell how many samples should be thrown away, and, that a good alternative is a seed that you do not mind having as a sample.

Generating such an acceptable sample via one of the MC methods would be easy, but the special setting of rendering allows us to remove start-up bias completely[1]. The following method was introduced by Metropolis Light Transport (MLT).

We generate a pool of L starting paths (also called luminance samples) using the alternative strategy p_0 (Figure 2.13). The weight $\hat{W}$ is the average of path intensities in the pool, i.e., $\hat{W}_L = 1/L\sum_{i=1}^{L} f(x_i)/p_0(x_i)$. The starting path is re-sampled from the pool, using a discrete probability distribution proportional to $f(x_i)/p_0(x_i)$.

With a large pool size, the weight W is relatively precise and the re-sampling leads to a starting path, that is approximately distributed proportional to f. With a small pool size there is more start-up bias, but that is compensated by W, overall resulting in an unbiased estimator. All proofs can be found in Veach's PhD thesis [Vea97], which is also the base for this summary.

[1] The presented method is not the only one to remove start-up bias. For instance *coupling from the past* [BGJM11] is an alternative used in other areas.

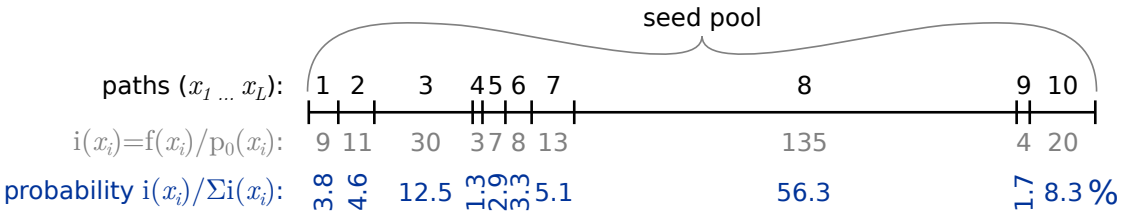


Figure 2.13: The starting path of an MLT chain is generated as follows: A large number L of paths x_i is generated using an alternative strategy p_0 and stored in a seed pool. Starting paths are then sampled from the pool using a discrete probability distribution proportional to $f(x_i)/p_0(x_i)$. The same weight $W = \frac{1}{L}\sum_{i=1}^{L} f(x_i)/p_0(x_i)$ is used for all paths sampled from the pool. It is possible that two or more chains are started from the same starting path.

It is possible to start several chains from the same pool by repeatedly sampling the discrete distribution. This can result in several chains starting from the same seed path, creating correlation between chains. Using more uniformly distributed quasi random numbers or stratified sampling can help here. It is even possible to use non-random equally spaced grid positions.

2.4.4 Application in rendering

We will cover two MLT algorithms: path space MLT (abbreviated just MLT) and Primary Sample Space MLT (PSSMLT). Both of them are using the basic algorithm and seeding described above, but the mutations are different. Additionally we briefly explain Energy Redistribution Path Tracing (ERPT), which is not an MLT algorithm, but uses the same mutations as path space MLT.

Path space path space MLT (MLT), also called *Veach-MLT* [VG97], operates directly on path space, e.g. the mutations have access to the materials, vertex positions and global data like information about lights etc. This makes it possible to create very powerful mutations. For instance Jakob and Marschner [JS12] created a mutation that exploits surface derivatives in order to improve convergence of specular reflections (Manifold Exploration path space MLT (MEMLT)). A drawback is the high complexity of probability computation for mutations.

This was part of the motivation for PSSMLT, also called Kelemen-style MLT [KSKAC02], which runs the default sampling code of PT or BDPT. There are two types of mutations, large and small ones, with a parameter setting the percentage of large ones. Small mutations are generated by a specially adapted random number generator. Instead of mutating in path space, random numbers are mutated and the PT or BDPT sampling code is used to create the path. If no path can be created using the new random numbers, the mutation is automatically rejected. Size of small mutations is determined probabilistically, based on a parameter. Large mutations on the other hand create a new and independent path. This makes the mutation set ergodic, i.e., they are used

for exploring path space. Mutation probabilities can be completely ignored as they are symmetric.

ERPT is an MCMC method based on a large number of short and independent Markov chains. Cline et al. [CTE05] were able to relax certain conditions which led to the acceptance probability in Equation 2.12. They replaced the re-sampling process by probabilistically sampling starting paths using MC methods, and they adapted and removed certain mutation types. This led to an algorithm, which redistributes the energy of independent MC samples into a small number of correlated Markov chain samples. The advantage is the usage of MC methods to explore the integration domain, while retaining the property of reusing samples with a large f.

2.5 Photon mapping

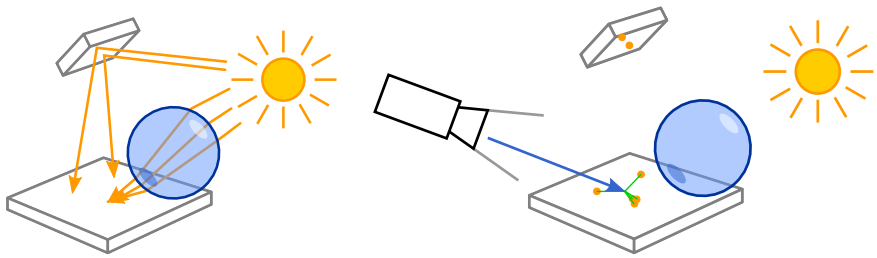
We also want to briefly explain Photon mapping. It is not core to this work, but we do cover it in the result section.

In contrast to MC and MCMC methods, photon mapping is biased. However, it is consistent, meaning that it would converge to the true solution given infinite time and memory. It was first published by Jensen [Jen96] and several improvements followed [HJ08][HJ09].

Photon mapping consists of two phases: mapping and gathering. In the mapping phase a large number of photons are traced from all light sources and each surface-hit-point (Figure 2.14a) is stored in a tree structure – the camera is not used. In the gathering phase, rays are traced from the camera, similar to PT. On each hit-point the photon map is queried for close-by photons, which are then used for approximating the amount of light falling on that surface (Figure 2.14b).

Obviously the approximation is more precise with a larger number of photons, and becomes exact in the limit.

An improved versions of the method is Stochastic Progressive Photon Mapping (SPPM). It casts gathering points from the camera into the scene. In the second processing phase,



(a) Mapping *(b) Gathering*

Figure 2.14: Two phases of photon mapping

photons are shot from the light sources (the phases are reversed). When they reach the vicinity of a gathering point, the contribution is added to the corresponding pixel. Those two phases are iterated while gradually reducing the radius of vicinity. This iteration improves the quality of the rendering over time without storing an infinite number of photons [HJ09].

2.6 Frequency analysis

In this work we analyse two dimensional discrete data, focusing on the frequency content. Possibilities of transforming spatial data into the frequency domain are Fourier and related transforms [Boa66], wavelet transforms [DS02], steerable pyramids [FA91], and others. However, we limit ourselves to the well known Fourier transform.

This is only a very short summary, focusing on the discrete 2D case and some properties we need later. The basics are covered for instance in Osgood's introduction [Osg07].

Equation 2.18 shows the basic one dimensional complex version of the Discrete Fourier Transform (DFT). Every element of $\vec{F} = \mathcal{F}\vec{f}$ is a weighted sum of the input signal. Resulting factors can be transformed into a more illustrative sine and cosine form using Euler's Formula 2.19, but the packed complex number form is shorter in notation. Note that up to a constant scaling the sin factors are equal to imaginary – and cos to real – part of the complex numbers.

$$\vec{F}[x] = \sum_{m=0}^{M-1} \vec{f}[m] e^{-2\pi i x m/M}, x = 0, 1, ..., M-1 \tag{2.18}$$

$$e^{iy} = \cos(y) + i\sin(y) \tag{2.19}$$

It is further possible to modify the sin-cos-form into an amplitude-phase-form, e.g. $a\sin(t) + b\cos(t) = A\cos(t - \Phi)$. When applied to the coefficient vector, we have

$$\vec{A} = \left|\vec{F}\right| = \sqrt{\mathrm{Re}(\vec{F})^2 + \mathrm{Im}(\vec{F})^2} \tag{2.20}$$

$$\vec{\Phi} = \mathrm{atan2}(\mathrm{Im}(\vec{F}), \mathrm{Re}(\vec{F})), \tag{2.21}$$

using point-wise operations and the quadrant aware arctangent function atan2. In this representation A (amplitude) tells how strong a certain frequency wave is and Φ (phase) the position.

When squaring the amplitude, we get the power spectrum

$$\vec{F}_{ps} = \vec{A}^2 = \mathrm{Re}(\vec{F})^2 + \mathrm{Im}(\vec{F})^2. \tag{2.22}$$

$\vec{F}_{ps}$ quantifies the amount of power in a certain frequency and satisfies Parseval's theorem: The sum of Energy in frequency space is equal to M times the sum of Energy in spatial space

$$M \sum_{m=0}^{M-1} \vec{f}\,[m]^2 = \sum_{x=0}^{M-1} \vec{F}\,[x]^2 = \sum_{x=0}^{M-1} \vec{F_{ps}}[x]. \tag{2.23}$$

The above also applies to the two dimensional form – which we will use for the image analysis – defined as

$$\vec{F}[x,y] = \sum_{l=0}^{L-1} \sum_{m=0}^{M-1} \vec{f}[l,m] e^{-2\pi i(xl/L + ym/M)},$$
$$x = 0, 1, ..., L-1,\ y = 0, 1, ..., M-1.$$

Since we can arrange the sum freely, we can see the transform as

- applying a 1D DFT first in one direction and then in the other on the partially transformed signal
- every $\vec{F}[x,y]$ being a weighted sum of all elements $f[l,m]$
- in sin-cos-form, every $\vec{F}[x,y]$ being a dot products of the 2D signal $\vec{f}$ with a sin respectively cos plane wave (Figure 2.15). Its frequency and orientation depend on x and y from $\vec{F}[x,y]$.

An important property is linearity

$$\mathcal{F}(a\vec{B}) = a\mathcal{F}\vec{B}, \text{ and}$$
$$\mathcal{F}(\vec{A} + \vec{B}) = \mathcal{F}A + \mathcal{F}B.$$

Throughout this work we will use centred DFTs everywhere, meaning that the constant factor (*DC term*) and low frequencies will be in the middle. In our context elements of the signal vector in the spatial domain are pixels. Their analogy are the elements of Fourier coefficient vectors, which we call *Fourier frequencies*.

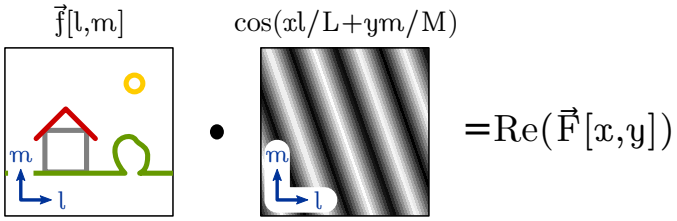
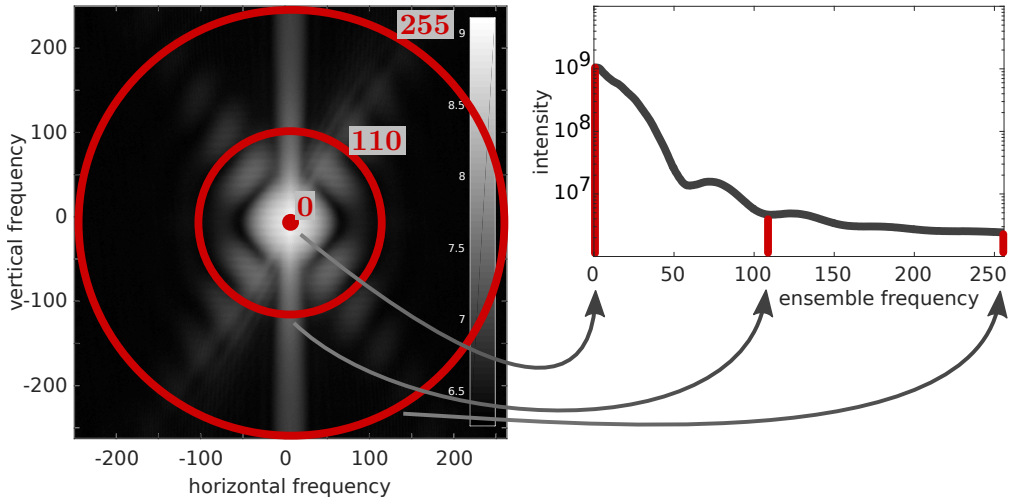


Figure 2.15: A 2D Fourier transform can be visualised as applying the dot product between the signal and a sine / cosine plane wave. Constants omitted, sinus / imaginary part analogous.

2.6.1 Radial average

When investigating a 2D spectrum, sometimes properties like phase and orientation of Fourier frequencies are of lesser importance. The spectrum can be compressed into a vector by performing a radial aggregate of the power spectrum (Figure 2.16). In this process all Fourier frequencies on a circle around the DC term are aggregated into one bucket, with the radius being the new *ensemble frequency*.

In many cases the aggregating function is averaging (hence *radial average*), but quantiles, sums and others are equally possible.



(a) error power spectrum *(b) radial average*

Figure 2.16: In order to reduce dimensionality, we aggregate (e.g. average) all Fourier frequencies (a) of a particular wavelength but different orientations into one bucket (i.e., a circle in Fourier space). (a) shows examples for frequencies 0 *(DC term),* 110 *and* 255*, which are averaged into one ensemble frequency each (b).*

2.7 Simple error measures

For reference we will show the definitions of some common error measures. $\vec{R}$ is the vector we are trying to estimate (the reference) and $\hat{\vec{I}}$ an observation of which the error is computed:

Measure	Formula
Mean Square Error (MSE)	$\frac{1}{M}\sum_{m=1}^{M}(\hat{\vec{I}}[m]-\vec{R}[m])^2$
Root Mean Square Error (RMSE)	$\sqrt{\frac{1}{M}\sum_{m=1}^{M}(\hat{\vec{I}}[m]-\vec{R}[m])^2}$
relative MSE	$\frac{1}{M}\sum_{m=1}^{M}((\hat{\vec{I}}[m]-\vec{R}[m])/(\epsilon+\vec{R}[m]))^2$
relative RMSE	$\sqrt{\frac{1}{M}\sum_{m=1}^{M}((\hat{\vec{I}}[m]-\vec{R}[m])/(\epsilon+\vec{R}[m]))^2}$.

Note that MSE and RMSE are conceptually similar to variance respectively standard deviation, but operate on M different random variables instead of a single one.

CHAPTER 3

Related Work

As mentioned in the problem statement in Section 1.1, convergence rates are often compared visually and by evaluating simple error metrics in the spatial domain. We show several examples of this method in Section 3.1.

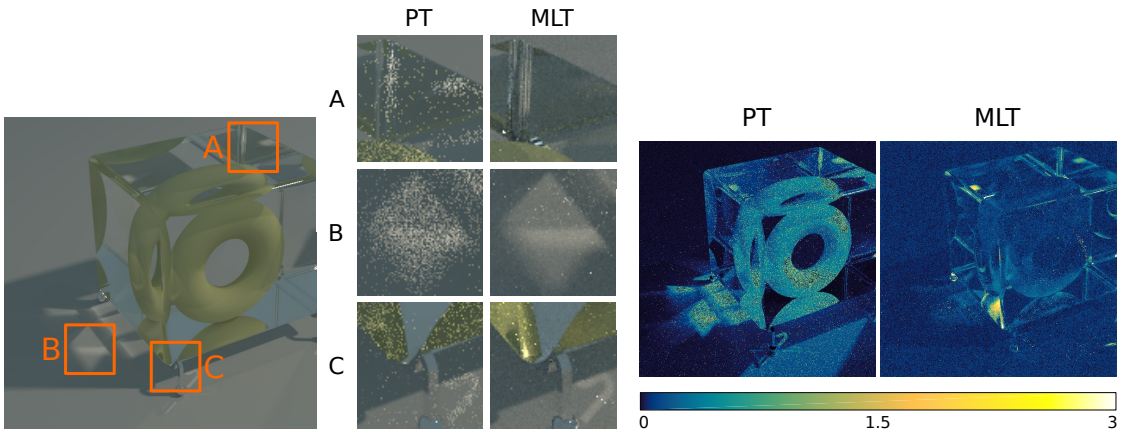
Although we are not aware of any established error – or convergence metric, research was conducted on the error made by rendering algorithms, sampling, and the complexity of path space. We summarise several papers in Sections 3.2 and 3.3.

Finally, there is some work related to one of our tools, the Fourier error spectrum. In signal processing, and in particular in medical imaging, noise power spectra are used. We cover some papers in Section 3.4.

3.1 Simple comparisons

Several papers directly compare equal-time renderings between old and new algorithms [VG95] [CTE05] [LKL+13] [KMA+15] [MKA+15]. Often, interesting details like *caustics* or reflections are shown in close-ups (Figure 3.1b). Some of the papers complement this with error images (Figure 3.1c) or provide full images in supplemental material or websites.

In addition or as an alternative, simple error measures are computed. There is no single established metric, examples are Peak Signal to Noise Ratio (PSNR) [LKL+13], MSE [KMA+15], RMSE (L2 error norm), [HKD14], relative MSE [MKA+15] and relative RMSE [VG97] (Figure 3.1d). Sometimes also L1 and L∞ norms are reported, or the error is expressed in percentage of a reference algorithm [VG97].



(a) Reference *(b) Details* *(c) Absolute Error* $\left|\vec{R} - \hat{\vec{I}}\right|$ *(clamped)*

	MSE	RMSE	relative MSE	relative RMSE	PSNR*
PT	0.002136	0.004622	0.1077	0.3282	74.83dB
MLT	0.002786	0.005278	0.1446	0.3802	73.68dB

(d) Error Metrics (higher is better)*

Figure 3.1: Examples of commonly used equal time comparisons.

3.2 Convergence and error

It is well known that MC methods have a variance of $\Theta(1/N)$, where N is the number of samples [Dut96][PH10]. The same rule applies to MCMC methods, though not for consecutive samples and only when equilibrium is reached by the Markov chain [Vea97][APSS01].

Arvo et al. identified three sources of error, input data -, discretisation - and computational error, and provide bounds [ATS94], but the paper is somewhat theoretical. We completely ignore the input and discretisation error, do not attempt to provide bounds, and simply measure the computational error instead.

Szirmay-Kalos et al. [SKDP99] analysed start-up bias and convergence of MLT algorithms, partly using the Fourier transform, but they did not consider Veach's technique to avoid the start-up bias altogether. In addition, a lot of their analysis is based on toy problems and not on real rendering algorithms.

Subr and Kautz investigated the Fourier spectrum of MC sampling patterns and how it affects the bias and variance of the pixel estimate [SK13], but they do not look at the error image as a whole and they do not look at MCMC methods. Pilleboue et al. analysed MC variance and the convergence rate of various sampling patterns for low-dimensional integrands [PSC+15]. Note that it is possible to achieve convergence rates better than $\Theta(1/N)$ using specialised sampling patterns for low dimensions. However,

this is not generalisable to higher-dimensional integrands found in scenes with complex light transport.

3.3 Path-space analysis

Durand et al. analyse the frequency content of the light-transport function for different phenomena [DHS+05]. They look at the actual radiance function and not at algorithms computing a pixel estimate. Belcour et al. extended that analysis to participating media [BBS14] and proposed improvements to existing algorithms.

Zirr et al. developed methods visualising the structure of light transport [ZAD15], but similarly to Durand their work is aimed at getting insight into the physics and not into the algorithms.

Kettunen et al. conducted a theoretical analysis of the sampling process in Gradient-Domain Path Tracing based on the Fourier transform [KMA+15]. Their analysis explains why their method performs well, but it is not meant to be used for comparisons.

3.4 Noise power spectrum

Noise Power Spectra (NPSs) are retrieved by transforming the noise of a signal into frequency space and then squaring the magnitude. Noise usually means measurement error, but it is conceptually similar to our computation error.

Siewerdsen et al. proposed a framework for the analysis of n-dimensional NPSs [SCJ02], which is focused on the physics of medical imaging and does not treat light-transport algorithms at all. We believe that the challenges are quite different.

Hanson introduced a method for fast computation of approximate NPSs [Han98], similar in concept to the wavelet transform, but we do not have performance issues with what they call the gold standard of the Fourier transform.

3.5 Radial averages of the Fourier power spectrum

Several papers used radial averages of the Fourier power spectrum as an aid to assess error or the distribution of samples.

Similar to our proposition in Chapter 5, Lehtinen et al. [LKL+13] used radial averages (they call it the Ensemble Power Spectrum) to compare error over frequencies. Contrary to our proposal, they used equal-time renderings.

Lagae and Dutré analysed Poisson disk sampling algorithms using radial averages [LD08]. Subr et al. used radial averages of the amplitude, power and variance spectrum for investigation of sampling patterns [SK13][PSC+15].

CHAPTER 4

Statistical Properties of Rendering Error

In this chapter we formalise the anecdotal knowledge we mentioned in the introduction (Chapter 1). Some of the findings help to understand the descriptor (Chapter 5) and test results (Chapter 6). We start with an explanation why the Central Limit Theorem (CLT) is beneficial for the analysis of error and how we adapt existing algorithms to use those benefits. Furthermore, statistical properties of error are discussed, and finally we talk about the constant in the $\Theta(1/N)$ notation.

4.1 The Proxy algorithm

Algorithms that obey the CLT, have a straight-forward error distribution in the limit: The error at high sample counts ($N \to \infty$) tends towards a normal distribution with expectation 0 and variance $\mathrm{Var}(\mathcal{E})/N$ (Section 2.2). The sample variance (s^2) can be used to estimate the expected squared error at a certain sample count (N):

$$\mathrm{E}\left[\mathcal{E}_N^2\right] = \frac{\mathrm{Var}\left(\mathcal{E}\right)}{N} \approx \frac{s^2}{N}. \tag{4.1}$$

This is the same as saying that the error is $\Theta(1/N)$.

Sample variance is important, but incomplete in describing the error. The sample PDF and correlation between pixels also have a large impact on the performance (Section 4.2).

In MC algorithms it is obvious that CLT applies, and estimating the per-pixel variance and PDF would be straightforward. However, MCMC algorithms are different in nature. They have more than one parameter influencing the rendering time (at least seed pool size and chain length, Section 2.4), hence even formulating a convergence rule like $\Theta(1/N)$ is problematic. Moreover, the concept of a sample, and therefore sample variance and

PDF, is very different. In order to make the benefits of easy analysis available to MCMC, we propose to use a *proxy algorithm*: We configure any unbiased algorithm to produce $\mathcal{N}$ short renderings ($\vec{I}_i$) and compute the final solution ($\hat{\vec{I}}_{\mathcal{N}}$) as an average

$$\hat{\vec{I}}_{\mathcal{N}} = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \vec{I}_i. \tag{4.2}$$

$\mathcal{N}$ is the 'number of samples' in the proxy algorithm.

The proxy algorithm has the following properties:

- The short renderings ($\vec{I}_i$) are i.i.d. random variable vectors with finite variance, hence CLT applies for their mean ($\hat{\vec{I}}_{\mathcal{N}}$). Pixel values have the same limiting probability distribution in all algorithms, namely a normal distribution – only variance differs. The convergence speed to the normal distribution depends on the short rendering's probability distribution.
- Asymptotic convergence of error is guaranteed to be $\Theta(1/\mathcal{N})$.
- The per-pixel standard deviation or variance can be computed easily.

4.1.1 Changes in rendering algorithms

The solution of MC algorithms is an average of i.i.d. random variables, it is equivalent to the proxy solution.

However, in the majority of other algorithms, proxy and pure algorithms are not equivalent:

1. Many rendering systems use stratified sampling or low-discrepancy sequences in MC methods to improve the convergence rate for low dimensions of the light transport integral (Section 3.2). When using those sampling patterns, the random variables ($\vec{I}_i$) depend on i and the resulting algorithm can perform better than $\Theta(1/N)$.

   We propose to test this kind of situations by running the proxy algorithm several times with increasing parameter settings. For example, in case of sampling patterns, one could run the short-time rendering 1, 16 and 256 samples per pixel. This would increase the rendering time accordingly, but time scaling (Section 4.5) allows to compare the results. We performed a similar experiment for SPPM and show results in Section 6.5.

2. Metropolis Light Transport (MLT)

   - The original MLT algorithm used a single chain for all samples. In other words, the samples of MLT depend on each other and the CLT does not necessarily

apply. However, if mixing of the chain is strong enough, i.e., the dependence is not too big, a weaker version of the CLT would apply. The mixing coefficient depends on MLT parameters and the scene, and we are unaware of any work measuring the mixing or adjusting the parameters. In our experiments we assumed that two chain samples are virtually independent if there are enough samples in between to cover the whole image, and that in such a case the CLT also applies to the pure MLT algorithm.

- In pure MLT there are two parameters controlling the rendering time: luminance sample count and number of chain samples (chain length). The asymptotic convergence rate of relative intensity is $\Theta(1/chainlength)$ [APSS01], where the chain length is sometimes interpreted as N. However, it is not possible to reduce error by indiscriminately increasing the chain length while the luminance sample count stays the same. The proxy, on the other hand, increases both parameters with the same rate by design.
- In the rendering system we use, the default is to start many chains from a single seed pool for parallelism. This means that several chains can start from the same path, which can influence the performance. For the proxy however, the seed pool can not be shared between the different short renderings.

We investigate effects of chain length, chain count and luminance sample count in Section 6.4. In short: The number of luminance samples has a large influence (Section 6.4.1). Reducing the chain length below a certain threshold has considerable impact on some of the scenes (Section 6.4.3). Changing the seeding method has little effect (Section 6.4.2), which is good because it means that our measurement also applies to the default configuration of the rendering system we used.

3. ERPT has several parameters with complex interrelationships controlling rendering time and quality. There is no straight forward way to split up a rendering into a sum of shorter ones, nor there is a sample count that could be increased like N in other algorithms. Hence it is not even clear how CLT could apply, and the differences between pure and proxy algorithm are relatively large. We compare ERPT to MLT in Section 6.4.4.

4.2 Error in the spatial domain

We are going to use the proxy algorithm for the following analysis, which ensures that the per-pixel distribution of values converges to a normal distribution while increasing $\mathcal{N}$ (Section 4.1). The speed of convergence depends on the per-pixel PDF of the short renderings. We show several examples of such PDFs in this section.

Traditionally, many papers show equal-time renderings or rendering error for the surveyed algorithms. In our framework, this is comparable to running the proxy algorithm with a certain $\mathcal{N}$ for the respective algorithms and comparing the rendering result. However,

the data from the proxy algorithm allows to make a more detailed analysis compared to pure algorithms.

We demonstrate differences between the analysis using the rendering result (traditional), and more in-depth data from the proxy algorithm. Finally, some papers use relative and others absolute error (Section 3.1), and we compare those two as well.

4.2.1 Basics

No matter whether pure or proxy algorithm, or a single short render from the proxy, the outcome of a rendering algorithm ($\hat{\vec{I}}$) is a random variable vector. Its expectation is the unknown exact solution of the light-path integral (Section 2.1). We approximate the exact solution by a high-quality reference image ($\vec{R}$). A small error is made and neglected, as it is several orders of magnitudes below that of $\hat{\vec{I}}$. Hence, the (signed) absolute error $\hat{\vec{\mathcal{E}}}$ is computed by

$$\hat{\vec{\mathcal{E}}} = \hat{\vec{I}} - \vec{R}. \tag{4.3}$$

Since $\hat{\vec{\mathcal{E}}}$ is signed, it has a lower bound of $-\vec{R}$, and since we only investigate unbiased algorithms, the expected value is

$$\mathrm{E}\left[\hat{\vec{\mathcal{E}}}\right] = \vec{0}. \tag{4.4}$$

4.2.2 Per-pixel and per-sample error PDF ($\mathcal{N} = 1$)

We start our analysis by looking at the error of short-time renderings, i.e., the 'samples' of the proxy algorithm. Since we cannot compute the PDF analytically, we look at

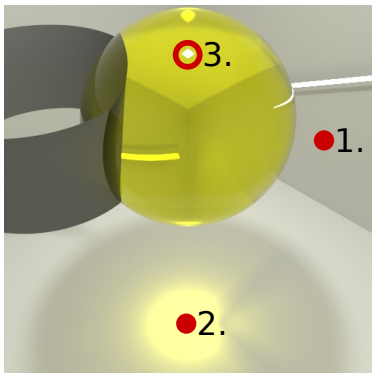
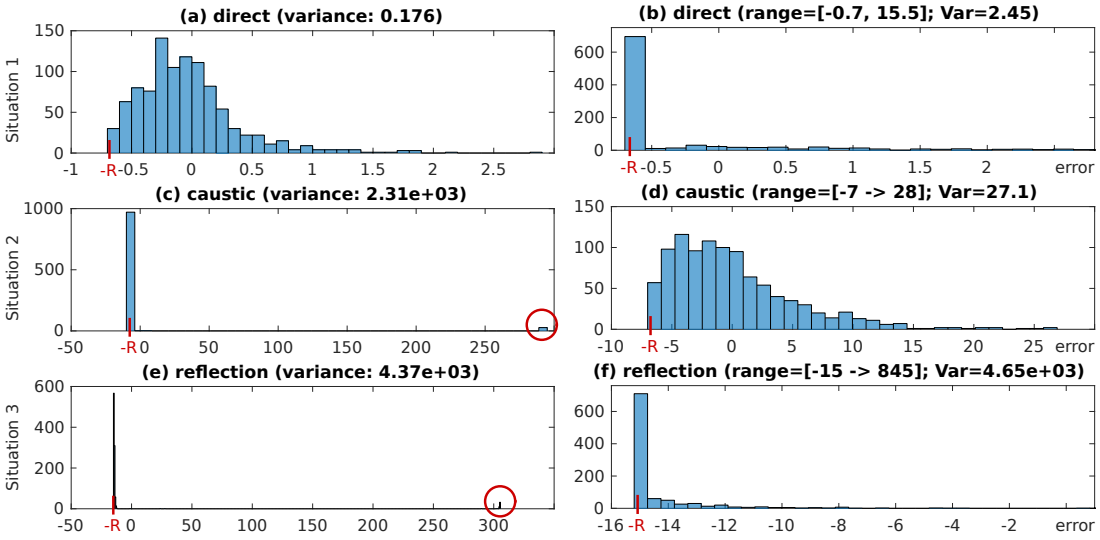


Figure 4.1: Pixel positions for the histograms in Figure 4.2. Situation 1 is direct and indirect light, 2 is a caustic plus indirect light and 3 is a refraction plus a reflection of an area light.



PT | *MLT, (large values clipped because bars were invisible)*

Figure 4.2: Pixel error histograms for three situations (Figure 4.1) of 1000 renderings with one sample per pixel. PT: Direct *is the easiest, the histogram resembles a Gaussian. In* caustic/reflection*, most observations are of indirect light/the refraction (large peaks), only a few sample the caustic/reflection (marked peaks). MLT: The* caustic *can be sampled efficiently, resulting in a Gaussian shape. In* direct *and* reflection *the chain often did not reach the pixels, resulting in a black colour ($\mathcal{E} = -R$). In order to compensate, some of the observations have a large value – the PDF has a long tail (clipped due to invisibility of the bars). In* reflection *the chain sometimes reached the pixel, but the refraction was sampled and the pixel was too dim (bins below 0, but not the peak).*

absolute error histograms of a large number of one-sample-per-pixel renderings instead. We choose a pixel for three representative lighting situations (Figure 4.1). The first is a point on the wall with mostly direct lighting, the second from a caustic and the last from the specular reflection of a light source on a glass sphere. The histograms were computed for MLT and PT algorithms (Figure 4.2). Since the histogram was made for the error and not value, zero marks the expected value and $-R$ the lower bound.

When error is distributed closely to a normal distribution (PT *direct* and MLT *caustic*, Figures 4.2a and d), the mean quickly converges to a perfect normal distribution.

In some lighting situations, the distribution can be multi-modal. PT shows two peaks for *caustic* and *reflection* (Figures 4.2c and e). In *caustic* one peak is normal indirect lighting and the other the hard-to-sample caustic. In *reflection*, one of them is the light-source reflection and the other the refraction into the material. The higher peaks are more probable when sampling locally, therefore most samples are lost there. Convergence

depends on the distance between and relative height of the peaks.

MLT *direct* light and *reflection* contain a peak at the lower bound, which means that many $\vec{I}_i$ stayed black (Figures 4.2b and f). The reason is that the sample count was too low to cover the dim areas with enough samples. This is inherent to the MLT algorithm, as the number of samples is directly proportional to pixel brightness. However, this is not problematic if the variance remains low (Figure 4.2b).

Reflection is a situation where the Markov chain has difficulties to find a lighting effect (Figure 4.2f). Once the reflection was found, the chain stays there for very long, visible by the range of up to 845. This is an example of a long PDF tail and quite problematic for convergence to normal distribution [JMD15].

4.2.3 Increasing the rendering time (increasing $\mathcal{N}$)

In Section 4.2.2 we were looking at the error PDF of particular pixels with $\mathcal{N} = 1$. The logical next step is to look at what happens to the error PDFs when increasing $\mathcal{N}$. That is, we run the proxy many times with $\mathcal{N} = 1$, $\mathcal{N} = 64$ and $\mathcal{N} = 4096$ each, take a representative pixel of the results, and look at the differences in the three histograms.

Many papers analyse algorithms by looking at renderings with a long rendering time or the corresponding error. In our framework, long-time renderings of pure algorithms are comparable with the final result of the proxy algorithm with a large $\mathcal{N}$ (argument and caveats in Section 4.1), e.g., of $\mathcal{N} = 4096$. We use that similarity to make an argument that long-time renderings are a compromise between showing the amount of error (standard deviation or variance) and error behaviour (sample PDF and correlation), both of which can be shown more precisely separately.

When looking at the absolute error ($\hat{\vec{\mathcal{E}}}_\mathcal{N}$) of proxy renderings with small $\mathcal{N}$, it is largely influenced by the per-pixel error PDF specific to the algorithm (Section 4.2.2). For instance, the lower peak of a bi-polar distribution can show up as *pixel outliers*. When $\mathcal{N}$ is increased, the error distribution becomes more and more normal due to CLT, where the expectation is zero and the per-pixel variance depends on the per-pixel error PDF. The rendering looks more and more like a draw from a collection of normal distributions with algorithm-specific standard deviations. This is demonstrated in Figure 4.3 (a and b), showing the caustic in the parametric Box scene (location 2 in Figure 4.1).

Likewise, the distribution of absolute error magnitudes ($|\hat{\vec{\mathcal{E}}}_\mathcal{N}|$), used by some papers for error visualisation (Chapter 3) approaches a half-normal distribution (Section 2.2.1). The only parameter of the half normal distribution is the per-pixel standard deviation of $\mathcal{N} = 1$, and its expectation is the standard deviation times a constant:

$$\mathrm{E}\left[\lim_{\mathcal{N}\to\infty}\left|\mathcal{N}\times\hat{\vec{\mathcal{E}}}_\mathcal{N}\right|\right] = \sqrt{2/\pi}\times\mathrm{Stddev}(\vec{\mathcal{E}}). \tag{4.5}$$

Convergence – from a distribution specific to the algorithm towards a half-normal distribution – is visible in Figure 4.3 (c and d). However, even with $\mathcal{N} = 4096$ the PDF

is not completely Gaussian, therefore (c) appears brighter than (d). In other words, absolute error magnitudes approach the scaled standard deviation with added noise.

We summarise: The behavioural difference of algorithms is visible in short renderings while the amount of error can be visualised by standard deviation images. Instead of mixing behaviour and error expectation in a single long-time rendering, we propose to look at them separately by looking at particular short time renderings and standard deviation per pixel. Neither per-pixel standard deviation ($\sqrt{\mathrm{E}[\vec{\mathcal{E}}^2]}$), variance ($\mathrm{E}[\vec{\mathcal{E}}^2]$) nor short-time renderings are routinely shown in research papers (Chapter 3).

This separation is demonstrated in the Torus scene (Figure 4.4). With small $\mathcal{N}$, most of the PT samples miss the caustic and underestimate the value. Occasionally there are

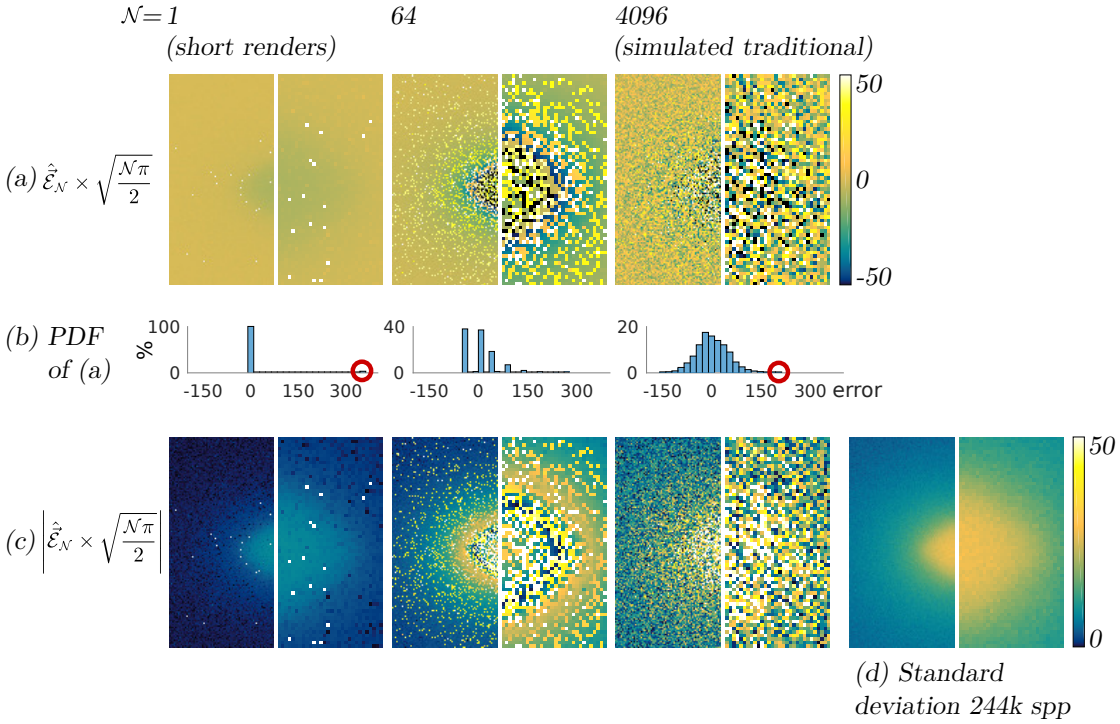
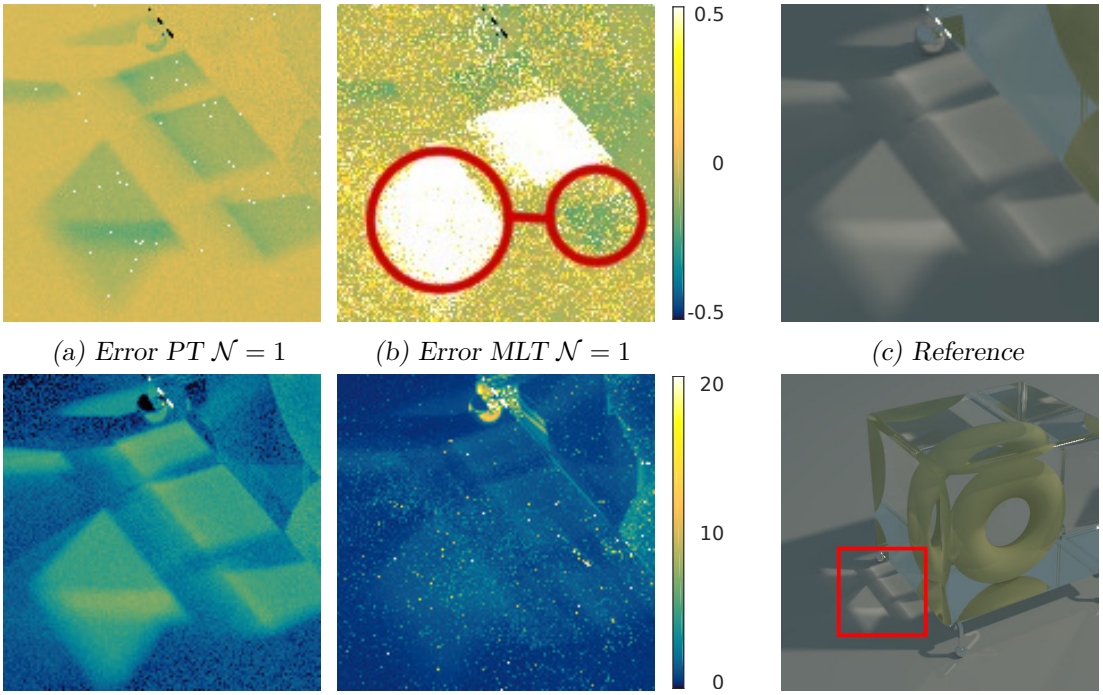


Figure 4.3: Error behaviour while increasing sample counts for a caustic (PT, location 2 in Figure 4.1). (a) and (c) show rendering error ($\hat{\vec{\mathcal{E}}} = \hat{\vec{I}} - \vec{R}$.), respectively rendering error magnitudes ($|\hat{\vec{\mathcal{E}}}|$). The error is scaled to matching ranges and a close-up is shown on the right side. The histograms (b) are from the centre of the caustic and were made from the results of 1000 proxy renderings with $\mathcal{N}$ short renderings each.. The bi-modal distribution at $\mathcal{N} = 1$ becomes more and more Gaussian as $\mathcal{N}$ is increased. Small $\mathcal{N}$ show the error distribution of the algorithm while larger $\mathcal{N}$ approach standard deviation times Gaussian noise (a). Scaled absolute error (c) becomes more and more similar to the standard deviation (d). However, noise remains even when further increasing $\mathcal{N}$.



(a) Error PT $\mathcal{N} = 1$ (b) Error MLT $\mathcal{N} = 1$ (c) Reference

(d) Standard deviation PT (e) Standard deviation MLT (f) Torus scene

Figure 4.4: Behaviour vs. error expectation for a caustic on the floor (c) of the Torus scene (f): PT underestimates with most samples and compensates with outliers (a), resulting in a much higher standard deviation (d). (b) on the other hand shows that the caustics appear as a whole or not at all, which is typical for MLT. However, this fact alone is not problematic as standard deviation is low (e). This demonstrates that (a) and (b) show behavioural differences in the algorithms and (d) and (e) the amount of error to be expected.

outliers for compensation (Figure 4.4a). In MLT, correlation between pixels is visible as the caustic patches pop up as a whole or not at all (Figure 4.4b). Although not fully converged, the standard deviation pictures (Figures 4.4d and e) show a larger value for PT's pixel outliers compared to MLT's correlated "pop-ups". This means, that PT is inferior to MLT in this instance.

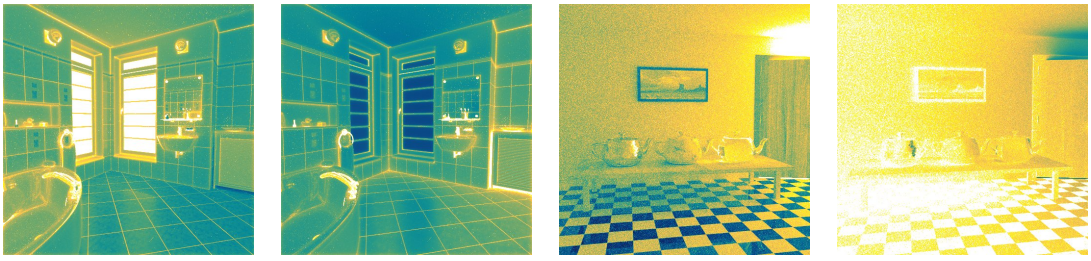
4.2.4 Absolute vs. relative error

Some papers use a relative error metric based on

$$\vec{\mathcal{E}}_r = \vec{\mathcal{E}}/(\vec{R} + \epsilon),$$

where ϵ is a small constant (see Section 2.7 for the metrics and 3.1 for the papers).

This might seem intuitive to avoid over-valuation of error in bright areas. Look for instance at Figure 4.5a and 4.5b, where most of the light comes through diffuse windows.



(a) absolute (b) relative (c) absolute (d) relative

Figure 4.5: The pictures show standard deviation of absolute ($\vec{\mathcal{E}}$) and relative error ($\vec{\mathcal{E}}/(\vec{R}+\epsilon)$). Absolute and relative error can both lead to excessive error values. In (a) that is the case in the window, while relative error (b) shows no such artefact. The contrary is the case in (c) and (d), where relative error is excessive in the shadow.

The presented algorithm has large absolute error in those areas. However, this is of little practical importance as the amplitude is large and the error will be compressed by tone mapping or clamped altogether.

Relative error, on the other hand, has the drawback of exaggerating error in dim areas of the image. Figure 4.5d shows excessive relative error on dark tiles and in shadows. In most areas of the shadow in Figure 4.5d, the ϵ does not take effect, even when relatively large at $\epsilon = 0.01$ (0.001 was used elsewhere [KMA+15]).

4.3 Error in the Fourier domain

In addition to the spatial analysis in the previous section, we also perform a frequency analysis of the error. This is interesting because it is known that algorithms differ in the spectral distribution of error.

We use the Fourier transform for analysis, but other methods exist (Section 2.6). Several questions come to mind, which we try to answer in this section:

- What is the probability distribution of error in Fourier frequencies?
- Does CLT also apply to the Fourier domain and if, in what way?
- How do outliers (single- and multi-pixel ones) transform?
- There can be correlation between pixels in the spatial domain, what is the effect in the frequency domain?

We will use $\mathcal{F}\hat{\vec{\mathcal{E}}}$ to denote the complex Fourier spectrum of an error image $\hat{\vec{\mathcal{E}}}$. The magnitude spectrum is $|\mathcal{F}\hat{\vec{\mathcal{E}}}|$ and the error power spectrum $|\mathcal{F}\hat{\vec{\mathcal{E}}}|^2$.

4.3.1 Distribution of a particular frequency and CLT

Real and imaginary parts of every Fourier frequency are a weighted sum of pixel errors and therefore random variables themselves:

$$X = \sum_{i=1}^{M} w_i \hat{\vec{\mathcal{E}}}[i],$$

where X stands for the real or imaginary part of any frequency and w_i are the corresponding weights. Its expectation is 0 because the expectation operator is linear and expected pixel error is 0:

$$\begin{aligned} \mathrm{E}\left[X\right] = \mathrm{E}\left[\sum_{i=1}^{M} w_i \hat{\vec{\mathcal{E}}}[i]\right] = \sum_{i=1}^{M} w_i \,\mathrm{E}\left[\hat{\vec{\mathcal{E}}}[i]\right] = \sum_{i=1}^{M} w_i 0 = 0, \\ \Rightarrow \mathrm{E}\left[\mathrm{Re}(\mathcal{F}\hat{\vec{\mathcal{E}}})\right] = \vec{0}, \\ \text{and } \mathrm{E}\left[\mathrm{Im}(\mathcal{F}\hat{\vec{\mathcal{E}}})\right] = \vec{0}. \end{aligned}$$

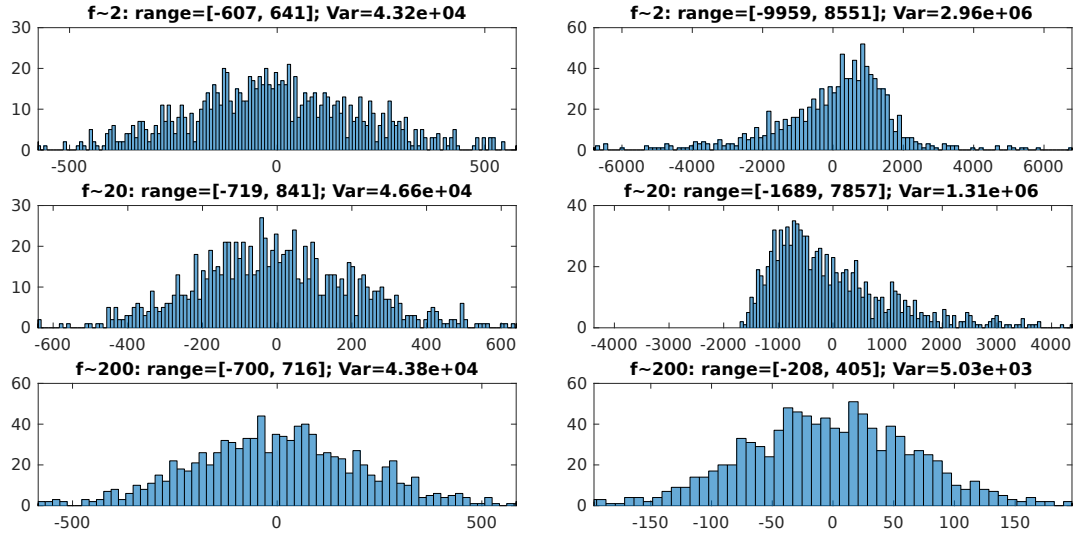
Figure 4.6 shows histograms of the real part of Fourier frequencies. They look quite Gaussian for PT and higher frequencies in MLT. The reason is that the Fourier frequencies are linear combinations of many random variables (the pixels). The effect of this linear combination is similar to CLT, i.e., the resulting distributions are more Gaussian, especially if the terms are uncorrelated.

Since the Fourier transform is linear, CLT holds in our proxy algorithm for real and imaginary part of the Fourier spectrum as $\mathcal{N}$ goes to infinity:

$$\mathcal{F}\hat{\vec{I}}_{\mathcal{N}} = \mathcal{F}\left(\frac{1}{\mathcal{N}}\sum_{i=1}^{\mathcal{N}} \vec{I}_i\right) = \frac{1}{\mathcal{N}}\sum_{i=1}^{\mathcal{N}} \mathcal{F}\vec{I}_i.$$

The Fourier spectrum of $\hat{\vec{I}}_{\mathcal{N}}$ is the mean of the Fourier spectra of $\vec{I}_i$. Since all $\vec{I}_i$ are equally distributed, its Fourier transforms are as well. The expectation is well defined ($\vec{0}$) and the variance is finite if the same is true for $\vec{I}$ (because elements in $\mathcal{F}\vec{I}$ are just linear combinations of $\vec{I}$). Hence CLT applies.

We summarise: Real and imaginary part of the Fourier spectrum tend towards a zero-centred normal distribution as the sample count $\mathcal{N}$ goes to infinity. The variance of real and imaginary part is $\Theta(1/\mathcal{N})$. The properties of the Fourier spectrum of the proxy algorithm are similar to the spatial domain. Hence, we deem it suitable for analysing performance.



(a) Path tracing *(b) MLT*

Figure 4.6: The figures show data of 1000 short renderings per algorithm. The absolute error was transformed into the Fourier domain and 3 random Fourier frequencies were selected on a circle around the DC term with distance f. The histograms were made with the real part of the complex number. More data in Appendix C.2.

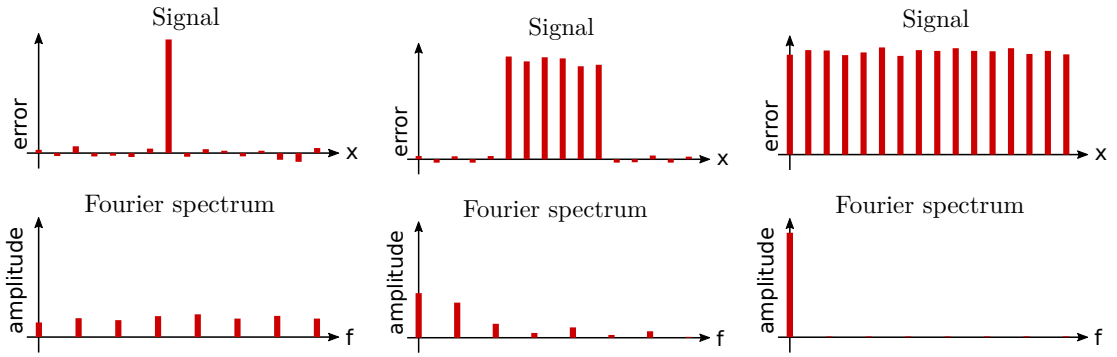
4.3.2 Outliers and Correlation

In order to understand the effect of outliers on our descriptor (Chapter 5), we need to know how they are transformed into Fourier space. It is possible to look at outliers separately, because the Fourier transform is linear. We use a one-dimensional discrete function for demonstration, but the mechanisms are equal for 2D. Outliers in the spatial domain are areas with a very large value. One extreme case of an outlier is shrinking the area to an isolated pixel, the other is to make it larger until covering the whole image. Their correspondents in 1D are a single spike (Figure 4.7a), a segment (Figure 4.7b) or the whole domain (Figure 4.7c).

The discussion is based on seeing the Fourier transform as many dot products (Figure 4.8). Every element in the Fourier spectrum is the sum of the original signal multiplied with a sine/cosine wave, whose frequency is the position of the element in the Fourier spectrum.

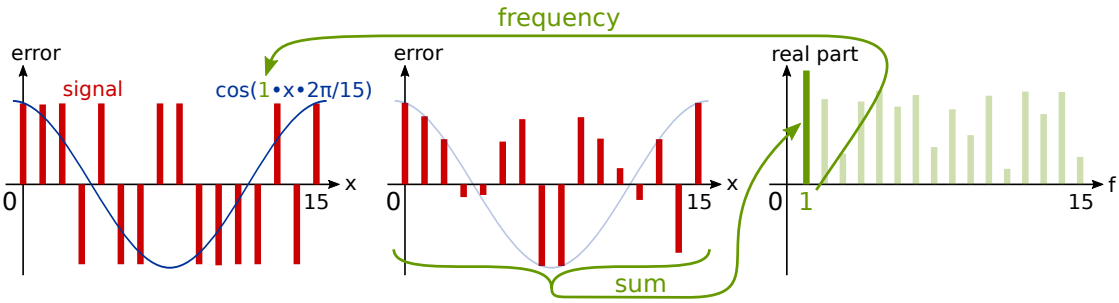
In Figure 4.7, we can observe that the size of the outlier area is roughly inversely proportional to the largest affected frequency. In other words, frequencies below that largest frequency are correlated. A single pixel outlier affects the whole spectrum, a small area the lower frequencies and the whole area only the 0th frequency (the DC term). This can be explained by cancellation (Figure 4.9). The waves have positive and negative sides. If the outlier covers both, then some of the error is cancelled out and does not appear at that frequency in the spectrum. If the error covers one wavelength, it is completely cancelled. Otherwise, the outlier energy is split between the real (cosine) and imaginary (sine) part of the spectrum. The ratio depends on the position of the outlier and the values of cosine and sine at that position (see also appendix Section C.1).

A similar reasoning of cancellation can be made for correlation of pixels in the spatial domain (instead of outliers). Uncorrelated algorithms (MC) have flat spectra while correlated ones (MCMC) have a peak in low frequencies. The width of the peak is inversely proportional to the correlation radius.



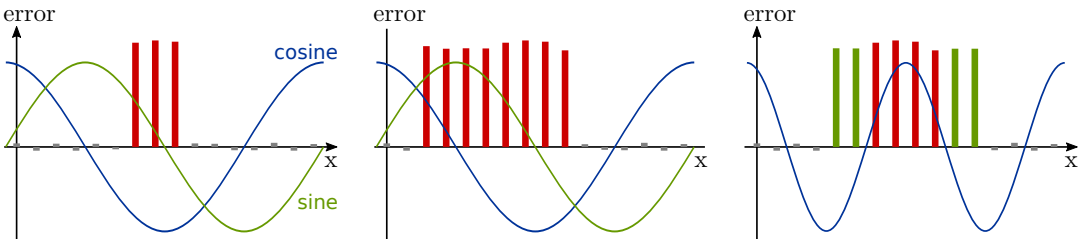
(a) Isolated single pixel (b) Area (c) Whole image

Figure 4.7: For the discussion, we use one dimensional analogies of 2D error images. The graphs show one half of the symmetric Fourier spectrum amplitudes. The largest affected frequencies are roughly inversely proportional to the outlier's size in the spatial domain.



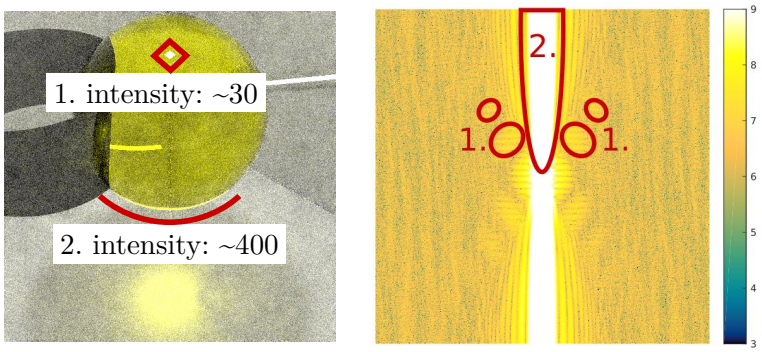
(a) Signal and cosine (b) Signal multiplied with cosine (c) Discrete Fourier spectrum

Figure 4.8: The value of a Fourier frequency (c) is the sum of values in (b). (b) is the result of multiplying the original signal with a sine or cosine in the spatial domain (a), whose frequency is specified by the position in Fourier spectrum (c).



(a) Outlier area is small, (b) half a wavelength, (c) and a whole wavelength.

Figure 4.9: The largest affected frequency is roughly proportional to the spatial outlier extent because of cancellation. If the outlier area is small relative to the wavelength (a), then the contribution to either sine or cosine part is substantial (little cancellation). With larger areas, a part of the outlier is multiplied with small or negative factors and the contribution is reduced (b). At one wavelength, the negative factors of the waves cancel the positive ones and there is no contribution (c, full cancellation).

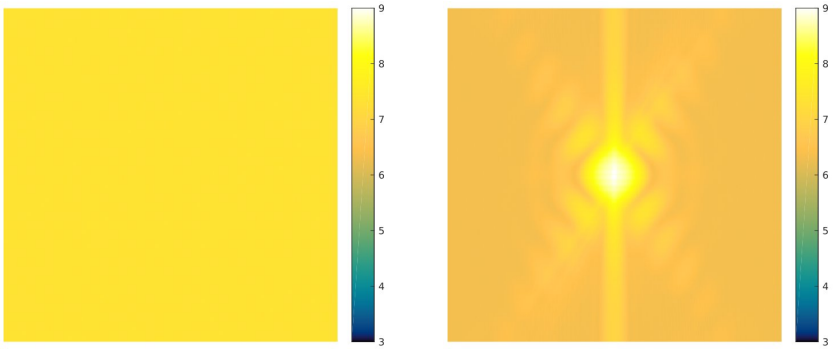


(a) Image ($\vec{I}_i$)

(b) Power spectrum ($\vec{\mathcal{E}}_{ps}$)

Figure 4.10: Example of an area outlier (100 samples per pixel on average, MLT) and its power spectrum (log10). The outliers (1. and 2.) in the spatial domain (a) are visible in the power spectrum (b, only the top half of the rotational-symmetric spectrum is annotated). The marked bright spots in (b) match the direction of edges in (a). As predicted, a large area affects smaller frequencies (1.), while a small area affects all (2., vertical direction).

Let us look at a 2D error outlier spectrum of an actual rendering algorithm (Figure 4.10). There are two outlier areas in the image (annotated in a), which are also visible in the spectrum (annotated in b). As predicted, a large area affects smaller frequencies (1. annotation), while a small area affects all (2. annotation, vertical direction). Single pixel outliers would appear as an added constant to all Fourier frequencies of the amplitude or power spectrum. Whole image outliers, e.g. wrong brightness scaling due to a bad MLT weight $\hat{W}$, show up in the DC term.



(a) PT

(b) MLT

Figure 4.11: Typical mean power spectra for MC and MCMC methods (log10, 100 samples per pixel). MC methods have power spectra that resemble white noise (a), while MCMC methods typically include a peak in low frequencies (b). It is often possible to see the edge direction of hard to sample lighting effects.

Figure 4.11 shows power-spectrum expectations ($\mathrm{E}\left[|\mathcal{F}\vec{\mathcal{E}}|^2\right]$) for MC and MCMC algorithms. For MC, it is a constant. MCMC, on the other hand, has a large peak in low frequencies (centre) and smaller peaks around it. These peaks come from structured error in the spatial domain, caused by the correlation between samples in MCMC methods. It is quite common to see the direction of edges from certain hard-to-find lighting effects in MCMC power spectra.

As a summary: Outliers with a small area (single-pixel outlier) influence all frequencies equally. When outliers get larger, the influence concentrates more and more on smaller frequencies.

The discrete Fourier transform is injective and linear, hence we can also go the other way round: Larger amplitudes in all frequencies of a single short-time rendering, compared to the proxy average, indicate a small area outlier. Larger amplitudes in low frequencies, on the other hand, signal the existence of a larger area outlier.

4.4 Variance and Kurtosis in the Fourier domain

In the spatial domain, per-pixel variance and standard deviation can be used to measure error. They are $\Theta(1/\mathcal{N})$ and $\Theta(1/\sqrt{\mathcal{N}})$, respectively. We show that the power spectrum and its square root are their Fourier-domain equivalents.

The variance and standard deviation of real and imaginary part are $\Theta(1/\mathcal{N})$, respectively $\Theta(1/\sqrt{\mathcal{N}})$ (Section 4.3.1). However, each of them contains roughly only half of the error.

Since $E\left[\mathcal{F}\vec{\mathcal{E}}\right] = \vec{0}$ the variance is simply calculated as

$$\vec{\mathcal{V}}_{Re} = \mathrm{Var}\left(\mathrm{Re}(\mathcal{F}\hat{\vec{\mathcal{E}}})\right) = \mathrm{E}\left[\mathrm{Re}(\mathcal{F}\hat{\vec{\mathcal{E}}})^2\right],$$

and analogous for the imaginary part. Taking the expected value is linear even for correlated random variables (Section 2.2), therefore the expectation of the error power spectrum

$$\mathrm{E}\left[\left|\mathcal{F}\hat{\vec{\mathcal{E}}}\right|^2\right] = \mathrm{E}\left[\mathrm{Re}(\mathcal{F}\hat{\vec{\mathcal{E}}})^2 + \mathrm{Re}(\mathcal{F}\hat{\vec{\mathcal{E}}})^2\right] = \mathrm{E}\left[\mathrm{Re}(\mathcal{F}\hat{\vec{\mathcal{E}}})^2\right] + \mathrm{E}\left[\mathrm{Re}(\mathcal{F}\hat{\vec{\mathcal{E}}})^2\right] = \vec{\mathcal{V}}_{Re} + \vec{\mathcal{V}}_{Im}$$

converges with $\Theta(1/\mathcal{N})$ to zero. The same is true for radial averages of the power spectrum as they are a linear combination. The respective square roots are $\Theta(1/\sqrt{\mathcal{N}})$, like the standard deviation.

It could be useful to measure the tail of the error distribution, or, in other words, the tendency towards outliers. Kurtosis (Section 2.2) is a tool for that [Wes14]. Using a similar analogy and derivation as for variance, we get

$$\hat{\vec{\mathcal{E}}}_{ps} = \left|\mathcal{F}\hat{\vec{\mathcal{E}}}\right|^2$$

$$\mathrm{Kurt}\left[\mathcal{F}\hat{\vec{\mathcal{E}}}\right] = \frac{\mathrm{E}\left[(\hat{\vec{\mathcal{E}}}_{ps})^2\right]}{\left(\mathrm{E}\left[\hat{\vec{\mathcal{E}}}_{ps}\right]\right)^2}.$$

Unfortunately, it is so sensitive to the outliers – that it "can merely tell whether they are present or not" [Liv07]. Experience and literature [KW04] have shown – that the variation between several random experiments can be so big – that the calculated number gives little additional knowledge.

4.5 Constant of convergence

When quantitatively comparing light-transport algorithms, error measures like MSE or RMSE of equal-time renderings are often used (Section 3.1). The result of those error measures is a random variable, and in scenes with complex light transport its variance can be significant. We therefore propose to estimate expectations of the respective error measures and compare those instead of observations. The standard deviation or variance of the measure can be further used to characterise the algorithm.

We use a toy example for illustrating the problem of comparing equal-time error and our improvement (Figure 4.12). Two MC algorithms (Section 2.3) estimating

$$f(x) = \int_0^1 \frac{1}{0.0001 + x}\,\mathrm{d}x$$

are compared. The first takes N samples according to a uniform PDF, while the other performs importance sampling (PDF shape of $\diagdown$ with $\mathrm{pdf}(0) = 2$ and $\mathrm{pdf}(1) = 0$). The plots show various methods of measuring the squared error ($g(N)$) over processing time (N) in logarithmic scale.

Comparing equal-time MSE error corresponds to running the two algorithms once for a fixed N, compute the squared error and compare. Figure 4.12 (a) shows squared error for multiple N, demonstrating that a comparison would be unreliable due to fluctuation no matter the N. The graph was made using

$$g_a(N) = \hat{\mathcal{E}}_N^2 = \left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{E}_i\right)^2.$$

Random fluctuation can be reduced by averaging several runs (10) as seen in Figure 4.12 (b), but this is expensive in terms of computation time:

$$g_b(N) = \frac{1}{10}\sum_{j=1}^{10}\hat{\mathcal{E}}_N^2 = \frac{1}{10}\sum_{j=1}^{10}\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{E}_{j,i}\right)^2.$$

We can do better by comparing expected square error ($\mathrm{E}[\hat{\mathcal{E}}_N^2]$), which can be computed from sample variance ($\mathrm{Var}(\mathcal{E})$):

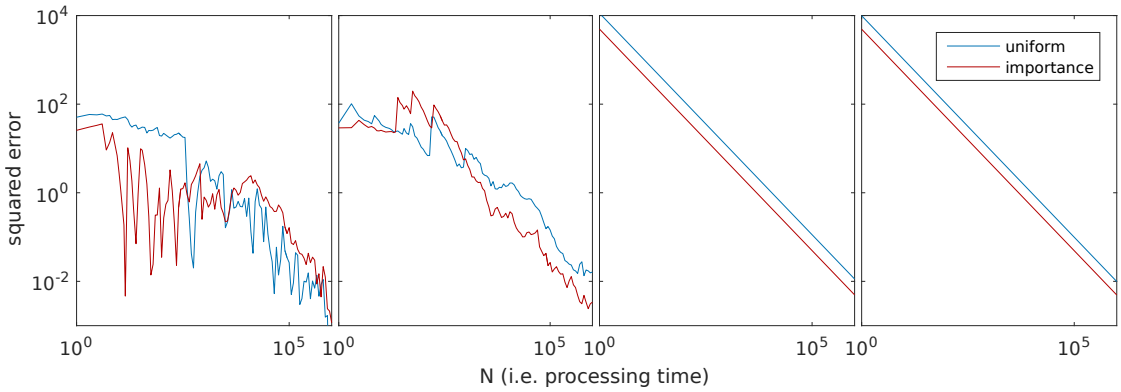
Since CLT applies and $\mathrm{E}[\mathcal{E}] = 0$:

$$\mathrm{E}\left[\hat{\mathcal{E}}_N^2\right] = \mathrm{Var}\left(\hat{\mathcal{E}}_N\right) = \frac{\mathrm{Var}(\mathcal{E})}{N}, \tag{4.6}$$

where $\mathrm{Var}(\mathcal{E})$ is the *constant of convergence*. This is a more precise version of the statement that the squared error of an MC algorithms is $\Theta(1/N)$ (Section 3.2).

Estimating the constant of convergence (variance) directly gives a much more reliable comparison. Equation 4.6 was used to produce Figures 4.12 (c) and (d):

$$g_c(N) = \frac{1}{N}\frac{1}{10^5}\sum_{i=1}^{10^5}\left|\hat{\mathcal{E}}_i\right|^2$$
$$g_d(N) = \frac{1}{N}\mathrm{Var}(\mathcal{E}),$$



(a) one process *(b) average of 10* *(c) estimated expectation (10^5 samples)* *(d) closed form expectation*

Figure 4.12: Toy experiment for measuring the error of two MC algorithms (see text): The plots show squared error over processing time in logarithmic scale. Hence, the expected slope is the asymptotic convergence exponent (-1), while the vertical position depends on the convergence constant. Many papers report equal-time rendering error, which corresponds to picking a certain N in (a) and reading the values of the algorithms – the information would be unreliable. Repeating the experiment and averaging the error (i.e., estimating expectation) improves reliability (b), but this is expensive in terms of computation. Since the asymptotic convergence rate is always the same ($\Theta(1/N)$), it is enough to estimate the convergence constant directly by computing the sample variance. (c) shows the resulting estimate of the expectation of squared error. It is close to the analytically computed expectation (d) while using a smaller sample budget than (a).

where $\mathrm{Var}(\mathcal{E})$ was analytically computed. The total number of samples in Figure 4.12 (c) is one order of magnitude smaller than in the first version, yet the estimation of expected error is close to the analytically computed truth in Figure 4.12 (d).

4.5.1 Application in rendering

In rendering, every pixel is a separate random variable. Instead of squared error, MSE is often used for measurement:

$$\mathrm{MSE} = \frac{1}{M}\sum_{m=1}^{M}(\hat{\vec{I}}[m] - \vec{R}[m])^2 = \frac{1}{M}\sum_{i=1}^{M}(\hat{\vec{\mathcal{E}}}[i])^2,$$

where M is the pixel count, R the reference and I the rendering.

This averaging process reduces the variance of MSE compared to the toy example. We can see that by looking at its formula (Equations 2.6, 2.7 and the definition of MSE were used):

$$\mathrm{Var}\,(MSE) = \mathrm{Var}\left(\frac{1}{M}\sum_{i=1}^{M}(\hat{\vec{\mathcal{E}}}[i])^2\right) = \frac{1}{M^2}\,\mathrm{Var}\left(\sum_{i=1}^{M}(\hat{\vec{\mathcal{E}}}[i])^2\right) \tag{4.7}$$

$$= \frac{1}{M^2}\left(\sum_{i=1}^{M}\mathrm{Var}\left((\hat{\vec{\mathcal{E}}}[i])^2\right) + \sum_{i\neq j}^{M}\mathrm{Cov}\left((\hat{\vec{\mathcal{E}}}[i])^2, (\hat{\vec{\mathcal{E}}}[j])^2\right)\right), \tag{4.8}$$

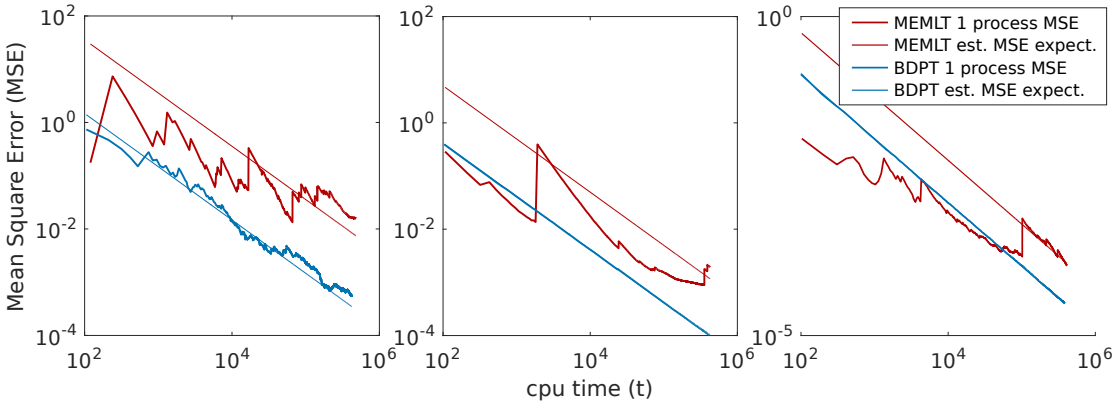
where M is the number of pixels and $\hat{\vec{\mathcal{E}}}$ the error vector. Per-pixel variance ($\mathrm{Var}(\hat{\vec{\mathcal{E}}}[i])$) can be easily estimated, but inter-pixel covariance ($\mathrm{Cov}(\hat{\vec{\mathcal{E}}}[i], \hat{\vec{\mathcal{E}}}[j])$) cannot. However, for algorithms with independent pixels (MC), the covariances are zero. Hence, MSE's variance is $1/M^{\mathrm{th}}$ of the average squared-pixel-error variance. In case of correlated pixels (MCMC), MSE's variance increases with rising inter-pixel dependence.

We conclude that MSE measurements of MC algorithms are closer to its expectation with a high probability because its variance is lower (all other parameters being equal). In other words, MSE measurements for MC algorithms are more reliable compared to MCMC algorithms. This was tested in an experiment (Figure 4.13). It shows that the correlated algorithm (MEMLT) has more MSE variance in all three examples, but also MC algorithms can show a significant amount (BDPT in Figure 4.13a). More examples are presented in Appendix A.

Like in the toy example, we can estimate the constant of convergence instead of using one observation for error estimation. This is easy when using the proxy algorithm:

Equation 4.6 applies to every particular pixel. MSE is a linear combination of squared pixel errors, hence the average of pixel variances is the constant of convergence with regard to $\mathcal{N}$.

It is possible to accommodate for algorithms with different sample costs. We can *scale* the MSE expectation to a different rendering time:



(a) Bottle *(b) Kitchen* *(c) Torus*

Figure 4.13: Examples of MSE over time of a single MLT and BDPT process plus the estimated expectation (average of all per pixel variances divided by t). Although the error measure is smoother due to a large pixel count, comparisons of equal time MSE are still unreliable in some cases. In particular inter-pixel correlation in MLT causes large MSE variance, but also MC algorithms can have significant variance (BDPT in a). Comparing estimated expectations of MSE is more reliable.

$$\mathrm{E}\left[MSE_{t'}\right] \times t' = \mathrm{E}\left[MSE_{t}\right] \times t, \tag{4.9}$$

$$\mathrm{E}\left[MSE_{t'}\right] = \frac{\mathrm{E}\left[MSE_{t}\right] \times t}{t'}, \tag{4.10}$$

where $\mathrm{E}\left[MSE_{t}\right]$ is the MSE expectation of our short renderings (after t seconds, which we have estimated) and $\mathrm{E}\left[MSE_{t'}\right]$ is the MSE expectation for an arbitrary t' (we use $t' = 1$ second).

$\mathrm{E}\left[MSE_{t'=1}\right]$ is the constant of convergence with regard to time. Algorithms can be directly compared using $\mathrm{E}\left[MSE_{t'=1}\right]$. Since, as a result of the proxy algorithm, all algorithms have the same asymptotic convergence, we can go a bit further: Instead of stating that an algorithm is $\Theta(1/t)$ (replacing the sample count N, or $\mathcal{N}$ with the time t), we can be more precise and say:

$$\mathrm{E}\left[MSE_{t}\right] = \frac{\mathrm{E}\left[MSE_{t=1}\right]}{t}. \tag{4.11}$$

Figure 4.13 also shows estimated MSE expectations using Equation 4.11.

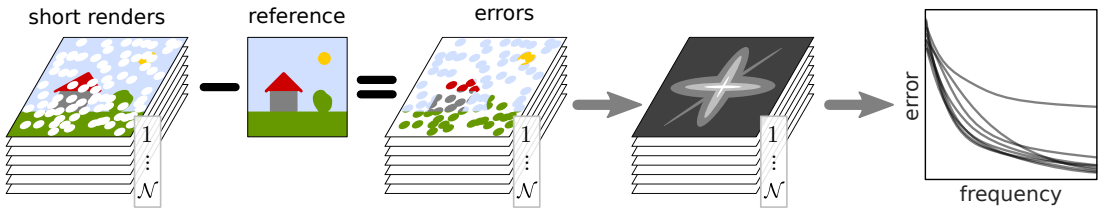
CHAPTER 5

Error Spectrum Ensemble: A New Tool for Measuring Convergence

In this chapter we present the Error Spectrum Ensemble (ESE), a descriptor for the convergence behaviour of unbiased light-transport algorithms. It visualises error and the tendency towards outliers in different frequencies.

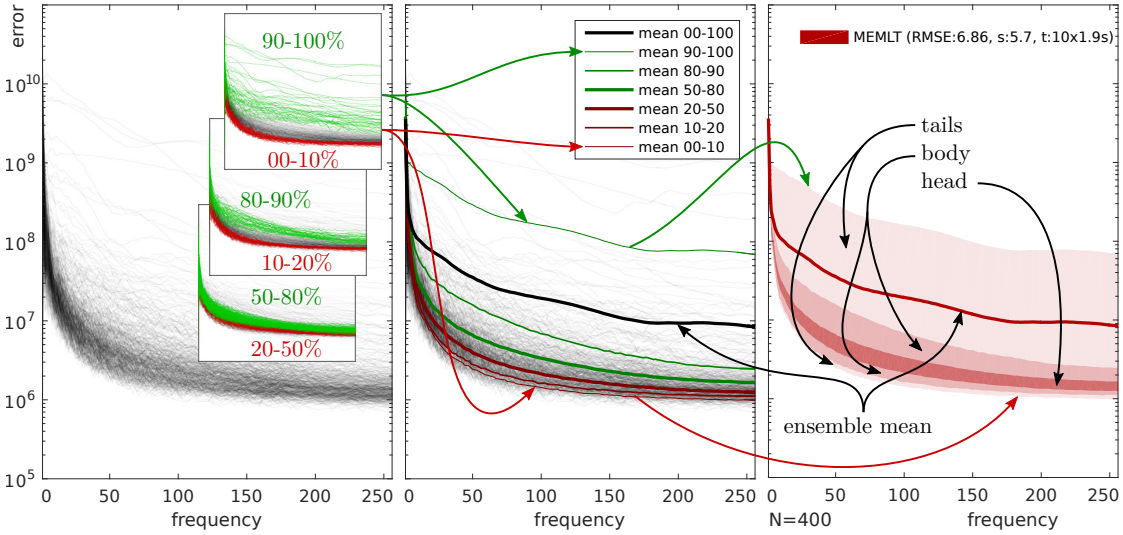
5.1 Overview

As motivated in Section 4.5, we generate a large number ($\mathcal{N}$) of short renderings (few samples per pixel). Using a high-quality reference image, we get $\mathcal{N}$ signed error images



(a) calculate error images *(b) stack of error power spectra* *(c) $\mathcal{N}$ radial averages (+$\mathcal{N}$ MSE values)*

Figure 5.1: We start by generating a large number ($\mathcal{N}$) of short renderings and computing the error using the corresponding reference (a). Then Fourier power spectra of the error images are computed (b). For each rendering, the MSE and the power spectrum radial averages are stored (c).



(a) 400 radial averages *(b) Area borders and ensemble mean* *(c) Error Spectrum Ensemble*

Figure 5.2: The power spectrum radial averages (grey curves in (a)) are used to compute the ESE descriptor (c): The data is used twice, computing the ensemble mean and the 'regions' (head, body and tail). The ensemble mean is the per frequency average of all radial averages. In order to compute the regions, the radial averages are sorted into 6 buckets using MSE (a). For each bucket, the per frequency mean is computed in a similar process as for the ensemble mean (b). The bucket means form the borders of the regions. They are used in pairs, for instance top and bottom 10% bucket means delimit the tail (c). The descriptor (c) also includes additional information: overall RMSE, RMSE-standard-deviation (s) and average-time-per-sample (t = number-of-samples-per-pixel × rendering-time-of-one-sample). Normalisation makes direct comparison possible for all descriptor parameters.

(Figure 5.1a). The error images are scaled to match the same rendering time $t = 1s$ for all algorithms (Section 5.3.3). The error images are then transformed into the discrete Fourier domain, the absolute value is taken and squared, resulting in $\mathcal{N}$ Fourier power spectra (Figure 5.1b). Power spectra are $\Theta(1/\mathcal{N})$ and an analogy to per pixel variance (Section 4.4) and hence well suited for analysis.

For every rendering we compute the radial average of the error power spectrum (a vector, Section 2.6.1) and the MSE (Figure 5.1c). These $\mathcal{N}$ datasets are used twice:

- The *ensemble mean* is formed by the per frequency mean of all $\mathcal{N}$ radial averages (Figures 5.2b and c).
- *Tail*, *body* and *head* regions are calculated as follows: Using the per-rendering MSE, we sort the $\mathcal{N}$ radial averages into 6 buckets of bottom and top 10%, 10-20% and 20-50%, respectively (Figure 5.2a). Then, the per frequency mean curves inside

those buckets are computed, in other words a 'per-bucket ensemble mean'. The means are used to delimit the regions. The tails span between bottom and top 10% mean pairs, the bodies between the 10-20% pairs and the heads between the 20-50% ones.

This fairly complicated sorting and averaging step is performed in order to maintain the correlation information between frequencies of a single spectrum. It would be lost for instance when selecting per-frequency quantiles. Averaging the error of several images together increases *confidence* and smooths the ensemble curves. This is especially important for low ensemble frequencies because the discrete Fourier spectrum contains fewer samples for those. We refrain from computing the variance of radial averages, because correlation between Fourier frequencies in MCMC algorithms and the number of averaged Fourier frequencies would have to be considered (compare with Section 4.5.1).

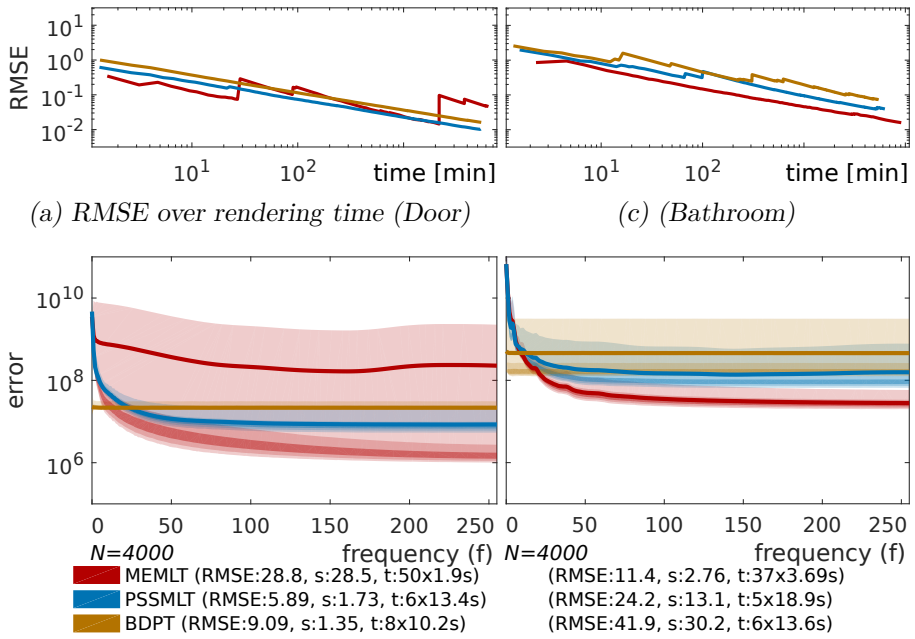
Overall RMSE, RMSE-standard-deviation (s) and the original rendering time ($t = $ *number of samples per pixel* $\times$ *average rendering time of an image with one sample per pixel*, just an additional information as data is time-normalised) are written to the legend. N in the figure denotes $\mathcal{N}$, i.e. the number of short renderings used. Together with t it is possible to calculate the processor time used to compute the data for that algorithm ($400 \times 10 \times 1.9$ seconds ≈ 2.1 hours for the example in Figure 5.2).

The ensemble mean is an estimator for the expected radial power spectrum average at $t = 1$. Values for an arbitrary time t can be computed simply by dividing through t (same for head, body and tail). The ensemble mean and the actual observed rendering error can be heavily influenced by outliers, which are visualised by the tail. We look at the ensemble mean to see how large error is *on average*, while body and head show how it is *most of the time*.

5.2 Examples and details

Let us take a look at the example in Figure 5.3.

(a) and (b), respectively (c) and (d), are two different visualisations of the same data. RMSE-over-rendering-time was generated by averaging the first n (x-Axis) short renderings. Outlier renderings are the reason for RMSE jumps. They also go into the the bucket with 10% largest error, and hence they only influence the tails (tails – analogous to PDFs). That explains the link between jumps, large tail area (long tails) and RMSE standard deviation demonstrated by MEMLT in (a/b) and BDPT in (c/d). However, tails do not give any information about the number of outliers, i.e., a long tails can be caused by one large outlier (jump) or several smaller ones. Similarly, a wide body indicates a large variation in the bulk of short renderings (it is more sensitive than RMSE-over-time). The head is something like the mode in a probability density function, i.e., a typical error spectrum.



(a) RMSE over rendering time (Door) *(c) (Bathroom)*

(b) Error Spectrum Ensemble (Door) *(d) (Bathroom)*

Figure 5.3: First and second row are different visualisations of the same data. A long tail in ESE are linked to RMSE jumps because the tails are generated from RMSE outliers, demonstrated by MEMLT in (a/b) and BDPT in (c/d). Typically, MC algorithms have a flat error spectrum while MCMC ones contain more error in low frequencies. Outliers influence the shape of the ensemble mean (a, MEMLT), spectral flatness indicates spatially small outliers.

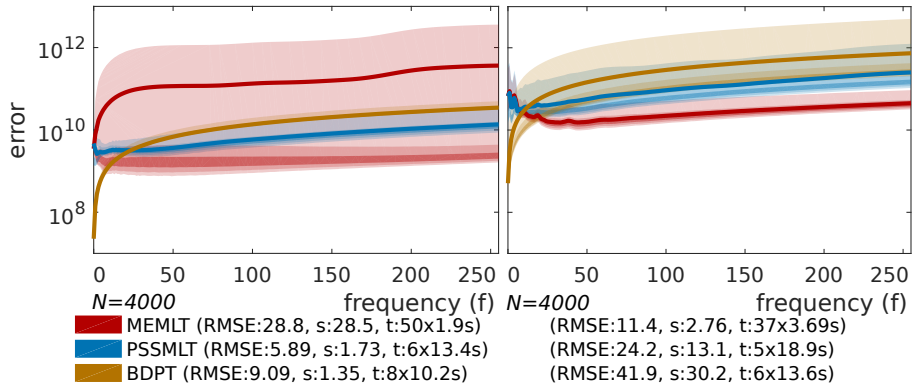
MC spectra are typically flat, while MCMC spectra show more error in low frequencies (see also Sections 4.3.2 and 4.3.2). The DC term (frequency 0) in MLT methods shows a very large error because it is estimated in a separate process step (Section 2.4.3), usually taking orders of magnitude fewer samples than during the relative estimation of the chain phase. Since PSSMLT uses a portion of its samples for independent candidates (effectively MC samples, Section 2.4.4), its ensemble mean can be often found between path-space MLT and MC methods. Error in high frequencies flattens out more quickly (compare heads in Figure 5.3b), while in low frequencies ($\neq$ DC term) it stays below path-space MLT (Figure 5.3d).

In Figure 5.3a, the upper border of the MEMLT tail has the same shape as its ensemble mean. It is common that outliers dominate the mean. Single-pixel outliers produce a constant spectrum, while error is concentrated more and more in low frequencies as the outlier area grows (Section 4.3.2). Therefore, we can say that the BDPT outliers in Figure 5.3d are single-pixel ones and those in MEMLT in Figure 5.3b have a small area (the upper tail is not completely flat).

5.2.1 Error sum per frequency

Until now we were talking about average error per frequency, not error sum or total error per frequency. The Fourier spectrum contains more directions in high frequency, the number is approximately[1] $2\pi f$ (where f is the ensemble frequency). Hence, the error sum (Figure 5.4) can be approximated by multiplying average error with $2\pi f$. It becomes clear that the amount of error in high frequencies is influenced not only by the sampling algorithm, but also by the dimensionality of the output (2D in case of rendering). In contrast to the ensemble mean, RMSE and MSE take dimensionality into account. In fact, averaging the ensemble sum curves would result in the spatial domain MSE[2].

However, we stick to average error because it is easier to read: Comparing summed error in low and high frequencies visually is difficult due to logarithmic scale and detecting single pixel outliers or measuring the width of the MCMC bump at low frequencies becomes hardly possible.



(a) Door *(b) Bathroom*

Figure 5.4: Error sum per frequency. Same data source as for Figure 5.3, where average error was shown. There are more Fourier directions in high frequencies, the curves are the result of scaling Figure 5.3 by $2\pi f$ (f being ensemble frequency).

[1] It is not exact because of discretisation in the 2D grid.

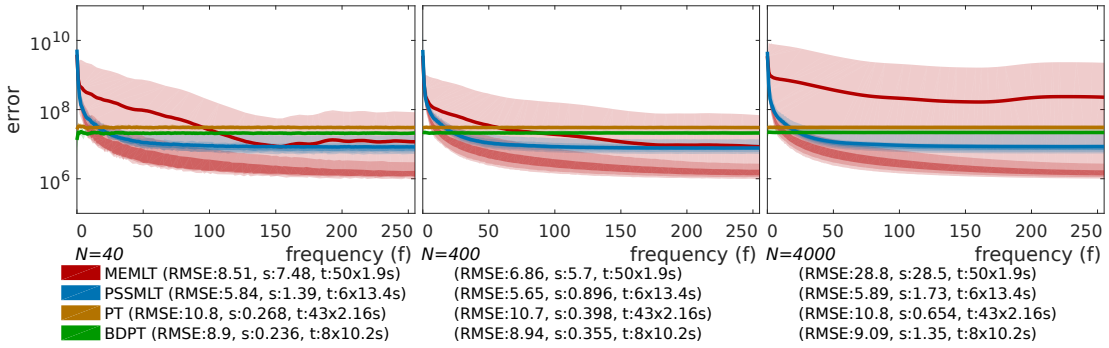
[2] Parseval's theorem, Section 2.6, a small error is introduced by ignoring frequencies larger than $0.5 \times \min(width, height)$.

5.2.2 Influence of $\mathcal{N}$, the number of short renderings

The ESE descriptor increases its accuracy (confidence) with more samples or longer rendering time, and converges to its expectation. Unlike traditional MSE, it does not become smaller. This is demonstrated in Figure 5.5, where heads and bodies are already relatively precise with small $\mathcal{N}$ (b). Tails and ensemble mean need more samples in some cases (compare b and c), because outliers are hard to catch.

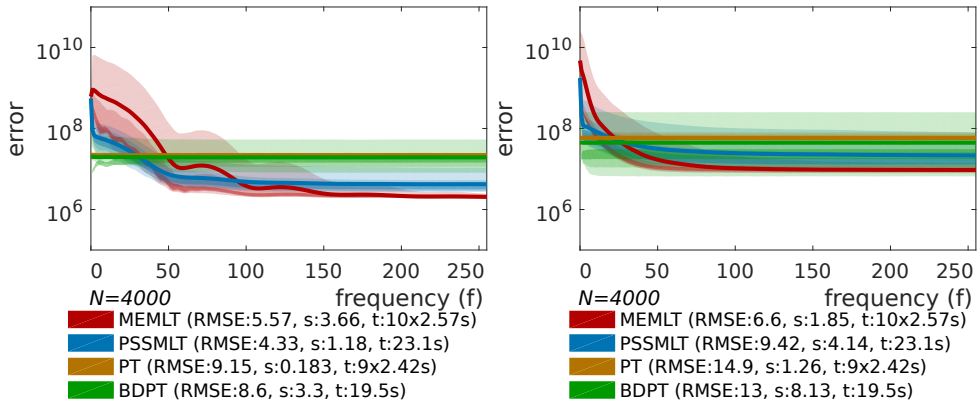
5.2.3 Absolute versus relative error

There are two widespread methods to compute error: (signed) absolute error $I - R$ and relative error $(I - R)/R$, where I is the image and R the reference (Section 4.2). We use absolute error but also tested relative. The two error types behave similarly in the sense that everything said above applies. As described in Section 4.2 and shown in Figure 5.6, problems exist in both cases. We decided on absolute error because it does not explode when the reference is dark or even zero.



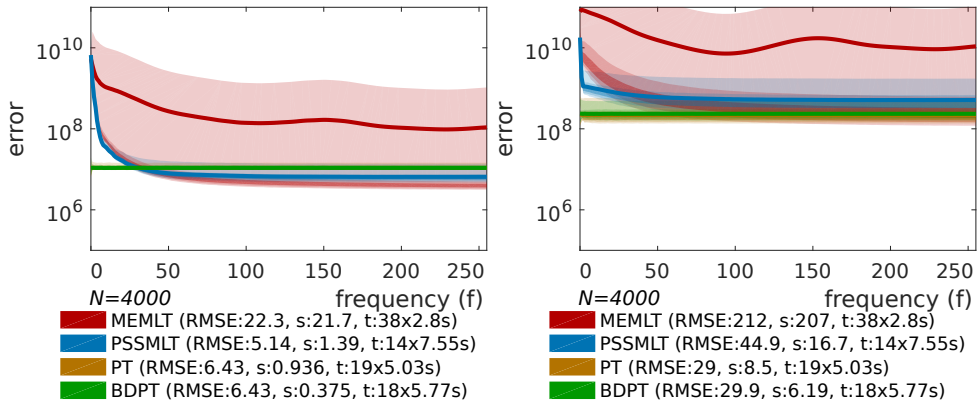
(a) Very small $\mathcal{N}$ *(b) Small $\mathcal{N}$* *(c) Large $\mathcal{N}$*

Figure 5.5: ESE for door scene with increasing $\mathcal{N}$. (a) is more wiggly (when zoomed), showing little confidence due to the small $\mathcal{N}$. The head of MEMLT is wider than with higher $\mathcal{N}$. (b) is smoother, the heads and bodies change only little when increasing $\mathcal{N}$ further. (c) caught more outliers, tail and mean of MEMLT are larger, showing that in fact MEMLT is the worst algorithm in this scene.



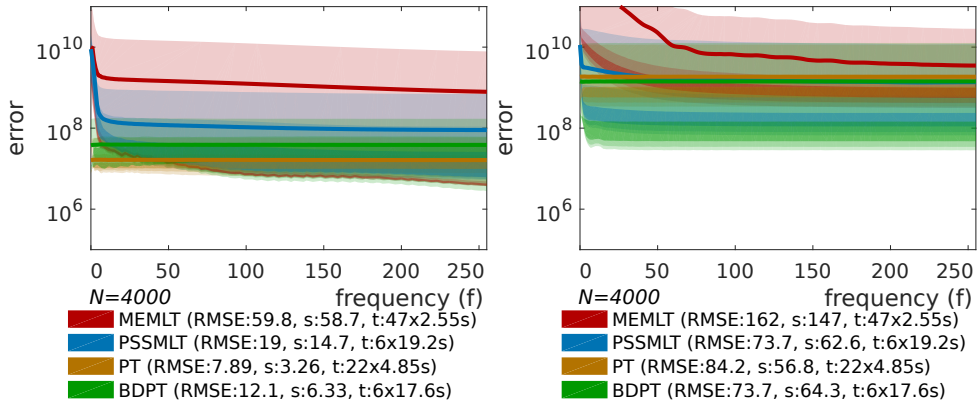
(a) Box scene absolute error

(b) Box scene relative error



(c) Kitchen absolute error

(d) Kitchen relative error



(e) Bottle absolute error

(f) Bottle relative error

Figure 5.6: Comparison between absolute and relative ESEs: Two bumps in absolute ESE of the Box scene (a) do not show in relative (b). In another scene there is a larger bump in relative error (d vs. c). Tails and body are sometimes wider in relative error (b and d). In absolute ESE of the Bottle scene, pixel or small area outliers can be seen (e), but not in relative error (f).

5.3 Implementation details

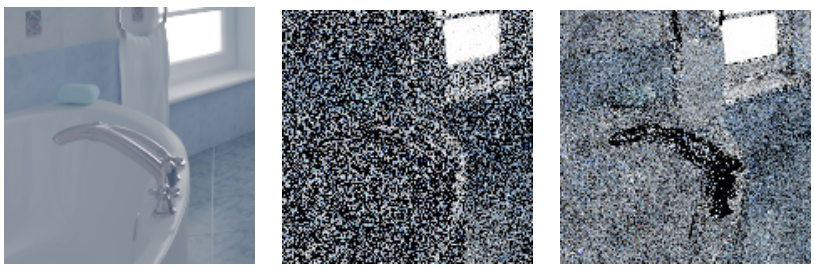
Images are stored in a 16 bit lossless high-dynamic-range format. Square images with a power-of-two size are beneficial for the discrete Fourier transform and the radial average (fewer cut-away frequencies).

5.3.1 References

In order to avoid measurable error in the references, we compute them on a cluster using a very large sample count, in some cases millions per pixel. Noise is not visible in the references, while it is heavy in short renderings (Figure 5.7). We verify that all algorithms converge to the same solution by looking at RMSE over time (Appendix D.2).

5.3.2 Short renderings

We adjusted the (average) number of samples per pixel – so that competing algorithms had similar rendering times, i.e., computer work load. This also has the effect that the raw error has a similar order of magnitude, since none of the algorithms is more than an order of magnitude better or worse than another. By manual inspection we verified that we have enough MLT samples, so that the chain covers the whole image and the majority of pixels have a reasonable value. This is not guaranteed by the MLT algorithm, e.g., one could configure MLT to have a single chain with a length that is smaller than the number of pixels. Pixels not reached by the chain would stay black, which results in an error value equal to the negative of the pixel value in the reference image. This could cause strange effects and hence we wanted to avoid it.



(a) Reference *(b) Path tracing* *(c) MLT*

Figure 5.7: Comparison between reference and short renderings: No visible noise in the reference, but heavy in short renderings.

5.3.3 Compute the error

The three colour channels are reduced to luminance before any non-linear operation, i.e., before raising to the power of two. The weights are

$$\hat{\bar{\mathcal{E}}} = 0.212671\hat{\bar{\mathcal{E}}}_R + 0.715160\hat{\bar{\mathcal{E}}}_G + 0.072169\hat{\bar{\mathcal{E}}}_B,$$

which are the same as in the used renderer, Mitsuba [Jak10].

In order to accommodate for different sampling costs, the error is scaled to match one second of (single-threaded) computation time on the rendering machine. All renderings of one algorithm use the same configuration, but there can be variations due to load, therefore we use the mean rendering time $\bar{t}$:

$$\hat{\bar{\mathcal{E}}}_{scaled} = \hat{\bar{\mathcal{E}}} \times \sqrt{\bar{t}}.$$

Scaling with $\sqrt{\bar{t}}$ in the spatial domain is equivalent to scaling per pixel variance, power spectrum, MSE or ESE with $\bar{t}$.

Next, using standard library functions, we compute the discrete Fourier power spectra of the error images. Dividing the mean of the power spectrum by the number of pixels yields MSE (Parseval's theorem, Equation 2.23).

5.3.4 Radial averages

There is no standard function for performing a radial average and in our context it is somewhat performance relevant. It is also important to be careful with aliasing errors (Section 5.4.1).

The steps of our method are

1. Generate a $\Phi \times F$ grid of polar coordinates with equidistantly spaced values $\varphi \in [0, 2\pi[$ and $r = f \in [0, 0.5 \times \min(width, height)[$. The sizes are $\Phi = 4 \times \min(width, height)$ and $F = 0.5 \times \min(width, height)$.

2. Transform the polar coordinates into Cartesian space (resulting in strong oversampling of low frequencies, which we do not mind because performance is not an issue).

3. The Cartesian coordinates are then used for bi-linearly interpolated sampling of values in the 2D power spectrum. The result is a matrix of size $\Phi \times F$, containing the power spectrum with axes being φ and f.

4. Radial means can be calculated as averages over the φ dimension. It is possible to experiment with quantiles from this transformed version, but they are just an approximation.

5.3.5 Generate descriptor

Ensemble mean, tails, body and head can now be computed as described in Section 5.1.

We estimate RMSE as $\sqrt{1/N \sum_{i=1}^{N} \mathcal{E}_{MSE,i}}$ (where $\mathcal{E}_{MSE,i}$ is the MSE of short rendering i). Note that this is different to the mean of per-rendering RMSEs, because the order of operations matters. Our way of computation has a larger confidence than computing RMSE of a single image with a rendering time of $\mathcal{N} \times t$ (the traditional method, Section 3.1). RMSE is used because contrary to MSE it has a linear scale (like standard deviation).

RMSE standard deviation is $\sqrt{\mathrm{Var}\left(\mathcal{E}_{RMSE}\right)}$, i.e., it shows the standard deviation of the random variable *RMSE of a 1-second rendering*. Both, RMSE and its standard deviation, become more precise (more confident) with increasing $\mathcal{N}$.

Results are plotted in a figure with logarithmic error axis.

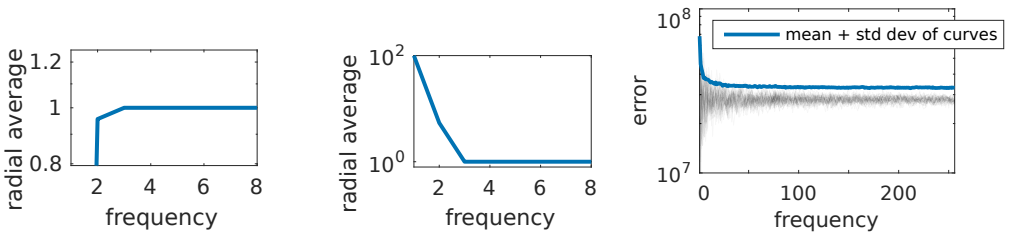
5.4 Caveats

5.4.1 Radial average

The following refers to the radial average method described in Section 5.3.4.

Due to bi-linear interpolation, neighbouring Fourier frequencies with different wavelengths (and directions) are mixed together, resulting in smoothing of the ensemble graphs. This is visible in very low frequencies (Figures 5.8 a and b). In particular, MLT's DC term influences the first ensemble frequency.

The discrete Fourier transform has fewer directions in low frequencies. Therefore, the averages have more variance in low frequencies, just because there is less data (Figure



(a) $DC = 0.01$ (b) $DC = 100$ (c) *Standard deviation of radial averages.*

Figure 5.8: (a) and (b): The DC term influences the first ensemble frequency due to bi-linear interpolation. This can be demonstrated by a radial average of a 16×16 *power spectrum with non-DC frequencies set to* 1*. (c): The standard deviation of radial averages can be higher for low ensemble frequencies although standard deviation is equal for all Fourier frequencies. The reason is that the radial average has fewer power spectrum values in low ensemble frequencies. A correction would need to take the correlation between Fourier frequencies into account (compare with Equation 4.7).*

5.8c). However, this does not mean that there is more variation in the Fourier frequencies themselves. This uncertainty could only be solved by a different (custom) frequency-transform method with more directions in low frequencies, but the usefulness is unclear. Moreover, uncertainty is alleviated by larger numbers of renderings (Figure 5.5).

5.4.2 Outliers

Outliers can be so rare that they are really hard to "catch", while still having a significant impact on the mean due to their large value (in particular correlated area outliers). This is why we use a relatively large rendering count ($\mathcal{N}$). However, in practice, if the tail is significantly larger than twice the body, do not trust the ensemble mean.

CHAPTER 6

Results

Sections 6.1 and 6.2 show, based on several algorithms and scenes, how insights can be gained using ESE and standard deviation. However, compared to analysing different algorithms, it might be more interesting to analyse the effects of parameter changes on a single algorithm: Section 6.3 shows the influence of mutation size on MLT algorithms, and Section 6.4 studies the influence of various seeding strategies. One important result is the fact, that the proxy algorithm performance is similar to the unmodified (pure) MLT algorithm. Finally, it is possible to analyse biased but consistent algorithms like SPPM, and algorithms that have a better convergence rate than $\Theta(1/N)$ (for instance sampling patterns in MC), with ESE. However, the process is different and we show an example in Section 6.5.

We generated all data using the Mitsuba renderer [Jak10], which implements all algorithms that we examine. All the processing was done in Matlab, using the EXR format for HDR and a resolution of 512×512. Some of the renderings (experiments and references) were performed using cluster computer resources within the Aalto University School of Science "Science-IT" project. All short renderings were made on a workstation with 40 virtual cores (20 physical) using 40 separate, single threaded processes.

The colour map for standard-deviation pictures in this chapter is shown in Figure 6.1, values above 20 are clamped. All ESEs have the same scale.

We set the maximum number of light bounces to infinity. If used, the respective MCMC algorithms are employed for all light bounces, including direct light (unlike proposed in the original work [Vea97]). PSSMLT uses the BDPT sampler in all of our tests.

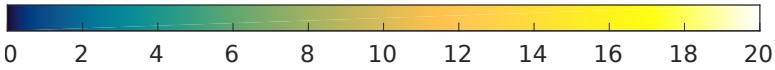


Figure 6.1: Colour map for standard deviation pictures

6.1 Simple parametric scene

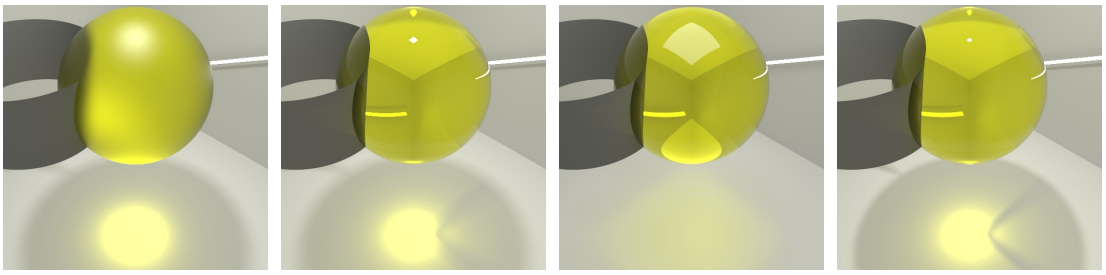
The box test scene (Figure 6.2) contains a parametrised light source at the ceiling, a large sphere with parametrised roughness in the middle, a small light on the right wall and cylinder walls left from the centre (intersecting the sphere). Sphere and top light produce a caustic on the floor, whose size and sharpness depends on the parameters. Light-source size and intensity are correlated so that the total power remains constant. The default light size is 50 and roughness 1/256.

This test serves mainly to verify the descriptor, as we can predict how it should react (e.g., larger error with smaller light and smoother surfaces). Figure 6.3 shows a standard-deviation image for a hard configuration. Due to squaring, convergence of standard deviation takes longer than the actual image. However, areas with large error are visible nonetheless. More configurations and example renderings are in Appendix B.1.

When looking at Figures 6.4 and 6.5, we can see several things:

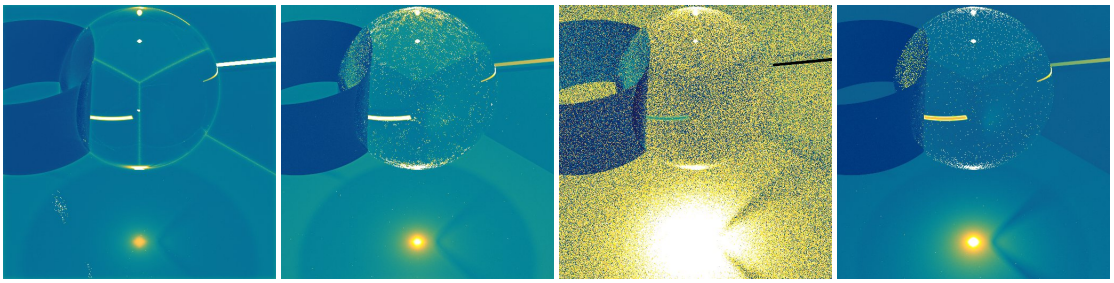
- As expected, error levels and number of outliers rise in general when increasing the difficulty of the scene (with the following exception).
- It seems like – contrary to expectations – the scenes from Figures 6.4c and 6.5b are harder to render than those with smaller light source or smoother surface. It might just be a demonstration of the effect of outliers, but on the other hand tails are consistent for 3 algorithms in 6.5b and 2 in 6.4c (both sampling via BDPT). That might mean that there is a group of light paths that are just on the verge of being impossible and therefore sampling probability is low. When further changing the parameters, they become impossible and chances they produce outliers are nil.
- Outliers can offset the ensemble mean very far, it often has the shape of the upper tail edge (for instance Figures 6.5b and 6.4e).
- The tail width roughly corresponds to RMSE jumps, though it does not tell whether there is one big jump (Figure 6.5b MEMLT) or many small ones (Figure 6.5e MEMLT).
- As expected, MC algorithms have a flat spectrum and MCMC algorithms show more error in low frequencies. In almost all figures (including complex scenes in the Appendix A) it is visible that PSSMLT is a mixture between MC and MCMC. PSSMLT's head position and tail width is between MEMLT's and BDPT's in many cases. There is more error in low frequencies, but not as pronounced as in MEMLT, and PSSMLT flattens out in high frequencies more quickly than MEMLT.
- In Figures 6.4e and 6.5e (highly specular and small light) MEMLT is a relatively good algorithm. However the bodies and tails are wide and high in low frequencies, which indicates that there is a large variation between short renderings on large areas. This in turn means that the chain had problems jumping between light effects.

- Changing roughness below 1/16 (Figure 6.4b) does not influence PT very much. The reason are shorter rendering times, which are visible in the legend. Time drops from 2.69 seconds for one sample per pixel (1/16) to 2.42 (1/1024). Probably glossy surfaces result in close to parallel rays, causing cache hits and therefore more efficient samples.
- We were a bit surprised by BDPT's tendency towards outliers, but the behaviour is consistent across several scenes, also complex ones. The reason might simply be that samples are more expensive and fewer can be taken. This is bad in cases a light effect is not well covered by any bidirectional sampling strategy.



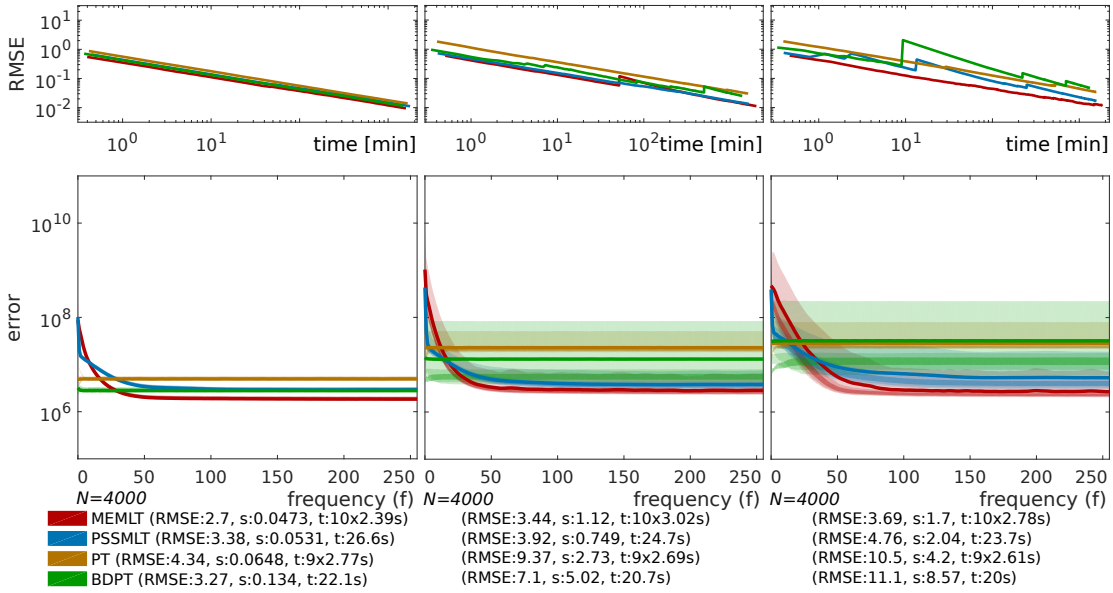
(a) Roughness 1/4 *(b) Roughness* 1/1024 *(d) light size 400* *(e) light size 25*
light size 50 *Roughness* 1/256

Figure 6.2: Reference images for box scene with selected parameter combinations.

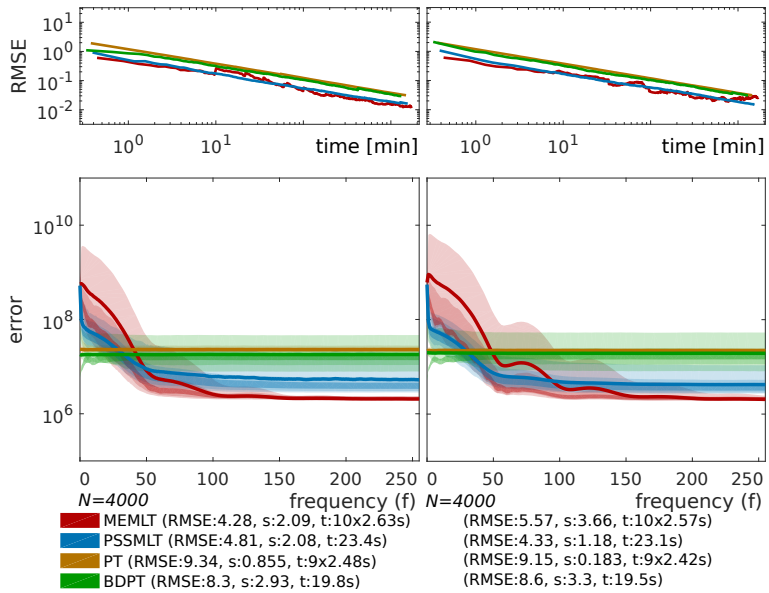


(a) MEMLT *(b) PSSMLT* *(c) PT* *(d) BDPT*

Figure 6.3: Standard deviation per pixel for box scene with light size 25*. Roughness is* 1/256*. More parameter configurations in Appendix B.1.*

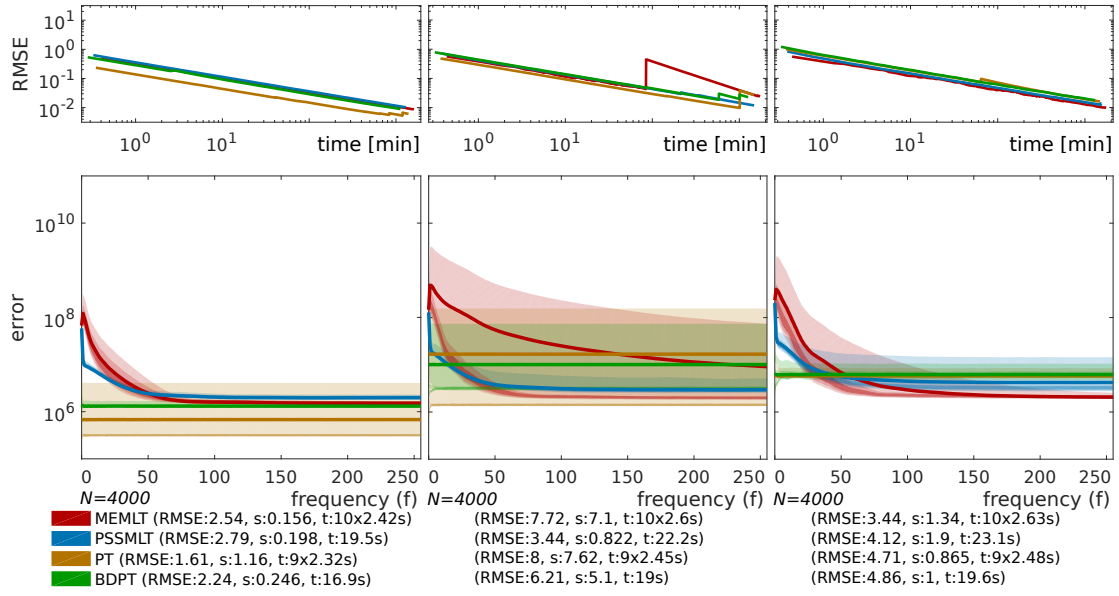


(a) Roughness 1/4 *(b) Roughness* 1/16 *(c) Roughness* 1/64

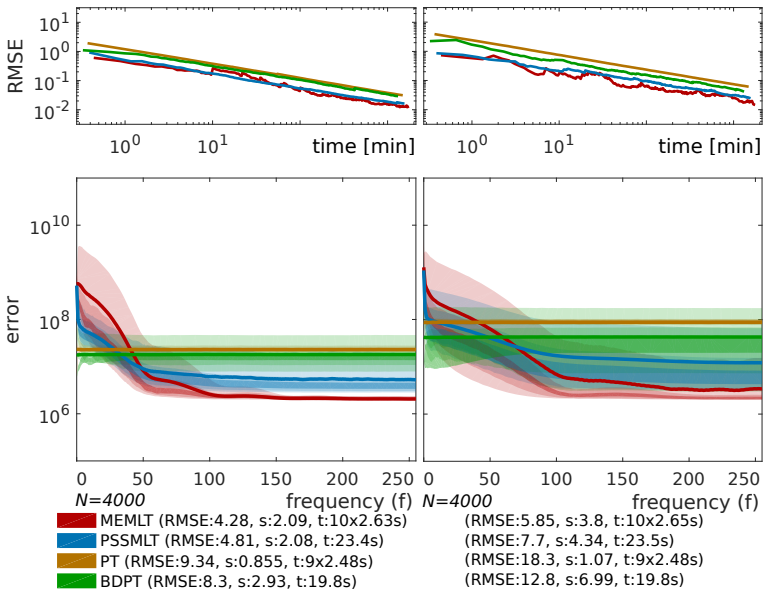


(d) Roughness 1/256 *(e) Roughness* 1/1024

Figure 6.4: Error descriptors for box scene with varying roughness. Light size is 50. See text for details.



(a) Size 400 *(b) Size 200* *(c) Size 100*



(d) Size 50 *(e) Size 25*

Figure 6.5: Error descriptors for box scene with varying light size. Roughness is 1/256. *See text for details.*

6.2 Complex scenes

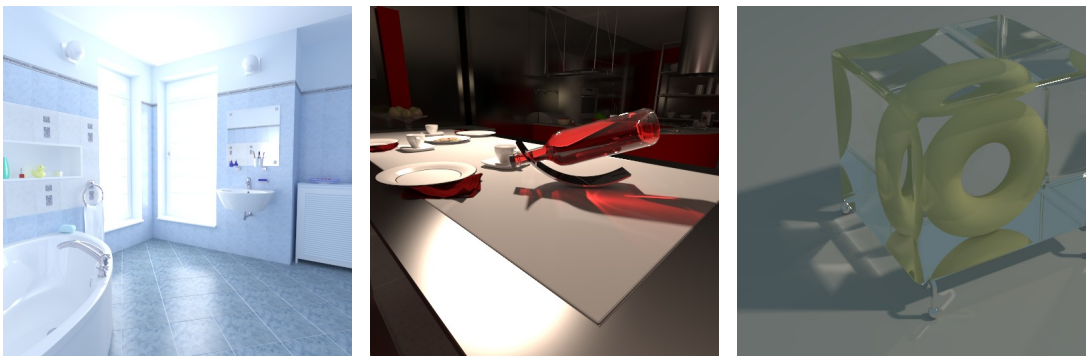
The results discussed here are limited to only three scenes (Figure 6.6), but we show all of them in Appendix A, including figures with relative error.

One of the things we want to stress here is the usefulness of standard-deviation images. They uncover which light effect is hard to sample for a certain algorithm. An example are corners and edges, problematic for MEMLT (see Bathroom in Figure 6.7c and Kitchen, Sponza and Door in the appendix).

In the Bathroom scene (Figure 6.7), BDPT shows worse performance than PT (both heads and ensemble means). MEMLT has highest performance. Bathroom and the diffuse-only Sponza scene are the only ones from the test set without MEMLT outliers.

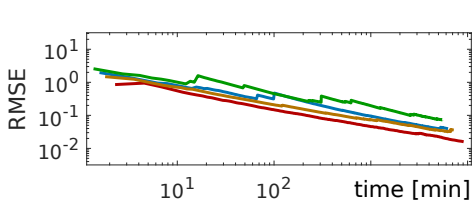
The challenge in the Bottle scene (Figure 6.8) is the infinite depth of refractions. ESE shows that all algorithms produce pixel outliers, visible by a flat upper tail edge. The largest outliers are in MEMLT, which might be a bit surprising at first: MEMLT includes a very powerful manifold mutation designed to perform well on specular refractions, but there is no efficient large mutation to jump between manifolds.

In the standard-deviation pictures of the Torus scene (Figure 6.9), MEMLT shows almost no error *on* the torus, but large error in deep refractions. Just as with the bottle, we suspect poor performance of global mutations. From ESE we see that MEMLT is worst due to outliers, the head is lowest. Both head and body are relatively wide, showing a large variation between runs, i.e., there would be a lot of flickering when observing the rendering process. PT is the best algorithm, even tough there is a lot of error in the caustic (see the standard-deviation picture).

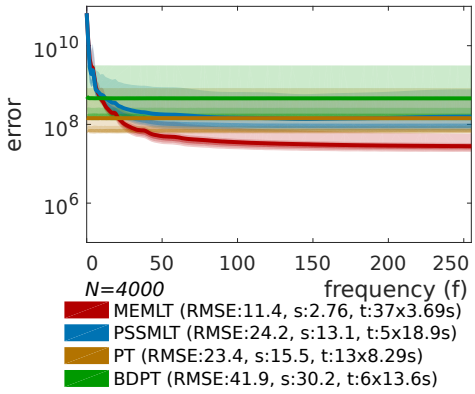


(a) Bathroom *(b) Bottle* *(c) Torus*

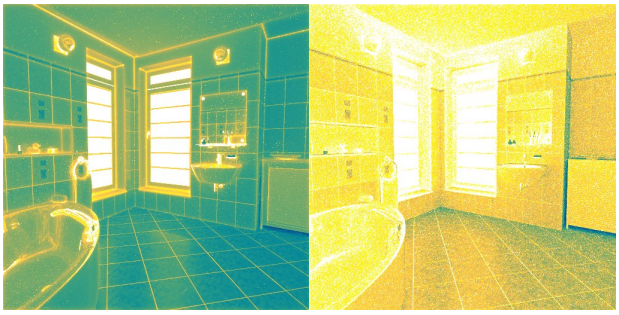
Figure 6.6: Reference images for the complex scenes used in our tests.



(a) *RMSE over time*



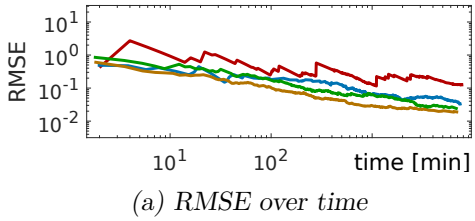
(b) *Ensemble power spectrum*



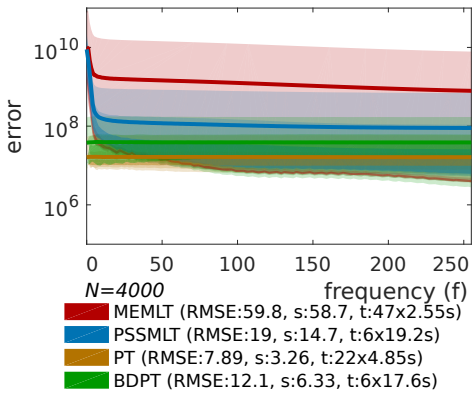
(c) *MEMLT* (d) *PSSMLT*

(e) *PT* (f) *BDPT*

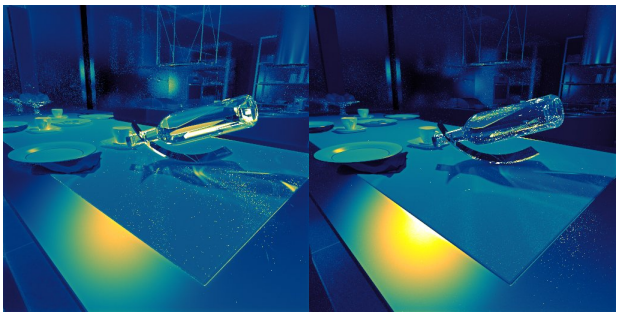
Figure 6.7: Bathroom scene, standard deviation per pixel on the right.



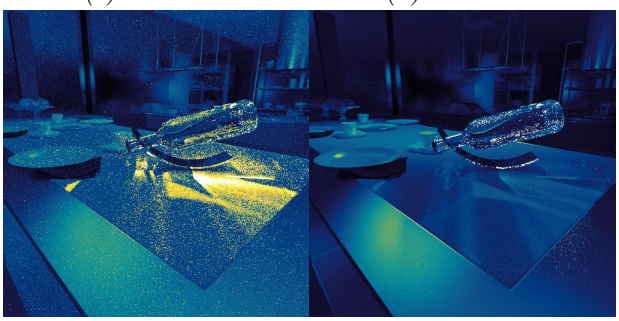
(a) *RMSE over time*



(b) *Ensemble power spectrum*

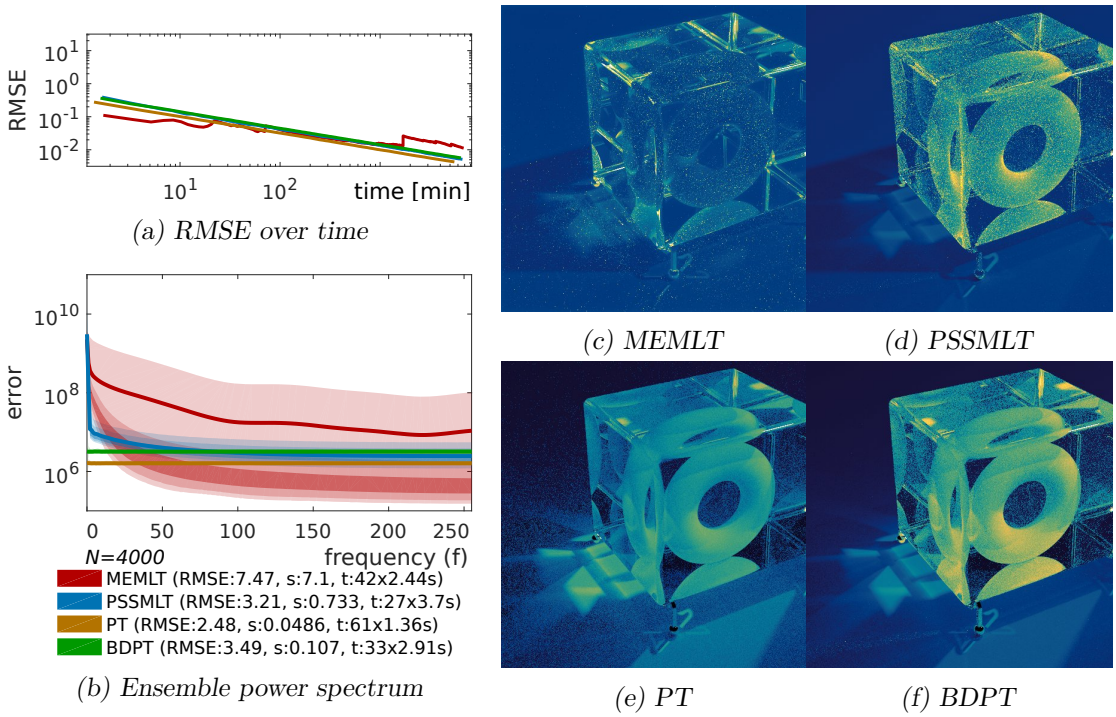


(c) *MEMLT* (d) *PSSMLT*



(e) *PT* (f) *BDPT*

Figure 6.8: Bottle scene, standard deviation per pixel on the right.



(a) RMSE over time

(b) Ensemble power spectrum

(c) MEMLT

(d) PSSMLT

(e) PT

(f) BDPT

Figure 6.9: Torus scene, standard deviation per pixel on the right.

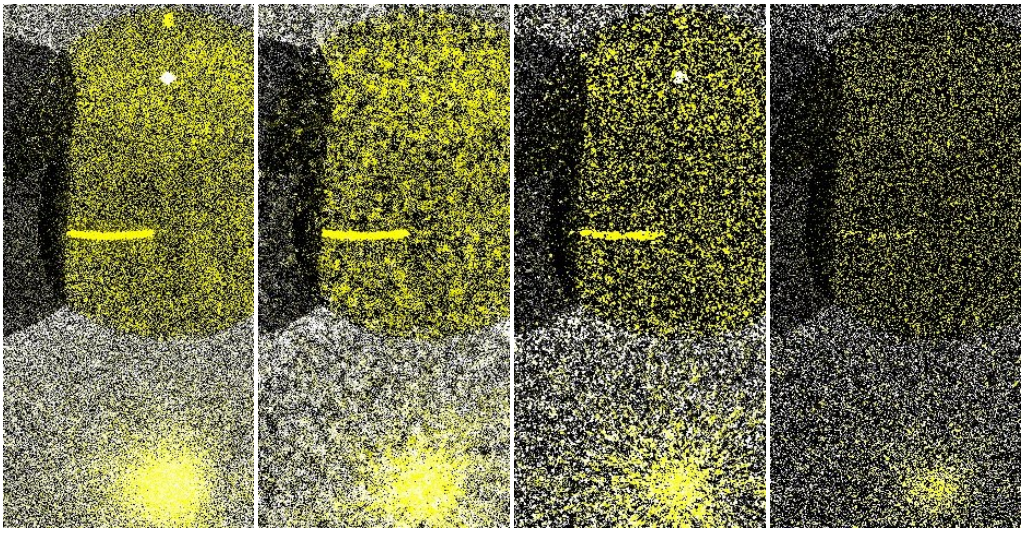
6.3 PSSMLT parameter variation

PSSMLT has parameters to directly change mutation size (small mutations) and the length of correlated chains (percentage of large mutations). We are interested in how changing those parameters affects the error. In addition to the original algorithm, the same parameter changes were tested with a *fake* PSSMLT algorithm. The fake algorithm operates similarly to PSSMLT, small and large mutations, acceptance probability and seeding are the same. Contrary to the original, rays are not traced – instead, the reference image is used as the target function of the MLT mutation. It does not get stuck so easily due to the reduced dimensionality. When applied to a reference image with constant intensity, the result of the fake algorithm resembles Brownian motion. We used the box scene described in Section 6.1 with roughness 1/256 and light size 50.

The effect of changing the mutation size is well visible when looking at example renders (Figure 6.10, full size and standard deviation in appendix Section B.3). With very small mutations, samples stay pretty much in place, they do not move over the image plane (Figure 6.10d, the image looks dark because large values are clamped). When increasing mutation size, the small area outliers morph into larger and less intense patches (Figures 6.10b and 6.10c). We could say that changing the mutation size changes the *correlation radius* of pixels. Finally, mutations are so large that the pixels look independent (Figure 6.10a), almost like BDPT (compare with Figure B.3d).

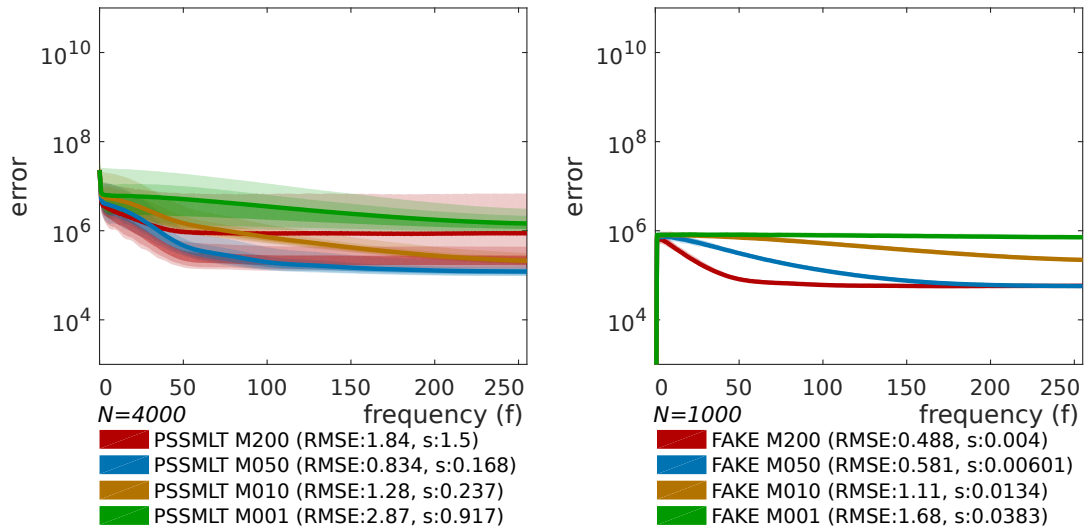
The correlation radius is also visible in ESE for both – real and fake – PSSMLT algorithms (Figure 6.11). Increasing the mutation size reduces the width of the error peak in low frequencies and vice versa. Naturally, in real PSSMLT there are limits, as the integration domain is more complex.

Increasing the percentage of large mutations moves between a strongly correlated MLT algorithm and an independent MC sampling method (Figure 6.12). ESEs (Figure 6.13) show a reduction of height of the error peak in low frequencies, but the width stays the same. Error in high frequencies becomes larger only when 100% large mutations are reached and the spectrum becomes flat.



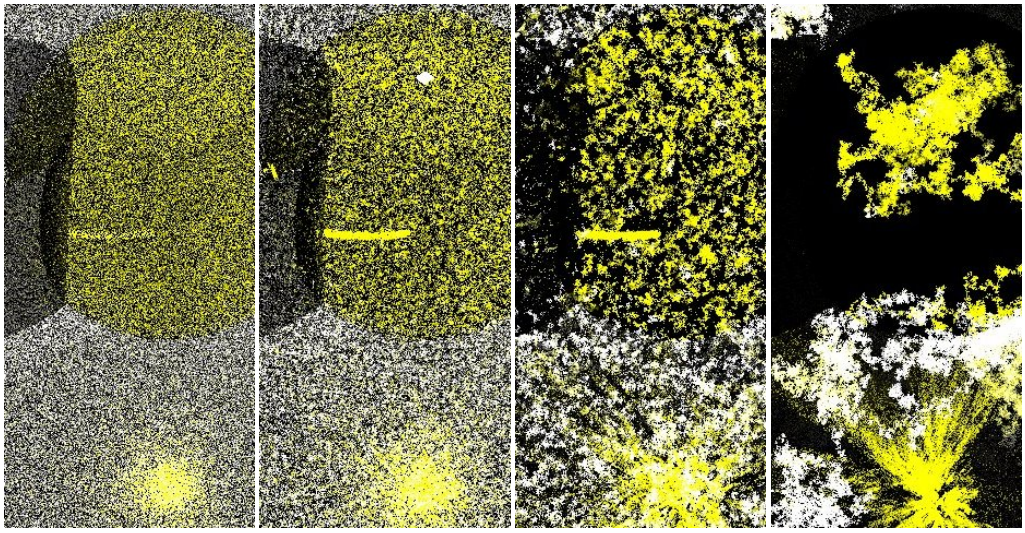
(a) 200% (b) 50% (c) 10% (d) 1% mutation size

Figure 6.10: Example render for changing small mutation size (30% percentage large mutations): (a) looks almost like BDPT as mutations are so large. Correlation increases in (b) and (c), but the samples still visibly move around the image. In (d) the chain hardly moves away from high-intensity samples, producing small-area outliers.



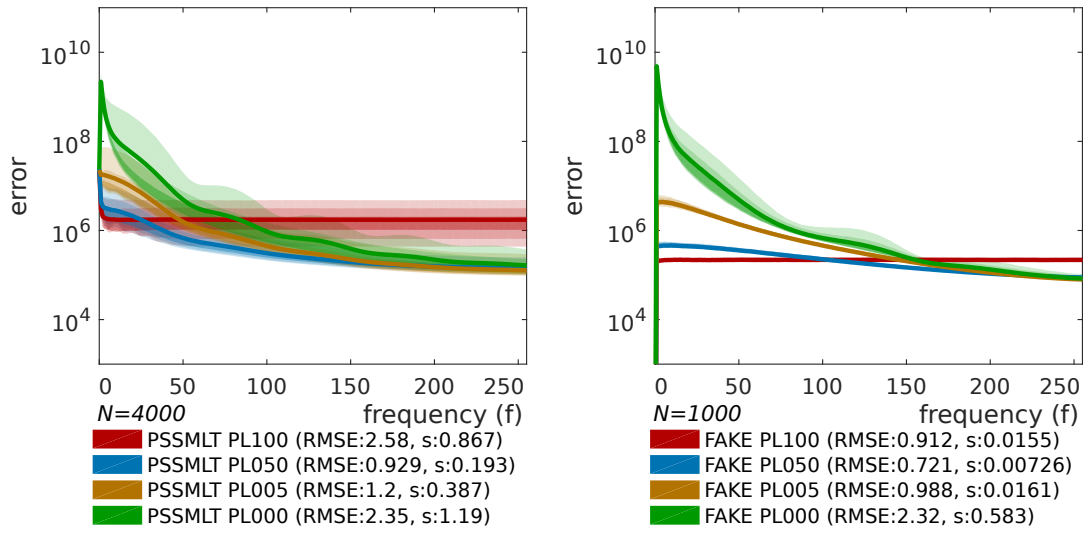
(a) PSSMLT (b) Fake MLT

Figure 6.11: ESEs for 200, 50, 10 and 1% of default small mutation size, 30% percentage large mutations. In fake MLT, the size of small mutations is directly related to the width of the error peak in low frequencies. In PSSMLT the effect also exists, but it is not as clear as there are other overlaying effects, e.g. outliers.



(a) 100% *(b) 50%* *(c) 5%* *(d) 0% large mutations*

Figure 6.12: Example renders for changing percentage of large mutations (small mutation size 25% of default). Correlation between pixels is related to the percentage of large mutations. Pixels in (a) are independent, while in (d) they show strong correlation.



(a) PSSMLT *(b) Fake MLT*

Figure 6.13: ESEs for changing percentage of large mutations, small mutations 25% of default value. When increasing the percentage of large mutations, error in low frequencies is reduced, making the ESE more similar to MC spectra. When increasing up to 100% large mutations, ESE is completely flat at the cost of larger error in high frequencies. The same effect takes place for the default small mutation size, but not as pronounced (see appendix, Section B.2).

6.4 MLT seeding and chain length configurations

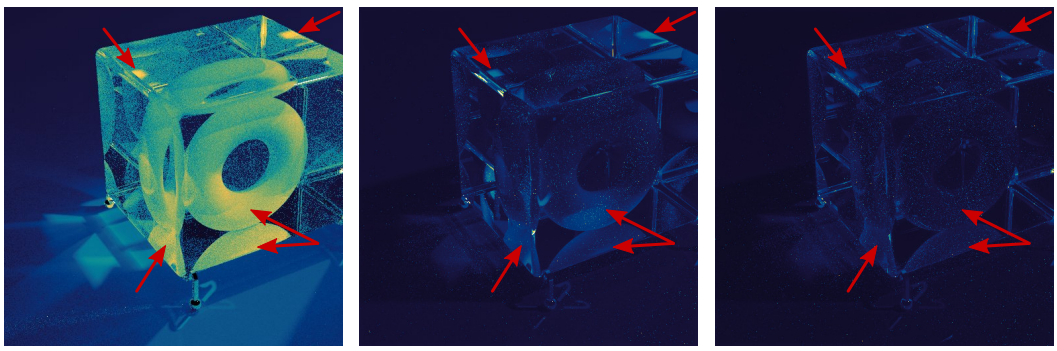
We test how seeding parameters and chain lengths influence MLT performance. Only path-space MLT is studied, but some of the results also apply to PSSMLT. One large difference in PSSMLT are the large mutations, which produce independent samples and therefore *break* the chain into chunks. This should make PSSMLT less sensitive to variation in chain length and more independent of the starting path.

The data in the following three sections is generated with 50 samples per pixel. We removed the time measurement and error scaling to avoid inaccuracies due to inefficient workloads. Only the most interesting results are shown here, the rest in Appendix B.5.

6.4.1 Seed-pool size (number of luminance samples)

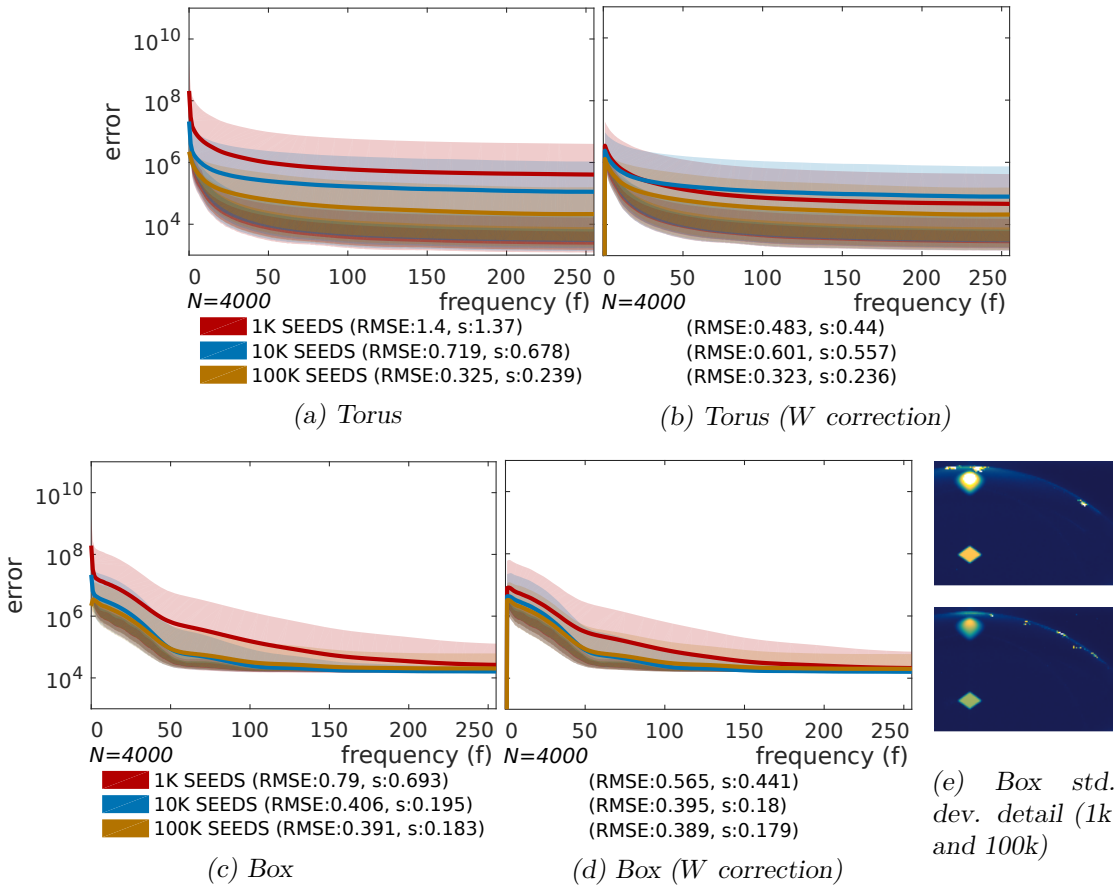
This parameter has two implications. The first one is the accuracy of the scaling factor W in Equation 2.17. It is possible to factor out the influence in short renders by rescaling with the luminance of the reference (labelled W correction). The resulting algorithm becomes biased. When comparing the two variants in Figure 6.15, it is visible that with few luminance samples, the variance of W significantly increases the error.

The second implication is the distribution of starting paths sampled from the pool. With a small pool the probability of starting several chains from the same seed path is relatively high, especially when the pool contains MC outliers. Additionally, with a larger seed pool the starting points are distributed closer to the equilibrium distribution. The effect is visible in Figures 6.15(b and d): error in low frequencies is strictly decreasing with rising pool sizes. In the standard deviation image (Figure 6.15e) it is visible that with more seeds there is less error in the two hard-to-sample light effects. Figure 6.14 shows that light effects that are problematic for the seeding method (a) also show more error in the MLT image if the seed pool is too small (compare b and c).



(a) BDPT *(b) 1k seeds* *(c) 100k seeds*

Figure 6.14: Torus standard deviation: Light effects that are hard to sample for the algorithm populating the seeding pool (BDPT) are also problematic if the pool size is small (indicated by the arrows).



(a) Torus *(b) Torus (W correction)*

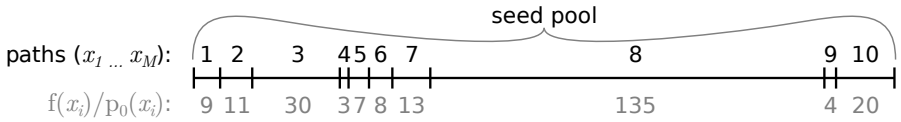
(c) Box *(d) Box (W correction)*

(e) Box std. dev. detail (1k and 100k)

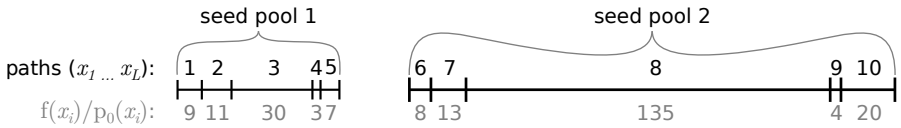
Figure 6.15: Varying seed pool-sizes (number of luminance samples): Reducing the size of the seed pool increases variance of the chain weight W (Section 2.4.3), which in turn increases overall error (a and c). But this is not the only effect as can be seen in (b) and (d), where W was replaced by the reference's W: An increased pool size benefits low frequencies (while high frequencies are inconclusive). (e) shows two bright and hard-to-sample light effects: If the seed pool is small, chances are no sample will cover those effects and W is relatively small, reducing the contribution of the whole chain. Once luminance samples do cover the light effects, W and therefore the contribution will be larger. Additionally the probability of starting several chains from the same path is high. This means that in case of small seed pools, light effects that are hard to sample for the seeding method, cause outliers. More scenes in Figure B.15.

6.4.2 Number of seeds per pool

Instead of having one large seed pool (Figure 6.16a) it is possible to subdivide the luminance samples into several smaller pools (Figure 6.16b). In case there is one weight W per pool, some chains become more *important* than others. Benefits and drawbacks apply as described in Section 6.4.1. We implement this type of configuration implicitly for our descriptor by having many short renderings (Section 4.1.1).



(a) One seed pool of size $L = 10$



(b) $S = 2$ seed pools

Figure 6.16: $f(x_i)/p_0(x_i)$ is the contribution of the path. Sampling probability of the starting path is proportional to that number and the per-pool weight W is the average. Conventionally, one seed pool is used in MLT, from which C chains are sampled (a). It is possible to divide the L seed paths into S pools (b) and sample C/S starting paths from each, resulting in S different chain weights. The probability of starting several chains from the same path is lower, but weights W are less precise.

The experiment works as follows: The same number of luminance samples ($L = 10k$), chain length (512×512) and count ($C = 50$) are used, but different pooling strategies are tested by computing the ESE descriptor from $\mathcal{N} = 4000$ short renders:

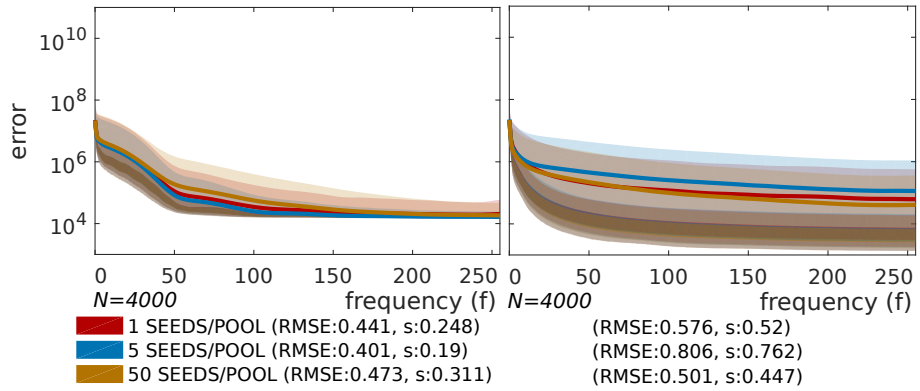
1. One chain per pool, luminance samples are divided into 50 pools of size 200. This strategy guarantees that no sample is used twice as a starting path.

2. Five seeds per pool, luminance samples are divided into 10 pools of size 1000.

3. All 50 seeds from one pool. This is the default implementation in Mitsuba [Jak10].

Figure 6.17 shows the results. There are differences in ensemble mean, but they are not consistent and likely due to the hard-to-measure outliers. The heads are almost equal, differences in standard deviation are inconclusive and the strategies have a smaller effect when the pool size is increasing.

We therefore conclude that the changes introduced to the MLT algorithm by our proxy are minor and hardly measurable.



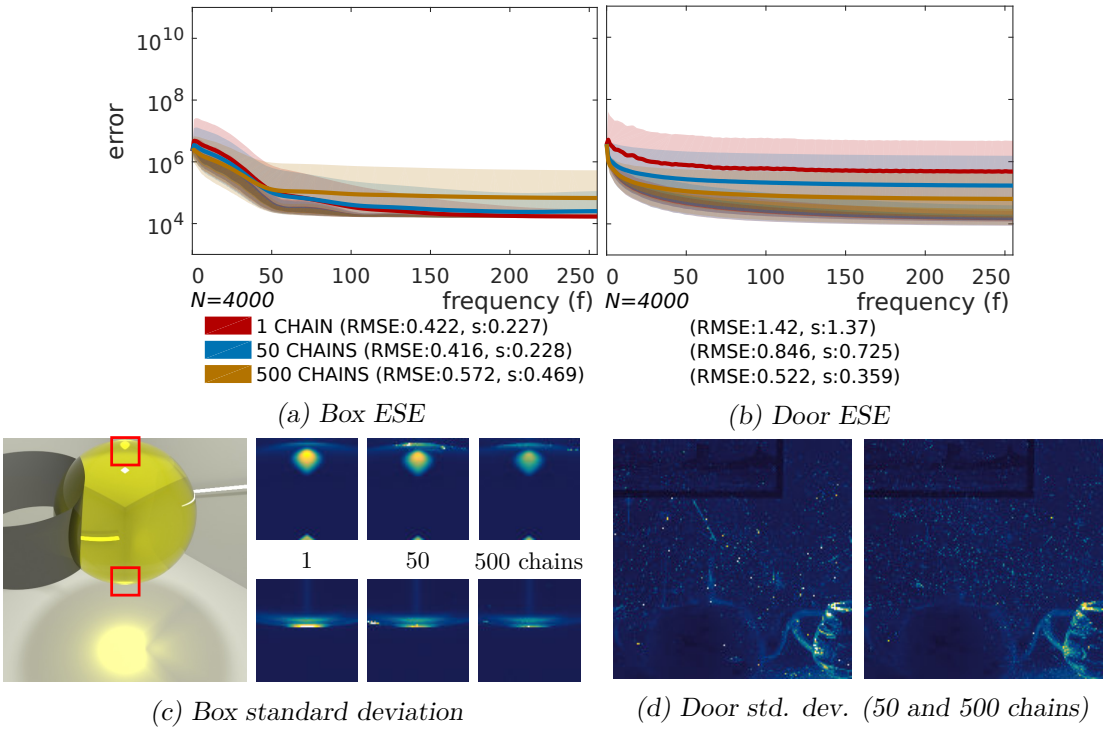
(a) Box *(b) Torus*

Figure 6.17: ESEs for various pooling strategies of the luminance samples into seed pools. The total number of luminance samples is 10k, which is partitioned randomly into 50, 10 and 1 seed pools with 1, 5 and 50 chains, each. The body and heads of the variants are almost equal, differences in tails and standard deviation images are inconclusive. This indicates that the strategy plays only a minor role, which becomes even smaller when increasing the overall pool size. More scenes in Figure B.16.

6.4.3 Chain length

Given a fixed sampling budget, it is possible to run either one very long chain (the original Veach-MLT), several shorter ones (like in Mitsuba [Jak10]), or many very short ones. Figure 6.18 shows how these options influence the error. Note that increasing the number of chains puts *pressure* on the seed pool, i.e., the number of luminance samples should increase, otherwise problems described in Section 6.4.1 start appearing.

Reducing the chain length seems to fix two problems: First, bright light effects that are hard to reach for chain mutations are now included through the re-sampling phase (the starting path is drawn from an almost perfect equilibrium distribution). This is visible in hard-to-sample refractions of the Box scene (Figure 6.18c), ensemble mean strictly moves down in low and up in high frequencies (Figure 6.18a). Second, if the chain is stuck somewhere, it is broken up earlier by stopping. This results in less intense pixel outliers in the door scene, cutting RMSE error to 36% (Figures 6.18b and d).



(a) Box ESE *(b) Door ESE*

(c) Box standard deviation *(d) Door std. dev. (50 and 500 chains)*

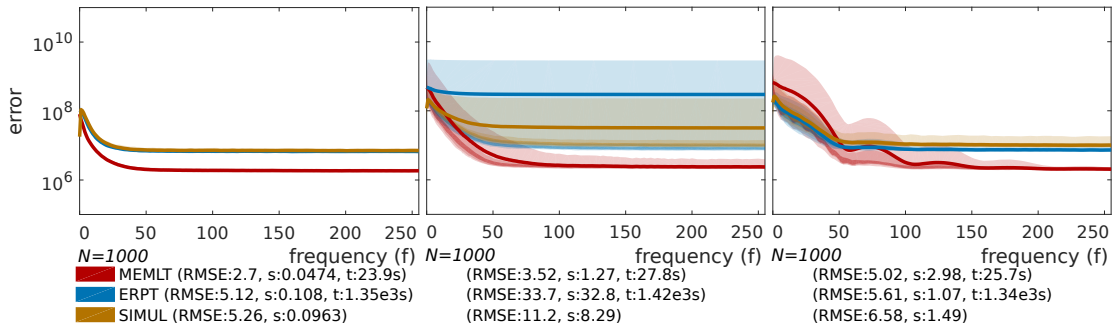
Figure 6.18: ESEs for varying chain lengths (100k seed pool, 50 samples per pixel). Reducing chain length shifts error in the box scene from low to high frequencies (a), i.e., less error in refractions (c) and more in pixel outliers. In the door scene, short chains reduce the amplitude of pixel outliers (d), cutting RMSE error to 36% (b). More scenes in Figure B.14.

6.4.4 Comparison with Energy Redistribution Path Tracing

In this experiment we want to test how well ERPT can be simulated by MLT configured for short chains and a large seed pool (labelled SIMUL).

We used the following settings:

- 10 samples per pixel (those are seeding paths) for ERPT and accordingly $512 \times 512 \times 10 \approx 2.6e6$ luminance samples for MLT.
- on average 1 chain per pixel (they are started probabilistically) for ERPT and accordingly $512 \times 512 \approx 2.6e5$ chains for MLT
- chain length of 200 for ERPT and accordingly 200 samples per pixel for MLT
- ERPT and MLT use only the manifold mutation (MEMLT [JS12]).



(a) Roughness 1/4 *(b) Roughness* 1/64 *(c) Roughness* 1/1024

Figure 6.19: ERPT test with box scene (varying roughness, light size is 50). SIMUL is MEMLT with parameters tuned to simulate ERPT (see text). MEMLT was not scaled to match ERPT/SIMUL, therefore only the shape can be compared. The figure shows that ERPT can be approximated pretty closely by tuning MLT parameters. (c) shows in addition that ERPT and SIMUL follow the head of MEMLT, avoiding the outliers in the tail of MEMLT. (b) on the other hand demonstrates that ERPT and SIMUL can also have a long tail. SIMUL is very slow because the architecture of the used rendering system was not designed for the tuned parameters (see text).

The rendering time for SIMUL is very long because our MLT implementation is not designed for this type of workload. Since the total number of samples is equal in both cases, we scale the error of the inefficient SIMUL using ERPT's time.

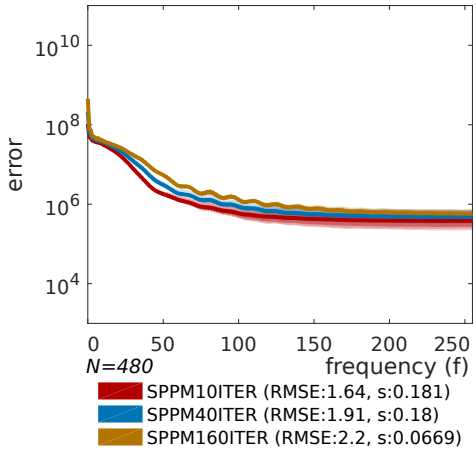
The results are in Figure 6.19, along with a normal MLT algorithm for comparison. The performance of ERPT and SIMUL are similar: The head of ERPT is slightly below SIMUL. ERPT has more outliers with roughness 1/64 (b), but that could be coincidence. MEMLT has the best performance in all three scenes, but that might be due to a less optimised implementation or a bad parameter choice for ERPT. It is interesting however, that SIMUL and ERPT in Figure 6.19c seem like a continuation of further increasing the number of chains in Figure 6.18a: Error in low frequencies is below default MEMLT and in high frequencies it is above.

6.5 Stochastic Progressive Photon mapping

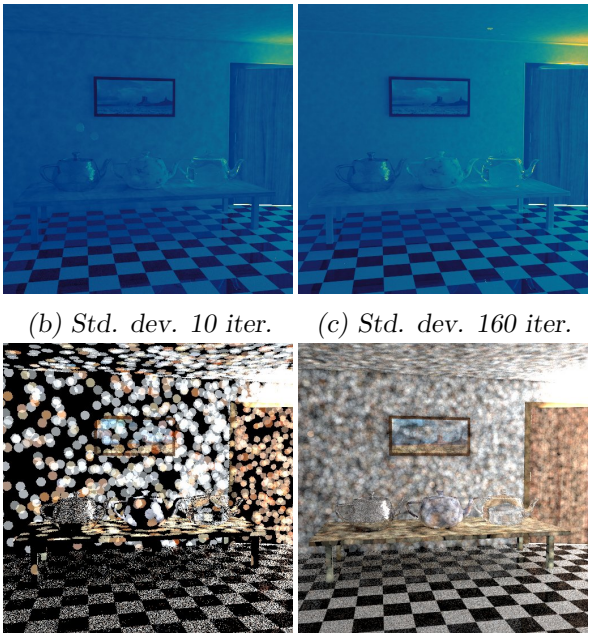
Unlike other rendering algorithms discussed in this work, SPPM is biased but consistent. This means that the output converges to the solution when increasing iteration count, but its expectation is not the solution. In other words, running SPPM 1000 times for 1 iteration and averaging the result is less precise than running it once for 1000 iterations. Therefore we cannot create an ESE and expect all frequencies to behave like $\Theta(1/iter)$.

Instead, we run SPPM with increasing iteration counts, multiply the error curves with the iteration count and study the movement of the ensemble mean. Similar to the scaling that we use to accommodate for varying sampling costs (Section 5.3.3), multiplication *neutralises* the higher quality of more iterations. If the curves are overlapping, it means that the convergence rate is exactly $\Theta(1/iter)$. If the curve with larger iteration count is higher, it means that convergence was slower than $\Theta(1/iter)$, otherwise it was faster. The tails are relatively small, which means that all runs behave approximately the same and problematic outliers were not found.

Our tests show that the convergence rate is not $\Theta(1/iter)$ for all frequencies. In the Door scene (Figure 6.20a) it is slower for mid and high frequencies, while in the Torus scene (Figure 6.21a) it is slower for low frequencies. This is also visible in standard deviation images (scaled by $\sqrt{iter}$). In the Door scene, error moves to the area above the door (Figure 6.21c), while in the Torus scene, it moves more and more into the edges of caustics and shadows (Figure 6.21b).



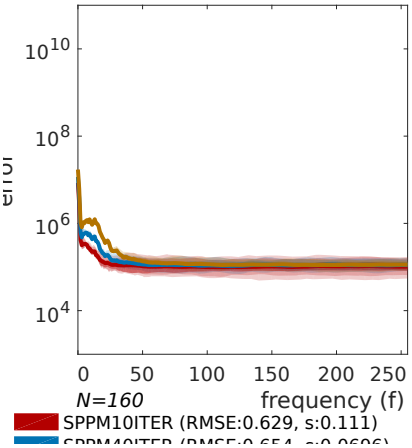
(a) Error Spectrum Ensemble



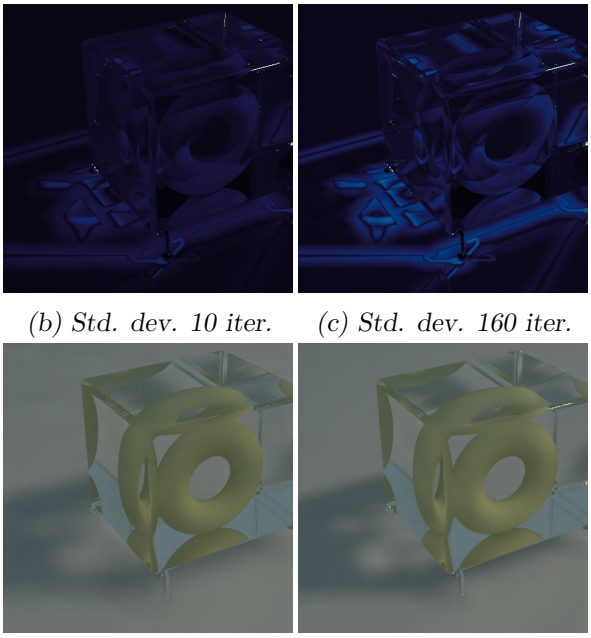
(b) Std. dev. 10 iter. *(c) Std. dev. 160 iter.*

(d) Example 10 iter. *(e) Example 160 iter.*

Figure 6.20: Error descriptors for SPPM on door scene. Error scaled with $\sqrt{iter}$. ESE shows that convergence is slower than $\Theta(1/iter)$ for mid and high frequencies.



(a) Error Spectrum Ensemble



(b) Std. dev. 10 iter. *(c) Std. dev. 160 iter.*

(d) Example 10 iter. *(e) Example 160 iter.*

Figure 6.21: Error descriptors for SPPM on torus scene. Error scaled with $\sqrt{iter}$. ESE shows that convergence is slower than $\Theta(1/iter)$ for low frequencies.

CHAPTER 7

Conclusion

7.1 Synopsis

The aim of this thesis was to improve methods of measuring error in rendering algorithms.

In quantitative comparisons, error is often measured with equal-time MSE or similar metrics. This can be unreliable due to the variance of MSE. We propose to estimate MSE's expectation using a large number of short renderings, and use that for comparisons instead. This is particularly important for algorithms with correlated pixel values, because correlation is directly linked with increased variance.

When comparing error qualitatively, state-of-the-art publications use equal-time renderings. We show that this is a compromise between visualising error behaviour (correlation, PDF) and quantity (per pixel or light effect). The former can be more accurately shown by very short renderings and the latter by standard-deviation-per-pixel images.

In addition to those two, we develop a new tool for investigating the frequency spectrum and outliers of a given algorithm: The Error Spectrum Ensemble (ESE). This descriptor shows the expectation of the error over frequency, the error's variability between observations, its typical behaviour and tendency towards outliers.

In order to validate these tools, understand how they behave and to find limitations, we ran a number of tests. The main findings are:

- MC algorithms generally have equal error expectation in all frequencies while MCMC ones have a peak in low frequencies.
- In many scenes, the majority of MLT observations have a better performance than MC algorithms. However, overall error expectation is worse due to outliers.
- Similarly, BDPT can sample many light effects more efficiently than PT. However, our tests show that pixel outliers are frequent, resulting in a large error expectation.

- Tests with PSSMLT show, that the mutation size of MCMC algorithms is roughly inverse-proportional to the error-peak width.
- MLT algorithm parameters like number of luminance samples and chain length (number of chains) have an influence on performance and behaviour.
- When testing SPPM, we found that its asymptotic convergence rates in low and high frequencies are different (unlike unbiased methods).

7.2 Discussion

When developing rendering algorithms, two things are important:

1. Does the change improve the expected rendering quality and how much?
2. How does the change influence characteristics of the error and what is the logic behind?

We improved the methods that are used to answer those questions.

Comparing MSE expectation is more accurate than MSE observations, which is especially important for algorithms with correlated pixel values. In addition, it is possible to estimate its variance, which would make confidence intervals possible.

As for characteristics, we came to the conclusion that it is important to look at error from 3 different angles:

- Standard-deviation images show the quantity and location of the error. This shows the performance depending on lighting effects, geometric features, etc. However, when converged, it tells nothing about frequency content, outliers or correlation.
- Particular short time renderings show type of noise and correlation. However, they are not a good way to measure error quantity even when using longer rendering times.
- Finally, ESE shows the frequency content, type of and tendency towards outliers.

7.3 Future work

Currently, the descriptor does not show whether the tail is long due to a single very large outlier, or several smaller ones. This might be important if dealing with outlier-filtering techniques. There are certainly also other directions on improving the visualisation.

We only look at final renderings. If integrated into a rendering system, it would be possible to get more fine-grained information for error analysis. Interesting questions could be:

- Which lighting effect (light path / number of bounces) produced the error?
- Which MCMC mutations are responsible for error?
- What is the mixing coefficient of a Markov chain, which mutations are responsible for mixing and how does it affect the error?

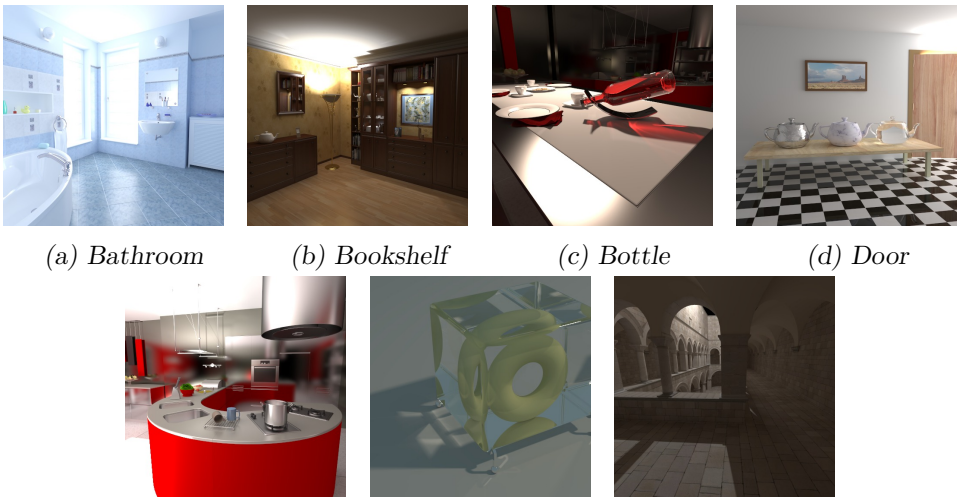
Our descriptor needs a high-quality reference image. Online methods, e.g., without reference, could be useful to adapt rendering parameters to the scene at hand (MCMC mutation size etc.).

Finally, in this work we completely ignored the error perception of the human visual system. It might very well be that rendering error in certain frequencies is more bothering than in others.

APPENDIX A

Results for Complex Scenes

Reference images are visible in Figure A.1, the colour map of standard deviation images in Figure A.2 with $r = 20$ in case of absolute error, $r = 40$ for relative error and $r = 5$ respectively 10 for the Sponza scene.



(a) Bathroom *(b) Bookshelf* *(c) Bottle* *(d) Door*

(e) Kitchen *(f) Torus* *(g) Sponza*

Figure A.1: Reference images for complex scenes.

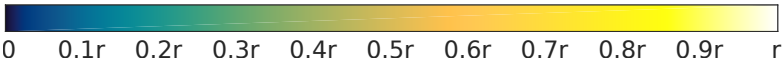
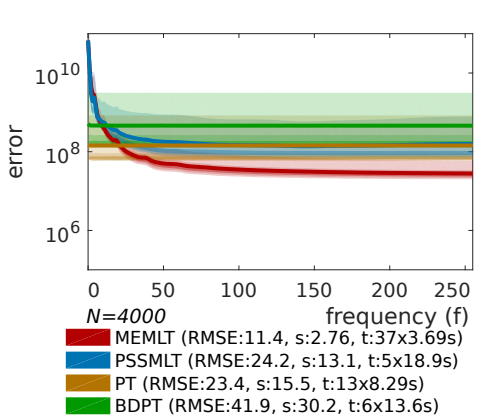
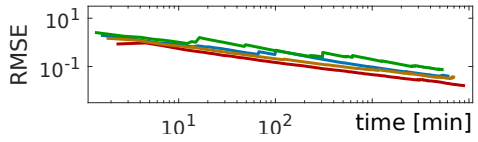


Figure A.2: Colour map for standard deviation pictures, see text for values of r.



(a) Ensemble power spectrum

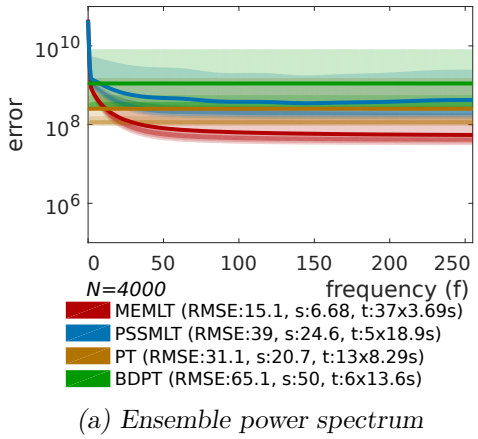


(b) RMSE over time

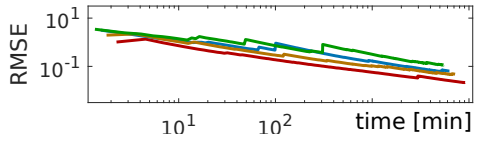
(c) MEMLT (d) PSSMLT

(e) PT (f) BDPT

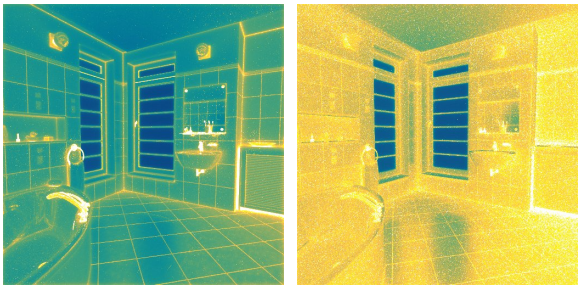
Figure A.3: Bathroom scene, absolute error. There are no MEMLT outliers, we think well suited for bidirectional mutations. BDPT is worse than PT, MEMLT is best although it is visible, that there is additional error on edges.



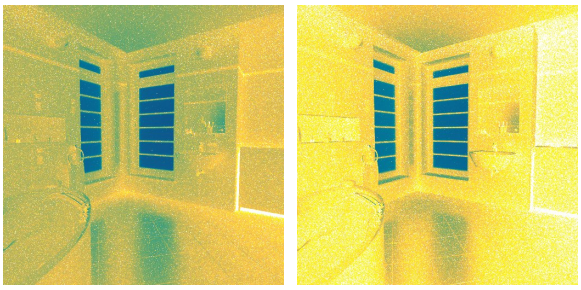
(a) Ensemble power spectrum



(b) RMSE over time

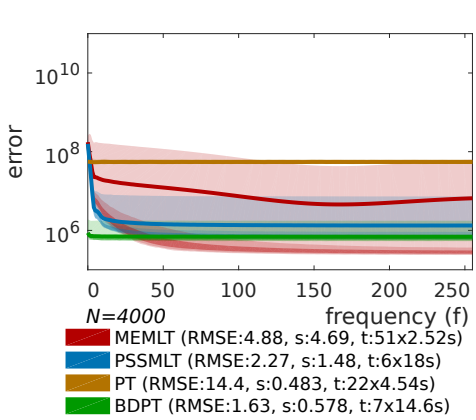


(c) MEMLT (d) PSSMLT

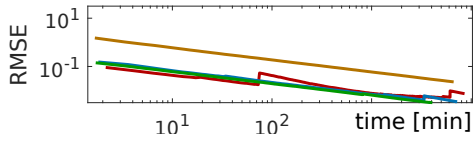


(e) PT (f) BDPT

Figure A.4: Bathroom scene, relative error.



(a) *Ensemble power spectrum*



(b) *RMSE over time*

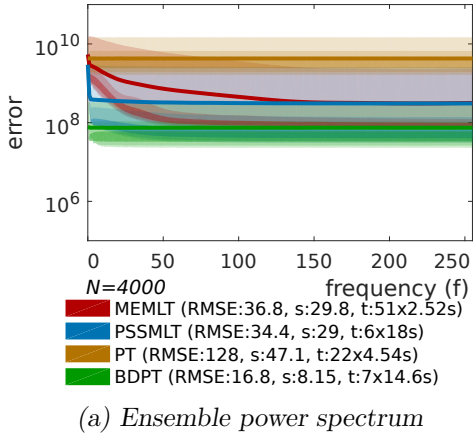
(c) *MEMLT*

(d) *PSSMLT*

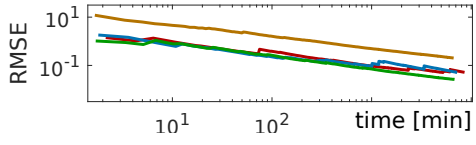
(e) *PT*

(f) *BDPT*

Figure A.5: Bookshelf scene, absolute error. This scene is very difficult for PT, probably because many lights are obscured, hidden behind glass in the shelf and direct light sampling does not work. All other algorithms have problems with the small glossy objects inside the shelf.

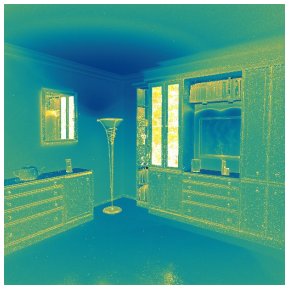


(a) *Ensemble power spectrum*

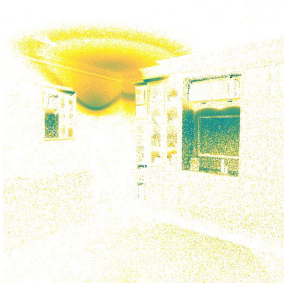


(b) *RMSE over time*

(c) *MEMLT*



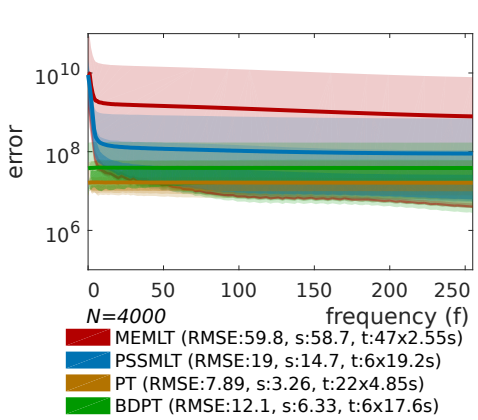
(d) *PSSMLT*



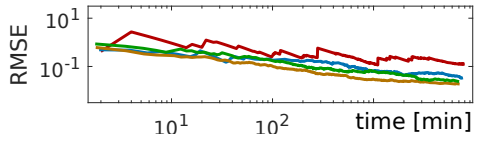
(e) *PT*

(f) *BDPT*

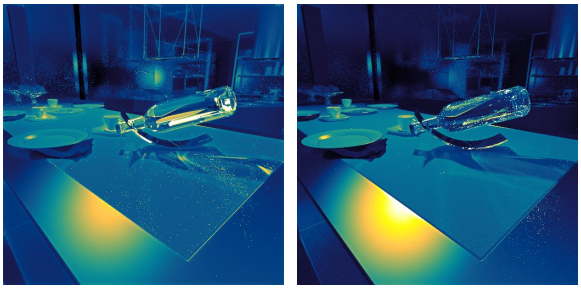
Figure A.6: Bookshelf scene, relative error.



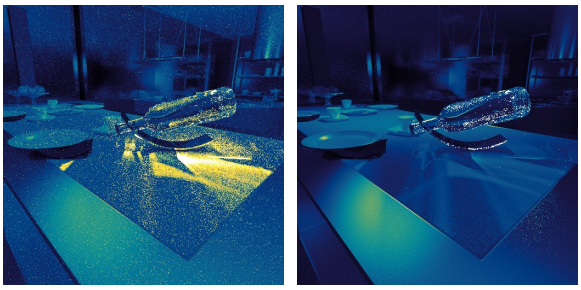
(a) Ensemble power spectrum



(b) RMSE over time

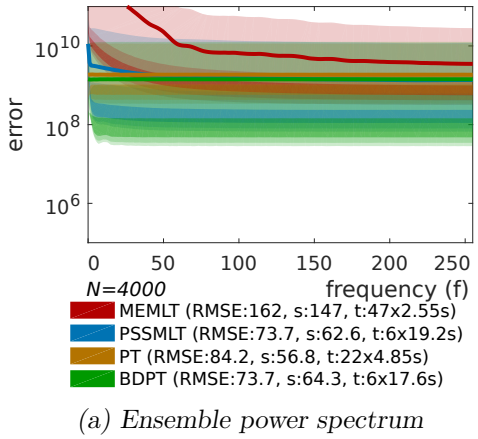


(c) MEMLT (d) PSSMLT

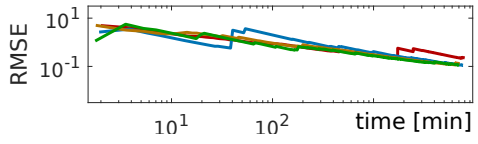


(e) PT (f) BDPT

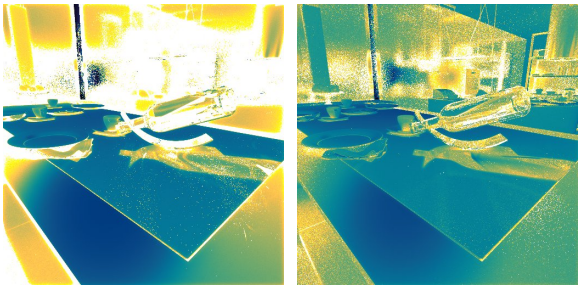
Figure A.7: Bottle scene, absolute error. It is very hard due to depth of refractions. ESE shows that all algorithms produce pixel outliers, largest one in MEMLT. This might be a bit surprising because manifold exploration is made for specular chains. We speculate that bad performance comes from suboptimal large mutation.



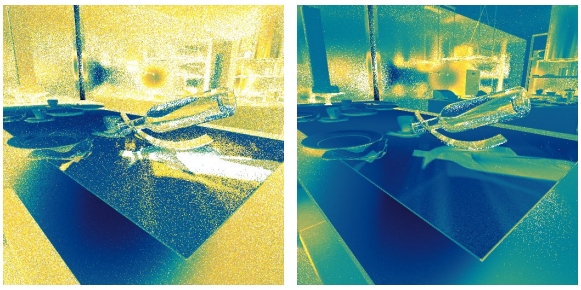
(a) Ensemble power spectrum



(b) RMSE over time

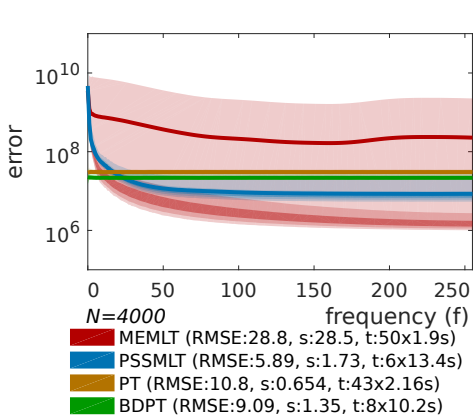


(c) MEMLT (d) PSSMLT

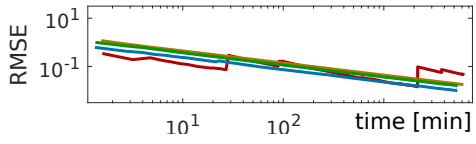


(e) PT (f) BDPT

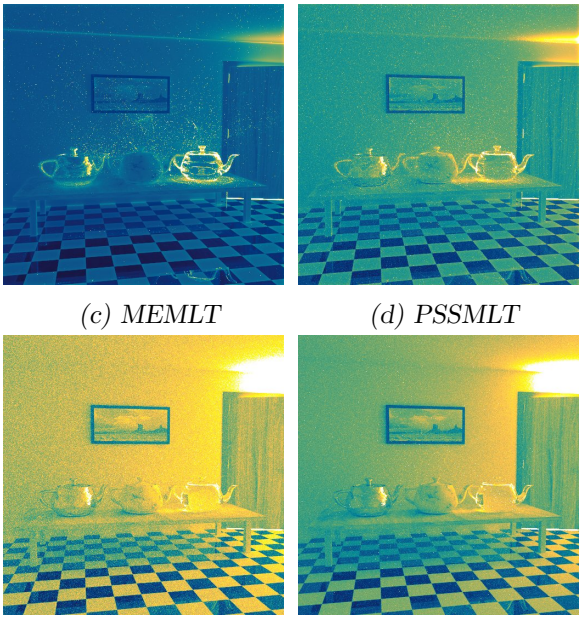
Figure A.8: Bottle scene, relative error.



(a) Ensemble power spectrum



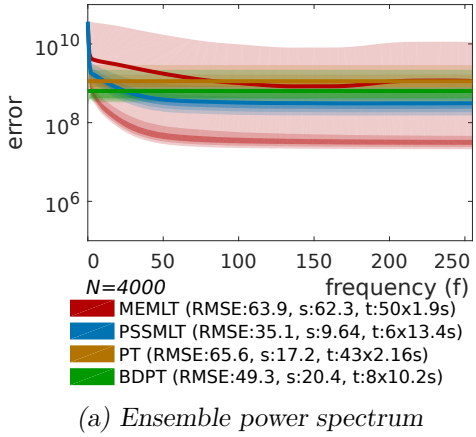
(b) RMSE over time



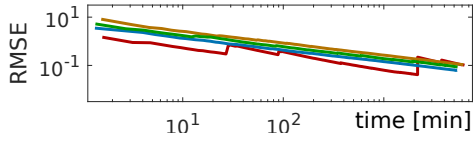
(c) MEMLT (d) PSSMLT

(e) PT (f) BDPT

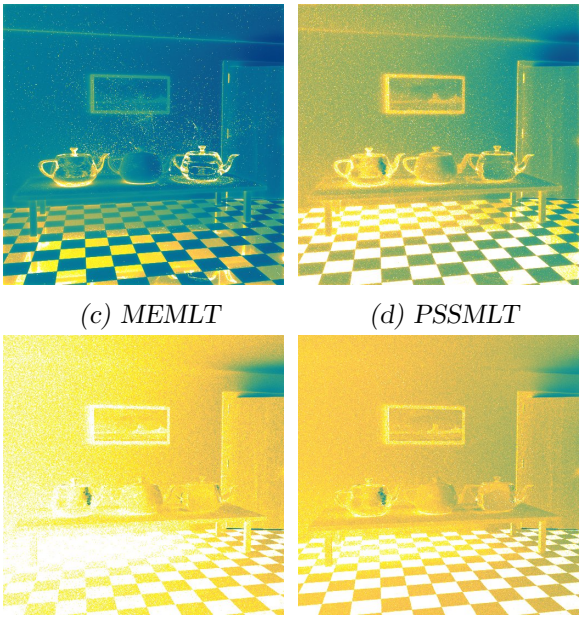
Figure A.9: Door scene, absolute error. It is a bit surprising that MEMLT performs worst (due to outliers), as it was used as a reference scene for path space MLT. Since head and body are both bottom most, it could be that the outliers were just not caught *in the original paper.*



(a) Ensemble power spectrum



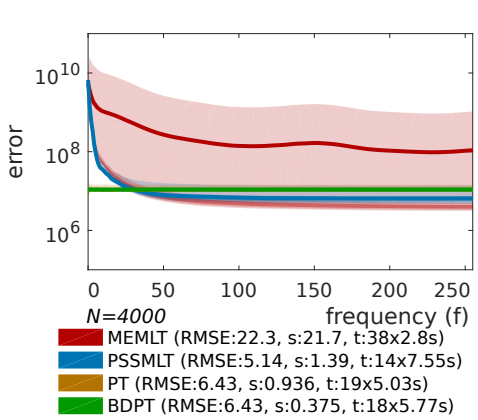
(b) RMSE over time



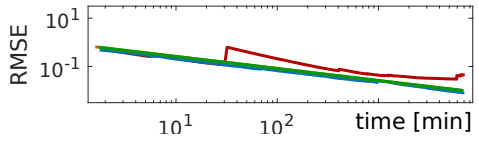
(c) MEMLT (d) PSSMLT

(e) PT (f) BDPT

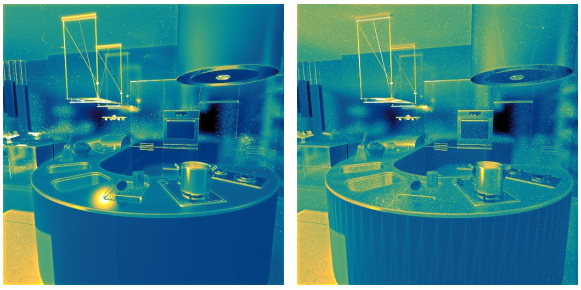
Figure A.10: Door scene, relative error.



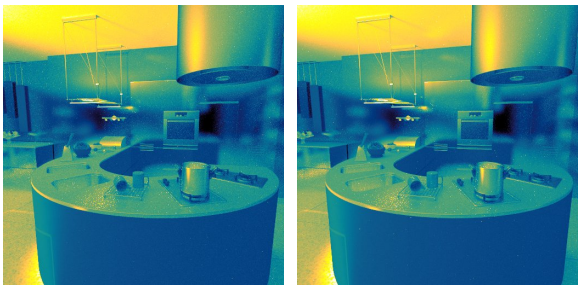
(a) Ensemble power spectrum



(b) RMSE over time

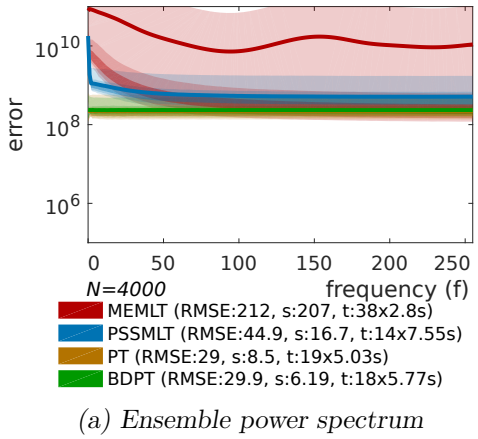


(c) MEMLT (d) PSSMLT

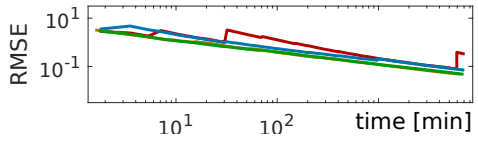


(e) PT (f) BDPT

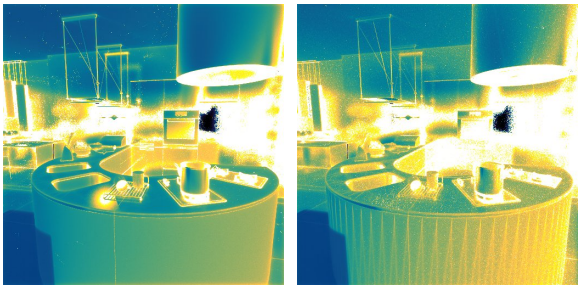
Figure A.11: Kitchen scene, absolute error. PSSMLT has artefacts on triangle edges of the red counter, the reason is unclear to us, it might be an implementation bug. Normals are treated equally in BDPT and PT (strict normals on). MEMLT has large error next to the dish rack on the counter (c). (d) to (f) did not converge at that place.



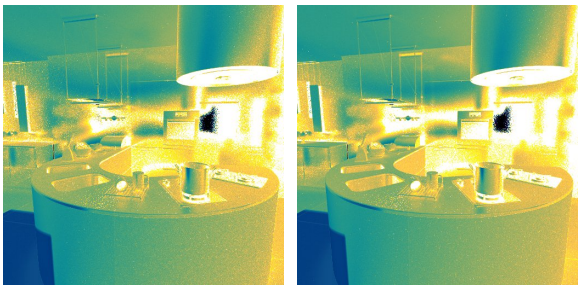
(a) Ensemble power spectrum



(b) RMSE over time

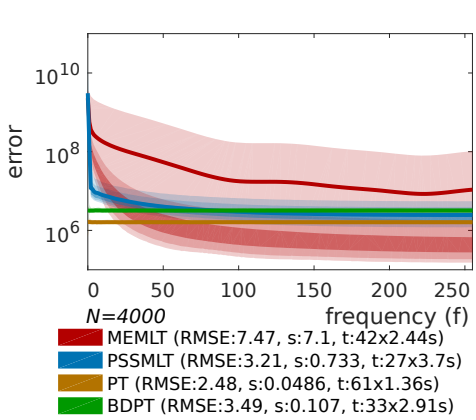


(c) MEMLT (d) PSSMLT

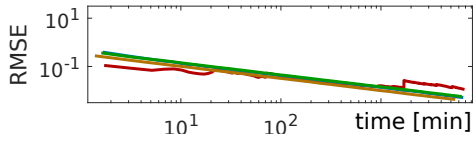


(e) PT (f) BDPT

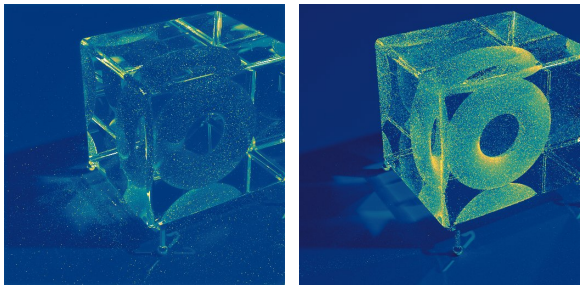
Figure A.12: Kitchen scene, relative error.



(a) *Ensemble power spectrum*

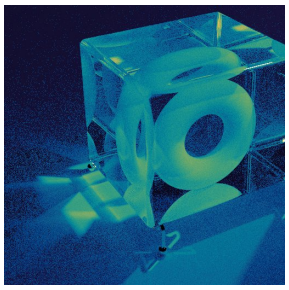


(b) *RMSE over time*

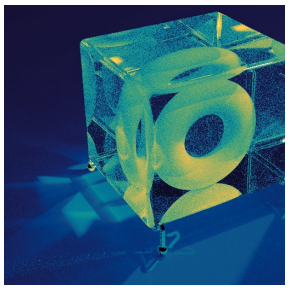


(c) *MEMLT*

(d) *PSSMLT*

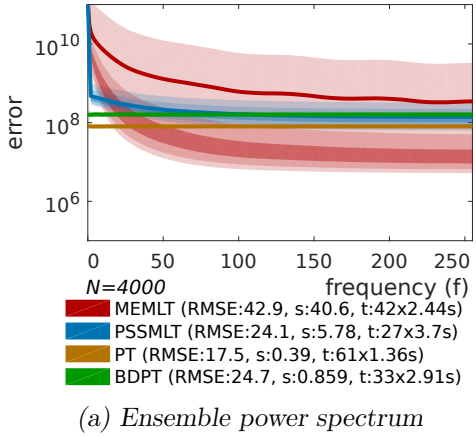


(e) *PT*

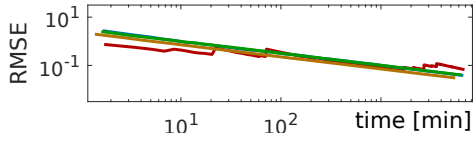


(f) *BDPT*

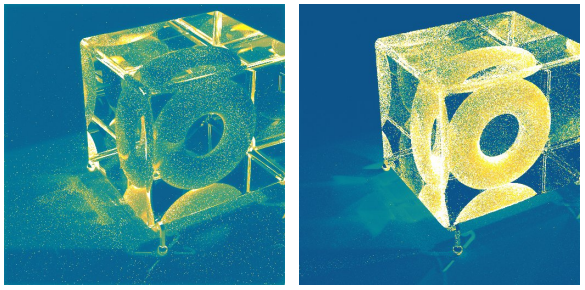
Figure A.13: Torus scene, absolute error. MEMLT shows almost no error on the torus, but a lot of error in deep refractions. Just as with the bottle we suspect poor performance of global mutations. MEMLT is worst due to outliers, PT is the best although there is a lot of error in the caustic.



(a) *Ensemble power spectrum*

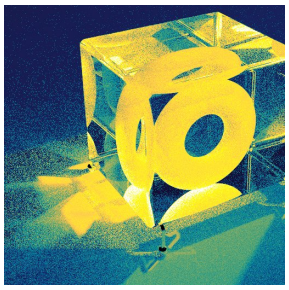


(b) *RMSE over time*



(c) *MEMLT*

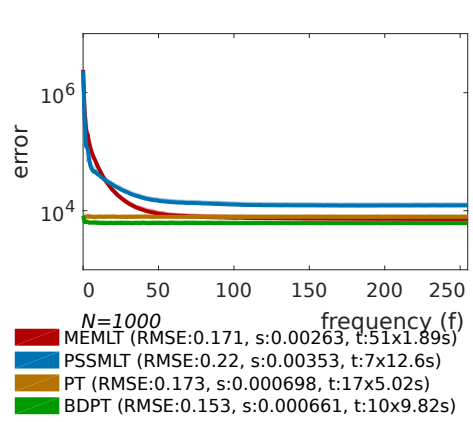
(d) *PSSMLT*



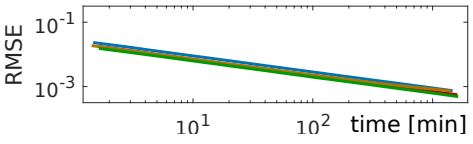
(e) *PT*

(f) *BDPT*

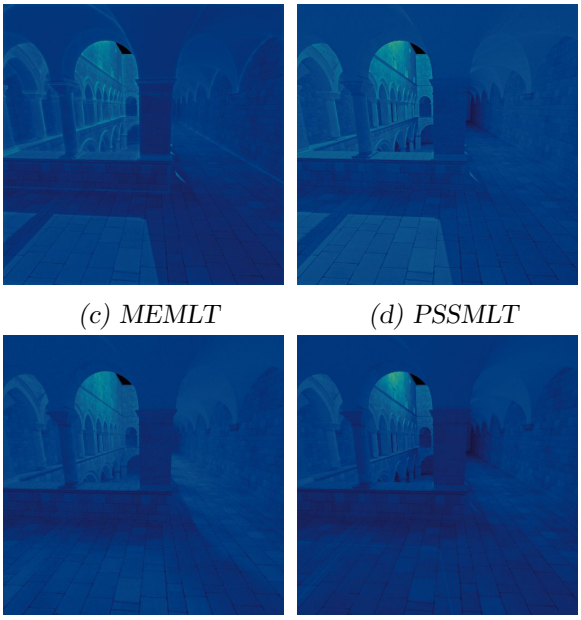
Figure A.14: Torus scene, relative error.



(a) *Ensemble power spectrum*



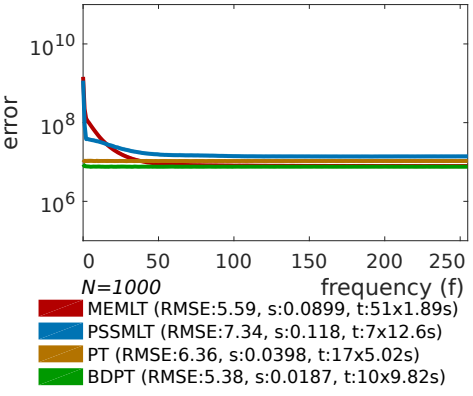
(b) *RMSE over time*



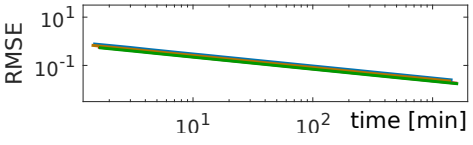
(c) *MEMLT* (d) *PSSMLT*

(e) *PT* (f) *BDPT*

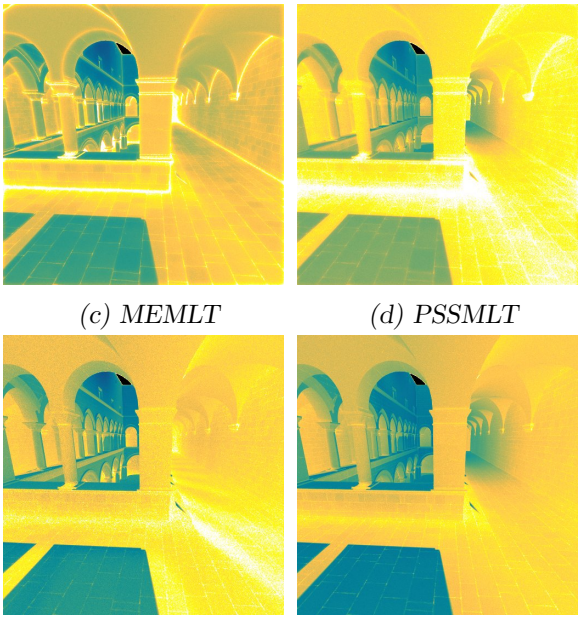
*Figure A.15: Sponza scene, absolute error (different ESE scale, std. dev. white >= **5**). Not much surprises, directly lit areas on the floor have larger MLT error (like in the Kitchen scene). MEMLT shows tendency to error on edges (like in Kitchen, Door and Bathroom).*



(a) *Ensemble power spectrum*



(b) *RMSE over time*



(c) *MEMLT* (d) *PSSMLT*

(e) *PT* (f) *BDPT*

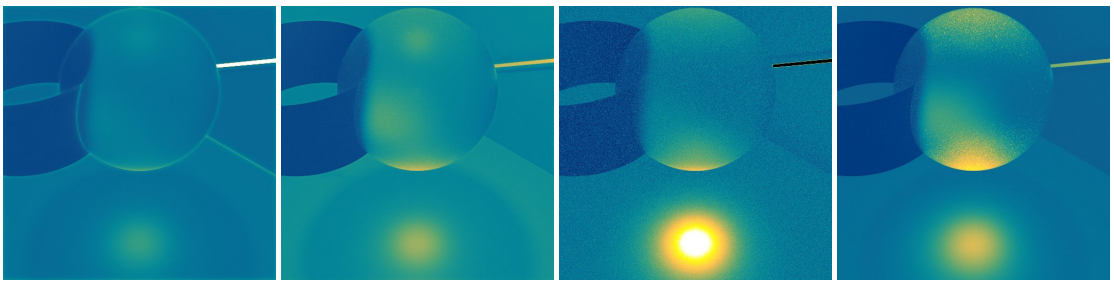
*Figure A.16: Sponza scene, relative error (different ESE scale, std. dev. white >= **10**).*

APPENDIX B

Additional Test Results

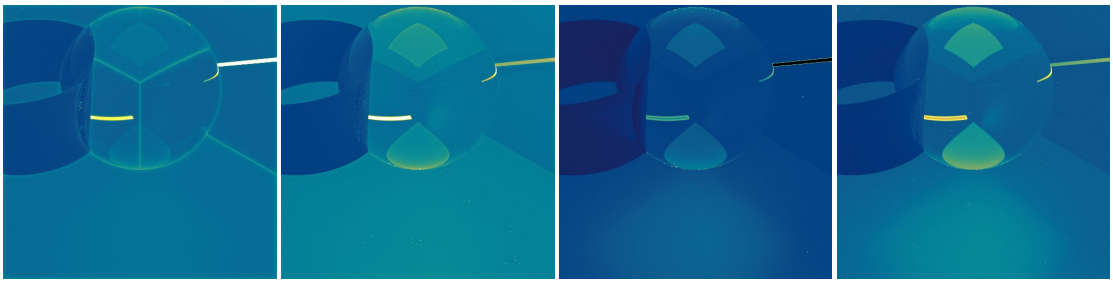
B.1 Standard deviation and example renderings for box scene with various parameters

One more standard deviation image is in Figure 6.3.



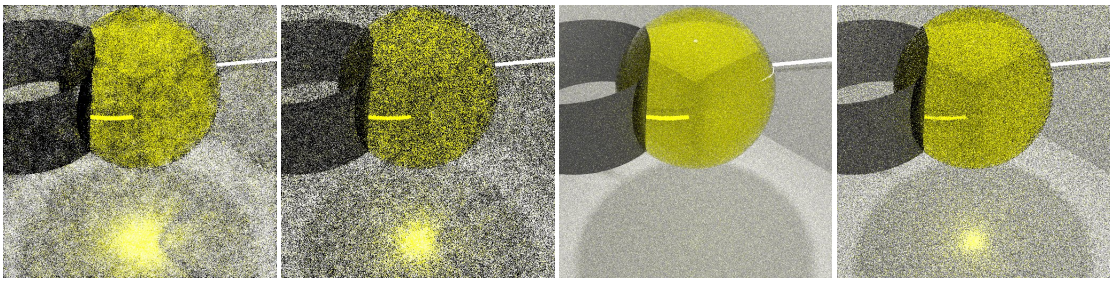
(a) MEMLT *(b) PSSMLT* *(c) PT* *(d) BDPT*

Figure B.1: Standard deviation per pixel for box scene with roughness 1/4*. Light size is 50.*



(a) *MEMLT* (b) *PSSMLT* (c) *PT* (d) *BDPT*

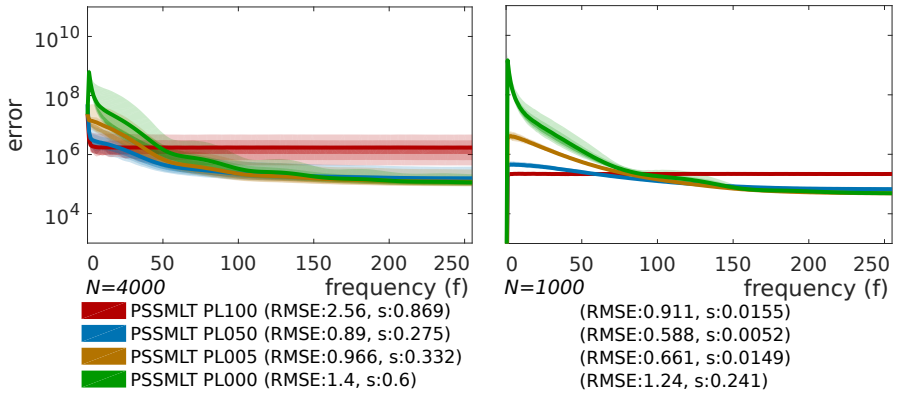
Figure B.2: Standard deviation per pixel for box scene with light size 400. *Roughness is* 1/256.



(a) *MEMLT* (b) *PSSMLT* (c) *PT* (d) *BDPT*

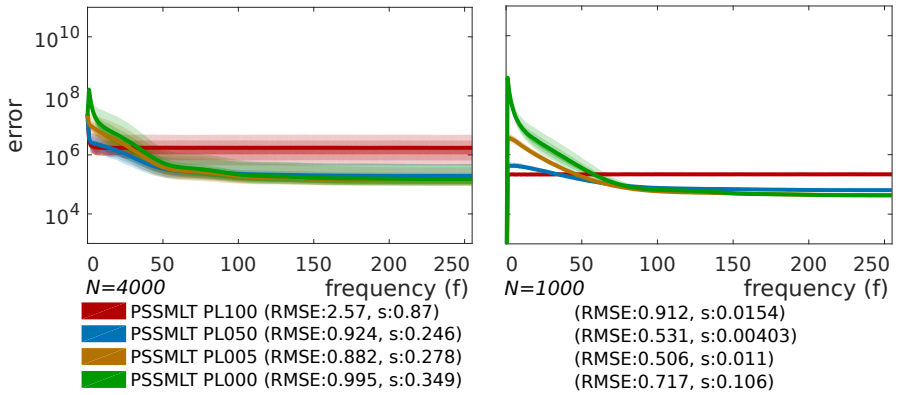
Figure B.3: Example renderings for box scene with light size 25, *roughness is* 1/256.

B.2 ESE for changing percentage of large mutation in PSSMLT algorithms



(a) PSSMLT *(b) Fake MLT*

Figure B.4: ESE for changing percentage of large mutations, small mutations 50% of default value. Box scene with roughness 1/256 *and light size* 50.

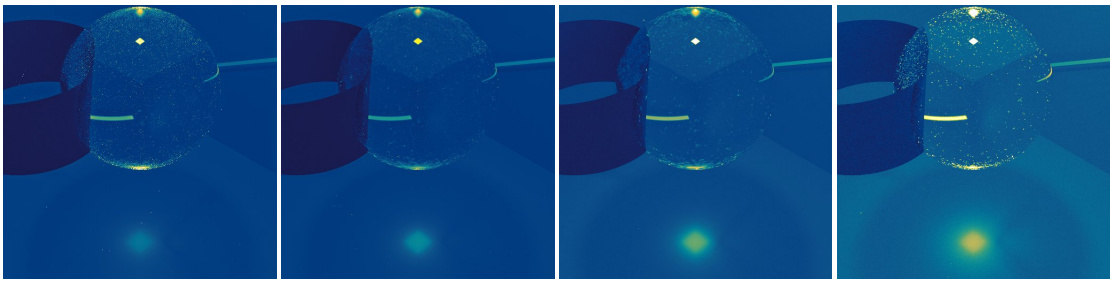


(a) PSSMLT *(b) Fake MLT*

Figure B.5: ESE for changing percentage of large mutations, default sized small mutations. Box scene with roughness 1/256 *and light size* 50.

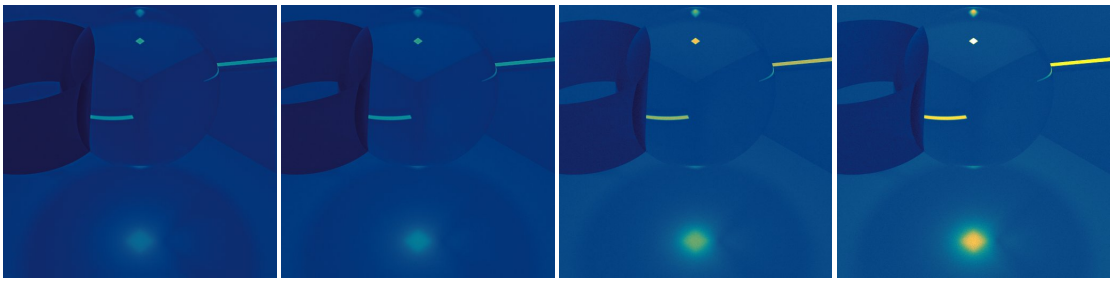
B.3 Standard deviation and example renderings for changing size of small PSSMLT and fake algorithm mutations

The percentage of large mutations is 30%.



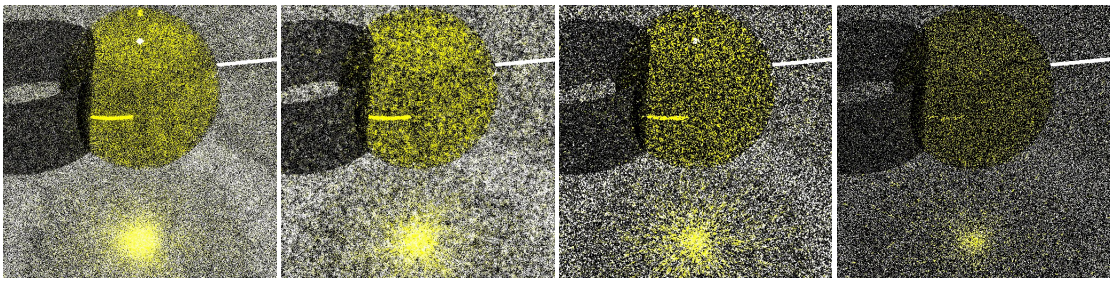
(a) 200% mutation size *(b) 50% mutation size* *(c) 10% mutation size* *(d) 1% mutation size*

Figure B.6: Standard deviation for changing small mutation size.



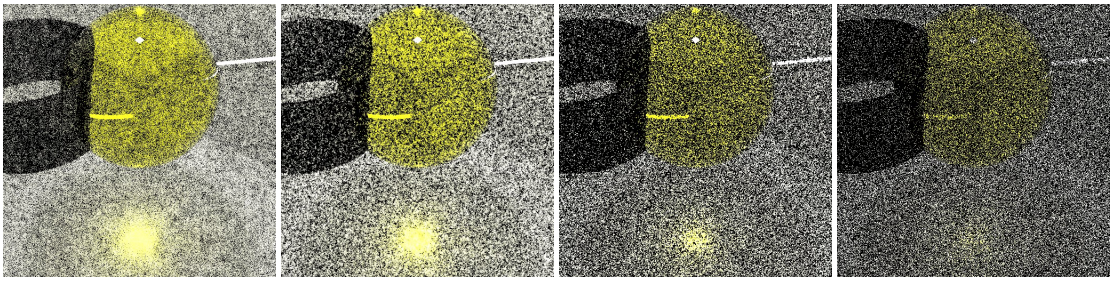
(a) 200% mutation size *(b) 50% mutation size* *(c) 10% mutation size* *(d) 1% mutation size*

Figure B.7: Standard deviation for changing small mutation size (fake algorithm).



(a) 200% mutation size *(b) 50% mutation size* *(c) 10% mutation size* *(d) 1% mutation size*

Figure B.8: Example renderings for changing small mutation size.

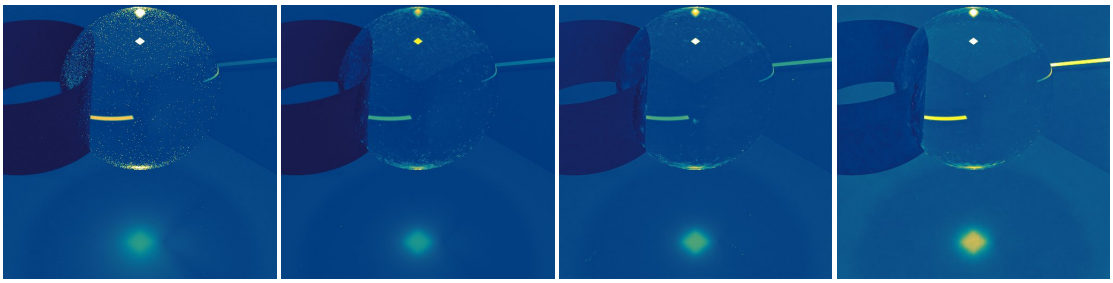


(a) 200% mutation size (b) 50% mutation size (c) 10% mutation size (d) 1% mutation size

Figure B.9: Example renderings for changing small mutation size (fake algorithm).

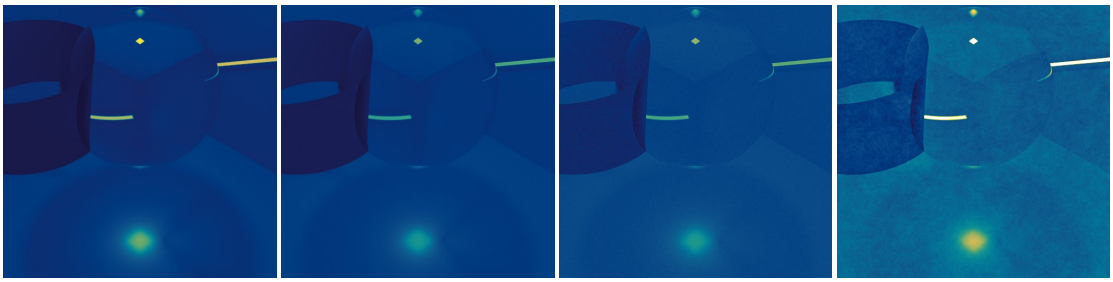
B.4 Standard deviation and example renderings for changing percentage of large mutations in PSSMLT and fake algorithm

Small mutations are 25% of the default size.



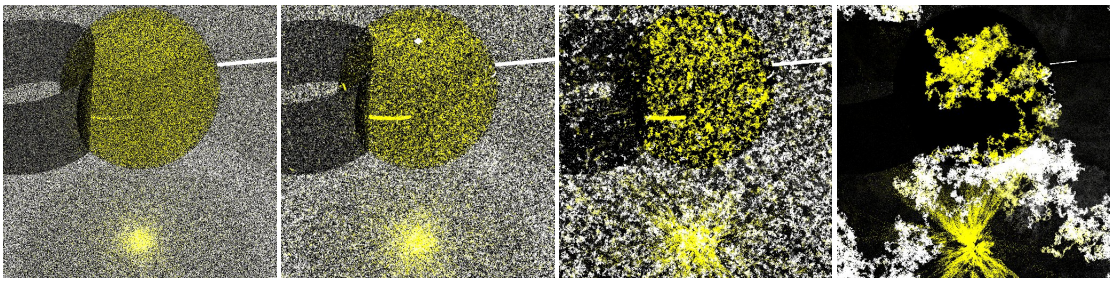
(a) 100% large (b) 50% large (c) 5% large (d) 0% large

Figure B.10: Example renderings for changing percentage of large mutations.



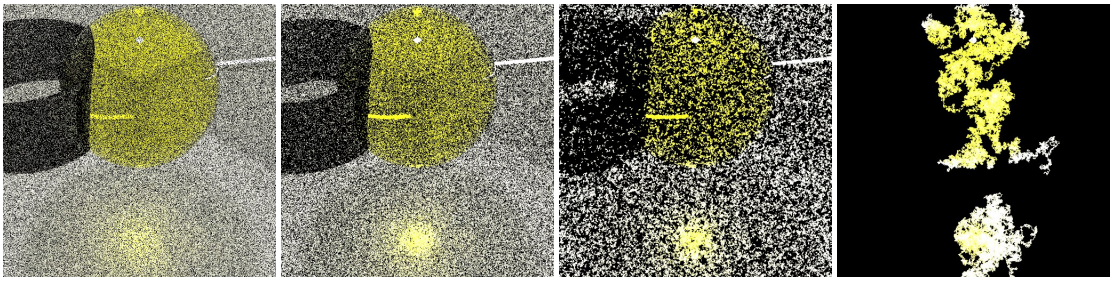
(a) 100% large (b) 50% large (c) 5% large (d) 0% large

Figure B.11: Example renderings for changing percentage of large mutations (fake algorithm).



(a) 100% large (b) 50% large (c) 5% large (d) 0% large

Figure B.12: Example renderings for changing percentage of large mutations.

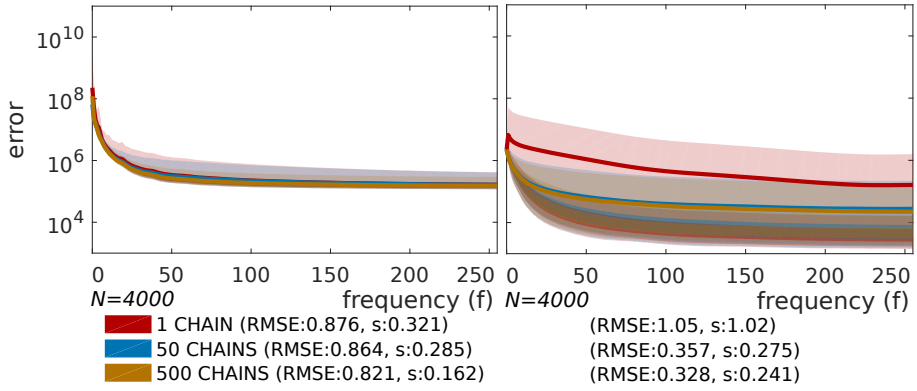


(a) 100% large *(b) 50% large* *(c) 5% large* *(d) 0% large*

Figure B.13: Example renderings for changing percentage of large mutations (fake algorithm).

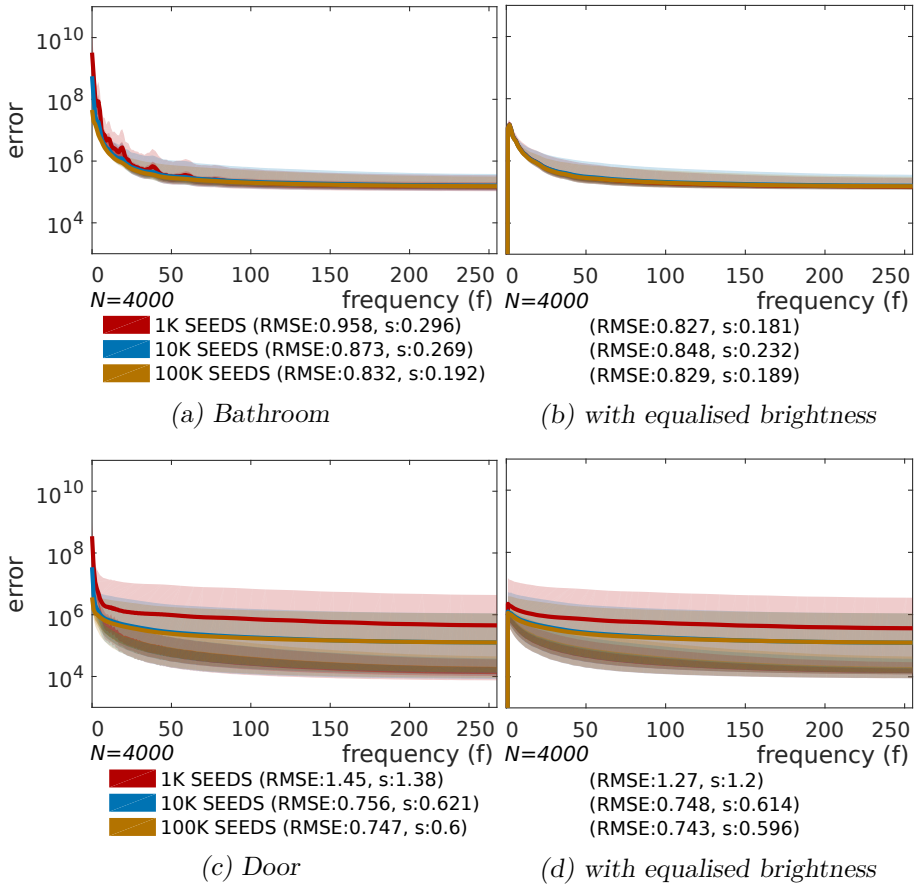
B.5 MLT seeding and chain length configurations

Here we present ESEs of tests not shown in Section 6.4. All tests have 50 samples per pixel on average. If nothing else is written there are 50 chains, all sampled from a single seed pool. The glass roughness parameter of the box scene is set to 1/256 and light size to 50. Brightness equalisation means, that the image was scaled to match the average luminance of the reference.



(a) Bathroom *(b) Torus*

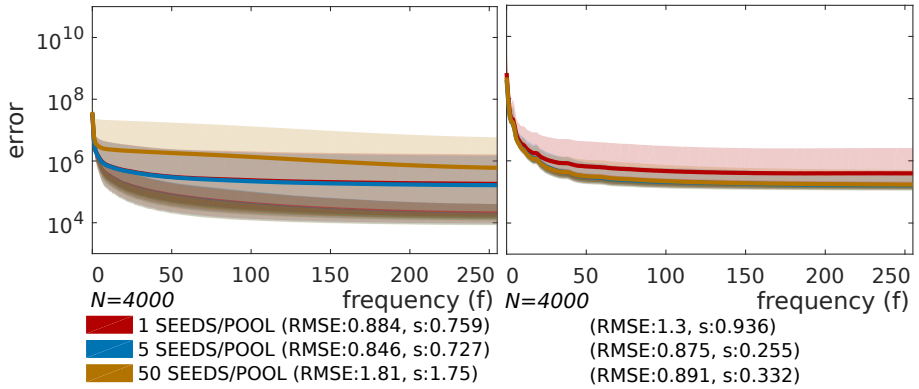
Figure B.14: More ESEs for varying chain lengths and 100k seed pool size. We believe that the acceptance rate of mutations in those two scenes is higher, therefore they benefit less from a restart. See also Figure 6.18.



(a) Bathroom *(b) with equalised brightness*

(c) Door *(d) with equalised brightness*

Figure B.15: More ESEs for varying size of the seed pool / number of luminance samples. The advantage that can be gained by increasing the pool size depends on the scene. Note however, that RMSE is strictly monotone falling in (a) and (c). See also Figure 6.15



(a) Door *(b) Bathroom*

Figure B.16: More ESE for various partion variants of the luminance samples into seed pools. Partitioning plays only a minor role. See also Figure 6.17.

APPENDIX C

Statistics

In this appendix we show additional figures and findings to Chapter 4.

C.1 Correlation between real and imaginary part

If an outlier lies in the same small area in many observations, then there can be correlation between real and imaginary part. Samples are distributed close to their axis going through the origin in the complex plane, as shown in Figure C.2b. The orientation depends on the spatial position of the outlier area and more specifically on the ratio of sine and cosine values of the corresponding Fourier frequency at that position (Figure C.1). It is possible, that there are several outlier areas, in which case samples are distributed around more than just one axis.

We found that in our experiments this type of correlation was mostly found in low frequencies of MLT algorithms. This can be observed for instance in Figure C.3, it shows the correlation coefficients between real and imaginary part for a typical scene when rendering using MLT and PT. The same can be seen in the phase histogram (Figure C.8) and mean phase image (Figure C.12).

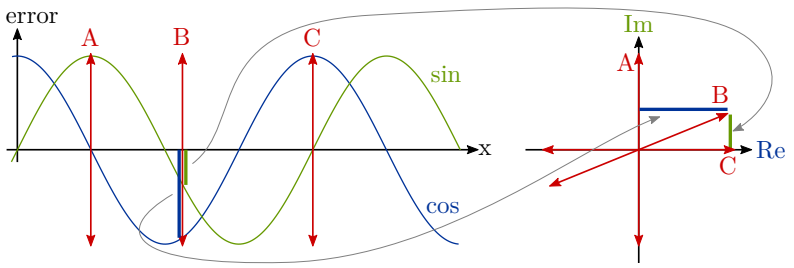
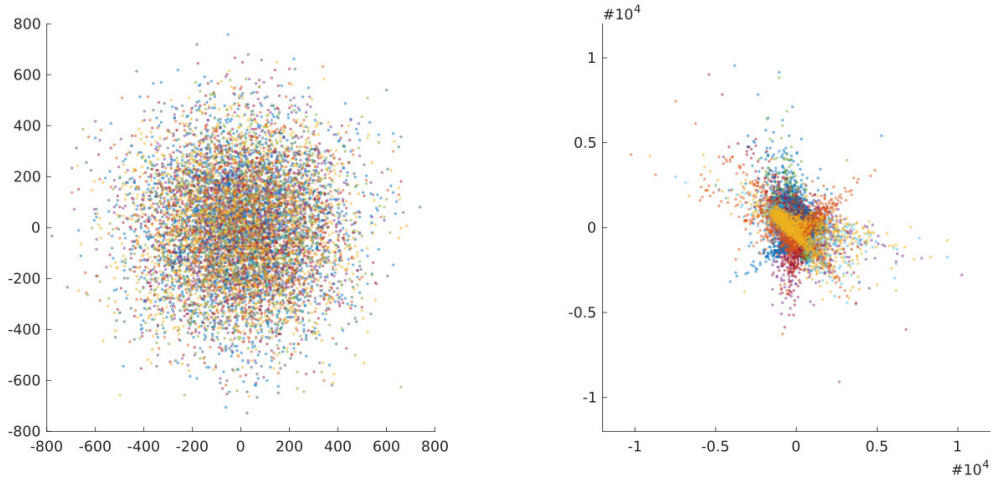
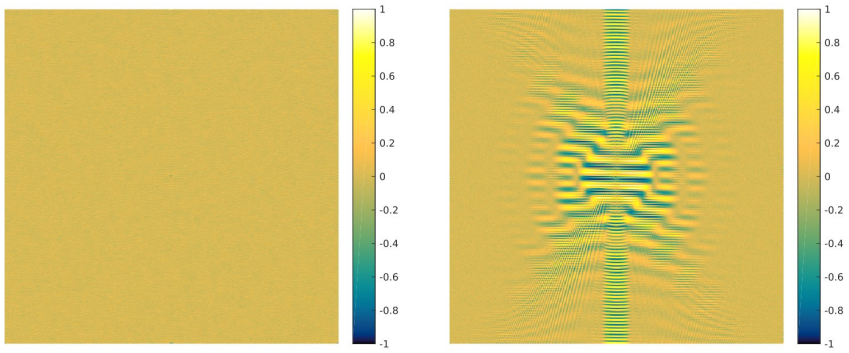


Figure C.1: Dependence between real and imaginary part in the Fourier Spectrum. The left shows the spatial domain, sin *and* cos *waves are weights of the Fourier sum resulting in the Fourier coefficient of a specific frequency. The right shows the complex plane of that frequency. A-C are 3 independent examples of dominating error. Assuming the rest of the error signal in the spatial domain is zero, position A, respectively B and C, produce values on the corresponding axes in the complex plane. For instance: If outliers are concentrated at area A and a specific Fourier frequency has the sine and cosine waves given, then error observations will be along the imaginary axis.*



(a) Path tracing *(b) MLT*

Figure C.2: Correlation between real and imaginary part of the error spectrum. X and Y axis are real and imaginary part, respectively. Every colour belongs to a different frequency (selected randomly in [1,10]), every data point to a different rendering. (a) shows strong correlation, (b) not. The data is taken from renderings with 100 samples per pixel.

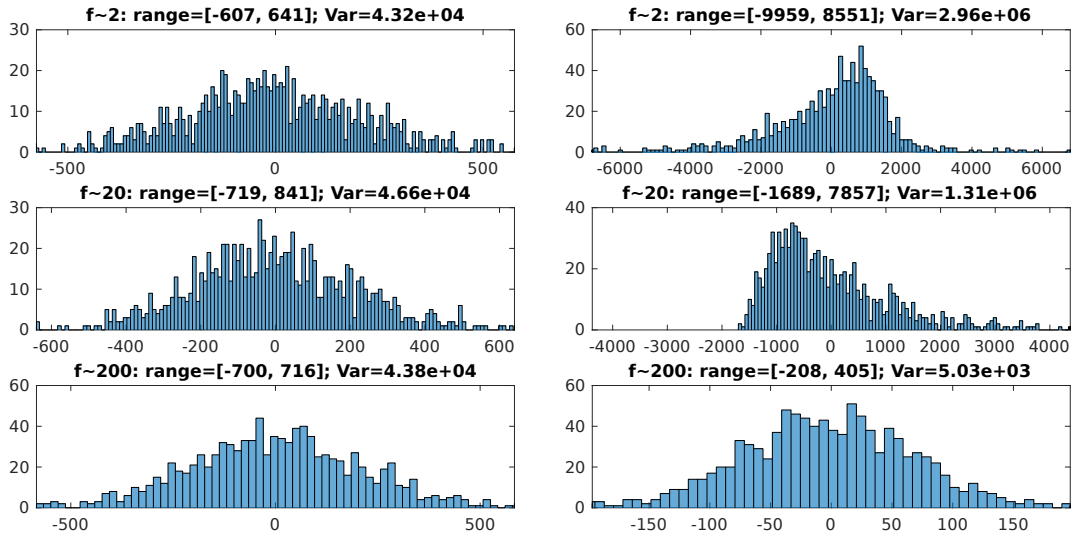


(a) Path tracing *(b) MLT*

Figure C.3: Correlation between real and imaginary DFT part, same data as in Figure C.2. Most correlation happens in low frequencies of MLT renderings.

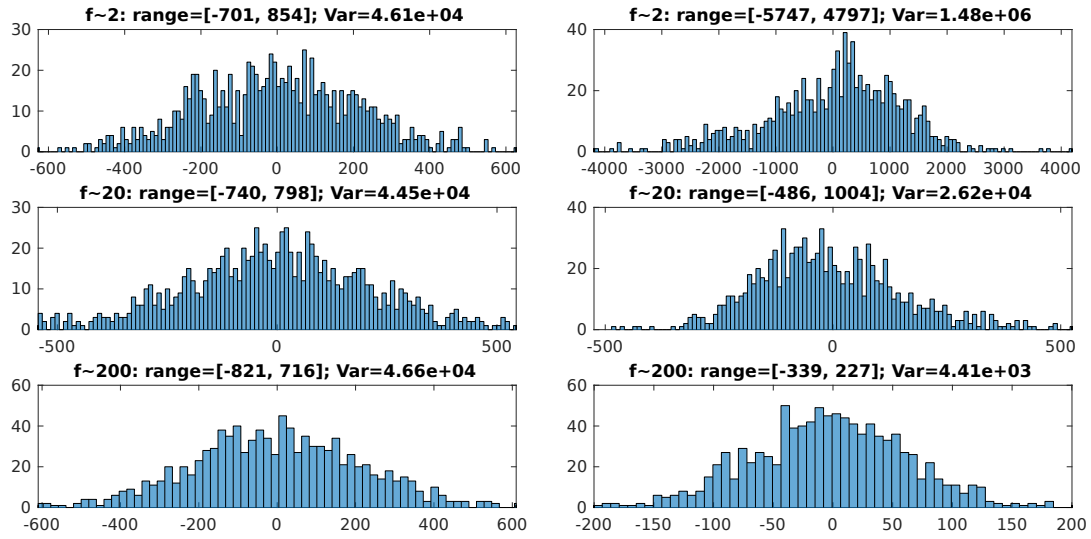
C.2 Histograms of Fourier frequencies

Three representative frequencies were selected (0/1, 8/3 and 93/183) which correspond to plane waves with approximately one, 10 and 200 periods.



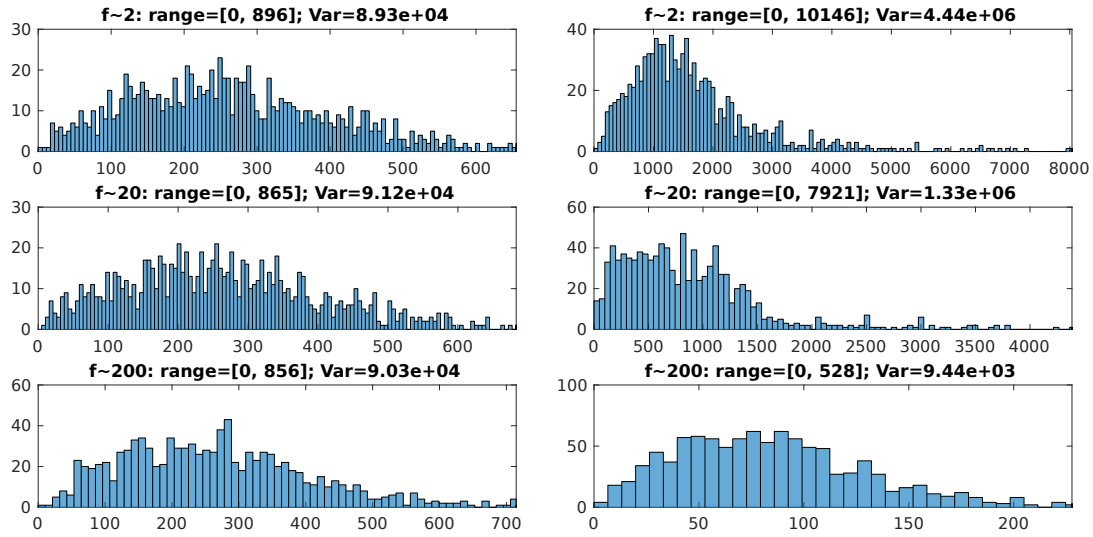
(a) Path tracing *(b) MLT*

*Figure C.4: Fourier error distribution for different frequencies (**real** part)*



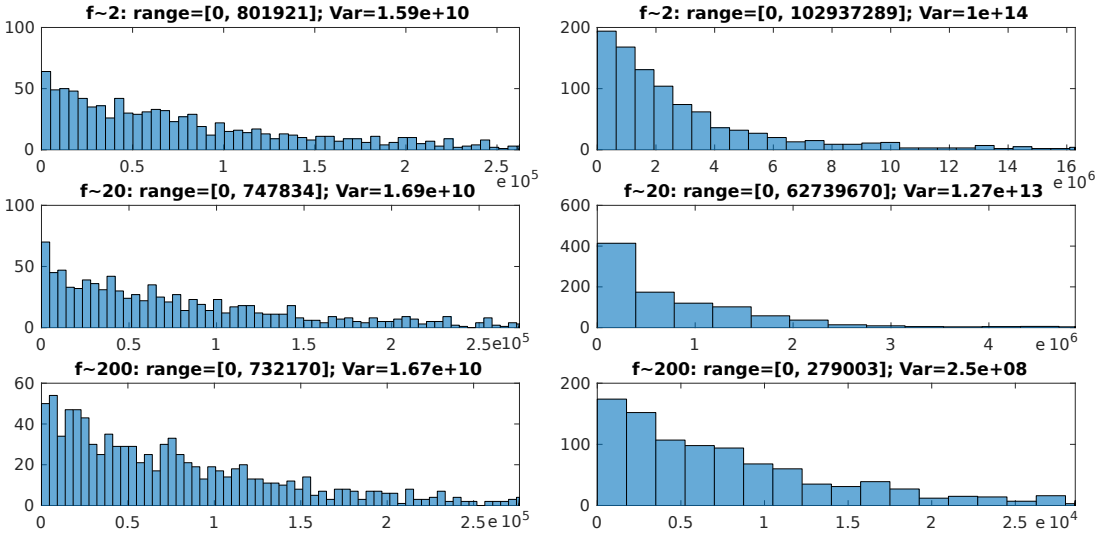
(a) Path tracing *(b) MLT*

*Figure C.5: Fourier error distribution for different frequencies (**imaginary** part)*



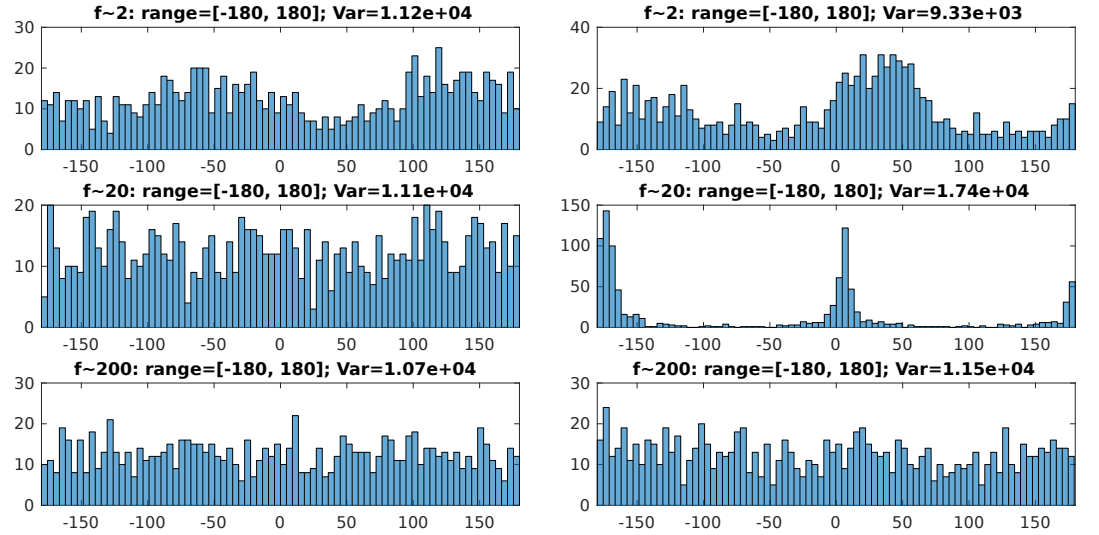
(a) Path tracing *(b) MLT*

*Figure C.6: Fourier error distribution for different frequencies (**magnitude**)*



(a) Path tracing *(b) MLT*

*Figure C.7: Fourier error distribution for different frequencies (**power** (magnitude squared)). Range clipped for MLT due to long tails.*

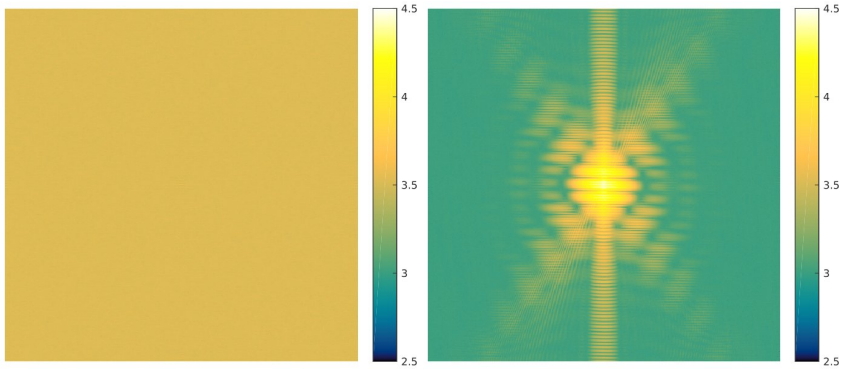


(a) Path tracing *(b) MLT*

*Figure C.8: Fourier error distribution for different frequencies (**phase**)*

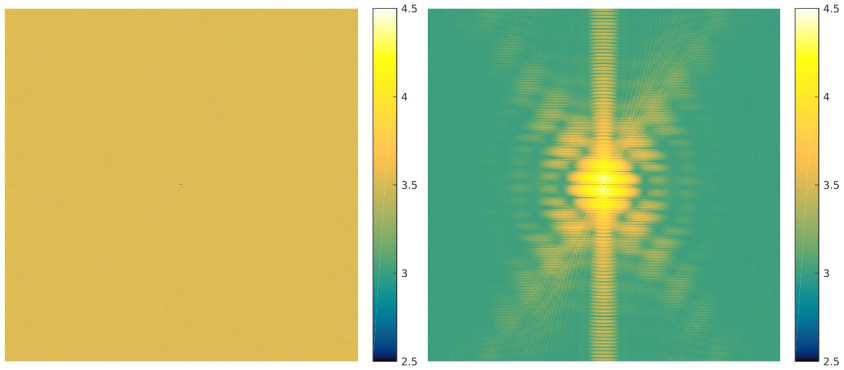
C.3 Full discrete Fourier spectra

The next plots show data of the whole Fourier spectrum, low frequencies are in the centre. The shape of the centre in MLT obviously depends on the scene and parameters, but PSSMLT and MLT are roughly the same.



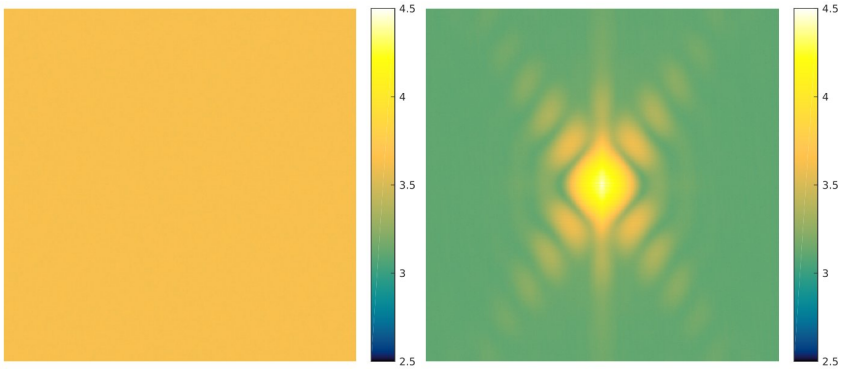
(a) Path tracing *(b) MLT*

*Figure C.9: Standard deviation of **real** Fourier part (log10 domain)*



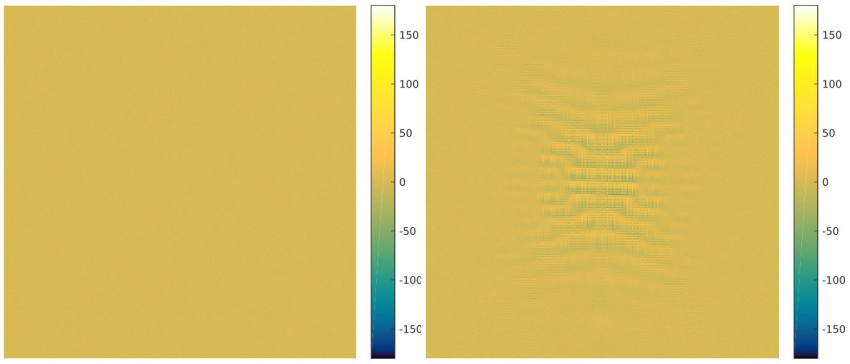
(a) Path tracing *(b) MLT*

*Figure C.10: Standard deviation of **imaginary** Fourier part (log10 domain)*



(a) Path tracing *(b) MLT*

*Figure C.11: Mean **amplitude** spectrum (log10 domain)*



(a) Path tracing *(b) MLT*

*Figure C.12: Mean **phase** of Fourier error spectrum (-180 to 180 degrees)*

APPENDIX D

Pitfalls

D.1 Outliers

Outliers are some times very hard to *catch*. For instance in some cases we saw outliers that were more than 2000 times the mean and occurred only in one out of 1000 renderings.

If outliers are simply clamped away, the algorithm becomes biased.

D.2 Bias of implementation

Implementations of otherwise unbiased algorithms can contain programming errors. While sometimes not visible to the human eye, they can render error measurements useless.

We tested for bias by looking at the RMSE-over-time plot. If the RMSE graph does not become horizontal, but continues to be $\Theta(1/N)$, there is no measurable bias. Jumps are not a problem.

APPENDIX E

Table of Rendering Algorithms

E.1 Monte Carlo (MC) methods

PT	Path Tracing starts independent paths from the camera. It samples the BSDF locally and the light sources directly.
BDPT	Bidirectional Path Tracing starts two paths, one from the camera and one from a light source. It then connects all vertices from the camera path with all vertices from the light path.

E.2 Markov chain Monte Carlo (MCMC) methods

MLT	Path space Metropolis Light Transport employs mutations of existing paths to search important paths close to the old one. The mutations are in path space, i.e., they can employ all informations available to the path.
MEMLT	Manifold Exploration MLT, it adds an additional mutation type that employs surface derivatives for a better exploration of specular chains.
PSSMLT	Primary Sample Space MLT, instead of mutating in path space, the random number generator is manipulated to create correlated chains.

ERPT Energy Redistribution Path Tracing, uses very short chains compared to MLT. The chains are started probabilistically from PT samples.

E.3 Biased but consistent methods

SPPM Stochastic Progressive Photon Mapping casts gathering points from the camera into the scene. In a second processing step photons are shot from the light sources. When they reach a gathering point, the contribution is added to the corresponding pixel. The connection is not exact, hence the bias.

APPENDIX F

Table of Symbols

F.1 Notation

$\vec{X}$	Vectors represent the whole image or spectrum
$\vec{X}[j]$	Element j in vector $\vec{X}$
$\hat{X}_N$	Hats denote the mean of N draws (N can be left out for brevity)
$\mathrm{E}[X]$	Expectation of X (element-wise for vectors)
$\mathrm{Var}(X)$	Variance of X (element-wise for vectors)
$\mathcal{F}\vec{X}$	Fourier transform of image $\vec{X}$
Re and Im	Real and imaginary part
$\lvert X\rvert$	magnitude of a complex number
N	Number of samples
$\mathcal{N}$	Number of short renderings in the proxy algorithm
M	Number of elements in a vector or pixels in an image
$\hat{\vec{I}}_N$	Random variable for rendered image after taking N samples per pixel
$\vec{R}$	Reference solution, $\vec{R} \approx \mathrm{E}\left[\vec{I}\right]$
$\hat{\vec{\mathcal{E}}}$	Error, computed by $\hat{\vec{I}} - \vec{R}$
$f(x)$	Light transport function on path x
$h_j(x)$	Pixel filter for pixel j

F.2 Variables

Unimportant symbols used only in one section are left out.

I	Integral being computed
$\vec{I}$	Image being computed

List of Figures

Glossary

biased Biased means, that the expected value of the solution is not correct. i.e., when running the algorithm infinitely many times and taking the mean, it will not be correct. 2, 8, 15, 20, 72

body Position and dispersion of the majority of radial averages of error power spectra. 50

caustic Concentration of light produced by specular objects, for instance glass. 25

confidence Confidence in the statistical sense, i.e., accuracy of the measurement. It depends on the number of samples and their variance. Confidence is increased when taking more samples. 51, 54, 58

constant of convergence Constant in $\Theta(1/N)$. 45, 47

DC term Constant factor of the DFT transform, i.e., the 0th frequency and the mean of the signal. 22, 23, 40, 42, 52, 58

ensemble frequency Frequency in a radial average or sum of a Fourier (power) spectrum. In ESE the ensemble frequencies are on the x-axis. 23, 51, 53, 58

ensemble mean Estimator for the radial average of $\mathrm{E}\left[|\mathcal{F}\hat{\tilde{\mathcal{E}}}|^2\right]$. 50

Fourier frequency The element (pixel) of a discrete Fourier spectrum or power spectrum. 22, 38, 58

head Position and dispersion of the *mode* of radial averages of error power spectra. 50

importons The dual of photons, i.e., an importance particle emitted from the sensor which measures the amount of light that will be added to the sensor element when a light is hit. 7, 14

light transport function The function that measures the contribution of a certain path to the sensor. When multiplied with the pixel filter, it is the contribution to the pixel. 6, 10

photon mapping Approximate rendering method, it is biased but consistent. 2, 4

pixel outlier An isolated, very bright pixel (firefly). It can appear when the error distribution has a long tail. 34, 43, 81

radial average A vector whose elements are averages of centred circles in a square matrix (in our context error power spectra). The radii are 0, 1, 2, ... 0.5 × the matrix size. 23, 50, 56–58

tail Position and dispersion of outlier radial averages of error power spectra. A large tail means that the error (MSE) jumps when plotted over time. 50

unbiased Unbiased algorithms provide the correct solution on average. 2, 3, 13

Veach-MLT Veach style MLT, i.e., path space MLT with Veach's original mutation set. 19, 75

Acronyms

BDPT BiDirectional Path Tracing. 14, 19, 46, 61, 63, 69, 81

BSDF Bidirectional Scattering Distribution Function. 7, 9, 14

CLT Central Limit Theorem. 3, 11, 29, 34, 38, 45

DFT Discrete Fourier Transform. 21, 22

ERPT Energy Redistribution Path Tracing. 4, 19, 20, 76

ESE Error Spectrum Ensemble. 49, 57, 61, 69, 81, 82

i.i.d. independent and identically distributed. 11, 30

MC Monte Carlo. 2, 3, 10, 12–15, 18, 20, 26, 29, 30, 40, 43, 44, 52, 61, 81

MCMC Markov Chain Monte Carlo. 2, 3, 10, 15, 16, 20, 26, 29, 40, 43, 52, 61, 81

MEMLT Manifold Exploration path space MLT. 19, 46

MIS Multiple Importance Sampling. 14, 15

MLT Metropolis Light Transport. 3, 15, 18, 19, 26, 30, 33, 52, 56, 58, 61, 72, 81, 103, 109

MLT path space MLT. 19, 109

MSE Mean Square Error. 3, 24, 25, 44, 46, 49, 50, 57, 81

NPS Noise Power Spectrum. 27

PDF Probability Density Function. 11, 18, 29, 31, 32, 34, 44, 81

PSSMLT Primary Sample Space MLT. 4, 19, 52, 61, 69, 72, 82, 109

PT Path Tracing. 14, 19, 20, 33, 63, 81, 103

RMSE Root Mean Square Error. 24, 25, 44, 50, 56, 58

SPPM Stochastic Progressive Photon Mapping. 20, 30, 61, 78, 82

Bibliography

[APSS01] Michael Ashikhmin, Simon Premože, Peter Shirley, and Brian Smits. A variance analysis of the Metropolis light transport algorithm. *Computer & Graphics*, 25(2):287–294, 2001.

[ATS94] James Arvo, Kenneth Torrance, and Brian Smits. A framework for the analysis of error in global illumination algorithms. In *Proceedings of SIGGRAPH '94*, pages 75–84, 1994.

[BBS14] Laurent Belcour, Kavita Bala, and Cyril Soler. A local frequency analysis of light scattering and absorption. *ACM Transactions on Graphics (TOG)*, 33(5):163:1–163:17, 2014.

[BGJM11] Steve Brooks, Andrew Gelman, Galin L. Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall / CRC, Boca Raton, 2011.

[Boa66] Mary Layne Boas. *Mathematical methods in the physical sciences*. Wiley New York, 1966.

[CTE05] David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tracing. *ACM Transactions on Graphics (TOG)*, 24(3):1186, 2005.

[DHS+05] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X. Sillion. A frequency analysis of light transport. *ACM Transactions on Graphics (TOG)*, 24(3):1115–1126, 2005.

[DS02] Lokenath Debnath and Firdous Ahmad Shah. *Wavelet transforms and their applications*. Birkhäuser Boston, 2002.

[Dut96] Philip Dutré. *Mathematical frameworks and Monte Carlo algorithms for global illumination in computer graphics*. Phd thesis, KU Leuven, 1996.

[FA91] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[Gey11] Charles J. Geyer. Introduction to Markov Chain Monte Carlo. In Steve Brooks, Andrew Gelman, Galin L. Jones, and Xiao-Li Meng, editors, *Handbook of Markov Chain Monte Carlo*, pages 3–48. Chapman & Hall / CRC, 2011.

[Han98] Kenneth M. Hanson. Simplified method of estimating noise-power spectra. In *Medical Imaging 1998: Physics of Medical Imaging*, volume 3336, pages 243–250, 1998.

[HJ08] Toshiya Hachisuka and Henrik Wann Jensen. Progressive photon mapping. *ACM Transactions on Graphics (TOG)*, 27(5):130, 2008.

[HJ09] Toshiya Hachisuka and Henrik Wann Jensen. Stochastic progressive photon mapping. *ACM Transactions on Graphics (TOG)*, 28(5):141, 2009.

[HKD14] Toshiya Hachisuka, Anton S. Kaplanyan, and Carsten Dachsbacher. Multiplexed metropolis light transport. *ACM Transactions on Graphics (TOG)*, 33(4):100, 2014.

[HKD15] Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. Improved half vector space light transport. *Computer Graphics Forum*, 34(4):65—-74, 2015.

[Hof13] Naty Hoffman. Background: Physics and math of shading. *Physically Based Shading in Theory and Practice*, 24(3):211—-223, 2013.

[Jak10] Wenzel Jakob. Mitsuba renderer. *http://www.mitsuba-renderer.org*, 2010.

[Jak13] Wenzel Jakob. *Light transport on path-space manifolds*. Phd thesis, Cornell University, 2013.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques '96*, volume 96, pages 21–30, 1996.

[JMD15] Jin Woo Jung, Gary Meyer, and Ralph Delong. Robust statistical pixel estimation. *Computer Graphics Forum*, 34(2):585–596, 2015.

[JS12] Wenzel Jakob and Steve Marschner. Manifold exploration: A Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (TOG)*, 31(4):58, 2012.

[Kaj86] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, 1986.

[KMA+15] Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Fredo Durand, and Matthias Zwicker. Gradient-domain path tracing. *ACM Transactions on Graphics (TOG)*, 34(4):123, 2015.

[KSKAC02] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A simple and robust mutation strategy for the Metropolis light transport algorithm. *Computer Graphics Forum*, 21(3):531–540, 2002.

[KW04] Tae-Hwan Kim and Halbert White. On more robust estimation of skewness and kurtosis: Simulation and application to the S&P500 index. *Finance Research Letters*, 1(1):56–73, 2004.

[LD08] Ares Lagae and Philip Dutré. A comparison of methods for generating poisson disk distributions. *Computer Graphics Forum*, 27(1):114–129, 2008.

[Liu01] Jun S. Liu. *Monte Carlo strategies in scientific computing*. 2001.

[Liv07] John H. Livesey. Kurtosis provides a good omnibus test for outliers in small samples. *Clinical Biochemistry*, 40(13):1032–1036, 2007.

[LKL+13] Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. Gradient-domain metropolis light transport. *ACM Transactions on Graphics (TOG)*, 32(4):1, 2013.

[MKA+15] Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. Gradient-domain bidirectional path tracing. In *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*. The Eurographics Association, 2015.

[NB99] Mark E.J. Newman and G.T. Barkema. *Monte Carlo methods in statistical physics*. Clarendon Press, 1 edition, 1999.

[Osg07] Brad Osgood. The Fourier transform and its applications (lecture notes). Technical report, 2007.

[PH10] Matt Pharr and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, Burlington, USA, second edi edition, 2010.

[PSC+15] Adrien Pilleboue, Gurprit Singh, David Coeurjolly, Michael Kazhdan, and Victor Ostromoukhov. Variance analysis for Monte Carlo integration. *ACM Transactions on Graphics (TOG)*, 34(4):124:1—-124:14, 2015.

[SCJ02] J H Siewerdsen, I a Cunningham, and D a Jaffray. A framework for noise-power spectrum analysis of multidimensional images. *Medical physics*, 29(2002):2655–2671, 2002.

[SK13] Kartic Subr and Jan Kautz. Fourier analysis of stochastic sampling strategies for assessing bias and variance in integration. *ACM Transactions on Graphics (TOG)*, 32(4):128, 2013.

[SKDP99] Laszlo Szirmay-Kalos, Peter Dornbach, and Werner Purgathofer. On the start-up bias problem of metropolis sampling. In *Winter School of Computer Graphics '99*, pages 273–280, 1999.

[Vea97] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. Phd thesis, Stanford University, 1997.

[VG94] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*, pages 145–167. Springer Berlin Heidelberg, 1994.

[VG95] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of SIGGRAPH '95*, pages 419–428, New York, New York, USA, 1995. ACM Press.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of SIGGRAPH '97*, pages 65–76, New York, New York, USA, 1997. ACM Press.

[Wes14] Peter H. Westfall. Kurtosis as peakedness, 1905 - 2014. R.I.P. *The American statistician*, 68(3):191–195, 2014.

[ZAD15] Tobias Zirr, Marco Ament, and Carsten Dachsbacher. Visualization of coherent structures of light transport. *Computer Graphics Forum*, 34(3):491–500, 2015.