Sampling Analysis using Correlations for Monte Carlo Rendering

A. Cengiz Öztireli and Gurprit Singh

A. Cengiz Öztireli & Gurprit Singh

November 2018

Contents

Listings

Abstract

Point patterns and stochastic structures lie at the heart of Monte Carlo based numerical integration schemes. Physically based rendering algorithms have largely benefited from these Monte Carlo based schemes that inherently solve very high dimensional light transport integrals. However, due to the underlying stochastic nature of the samples, the resultant images are corrupted with noise (unstructured aliasing or variance). This also results in bad convergence rates that prohibit using these techniques in more interactive environments (e.g. games, virtual reality). With the advent of smart rendering techniques and powerful computing units (CPUs/GPUs), it is now possible to perform physically based rendering at interactive rates. However, much is left to understand regarding the underlying sampling structures and patterns which are the primary cause of error in rendering.

This course surveys the most recent state-of-the-art frameworks that are developed to better understand the impact of samples' structure on the error and its convergence during Monte Carlo integration. It provides best practices and a set of tools for easy integration of such frameworks for sampling decisions in rendering. We revisit stochastic point processes that offers a unified theory explaining stochastic structures and sampling patterns in a common principled framework. We show how this theory generalizes spectral tools developed over the years to analyze error and convergence rates, and allows for analysis of more complex point patterns with adaptive density and correlations. At the end of the course, the audience will have a comprehensive understanding of both theoretical and practical aspects of point processes that would guide them in choosing and designing sampling strategies for applications specific to Monte Carlo rendering. A codebase and web application for easy use of the introduced techniques will also be made available on https://github.com/sinbag/SamplingAnalysisWithCorrelations.

Intended Audience and Prerequisites

This course is structured for researchers, practitioners, and graduate students in the field of Monte Carlo rendering, who would like to get a theoretical and practical understanding of how to utilize methods from stochastic point processes to enhance offline, interactive and/or real-time rendering systems.

We assume a basic understanding of statistics, signal processing, and calculus that can be obtained via undergraduate introduction level courses. In particular, we assume no background on stochastic point processes, Fourier theory or Monte Carlo integration, which in general would be a core application for demonstration.

About the Lecturers

Dr. A. Cengiz Öztireli

is currently a Research Scientist at Disney Research Zürich. His research interests are in computer graphics, vision, and machine learning. With his collaborators from academia and industry, he has been publishing works in international journals and conferences. He obtained his M.S. and Ph.D. degrees in computer science from ETH Zurich (jointly funded by the Swiss National Science Foundation), and completed a double major in computer engineering and electronics engineering at Koc University (valedictorian). He has been honored with several awards including the Eurographics Best Ph.D. Thesis Award and Fulbright Science and Technology Award.

Dr. Gurprit Singh

is currently a post doctoral research associate at Max Planck Institute for Informatics, Saarbrücken. His research is focused on analyzing the impact of different sampling patterns on Monte Carlo based rendering algorithms. His recent works include developing advanced Fourier domain tools for convergence analysis that also results in new design principles for novel futuristic samplers. He obtained his Ph.D. in 2015 from the Université de Lyon 1, France, under the supervision of Prof. Victor Ostromoukhov which was followed by a two year post-doc at Dartmouth College with Prof. Wojciech Jarosz.

Course Syllabus

The course is of duration 1 hour and 45 minutes, and will follow the following schedule, after the structure of this script.

Part	Topics	Duration
Introduction	Point Patters in Computer Graphics Stochastic Point Processes Signal Processing This Course	15 min.
Background	Probability General Point Processes Fourier Transform Discrepancy Theory Numerical Integration	15 min.
Break and Q/A Session 5 min.		
Sampling Analysis	Spatial Measures Spectral Measures Discrepancy	25 min.
Error Analysis in Monte Carlo Integration	General Point Patterns Non-adaptive Point Patterns Adaptive Point Patterns	35 min.
Conclusions	Recent Developments and Future Directions Resources for Stochastic Point Processes	5 min.
Q/A Session 5 min.		

1. Introduction

1.1 Point Patterns in Computer Graphics

Sampling patterns with points distributed according to certain rules but otherwise randomly arise in many applications in computer graphics including anti-aliasing, representing and integrating functions for rendering, image, video, and geometry processing, physically-based simulations, crowd simulations, texture representations and synthesis, non-photorealistic rendering, stippling and halftoning, or modeling natural distributions.

We show examples of some of these applications in Figure 1.1. Although the outputs are quite different for these cases, a common pattern can be observed: they all rely on random structures that follow certain rules. This naturally leads to the idea of using statistical measures for handling the problems encountered. However, classical statistics are of limited utility for these cases. A classical example of limitations of simple statistics is depicted in Figure 1.2, left. Here, the one on the left is a completely random distribution, and the other is a so-called blue noise distribution. There is a clear visual difference in how points are distributed for these two cases. However, if we simply use the common statistical measure of expected local density, e.g. count the number of points falling into an arbitrary fixed circle, and take the average of those numbers over many different random or blue noise distributions, we will get a constant number for both cases. Thus, we cannot explain how a random distribution as on the left is different from the one on the right by considering such simple measures.

As a result, traditionally, practitioners of computer graphics utilize a different set of sophisticated measures and techniques to handle problems in different applications. However, it has also been observed that many problems can be interpreted as analyzing and synthesizing stochastic structures represented with point samples. This observation naturally leads to utilizing ideas and techniques from the field of stochastic point processes, which provides a comprehensive theory and practical set of techniques to handle general point patterns.

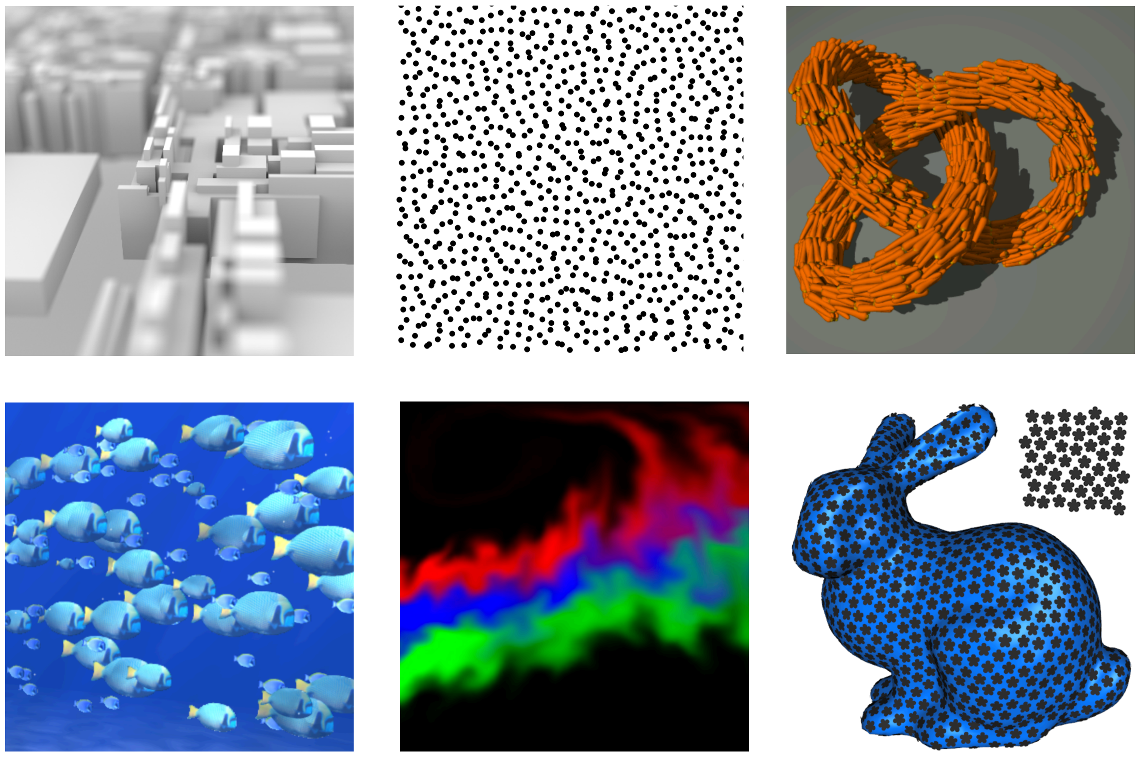


Figure 1.1: Point patterns arise in several applications in computer graphics. Some examples are (from left to right, top to bottom): rendering, sampling for anti-aliasing, object tiling, dynamic distributions, physical simulations, fabrication (the images are from, respectively, [9, 15, 16, 20, 21, 25]).

1.2 Stochastic Point Processes

Point processes are mathematical models for underlying processes that generate random point distributions with certain characteristics. Some examples of generating processes are natural processes such as the environmental factors that result in certain distributions of trees, or algorithms that generate random sampling patterns for rendering. Although each of the point distributions generated by a point process is different, they all share common characteristics implied by the point process.

The theory of stochastic point processes offers a unified and principled treatment of such point patterns. The main goal of this field is developing specialized statistical tools for analyzing and synthesizing point distributions. Each point in a distribution is assumed to have a random location and a mark associated with it. The correlations among point locations and marks then give rise to a pattern. It is thus central to point processes to analyze these correlations, to reveal the underlying patterns. A familiar example is a regular grid of arbitrary orientation and location as in Figure 1.2, right: the locations of points are highly correlated such that knowing only two points of the grid is enough to deduce the locations of the rest of the points.

Techniques in stochastic point processes start from very general assumptions and theorems from probability and measure theory. These are then specialized for common and real-world cases, which are of great interest for the computer graphics community. Physicists and statisticians have already been using such versions of the theory and associated techniques to represent and understand many real-world structures for decades. We thus

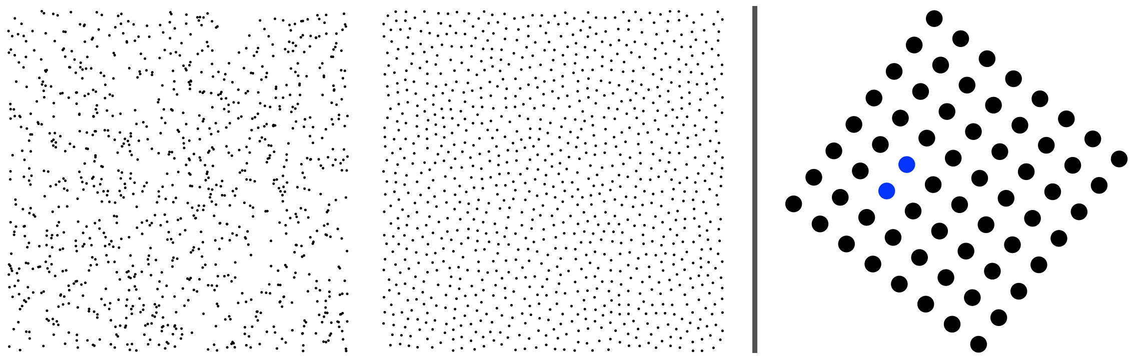


Figure 1.2: (Left) The two distributions have the same density of points, while they look quite different. We need to consider correlations among point locations to characterize such distributions. (Right) As the points in a regular grid are highly correlated, knowing the locations of the two blue points is enough to exactly get the locations of all other points.

now have a wealth of ideas to be adopted, adapted, and improved upon to solve some of the fundamental problems in graphics.

1.3 Signal Processing

Signal processing is a branch of science that looks at each function as a signal and helps analyze, synthesize and modify these signals based on rigorous Fourier and Wavelet basis. From early times in computer graphics, images and the underlying functions are studied as signals using Fourier basis and has played a crucial role in both faithful reconstruction and integration of signals. This course uses stochastic point processes to generalize the tools from signal processing to give a general perspective on how to study error due to different sampling patterns in Monte Carlo rendering.

1.4 This Course

This course presents a thorough introduction to stochastic point processes as used for point pattern analysis in rendering, as well as algorithms to compute and utilize the associated tools in practice. The framework introduced unifies and generalizes all previous measures in spatial and spectral domains for error analysis in Monte Carlo rendering, and hence the course differs from previous SIGGRAPH courses on rendering or spectral domain error analysis.

2. Background

This section is dedicated to provide an applied view of stochastic point processes from a rendering perspective. We will cover the theoretical underpinnings on an accessible level, and without going into extensive technicalities. In particular, although the whole theory extends to measure spaces, we will be dealing with points living in Euclidean spaces, as many problems can be interpreted with Euclidean embeddings. For a more complete theoretical exposition, we refer the reader to excellent books on point processes [11, 18]. We will provide an extensive and annotated set of references for the point processes literature in Section 5.2.

2.1 Probability

For reference, we start by reviewing some basic concepts from probability and statistics that are required to understand the next chapters. For in-depth discussions on these concepts, please refer to standard texts on probability and statistics. This chapter also introduces some of basic notations we will use throughout this work. We assume a basic understanding of calculus. We refrain from using technical definitions. If some technical concepts are not familiar, they can be safely ignored to follow the discussion.

2.1.1 Random Variables

A random variable X is a variable whose value depends on the outcome of a random event. A familiar example is assigning a number to the outcome of coin flipping. We can assume that for a head the value of the random variable is 1, and for a tail it is 0. Hence, the value of X depends on the random outcome of flipping the coin. We can define the probability $\mathbb{P}(X = x)$ of X having a particular value x. In this example, for a fair coin, we can set $\mathbb{P}(X = 0) = \mathbb{P}(X = 1) = 0.5$.

For our purposes, we will be mostly dealing with continuous random variables. For such a random variable X, we can define a probability density function (PDF) $p_X(x)$. The PDF $p_X(x)$ is a non-negative Lebesgue-integrable function. The probability of X having a range of values is given by $\mathbb{P}(x_1 \leqslant X \leqslant x_2) = \int_{x_1}^{x_2} p_X(x)dx$. In general, the random variable X can have values in a multi-dimensional space. In this case, we write $p_X(\mathbf{x})$, where lower-case bold letters are used to express vectors. Similar to the one-dimensional case, we can define the probability that X belongs to some Borel set $\mathcal{B}$ by $\mathbb{P}(X \in \mathcal{B}) = \int_{\mathcal{B}} p_X(\mathbf{x})d\mathbf{x}$. If the range of values that X can take lie in a domain $\mathcal{D}$, it is then true that $\mathbb{P}(X \in \mathcal{D}) = \int_{\mathcal{D}} p_X(\mathbf{x})d\mathbf{x} = 1$.

We can also regard a multi-dimensional random variable as multiple random variables. Then, we can talk about joint probabilities that describe how these random variables depend on each other. For example, for two random variables X_1 and X_2, $\mathbb{P}(X_1 \in \mathcal{B}_1, X_2 \in \mathcal{B}_2)$ gives the joint probability of finding X_1 and X_2 in $\mathcal{B}_1$ and $\mathcal{B}_1$, respectively. For the continuous case, we can similarly write the joint PDF as $p_{X_1,X_2}(x_1, x_2)$. As in the notation above, we can equivalently denote this as $p_X(\mathbf{x})$, where X denotes the two random variables, and $\mathbf{x}$ the samples in the corresponding two-dimensional space.

2.1.2 Expected Value, Mean, Variance, and Covariance

For a continuous random variable X with support $\mathcal{D}$ as above, the expected value of X is defined by the integral $\mathbb{E}_X[X] = \int_{\mathcal{D}} \mathbf{x} p_X(\mathbf{x})d\mathbf{x}$. A function $f(X)$ of the random variable X is also a random variable. Hence, we can similarly define the expected value of $f(X)$ as $\mathbb{E}_X[f(X)] = \int_{\mathcal{D}} f(\mathbf{x})p_X(\mathbf{x})d\mathbf{x}$. This expected value can also be called the *mean* of $f(X)$, as it is the mean of the values $f(X)$ can take, as X takes on its random values with probabilities proportional to the PDF p_X.

To measure how much $f(X)$ varies as we take random samples from X, *variance* is defined as $var_X[f(X)] = \mathbb{E}_X[(f(X) - \mathbb{E}_X[f(X)])^2] = \mathbb{E}_X[f^2(X)] - (\mathbb{E}_X[f(X)])^2$. Hence, variance actually computes the expected value of squared distances of $f(X)$ from its mean. We often drop the subscripts X and simply write $\mathbb{E}[f(X)]$ and $var[f(X)]$ for the mean and variance in the rest of this work. Finally, we can define how more than one random variable vary together by defining the covariance matrix $\mathbf{C}$, with elements given by $C_{ij} = \mathbb{E}[X_iX_j] - \mathbb{E}[X_i]\mathbb{E}[X_j]$, where X_i's are scalar random variables. This is a multi-dimensional generalization of variance, where the diagonal entries capture individual variances $C_{ii} = var_{X_i}(X_i)$, and the non-diagonals capture dependencies among the random variables. Here, $\mathbb{E}[X_i]$ is defined as before, and $\mathbb{E}[X_iX_j] = \int_{\mathcal{D}\times\mathcal{D}} x_i x_j p_{X_i,X_j}(x_i, x_j)dx_i dx_j$.

2.2 General Point Processes

Intuitively, a point process is a generating process for a set of point distributions with common characteristics. Hence, each distribution generated by a point processes can be regarded as a realization of that point process. To characterize a point process, we can compute statistics over different realizations. To capture how realizations vary, the study of point processes starts by assigning a random variable $X(\mathcal{B})$ to each Borel set $\mathcal{B} \in \mathcal{D}$ for a given domain $\mathcal{D}$. This implies that to characterize a point process, we actually need *infinitely many* number of random variables, each of which is assigned to one of the infinitely many sets in the given domain.

In order to make the analysis tractable, we can assume that we select a number of sets $\mathcal{B}_1$ to $\mathcal{B}_n$. We can then define the joint probability $\mathbb{P}(X(B_1) \leqslant b_1, \cdots, X(B_n) \leqslant b_n)$. A point process can be formally described by characterizing this probability for *all* different n and group of sets $\mathcal{B}_i$.

The most commonly used random variable is the number of points $N(\mathcal{B})$ that fall into the set $\mathcal{B}$. This is indeed a random variable: for each realization of the point process, i.e. a point distribution, this number changes randomly for a fixed $\mathcal{B}$. Hence, for this $\mathcal{B}$, by generating different distributions from the same point process, we can compute the PDF of $N(\mathcal{B})$. We illustrate this idea in Figure 2.1 on three distributions generated by a

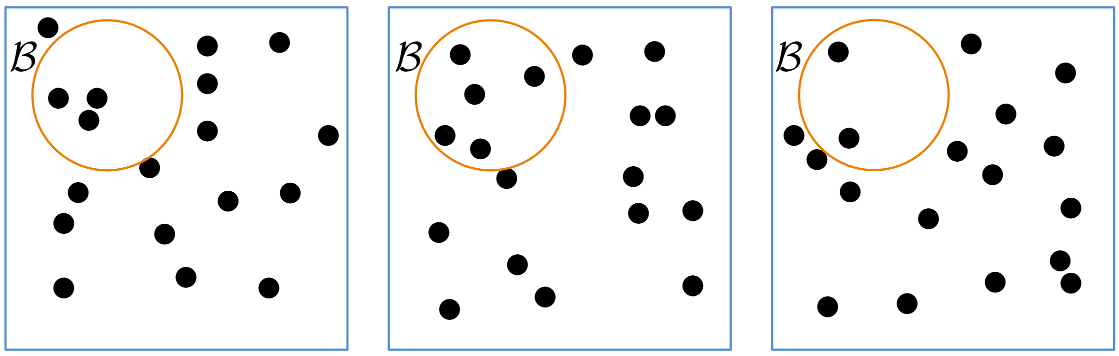


Figure 2.1: For a given set $\mathcal{B}$, the random variable $N(\mathcal{B})$ counts the number of sample points that fall into $\mathcal{B}$ for different realizations. For these three realizations of a point process, it has values $3, 5, 2$, respectively. By generating more distributions from the same point process, we can get an estimate of the PDF of $N(\mathcal{B})$.

point process. Similarly, we can consider the joint probability and corresponding PDF of $N(\mathcal{B}_1), \cdots, N(\mathcal{B}_1)$, which will be a fundamental tool in the next sections.

2.2.1 Gaussian Point Processes

An important case that we will encounter a lot in the coming sections is Gaussian point processes. For these processes, the random variables $N(\mathcal{B}_i)$ follow a Gaussian distribution. Hence, for any group of given sets $\mathcal{B}_i$, the joint PDF of $N(\mathcal{B}_i)$ is a Gaussian. Defining the random variable $X = [N(\mathcal{B}_1), \cdots, N(\mathcal{B}_n)]$, we can write this as $p_X(\mathbf{x}) \propto e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$, where the vector $\mathbf{x}$ stores the number of points in each set $N(\mathcal{B}_i)$ for a given realization (distribution), and the mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{C}$ are the only parameters of the distribution that defines the PDF.

This leads to significant simplifications in the analysis, as the random variables can now be fully described by only the mean and covariance, which correspond to first and second order statistics on the random variables $N(\mathcal{B}_i)$. We will see in the next sections that many important results rest on this assumption. In practice, physicists have observed that most point distributions are generated by Gaussian point processes. This is termed as the *second order dogma* in the literature.

2.2.2 Point Process Statistics

The view of point processes with joint probabilities $\mathbb{P}(X(\mathcal{B}_1) \leqslant b_1, \cdots, X(\mathcal{B}_n) \leqslant b_n)$ naturally gives rise to a PDF based description. As all works in the rendering literature, we will utilize this description for analyzing a point process.

Product Densities

Let us $\mathbf{x}_i$ denote some arbitrary points in $\mathbb{R}^d$, and $\mathcal{B}_i$ infinitesimal spheres centered at these points with volumes $d\mathbf{x}_i = |\mathcal{B}_i|$. If we now define $\mathbb{P}(\mathbf{x}_1, \cdots, \mathbf{x}_n)$ as the joint *probability of having a point* of a point process $\mathcal{P}$ in each of the infinitesimal spheres $\mathcal{B}_i$, then the k^{th} order *product density* $\varrho^{(k)}$ is defined by $\mathbb{P}(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \varrho^{(k)}(\mathbf{x}_1, \cdots, \mathbf{x}_n) d\mathbf{x}_1 \cdots d\mathbf{x}_n$. The product densities are thus simply PDF functions for joint probabilities of random variables of a certain kind that define the point process $\mathcal{P}$.

We will see in the next sections that for our purpose of using point processes for analyzing error in monte carlo integration, it is sufficient to consider product densities, and in particular first and second order product densities $\varrho^{(1)}$, $\varrho^{(2)}$. The first order product

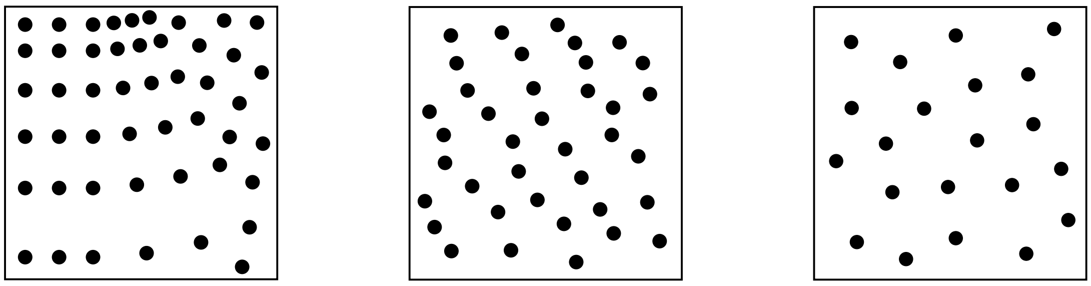


Figure 2.2: (Left) A general point process generates point distributions with spatially varying characteristics. (Middle) A stationary point process has translation invariant characteristics. (Right) An isotropic point process further generates rotation invariant point distributions so that we may translate or rotate the distributions without affecting their properties.

density is simply given by $\varrho^{(1)}(\mathbf{x})d\mathbf{x} = \mathbb{P}(\mathbf{x})$. We can show that the expected number of points in a set $\mathcal{B}$, where the expectation is computed over different realizations of the point process $\mathcal{P}$, can be written as the integral of this expression: $\mathbb{E}_{\mathcal{P}}\left[N(\mathcal{B})\right] = \int_{\mathcal{B}} \varrho^{(1)}(\mathbf{x})d\mathbf{x}$. Hence, $\varrho^{(1)}(\mathbf{x})$ measures the local expected density of points of the point process. It is thus usually called the *intensity* of $\mathcal{P}$, and denoted by $\lambda(\mathbf{x}) = \varrho^{(1)}(\mathbf{x})$.

We can similarly define the second order product density $\varrho^{(2)}(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} = \mathbb{P}(\mathbf{x},\mathbf{y})$. As we will not need higher order product densities, we call this product density simply as $\varrho(\mathbf{x},\mathbf{y})$. It thus gives us the joint probability of finding a pair of points at certain locations in space. Intuitively, it can be estimated by generating infinitely many distributions from the point process, and counting the number of pairs with points falling into the volumes around $\mathbf{x}$ and $\mathbf{y}$ at the same time.

Stationary and Isotropic Point Processes

There are two very important special cases of point processes considered in the literature. We will see that these cases are typically encountered in practice for rendering as well, and can be easily extended if more complex distributions are needed. *Stationary point processes* refer to processes which are translation invariant, i.e. the distributions generated by such point processes will have the same statistics (e.g. product densities) regardless of where we look at in space. We show an example distribution generated by a stationary point process in Figure 2.2 (middle). For this case, the intensity becomes a constant $\lambda(\mathbf{x}) = \lambda$, and second order product density is a function of the *difference* between locations in space $\varrho(\mathbf{x},\mathbf{y}) = \varrho(\mathbf{x}-\mathbf{y})$. This is what makes most of the statistics encountered in the rendering literature meaningful, as we will elaborate in Chapter 3. The second order product density in this case is typically given by the normalized *pair correlation function* (PCF) g as $\varrho(\mathbf{x}-\mathbf{y}) = \lambda^2 g(\mathbf{x}-\mathbf{y})$.

We can further assume that a point process is rotation invariant, meaning that the characteristics of distributions generated by that point process do not depend on where we are, and how we are oriented in space. In other words, you may freely rotate the distributions and will always get the same characteristics (Figure 2.2 (right)). For this case, $\lambda(\mathbf{x}) = \lambda$, and $\varrho(\mathbf{x},\mathbf{y}) = \varrho(\|\mathbf{x}-\mathbf{y}\|)$ and thus $g(\mathbf{x}-\mathbf{y}) = g(\|\mathbf{x}-\mathbf{y}\|)$. Hence, instead of considering the high dimensional $\varrho(\mathbf{x},\mathbf{y})$ statistic, we can simply work with the 1D statistic $g(\|\mathbf{x}-\mathbf{y}\|)$ to describe the second order correlations of such processes.

2.2.3 Campbell's Theorem

The importance of product densities comes from the fact that they can be used to compute expectations over distributions generated by a point process. To analyze error, i.e. bias and variance, in monte carlo rendering, we need to compute such expectations for functions that are evaluated and summed at the sampling points $\mathbf{x}_i$ (we will elaborate more in Section 2.4 and Chapter 4), i.e. $\mathbb{E}_{\mathcal{P}}\left[\sum f(\mathbf{x}_i)\right]$, where the expectation is computed over different realizations (point distributions) of a given point process. Here, $\mathbf{x}_i$ are the points in a given realization. As the number of total points in a given realization is random, it is omitted for the sums. This can also be generalized to functions of more than one variable as $\mathbb{E}_{\mathcal{P}}\left[\sum_{ij} f(\mathbf{x}_i, \mathbf{x}_j)\right]$.

Campbell's theorem relates the expected values of such sums to integrals of arbitrary positive functions f, and the product densities of the given point process:

$$\mathbb{E}_{\mathcal{P}}\left[\sum f(\mathbf{x}_i)\right] = \int_{\mathbb{R}^d} f(\mathbf{x})\lambda(\mathbf{x})d\mathbf{x}, \tag{2.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}^+$, and

$$\mathbb{E}_{\mathcal{P}}\left[\sum_{i\neq j} f(\mathbf{x}_i, \mathbf{x}_j)\right] = \int_{\mathbb{R}^d\times\mathbb{R}^d} f(\mathbf{x}, \mathbf{y})\varrho(\mathbf{x}, \mathbf{y})d\mathbf{x}d\mathbf{y}, \tag{2.2}$$

where $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ and under the common assumption [11] that no two points in a process can be at the same location almost surely.

2.3 Fourier Transform

Spatial statistics are best captured by first and second order correlations. The second order correlations are directly computed from the pair-wise distances (differentials) between each pair of samples. In differential terms, for sampling patterns, it is straightforward to show that the cosine transform of the differentials give the corresponding expected power spectrum. To study different isotropic and anisotropic samplers, we leverage a couple of important theorems (listed below) that bridge the gap between spatial and spectral domains. We will cover the necessary background for these theorems, which will be used later to build sound mathematical formulations that would help derive closed-form formulations of error (bias and variance) for Monte Carlo integration.

2.3.1 Autocorrelation Theorem

Spatial statistics are directly linked to the spectral domain. According to the autocorrelation theorem, the Fourier transform of the autocorrelation of any function is equivalent to its power spectrum. Öztireli [20] used this theorem to link variance during Monte Carlo integration with the spectral domain variance formulation.

2.3.2 Projection-Slice Theorem

In two-dimensions, the Projection-Slice theorem states that *if we take a 2D function $\mathcal{I}(\mathbf{x})$, project it onto a one-dimensional line $L(u)$, and take the Fourier transform of this projection, then this is equivalent to taking that same function $\mathcal{I}(\mathbf{x})$, but do a two-dimensional Fourier transform, and then slice it through the origin, in a direction parallel to the projection line $L(u)$.* In Sec. 4.2.2, this theorem would be helpful in understanding the convergence rate improvements in Monte Carlo rendering due to samplers with anisotropic power spectra.

2.4 Numerical Integration

Given a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^+$ representing certain parts of light transport, rendering requires us to estimate its integral in order to compute the color of each pixel. We thus would like to study the error in estimating the integral $I := \frac{1}{|\mathcal{D}|} \int_{\mathcal{D}} f(\mathbf{x}) d\mathbf{x}$, where $\mathcal{D}$ is the support of the function f with $\forall \mathbf{x} \notin \mathcal{D}, f(\mathbf{x}) = 0$, and $|\mathcal{D}|$ is its volume. The estimator for this integral is of the form

$$\hat{I} := \sum_{i=1}^{n} w_i f(\mathbf{x}_i), \tag{2.3}$$

for some positive weights w_i and sample points $\mathbf{x}_i$. The error in this estimator consists of a bias $bias_{\mathcal{P}}[\hat{I}] = I - \mathbb{E}_{\mathcal{P}}[\hat{I}]$, and a variance $var_{\mathcal{P}}[\hat{I}] = \mathbb{E}_{\mathcal{P}}[\hat{I}^2] - (\mathbb{E}_{\mathcal{P}}[\hat{I}])^2$ term, where the expectations are over point distributions generated by the point process $\mathcal{P}$ as before.

2.4.1 Quadrature Rules

A number of solutions are developed for the numerical solution of integrals. Most prominent are the Quadrature rules, where the weights w_i and the sample positions $\mathbf{x}_i$ from Equation 2.3 are determined in advance. The well-known Newton-Cots formula is used with Midpoint rule (1-sample), Trapezoid rule (2-samples), Simpson rule (3-samples), where samples are powers of 2 and approximates the integral as a sum of weighted, equidistant samples. Similarly, Gauss quadratures were designed which extends freedom of choosing sample locations. In both constructions, the convergence rate achieved is $\mathcal{O}(n^{-r})$ given n samples and a smooth integrand that has r-continuous derivatives. These approaches, however, suffer from the curse of dimensionality, requiring n^d samples for a d-dimensional integral for the convergence rate $\mathcal{O}(n^{-r/d})$. It also requires special treatment to adapt these rules to non-square domains which are typical in rendering.

2.4.2 Monte Carlo Integration

Estimating a higher dimensional integral with Monte Carlo (MC) based methods has huge success. The MC based approach involves generating random samples over the domain (with no two samples at the same location), is independent of dimensionality of the problem or the underlying topology of the domain and would always give the variance convergence rate of $\mathcal{O}(n^{-1})$ with completely random samples, irrespective of the smoothness properties of the underlying integrand. We will investigate how to go beyond this convergence guarantee with careful sampling.

2.4.3 Quasi-Monte Carlo Integration

Monte Carlo integration suffers, apart from the slow convergence rate, from the disadvantages that only probabilistic statements on convergence and error limits are possible. The success of any Monte Carlo procedure stands or falls with the quality of these random samples. If the distribution of the sample points is not uniform, then there are large regions where there are no samples at all, which can increases the error. Closely related to this is the fact that a smooth function is evaluated at unnecessary many locations if samples are clumped.

Quasi-Monte Carlo (QMC) based integration follows deterministic generation of samples, while making sure uniform distributions. The underlying algorithms are based on number-theoretic approaches. As a result, sample sequences with good uniformity properties can be generated in very high dimensions. The underlying sample generation routines are also pretty fast and progressive with almost no pre-processing required. In these notes, we will focus mainly on the stochastic samplers (MC based approaches), and the error caused by these samplers during MC rendering.

2.5 Discrepancy Theory

Different sequences can be uniformly distributed. But looking at various uniformly distributed sequences, one will realize that there exist sequences with a very good distribution behavior, whereas other sequences might just barely be uniformly distributed. Discrepancy is a quality measure that measures the deviation of the sequence from an ideal distribution.

Shirley [26] introduced the notion of discrepancy to the computer graphics community to compute the quality of sampling patterns. Even though Fourier analysis tools have been used to study the behavior of various sampling distributions, this approach only provides a qualitative analysis in the form of two-dimensional graphs. Computation of discrepancy allows assigning a single quality number, the discrepancy, to the point set. This allows to order point sets according to their discrepancy and compare which one is the best with respect to this measure.

2.5.1 Koksma-Hlawka Inequality

The quality criteria derived from the discrepancy of sample positions is related to the Koksma-Hlawka inequality. The Koksma-Hlawka inequality is a tight error bound on the approximation of an integral by the sample average of integrand values:

$$\left| \frac{1}{n} \sum_{k=1}^{n} \mathcal{I}(x_k) - \int_0^1 \mathcal{I}(x) dx \right| \leqslant \mathcal{H}(x_k) \mathcal{C}(\mathcal{I}) \tag{2.4}$$

In this inequality $\mathcal{H}(x_k)$ is the discrepancy of the points $0 \leqslant x_k \leqslant 1$ and $\mathcal{C}(\mathcal{I})$ is the total variation of the function $\mathcal{I}$:

$$\mathcal{H}(x_k) := \sup_{0 \leqslant t \leqslant 1} \left\{ \left| \frac{1}{n} \chi[0,t](x_k) - t \right| \right\}, \tag{2.5}$$

$$\mathcal{C}(\mathcal{I}) := \sup_{0 = y_0 < y_1 < \ldots < y_n} \left\{ \sum_{k=1}^{n} \mathcal{I}(y_k) - \mathcal{I}(y_{k-1}) \right\}, \tag{2.6}$$

where $\chi[a,b]$ is a characteristic function which is non-zero only in the range $[a,b]$. Hickernell [10] gives a detailed overview of this inequality and mentions that although the Koksma-Hlawka inequality was originally derived for a particular integration domain $\mathcal{D}$, and a particular space of integrands, $\mathcal{I}$, the same may be applied to similar inequalities that have been derived for other $\mathcal{D}$ and $\mathcal{I}$. In these inequalities, the integration error is bounded by a product of two terms, the discrepancy of the sample points, and the variation of the integrand.

In other words, the Koksma-Hlawka inequality splits the error into the part due to the quality of the sample points, and the part due to the roughness of the integrand. When $\mathcal{I}$ is a reproducing kernel Hilbert space, the Koksma-Hlawka inequality is straightforward to derive, and there is a simple formula for the discrepancy. This discrepancy also has several other useful interpretations, including, (i) how the proportion of sample points in a box deviates from the volume of the box, (ii) the average-case integration error, and (iii) a goodness-of-fit statistic. Integration lattices and digital sequences are two popular families of low discrepancy sample points. These sets typically give better convergence rates for the discrepancy than a simple random sample. The Koksma-Hlawka inequality plays a key role in the development of quasi-Monte Carlo methods. It has also influenced the study of experimental design and led to the creation of uniform designs.

3. Sampling Analysis

We need computational tools and algorithms to realize the theoretical analysis tools as proposed in the last section. We expect that the resulting statistics provide valuable insights into the sampling patterns and thus allow us to understand, and intelligently choose and adapt sampling patterns according to the particular rendering problem. Several measures for this purpose have been proposed in the last decades. In this section, we will summarize and elaborate on each of them in the common framework of point processes.

A very important concern here is the details of the resulting techniques, such as normalization constants and stability factors, which are typically never explained in the rendering literature. We thus put a particular emphasis on explaining such choices and best practices, with theoretical justifications and practical code snippets.

3.1 Spatial Measures

Point process statistics are naturally defined in the spatial domain, e.g. in terms of distances between points. Historically, such spatial measures of correlations have come quite late into the rendering literature, but has turned out to be very powerful, as they contain all information on a point pattern via the underlying point process. Indeed, all other measures can be derived in terms of these statistics from point processes, as we will elaborate on in the following sections.

The fundamental statistics we will be dealing with are first and second order product densities $\lambda(\mathbf{x})$ and $\varrho(\mathbf{x}, \mathbf{y})$ (Section 2.2.2), which capture first and second order correlations of point locations in point distributions generated by an underlying point process. In particular, we will assume stationarity, leading to a constant λ, and $g(\mathbf{x}-\mathbf{y}) = \varrho(\mathbf{x}-\mathbf{y})/\lambda^2$. Almost all statistics proposed in the rendering literature operate under this assumption, allowing to work with the tractable $g(\mathbf{r})$ function, where $\mathbf{r} = \mathbf{x} - \mathbf{y}$, or its isotropic counterpart $g(r)$ with $r = \|\mathbf{x} - \mathbf{y}\|$.

3.1.1 Estimator for Intensity

All estimators for λ and g can be derived starting from Campbell's theorem (Section 2.2.3). For the case of lambda, we can write the following expression

$$\mathbb{E}_{\mathcal{P}}\left[\sum \mathbb{I}_{\mathcal{D}}(\mathbf{x}_i)\right] = \mathbb{E}_{\mathcal{P}}\left[\sum_{\mathbf{x}_i \in \mathcal{D}} 1\right] = \int_{\mathcal{D}} \lambda d\mathbf{x} = \lambda \int_{\mathcal{D}} d\mathbf{x} = \lambda |\mathcal{D}|, \tag{3.1}$$

where $\mathbb{I}_{\mathcal{D}}(\mathbf{x})$ is the indicator function which gives 1 if $\mathbf{x} \in \mathcal{D}$, and 0 otherwise, $\mathcal{D}$ is part of the domain $\mathbb{R}^d$ where the point process in defined, and $|\mathcal{D}|$ is its volume. This simply tells us to generate different distributions $\mathcal{P}_k$ from the point process, for each distribution count the number of points $N_k(\mathcal{D})$ that fall into a domain $\mathcal{D}$, and average those numbers. It thus leads to the following unbiased estimator

$$\hat{\lambda} = \frac{\sum_{\mathcal{P}_k} N_k(\mathcal{D})}{K|\mathcal{D}|}, \tag{3.2}$$

for K distributions generated by the point process.

3.1.2 Estimators for the PCF

Similarly, we can derive an estimator for the pair correlation function (PCF) $g(\mathbf{r})$ as follows.

$$\begin{aligned}\mathbb{E}_{\mathcal{P}}\left[\sum_{i \neq j} \delta(\mathbf{r} - (\mathbf{x}_i - \mathbf{x}_j))\right] &= \int_{\mathbb{R}^d \times \mathbb{R}^d} \delta(\mathbf{r} - (\mathbf{x} - \mathbf{y}))\varrho(\mathbf{x} - \mathbf{y}) d\mathbf{x} d\mathbf{y} \\ &= \lambda^2 \int_{\mathbb{R}^d \times \mathbb{R}^d} \delta(\mathbf{r} - (\mathbf{x} - \mathbf{y})) g(\mathbf{x} - \mathbf{y}) d\mathbf{x} d\mathbf{y} \\ &= \lambda^2 g(\mathbf{r}),\end{aligned} \tag{3.3}$$

where δ is the d dimensional Dirac delta. The estimator is thus given by

$$\hat{g}(\mathbf{r}) = \frac{1}{K\lambda^2} \sum_{\mathcal{P}_k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{P}_k, i \neq j} \delta(\mathbf{r} - (\mathbf{x}_i - \mathbf{x}_j)). \tag{3.4}$$

In practice, this means for each point distribution, we compute a histogram of difference vectors $\mathbf{x}_i - \mathbf{x}_j$ [30]. We provide the pseudocode for computing the statistic below for $d = 2$. One important aspect here is that this estimator is computed using *only one distribution* from the point process, as we typically observe a limited number of distributions. If we have more than one distribution generated by the same point process, we can of course average the statistic given by the below code for a better estimator as given in Equation 3.4.

Another important step for a practical estimator is using normalizations. The difference vectors $\mathbf{x}_i - \mathbf{x}_j$ are normalized by the estimator of λ, i.e. the number of points n for this given distribution, as $(\mathbf{x}_i - \mathbf{x}_j)\sqrt{n}$. This ensures that intensity does not alter this statistic, and hence distributions with different number of points but the same second order correlations will generate the same statistic. Similarly, the final histogram is normalized to cancel the affect of intensity on the histogram values.

Finally, we should consider the finite extent of the observation window for a distribution. The original estimator in Equation 3.4 implicitly assumes an infinite window, i.e. the whole Euclidean space in which the point process is defined. For a practical estimator, we get points in a finite domain, typically a square. We are thus missing difference vectors that have length larger than half of the length of the edges of this square. This means we can only estimate $g(\mathbf{r})$ for a limited range of $\|\mathbf{r}\|$.

```
procedure differential_histogram(samples, histoWidth, histoHeight)
  int N = samples.size();

  //distLimit allows to select a fractional region of the domain
  double distLimit = 0.125;

  double fracLimit = 1.0 / (2.0*distLimit);
  for p: 0 → N{
    for q: 0 → N{
    if (p == q)
      continue:
    double dx = samples[p].x - samples[q].x;
    double dy = samples[p].y - samples[q].y;

    //Scale by the mean distance (in a regular 2D grid mean distance = $\sqrt{N}$)
    dx *= $\sqrt{N}$
    dy *= $\sqrt{N}$

    //Only take points which are within a mutual distance distLimit
    if(fabs(dx) < distLimit && fabs(dy) < distLimit){
      int hcol = _xRes * fracLimit * (dx + distLimit);
      int hrow = _yRes * fracLimit *  (dy + distLimit);
      histogram2d[hrow*_xRes+hcol] += 1;
    }

  //Normalize differential domain to make sure it converges to 1 as
    dist->inf
  double domainLength = 2*distLimit;
  double area = domainLength*domainLength;
  double numBins = histoWidth*histoHeight;
      for(int k = 0; k < histoHeight; k++){
            for(int j = 0; j < histoWidth; j++){
                histogram2d[k*histoWidth+j] *= numBins;
                histogram2d[k*histoWidth+j] /= (double)(N*area) ;
            }
      }
return histogram2d;
```

Listing 3.1: Differential Histogram

A smoothed version of this histogram can also be computed with a Gaussian kernel [30]. This smoothing can be realized with a convolution with a Gaussian kernel on the computed histogram above, treating it as a 2-dimensional image.

Extensions for Finite Domains

The derivation for the case of a finite domain $\mathcal{D}$ proceeds similar to the derivation in Equation 3.3

$$\begin{aligned}
\mathbb{E}_{\mathcal{P}}\left[\sum_{\mathbf{x}_i,\mathbf{x}_j\in\mathcal{D},i\neq j}\delta(\mathbf{r}-(\mathbf{x}_i-\mathbf{x}_j))\right] &= \int_{\mathcal{D}\times\mathcal{D}}\delta(\mathbf{r}-(\mathbf{x}-\mathbf{y}))\varrho(\mathbf{x}-\mathbf{y})d\mathbf{x}d\mathbf{y} \\
&= \lambda^2\int_{\mathcal{D}\times\mathcal{D}}\delta(\mathbf{r}-(\mathbf{x}-\mathbf{y}))g(\mathbf{x}-\mathbf{y})d\mathbf{x}d\mathbf{y} \\
&= \lambda^2 a_{\mathbb{I}_{\mathcal{D}}}(\mathbf{r})g(\mathbf{r}).
\end{aligned} \tag{3.5}$$

Here, a_f is the autocorrelation function of f, for any function f. The resulting estimator is given by

$$\hat{g}(\mathbf{r}) = \frac{1}{K\lambda^2 a_{\mathbb{I}_{\mathcal{D}}}(\mathbf{r})} \sum_{\mathcal{P}_k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{P}_k, i \neq j} \delta(\mathbf{r} - (\mathbf{x}_i - \mathbf{x}_j)). \tag{3.6}$$

Hence, the only change to the estimator in Equation 3.4 is that for each $\mathbf{r}$, we need to normalize the estimated $g(\mathbf{r})$ with the autocorrelation $a_{\mathbb{I}_{\mathcal{D}}}(\mathbf{r})$ of the indicator function $\mathbb{I}_{\mathcal{D}}$ for the domain $\mathcal{D}$. This entails two changes to the code above: 1) we can now take all possible difference vectors, without any limitation on its length, 2) we need to normalize the histogram with the autocorrelation before the final normalization. This normalization with the autocorrelation takes care of the fact that we are getting less difference vectors at larger distances in a finite domain. The autocorrelation of the indicator function for a square domain is given by $a_{\mathbb{I}_{\mathcal{D}}}(\mathbf{r}) = \prod_{l=1}^{d}(|\mathcal{D}|^{1/d} - |(\mathbf{r})_l|)$ when $|\mathcal{D}|^{1/d} > |(\mathbf{r})_l|$ for all l, and 0 otherwise, where $(\mathbf{r})_l$ is the l^{th} component of the vector $\mathbf{r}$. In general, this autocorrelation can be computed numerically for a general domain.

Isotropic Point Processes

For an isotropic point process, we can similarly derive an estimator by starting from a radially symmetric Dirac delta

$$\begin{aligned} \mathbb{E}_{\mathcal{P}}\left[\sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}, i \neq j} \delta(r - \|\mathbf{x}_i - \mathbf{x}_j\|)\right] &= \int_{\mathcal{D}\times\mathcal{D}} \delta(r - \|\mathbf{x} - \mathbf{x}\|)\varrho(\mathbf{x} - \mathbf{y}) d\mathbf{x} d\mathbf{y} \\ &= \lambda^2 \int_{\mathcal{D}\times\mathcal{D}} \delta(r - \|\mathbf{x} - \mathbf{x}\|) g(\|\mathbf{x} - \mathbf{y}\|) d\mathbf{x} d\mathbf{y} \\ &= \lambda^2 g(\mathbf{r}) \int_{\|\mathbf{r}'\| = r} a_{\mathbb{I}_{\mathcal{D}}}(\mathbf{r}') d\mathbf{r}', \end{aligned} \tag{3.7}$$

with the resulting estimator given by

$$\hat{g}(r) = \frac{1}{K\lambda^2 \int_{\|\mathbf{r}'\|=r} a_{\mathbb{I}_{\mathcal{D}}}(\mathbf{r}') d\mathbf{r}'} \sum_{\mathcal{P}_k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{P}_k, i \neq j} \delta(r - \|\mathbf{x}_i - \mathbf{x}_j\|), \tag{3.8}$$

defined for the range of r values such that the denominator is non-zero. The version in a previous work [21] can be obtained if the correction due to the domain $\mathcal{D}$ is omitted, and instead all points in the point process in $\mathbb{R}^d$ are considered

$$\begin{aligned} \mathbb{E}_{\mathcal{P}}\left[\sum_{i \neq j} \delta(r - \|\mathbf{x}_i - \mathbf{x}_j\|)\right] &= \int_{\mathbb{R}^d\times\mathbb{R}^d} \delta(r - \|\mathbf{x} - \mathbf{x}\|)\varrho(\mathbf{x} - \mathbf{y}) d\mathbf{x} d\mathbf{y} \\ &= \lambda^2 \int_{\mathbb{R}^d\times\mathbb{R}^d} \delta(r - \|\mathbf{x} - \mathbf{x}\|) g(\|\mathbf{x} - \mathbf{y}\|) d\mathbf{x} d\mathbf{y} \\ &= \lambda^2 g(r) r^{d-1} |\mathcal{S}_d|, \end{aligned} \tag{3.9}$$

where $|\mathcal{S}_d|$ denotes the volume of the hypersphere in d dimensions. The resulting estimator is thus

$$\hat{g}(r) = \frac{1}{K\lambda^2 r^{d-1} |\mathcal{S}_d|} \sum_{\mathcal{P}_k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{P}_k, i \neq j} \delta(r - \|\mathbf{x}_i - \mathbf{x}_j\|). \tag{3.10}$$

As we did for the estimator of $g(\mathbf{r})$, we can practically omit averaging over different distributions, and also use a normalized smoothing kernel $k(r)$ with $\int_{\mathbb{R}} k(r)dr = 1$ to obtain

$$\hat{g}(r) = \frac{1}{\lambda^2 r^{d-1}|\mathcal{S}_d|} \sum_{i \neq j} k(r - \|\mathbf{x}_i - \mathbf{x}_j\|). \tag{3.11}$$

The discussion so far implies that the estimator in the above equation is biased due to the use of a smoothing kernel, and more importantly, in order to use this version of the estimator, we need to consider an infinite domain. The practical version of this infinite domain is a toroidal domain with periodic boundary conditions. We provide how this can be implemented below.

```
#adapted from:
#https://github.com/cfinch/Shocksolution_Examples/tree/master/PairCorrelation
procedure pairCorrelationFunction2D(x, y, S, rMax, dr):
    """Compute the two-dimensional pair correlation function, also known
    as the radial distribution function, for a set of circular particles
    contained in a square region of a plane.  This simple function finds
    reference particles such that a circle of radius rMax drawn around the
    particle will fit entirely within the square, eliminating the need to
    compensate for edge effects.  If no such particles exist, an error is
    returned. Try a smaller rMax...or write some code to handle edge
   effects! ;)

    Arguments:
        x               an array of x positions of centers of particles
        y               an array of y positions of centers of particles
        S               length of each side of the square region of the
   plane
        rMax            outer diameter of largest annulus
        dr              increment for increasing radius of annulus

    Returns a tuple: (g, radii, interior_indices)
        g(r)            a numpy array containing the correlation function
   g(r)
        radii           a numpy array containing the radii of the
                        annuli used to compute g(r)
        reference_indices   indices of reference particles
    """
    from numpy import zeros, sqrt, where, pi, mean, arange, histogram
    # Number of particles in ring/area of ring/number of reference
   particles/number density
    # area of ring = pi*(r_outer**2 - r_inner**2)

    # Find particles which are close enough to the box center that a
   circle of radius
    # rMax will not cross any edge of the box
    bools1 = x > rMax
    bools2 = x < (S - rMax)
    bools3 = y > rMax
    bools4 = y < (S - rMax)
    interior_indices, = where(bools1 * bools2 * bools3 * bools4)
    num_interior_particles = len(interior_indices)

    if num_interior_particles < 1:
        raise  RuntimeError ("No particles found for which a circle of
   radius rMax\
                  will lie entirely within a square of side length S.
   Decrease rMax\
```

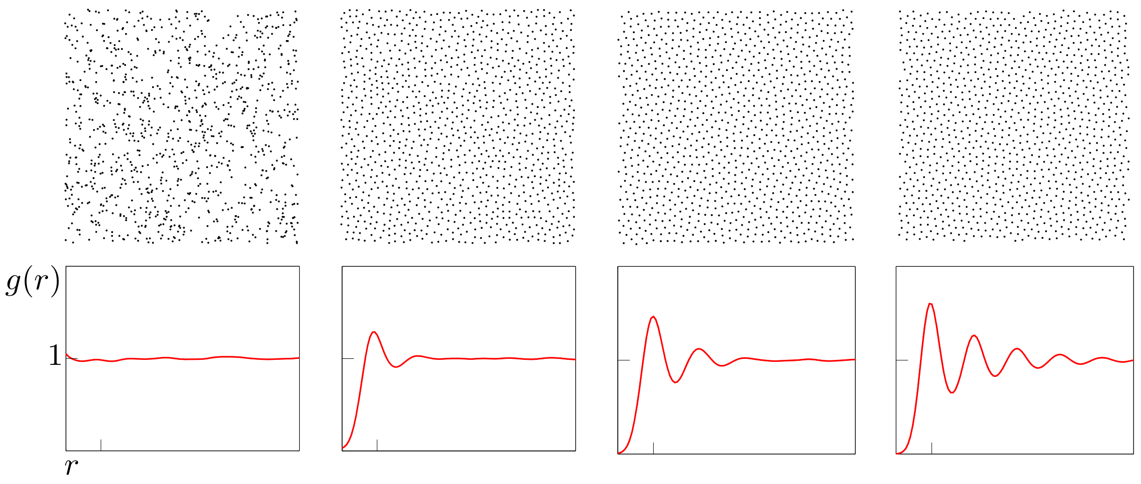


Figure 3.1: An example distribution by the underlying point process (top), and the estimated PCF of the point process (bottom). The r axis is normalized with the maximum possible distance between pairs of points for this square domain and number of points (1024) [21]. The distributions become more regular from left to right. This is reflected in the PCF's: the first peak shifts to higher values of r, and more peaks appear as the distributions become more regular.

```
                or increase the size of the square.")

    edges = arange(0., rMax + 1.1 * dr, dr)
    num_increments = len(edges) - 1
    g = zeros([num_interior_particles, num_increments])
    radii = zeros(num_increments)
    numberDensity = len(x) / S**2

    # Compute pairwise correlation for each interior particle
    for p in range(num_interior_particles):
        index = interior_indices[p]
        d = sqrt((x[index] - x)**2 + (y[index] - y)**2)
        d[index] = 2 * rMax

        (result, bins) = histogram(d, bins=edges)
        g[p, :] = result/numberDensity

    # Average g(r) for all interior particles and compute radii
    g_average = zeros(num_increments)
    for i in range(num_increments):
        radii[i] = (edges[i] + edges[i+1]) / 2.
        rOuter = edges[i + 1]
        rInner = edges[i]
        g_average[i] = mean(g[:, i]) / (pi * (rOuter**2 - rInner**2))
    return (g_average, radii, interior_indices)
```

Listing 3.2: Pair Correlation Function

Examples of estimated PCF's for isotropic point processes, along with an example distribution generated by the underlying point process, are shown in Figure 3.1.

3.2 Spectral Measures

Fourier domain provides another set of tools to analyze and understand the sample distributions. The most popular tools are the periodograms or the power spectrum. Ulichney [28] was the first to provide qualitative characterization of a good sampling pattern, which is now commonly called Blue Noise. Mitchell [17] has also pointed out that energy in the low-frequency part of the Fourier spectrum of the sampling pattern should be avoided. These studies show the significance of Fourier tools in understanding error in Monte Carlo rendering. In this section, we will see how periodograms (power spectra) are fundamentally related to the pair correlation function, and provide code snippets on how to implement power spectra and their corresponding radial counterparts that help analyze error in integration.

3.2.1 Power spectrum

In order to define power spectrum, we first define a sampling function, which is a sum of Dirac impulses at the sampling points for a particular distribution $s(\mathbf{x}) = \sum_i \delta(\mathbf{x} - \mathbf{x}_i)$. Denoting the Fourier transform of this function with $S(\boldsymbol{\nu}) = \mathscr{F}[s(\mathbf{x})](\boldsymbol{\nu})$, we can then define the power spectrum $P(\boldsymbol{\nu})$ as

$$P(\boldsymbol{\nu}) = \frac{1}{\lambda}\mathbb{E}_{\mathcal{P}}\left[S(\boldsymbol{\nu})\overline{S(\boldsymbol{\nu})}\right], \tag{3.12}$$

where $\overline{S(\boldsymbol{\nu})}$ is the complex conjugate of $S(\boldsymbol{\nu})$. This expression can be expanded by plugging in the expression for $S(\boldsymbol{\nu})$ as

$$P(\boldsymbol{\nu}) = \frac{1}{\lambda}\mathbb{E}_{\mathcal{P}}\left[\sum_{jk} e^{-2\pi i \boldsymbol{\nu}^T \mathbf{r}_{jk}}\right]. \tag{3.13}$$

It is important to see how $P(\boldsymbol{\nu})$ relates to $g(\mathbf{r})$. We start by deriving the Fourier transform G of PCF g, starting from its expression in Equation 3.3

$$\begin{aligned} G(\boldsymbol{\nu}) = \mathscr{F}[g(\mathbf{r})](\boldsymbol{\nu}) &= \frac{1}{\lambda^2}\mathbb{E}_{\mathcal{P}}\left[\sum_{j\neq k} \mathscr{F}[\delta(\mathbf{r} - (\mathbf{x}_j - \mathbf{x}_k))](\boldsymbol{\nu})\right] \\ &= \frac{1}{\lambda^2}\mathbb{E}_{\mathcal{P}}\left[\sum_{j\neq k} e^{-2\pi i \boldsymbol{\nu}^T \mathbf{r}_{jk}}\right]. \end{aligned} \tag{3.14}$$

We can also derive the following using Campbell's theorem

$$\mathbb{E}_{\mathcal{P}} \sum_j 1 = \lambda \int_{\mathcal{V}} d\mathbf{x} = \lambda. \tag{3.15}$$

Finally, we can sum these to get

$$\begin{aligned} \lambda G(\boldsymbol{\nu}) + 1 &= \frac{1}{\lambda}\mathbb{E}_{\mathcal{P}}\left[\sum_{j\neq k} e^{-2\pi i \boldsymbol{\nu}^T \mathbf{r}_{jk}}\right] + \frac{1}{\lambda}\mathbb{E}_{\mathcal{P}}\sum_j 1 \\ &= \frac{1}{\lambda}\mathbb{E}_{\mathcal{P}}\left[\sum_{jk} e^{-2\pi i \boldsymbol{\nu}^T \mathbf{r}_{jk}}\right] = P(\boldsymbol{\nu}). \end{aligned} \tag{3.16}$$

Hence, we can write the relation

$$P(\boldsymbol{\nu}) = \lambda G(\boldsymbol{\nu}) + 1. \tag{3.17}$$

This equation makes it clear that power spectrum and pair correlation function carry the same information regarding the second order characteristics of a point process, and both only make sense for stationary point processes. The difference is the ease certain properties can be read out from their estimators.

There exist many fast variants of estimating power spectrum with *discrete* Fourier transform, however, following Heck et al. [9], we emphasize on using the *continuous* Fourier transform version as above to analyze sampling patterns as we show in the code snippet below.

```
procedure powerSpectrum(samples, spectrumWidth, spectrumHeight)
  int N = samples.size()
  for u: 0 → spectrumWidth{
    for v: 0 → spectrumHeight{
    double real = 0, imag = 0;

    //compute the real and imaginary fourier coefficients
    for(int k=0;k<N;k++){
      real += cos(2 * π * (u * samples[k].x + v * samples[k].y));
      imag += sin(2 * π * (u * samples[k].x + v * samples[k].y));
    }

    //power spectrum is the magnitude square value of the coefficients
    power[u * spectrumWidth + v] = (real*real + imag * imag) / N;
    }
  }
  return power;
}
```

Listing 3.3: Fourier power spectrum

The expected power spectrum is computed by averaging these power spectra over multiple (usually 1000) realizations.

3.2.2 Radial Domain Analysis

We have already seen that for isotropic point processes, the PCF is a 1-dimensional function representing distribution of distances between pairs of points. Similarly, power spectrum can also be summerized with a 1-dimensional radial average for isotropic point processes. This *radially averaged* Fourier power spectrum [28] has been perhaps the most widely used tool to analyze point samples, characterizing various stochastic sampling patterns ranging from white noise to blue noise, and more recently used to derive variance convergence rates of various stochastic samplers [22]. The pseudo-code (Listing 3.4) to generate radial average is quite simple and requires only the expected power spectrum.

```
procedure radialAverage(expectedPowerSpectrum, spectrumWidth,
    spectrumHeight){
  int halfWidth = 0.5 * spectrumWidth;

  int* histoCounter = new int[halfWidth]();
  double* radialHistogram = new double[halfWidth]();

  //center corresponds to the DC frequency
  int center = halfWidth;
  for r: 0 → spectrumWidth{
    for c: 0 → spectrumHeight{
      double dx = center-c;
      double dy = center-r;
      double distance = sqrt(dx*dx+dy*dy);
      int imgIndex = r*spectrumWidth+c;
      int index = distance;

      //consider frequencies that are less than halfWidth
      if(distance > halfwidth-1)
        continue;
      else{
        radialHistogram[index] += expectedPowerSpectrum[imgIndex];
        histoCounter[index] += 1;
      }
    }
  }
  //normalize all the bins
  for(int i = 0; i < halfwidth; i++){
    radialHistogram[i] /= double(histoCounter[i]);
  }
  //dump it in a file
  std::ofstream file;
  file.open(filename.c_str());

  for(int i = 0; i < halfwidth-1; i++)
    file << i << " " << std::fixed << std::setprecision(15) <<
   radialHistogram[i] << std::endl;
  file.close();
  //clear memory
  delete [] histoCounter;
  delete [] radialHistogram;
}
```

Listing 3.4: Radially averaged power spectrum

Radial Variance & Anisotropy

While radial averaging is appropriate for analyzing *isotropic* power spectra, many of the stochastic point sampling strategies used in rendering—such as N-rooks [26] or even jittered sampling [4]—are in fact *anisotropic*. Instead of considering the full 2-dimensional power spectrum, an easy way to inspect anisotropy within a spectrum is by computing the *radial variance* of the expected power spectrum. Following [13], radial variance computes the squared deviation of the expected power spectrum from its radially averaged version:

$$\mathcal{V}(\rho) = \frac{1}{n(\rho)-1}\sum_{j=1}^{n(\rho)}(P(\boldsymbol{\nu}_j) - \breve{P}(\rho))^2, \tag{3.18}$$

where $n(\rho)$ is the number of considered frequency samples $\boldsymbol{\nu}_j$ with $\|\boldsymbol{\nu}_j\|^2 = \rho$ for computing the estimator at this radius ρ, and $\breve{P}(\rho) = \frac{1}{n(\rho)}\sum_{j=1}^{n(\rho)} P(\boldsymbol{\nu}_j)$ is the radial average of the power spectrum $P(\boldsymbol{\nu})$. The corresponding *radial anisotropy* can be computed as follows:

$$\mathcal{A}(\rho) = 10 * \log_{10}\left[\frac{\mathcal{V}(\rho)}{\breve{P}(\rho) * \breve{P}(\rho)}\right] \tag{3.19}$$

The anisotropy is typically defined when the power spectra is averaged only over 10 realizations in the literature. This however could result in a noisy estimate. To compute anisotropy for any given number of realizations (trials) we can simply divide the radial anisotropy computed from (3.19) by $\log_{10}(\text{trialCount})$, where trialCount is the trial count.

```
procedure radialVariance(expectedPowerSpectrum, radialHistogram,
    spectrumWidth, spectrumHeight)

  int halfWidth = spectrumWidth*0.5;
  int center = halfWidth;

  int* histoCounter = new int[halfWidth]();

  for r: 0 → spectrumWidth{
    for c: 0 → spectrumHeight{
      double dx = center-c;
      double dy = center-r;
      double distance = sqrt(dx*dx+dy*dy);
      int imgIndex = r*spectrumWidth+c;
      int index = distance;

      //Only consider frequencies that are less than halfWidth
      if(distance > halfWidth-1)
        continue;
      else{
        double deviation = expectedPowerSpectrum[imgIndex] -
    radialHistogram[index];
        radialVariance[index] += (deviation*deviation);
        histoCounter[index] += 1;
      }
    }
  }
  for(int k = 1; k < halfWidth; k++){
    radialVariance[k] /= (histoCounter[k]-1);
    radialAnisotropy[k] = 10*log10(radialVariance[k] /
   (radialHistogram[k]*radialHistogram[k]));

    /// Anisotropy scales by log10(numTrials), therefore, we normalize
    /// below to make sure Anisotropy always scales to -10dB.
    if(trialCount > 1)
      radialAnisotropy[k] /= log10(trialCount);
  }
  delete [] histoCounter;
}
```

Listing 3.5: Radial variance (anisotropy)

The radially averaged power spectrum as in the last section, along with the variance as defined here, provide a summary of the power spectrum of a point pattern. We will provide a more detailed analysis on anisotropy in the power spectrum in the next chapter.

3.3 Discrepancy

We will focus on *star* discrepancy of a sample distribution which can be easily computed by counting the number of samples within axis-aligned boxes $B = \{[0, v_1] \times [0, v_2] \times [0, v_3] \times \cdot \times [0, v_N]\}$ where $0 \leqslant v_i < 1$. Given a sequence of sample points $S = x_1, ..., x_n$, the discrepancy of S with respect to B is

$$\mathcal{H}(B, S) = \sup_{b \in B} \left| \frac{\sharp\{x_k \in b\}}{n} - V(b) \right|, \tag{3.20}$$

which can be easily written in a code snippet as below:

```
procedure StarDiscrepancy(samples, N){
  int totalNumTrials = 100000;

  std::vector<double> supDiscrepancy(totalNumTrials, 0.0);

  for(int trial = 0; trial <= totalNumTrials; trial++){

    double xBoxLimit = drand48();
    double yBoxLimit = drand48();
    double Area = xBoxLimit * yBoxLimit;

    int totalCount = 0;
    for(int k=0; k < n; k++){
    double x = samples[k].x;
    double y = samples[k].y;

    //assuming samples ∈ [0,1) × [0,1) in 2D
    if(x < xBoxLimit && y < yBoxLimit)
      totalCount += 1;
    }
    supDiscrepancy[trial] = fabs((totalCount / double(n)) - Area);
  }
  std::sort(supDiscrepancy.begin(), supDiscrepancy.end());
  std::cerr << "\nStar Discrepancy: " << supDiscrepancy[totalNumTrials-1]
   << std::endl;
}
```

Listing 3.6: Star discrepancy

4. Error Analysis in Monte Carlo Integration

In this chapter, we apply the analysis tools explained so far for analyzing error in the estimator $\hat{I} := \sum_{i=1}^{n} w_i f(\mathbf{x}_i)$ for the integral $I = \frac{1}{|\mathcal{D}|}\int_{\mathcal{D}} f(\mathbf{x})d\mathbf{x}$, specifically focusing on common patterns encountered in rendering. As we will see below, the expressions for error, consisting of bias and variance terms, turn out to only depend on first and second order statistics of point processes with relatively simple formulas either in spatial or spectral domains [20, 22].

4.1 General Point Patterns

We start with the case of general point processes, where we have no prior assumptions on any order of correlations. Note that this case covers *all* possible point patterns. To derive bias and variance, we need to derive expressions for $\mathbb{E}_{\mathcal{P}}[\hat{I}]$ and $\mathbb{E}_{\mathcal{P}}[\hat{I}^2]$. Assuming that the weights w_i in the estimator $\hat{I}$ are sampled from a continuous positive function such that $w_i = w(\mathbf{x}_i)$, and setting $f_i = f(\mathbf{x}_i)$, the expected values can then be derived as follows

$$\mathbb{E}_{\mathcal{P}}[\hat{I}] = \mathbb{E}_{\mathcal{P}}\left[\sum w(\mathbf{x}_i)f(\mathbf{x}_i)\right] = \int_{\mathcal{D}} w(\mathbf{x})f(\mathbf{x})\lambda(\mathbf{x})d\mathbf{x}. \tag{4.1}$$

In order to derive $\mathbb{E}_{\mathcal{P}}[\hat{I}^2]$, we first rewrite it as $\mathbb{E}_{\mathcal{P}}\left[\sum_{i\neq j} w_i f_i w_j f_j\right] + \mathbb{E}_{\mathcal{P}}\left[\sum (w_i f_i)^2\right]$. Using Equations 2.2 and 2.1 for the first and second terms, respectively

$$\mathbb{E}_{\mathcal{P}}\left[\hat{I}^2\right] = \int_{\mathcal{D}\times\mathcal{D}} w(\mathbf{x})f(\mathbf{x})w(\mathbf{y})f(\mathbf{y})\varrho(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} + \int_{\mathcal{D}} w^2(\mathbf{x})f^2(\mathbf{x})\lambda(\mathbf{x})d\mathbf{x}. \tag{4.2}$$

Note that although the integrals are over the support $\mathcal{D}$ of the function f, we could equivalently write them over $\mathbb{R}^d$. Using these expressions, we can then write the expressions for bias $bias_{\mathcal{P}}[\hat{I}] = I - \mathbb{E}_{\mathcal{P}}[\hat{I}]$, and variance $var_{\mathcal{P}}[\hat{I}] = \mathbb{E}_{\mathcal{P}}[\hat{I}^2] - (\mathbb{E}_{\mathcal{P}}[\hat{I}])^2$ as

$$bias_{\mathcal{P}}[\hat{I}] = \int_{\mathcal{D}} w(\mathbf{x})f(\mathbf{x})\lambda(\mathbf{x})d\mathbf{x} - I, \tag{4.3}$$

$$
\begin{aligned}
var_{\mathcal{P}}[\hat{I}] = \int_{\mathcal{D}\times\mathcal{D}} w(\mathbf{x})f(\mathbf{x})w(\mathbf{y})f(\mathbf{y})\varrho(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} + \int_{\mathcal{D}} w^2(\mathbf{x})f^2(\mathbf{x})\lambda(\mathbf{x})d\mathbf{x} \\
- \left(\int_{\mathcal{D}} w(\mathbf{x})f(\mathbf{x})\lambda(\mathbf{x})d\mathbf{x}\right)^2 \quad (4.4) \\
= \int_{\mathcal{D}\times\mathcal{D}} w(\mathbf{x})f(\mathbf{x})w(\mathbf{y})f(\mathbf{y})[\varrho(\mathbf{x},\mathbf{y}) - \lambda(\mathbf{x})\lambda(\mathbf{y})]d\mathbf{x}d\mathbf{y} + \int_{\mathcal{D}} w^2(\mathbf{x})f^2(\mathbf{x})\lambda(\mathbf{x})d\mathbf{x}.
\end{aligned}
$$

These expressions prove that error in integral estimation merely depends on the first and second order product densities of a point process. This is a considerable simplification and justifies our focus on first and second order statistics λ and ϱ. Sampling with adaptive density, such as importance sampling, is nicely captured by $\lambda(\mathbf{x})$. Note, though, that such an adaptive density can in general also affect $\varrho(\mathbf{x},\mathbf{y})$. Later we will show that we can actually model adaptivity to factorize the effect of it on ϱ. But we first focus on a simpler and yet very common case.

4.2 Stationary Point Patterns

A very important special case is when the points in a pattern do not follow any given probability density, i.e. the pattern is indifferent to where we are in space. Unless importance sampling is introduced, most patterns in rendering fall into this category. These patterns can be conveniently modelled by stationary or isotropic point processes, or their approximations. These will also form the basis for the analysis of adaptive patterns.

As discussed in Section 2.2.2, for stationary point processes $\lambda(\mathbf{x}) = \lambda$, and $\varrho(\mathbf{x},\mathbf{y}) = \lambda^2 g(\mathbf{x}-\mathbf{y})$. As λ is a constant, we choose $w(\mathbf{x}) = 1/(\lambda|\mathcal{D}|)$. Plugging these into Equations 4.3 and 4.4, we get the following expressions for bias and variance for this case

$$
bias_{\mathcal{P}}[\hat{I}] = \frac{\lambda}{\lambda|\mathcal{D}|}\int_{\mathcal{D}} f(\mathbf{x})d\mathbf{x} - I = 0, \tag{4.5}
$$

$$
var_{\mathcal{P}}[\hat{I}] = \frac{1}{|\mathcal{D}|^2}\int_{\mathcal{D}\times\mathcal{D}} f(\mathbf{x})f(\mathbf{y})[g(\mathbf{x}-\mathbf{y}) - 1]d\mathbf{x}d\mathbf{y} + \frac{1}{|\mathcal{D}|^2\lambda}\int_{\mathcal{D}} f^2(\mathbf{x})d\mathbf{x}. \tag{4.6}
$$

A few important points can be directly observed here:

- The dependence on intensity λ appears only in the second term explicitly, and for a point process generating completely random distributions with $g(\mathbf{r}) = 1$, the variance $var_{\mathcal{P}}[\hat{I}] = \frac{1}{|\mathcal{D}|^2\lambda}\int_{\mathcal{D}} f^2(\mathbf{x})d\mathbf{x}$ decreases with λ^{-1} as well-known. For patterns with second order correlations, the first term will contribute to decreasing the error, and hence affect convergence.
- The goal of unadaptive sampling algorithms is then finding a g in an offline step that will adapt to typical functions encountered in practice to minimize the first term. Note that g compresses as λ is increased, as the distances between points are getting smaller for higher number of points. The final convergence rate with respect to λ thus depends on the interplay between g and the class of integrands considered.

Spectral Analysis

In order to derive the spectral form of error, we need to assume a toroidal domain for the space on which the point process is defined [20, 22]. We thus assume that $\mathcal{D}$ is toroidal with unit volume. The expression in Equation 4.6 can then also be written in the following form

$$var_{\mathcal{P}}[\hat{I}] = \frac{1}{\lambda}\int_{\mathcal{D}} f^2(\mathbf{x})d\mathbf{x} + \int_{\mathbb{R}^d} a_f(\mathbf{r})g(\mathbf{r})d\mathbf{r} - \left(\int_{\mathcal{D}} f(\mathbf{x})d\mathbf{x}\right)^2. \tag{4.7}$$

This form with the autocorrelation of f provides a direct link to the spectral counterpart. In order to derive the spectral expression, first note that $\mathcal{F}[a_f(\mathbf{r})](\boldsymbol{\nu}) = P_f(\boldsymbol{\nu})$, where $\mathcal{F}$ is the Fourier transform, and $P_f(\boldsymbol{\nu})$ is the power spectrum of the function f. Following Equation 3.13, we can also easily show that $P(0) = \frac{1}{\lambda}\mathbb{E}_{\mathcal{P}}[\sum_{ij} 1] = \frac{\lambda^2}{\lambda} = \lambda$. Utilizing properties of the Fourier transform, and the relation $P(\boldsymbol{\nu}) = \lambda G(\boldsymbol{\nu}) + 1$ (Equation 3.17), we can then write the expression in the spectral domain as follows:

$$\begin{aligned} var_{\mathcal{P}}[\hat{I}] &= \frac{1}{\lambda}\int P_f(\boldsymbol{\nu})d\boldsymbol{\nu} + \int P_f(\boldsymbol{\nu})G(\boldsymbol{\nu})d\boldsymbol{\nu} - P_f(0) \\ &= \int \left(\frac{P_f(\boldsymbol{\nu})}{\lambda} + P_f(\boldsymbol{\nu})\frac{P(\boldsymbol{\nu})-1}{\lambda}\right)d\boldsymbol{\nu} - P_f(0)\frac{P(0)}{\lambda} \\ &= \frac{1}{\lambda}\int_{\boldsymbol{\nu}\neq 0} P_f(\boldsymbol{\nu})P(\boldsymbol{\nu})d\boldsymbol{\nu}. \end{aligned} \tag{4.8}$$

Depending on the problem and class of functions considered, one can thus either try to adapt the PCF using Equation 4.6, or the power spectrum with the above equation.

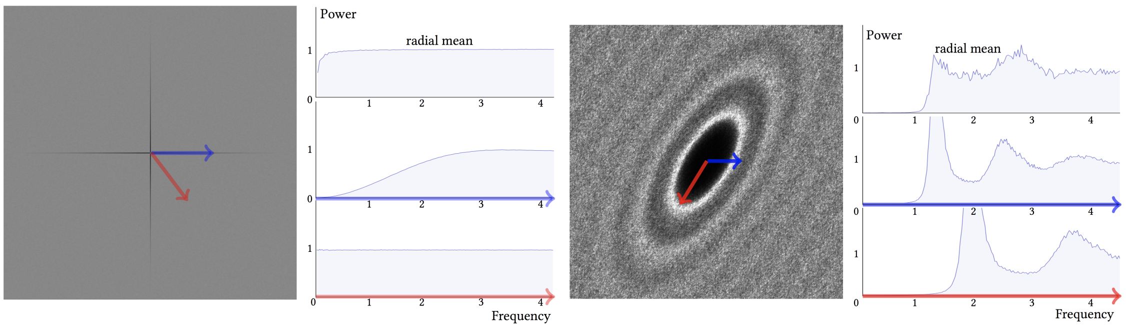
4.2.1 Isotropic Point Patterns

For isotroic point patterns generated by isotropic point processes, we can simply use the fact that $g(\mathbf{r}) = g(r)$, and $P(\boldsymbol{\nu}) = P(\nu)$, i.e. both are radially symmetric. Then the integrals in the second term of Equation 4.7, and Equation 4.8 can be integrated along radial and all non-radial directions individually, giving rise to radial averages of a_f or P_f multiplied and integrated with $g(r)$ or $P(\nu)$, respectively.

To analyze the variance and convergence rate of specific sampling patterns, Pilleboue et al. [22] further simplify Equation 4.8 by going to polar coordinates and collapsing the integrand's power spectrum P_f and the expected sampling power spectrum P —under the assumption of isotropic sampling power spectra—into their radial averages $\breve{P}_f$ and $\breve{P}$ arriving at

$$var_{\mathcal{P}}[\hat{I}] = \frac{1}{\lambda}\int_0^{\infty} \rho^{d-1}\breve{P}_f(\rho)\breve{P}(\rho)d\rho. \tag{4.9}$$

With this simplification, their primary contribution was showing that if the radially averaged sampling power spectra can be expressed analytically, then the corresponding variance convergence rates can be derived for a given class of functions. To more easily apply this idea to complex radial power spectra, they showed that it is often sufficient to piecewise bound the radial mean power spectrum using a monomial in the low-frequency region and a constant for high frequencies, with the degree of the low-frequency monomial bound ultimately determining the convergence rate. Unfortunately, by relying on radially averaged power spectra, Pilleboue et al.'s analysis only truly applies to isotropic point sampling spectra. This also restricts the scope of their convergence tools since the radial mean would not take into account the anisotropy present within the integrand and therefore, cannot exploit it to improve convergence rates.



(a) N-rook power spectrum (b) Blue noise power spectrum

Figure 4.1: Anisotropic structures within N-rooks sampling spectrum in (a) are due to the underlying construction which enforces dense 1D stratification along the canonical X and Y axes. This results in hairline anisotropies along the canonical axes in the resulting power spectrum. In (b), the isotropic blue noise sampling power spectrum is sheared (e.g. to adapt to an integrand) that causes anisotropy within samples and its spectrum. Note that these anisotropic structures are not reflected in their radially averaged counterpart.

4.2.2 Anisotropy in Point Patterns

Correlations present within the samples can be adapted to the integrand to improve error. However, this might result in introducing certain kind of anisotropy within the samples. Anisotropy can also be introduced from the way samples are constructed. For example, N-rooks, jittered and multi-jittered samplers are classic examples of anisotropic samplers (Figure 4.1).

While radial averaging is appropriate for analyzing isotropic Fourier power spectra, for anisotropic sampling power spectra, radial averaging can be less informative, or worse, misleading. For example, in Figure 4.1, the 2D N-rooks sampling pattern has radial behavior of a jittered sampling power spectrum along the canonical axes, but a flat, white noise radial behavior in other directions. This information is lost in the radially averaged power spectrum shown at the top of the radial plots. Most of the signals that we encounter in light transport are also anisotropic in nature, with their spectra having most of their energy confined to a wedge shape [5]. Existing sampling patterns, including quasi-random samples (e.g. Halton, Sobol), have not been able to exploit this knowledge despite having strong anisotropic properties in most projections. Singh and Jarosz [27] establishes a direct relation between the anisotropy of the sampler and the integrand under study, generalizing and extending the reach of prior analyses [22] that relied on radial averaging. Singh and Jarosz proposed the following variance formulation for any anisotropic sampling power spectra:

$$var_{\mathcal{P}}[\hat{I}] = \frac{1}{\lambda} \lim_{m\to\infty} \sum_{k=1}^{m} \int_0^{\infty} \rho^{d-1} \mathbb{E}\left[P(\rho \mathbf{n}_k)\right] P(\rho \mathbf{n}_k) d\rho \, \Delta \mathbf{n}_k \tag{4.10}$$

where, $\Delta\mathbf{n}_k$ is the differential volume of the k-th cone. In the limit, no angular variation is assumed within the k-th differential cone, which allows to consider a single direction corresponding to each cone. Here, $\Delta\mathbf{n}_k$ is a constant that becomes infinitesimally small as m tends to infinity. From (4.10), variance of Monte Carlo integration can be obtained by summing the radially integrated terms along each individual direction $\mathbf{n}_k$. This implies that, irrespective of whether our expected sampling power spectrum is isotropic or not, we can analyze each direction independently to know the overall behavior of the underlying sampler.

4.2.3 Quasi-Monte Carlo Sampling

Distributions generated by Quasi-Monte Carlo sampling algorithms are not strictly random, as they result from deterministic processes. However, ideally, a quasi-Monte Carlo sampling algorithm should mimic a stationary point process. Indeed, more advanced techniques such as Halton and Sobol sequence sampling generate almost stationary point patterns, and hence lead to minimally biased integration results. We can thus use intensity and pair correlation function to study such patterns as well.

4.3 Point Patterns with Non-constant Intensity

In practice, adaptivity is an important concern in point patterns used in rendering, as it can be shown that adaptivity improves convergence rates. In this section, we show how adaptive density can be modeled within the framework of point processes, and the analysis for the constant intensity case can be extended for the common ways adaptivity is introduced into patterns in rendering. Adaptive sampling is obtained by assuming an interaction model between points, such as repulsion, and setting a specialized spatially varying density for a given integrand, which changes both λ and ϱ. There are several ways of introducing it.

Locally Scaled Processes and Warping

One way of defining a point process with adaptive density is via defining a distance measure such that distances become smaller in areas of high density. This is the idea behind several extensions of blue noise sampling commonly used for artistic applications and non-photorealistic rendering [3, 14, 30], and also known as locally scaled point processes in statistics [8]. Although useful for applications such as halftoning and stippling [6, 24], distributions from such point processes can only be generated via costly algorithms. Hence, such point processes are not used in rendering.

Another way to implement adaptive density is by starting from a non-adaptive, e.g. stationary, point process, and warping the resulting distributions with a given function. This is typically used in rendering for importance sampling, i.e. to place more samples into regions where the integrand has higher values. For these cases, analysis is challenging as $\varrho(\mathbf{x},\mathbf{y}) = g(t^{-1}(\mathbf{x}) - t^{-1}(\mathbf{y}))$ for a warping function t, and PCF g of the underlying stationary process to start with.

Unbiased Adaptive Sampling

The expressions start to simplify if we assume an unbiased estimator with $w(\mathbf{x}) = 1/\lambda(\mathbf{x})$. Starting from Equation 4.4, the variance in this case can be easily rewritten as

$$var_{\mathcal{P}}(\hat{I}) = \int \frac{f^2(\mathbf{x})}{\lambda(\mathbf{x})} d\mathbf{x} + \int f(\mathbf{x}) f(\mathbf{y}) \frac{\varrho(\mathbf{x},\mathbf{y})}{\lambda(\mathbf{x})\lambda(\mathbf{y})} d\mathbf{x} d\mathbf{y} - \left(\int f(\mathbf{x}) d\mathbf{x} \right)^2 . \tag{4.11}$$

The first term shows the well-known observation in importance sampling that the intensity should follow the integrand for minimal variance. Note, however, that this is only true if no adaptivity is assumed for ϱ. Conversely, if we fix a certain density by fixing the intensity $\lambda(\mathbf{x})$, $\varrho(\mathbf{x},\mathbf{y})$ should be kept as small as possible when the term $\frac{f(\mathbf{x})f(\mathbf{y})}{\lambda(\mathbf{x})\lambda(\mathbf{y})}$ is high.

Intensity-reweighted Stationary Point Processes

Another set of techniques estimate measures such as the variation of the integrand, and distribute more samples in those regions, often iteratively [2, 7, 19, 31]. The variance of the resulting point patterns can be described by second-order intensity-reweighted stationary point processes [11].

For these processes, $\varrho(\mathbf{x}, \mathbf{y}) = \lambda(\mathbf{x})\lambda(\mathbf{y})g(\mathbf{x} - \mathbf{y})$. We can plug this form of ϱ into the equations for bias and variance of general point processes in Section 4.1, with the definition $v(\mathbf{x}) = \lambda(\mathbf{x})w(\mathbf{x})f(\mathbf{x})$ to get

$$bias_{\mathcal{P}}(\hat{I}) = I - \int_{\mathcal{D}} v(\mathbf{x})d\mathbf{x}, \tag{4.12}$$

$$var_{\mathcal{P}}(\hat{I}) = \int_{\mathcal{D}} \frac{v^2(\mathbf{x})}{\lambda(\mathbf{x})} d\mathbf{x} + \int_{\mathcal{D}\times\mathcal{D}} v(\mathbf{x})v(\mathbf{y})(g(\mathbf{x} - \mathbf{y}) - 1)d\mathbf{x}d\mathbf{y}. \tag{4.13}$$

For unbiased sampling, $w(\mathbf{x}) = 1/\lambda(\mathbf{x})$, and hence $v(\mathbf{x}) = f(\mathbf{x})$. In this case, we get the most simplified form for variance when assuming an adaptive model

$$\begin{aligned} var_{\mathcal{P}}(\hat{I}) &= \int_{\mathcal{D}} \frac{f^2(\mathbf{x})}{\lambda(\mathbf{x})} d\mathbf{x} + \int_{\mathcal{D}\times\mathcal{D}} f(\mathbf{x})f(\mathbf{y})(g(\mathbf{x} - \mathbf{y}) - 1)d\mathbf{x}d\mathbf{y} \\ &= \int_{\mathcal{D}} \frac{f^2(\mathbf{x})}{\lambda(\mathbf{x})} d\mathbf{x} + \int_{\mathbb{R}^d} a_f(\mathbf{r})g(\mathbf{r})d\mathbf{r} - \left(\int_{\mathcal{D}} f(\mathbf{x})d\mathbf{x}\right)^2. \end{aligned} \tag{4.14}$$

We see a clear a separation of the effect of the first order λ, and second order correlations captured by g in this case. The first term tells us to set λ proportional to f, as in importance sampling, and the second term suggests a further optimization depending on a_f. Note that classical importance sampling is a special case with $g(\mathbf{r}) = 1$.

5. Conclusions

5.1 Recent Developments and Future Directions

The main motivation behind this work has been the theoretical advances to understand the interplay between first and second order correlations of point patterns, and integrands encountered in rendering. As we elaborated on in previous sections, it is now possible to study the full range of correlations at the same time for choosing best sampling patterns. We believe, however, that these are yet to be utilized for more practical improvements.

One problem is the high dimensional nature of $\varrho(\mathbf{x}, \mathbf{y})$, especially for the high dimensional integrands encountered in rendering, creating problems for visualization and developing insight. This is currently handled by assuming a stationary point process, and 2-dimensional slices from the pair correlation function. Although there are attempts in other domains [23], statistics summarization and visualization remain a challenge.

If the integrand is known, or can be estimated on the fly, point distribution can be adapted accordingly. So far, this has been the standard practice for first order correlations, i.e. point density, in importance sampling. Recently, adapting anisotropy for stationary point processes [27] or general correlations with the assumption of locally stationary point processes [23] have been proposed. We believe general adaptations with fast algorithms integrated well with rendering frameworks can significantly improve render times and quality.

Finally, although it is now possible to carefully define forms for point process statistics based on extracted or learned properties of integrands, and such statistics can then be used to synthesize new point distributions with existing algorithms [1, 9, 12, 21, 29, 32], such synthesis techniques are still either costly, or does not extend to progressive high dimensional sampling. Developing efficient and progressive algorithms that scale well with dimension and integrate with current renderers remains a major challenge.

5.2 Resources for Stochastic Point Processes

Please check our course repository for a list of sources: https://github.com/sinbag/SamplingAnalysisWithCorrelations.

Bibliography

Books

[11] Janine Illian et al., editors. *Statistical Analysis and Modelling of Spatial Point Patterns*. John Wiley and Sons, Ltd., 2008 (cited on pages 17, 21, 41).

[18] Jesper Møller and Rasmus Plenge Waagepetersen. *Statistical inference and simulation for spatial point processes*. Boca Raton (Fl.), London, New York: Chapman & Hall/CRC, 2003, 2004. ISBN: 1-584-88265-4 (cited on page 17).

[28] Robert Ulichney. *Digital Halftoning*. Cambridge, MA, USA: MIT Press, 1987. ISBN: 0-262-21009-6 (cited on pages 30, 32).

Articles

[1] Abdalla G. M. Ahmed, Hui Huang, and Oliver Deussen. "AA Patterns for Point Sets with Controlled Spectral Properties". In: *ACM Trans. Graph.* 34.6 (Oct. 2015), 212:1–212:8. ISSN: 0730-0301 (cited on page 43).

[2] Laurent Belcour et al. "5D Covariance Tracing for Efficient Defocus and Motion Blur". In: *ACM Trans. Graph.* 32.3 (July 2013), 31:1–31:18. ISSN: 0730-0301 (cited on page 41).

[3] Jiating Chen et al. "Bilateral Blue Noise Sampling". In: *ACM Trans. Graph.* 32.6 (Nov. 2013), 216:1–216:11. ISSN: 0730-0301 (cited on page 41).

[4] Robert L. Cook. "Stochastic sampling in computer graphics". In: *ACM Trans. Graph.* 5.1 (1986), pages 51–72. ISSN: 0730-0301 (cited on page 33).

[5] Frédo Durand et al. "A Frequency Analysis of Light Transport". In: *ACM Trans. Graph.* 24.3 (July 2005), pages 1115–1126. ISSN: 0730-0301 (cited on page 40).

[6] Raanan Fattal. "Blue-noise Point Sampling Using Kernel Density Model". In: *ACM Trans. Graph.* 30.4 (July 2011), 48:1–48:12. ISSN: 0730-0301 (cited on page 41).

[7] Toshiya Hachisuka et al. "Multidimensional Adaptive Sampling and Reconstruction for Ray Tracing". In: *ACM Trans. Graph.* 27.3 (Aug. 2008), 33:1–33:10. ISSN: 0730-0301 (cited on page 41).

[8] Ute Hahn et al. "Inhomogeneous spatial point processes by location-dependent scaling". In: *Adv. in Appl. Probab.* 35.2 (2003), pages 295–550 (cited on page 41).

[9] Daniel Heck, Thomas Schlömer, and Oliver Deussen. "Blue Noise Sampling with Controlled Aliasing". In: *ACM Trans. Graph.* 32.3 (July 2013), 25:1–25:12. ISSN: 0730-0301 (cited on pages 14, 32, 43).

[10] F.J. Hickernell. In: *John Wiley & Sons, Ltd,* (2014) (cited on page 23).

[12] Bhavya Kailkhura et al. "Stair Blue Noise Sampling". In: *ACM Trans. Graph.* 35.6 (Nov. 2016), 248:1–248:10. ISSN: 0730-0301. DOI: 10.1145/2980179.2982435. URL: http://doi.acm.org/10.1145/2980179.2982435 (cited on page 43).

[13] Ares Lagae and Philip Dutré. "A Comparison of Methods for Generating Poisson Disk Distributions". In: *Comput. Graph. Forum* 27.1 (2008), pages 114–129 (cited on page 33).

[14] Hongwei Li et al. "Anisotropic Blue Noise Sampling". In: *ACM Trans. Graph.* 29.6 (Dec. 2010), 167:1–167:12. ISSN: 0730-0301 (cited on page 41).

[15] Chongyang Ma, Li-Yi Wei, and Xin Tong. "Discrete element textures". In: *ACM Trans. Graph.* 30.4 (Aug. 2011), 62:1–62:10. ISSN: 0730-0301 (cited on page 14).

[16] Chongyang Ma et al. "Dynamic Element Textures". In: *ACM Trans. Graph.* 32.4 (July 2013), 90:1–90:10. ISSN: 0730-0301. DOI: 10.1145/2461912.2461921. URL: http://doi.acm.org/10.1145/2461912.2461921 (cited on page 14).

[17] Don P. Mitchell. "Spectrally Optimal Sampling for Distribution Ray Tracing". In: *SIGGRAPH Comput. Graph.* 25.4 (July 1991), pages 157–164. ISSN: 0097-8930 (cited on page 31).

[19] Ryan S. Overbeck, Craig Donner, and Ravi Ramamoorthi. "Adaptive Wavelet Rendering". In: *ACM Trans. Graph.* 28.5 (Dec. 2009), 140:1–140:12. ISSN: 0730-0301 (cited on page 41).

[20] A. Cengiz Öztireli. "Integration with Stochastic Point Processes". In: *ACM Trans. Graph.* 35.5 (Aug. 2016), 160:1–160:16. ISSN: 0730-0301 (cited on pages 14, 21, 37, 38).

[21] A. Cengiz Öztireli and Markus Gross. "Analysis and Synthesis of Point Distributions Based on Pair Correlation". In: *ACM Trans. Graph.* 31.6 (Nov. 2012), 170:1–170:10. ISSN: 0730-0301 (cited on pages 14, 28, 30, 43).

[22] Adrien Pilleboue et al. "Variance Analysis for Monte Carlo Integration". In: *ACM Trans. Graph.* 34.4 (July 2015), 124:1–124:14. ISSN: 0730-0301 (cited on pages 32, 37, 38, 40).

[23] Riccardo Roveri, A. Cengiz Öztireli, and Markus Gross. "General Point Sampling with Adaptive Density and Correlations". In: *Computer Graphics Forum* (2017) (cited on page 43).

[24] Christian Schmaltz et al. "Electrostatic Halftoning". In: *Comput. Graph. Forum* 29.8 (2010), pages 2313–2327. ISSN: 1467-8659 (cited on page 41).

[25] Christian Schumacher, Bernhard Thomaszewski, and Markus Gross. "Stenciling: Designing Structurally-Sound Surfaces with Decorative Patterns". In: *Computer Graphics Forum* (2016). ISSN: 1467-8659. DOI: 10.1111/cgf.12967 (cited on page 14).

[27] Gurprit Singh and Wojciech Jarosz. "Convergence Analysis for Anisotropic Monte Carlo Sampling Spectra". In: *ACM Trans. Graph.* 36.4 (July 2017), 137:1–137:14. ISSN: 0730-0301. DOI: 10.1145/3072959.3073656. URL: http://doi.acm.org/10.1145/3072959.3073656 (cited on pages 40, 43).

[29] Florent Wachtel et al. "Fast Tile-based Adaptive Sampling with User-specified Fourier Spectra". In: *ACM Trans. Graph.* 33.4 (July 2014), 56:1–56:11. ISSN: 0730-0301 (cited on page 43).

[30] Li-Yi Wei and Rui Wang. "Differential domain analysis for non-uniform sampling". In: *ACM Trans. Graph.* 30.4 (July 2011), 50:1–50:10. ISSN: 0730-0301 (cited on pages 26, 27, 41).

[31] Turner Whitted. "An Improved Illumination Model for Shaded Display". In: *Commun. ACM* 23.6 (June 1980), pages 343–349. ISSN: 0001-0782 (cited on page 41).

[32] Yahan Zhou et al. "Point Sampling with General Noise Spectrum". In: *ACM Trans. Graph.* 31.4 (July 2012), 76:1–76:11. ISSN: 0730-0301 (cited on page 43).