Fourier Analysis of Numerical Integration in Monte Carlo Rendering: Theory and Practice

Understanding estimation error in Monte Carlo Image Synthesis

Kartic Subr, Gurprit Singh and Wojciech Jarosz

SIGGRAPH 2016 COURSES, JULY 24-28, 2016, ANAHEIM, CA.

ACM 978-1-4503-4289-6/16/07.
http://dx.doi.org/10.1145/2897826.2927356

Abstract

Since being introduced to graphics in the 1980s, Monte Carlo sampling and integration has become the cornerstone of most modern rendering algorithms. Originally introduced to combat the effect of aliasing when estimating pixels values, Monte Carlo has since become a more general tool for solving complex, multi-dimensional integration problems in rendering. In this context, MC integration involves sampling a function at various stochastically placed points to approximate an integral, e.g. the radiance through a pixel integrated across the multi-dimensional space of possible light transport paths. Unfortunately, this estimation is error-prone, and the visual manifestation of this error depends critically on the properties of the integrand, placement of the stochastic sample points used, and the type of problem (integration vs. reconstruction) that is being solved with these samples.

We describe how errors present in rendered images may be analyzed as a function of the spectral (Fourier domain) statistics of the underlying sampling patterns fed to the renderer. Fourier analysis, along with the Nyquist theorem, has long been used in graphics to motivate more intelligent sampling strategies which try to minimize errors due to noise and aliasing in the *pixel reconstruction problem*. Only more recently, however, has the community started to apply these same Fourier tools to analyze error in the Monte Carlo *integration problem*. Loosely speaking, in the context of rendering a 2D image, these two problems are concerned with errors introduced across pixels (reconstruction) vs. the errors introduced within any individual pixel (integration). In this course, we focus on the latter, and survey the recent developments and insights that Fourier analyses have provided about the magnitude and convergence rate of Monte Carlo integration error. We provide a historical perspective of Monte Carlo in graphics, review the necessary mathematical background, summarize the most recent developments, discuss the practical implications of these analyzes on the design of Monte Carlo rendering algorithms, and identify important remaining research problems that can propel the field forward.

Relationship to courses previously presented at SIGGRAPH

Physically based rendering is a broad area that subsumes many active sub-areas of research. Rendering courses, which have been previously presented at SIGGRAPH, have focused on aspects of physically-based shading computation, the benefits of specific algorithms for rendering, summarization of state-of-the-art algorithms for light transport simulation, etc. There have also been courses on the fundamentals of Monte Carlo and quasi Monte Carlo sampling in the context of rendering. The focus of this course is complementary to previous rendering courses. We do not expect (or require) the audience to have attended any of the aforementioned courses.

Fourier analysis is also extensively used in the design of computational displays and cameras. Our focus on numerical integration is fundamentally different than those applications, where integration happens physically (implicitly) by propagation of real light energy (e.g. blurring of images due to a wide aperture). There have been conspicuously few courses on the application of Fourier analysis at SIGGRAPH and we hope that the sections containing generalized treatment of the problem will provide theoretical insights into applications other than rendering.

Intended audience, prerequisites and level of difficulty

This course is designed for graduate students and professionals seeking to understand Monte Carlo rendering, Fourier analysis and/or errors due to sampling. We expect attendees to possess undergraduate-level proficiency in probability theory and calculus. We will begin by reviewing much of the intuition required for the appreciation of an intricate combination of Monte Carlo rendering, Fourier analysis, sampling theory and probability, which is the focus of this course.

About the lecturers

Prof. Kartic Subr <k.subr@hw.ac.uk>
Heriot Watt University http://home.eps.hw.ac.uk/~ks400/

Kartic Subr is an Associate Professor of Engineering at Heriot Watt University and a Royal Society University Research Fellow. He has enjoyed a variety of positions in companies such as Hewlett Packard, Rhythm and Hues Studios, NVIDIA Corporation and Disney Research as well as academic post-doctoral research positions at INRIA-Grenoble and University College London. He obtained his Ph.D. (2008) and M.S. (2005) in Visual Computing from UC Irvine.

Kartic's research explores the benefits of stochastic and adaptive sampling when applied to a variety of classes of problems encountered in computer graphics such as numerical integration, signal reconstruction, edge-preserving smoothing of images, assisted segmentation of images and image completion. More recently, he has focused on error analyses of popular stochastic sampling strategies for numerical integration in the context of simulating light transport for rendering.

Dr. Gurprit Singh <gurprit.singh@dartmouth.edu>
Dartmouth College http://www.cs.dartmouth.edu/~gsingh

Gurprit Singh is a post-doctoral researcher at Dartmouth College, working with Prof. Wojciech Jarosz in the Dartmouth's Visual Computing Lab. Before joining Dartmouth in 2015, Gurprit was a PhD candiate at the Univerisité de Lyon 1, France, where he successfully obtained his doctorate (2015) under the supervision of Prof. Victor Ostromoukhov. Gurprit did his Masters (2012) from INP Grenoble, France and obtained his bachelor's thesis (2010) from IIT Delhi, India, under the supervision of Prof. Prem Kalra and Prof. Niloy J. Mitra.

Gurprit's research is mainly focused on understanding point sample distributions from the Fourier perspective. During his thesis work, Gurprit developed a theoretical model in the spherical domain that can characterise various sampling methods (jitter, Poisson Disk, Blue noise) based on their frequency content. This theoretical framework can be used to predict the exact variance for Monte Carlo integration. Later on, this framework is also used to analytically derive the variance convergence rates of various well known sampling patterns (jitter, Poisson Disk) that are extensively used in production rendering.

Prof. Wojciech Jarosz <wojciech.k.jarosz@dartmouth.edu>
Dartmouth College http://www.cs.dartmouth.edu/~wjarosz

Wojciech Jarosz is an Assistant Professor of Computer Science at Dartmouth College and co-founder of Dartmouth's Visual Computing Lab. Before joining Dartmouth in 2015, Wojciech was a Senior Research Scientist heading the rendering group at Disney Research Zürich, and an adjunct lecturer at ETH Zürich. He obtained his Ph.D. (2008) and M.S. (2005) in computer graphics from UC San Diego, and a B.S. (2003) in computer science from the University of Illinois at Urbana–Champaign.

Prof. Jarosz's research covers many areas of computer graphics and rendering, with a particular emphasis on *light*: how to capture it, how to more efficiently simulate its interaction and propagation within scenes, how to author or edit appearance, and how to create physical objects with prescribed light interactions. Much of his work has involved developing more sophisticated sampling techniques in the context of Monte Carlo integration and the resulting methods have been incorporated into production rendering systems for the making of feature films, including Disney's *Tangled* and *Big Hero 6*. In 2013, Prof. Jarosz received the Eurographics Young Researcher Award.

Course Syllabus

Table 1: Course topics, overview and schedule

Section	**Description**	**Speaker**	**Dur.** (mins)
Introduction	Photorealistic rendering Sampling and reconstruction Monte Carlo (MC) rendering	Subr	15
Math. Review	Probability & Fourier theory MC integration Fourier analysis of MC	Subr	20
Q. questions and/or break (5 mins)			
Analysis of error due to sampling	Assessing sampling patterns Beyond canonical domain Manifestation of error	Singh	15
Sampling patterns: State of the art	Classical MC & QMC Blue noise Patterns with targeted spectra	Jarosz	15
Q. questions and/or break (5 mins)			
Experiments	Case Studies	Singh	10
	Rendering test suite	Subr	5
90 mins = 50 (theory) + 20 (practice) + 10 (state of the art) + 10 (Q.)			

Contents

1. Introduction

Persuading our sophisticated visual systems of the "realism" of computer graphics imagery is a challenging task. This problem is exacerbated by the presence of fantastic objects and/or characters embedded in the virtual scene, thereby loosening the notion of realism in the resulting depiction. Realistic depiction is typically achieved by an artful combination of design considerations such as the degree of detail in geometrical modelling of objects in the environment, application of plausible models for dynamics, tactful animation of objects and characters, the choice of display technology used to present the visuals to the observer, etc.

While the totality of the above decisions impacts the illusion of reality, the focus of this course is on a vital component in the pipeline — *the synthesis of photorealistic images* given the various models that describe the environment. The state of the art in the generation of plausible pictures – which we will henceforth refer to as *rendering* – involves simulation of the physics of light. The simulation of light transport is typically tedious and error-prone. Error-free rendering is a computationally expensive process that needs to be performed repetitively, on every frame, to generate high (production) quality video sequences.

1.1 Rendering

We refer to rendering as the process of producing a picture which represents a particular view of an environment, captured using a virtual camera, at a given time. The input to the rendering system, or renderer, is a detailed representation of all the objects in the scene. This representation is typically provided in a *scene description file* that specifies the objects in the scene, their shapes, their material properties (rough, smooth, shiny, diffuse, translucent, etc.), their motion parameters and whether these objects emit light energy. In addition to the environment, the renderer is also provided with a description of the virtual camera used to capture the rendered view including its position, orientation, optical characteristics (any lenses, aperture, etc.), shutter parameters (exposure time) and parameters of the virtual sensor (aspect ratio, optical sensitivity, etc.) embedded within it.

1.2 The interplay between reconstruction and integration

Images are typically represented and displayed on a regular grid. Rendering algorithms, however, may estimate virtual measurements at arbitrarily points on the sensor. We refer to the process of resampling the measurements I on a regular grid as *2D reconstruction*. Some rendering algorithms [18, 16, 45, 5] partition the space of paths (Ω in equation 4.1) explicitly into dimensions that are dependent on the camera, such as exposure time and aperture, and those that depend on the scene. Since the sensor is two-dimensional, exposure time is one-dimensional and the aperture is two-dimensional, they estimate virtual measurements in a five-dimensional, camera-dependent space. This involves a reconstruction in 5D followed by a further projection (integration) down to the two-dimensional plane of the sensor. Such a partitioning is exploited for efficient rendering of camera-dependent blur effects resulting from relative motion between the camera and the scene or from defocus due to a finite (rather than infinitesemal) aperture.

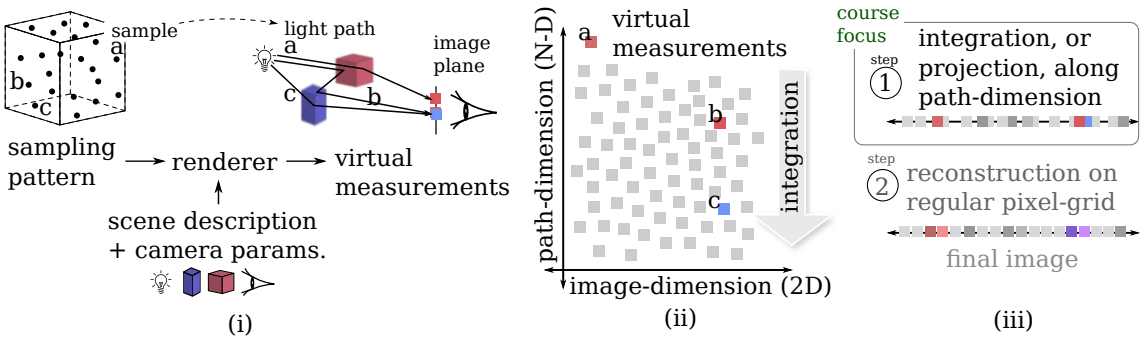


Figure 1.1: *The renderer implicitly maps samples from some high-dimensional space to light paths in a specified scene. The contributions of radiant light energy along these paths are recorded on the image plane as virtual measurements. Three light paths (a,b,c) are shown along with their canonical samples in the (hyper-) cube. The next two steps of rendering are integration along the path-dimension (step 1) followed by reconstruction on a regular grid (step 2). In this course, we focus on the impact of Fourier spectral properties of the canonical samples on the errors due to numerical integration.*

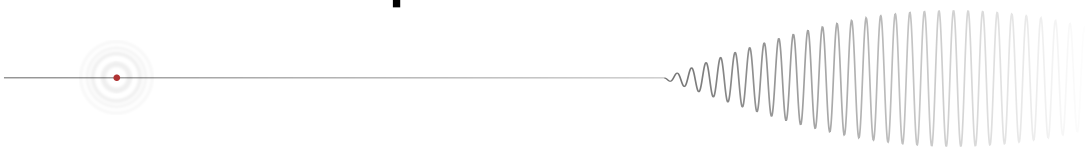
1.3 The role of sampling

There are two distinctly different applications of sampling in the rendering process (see figure 1.1). One of them is for reconstruction, either in 2D or in 5D (as described above), and the other is for estimation of integration, which lies at the core of light transport (equation 4.1). Most rendering algorithms focus on the latter and perform reconstruction on a two-dimensional regular grid of pixels. Path-tracing, a popular rendering algorithm, proceeds by estimating the virtual image measurement at each pixel by averaging the values of the measurement contributions of a chosen set of paths through each pixel. Each path in the chosen set is obtained via a mapping from a sampling pattern (of appropriate dimensionality) to a set of rays in the scene. These sampling patterns may be stochastic (Monte Carlo sampling) or deterministic patterns that satisfy desirable criteria (Quasi-Monte Carlo). The statistics of the estimates of the virtual measurements yielded by these two broad classes of algorithms are strikingly different, as is the mathematical machinery used for their analyses.

1.4 Sources and manifestation of error

Error in the context of estimating global illumination [2] arises from three sources: inaccuracies in the input data or simulation, errors due to discretization and computational errors such as loss of precision. In this course, we focus on the first two sources. The rendered image is represented on a discrete grid, while the distribution of the underlying light energy across the sensor is a continuous function. A major source of error stems from this discretization. The underlying continuous function is a high-dimensional integral at each point on the sensor (equation 4.1). Any sampling-based approach used to estimate this function results in an approximation whose measured deviation from the true function is known referred to as *approximation error*. Typically this error manifests in rendered images as structured artifacts or noise.

2. Mathematical preliminaries



2.1 Random variables, expectation and variance

For the purpose of this course, it will suffice to think of random variates simply as variables whose value is subject to chance. For formal definitions of random variables and functions of random variables, we refer interested readers to introductory textbooks on probability or Monte Carlo simulation [41]. We will restrict ourselves to continuous random variables, denoted using bold capital letters ($\mathbf{X}$, $\mathbf{Y}_i$, etc.), that assume values in some domain $\mathscr{D}$. The relative likelihood for continuous random variables to take on a given value is specified using a (normalized) probability density function (pdf) $g : \mathscr{D} \to \mathbb{R}$. We denote that $\mathbf{X}$ is distributed according to $g(x)$ by $\mathbf{X} \sim g(x)$. The probability that the random variable lies in a subdomain $\mathscr{D}_i \subset \mathscr{D}$ is obtained as $\mathbb{P}(\mathbf{X} \in \mathscr{D}_i) = \int_{\mathscr{D}_i} g(x)\ \mathrm{d}x$.

A function $\phi : \mathscr{D} \to \mathbb{R}$ evaluated at a location specified by a random variable, $\phi(\mathbf{X})$, is also a random variable. In this course, we are primarily concerned by the first two moments of random variables (specifically functions of random variables). The first moment, or "average value" of a random variable is captured by the mathematical concept of the *expected value* of the random variable. We denote the expected value of $\mathbf{X}$ as $\mathrm{E}[\phi(\mathbf{X})]_g \equiv \int_{\mathscr{D}} \phi(x)\, g(x)\ \mathrm{d}x$.

If the pdf has an expected value, the variance of the random variable is its second central moment which we denote $\mathrm{V}[\phi(\mathbf{X})]_g \equiv \mathrm{E}\left[\phi^2(\mathbf{X})\right]_g - \mathrm{E}[\phi(\mathbf{X})]_g^2$. When $\mathbf{X}$ is distributed uniformly within the domain (i.e. $g(x)$ is a constant), we drop the subscript and write the expectation as $\mathrm{E}[\phi(\mathbf{X})]$ and the variance as $\mathrm{V}[\phi(\mathbf{X})]$.

2.2 Estimators

Consider Monte Carlo estimation of the multidimensional integral $I = \int_{\mathscr{D}} f(x)\ \mathrm{d}x$, $x \in \mathscr{D}$.A simple *primary* MC estimator for I is $\tilde{\mu}_1 \equiv f(\mathbf{X})$, $\mathbf{X} \in \mathscr{D}$. When $\mathbf{X}$ is distributed uniformly, the estimator is unbiased. That is, its expected value is the integral:$\mathrm{E}[\tilde{\mu}_1] = I$. The function $f(\mathbf{X})$ is itself a random variable with an arbitrary distribution and, typically, a large variance. A more practical MC estimator is obtained by averaging a fixed number of (say N) primary estimates: $\tilde{\mu}_\mathrm{N} = \sum f(\mathbf{X}_i)/N$, $i = 1, 2, ..., N$. Such *secondary estimators* are known to be unbiased and approach Gaussian distributions as N is increased, if the primary estimator has finite variance.

2.3 The continuous Fourier transform

The process of solving differential equations is simplified by projecting relevant functions onto a basis formed by eigenfunctions of the differential operator. The class of functions $x \to \mathrm{e}^{\imath\omega x}$ forms a natural choice since $\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{e}^{\imath\omega x} = \imath\omega.\mathrm{e}^{\imath\omega x}$. The projection of a given function, $f : \Re \to \mathbb{C}$, results in a set of coefficients — one coefficient corresponding to every function in the basis (so, per choice of ω). If the basis is chosen so that it contains continuous (real) values of ω, the corresponding coefficients may be viewed as a function of ω. $\hat{f} : \Re \to \mathbb{C}$. Under certain conditions [1] the relationship between

[1] We refer readers to textbooks [7, 21] on the subject for a mathematically rigorous treatment.

$f(x)$ and its coefficients of projection $\hat{f}(\omega)$ can be written as

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x)\, \mathrm{e}^{-\imath 2\pi\omega x}\, \mathrm{d}x \tag{2.1}$$

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\omega)\, \mathrm{e}^{\imath 2\pi\omega x}\, \mathrm{d}\omega. \tag{2.2}$$

The integral transforms in eq. 2.1 and eq. 2.2 are known as the *Fourier transform* and the *inverse Fourier transform* respectively. Due to Euler's formula in complex analysis, $\mathrm{e}^{\imath 2\pi\omega x} = \cos(2\pi\omega x) + \imath\sin(2\pi\omega x)$, the projection of functions onto complex exponentials is equivalent to a projection onto sinusoids of frequencies given by $2\pi\omega$. For this reason, ω is referred to as the *frequency variable* and $\hat{f}(\omega)$ is said to be a representation of $f(x)$ in the *frequency domain*. We denote the Fourier transform pair using $f(x) \overset{\mathscr{F}}{\Longleftrightarrow} \hat{f}(\omega)$. In summary, the continuous Fourier transform decomposes a function on the infinite (real) domain into a continuous sum (integral) of sinsuoids of continuous and unbounded set of frequencies. The set of continuous complex coefficients arising from this decomposition is called the *Fourier spectrum* of the function.

2.4 Fourier series

The projection of periodic functions onto complex exponentials leads to a special case. Consider a general periodic function $f_T(x)$ obtained by convolving a function $f(x)$ with a Dirac comb $\perp\!\!\!\perp\!\!\!\perp_T(x)$ with period T. Since $f_T(x) = f(x) \otimes \perp\!\!\!\perp\!\!\!\perp_T(x)$, the Fourier transformed periodic function is a product $\hat{f}_T(\omega) = \hat{f}(\omega)\hat{\perp\!\!\!\perp\!\!\!\perp}_T(\omega)$ (due to the convolution theorem). Since $\hat{\perp\!\!\!\perp\!\!\!\perp}_T(\omega)$ is also a Dirac comb (with a period of $1/T$ in the frequency domain), the Fourier spectrum of $f_T(x)$ is only non-zero at the set of locations where $\hat{\perp\!\!\!\perp\!\!\!\perp}_T(\omega)$ is non-zero. Therefore, the Fourier spectrum of a continuous periodic function is non-zero at discrete (regular) frequencies.

The inverse Fourier transform of the periodic function can then be written as a summation:

$$f_T(x) = \sum_{n\in\mathbb{Z}} c_n\, \mathrm{e}^{\imath 2\pi n x/T}. \tag{2.3}$$

The c_n, known as the *Fourier series coefficients*, are obtained as

$$c_n = \frac{1}{T}\int_{x_0}^{x_0+T} f(x)\, \mathrm{e}^{-\imath 2\pi n x/T}\, \mathrm{d}x = \frac{\hat{f}(n/T)}{T} \tag{2.4}$$

where x_0 is any choice for x so that $f(x)$ is integrable within $[x_0, x_0+T]$. Typically the complex exponential in eq. 2.4 is expanded in terms of sin and cos, leading to two sets of coefficients. This crucial observation, that even periodic functions with discontinuities may be expressed as the sum of an infinite set of harmonics was made by Joseph Fourier in his study of heat. It was subsequently rigorously proved by Fourier's protégé, Peter Dirichlet, who also outlined requirements for convergence of the Fourier series. Dirichlet's results formed the theoretical foundation which led to the derivation of the continuous Fourier transform.

2.5 Discrete time Fourier transform (DTFT)

Recall that the Fourier transform of a periodic function results in a discrete spectrum. Conversely, the Fourier transform of a uniformly sampled continuous function results in a periodic Fourier spectrum. Consider the a discrete function obtained by sampling by a Dirac comb. i. e. the function is $f(x).\perp\!\!\!\perp\!\!\!\perp_s(x)$, where s is the sampling period. The resulting Fourier spectrum (of the discrete function) will be the convolved spectrum, $\hat{f}(\omega) \otimes \hat{\perp\!\!\!\perp\!\!\!\perp}_s(\omega)$, which is periodic (with a period of $1/s$).

2.6 The Discrete Fourier transform (DFT)

In practice, Fourier analysis is often performed on sampled functions that span a finite domain. In most cases, the discrete function is also assumed to be replaced by its periodic version with a period equal to the width of the domain. Combining the analysis for Fourier series and DTFT, we can write the input function as $f_{sT}(x) = (f(x).\perp\!\!\!\perp\!\!\!\perp_s(x)) \otimes \perp\!\!\!\perp\!\!\!\perp_T(x)$. The Fourier transform of this discrete-time periodic function is $(\hat{f}(\omega) \otimes \hat{\perp\!\!\!\perp\!\!\!\perp}_s(\omega)).\hat{\perp\!\!\!\perp\!\!\!\perp}_T(\omega)$ which is periodic (with a period of $1/s$) and discrete due to the frequency-domain sampling by $\hat{\perp\!\!\!\perp\!\!\!\perp}_T(\omega)$.

Given data f_n, $n = 0, 1, ..., N-1$, the k^{th} ($k \in \mathbb{Z}$) coefficients of the DFT and IDFT are

$$c_k = \sum_{n=0}^{N-1} f_n \left(\cos(-2\pi kn/N) + \imath \sin(-2\pi kn/N)\right) \text{ and} \tag{2.5}$$

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} c_k \left(\cos(-2\pi kn/N) + \imath \sin(-2\pi kn/N)\right) \tag{2.6}$$

respectively. Due to periodicity, the domain of k is typically $[0, N-1]$. The DFT is a linear, invertible transformation $\mathscr{F}_{DFT} : \mathbb{C} \to \mathbb{C}$ that exhibits most properties of the continuous Fourier transform such as completeness and orthogonality. Also, important theorems such as the convolution theorem, shift theorem, Plancherel's and Parseval's theorems also have their equivalents in the context of the DFT. Naïve calculation of the N coefficients c_k requires $O(N^2)$ computation since each coefficient involves a summation over N terms. Fast Fourier Transform (FFT) algorithms solve this with $O(N \log N)$ computation.

2.7 Power spectral density (PSD) and the periodogram

The energy of a signal $f(x)$ is generally considered to be the integral of the square of the signal (over its entire domain). Due to Parseval's theorem, this may also be expressed as the integral, over all frequencies, of the square (for real signals) of the Fourier transform of the signal. The integrand in this expression of power, $|\hat{f}(\omega)|^2$, is referred to as the *power spectrum* of $f(x)$.

Now consider a wide-sense stationary random processes, from which a single observation is made. The data from this observation may be subjected to Fourier analysis and the PSD of the observed data — called a *periodogram* — may be considered as an estimate of the PSD of the generating process. Although the periodogram is not a good estimate [29, Sec.14.2.2], it is convenient and hence popular [26].

We denote the PSD of $f(x)$ as $\mathscr{P}_f(\omega)$, which represents the squared amplitude of the Fourier coefficients at the frequency ω. While we use the same notation for stochastic signals, the interpretation is slightly different. $\mathscr{P}_{\mathbf{S}}(\omega) \equiv |\hat{\mathbf{S}}(\omega)|^2$ denotes the power spectrum of a single instance of a sampling function. This is not the PSD of the stochastic process, rather it is an estimate of the periodogram $\mathrm{E}\left[|\hat{\mathbf{S}}(\omega)|^2\right]$. The PSD of the stochastic process is $|\mathrm{E}\left[\hat{\mathbf{S}}(\omega)\right]|^2$.

2.8 Sampling functions in the canonical domain and their spectra

We express sampling functions as the sum of weighted unit impulses, or Dirac Delta functions:

$$S(x) = \frac{\sum_{i=1}^{N} w_i \, \delta(x - x_i)}{\sum_{i=1}^{N} w_i}. \tag{2.7}$$

where $x_i \in [0, 1]$ are the sampled locations. Sampling of a continuous signal $f(x)$ can then be seen as multiplication by the sampling function, and the resulting sampled function is $f(x)\, S(x)$.

If the samples are equally weighted, $w_i = 1/N$. If the locations x_i are random variables (chosen stochastically), then the resulting sampling function is also a random variable defined across instances of the sampled x_i. We are mostly interested with this type of sampling functions, which we will denote using boldface, $\mathbf{S}(x)$, to be consistent with our notation for random variables. We denote the Fourier transformed sampling function using $\hat{\mathbf{S}}(\omega)$. Since this is also a random variable defined over specific instances of N sample locations, we may define its expectation $\mathrm{E}\left[\hat{\mathbf{S}}(\omega)\right]$ and variance $\mathrm{V}\left[\hat{\mathbf{S}}(\omega)\right]$. The k-sample periodogram of the sampling function is obtained by averaging k estimates of $|\hat{\mathbf{S}}(\omega)|^2$. The PSD of the sampling process is $|\mathrm{E}\left[\hat{\mathbf{S}}(\omega)\right]|^2$. Clearly, in this case, the periodogram is not an accurate estimator for the PSD, since averaging is performed without phase information.

2.9 Normalizing Periodograms

Since periodograms characterise sample distributions, a common practical hurdle encountered while comparing periodograms of different sampling patterns is that they are not on a common scale. There is no well defined metric for such a comparison betweeen various sampling periodograms and visual inspection is clearly insufficient. We propose a normalization procedure to enable fair comparison. As a reference, we use the periodogram of random sampling for which we have an analytical form: $||\mathscr{F}_S(\omega)||^2 = \delta(\omega)N^2 + N$. Dividing the random sampling periodogram by the number of points N yields unity at $\omega \neq 0$. Using a similar approach for the other sampling periodograms, we propose to compute the power (magnitude squared Fourier coefficients) by dividing the magnitude square coefficient values by N. i. e. $||\mathscr{F}_S(\omega)||^2/N$. As a result of this normalization, the periodograms would always converge to the power value of unity towards higher frequencies, irrespective of the number of samples.

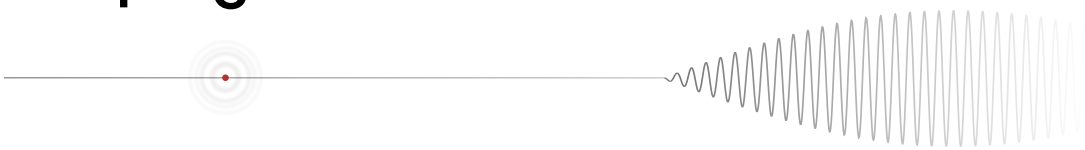
Heuristic for normalizing radially averaged periodograms

Once the above normalization is performed, we can compute the radially averaged periodograms with power converging to unity at higher frequencies. Increasing the number of samples causes energy to shift from the low frequency region of the radially averaged periodograms towards higher frequencies. During comparison, we would like to avoid this scaling without compromising the underlying information. So, we further divide the *radial frequency variable* ρ by $\frac{1}{\alpha \sqrt[D]{N}}$, where D represents the dimensionality of the space and $\alpha > 0$ is a parameter that be used to unify the scales of different radially averaged periodograms. For regular sampling patterns, $\alpha = 1$, since the first Fourier peak is at a distance of $\sqrt[D]{N}$ from the DC peak in the periodogram. As a result of normalizing ρ, the radial periodogram exhibits a first peak at $\rho = 1$ for regular patterns (Fig. 5.8). All averaged and radially averaged periodograms depicted in chapter 5 are obtained using the above normalization heuristics.

Table 2.1: A summary of notation used in this course.

Symbol	Definition
$f(x)$	integrand
$\tilde{\mu}_{\mathrm{N}}$	estimate of I using N samples
$\mathbf{X}$, $\mathbf{Y}$, $\mathbf{X}_i$, $\mathbf{Y}_i$	random variates in domain $\mathscr{D}$ with probability density functions $\phi(.)$
$\mathscr{P}_S(\omega)$	Power spectrum of a single instance of $\mathbf{S}(x)$ (periodogram)
$\breve{\mathscr{P}}_S(\rho)$	Radially-averaged power spectrum where $\rho = \|\omega\|_2$.
$\mathbf{S}(x) \overset{\mathscr{F}}{\Longleftrightarrow} \hat{\mathbf{S}}(\omega)$	Fourier transform pair: integrand and its transform
$\mathbf{S}(x) \overset{\mathscr{F}}{\Longleftrightarrow} \hat{\mathbf{S}}(\omega)$	Fourier transform pair: sampling function and its transform
$\mathrm{E}[\phi(\mathbf{X})]$	Expectation: $\int_{\mathscr{D}} x\, \phi(x)\ \mathrm{d}x$
$\mathrm{V}[\phi(\mathbf{X})]$	Variance : $\int_{\mathscr{D}} x\, \phi^2(x)\ \mathrm{d}x - \int_{\mathscr{D}} x\, \phi(x)\ \mathrm{d}x$

3. Sampling for reconstruction



The reconstruction problem entails approximating a function $f : \mathscr{D} \to \mathbb{R}$ as $f_a(x_i)$, $x_i \in \mathscr{D}$, $i = 1,2,3,...,M$ given $f(y_i)$, $y_i \in \mathscr{D}$, $j = 1,2,3,...,N$, where the sets of points $\{x_i\}$ and $\{y_j\}$ are mutually exclusive. If $f_a(y_j) = f(y_j)$, $\forall j$, the process is referred to as interpolation. This problem has received much attention in the signal processing literature [30, chapters 5 & 6] as well as in the rendering community [13, 11, 31, 6]. Such analyses have yielded effective reconstruction filters that are used in modern renderers, especially in low-dimensional domains such as the image plane.

Reconstruction is akin to nonlinear regression [42], a popular problem in the statistics (and more recently machine learning) literature, which offers effective solutions in high-dimensional settings when the approximand exhibits stationarity or accurate solutions for smooth and low-dimensional approximands. In rendering, the underlying functions represent a multitude of combinations (products, convolutions and sums) of a variety of non-stationary processes such as spatio-directional lighting, hard shadows, sharp reflections, textured materials, etc. The resulting discontinuities and variations in the radiant light energy (radiance function) across space-angle-time pose immense challenges which state of the art methods in non-linear regression are unable to cope with.

The reconstruction process generally consists of two steps. First, for any point x in the domain, the influence of each of the $f(y_j)$ samples is specified using some *weighting kernel* on the distance between x and y_i (typically Euclidean). Then, at each of the x_i, the approximate value is obtained by summing up the distance-weighted contributions of each of the $f(y_j)$. Depending on the choice of the kernel and the strategy for determining weights, the resulting methods are known by different names: Kernel Smoothers, Radial Basis Functions, Shepherd's method, etc.

3.1 Aliasing in reconstruction

The general reconstruction process (above) may be written in terms of the underlying function (or signal). The samples, $f(y_i)$, $y_i \in \mathscr{D}$, can be represented as the product of the true (often unknown) function $f(x)$ with a sampling function of the form $S(x) = \sum_{j=1}^{N} \delta(x - y_j)$. If $K(r)$ represents the kernel weight at a distance of r, the continuous approximand and its Fourier dual are

$$f_a(x) = (f(x).S(x)) \otimes K(x) \quad \overset{\mathscr{F}}{\Longleftrightarrow} \quad \hat{f}_a(\omega) = \left(\hat{f}(\omega) \otimes \hat{S}(\omega)\right).\hat{K}(\omega) \tag{3.1}$$

where $\otimes$ denotes convolution. That is, the effect of sampling can be seen in the Fourier domain as one of generating spurious replicas of $\hat{f}$ created due to the convolution with $\hat{S}$, which potentially hinders perfect reconstruction. The kernel, on the other hand, can be viewed as a low-pass filter which suppresses replicated, or aliased, copies of the true function's spectrum at high frequencies.

If $f(x)$ contains frequencies less than ω_b (is band-limited), if the sampling spectrum does not contain frequencies lower than $2\omega_b$, and if the kernel is a low-pass filter suppressing frequencies above ω_b, the resulting reconstruction will be perfect. If these conditions are not satisfied, the approximation will contain error. The first potential source of error is that $f(x)$ is not bandlimited and hence, regardless of the sampling function, the low-pass effect of the kernel results in the loss of high-frequencies. The other main source of error is due to the sampling spectrum containing low frequencies, which results in a $\hat{f}_a$ that contains a superposition of $\hat{f}(\omega)$ with its aliased copies.

3.2 Antialiasing

The error introduced by aliasing either appears as coherent structure or as noise, depending on the choice of sampling spectrum. Sampling on a regular grid corresponds to multiplication with a Dirac-train or Shah function, whose Fourier spectrum remains a Shah function. If the samples are not sufficiently dense in the image plane, the resulting spectrum contains impulses at lower frequencies which cause aliasing in the form of regular structure that has been shown to be particularly unpleasant to our perceptual system. If the function being reconstructed is bandlimited, and in the absence of a sampling budget, the sampling rate may be increased to eliminate aliasing. However, if the function contains arbitrarily high frequencies, a simple strategy is to introduce randomness in the spectrum thereby transforming coherent artifacts into visual noise. This has shown to be less objectionable to our visual systems.

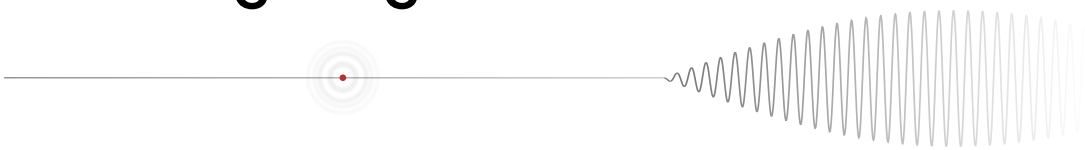
3.3 Halftoning

Halftoning is the process of determining a sampling function $S(x)$ so that a kernel density estimate of the sampled function $f_a(x) = f(x).S(x)$ provides the best approximation to $f(x)$. It is a special case of reconstruction where the kernel is implicit and dependent on our visual systems. Halftoning methods strive to identify sampling functions for which the *perceived* sampled function is as close to $f(x)$ as possible. This may be achieved by varying the distributions of the samples, by weighting them or using a combination of both. Densely sampled regions of the sampled function are perceived to have darker tone.

3.4 Antialiasing as integration

Antialiasing is commonly viewed as an integration problem. The value to be assigned to a pixel is often estimated by averaging sampled values within a square around the pixel. That is, pixels are viewed as little squares and the values assigned to pixels are obtained by integrating the underlying function within their respective squares. On the contrary, if pixels are viewed as simply being points on the image plane [44], this choice of integrating within the pixel appears to be an arbitrary choice. That is, it corresponds to performing reconstruction on the image plane exactly as in equation 3.1, with $K(x)$ chosen to be a box with a width that is half the inter-pixel distance, followed by resampling of $f_a(x)$ on a regular grid. Although this view of antialiasing is better than not performing antialiasing, it is well known that the box-filter is not the ideal choice. Modern renderers offer a variety of choices for the pixel reconstruction filter, since its choice is dependent on the rendering algorithm chosen and the corresponding sampling patterns used for estimating light energy arriving at the pixel [38, section 7.6].

4. Estimating integrals



It has long been known that the problem of numerical integration lies at the heart of rendering [37]. The main objective of rendering is to estimate the quantity of radiant light energy impinging on the various pixels (or cells) distributed on a virtual observer's camera (or eye). Current renderers perform this via physically-based simulation of light, following its intricate combinations of reflections within the scene. The virtual measurement I_j at a point j on the sensor [48, ch.8] is

$$I_j = \int_{\Omega} f_j(\bar{p}) \; \mathrm{d}\mu(\bar{p}) \tag{4.1}$$

where the measurement contribution function, $f_j(\bar{p})$, is the importance-weighted radiance arriving at j along the path $\bar{p}$ and $\mu(\bar{p})$ is a measure on the space of all paths Ω. The paths $\bar{p}$ span space as well as time and may be of arbitrarily high dimensionality.

The exact nature of the integrand depends on the complexity and structure of the scene being rendered, along with detailed modelling of the constituent materials and shapes. In general, the integrand is high-dimensional, discontinuous and costly to evaluate. Each evaluation of the integrand requires many rays to be traced within the environment, along with estimation of the radiant light energy along that path formed by linking the rays together. The rendering community strives to estimate such integrals with low error as well as low computational cost.

4.1 Numerical integration via sampling

Henceforth in this chapter, we abstract away the details about the integrand and domain of integration (equation 4.1) that is encountered in rendering. We study the integration problem in the canonical domain. Our goal is to analyze the of quality of numerical integration, within the interval $[0,T]$, of a function $f : \mathbb{R} \mapsto \mathbb{R}^+$. For example, a *primary* Monte Carlo estimator $\tilde{\mu}_1 = f(\mathbf{X}_i)$ for the integral

$$I \equiv \frac{1}{T} \int_0^T f(x) \; \mathrm{d}x \tag{4.2}$$

is well known to be unbiased if the single sample $\mathbf{X}_i \in [0,T]$ is drawn from a constant probability distribution function (pdf) within the domain $[0,T]$. i.e. $\mathrm{E}[\tilde{\mu}_1] = I$. The corresponding secondary estimator, obtained by averaging N primary estimates at different $\mathbf{X}_i$ has the same expected value. However, the averaging scales the variance down by a factor of N. i.e. $\mathrm{V}[\tilde{\mu}_\mathrm{N}] = \mathrm{V}[\tilde{\mu}_1]/N$.

4.2 Error due to sampling: Convergence rate, consistency, bias and variance

Numerical integration methods like Monte Carlo are very powerful in approximating the underlying multi-dimensional integrand using stochastically generated samples, but the overall process is highly prone to error. The nature of the error introduced depends on an intricate combination of three factors: the distribution of the sample locations; the weighting used to accumulate sampled contributions; and the arrangement (correlation) and structure of samples. The "error" of an estimator, which refers to its mean squared error (MSE), typically reduces as the number of samples

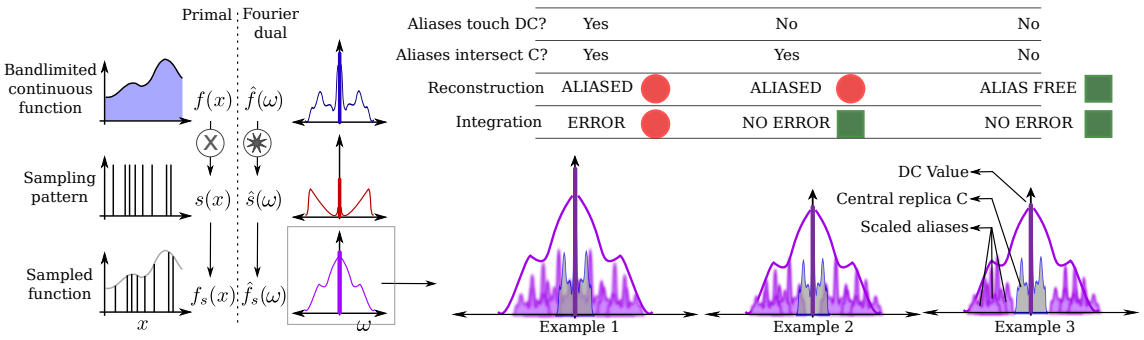


Figure 4.1: *This figure highlights the difference in the analysis of sampling rates for error-free reconstruction vs numerical integration of a bandlimited function. Fewer samples (factor of 2 per dimension [5]) are sufficient for error-free numerical integration.*

is increased. The rate at which this reduction occurs is known as the estimator's *convergence rate*. Estimators whose errors vanish in the limit are said to be *consistent*.

The MSE of an estimator is comprised of two different types of error: One that is associated with the accuracy of the estimator, and another which describes the precision of the estimates. The former, which is the deviation of the expected value from the true (or desired) result is known as *bias*. The latter, which quantifies the uncertainty of the estimator is known as *variance*. The MSE is expressed as the sum of the squared bias and the variance.

Consistent estimators, whether biased or unbiased, are desirable for image synthesis. Although objectionable image artifacts may be produced by the use of naïvely-designed biased estimators (say using regular sampling), in many cases consistent and biased estimators are used to reduce noise as well as to improve convergence rates [23].

4.2.1 Manifestation of error in rendering

For a given scene configuration (hence fixed integrand), the error in the rendered image is primarily dependent on the choice of sampling strategy. To a lesser extent, the error also depends on the algorithm used for rendering (path-tracing, Metropolis light transport, etc.). The motivation behind using stochastic sampling over regular samples was related to human perception. The bias caused by using regularly placed samples is considered more objectionable [13, 11] to our visual systems than its appearance as noise that is distributed across all frequencies (compare the strategies in the first column of fig. 4.2).

Consider an image rendered using path tracing with 64 samples per pixel. Each pixel in the final image can be viewed as a single instance of a secondary estimator, of the integral corresponding to that pixel, using 64 samples. In theory, the variance of each of these estimates can only be observed by rendering multiple images with 64 samples per pixel. In practice, even though the integrand at each pixel is different, the radiance varies smoothly over many parts of the image. As a result, if estimators for pixels in these smooth regions have high variance, their disagreement on particular instances of the estimate (in the rendered image) leads to a noisy appearance. The larger the disagreement of estimates in neighboring pixels in smooth regions, the noisier the image will appear. In other words, an estimator with a higher variance will lead to a noisier image.

The manifestation of bias is less intuitive. Typically, estimators with a structured bias — such as one that uses a regular sampling pattern — produce banded visual artifacts. In some cases, bias leads to artifacts that are inconsistent with our expectation of realism. e. g. bright patches at points where contact shadows are expected. Bias may also lead to incorrect estimates where the error is not visually perceivable. Such estimators are desirable, particularly if they can simultaneously improve

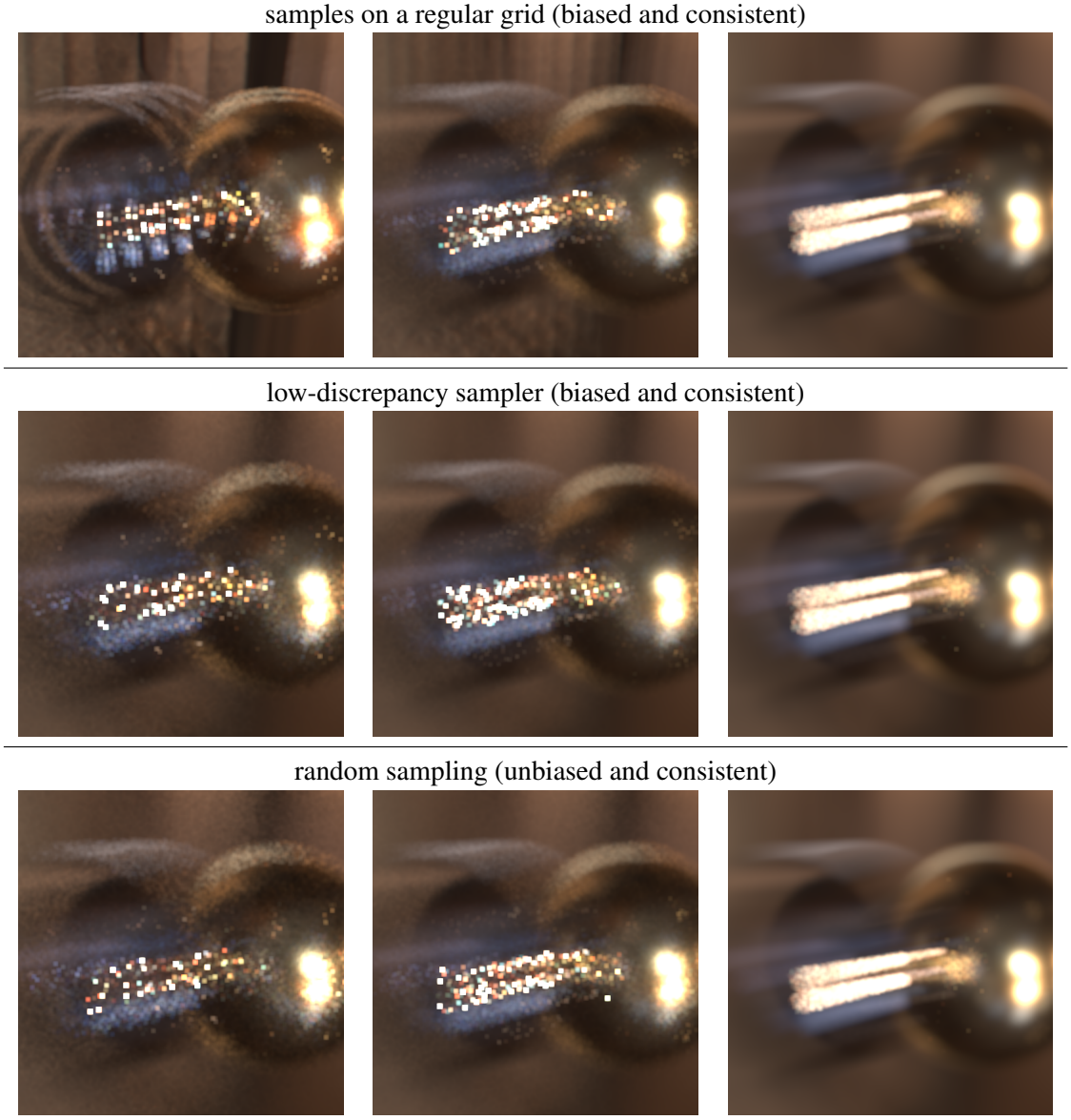


Figure 4.2: A comparison of rendered results obtained using 3 sampling strategies (rows) for various sampling counts (columns). The images depict two shiny balls: A golden ball that is out of focus and a mirrored ball that is moving rapidly through the plane in focus. All strategies lead to consistent estimators, therefore the images in the third column are virtually indistinguishable. The naïve, biased strategy of sampling on a regular grid leads to objectionable visual artifacts. A more sophisticated approach — low-discrepancy sampling (middle row) — although biased, not only eliminates the regular artifacts but also exhibits improved noise characteristics (for example in the background region, using 16 samples) compared to unbiased estimates obtained using random sampling (bottom row).

the visual noise. The solution to the inveterate problem of striking a reasonable compromise between bias and variance, for efficiency, depends on the nature of the scene, choice of sampling strategy, number of samples, etc.

The reduction of error, for a given sampling strategy, might be achieved by increasing the number of samples. So, in the above example, rendering with 128 or 256 samples per pixel will reduce noise. Some estimators (jittered sampling, QMC sampling) converge faster than others. Even for a fixed sampling strategy, primary and secondary estimators have different distributions and hence different statistics. Secondary estimators approach being Gaussian-distributed, as the number of the fixed samples used is increased. Choosing the "best" estimator depends on the application. For high-quality, offline rendering the estimator of choice will be the one with the quickest convergence. On the other hand, for applications with a time-budget, the estimator with the lowest error for the given budget of samples must be chosen.

4.2.2 Homogenization

The analytical characterization of both, bias as well as variance, across a general set of sampling strategies is challenging. However, focusing on unbiased estimators makes it possible to express error in the form of variance only. To apply this analysis to a wide variety of estimators, it would be beneficial to first be able to transform them into unbiased estimators, albeit by introducing extra variance. One way to mitigate the bias of an estimator is by ensuring that the sampling spectrum only contains energy at the DC [46]. Inspired by the process of generating unbiased estimators from randomly jittering a regular grid [40], Pilleboue et al [39] shifted each realization of the samples by uniformly random amounts. They termed this process *homogenization*. In the point processes literature homogenization refers to *stationary point processes* for which the average number of points per some unit of extent such as length, area, volume, or time, is constant depending on the underlying mathematical space.

4.3 Monte Carlo integration in the Fourier domain

Monte Carlo sampling can be represented as an inner product [14] of the integrand $f(x)$ and a random sampling pattern $\mathbf{S}(x)$ which is composed of a sum of impulses (delta functions) normalized by the number of impulses (see Sec. 2.8). Since inner products are preserved under Fourier Transforms, the naïve estimator can be expressed in the frequency domain as

$$\tilde{\mu}_{\mathrm{N}} \quad = \quad \int_{\mathscr{D}} f(x)\mathbf{S}(x)\ \mathrm{d}x \quad = \quad \int_{\Omega} \hat{\mathbf{S}}(\omega)\,\hat{f}^{*}(\omega)\ \mathrm{d}\omega, \tag{4.3}$$

where $\hat{f}^{*}(\omega)$ is the complex conjugate of the Fourier transformed integrand. Since the goal is to represent the spectrum of a stochastic process, which is generally not integrable over the infinite domain, it is important that the integral is restricted to the finite domain $\mathscr{D}$. The definite integral can either be viewed as a multiplication by a box of the appropriate dimensions [46] or as a portion of a periodic function on a toroidal domain [39]. The former leads to a 'virtual' integrand which subsumes the convolution by a sinc(()the Fourier transformed box), while the latter leads to a discrete Fourier spectrum (see Sec. 2.4). This reformulation of numerical integration provides a fresh perspective on the analysis of error. The derivation of MSE [14], convergence [39], bias and variance [46] in terms of the statistics of the Fourier spectra of the sampling function and the integrand provide valuable insight into the design of adaptive sampling strategies.

4.3.1 Qualitative assessment using periodograms

Following the seminal work by Robert Ulichney on *blue noise* dithering [47], sampling patterns with minimal energy in the low frequency zone of their power spectra are believed to be preferable,

for applications such as stippling and numerical integration. Tools such as the *periodogram* and its radially-averaged profile were introduced to the graphics community about three decades ago, and were primarily used for qualitative comparisons of sampling patterns. These tools have been used to compare the qualities of blue noise sampling patterns generated using various algorithms, and have been extended for comparisons between spatially-varying sampling distributions [50].

4.3.2 Quantifying error using the statistics of sampling spectra

More recently, the focus has been on quantification of estimation error in the Fourier domain. That is, the error of an estimator (e.g. $\tilde{\mu}_1$ for I in equation 4.2) has been expressed in terms of the Fourier spectra of the integrand and sampling function. Durand [14] derived MSE as the inner product of the spectral densities of the integrand and sampling function

$$\mathrm{E}\left[(I-\tilde{\mu}_1)^2\right] = \int_\Omega \mathrm{E}\left[|\hat{\mathbf{S}}(\omega)|^2\right]\ |\hat{f}(\omega)|^2\ \mathrm{d}\omega = \int_\Omega \mathrm{E}\left[\mathscr{P}_{\mathbf{S}}(\omega)\right]\mathscr{P}_f(\omega)\ \mathrm{d}\omega, \tag{4.4}$$

and verified that the convergence rate of the MSE of $\tilde{\mu}_\mathrm{N}$ is indeed $1/N$. While Durand's work focused on the case of random sampling, more recently, Pilleboue et al [39] showed that this holds for an entire class of (homogenized) sampling patterns. Further, they showed that since homogenization eliminates bias, that this error is exactly the variance of the N-sample homogenized estimator. i.e.

$$\mathrm{V}\left[\tilde{\mu}_\mathrm{N}\right] = \int_\Omega \mathrm{E}\left[|\hat{\mathbf{S}}(\omega)|^2\right]\ |\hat{f}(\omega)|^2\ \mathrm{d}\omega = \int_\Omega \mathrm{E}\left[\mathscr{P}_{\hat{\mathbf{S}}}(\omega)\right]\mathscr{P}_f(\omega)\ \mathrm{d}\omega. \tag{4.5}$$

In earlier work, Subr and Kautz [46] expressed the bias and variance of $\tilde{\mu}_\mathrm{N}$ separately in terms of the spectra of the sampling pattern and integrand. They derived

$$\mathrm{E}\left[I-\tilde{\mu}_\mathrm{N}\right] = I - \int_\Omega \mathrm{E}\left[\hat{\mathbf{S}}(\omega)\right]\ \hat{f}(\omega)\ \mathrm{d}\omega, \quad \text{and} \tag{4.6}$$

$$\mathrm{V}\left[\tilde{\mu}_\mathrm{N}\right] = \int_\Omega \mathrm{V}\left[\hat{\mathbf{S}}(\omega)\right]\ |\hat{f}(\omega)|^2\ \mathrm{d}\omega \tag{4.7}$$

as the expressions for bias and variance respectively. i.e. An estimator's bias is dependant on the relationship between the expected Fourier spectrum of the sampling pattern and the integrand's spectrum. The second equation, shows that the estimator's variance is bounded by the variance present in the sampling spectrum relative to the spectral density of the integrand.

4.3.3 Quantifying convergence using periodograms

The primary advantage of studying the variance of the homogenized estimator is that it enables the derivation of convergence rates of various stochastic sampling patterns. To derive the convergence rate of the variance, the *radially averaged* expected power spectra $\mathrm{E}\left[\breve{\mathscr{P}}_{\mathbf{S}}(\rho)\right]$ are used for mathematical convenience. While dealing with sampling patterns in a 2D domain, $\breve{\mathscr{P}}_{\mathbf{S}}(\rho) = \int\int \delta_\rho(\omega_x,\omega_y)\ \mathscr{P}_{\mathbf{S}}(\omega_x,\omega_y)\ \mathrm{d}\omega_x\ \mathrm{d}\omega_y$ where $\delta_\rho(\omega_x,\omega_y) = 1$ if $\omega_x^2+\omega_y^2 = \rho^2$ and zero otherwise. The analytical expressions of radial power spectra are not known for many sampling strategies, but some mathematical tools have been proposed [39] to approximate the bounds of their sampling power spectra. These approximated bounds can be used directly in equation (4.5) to obtain convergence rates of the variance of homogenized estimators.

4.4 Analysis beyond the canonical domain

In rendering, Monte Carlo integration is not restricted to the Euclidean domain. For example, direct and global illumination methods involve integration over a visible hemisphere of directions at the surface hit points, and over the sphere of directions in the presence of participating media within the scene. In this section we discuss a variety of such contexts. In particular, we focus on a closed form formulation for variance in the (hemi-)spherical domain.

4.4.1 Spherical domain

On the sphere, the most commonly used tool for spectral analysis is spherical harmonics (SH) [17], which is analogous to Fourier analysis on a (hyper-) plane. While the bases for Fourier analysis are chosen to be Eigenfunctions of the derivative operator on the plane, SH bases are chosen to be Eigenfunctions of the rotation invariant differential operator (Laplacian) on the sphere and are

$$Y_l^m(\theta,\phi) \equiv \sqrt{(2-\delta_{0m})\frac{(2l+1)}{\mu(\mathscr{S}^2)}\frac{(l-m)!}{(l+m)!}}\,P_l^m(\cos\theta)\exp(im\phi)\,. \tag{4.8}$$

Here, δ_{ij} is the Kronecker delta function, $\mu(\mathscr{S}^2)=4\pi$, is the Lebesgue measure of a unit sphere, $Y_l^m(\theta,\phi)$ is the spherical harmonic basis function of degree l and order m and $P_l^m(x)$ denotes the *asssociated Legendre Polynomials*, for $x\in[-1,1]$. SH (Y_l^m) are orthonormal basis functions such that any integrable function G on $\mathscr{S}^2$ can be decomposed into SH components. as:

$$G(\mathbf{x}) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\mathscr{S}_G(l,m)\,Y_l^m(\mathbf{x})\,, \tag{4.9}$$

where $\mathscr{S}_G(l,m)$ are the (complex) spectral coefficients of $G(\mathbf{x})$. It can be easily shown that:

$$\int_{\mathscr{S}^2}||G(\mathbf{x})||^2 d\omega = \sum_{l=0}^{\infty}\sum_{m=-l}^{l}||\mathscr{S}_G(l,m)||^2\,. \tag{4.10}$$

which is the Parseval's theorem on the sphere. Analogously, the inner product between any two arbitrary functions, $G(\mathbf{x})$ and $f(\mathbf{x})$ defined over a unit sphere, is related to their corresponding spectral coefficients by:

$$\int_{\mathscr{S}^2} G(\mathbf{x})\,\bar{\mathrm{f}}(\mathbf{x})\,d\mathbf{x} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\mathscr{S}_G(l,m)\cdot\overline{\mathscr{S}_f(l,m)}\,, \tag{4.11}$$

where $\mathscr{S}_G(l,m)=\langle G,Y_l^m\rangle$ is the (l,m)-th spherical harmonic coefficients of $G(\mathbf{x})$. The angular mean power spectrum at a frequency band l is defined as the average energy distributed over different m for a given l, as follows:

$$\breve{\mathscr{P}}_G(l) := \frac{1}{2l+1}\sum_{m=-l}^{l}\|\mathscr{S}_G(l,m)\|^2\,. \tag{4.12}$$

The *mean* angular power spectrum $\breve{\mathscr{P}}_G(l)$ is invariant under a rotation of the coordinate system ([22, 28]), as they contain a sum over all orders m. A spherical harmonic power spectrum can be defined with or without the averaging factor. In our formulation, we prefer to work with the average power per degree, or power spectral density, $\breve{\mathscr{P}}_G(l)$, as this ensures that the spectral coefficients of a spherical Dirac delta function are constant and independent of degree l ([20]).

Based on this background, variance expression for Monte Carlo integration can be derived [39] in the spherical domain as:

$$Var(I_N) = \mu(\mathscr{S}^2) \sum_{l=0}^{\infty} (2l+1) \mathrm{E}\left[\mathscr{P}_{\mathbf{S}}(l)\right] \check{\mathscr{P}}_f(l) \tag{4.13}$$

which gives the final expression for the variance in terms of the angular mean power spectra of both the sampling pattern $\mathbf{S}$ and the integrand f.

4.4.2 Other domains

Recently, there has been interest in performing image synthesis by sampling in the gradient domain [24]. The path-tracing problem is reformulated to cover a multi-dimensional space spanning paths and pixels. Rather than estimating radiance at each pixel, these methods estimate the gradients of radiance in the subspace of pixels. Finally, the image is reconstructed from the estimated gradients. The primary benefit of applying the gradient operator is that it attenuates low-frequency content of the integrand. While jittered sampling reduces error due to attenuation of low-frequencies in the sampling pattern, gradient domain path tracing achieves its benefit by a similar effect in the integrand (gradients). Although the gradient operator amplifies the energy at higher frequencies, it has been observed to remain within practical range at the Nyquist rate.

The spectral analysis of error due to stochastic sampling makes assumptions about the sampling strategy as well as the sampling domain. For extensions to domains with complex topology, or when non-uniform sampling is used, it is possible that other mathematical tools may be required. For example, Wei and Wang [51] proposed a formulation equivalent to Fourier analysis, but based on the distribution of the differentials of sampling locations. The use of this *Differential Domain Analysis* permits the comparison of sampling strategies with different distributions. e. g. two instances of the same sampling strategy, but with different target sampling densities, can be identified to have similar spectral properties. The advantages of the analysis are that it can be performed on general domains, and for non-uniform sampling. However, its invariance to distributions makes it less suitable for the analysis of error in integration.

5. Popular sampling patterns

In this chapter, we will describe sampling strategies that are popularly used in modern renderers. For each of these, we provide insight on its error, explain the contexts where its behavior is best and/or worst and also comment on its expected convergence rate using its homogenized periodogram. For simplicity we will primarily discuss these sampling routines in the context of generating locations within a canonical unit square interval in 2D, and briefly explain how each routine generalizes to higher dimensions.

5.1 Classical

5.1.1 Independent random sampling

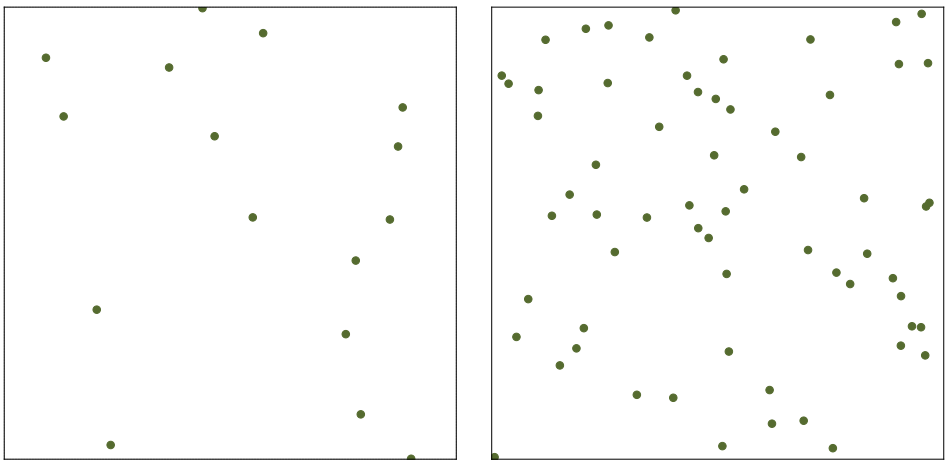


Figure 5.1: Random sampling pattern with $N = 16$ (left) and $N = 64$ samples (right).

Purely independent random sampling is the most basic form of sampling pattern. When uniformly sampling a 2D square domain, the x and y coordinates of each point are computed from independently drawn uniform pseudo-random numbers $\xi \in [0,1)$. New sample points are generated without regard to the previously generated sample points. Figure 5.1 shows an illustration of 16 and 64 independent random samples in a unit square.

One benefit of this approach is that the sampling routine is trivially *progressive*: we can continually add new samples without needing to know how many total sample points we will be generating. It is also trivial to generate samples in arbitrary dimensions by simply drawing a canonical random number ξ for each dimension of a sample point. Unfortunately, however, since samples are drawn completely independently, the resulting sampling patterns many contain many samples which fall in close proximity while large regions of the sampling domain receive no samples.

Independent random sampling leads to the most basic form of Monte Carlo integration. The variance of a Monte Carlo estimator using independent random samples has a convergence rate of $\mathcal{O}(N^{-1})$. This result can be easily derived from the definitions of expected value and variance, but it can likewise be derived in the frequency domain from the periodogram of the sampling pattern.

An important characteristic of this convergence rate is that (in contrast to deterministic quadrature techniques) the dimensionality of the integration problem does not appear in the expression for the convergence rate. Monte Carlo integration using independent random samples therefore does not suffer from the so-called "curve of dimensionality". This is one of the reasons why Monte Carlo techniques are so attractive for high-dimensional integration problems like the rendering of global illumination. Unfortunately, this convergence rate also means that error will always go down relatively slowly: we need four times as many samples to reduce the error by a factor of two.

Due to these limitations, many more sophisticated sampling techniques have been developed over the years which relax the assumption of independence in the hopes of reducing variance or improving convergence rate.

5.1.2 Stratified/Jittered sampling

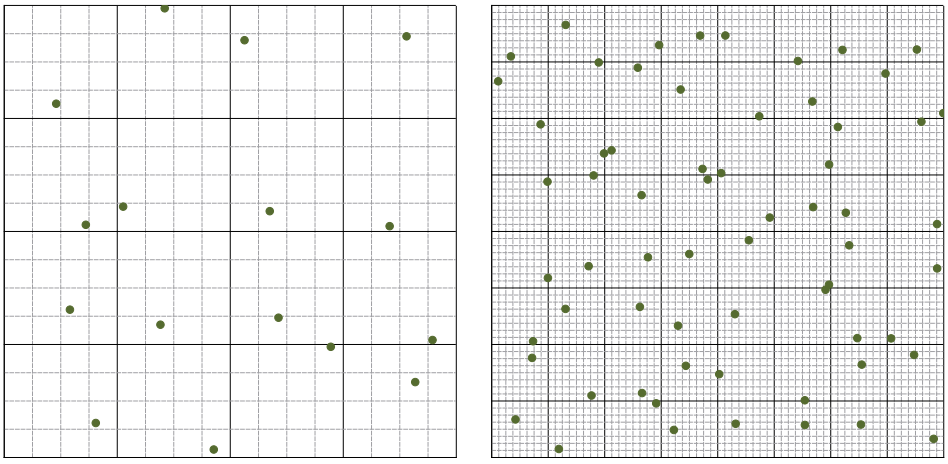


Figure 5.2: Jitter sampling pattern with $N = 16$ (left) and $N = 64$ samples (right).

Stratified sampling divides the integration domain into a number of disjoint regions (a.k.a "strata" or "cells"), and places one or more independent random samples within each region. Jittered sampling, introduced to graphics by Cook and colleagues [11], is a restricted form of stratified sampling where the strata are of equal size with each receiving exactly one sample. Jittered sampling of a 2D unit square involves dividing the square into a grid of $N = M \times M$ square cells, and placing a single sample randomly positioned inside each cell. For 2D, this process would result in N^2 samples. Figure 5.2 shows an illustration of $4 \times 4 = 16$ and $8 \times 8 = 64$ jittered sample points in a unit square.

Jittered sampling considerably reduces the likelihood that many samples will fall within close proximity, and, since each stratum is guaranteed one sample, also places a strict upper bound on the size of any region void of any samples. These two properties considerably improve how well evenly distributed the sample points are compared to independent random sampling. Unfortunately, this improvement comes at the cost of some inconvenience. Jittered sampling is not progressive, so we cannot continually add more samples, and we must know the total number of samples before constructing the point set. Furthermore, the total number of samples is only set indirectly through the number of subdivisions along each dimension. With the same number of subdivisions along each dimension, the total number of samples has to be a perfect square in 2D, a perfect cube in 3D, etc. This can be cumbersome, and becomes increasingly so as the dimensionality of the point set grows. Setting the total number of samples with fine granularity becomes increasing difficult, e.g. dividing each dimension of a five-dimensional domain into two cells results in $2^5 = 32 samples$

and increasing the divisions to three per dimension already increases the sample count to $3^5 = 243$. While it is possible to set an different number of subdivisions along each dimension to obtain finer granularity, it is not easy to determine these values automatically or optimally for a particular integration problem.

It can be shown that jittered sampling cannot increase the variance of a Monte Carlo estimator compared to using independent random samples. In fact, it can often reduce variance considerably. Moreover, analysis of the periodogram reveals that jittered sampling can result in asymptotically better convergence rates of $\mathcal{O}\left(N^{-1.5}\right)$.

5.1.3 Uniform jittered sampling

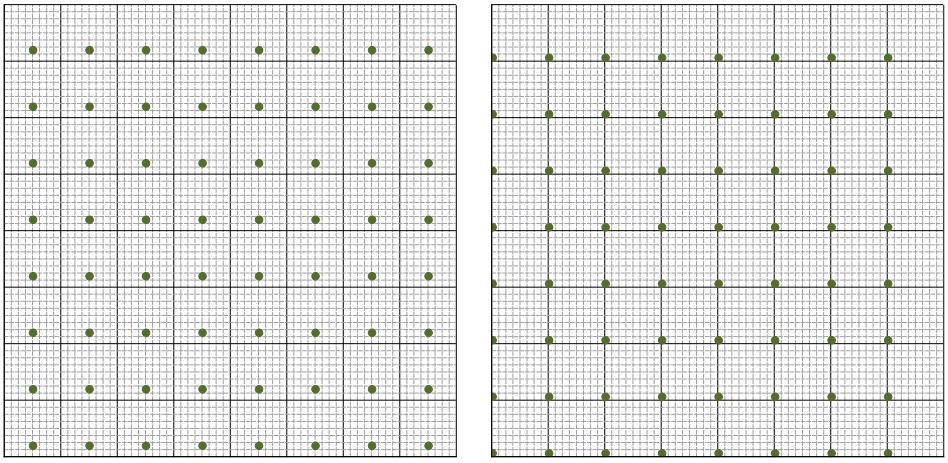


Figure 5.3: Uniform jitter sampling pattern with two realisations of $N = 64$ samples.

Uniform jittered sampling is a further restriction of jittered sampling where the random location of a point within its stratum is the same across the entire point set. In other words, a single jitter is applied to all samples of a regular uniform pattern (as opposed to jittering each stratum's sample independently as in jittered/stratified sampling). For the 2D unit square, any single realization of a uniform jittered sampling pattern is simply a randomly shifted rectangular grid pattern. Figure 5.3 shows two independent realizations of 64 uniform jittered points.

Uniform jitter sampling can be beneficial for some low-dimensional problems in rendering. For instance, Pauly et al. [36] used uniform jittered sampling for performing the one-dimensional line integral of radiance along each camera ray, and Ramamoorthi and colleagues [40] analyzed the error of uniform jittered sampling for integrating visibility from area light sources.

5.1.4 Uncorrelated Jitter, N-Rooks, and Latin hypercube sampling

To overcome the issue of generating jittered sample points in high dimensions, Cook et al. [12] proposed to randomly combine jittered sample sets over subspaces of the full domain. For instance, to generate sample distributions for a nine dimensional space, we could separately partitioning four two-dimensional subspaces (e.g. pixel area, camera lens area, reflection direction, and shadow ray direction) as well as one one-dimensional subspace (time) and then combine the subspace dimensions at random. He called this technique uncorrelated jittering.

In 2D, generating N samples would entail creating two separate 1D jittered point sets X and Y with N samples each, and then using the samples in X as the x-coordinate of each 2D point, and coupling each x with a y from Y in randomly permuted order. This construction ensures that, if we were to divide the 2D domain into N rows and N columns, exactly one 2D sample project into

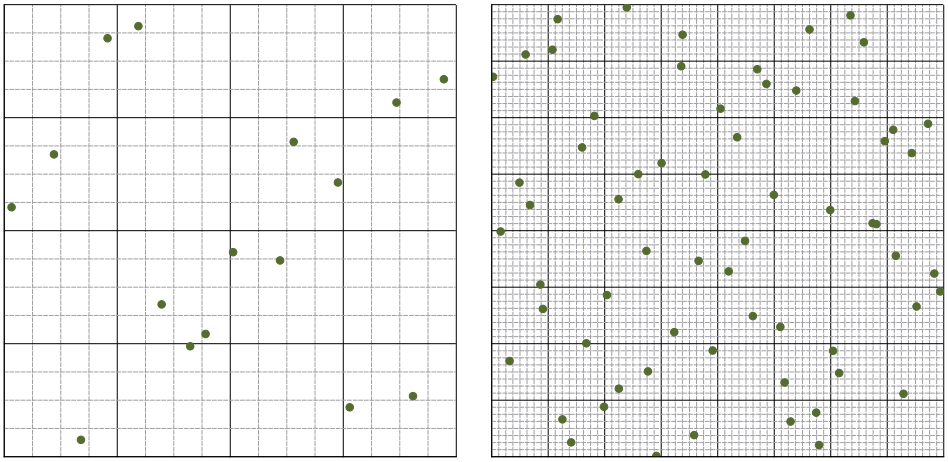


Figure 5.4: N-Rooks sampling pattern with $N = 16$ (left) and $N = 64$ samples (right).

any row or any column. Shirley [43] called this idea N-Rooks sampling since the sample positions correspond to the arrangement of rooks on an $N \times N$ chessboard where none of the rooks can capture another in a single move. Figure 5.4 shows an illustration of N-Rooks sampling patterns with 16 and 64 points in a unit square.

Combining D randomly permuted 1D jittered samples in this way for arbitrary dimension D is called Latin hypercube sampling. Like jittered sampling, we need to know the total number of samples in advance, so this sampling approach is not progress. However, we can easily use an arbitrary number of samples, and no longer have to deal with the difficulty of specifying a $N = M^D$ samples for a D dimensional domain.

Jittered sampling and Latin-hypercube sampling are related, but offer different stratification guarantees, and therefore have different variance reduction behavior. As an illustrative example, a 2D jittered sampling pattern with N points will ensure that few samples can clump together in 2D, but as many as $\sqrt{N}$ samples may project to exactly the same location along either of the two axes (in other words, the projections actually correspond to $\sqrt{N}$ strata with $\sqrt{N}$ points in each stratum). On the other hand, a 2D Latin-hypercube sampling pattern with N points provides less guarantees of clumping reduction in 2D, but ensures that along either of the axis projections each of the N sample points falls into a different stratum. Latin-hypercube sampling can lead to better variance and convergence if the integrand does not depend strongly on some of the dimensions. In the extreme, where the integrand depends only on the x coordinate, but not on y, Latin-hypercube sampling would provide better stratification of the relevant dimensions than jittered sampling.

5.1.5 Multi-Jittered sampling

Chiu et al. [9] introduce multi-jittered sampling, which effectively enforces the stratification guarantees of jittered sampling and N-rooks sampling simultaneously: for $N = M \times M$ points in 2D we generate exactly one sample in each of the $M \times M$ strata, but also guarantee that there is exactly one sample in each of the N rows and N columns (projected strata along the coordinate axes). Figure 5.5 shows an illustration of $4 \times 4 = 16$ and $8 \times 8 = 64$ multi-jittered sample points in a unit square. In Figures 5.6, we summarize the Fourier power spectra of these sampling patterns.

5.2 Quasi-Monte Carlo & Low-discrepancy Sampling

Instead of using random samples, Quasi-Monte Carlo averages the integrand at carefully crafted *deterministic* locations to approximate integrals. These locations are specially crafted to minimize

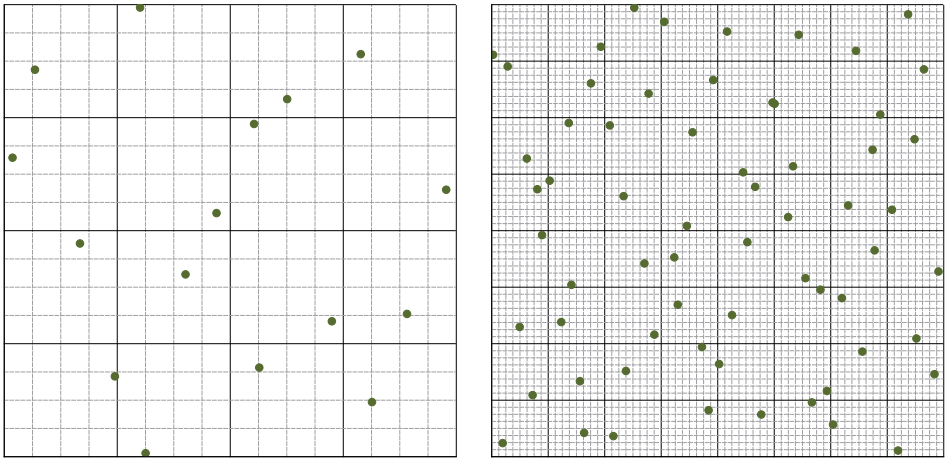


Figure 5.5: Multi-jitter sampling pattern with $N = 16$ (left) and $N = 64$ samples (right).

clumping in the hopes of reducing the variance or improving the convergence rate. One formal definition of such clumping is *discrepancy*, which describes the maximum deviation between the number of sample points in a region compared to the expected number of points that should fall into that sized region. There are a vast number of such low-discrepancy sampling routines [32], but we will briefly summarize just two such approaches as illustrative examples that can be used in rendering.

5.2.1 The van der Corput sequence

The one-dimensional van der Corput sequence is one of the simplest examples of a low-discrepancy sequence. It takes a prime number b as a base, and uses a deterministic procedure to transform a positive integer index i into a value in the unit one-dimensional interval $[0,1)$. The deterministic process that generates these values is often denoted as a function $\Phi_b(i)$ which divides the unit interval into b equal-sized pieces, iterates over these fractional locations, then repeats the process by successively subdividing those regions into b piece as well. For a base of 2, for instance, the interval is split into halves, then fourths, then eights, etc., forming the sequence: $\{1/2, 1/4, 3/4, 1/8, 5/8, 3/8, 7/8, \cdots\}$. In base 3, the sequence would be: $\{1/3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, 8/9 \cdots\}$.

This procedure can be performed for any arbitrary prime base b, and iterating with increasing index i effectively places the next point into the biggest "gap" in the current sequence. This sequence is progressive, so the number of samples need not be known in advance, but it only generates samples in one dimension.

5.2.2 Halton sequence

The Halton sequence has low-discrepancy, so the points tend to be more evenly distributed over the entire domain. It also provides even more strict stratification guarantees than jittered, n-rooks, or multi-jittered sampling, and is a progressive sequence. Figure 5.7 illustrates a Halton sequence with 16 and 64 points in 2D.

5.2.3 Hammersley point sets

By sacrificing progressivity, the discrepancy of the Halton sequence can be further improved by using a regular subdivision of the unit domain for the first dimension, and a Halton sequence for the remaining dimensions. For instance, in 2D, the x dimension would be $\{0/4, 1/4, 2/4, 3/4\}$, and the y dimension would take the van der Corput sequence of base $b = 2$. This modification of the Halton

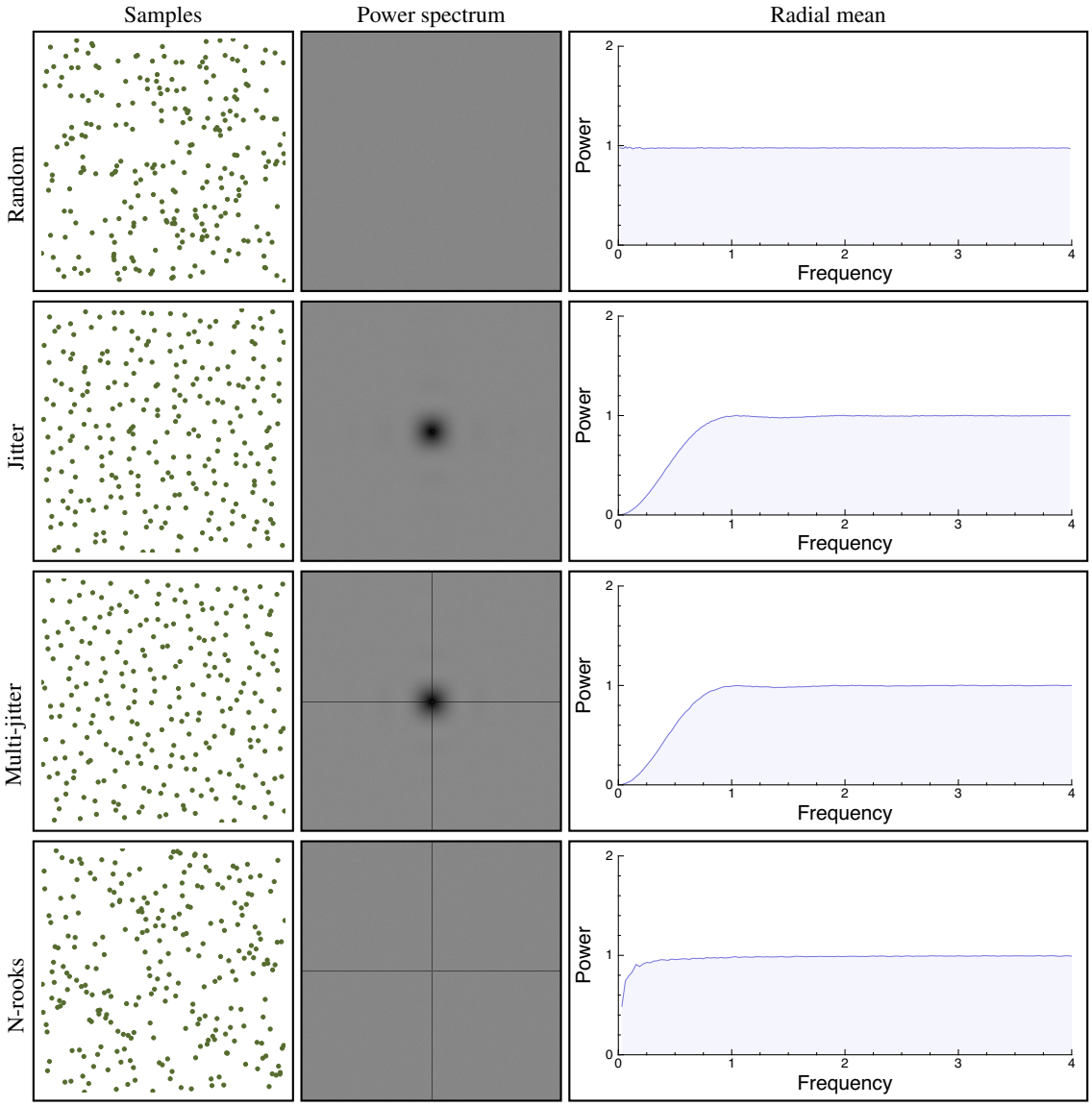


Figure 5.6: Illustration of random and some stochastic grid-based sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

sequence is called the Hammersley sequence, which can create a even lower discrepancy point set for arbitrary dimensions, but due to the first dimension being a regular sampling, knowledge of the number of total samples is necessary. Figure 5.7 illustrates the Hammersley point set with 16 and 64 points in 2D. The corresponding sampling power spectra for Halton and Hammersley samples (first two components) are summarised in Figures 5.8.

5.3 Blue noise

Any sampling pattern with Blue noise characteristics is suppose to be well distributed within the spatial domain without containing any regular structures. The term Blue noise was coined by Ulichney [47], who investivated a radially averaged power spectra of various sampling patterns. He advocated three important features for an ideal radial power spectrum; First, its peak should be at

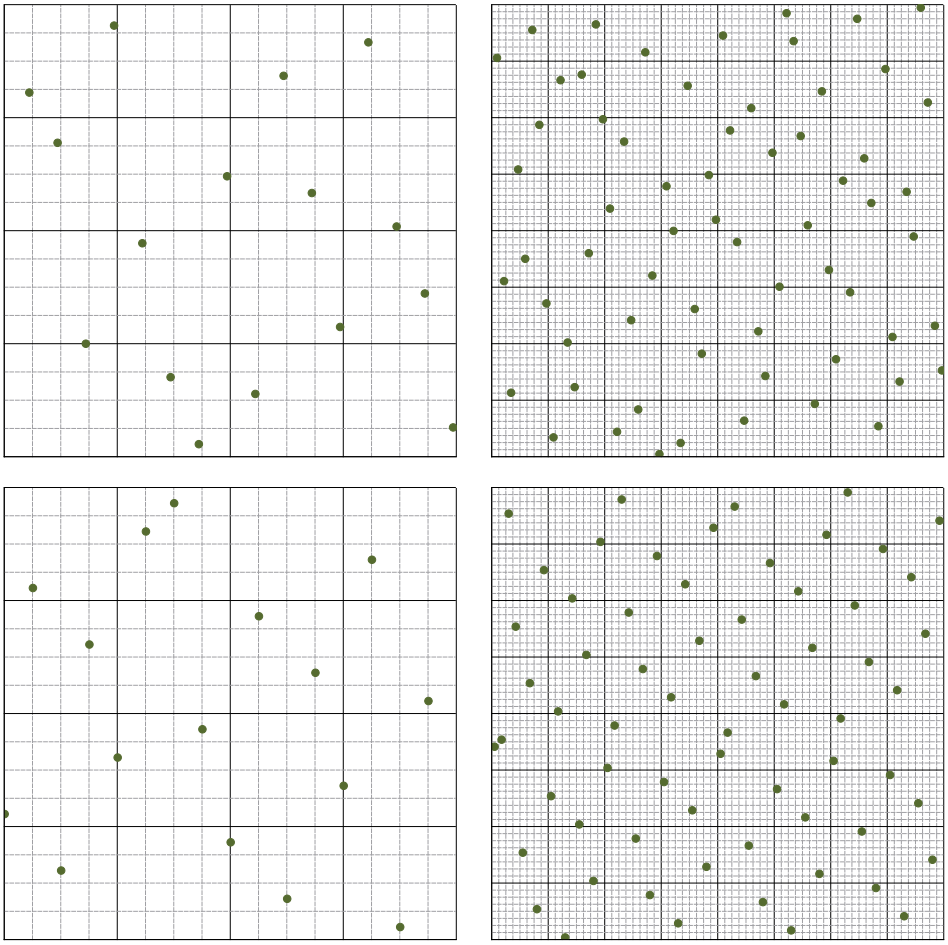


Figure 5.7: Halton (top) and Hammerslay (bottom) sampling pattern with $N = 16$ (left) and $N = 64$ samples (right).

the principal frequency. Second, the principal frequency marks a sharp transition level below which little or no energy exists. And finally, the uncorrelated high-frequency fluctuations are characterized by high-frequency white noise. Poisson Disk sampling pattern (shown in Fig. 5.9) closely resembles the above features in the low frequency zone.

5.3.1 Poisson-disk sampling

Poisson Disk sampling patterns can be generated by simply assigning a disk with minimum radius to each sample point and allowing new samples to get placed only outside the disk radii of existing samples. Cook [11] proposed the first *dart throwing* algorithm for generating Poisson Disk distributed point sets. Random samples are continually tested and only those that satisfy the minimum distance constraint relative to samples already in the distribution are accepted.

For many years, dart-throwing was the only available method for accurate Poisson-disk sampling. Its inefficiency led to the development of approximate Poisson Disk sampling algorithms. For a more comprehensive summary of Poisson sampling methods developed in early years we refer the interested readers to the survey by Lagae and Dutre [26].

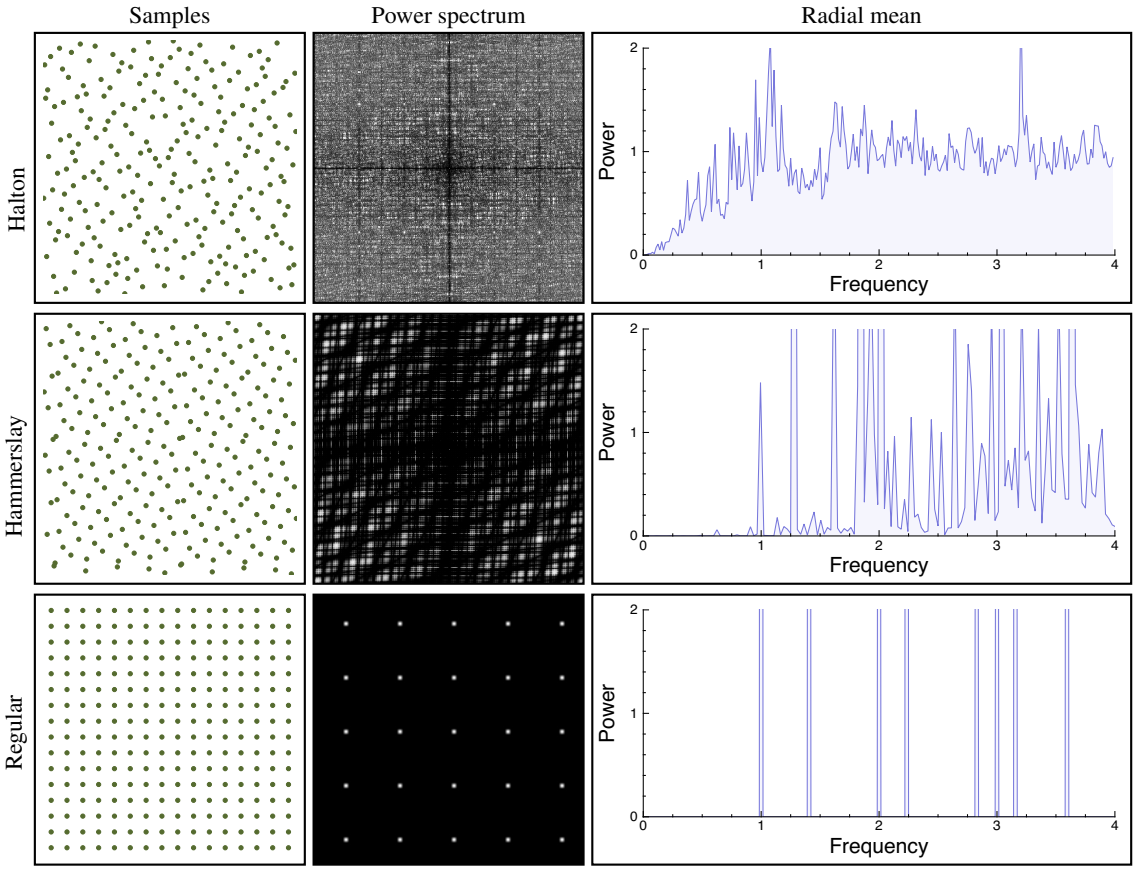


Figure 5.8: Illustration of some well known deterministic sampling patterns with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

5.3.2 Relaxation-based methods

There are many *relaxation* based methods for the generation of blue noise sample distributions. Lloyd [27] which is a powerful and flexible iterative method, is commonly used to enhance the spectral properties of existing distributions of points or similar entities. However, the results from Lloyd's method are satisfactory only to a limited extent. First, if the method is not stopped at a suitable iteration step, the resulting point distributions will develop regularity artifacts. A reliable universal termination criterion to prevent this behavior is unknown. Second, the adaptation to given heterogenous density functions is suboptimal, requiring additional application-dependent optimizations to improve the results.

Balzer and colleagues [4] present a variant of Lloyd's method, termed capacity constrained Voronoi tessellation (CCVT), which reliably converges towards distributions that exhibit no regularity artifacts and precisely adapt to given density functions. Like Lloyd's method it can be used to optimize arbitrary input point sets to increase their spectral properties while avoiding its drawbacks. They apply the so called capacity constraint that enforces each point in a distribution to have the same capacity. Intuitively, the capacity can be understood as the area of the point's corresponding Voronoi region weighted with the given density function. By demanding that each point's capacity is the same, Balzer et al. ensure that each point obtains equal importance in the resulting distribution. This is a direct approach to generating uniform distributions, whereas Lloyd's method achieves such distributions only indirectly by relocating the sites into the corresponding centroids.

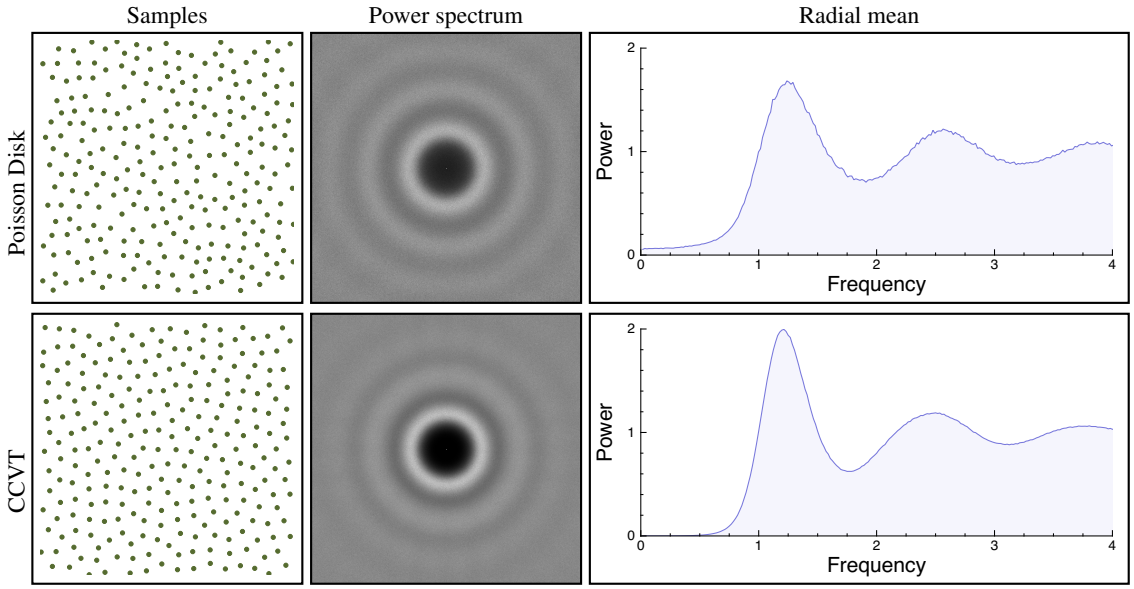


Figure 5.9: Illustration of some well known blue noise samplers with the corresponding Fourier expected power spectra and the corresponding radial mean of their expected power spectra.

5.3.3 Tiling-based methods

There are some tile-based approaches that can be used to generate blue noise samples Tile-based methods overcome the computational complexity of *dart-throwing* and/or *relaxation* based approaches in generating blue noise sampling patterns. In computer graphics community, two tile-based approaches are well known: First approach uses a set of precomputed tiles [10, 25], with each tile composed of multiple samples, and later use these tiles, in a sophisticated way, to pave the sampling domain. Second approach employed tiles with *one sample per tile* [34, 33, 49] and uses some relaxation-based schemes, with look-up tables, to improve the over all quality of samples. Although many blue noise sample generation algorithms exist, none of them are easily extendable to higher dimensions (> 3).

5.4 Interpreting and exploiting knowledge of the sampling spectra

Recently [39], it has been shown that the low frequency region of the radial power spectrum (of a given sampling pattern) plays a crucial role in deciding the overall variance convergence rates of sampling patterns used for Monte Carlo integration. Since blue noise sampling patterns contains almost no radial energy in the low frequency region, they are of great interest for future research to obtain fast results in rendering problems. Surprisingly, Poisson Disk samples have shown the convergence rate of $\mathcal{O}\left(N^{-1}\right)$ which is the same as given by purely random samples. This can be explained by looking at the low frequency region in the radial power spectrum of Poisson Disk samples (Fig. 5.9) which is not zero. The importance of the shape of the radial mean power spectrum in the low frequency region demands methods and algorithms that could eventually allow sample generation directly from a target Fourier spectrum.

5.4.1 Radially-averaged periodograms

Figures 5.6, 5.8 and 5.9 depict radially averaged periodograms of the various sampling strategies described in this chapter. These spectra reveal two important characteristics of estimators built using the corresponding sampling strategies.

First, the absence of energy in certain bands of the spectrum indicate that they will be efficient for integrating functions with complementary spectra. e. g. Jitter, multi-jitter, Halton and Hammerseley exhibit low energy in the low-frequency region. This makes these strategies efficient for estimating integrals of low-frequency functions. Of these, regular sampling contains arbitrarily high values at certain fixed mid- and high- frequencies, making it a poor choice if the integrand contains mid- or high-frequencies. Second, the convergence rate of the corresponding estimators can be derived as described in sec. 4.3.3, if the asymptotic behavior of the radially-averaged periodograms can be characterized analytically. e. g. we expect, from visual inspection that jittered sampling (or multi-jitter) will exhibit improved convergence compared to n-rooks sampling.

Understanding convergence rates

The convergence rate of these sampling patterns may be derived for a special class of functions [8] using tools [39] that allow approixmation of the shape of radial power spectra using simple polynomial profiles. It has been shown that the Poisson Disk sampler, despite its blue noise characteristics, has a convergence rate of random Monte Carlo sampling ($\mathcal{O}\left(N^{-1}\right)$). The radially averaged periodogram of Poisson Disk samples contains non-zero energy in the low frequency (approaching DC) region, implying that averaged periodgram in this region can only be approximated (bounded) using a *constant profile*. This similar to the case for radially averaged periodgram of random samples. Consequently, Poisson Disk samples, when homogenized, do not show improved variance convergence over random sampling. However, for CCVT [4], the radially averaged periodogram approaches zero in the low frequency region as we move towards the DC peak ($\rho = 0$). It has been shown that radially averaged periodogram of CCVT can be approximated by a *quadratic profile (b=2)*, which results in better variance convergence rate [39]. In Table 5.1, we summarise the theoretically-derived convergence rates for some sampling patterns in terms of the dimensionality D of the domain of integration.

Table 5.1: Convergence rates of different stochastic samplers

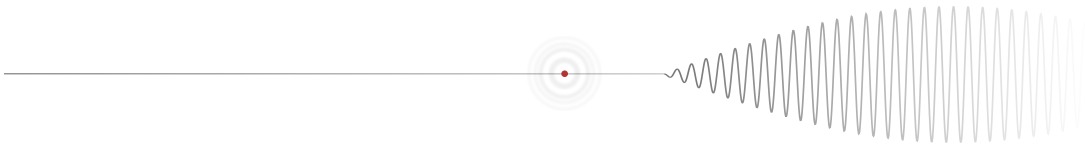
Sampler	Best Case	Worst Case
Random	$\mathcal{O}\left(N^{-1}\right)$	$\mathcal{O}\left(N^{-1}\right)$
Jitter	$\mathcal{O}\left(N^{-1-\frac{2}{D}}\right)$	$\mathcal{O}\left(N^{-1-\frac{1}{D}}\right)$
PoissonDisk	$\mathcal{O}\left(N^{-1}\right)$	$\mathcal{O}\left(N^{-1}\right)$
CCVT	$\mathcal{O}\left(N^{-1-\frac{2}{D}}\right)$	$\mathcal{O}\left(N^{-1-\frac{1}{D}}\right)$

5.4.2 Synthesis of sampling patterns with targeted spectral profiles

Since the theory expresses estimation error in terms of the spectral density of the integrand, indeed it would be desirable to tailor the sampling strategy to the given integrand. Although many methods have been proposed for obtaining spectral information about the integrand in special cases [16, 45, 40, 3] as well as more generally [15, 5], this information has only been used to adapt sampling rates and to design improved reconstruction filters. There has also been some work in tailoring sampling particularly for the problem of anti-aliasing [1, 19]. The next, natural step would be to synthesise customised sampling strategies, adaptively, for general integrands.

The importance of having *spectral control* on sampling methods has been explored. Many methods are able to generate samples that satisfy pre-assigned Fourier characteristics. Parker and colleagues [35] were the first to propose an algorithm for manipulating the power spectra of halftone patterns with blue-noise characteristics. They developed an algorithm that could generate samples whose spectral characteristics would match a given (input) step function. Zhou and colleagues [52] constructed point sets matching given target Fourier power spectra, by performing gradient-descent optimization on an energy derived from the autocorrelation function.

6. Software tool



In this chapter we explain the software toolkit provided in this course for empirical error analysis. The tool may be used to:

- obtain visualizations of point-sets generated by a sampling pattern (sec. 6.3.1);
- visualize Fourier spectra (sec. 6.3.2);
- compare errors across Monte Carlo estimators that use different sampling strategies (sec. 6.3.3);
- The source code can also be used with Matlab (sec. 6.2);

Sampling is performed on 2D domains only (unit square) and therefore integrands are 2D as well.

6.1 The C++ tool

The software package is written in C++ and can directly compile on Windows, Linux and Mac OS systems. Matlab scripts are also provided and explained in the next section.

6.1.1 Set up (Ubuntu)

The source code may be downloaded from the git repository `https://github.com/sinbag/EmpiricalErrorAnalysis`. Before building the tool the following need to be installed: cmake, CGAL, ZLIB. On Ubuntu, they may be installed using apt-get.

```
sudo apt-get install cmake
sudo apt-get install libcgal-dev
sudo apt-get install zlib1g-dev
```

To build the tool first create a build subdirectory; then run cmake and make within the build directory. The bash commands are shown below.

```
git clone –recursive git://
cd NAME OF REPO
mkdir build
cd build
cmake ../
make
cd ..
```

The binary for empirical error analysis, ‘eea’, is located in the build directory. The ‘–recursive’ argument automatically clones submodules that are dependencies of the toolkit. Currently the dependencies are: CGAL, OpenMP, Python (only for ‘`--itype Pbrt`’). Executing the binary without any arguments will display a list of options that need to be specified. The command line arguments for eea are divided into four sections in the following order:

```
./build/eea -S ... <-I ...> -A ... -G ...
```

Here,
-S precedes the type of samples to be used and associated parameters.
-I precedes the integrand type and its associated parameters
-A precedes analyzer name along with associated parameters
-G is a general section. e. g. to specify output filenames.

We will refer to these blocks of the command line as sampling, integrand, analysis and general sections. The integrand section is only needed for some types of analyzers.

6.1.2 Parallelization

Intel TBB multithreading library is used for parallelization in the Fourier analyzer. The code has been tested on Mac OS X and Linux and it works fine. FindTBB.cmake is provided within the repository that can help find TBB installation on your system.

6.1.3 Sampling section

The switch '--stype' is used to specify the type of sampling. Valid options for 'stype' are Random, Jittered, Grid, GJittered, BJittered and correspond to random sampling, jittered sampling, sampling using the regular grid, Gaussian jitter and box jittered sampling respectively. GJittered and Bjittered have associated parameters (width of the Gaussian and size of the box respectively) which may be specified using appropriate switches. The full list of currently implemented sampler sections is:

```
-S --stype Random
-S --stype Jittered
-S --stype Grid
-S --stype GJittered --sigma 0.1
-S --stype BJittered –boxwidth 0.2
```

--sigma 0.1: Gaussian has a std.dev. that is 0.1 times resolution of the sampling grid
–boxwidth 0.2: uniform jitter but with width 0.2 times that of standard jittered sampling

GJittered and BJittered result in biased estimators. They tend towards unbiased as sigma approaches infinity and boxwidth approaches unity respectively. In addition to the above samplers, stubs have been provided for Poisson disk, Halton, Latin hypercube and Sobol sampling but the core sampling routines are not implemented.

6.1.4 Integrand section

When an integrand is required for analysis, the switch '--itype' is used to specify the type of integrand. Valid options for 'itype' are QuadPix, PWConstant, Gaussian, Disk and Pbrt. They correspond to integrating a binary quad within a square pixel, piecewise constant function within the unit square, a Gaussian, a disk and regions of an image rendered using PBRT [38]. The full list of options is

```
-I --itype QuadPix --points 0.1 0.2 0.4 0.3
-I --itype QuadPix --random
-I --itype PWConstant --npts 10
-I --itype Gaussian --center 0.3 0.4 --sigma 0.05 0.1
-I --itype Disk --center 0.6 0.25 --rad 0.45
-I --itype Pbrt --epath ... --spath ... --pypath ... --crop 0.5 0.54 0.54 0.54 \
 .                --pbrtstype stratified –refnspp 0 --img pbrt-eea.exr
```

--points: positions on LTRB of unit square. Binary integrand is 1 only inside the quad formed
--random: random --points. Binary integrand is 1 only inside the quad formed by the 4 points
--npts: number of points for piecewise-constant integrand. Constants are set randomly
--center: specifies 2D location of the center in the unit square.
--sigma: standard deviations (along X,Y) of isotropic 2D Gaussian
--rad: radius of disk
--epath: full path to the pbrt executable
--spath: full path to the pbrt scene (.pbrt file)
--pypath: full path to the python script supplied
--crop: crop window within rendered image (PBRT's relative coordinates)
--pbrtstype: implemented options are limited to random and stratified only
–refnspp: number of samples to be used to compute reference image
--img: output (rendered) image filename.

Except for Pbrt, all other integrand types define functions on the unit square. These functions can be evaluated at the points sampled at the points generated by the sampler. The Pbrt integrand is unique in its behaviour that it does not use the samples generated by the sampling section. Instead, it uses its own sampler to generate the samples. The integrand domain is specified using the crop window. All pixels within the crop window are averaged using the PBRT sampler and each call to evaluate the Pbrt integrand results in one such averaged estimate.

6.1.5 Analysis section

The switch '--atype' is used to specify the type of analysis. Valid options for '--atype' are *pts, fourier, mse* and *var*. These options are for visualizing points, for visualizing Fourier spectra of the points, for computing the mean-squared-error and for computing variance. So the full list of currently implemented analysis sections, with associated parameters, is:

```
-A --atype pts --nsamps 16 --nreps 1
-A --atype fourier --nsamps 16 --nreps 1 --tstep 3 --wstep 1.0
-A --atype mse --nsamps 16 --nreps 10
-A --atype var --nsamps 16 --nreps 20
```

--nsamps: number of samples to be drawn from the sampler in each repetition
--nreps: number of repetitions to use in the estimation
--tstep: write intermediate results after 'tstep' repetitions
--wstep: step size for frequency (can be fractional)

With *mse* (resp. *var*), the number of samples used and the resulting mean-squared error (resp. variance) for the integrand specified in the integrand section is written to the output file. The prefix

for the output file is specified in the general section. With *pts*, '--nreps' point sets can be generated that get stored in a '.eps' file whereas, with *fourier*, the expected sampling power spectrum and the corresponding radially averaged power spectrum is generated after '--nreps' specified realizations. (intermediate expected power spectra can also be generated by setting '--tstep'.

6.2 MATLAB script and examples

The main scripts provided are in CollectConvData.m and PlotConvData.m for the generation of data and plots respectively. Helpers are provided in GenSamplStruct.m and GenIntegStruct.m to build structures containing the parameters with which the tool is called. The following snippet describes how data may be generated using the tool from a matlab script.

6.2.1 Usage

```
% use helpers to generate sampling and integrand structures from parameters
s(1) = GenSamplStruct('Jittered', '') ;
s(2) = GenSamplStruct('Random', '') ;
s(3) = GenSamplStruct('BJittered', '–boxwidth .3') ;
i(1) = GenIntegStruct('QuadPix', ['--points ' num2str(rand(1,4))]) ;
i(2) = GenIntegStruct('PWConstant', '--npts 10 --random') ;
% number of samples and repetitions for analyzer
ns = [9 36 100 1024] ;
nr = 50;
ofile = '../out/test';
binfile = '../build/eea' ;
atype = 'var' ;
system (['rm ' ofile '*.txt']) ; % tool appends successive stats to the output file
data = CollectConvData(ns, nr, ofile, binfile, s, i, atype) ;
```

CollectConvData serves two functions. First it sets up the command line arguments for the tool. Then, it collates the output produced by the tool. The resulting collated data may then be plotted using PlotConvData(data).

6.2.2 Example 1: Comparing PBRT integrands

The following matlab script creates two sampling structures and 3 integrands. The variances in the estimates due to all combinations of sampling strategies and integrands are collected for secondary estimators using 4, 16, 64 and 100 samples (30 repetitions for each variance estimation). The output is a convergence plot of all these estimator-integrand combinations and a scatter plot (dual of the convergence plot) based on the the convergence and variance of 1-sample estimator (see fig. 6.4).

```
clear ;
s(1) = GenSamplStruct('Jittered', '') ;
s(2) = GenSamplStruct('Random', '') ;
i(1) = GenIntegStruct('Pbrt', ...
[ '--epath /home/kartic/pbrt/pbrt-v3/pbrt ' ...
'--spath /home/kartic/pbrt/pbrt-v3-scenes/breakfast/breakfast-crop.pbrt '...
'--pypath /home/kartic/Projects/FAS2016/code/Analysis/python/pbrt-cl.py '...
'--crop 0.335 0.36 0.688 0.7 --pbrtstype $$SAMP$$ –refnspp 0 --img pbrt-eea.exr'], ...
legend ) ;

legend = 'bunny' ;
i(2) = GenIntegStruct('Pbrt', ...
[ '--epath /home/kartic/pbrt/pbrt-v3/pbrt ' ...
'--spath /home/kartic/pbrt/pbrt-v3-scenes/breakfast/breakfast-crop.pbrt '...
'--pypath /home/kartic/Projects/FAS2016/code/Analysis/python/pbrt-cl.py '...
'--crop 0.445 0.465 0.48 0.51 --pbrtstype $$SAMP$$ –refnspp 0 --img pbrt-eea.exr'], ...
legend ) ;

legend = 'floor' ;
i(3) = GenIntegStruct('Pbrt', ...
[ '--epath /home/kartic/pbrt/pbrt-v3/pbrt ' ...
'--spath /home/kartic/pbrt/pbrt-v3-scenes/breakfast/breakfast-crop.pbrt '...
'--pypath /home/kartic/Projects/FAS2016/code/Analysis/python/pbrt-cl.py '...
'--crop 0.168 0.2 0.89 0.91 --pbrtstype $$SAMP$$ –refnspp 0 --img pbrt-eea.exr'], ...
legend ) ;

ns = [4 16 64 100] ;
nr = 30;
ofile = '../out/test';
binfile = '../build/eea' ;
atype = 'var' ;

system (['rm ' ofile '*.txt']) ;
data = CollectConvData(ns, nr, ofile, binfile, s, i, atype) ;
```

$$SAMP$$ is replaced, within CollectConvData, by the sampling type corresponding to s(1) and s(2).

The integrands are small patches in images rendered by PBRT. In this example, three different patches are analysed in the same image (scene). The patches are specified using the '--crop' option. The average value within the patch is taken to be one evaluation of the integrand, where each pixel was rendered with an N-sample estimator (N=4,16,64,100). If the aim is to assess the variance at one pixel, the cropwindow needs to be tight enough to contain just that pixel. Although the same scene has been used in this example, patches from different scenes may be compared by modifying the scene parameter.

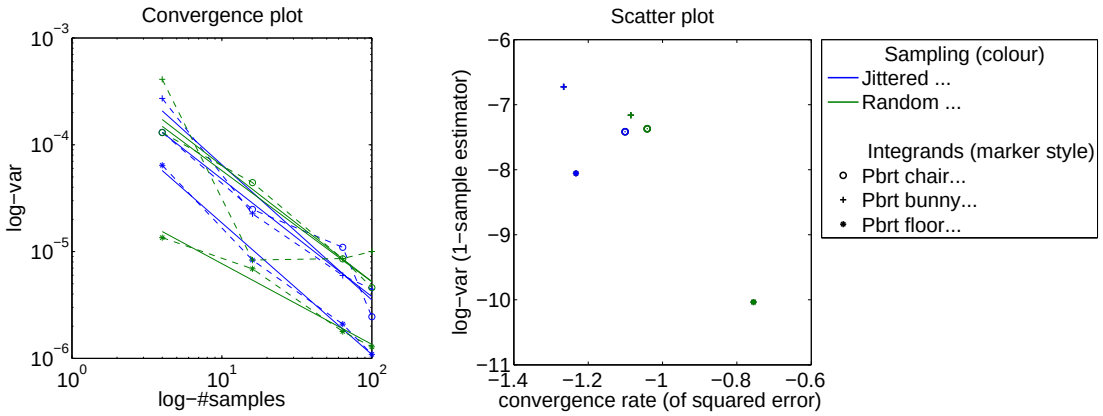


Figure 6.1: Example output plots for the code in sec. 6.2.2

6.2.3 Example 2: Comparing other integrands

```
clear;
% samplers
s(1) = GenSamplStruct('Jittered', '') ;
s(2) = GenSamplStruct('Random', '') ;
s(3) = GenSamplStruct('BJittered', '–boxwidth .3') ;
s(4) = GenSamplStruct('BJittered', '–boxwidth .5') ;
s(5) = GenSamplStruct('GJittered', '--sigma .3') ;
s(6) = GenSamplStruct('GJittered', '--sigma .5') ;

% Integrands
i(1) = GenIntegStruct('QuadPix', ['--points ' num2str(rand(1,4))]) ;
i(2) = GenIntegStruct('PWConstant', '--npts 100 --random') ;
i(3) = GenIntegStruct('Disk', '--center .5 .5 --rad .3') ;


% Analysis Parameters
ns = [9 36 100 1024 4096] ;
nr = 250;
ofile = '../out/test';
binfile = '../build/eea' ;
atype = 'mse' ;

system (['rm ' ofile '*.txt']) ;
tic
data = CollectConvData(ns, nr, ofile, binfile, s, i, atype) ;
toc
```

The above matlab code produces two plots (PDF format) with the prefix specified in ofile (see fig. 6.2).

6.3 C++ examples

The software supports the following analyzers that can be executed directly from a command line:

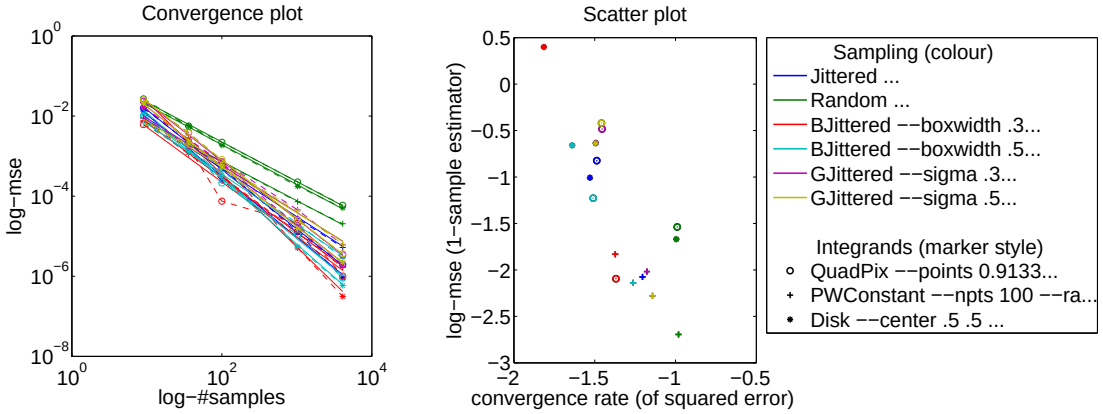


Figure 6.2: Example output plots for the code in sec. 6.2.3

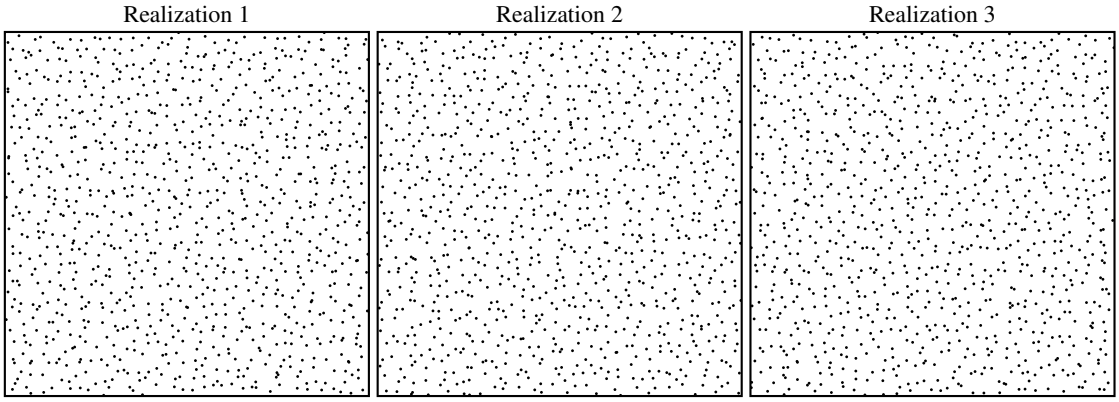


Figure 6.3: Example output point sets for the command line in sec. 6.3.1

6.3.1 Visualizing point sets

For visualizing point sets produced by different sampling strategies, no integrand section needs to be specified. The sampling section describes the type of sampling along with any parameters. The analysis section should be set to the type 'pts'. The general section should specify the prefix for the output file.

```
./build/eea -S --stype Jittered -A --atype pts --nsamps 1024 --nreps 1 -G --ofile points
```

Here,
--stype: sampler type ('Jittered').
--atype: Analyzer type ('pts')
--nsamps: Number of samples ('4096')
--nreps: Number of trials or realizations ('3')
--ofile: Prefix for your output filename ('points')

6.3.2 Visualizing Fourier spectra

For visualizing expected power spectra produced by different sampling strategies, no integrand section needs to be specified. For '--atype *fourier*', the sampling section describes the type of sampling along with any parameters. The analysis section should be set to the type 'fourier'. The general section should specify the prefix for the output file.

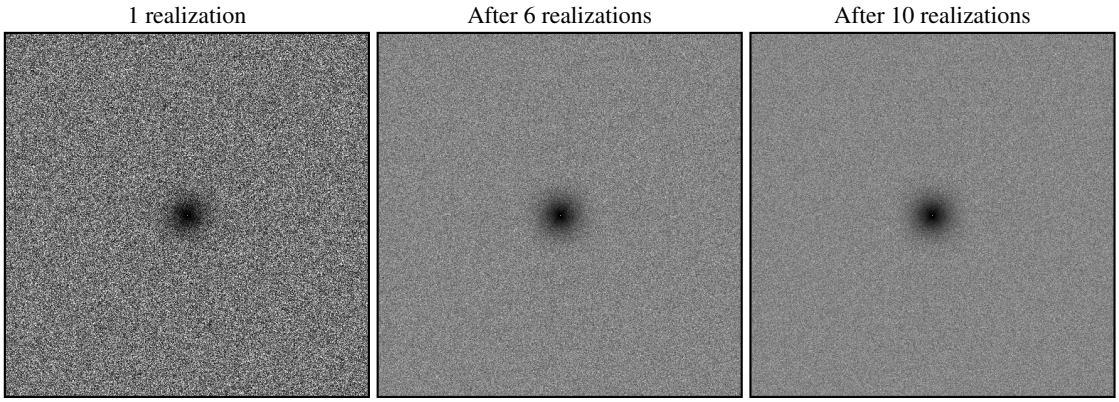


Figure 6.4: Example output expected power spectra averaged over a given number of realiations (for the command line in sec. 6.3.2).

```
./build/eea -S --stype Jittered -A --atype fourier --nsamps 4096 --nreps 10 --tstep 2
--wstep 1 -G --ofile powerspectrum
```

Here,
--stype: sampler type (Jittered, Random, etc.).
--atype: Analyzer type ('fourier')
--nsamps: Number of samples ('4096')
--nreps: Number of trials or realizations ('10')
--tstep: Output intermediate result after given number of trial steps ('2')
--wstep: Frequency step which is 1 for integer frequencies (it can be fractional)
--ofile: Prefix for your output filename ('powerspectrum') (could be anything)

6.3.3 Variance & MSE analyzer

Similar to matlab script, this package can be directly used to perform varianc and MSE analysis. Below we are showing how to execute variance analyzer for analytic functions (Disk, Gaussian, PWConstant and QuadPix).

```
./build/eea -S --stype Jittered -I --itype Gaussian --sigma 0.25 0.25 --center 0.5 0.5 -A
--atype var --nsamps 4096 --nreps 1000 -G --ofile gaussian
```

Here,
--stype: sampler type (Jittered, Random, etc.).
--atype: Analyzer type ('fourier')
--nsamps: Number of samples ('4096')
--nreps: Number of trials or realizations ('10')
--ofile: Prefix for your output filename ('powerspectrum') (could be anything)

To execute a Pbrt integrand, you can use the following command line:

```
./build/eea -S --stype Random -I --itype Pbrt --epath path-to-pbrt-v3-executable/pbrt
--spath path-to-pbrt-v3-scenes/anim-killeroos.pbrt --pypath path-to-python-script/pbrt-
cl.py --crop 0.25 0.75 0.25 0.75 --pbrtstype stratified --refnspp 1 --img pbrt-eea.exr -A
--atype var --nsamps 9 16 25 36 64 --nreps 200 -G --ofile pbrt-killeroos
```

Here,
--epath: pbrt executable path on your machine
--spath: pbrt scene file path
--pypath: path to the python script provided with this package
--crop: pbrt cropwindow coordinates that replaces existing coordinates
--pbrtstype: sampling pattern defined within pbrt
--refnspp: number of samples per pixel to compute the reference image
--img: image filename generated at each trial

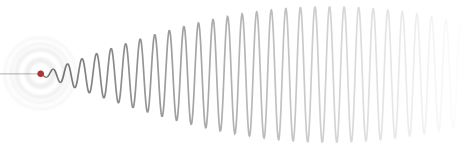
Note that, the '--refnspp 1' is kept to be minimum since reference image is not used to perform variance analysis. Variance computation is performed using an online algorithm that takes as input the mean value from the previous iteration. The *python script* is provided in the package in the 'src/python/' folder.

For MSE analyzer, the reference samples per pixel should be kept higher '--refnspp 4096':

```
./build/eea -S --stype Random -I --itype Pbrt --epath path-to-pbrt-v3-executable/pbrt
--spath path-to-pbrt-v3-scenes/anim-killeroos.pbrt --pypath path-to-python-script/pbrt-
cl.py --crop 0.25 0.75 0.25 0.75 --pbrtstype stratified --refnspp 4096 --img pbrt-eea.exr
-A --atype mse --nsamps 9 16 25 36 64 --nreps 200 -G --ofile pbrt-killeroos
```

Here,
--epath: pbrt executable path on your machine
--spath: pbrt scene file path
--pypath: path to the python script provided with this package
--crop: pbrt cropwindow coordinates that replaces existing coordinates
--pbrtstype: sampling pattern defined within pbrt
--refnspp: number of samples per pixel to compute the reference image
--img: image filename generated at each trial

Bibliography



[1] AHMED, A. G. M., HUANG, H., AND DEUSSEN, O. Aa patterns for point sets with controlled spectral properties. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 212:1–212:8.

[2] ARVO, J., TORRANCE, K., AND SMITS, B. A framework for the analysis of error in global illumination algorithms. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM, pp. 75–84.

[3] BAGHER, M. M., SOLER, C., SUBR, K., BELCOUR, L., AND HOLZSCHUCH, N. Interactive Rendering of Acquired Materials on Dynamic Geometry Using Frequency Analysis. *IEEE Transactions on Visualization and Computer Graphics 19*, 5 (May 2013), 749–761.

[4] BALZER, M., SCHLÖMER, T., AND DEUSSEN, O. Capacity-constrained point distributions: a variant of lloyd's method. *ACM Trans. Graph. 28* (July 2009), 86:1–86:8.

[5] BELCOUR, L., SOLER, C., SUBR, K., HOLZSCHUCH, N., AND DURAND, F. 5D Covariance tracing for efficient defocus and motion blur. *ACM Trans. Graph. 32*, 3 (2013), 31.

[6] BLINN, J. F. Jim Blinn's corner-return of the Jaggy [high frequency filtering]. *IEEE Computer Graphics and Applications 9*, 2 (1989), 82–89.

[7] BRACEWELL, R. N. *The Fourier Transform & Its Applications*, 3rd edition ed. McGraw-Hill, 2000.

[8] BRANDOLINI, L., COLZANI, L., AND TORLASCHI, A. Mean square decay of Fourier transforms in euclidean and non euclidean spaces. *Tohoku Math. J. (2) 53*, 3 (2001), 467–478.

[9] CHIU, K., SHIRLEY, P., AND WANG, C. *Multi-jittered Sampling*. Graphics Gems Series. Academic Press Professional, Inc., San Diego, CA, USA, 1994, pp. 370–374.

[10] COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. Wang tiles for image and texture generation. In *ACM SIGGRAPH 2003 Papers* (2003), SIGGRAPH '03, pp. 287–294.

[11] COOK, R. L. Stochastic sampling in computer graphics. *ACM Transactions on Graphics 5*, 1 (Jan. 1986), 51–72.

[12] COOK, R. L., PORTER, T., AND CARPENTER, L. Distributed ray tracing. *SIGGRAPH Comput. Graph. 18*, 3 (Jan. 1984), 137–145.

[13] DIPPE, M. A. Z., AND WOLD, E. H. Antialiasing through stochastic sampling. In *ACM SIGGRAPH '85)* (1985), B. A. Barsky, Ed., pp. 69–78.

[14] DURAND, F. A frequency analysis of Monte-Carlo and other numerical integration schemes. Tech. Rep. MIT-CSAIL-TR-2011-052, CSAIL, MIT, MA, 2011.

[15] DURAND, F., HOLZSCHUCH, N., SOLER, C., CHAN, E., AND SILLION, F. X. A frequency analysis of light transport. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (July 2005), 1115–1126.

[16] EGAN, K., TSENG, Y.-T., HOLZSCHUCH, N., DURAND, F., AND RAMAMOORTHI, R. Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Trans. Graph. 28*, 3 (July 2009), 93:1–93:13.

[17] GROEMER, H. *Geometric Applications of Fourier Series and Spherical Harmonics*. Cambridge University Press, 1996. Cambridge Books Online.

[18] HACHISUKA, T., JAROSZ, W., WEISTROFFER, R. P., DALE, K., HUMPHREYS, G., ZWICKER, M., AND JENSEN, H. W. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3 (Aug. 2008), 33:1–33:10.

[19] HECK, D., SCHLÖMER, T., AND DEUSSEN, O. Blue noise sampling with controlled aliasing. *ACM Trans. Graph. 32*, 3 (July 2013), 25:1–25:12.

[20] HIPKIN, R. The statistics of pink noise on a sphere: applications to mantle density anomalies. *Geophysical Journal International 144*, 2 (2001), 259–270.

[21] KAMMLER, D. W. *A First Course in Fourier Analysis*, 2nd edition ed. Cambridge University Press, 2008.

[22] KAULA, W. Theory of statistical analysis of data distributed over a sphere. *Reviews of Geophysics 5*, 1 (1967), 83–107.

[23] KELLER, A., PREMOZE, S., AND RAAB, M. Advanced (quasi) monte carlo methods for image synthesis. In *ACM SIGGRAPH 2012 Courses* (New York, NY, USA, 2012), SIGGRAPH '12, ACM, pp. 21:1–21:46.

[24] KETTUNEN, M., MANZI, M., AITTALA, M., LEHTINEN, J., DURAND, F., AND ZWICKER, M. Gradient-domain path tracing. *ACM Trans. Graph. 34*, 4 (2015).

[25] KOPF, J., COHEN-OR, D., DEUSSEN, O., AND LISCHINSKI, D. Recursive wang tiles for real-time blue noise. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), pp. 509–518.

[26] LAGAE, A., AND DUTRÉ, P. A comparison of methods for generating poisson disk distributions. *Comput. Graph. Forum 27*, 1 (2008), 114–129.

[27] LLOYD, S. P. Least squares quantization in PCM. *IEEE Transactions on Information Theory 28* (1982), 128–137.

[28] LOWES, F. Spatial power spectrum of the main geomagnetic field, and extrapolation to the core. *Geophysical Journal International 36*, 3 (1974), 717–730.

[29] MANOLAKIS, D. G., AND INGLE, V. K. *Applied Digital Signal Processing: Theory and Practice*, first edition ed. Cambridge University Press, 2012.

[30] MARKS, R. J. *Handbook of Fourier Analysis & Its Applications*. Oxford University Press, USA, 2008.

[31] MITCHELL, D. P. Generating antialiased images at low sampling densities. M. C. Stone, Ed., vol. 21, pp. 65–72.

[32] NIEDERREITER, H. *Quasi-Monte Carlo Methods*. John Wiley & Sons, Ltd, 1992.

[33] OSTROMOUKHOV, V. Sampling with polyominoes. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (July 2007), 78:1–78:6.

[34] OSTROMOUKHOV, V., DONOHUE, C., AND JODOIN, P. Fast hierarchical importance sampling with blue noise properties. In *ACM Trans. on Graphics* (2004), vol. 23, pp. 488–495.

[35] PARKER, K., MITSA, T., AND ULICHNEY, R. A new algorithm for manipulating the power spectrum of halftone patterns. In *Proceedings of SPSE's 7th International Congress on Non-Impact Printing* (1991).

[36] PAULY, M., KOLLIG, T., AND KELLER, A. Metropolis light transport for participating media. In *Rendering Techniques 2000 (Proceedings of the Eurographics Workshop on Rendering)* (June 2000), pp. 11–22.

[37] PERLIN, K. State of the art in image synthesis. In *ACM SIGGRAPH 1989 Courses Vol. 19* (New York, NY, USA, 1989), SIGGRAPH 1989, ACM.

[38] PHARR, M., AND HUMPHREYS, G. *Physically Based Rendering, Second Edition: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., 2010.

[39] PILLEBOUE, A., SINGH, G., COEURJOLLY, D., KAZHDAN, M., AND OSTROMOUKHOV, V. Variance analysis for Monte Carlo integration. *Transactions on Graphics 34*, 4 (July 2015), 124:1–124:14.

[40] RAMAMOORTHI, R., ANDERSON, J., MEYER, M., AND NOWROUZEZAHRAI, D. A theory of monte carlo visibility sampling. *ACM Trans. Graph. 31*, 5 (Sept. 2012), 121:1–121:16.

[41] ROBERT, C. P., AND CASELLA, G. *Monte Carlo Statistical Methods*. Springer, New York, NY, 2010.

[42] SEBER, G., AND WILD, C. *Nonlinear regression*. Wiley series in probability and statistics. Wiley-Interscience, Hoboken, NJ, 2003.

[43] SHIRLEY, P. Discrepancy as a quality measure for sampling distributions. In *Proc. Eurographics '91* (Sept. 1991), pp. 183–194.

[44] SMITH, A. R. A pixel is not a little square, a pixel is not a little square, a pixel is not a little square!

[45] SOLER, C., SUBR, K., DURAND, F., HOLZSCHUCH, N., AND SILLION, F. Fourier depth of field. *ACM Trans. Graph. 28*, 2 (May 2009), 18:1–18:12.

[46] SUBR, K., AND KAUTZ, J. Fourier analysis of stochastic sampling strategies for assessing bias and variance in integration. *Transactions on Graphics 32*, 4 (July 2013), 128:1–128:12.

[47] ULICHNEY, R. *Digital Halftoning*. MIT Press, 1987.

[48] VEACH, E. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162.

[49] WACHTEL, F., PILLEBOUE, A., COEURJOLLY, D., BREEDEN, K., SINGH, G., CATHELIN, G., DE GOES, F., DESBRUN, M., AND OSTROMOUKHOV, V. Fast Tile-based Adaptive Sampling with User-specified Fourier Spectra. *ACM Trans. Graph. 33*, 4 (July 2014), 56:1–56:11.

[50] WEI, L.-Y., AND WANG, R. Differential domain analysis for non-uniform sampling. *Transactions on Graphics 30*, 4 (2011), 50:1–50:10.

[51] WEI, L.-Y., AND WANG, R. Differential Domain Analysis for Non-uniform Sampling. *SIGGRAPH 2011* (August 2011).

[52] ZHOU, Y., HUANG, H., WEI, L.-Y., AND WANG, R. Point sampling with general noise spectrum. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4 (July 2012), 76:1–76:11.