SAARLAND UNIVERSITY

Faculty of Mathematics and Computer Science
Department of Computer Science

Toward Improving Monte Carlo Estimator for Rendering and Machine Learning

Dissertation zur Erlangung des Grades des Doktors der Ingenieurwissenschaften der Fakultät für Mathematik und Informatik der Universität des Saarlandes

Corentin Salaün

Saarbrücken
2025

Date of the defense: December 9, 2025

Dean: Prof. Dr. Roland Speicher

Defense committee:

Chair: Prof. Dr. Philipp Slusallek

Reviewers :
Prof. Dr. Hans-Peter Seidel
Dr. Gurprit Singh
Dr. Nicolas Bonneel

Academic assistant: Dr. Gereon Fox

UNIVERSITÄT
DES
SAARLANDES

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Declaration of original authorship

I hereby declare that this dissertation is my own original work except where otherwise indicated. All data or concepts drawn directly or indirectly from other sources have been correctly acknowledged. This dissertation has not been submitted in its present or similar form to any other academic institution either in Germany or abroad for the award of any other degree.

Saarbrücken, ***09/2025***

Corentin Salaün

Abstract

Monte Carlo integration is a fundamental computational tool for estimating high-dimensional integrals that cannot be solved analytically. Its ability to handle complex domains and irregular functions makes it indispensable in computer graphics. One classical application is physically-based rendering that uses Monte Carlo integration to simulate the transport of light with photorealistic accuracy. Similar challenges arise in machine learning, where stochastic gradient estimation underpins the training of modern models and requires high-dimensional gradient estimation. In both domains, the accuracy and efficiency of Monte Carlo methods directly determine the quality of the final results.

A central difficulty of Monte Carlo integration is the high variance of its estimators, which leads to noisy images in rendering and slow convergence in learning. Over the years, a variety of variance-reduction techniques have been developed to alleviate this problem, such as control variates, importance sampling, or carefully designed sampling patterns. These strategies have enabled impressive advances, from photorealistic global illumination to faster neural network training, but they generally need handcrafted inputs.

This thesis introduces a set of new variance reduction techniques for both rendering and machine learning. It proposes adaptive control variates that automatically learn an optimal control function from data, removing the need for hand-crafted designs while guaranteeing provable variance reduction. A scalable multi-class sampling framework has been developed to generate a single set of samples that simultaneously satisfies multiple, potentially conflicting target distributions. This framework has been further extended to optimize perceptual image quality by incorporating models of human visual sensitivity. Finally, the thesis presents efficient adaptive importance-sampling algorithms for stochastic gradient estimation, including a multi-distribution extension that combines several proposals with optimal weights to accelerate training. Together, these contributions advance the theoretical foundations of Monte Carlo integration and deliver practical algorithms that reduce error, improve efficiency, and enable new applications in photorealistic rendering and large-scale machine learning.

Zusammenfassung

Die Monte-Carlo-Integration ist ein grundlegendes Berechnungswerkzeug zur Schätzung hochdimensionaler Integrale, die analytisch nicht gelöst werden können. Aufgrund ihrer Fähigkeit, komplexe Domänen und unregelmäßige Funktionen zu verarbeiten, ist sie in der Computergrafik unverzichtbar. Eine klassische Anwendung ist das physikalisch basierte Rendering, bei dem die Monte-Carlo-Integration verwendet wird, um den Transport von Licht mit fotorealistischer Genauigkeit zu simulieren. Ähnliche Herausforderungen ergeben sich im maschinellen Lernen, wo die stochastische Gradientenschätzung die Grundlage für das Training moderner Modelle bildet und eine hochdimensionale Gradientenschätzung erfordert. In beiden Bereichen bestimmen die Genauigkeit und Effizienz der Monte-Carlo-Methoden direkt die Qualität der Endergebnisse.

Eine zentrale Schwierigkeit der Monte-Carlo-Integration ist die hohe Varianz ihrer Schätzer, die zu verrauschten Bildern beim Rendern und einer langsamen Konvergenz beim Lernen führt. Im Laufe der Jahre wurden verschiedene Techniken zur Varianzreduzierung entwickelt, um dieses Problem zu mildern, darunter Kontrollvariablen, Importance Sampling oder sorgfältig entworfene Stichprobenmuster. Diese Strategien haben beeindruckende Fortschritte ermöglicht, von fotorealistischer globaler Beleuchtung bis hin zu schnellerem Training neuronaler Netze, erfordern jedoch in der Regel manuell erstellte Eingaben.

Diese Arbeit stellt eine Reihe neuer Methoden zur Varianzreduktion sowohl im Rendering als auch im maschinellen Lernen vor. Sie schlägt adaptive Kontrollvariablen vor, die automatisch eine optimale Kontrollfunktion aus Daten lernen, wodurch manuell erstellte Designs überflüssig werden und gleichzeitig eine nachweisbare Varianzreduktion gewährleistet ist. Es wird ein skalierbares Multi-Class-Sampling-Framework entwickelt, um einen einzigen Satz von Stichproben zu generieren, der gleichzeitig mehrere, potenziell widersprüchliche Zielverteilungen erfüllt. Dieses Framework wird weiter ausgebaut, um die wahrgenommene Bildqualität durch die Einbeziehung von Modellen der menschlichen Sehempfindlichkeit zu optimieren. Schließlich werden in der Arbeit effiziente adaptive Importance-Sampling-Algorithmen für die stochastische Gradientenschätzung vorgestellt, darunter eine Multi-Distributions-Erweiterung, die mehrere Vorschläge mit optimalen Gewichten kombiniert, um das Training zu beschleunigen. Zusammen tragen diese Beiträge zur Weiterentwicklung der theoretischen Grundlagen der Monte-Carlo-Integration bei und liefern praktische Algorithmen, die Fehler reduzieren, die Effizienz verbessern und neue Anwendungen im fotorealistischen Rendering und im groß angelegten maschinellen Lernen ermöglichen.

Acknowledgements

I would first like to express my deepest gratitude to my advisor Gurprit Singh for his invaluable guidance and support throughout the course of this PhD. His expertise and thoughtful feedback have shaped not only this dissertation but also my development as an independent researcher. I am also grateful to Hans-Peter Seidel, director of the department, for providing an inspiring research environment and for his continuous support of my work. My sincere thanks go as well to Karol Myszkowski, whose advice and mentorship during the final stages of my PhD were essential and who offered me the exciting opportunity that follows this thesis.

I am indebted to my co-authors and collaborators, whose contributions have greatly enriched this work. In particular, I am thankful to Xingchang Huang, who was both a collaborator and an office mate. His technical expertise in machine learning, his readiness to help at any time, and the many stimulating discussions we shared were instrumental to the success of several projects. I also owe special thanks to Iliyan Georgiev, whose insight, careful reviews, and tireless help (especially in the final rush to meet paper deadlines) made several submissions possible. Our many conversations about new ideas were a constant source of inspiration. I am further grateful to Adrien Gruson and Toshiya Hachisuka, whose guidance and advice at the beginning of my thesis were crucial to my growth as a researcher and helped lay the foundations for this work.

I would also like to thank all my colleagues at the Max Planck Institute for Informatics for making everyday life at the institute so enjoyable. After the long and isolating months of the COVID-19 pandemic, it was a true pleasure to spend these years working, learning, and sharing memorable moments with all of you.

Finally, I am deeply thankful to my friends and family for their unwavering support, not only throughout the challenges of this PhD but also during the many years that led me here. Your encouragement and belief in me have been indispensable to my journey. Last but certainly not least, my heartfelt thanks go to Laure Muller, whom I met during this adventure and with whom I now share my life.

Contents

List of Figures

Chapter 1
Introduction

Integration is a fundamental task across many areas of computer science and applied mathematics. It appears in computer graphics for simulating light transport, in machine learning for computing expectations or gradients under probabilistic models, in physics for modeling particle dynamics, in finance for pricing derivatives, and in Bayesian statistics for marginal likelihood estimation. In many of these domains, computing integrals is essential for solving core problems.

In some cases, *analytic integration* is possible, and the integral can be expressed in closed form using known functions. However, this is not always feasible. Many real-world functions are high-dimensional, lack a known antiderivative, are defined piecewise, or exhibit structure that makes symbolic integration intractable. In some cases, analytic integration may be computationally feasible but requires significant time and resources, making it impractical. In such cases, *numerical estimation* of the integral is necessary.

Numerical integration methods fall into several categories: *quadrature rules*, which approximate the function with a weighted sum of evaluations at fixed points; *surrogate modeling*, which approximates the function with a simpler one; and *statistical estimators*, which use random sampling to estimate the integral. Among these, *Monte Carlo (MC) integration* stands out due to its simplicity, generality, and scalability to high dimensions.

1.1 Monte Carlo Method For Integral Estimation

Monte Carlo integration uses random sampling to estimate the expected value of a function. It requires only the ability to evaluate the function at arbitrary points in the integration domain. No antiderivative or closed-form knowledge is needed to estimate the integral. One interpretation of the integration is to find a constant function with the same integral of the function. This constant function can be obtain by averaging values of the function. The method is *unbiased* and has the attractive property that its error decreases as more samples are used: specifically, the root-mean-square error decreases at a rate proportional to $1/\sqrt{N}$, where N is the number of samples. Importantly, Monte Carlo integration *scales well with dimensionality*, unlike grid-based methods such as quadrature, which suffer from the curse of dimensionality. In fact, Monte Carlo methods can even be applied to problems with effectively infinite dimensions.

This flexibility makes Monte Carlo methods suitable for many complex integration tasks. In particular, they are widely used in *physically based rendering* and in *stochastic gradient estimation* for training machine learning models.

In the context of *rendering*, Monte Carlo integration forms the backbone of physically based light transport simulation. The goal is to estimate the amount of light reaching a virtual camera in a synthetic scene. This can be mathematically formulated as an integral over all light paths that connect light sources to the camera. Each path represents a sequence of interactions (reflections, transmissions, and scatterings) at surfaces within the scene. Accurately computing this high-dimensional integral is essential for generating photorealistic images.

Despite significant progress over the past decades, rendering complex scenes with high visual fidelity remains computationally intensive. In some cases, producing a single noise-free image can take hours or even days. Real-time rendering methods exist, but they operate under different constraints, prioritizing speed and interactivity over physical accuracy and relying on hardware-accelerated approximations.

A key challenge in rendering is variance: unbiased estimators based on random sampling produce noisy images unless many of samples are used. Poorly distributed samples can lead to structured artifacts or a high amount of noise that degrades image quality. Reducing this variance without increasing the number of samples to not slow down rendering and maintaining unbiasedness is a central research problem. Common strategies include importance sampling, control variate, and structured sampling techniques such as stratification, low-discrepancy sequences, and blue noise.

In *machine learning*, Monte Carlo methods play a key role in approximating gradients by integrating the gradient over randomly sampled data. Most modern training procedures rely on variants of stochastic gradient descent (SGD), where the true gradient, defined as the average over the entire dataset, is replaced with a noisy estimate computed from a small subset of data. While the exact gradient can in theory be computed, doing so at each iteration would be prohibitively expensive. This is especially true for large datasets, where computing the gradient with respect to every data set would be impossible. Instead, approximating the gradient introduces noise that, paradoxically, can be beneficial for escaping some local minima. However, excessive variance can significantly slow convergence, destabilize training, or lead to suboptimal solutions.

This gradient estimation process is repeated thousands to billions of times during training, often over the course of days or even weeks. Consequently, each gradient computation must be both fast and as accurate as possible for the training process to be efficient. Reducing the variance of these estimates is therefore critical. To address this, a range of variance reduction techniques have been proposed, including filtering, learned control variate, and importance sampling.

1.2 Summary Of Contributions

This thesis is concerned with advancing error reduction techniques in Monte Carlo methods, with applications in both physically based rendering and stochastic gradient estimation for machine learning. The work contributes to both the theoretical foundations and practical implementations of Monte Carlo estimators. In particular, the thesis develops and evaluates new strategies along three complementary research axes, each addressing a distinct class of challenges that arise in rendering and learning contexts. The main axes of contributions are:

1. Adaptive Control Variate For Monte Carlo Estimation The first axis introduces a new class of Monte Carlo estimators based on *adaptive control variate*. Traditional control variate rely on carefully chosen, fixed functions whose integrals are known in closed form. However, designing effective control variate is often problem-specific and requires deep insight into the integrand's structure.

In this work, we overcome these limitations by defining a *function class* from which the control variate is adaptively learned. Using least-squares regression, we fit an optimal control function that closely approximates the target integrand. Under mild assumptions, this approach provably reduces variance more effectively than traditional, fixed control variate, and it can adapt automatically to complex integrands without prior analytic knowledge.

The contributions within this axis are:

- Definition of adaptive control variate as regression-based approximations of the integrand.
- Demonstration of variance reduction in physically based rendering tasks.

These results detailed in Chapter 4 were presented in a talk at *MCQMC 2022* and published in *ACM SIGGRAPH 2022* (Salaün et al., 2022b).

2. Multi-Class Sampling And Perceptually Optimized Blue Noise The second axis develops a general-purpose *multi-class sampling framework* that addresses the problem of drawing samples for multiple, possibly conflicting, target distributions using a single shared set of samples. This multi-class framework is scalable, supporting up to millions of objectives, and is solved through a novel combination of *filtered sliced optimal transport* and gradient-based optimization.

Beyond classical multi-class tasks (such as progressive sampling, object placement, or stippling), this framework also enables a new formulation of sampling in rendering: optimizing directly for *perceptual error*. Instead of minimizing purely numerical integration error, we incorporate models of the human visual system's sensitivity to noise and reformulate the sampling task as a perceptually driven optimization problem. Optimizing samples for this perceptual error lead to improve image quality in Monte Carlo rendering by reducing visible noise.

The contributions within this axis are:

- A scalable optimization method for producing shared samples across diverse distributions.

- Demonstrating the performance of our scalable framework on classical multi-class applications
- Reformulation of the rendering noise problem into a perceptually driven optimization, leading to visually superior blue noise distributions.

These results are discussed in Chapter 5 and were published in ACM SIGGRAPH Asia 2022 (Salaün et al., 2022a). This project subsequently led to follow-up work extending the single-image perceptual error distribution to animated rendering, published in ACM SIGGRAPH Asia 2023 (Korać et al., 2023). This extension is briefly discussed at the end of Chapter 5.

3. Importance Sampling For Gradient Estimation In Machine Learning The third axis focuses on improving stochastic gradient estimation in machine learning by introducing an efficient importance sampling strategies. In standard mini-batch construction, samples are drawn uniformly, which can lead to high variance and slow convergence. Importance sampling addresses this by focusing the selection toward more informative samples in an unbiased manner. An optimal selection distribution can be computed but is often intractable. In this work we propose a novel efficient sampling distribution that can be used as a replacement of the optimal distribution.

We also show that using a single sampling distribution—although it improves gradient estimation—remains fundamentally limited, because each gradient component ideally requires its own optimal distribution. A single distribution therefore must approximate a mixture of these per-parameter optima, which is inherently suboptimal for all of them. Instead, we propose using multiple dedicated distributions and combining them through multiple importance sampling, yielding significantly better gradient estimates than any single-distribution approach.

Our contributions extend both single-distribution and multi-distribution importance sampling:

- A tractable upper bound on the optimal sampling distribution, enabling variance reduction without prohibitive overhead.
- A multiple importance sampling gradient estimator, combining multiple distribution with provably optimal weights (Kondapaneni et al., 2019). This estimator mitigates the limitations of single-distribution methods and yields more accurate gradient estimates, accelerating convergence across learning tasks.

These contributions are presented in Chapter 6, with peer-reviewed results published in *ICPRAM 2025* (Salaün et al., 2025b,a).

1.3 Thesis Organization

The remainder of this thesis follows the progression of the main research contributions, starting from foundational concepts and related work, then developing each major technique in dedicated chapters.

Chapter 2 – Background: This chapter reviews the fundamental concepts of Monte Carlo integration and techniques for variance reduction, including both improved estimators and sampling strategies. It also introduces the two primary application domains of this thesis: physically based rendering and machine learning. For each domain, it discusses relevant variance reduction techniques commonly used in practice.

Chapter 3 – Related Work Overview: This chapter surveys the most relevant prior work related to the core contributions of this thesis, including adaptive control variate, multi-class sampling, perceptually motivated error distributions, and improved mini-batch sampling techniques.

Chapter 4 – Regression based Control Variate: This chapter presents a new class of Monte Carlo estimators based on regression-driven control variate. By fitting a control variate function to sampled data—such as low-degree polynomials or neural networks—we show that it is possible to obtain improved estimator. Even simple polynomial control variate, when regressed from Monte Carlo samples, can substantially reduce integration error in rendering tasks with minimal overhead.

Chapter 5 – Multi-Class Sampling and Perceptually Based Sampling: This chapter introduces our Wasserstein-based multi-class sampling framework and a new optimization method based on filtered sliced optimal transport. We present several applications in point sampling, including color and animated stippling. Furthermore, we show how perceptual error in rendering can be approximated using a Monte Carlo error bound and formulated as a multi-class sampling problem. By optimizing perceptual error via our framework, we produce sample sets that distribute rendering noise in a perceptually pleasing way.

Chapter 6 – Importance Sampling for Learning: This chapter explores the use of importance sampling and multiple importance sampling (MIS) to improve gradient estimation in machine learning. We propose a novel method for importance sampling in mini-batch construction that adapts the sampling distribution based on training dynamics. Additionally, we extend this approach by combining multiple sampling distributions through optimal MIS, overcoming the limitations of single-distribution methods.

Chapter 7 – Conclusion and Future Work: This chapter summarizes the main findings of the thesis and outlines promising directions for future research.

Chapter 2
Background on Monte Carlo estimator for Rendering and Machine Learning

2.1 Introduction

Monte Carlo methods (Hammersley, 2013; Kalos and Whitlock, 2009) are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. They are particularly useful in numerical applications where expected values must be estimated but cannot be computed analytically due to the vast number of possibilities or the presence of continuous variables. In such cases, Monte Carlo methods generate random samples from the relevant distribution, allowing for an approximation of the entire behavior of the system. This approach introduces an inherent error; however, this error is measurable and can provide a favorable trade-off against the often intractable complexity of exact computations.

Monte Carlo integration (Caflisch, 1998; Glasserman, 2004) is widely used in fields such as physics, finance, and engineering due to its ability to handle complex, multidimensional integrals. This method leverages the law of large numbers, which states that the average of many independent, identically distributed random variables converges to their expected value.

In this chapter, we will explore the theoretical foundations of Monte Carlo integration, examine its applications in physically based rendering, and investigate its role in gradient estimation for machine learning optimization. The structure of the chapter is as follows:

- **Mathematical Background:** This section will develop the theoretical underpinnings of Monte Carlo integration. We will introduce the basic concepts, mathematical formulation, and properties of the Monte Carlo integration estimator. Additionally, we will discuss techniques to improve the efficiency and accuracy of Monte Carlo estimates, such as importance sampling, multiple importance sampling, stratified sampling, and control variate.

- **Physically Based Rendering:** As an application of Monte Carlo integration, this section will focus on physically based rendering (PBR). PBR is a technique in computer graphics that aims to render images in a way that accurately simulates the physical behavior of light. We will cover the rendering equation, light transport simulation methods like direct lighting, path tracing, and bidirectional rendering, as well as advanced techniques to enhance rendering quality and performance.
- **Monte Carlo For Gradient Estimation In Machine Learning Optimization:** This section will explore the use of Monte Carlo methods for gradient estimation in machine learning. Gradient estimation is crucial for optimization algorithms like stochastic gradient descent (SGD). We will discuss stochastic gradient descent and variance reduction methods for accurate gradient estimation.

2.2 Mathematical Basics

2.2.1 Monte Carlo Estimation

The problem of numerical integration arises when an integral cannot be evaluated analytically. Given a function $f(x)$ and an interval $[a, b]$, the goal is to compute the definite integral:

$$I = \int_a^b f(x)\,dx. \tag{2.1}$$

In some cases, the integral can be solved exactly using techniques from calculus. However, for many practical problems, such as high-dimensional integrals, integrals involving highly varying functions, or those with no elementary antiderivative, an exact solution is unattainable. In such cases, numerical methods must be employed to approximate the integral.

Deterministic Methods And Their Limitations One class of numerical integration techniques consists of deterministic methods, which aim to approximate the integral through systematic evaluation of the function at specific points.

A fundamental approach is *Riemann integration*, which partitions the interval $[a, b]$ into n subintervals of width $\Delta x = \frac{b-a}{n}$ and approximates the integral as a sum:

$$I \approx \sum_{i=1}^{n} f(x_i)\Delta x. \tag{2.2}$$

This method forms the basis of numerical quadrature techniques such as the *trapezoidal rule* and *Simpson's rule*, which improve accuracy by refining how function evaluations are combined.

While these deterministic methods work well for low-dimensional problems and smooth functions, they suffer from several limitations: Their accuracy depends on the function's smoothness; highly oscillatory or discontinuous functions can lead to poor approximations. They become computationally expensive in high-dimensional settings, as the

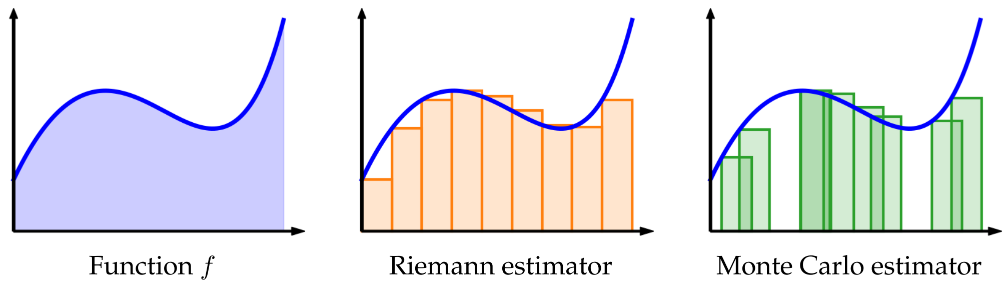


Figure 2.1: Comparison between Riemann integration and Monte Carlo integration. Riemann integration subdivides the domain into equal intervals, while Monte Carlo integration samples random points within the domain.

number of required function evaluations grows exponentially (the so-called "curse of dimensionality").

Statistical Approaches To Integration To overcome the limitations of deterministic methods, statistical approaches based on random sampling, such as *Monte Carlo integration*, offer an alternative framework. Instead of systematically evaluating the function at fixed points, Monte Carlo methods estimate the integral using randomly sampled points from a probability distribution.

As illustrated in fig. 2.1, traditional Riemann integration relies on fixed, evenly spaced boxes to approximate the area under a curve. In contrast, Monte Carlo integration randomly places sample points within the domain, capturing the function's behavior in a more flexible manner. This randomness allows Monte Carlo methods to overcome the curse of dimensionality, as the required number of samples does not necessarily grow exponentially with dimension. The statistical nature of Monte Carlo integration provides a more scalable approach.

Mathematical Definition Of Monte Carlo Integration Monte Carlo integration estimates an integral by leveraging random sampling. Given an integral of the form:

$$F = \int_{\Omega} f(x)\, dx, \tag{2.3}$$

defined over the domain Ω, the Monte Carlo estimator is:

$$\langle I_{MC} \rangle_N = \frac{|\Omega|}{N} \sum_{i=1}^{N} f(x_i), \tag{2.4}$$

where x_i are randomly sampled points from a uniform distribution over the integration domain Ω, and N is the number of samples.

Monte Carlo estimators have essential statistical properties such as unbiasedness and consistency. Unbiasedness means that the expectation of the Monte Carlo estimator equals the true integral value, expressed as $\mathbb{E}[\langle I_{MC} \rangle_N] = F$. It also means multiple

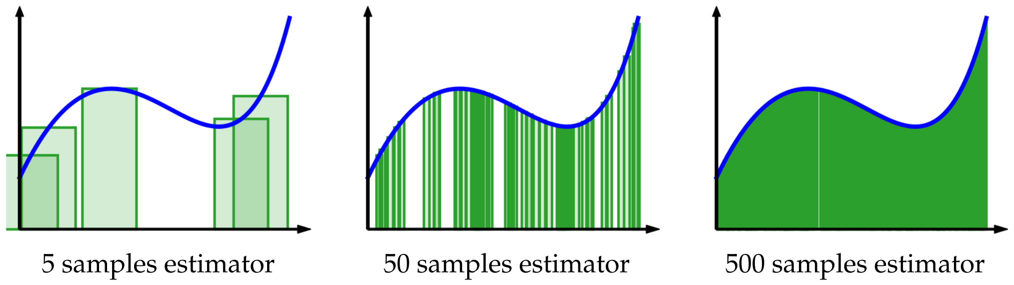


Figure 2.2: Monte Carlo integration of a function with 5, 50, and 500 samples, showing convergence to an accurate approximation.

independent Monte Carlo estimators can be combined, and the result is still unbiased. Consistency indicates that as the number of samples increases, the estimator converges to the true integral $\lim_{N\to\infty}\langle I_{MC}\rangle_N = F$.

The unbiased convergence property is illustrated in fig. 2.2, where Monte Carlo integration is applied to the same function with increasing sample sizes. It is visible that with an increase number of samples, the approximation of the function converge to the exact solution.

The variance of the Monte Carlo estimator plays a crucial role in determining accuracy. It is given by:

$$\mathrm{Var}\left[\langle I_{MC}\rangle_N\right] = \frac{|\Omega|}{N}\mathrm{Var}\left[f(x)\right] = \frac{|\Omega|}{N}\int_\Omega \left(f(x) - F\right)^2 \mathrm{d}x, \tag{2.5}$$

This indicates that the variance of the estimator asymptotically approaches $1/N$, leading to a decrease in the expected error proportional to $1/\sqrt{N}$. Consequently, halving the expected error necessitates a fourfold increase in the number of samples. Notably, both error and variance are theoretically independent of the function's dimensionality. However, in practice $\mathrm{Var}\left[f(x)\right]$ often increases with higher dimensions.

Primary Sample Space In many cases, the domain Ω can be transformed into a unit hypercube $\mathcal{H}$ of the same dimension through a change of variables. This transformation provides a standardized integration domain without altering the mathematical nature of the problem. If we define a mapping $x = \Phi(u)$ with a Jacobian $\left|\frac{\mathrm{d}x}{\mathrm{d}u}\right|$, the integral can be rewritten as:

$$I = \int_\Omega f(x)\,\mathrm{d}x = \int_\mathcal{H} f(\Phi(u))\left|\frac{\mathrm{d}x}{\mathrm{d}u}\right|\mathrm{d}u = \int_\mathcal{H} \hat{f}(u)\,\mathrm{d}u, \tag{2.6}$$

where $\hat{f}(u) = f(\Phi(u))$. Consequently, the Monte Carlo estimator in this space becomes:

$$I \approx \langle I_{MC}\rangle_N = \frac{1}{N}\sum_{i=1}^{N}\hat{f}(u_i). \tag{2.7}$$

This approach, known as the primary sample space Kelemen et al. (2002) allows for consistent treatment of sampling strategies.

Error Reduction For Monte Carlo Integration The inherent stochasticity of the Monte Carlo estimator introduces error. While accuracy statistically improves with increased sample counts, practical applications often face computational limitations, especially when generating each sample is expensive. To overcome this reliance on sheer sample numbers, various techniques aim to reduce estimation error by strategically selecting and utilizing samples. These variance reduction methods enhance the efficiency of Monte Carlo integration.

Variance reduction techniques broadly fall into two categories: improving the Monte Carlo estimator itself (section 2.2.2) and employing correlated samples (section 2.2.3), also known as pseudo-random sampling. The former refines the sampling strategy and the way samples contribute to the integral estimation, encompassing methods like importance sampling, multiple importance sampling, and control variate. These techniques adjust the sampling distribution or introduce auxiliary variables to boost estimator efficiency.

Conversely, pseudo-random sampling methods, such as Quasi-Monte Carlo and blue noise sampling, focus on improving the uniformity and distribution of sample points across the integration domain. By minimizing discrepancies in sample placement compared to purely random sampling, these methods can achieve faster convergence rates and lower error.

2.2.2 Variance Reduction I : Improved Estimator

Importance Sampling Uniform sampling, while straightforward, may not be optimal for Monte Carlo integration, as it assigns equal probability across the domain, disregarding function variations. In practice, some regions exhibit significant variations or high values, contributing more to the integral, while others are less informative. With limited sample budgets, uniform sampling risks missing crucial regions, leading to higher estimation errors.

Non-uniform sampling, guided by a probability density function (PDF) $p(x)$, addresses this. Common techniques include rejection sampling and inverse CDF sampling. Rejection sampling generates candidates from a proposal distribution and accepts them based on criteria ensuring they follow the desired PDF. Inverse CDF sampling transforms uniformly distributed samples using the inverse cumulative distribution function (CDF) of the target distribution, precisely generating samples according to the specified PDF.

However, non-uniform samples require a modified estimator to compensate for the biased distribution. The importance sampling estimator weights function values by the inverse of the sample's PDF:

$$\langle I_{IS} \rangle_N = \frac{|\Omega|}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}, \tag{2.8}$$

where the samples x_i are sample following the distribution $p(x)$. For an unbiased estimator, $p(x) > 0$ where $f(x) \neq 0$, and $p(x)$ must integrate to 1.

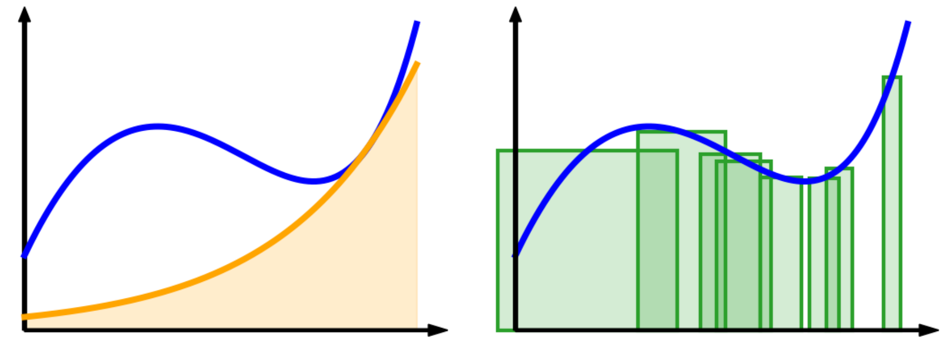


Figure 2.3: Visualization of Importance Sampling. Samples are drawn from a non-uniform distribution. Samples are less likely to be sampled from the left side as the PDF is lower. To compensate boxes are larger in low probability regions.

This normalization of the function value can be interpreted geometrically as the width of the box. The highest the sample probability density, the thinner is the box. Inversely the lower the PDF, the larger the box would be. Statistically a region with low PDF will receive less samples, thus it requires larger box to compensate the lack of samples. Inversely, regions with many samples require thinner boxes. The geometric visualization of the non-uniform sampling and varying-sized boxes is shown in Figure 2.3.

The variance of the importance sampling estimator is:

$$\mathrm{Var}\left[\langle I_{IS}\rangle_N\right] = \frac{|\Omega|}{N}\int_{\Omega}\left(\frac{f(x)}{p(x)} - F\right)^2 \mathrm{d}x, \tag{2.9}$$

where F is the true integral value. Variance is minimized when $p(x)$ is proportional to $|f(x)|$, focusing samples on regions with high function values or variation. While achieving a perfectly proportional PDF requires knowing the integral, even an approximate PDF capturing the function's main features can significantly reduce variance and improve accuracy. Ideally, if f/p is constant, the variance becomes zero. Unfortunately, achieving such a perfect probability density function is not feasible because one would need to know the integral of the function for normalization. This implies that, in order to obtain a perfect sampling distribution, the integration problem must be solved first.

Multiple Importance Sampling Constructing a single PDF that accurately approximates a complex function is often challenging. Multiple importance sampling (MIS) (Veach and Guibas, 1995) combines multiple PDFs, leveraging each distribution's strengths in regions where it aligns well with the function, while minimizing its influence elsewhere.

The MIS Monte Carlo estimator is:

$$\langle I_{MIS}\rangle_N = \sum_{j=1}^{J}\sum_{i=1}^{n_j} w_j(x_{ij})\frac{f(x_{ij})}{n_j p_j(x_{ij})}, \tag{2.10}$$

where $x_{ij} \sim p_j(x)$, and $w_j(x)$ are MIS weights summing to one for each sample ($\sum_{j=1}^{j} w_j(x) = 1$) to ensure unbiasedness. Unlike importance sampling, MIS requires only that at least one PDF is non-zero wherever the function is non-zero. If $p_j(x) = 0$, then $w_j(x)$ must also be zero. The accuracy of such estimator both depend on the choice of PDFs and the weighting strategy between them.

The variance of the MIS estimator depends on the PDFs and weights and is given by:

$$\mathrm{Var}\left[\langle I_{MIS}\rangle_N\right] = \sum_{j=1}^{J} \int_{\Omega} \frac{w_j(x)^2 f(x)^2}{n_j p_j(x)} \mathrm{d}x - \sum_{j=1}^{J} \frac{1}{n_j} \left(\int_{\Omega} w_j(x) f(x) \mathrm{d}x \right)^2, \tag{2.11}$$

Many weighting strategies have been developed for MIS with different strength and weakness. The most common strategies include the balance heuristic (Veach and Guibas, 1995):

$$w_{balance}(x_{ij}) = \frac{n_j p_j(x_{ij})}{\sum_{k=1}^{J} n_k p_k(x_{ij})} \tag{2.12}$$

which balances contributions proportionally to likelihoods and minimizes the first term of the variance eq. (2.11).

A variant of the balance heuristic that gives more emphasis to the high sampling part of each distribution is the power heuristic (Veach and Guibas, 1995):

$$w_{power}(x_{ij}, \beta) = \frac{(n_j p_j(x_{ij}))^{\beta}}{\left(\sum_{k=1}^{J} n_k p_k(x_{ij})\right)^{\beta}} \tag{2.13}$$

This heuristic modifies the balance heuristic with a power β, often reducing error in practice despite lacking formal theoretical support. The choice of β is often set to 2 in practice.

Other strategy that discard distribution under a threshold also exist like the cutoff strategy:

$$w_{cutoff}(x_{ij}) = \begin{cases} \frac{p_j(x_{ij})}{\sum_{k=1}^{J} \{p_k(x_{ij}) | p_k(x_{ij}) \geq \epsilon\}} & \text{if } p_j(x_{ij})) \geq \epsilon \\ 0 & \text{otherwise} \end{cases}. \tag{2.14}$$

This heuristic assigns zero weights to samples with low probabilities, avoiding some extreme values.

Figure 2.4 visualizes the three presented MIS weights to combine two distributions (green and orange), giving importance to two different parts of the function. Other advanced heuristics have been developed, such as maximum heuristic, variance-aware(Grittmann et al., 2019), or correlation-aware heuristics (Grittmann et al., 2021) offering specific advantages, but lack universal optimality.

Lastly, Kondapaneni et al. (2019) derived a variance-minimizing weight giving a optimal set of weight for given sampling distribution:

$$w_j(x_{ij}) = \alpha_j \frac{p_j(x_{ij})}{f(x_{ij})}; + \frac{n_j p_j(x_{ij})}{\sum_{k=1}^{J} n_k p_k(x_{ij})} \left(1 - \frac{\sum_{k=1}^{J} \alpha_k p_k(x_{ij})}{f(x_{ij})} \right), \tag{2.15}$$

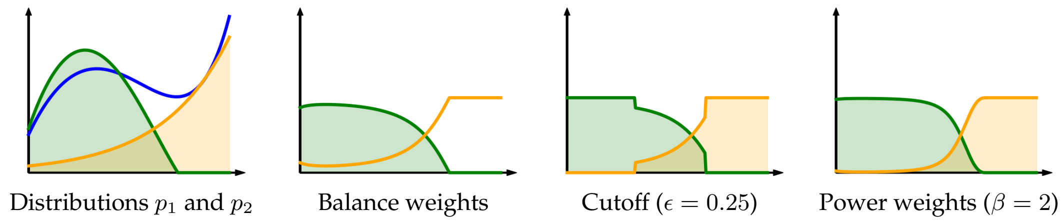


Figure 2.4: Visualization of Multiple Importance Sampling. Different PDFs are used to sample different regions of the function, with each sample weighted appropriately to balance their contributions.

where α is the solution to a linear system. While theoretically optimal, its computational complexity limits its practicality. Nevertheless, this choice offer an upper bound of error reduction using MIS and help measuring how close to optimal a heuristic can be for a given integration problem.

Control Variate Another class of variance reduction techniques is the use of auxiliary functions, denoted g, to simplify the estimator. Instead of directly estimating the integral of the target function f, the *control variate* method estimates the difference between the function and a related auxiliary function whose integral is known analytically.

The idea is to rewrite the function f as:

$$f(x) = \alpha g(x) + (f(x) - \alpha g(x)), \tag{2.16}$$

where α is an arbitrary scaling factor and g is the chosen auxiliary function, called the control variate function.

Using this reformulation, the integral of f becomes:

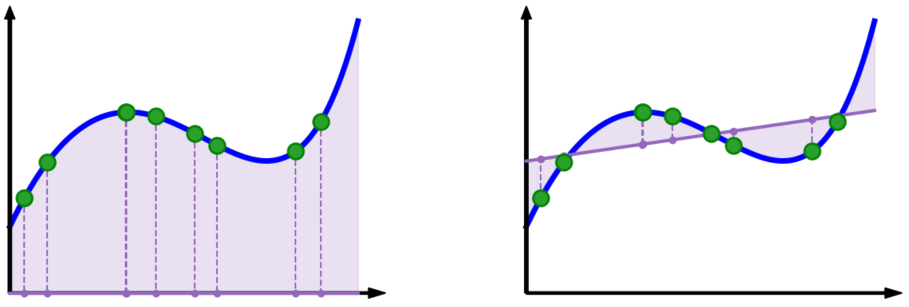
$$\int_\Omega f(x)\,\mathrm{d}x = \alpha G + \int_\Omega (f(x) - \alpha g(x))\,\mathrm{d}x, \tag{2.17}$$

where G is the known integral of g over the domain Ω.

Rather than estimating the original integral directly, we now estimate the integral of the difference between the target function and the control variate. This leads to the following Monte Carlo estimator, known as the control variate estimator:

$$\langle I_{CV} \rangle_N = \alpha G + \frac{|\Omega|}{N} \sum_{i=1}^{N} \frac{(f(x_i) - \alpha g(x_i))}{p(x_i)}, \tag{2.18}$$

Importantly, in this estimator the same set of samples x_i is used to evaluate both the target function $f(x_i)$ and the control variate function $g(x_i)$. This correlation between the two evaluations is essential for variance reduction.



Classical Monte Carlo estimation of f Control variate difference estimation of $f - g$

Figure 2.5: Visualization of the Monte Carlo estimation task and the estimation of the control variate difference. With a well chosen control variate function, the difference estimation as significantly lower variance.

Figure 2.5 illustrate the impact of the control variate estimator. Left side of the figure show the samples of a classical Monte Carlo estimator while the right side show the different $f - g$ evaluated at the same samples. The amount of error in the second estimation is smaller as the magnitude of the difference is smaller that the magnitude of the initial function f. This error reduction arises when the control variate function is correlated with the integrated function and can reduce integration error.

Independently to the choice of the control variate function g, the variance of the control variate estimator can be minimized by choosing the the scaling parameter α as:

$$\alpha = \frac{\mathrm{Cov}\left[f(x), g(x)\right]}{\mathrm{Var}\left[g(x)\right]}. \tag{2.19}$$

Overall, a control variate estimator achieves variance reduction compared to the standard Monte Carlo estimator when the variance of the modified integrand, $f - \alpha g$, is lower than that of the original function f, and the integral G is either analytically known or significantly easier to compute. The effectiveness of this approach depends on the choice of g, which must be tailored to the specific problem at hand. Nevertheless, even simple auxiliary functions that are well correlated with f can yield substantial variance reduction.

2.2.3 Variance Reduction II : Improved Sampling

Another key strategy to reduce variance involves using correlated samples to maximize uniformity over the sampling domain. Well-distributed samples avoid under-sampled or over-represented regions, leading to more accurate and efficient integral estimation. Correlated sampling methods generally falls into 2 main categories: Low discrepancy sampling and Blue noise sampling.

Low Discrepancy Sampling Low discrepancy sampling generates sample sets with a low measure of deviation from perfect uniformity. Discrepancy measure are based on local difference between number of points in a given region of the sampling domain and the expected number of points in the same area. A low discrepancy sample sets ensure

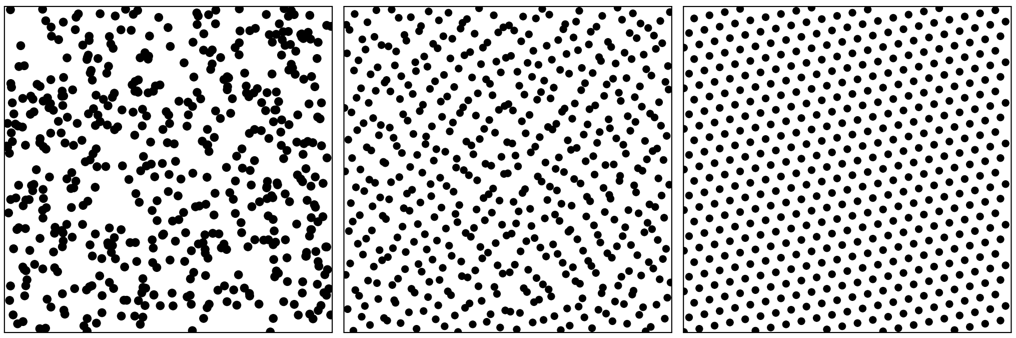


Figure 2.6: Visualization of three sampling patterns: an independent random sampler and two low-discrepancy sequences. While both low-discrepancy methods (right) provide improved coverage of the domain compared to random sampling (left), they differ significantly in structure. The Scrambled Sobol sequence exhibits a more randomized yet well-distributed pattern, whereas the Rank-1 lattice sequence shows highly regular arrangements.

that samples are spread out evenly, reducing clustering and gaps. Figure 2.6 illustrates three sampling patterns: purely random sampling and two low-discrepancy methods (Owen-scrambled Sobol and Rank-1 lattice). Compared to random sampling, where clusters of points and under-sampled regions are clearly visible, both low-discrepancy sequences provide more uniform coverage of the domain. However, there are notable differences between the two. The Owen-scrambled Sobol sequence retains some degree of randomness in the distribution, whereas the Rank-1 lattice exhibits a highly regular and structured pattern.

There are multiple variants of discrepancy metrics, but they all share the same construction strategy. The most common among them is the Star Discrepancy $\mathrm{D}_N^*(S)$:

$$\mathrm{D}_N^*(S) = \sup_{B \in J} \left| \frac{A(B;P)}{N} - \mathrm{Vol}(B) \right|, \tag{2.20}$$

where J is the set of axis-aligned rectangular regions originating from $[0, x)$. Other discrepancy metrics can be constructed using different sets of regions than those found in the Star Discrepancy or by replacing the supremum operator with another form of aggregation, such as the mean operator.

Figure 2.7 left illustrates the concept of star discrepancy. Given a sampling domain and a point set it find the box $[0, x)$ such as the difference of the number of points and the expected number (i.e. the density in the selected box) is maximal in absolute value.

One of the main reasons discrepancy has been studied to evaluate sampling distribution is its ability to indirectly measure the quality of a point set in producing low integration error in a Monte Carlo estimator. This link between the discrepancy of a sample set and Monte Carlo error is given by a Monte Carlo error bound called the Koksma-Hlawka

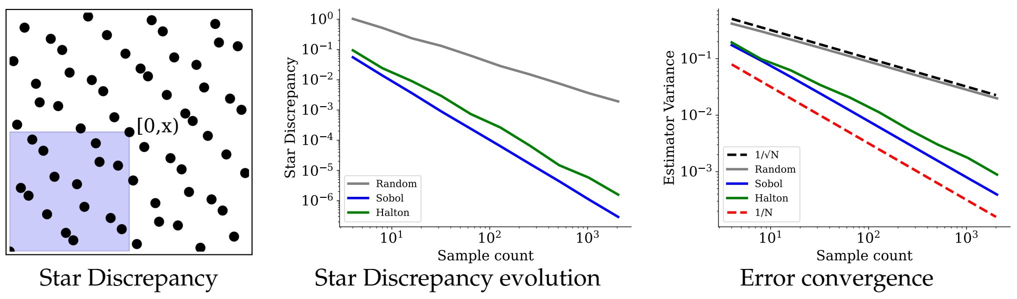


Figure 2.7: Visualization of the star discrepancy (left), which measures the maximum deviation between the empirical sample distribution and the uniform target density over axis-aligned rectangular regions anchored at the lower-left corner. The center and right plots show the evolution of star discrepancy and integration error as a function of the number of samples for random sampling, Sobol, and Halton sequences. Theoretical convergence rates are also indicated. These plots highlight how low-discrepancy sequences achieve significantly better convergence than random sampling.

inequality:

$$\left| \frac{1}{N} \sum_{i=1}^{N} f(x_i) - \int_{\boldsymbol{U}^D} f(x) \mathrm{d}x \right| \leq \mathrm{V}(f) \cdot \mathrm{D}_N^*(S), \qquad x_i \in S \tag{2.21}$$

This inequality bound the integration error (left side) with the product of 2 terms: the function variations in sense of Hardy and Krause$\mathrm{V}(f)$ and the sample set discrepancy $\mathrm{D}_N^*(S)$. The interest is that using such equation reducing the sample set discrepancy reduce the error bound interdependently of the integrated function. Similar bound can be made with other type of discrepancy metrics.

Another interest of low-discrepancy sequences in Monte Carlo integration lies in their superior convergence rate. While classical Monte Carlo estimators exhibit a convergence rate of $\mathcal{O}(1/\sqrt{N})$ with respect to the number of samples N, low-discrepancy sequences can achieve a faster convergence rate of $\mathcal{O}(1/N)$ under certain conditions. This improvement is closely linked to the concept of discrepancy, as illustrated in fig. 2.7 (center and right), where lower discrepancy correlates with reduced variance.

The practical implication of this enhanced convergence is that the same level of noise or error can be attained with significantly fewer samples, resulting in more efficient computations. In summary, low-discrepancy sequences not only provide theoretical guarantees on integration error bounds but also deliver performance gains. Many sequences have been proposed over time to achieve low discrepancy. Among the common low discrepancy sequences are Halton, Sobol, PMJ-02, and Rank-1.

Blue Noise Sampling Blue noise sampling refers to a class of sampling methods that produce sample sets with blue noise properties. Blue noise characteristics imply that the sample set has minimal low-frequency components. Low frequencies typically arise from points being too close together, creating clusters. By constructing point sets where the points are spaced at the maximum possible distance from each other, we naturally achieve an even distribution of points over the domain. Unlike discrepancy methods, which focus on how well the space is covered, blue noise sampling emphasizes the relative distances between points. This distinction is particularly significant when compared to star discrepancy-based samplers, such as Sobol sequences, which may have uneven point distance and may create some cluster. Blue noise sampling methods, by contrast, ensure even spacing in all directions, avoiding such clustering.

The Fourier properties of blue noise sampling are crucial for understanding and verifying its characteristics. The Fourier transform of a point set $P = \{x_1, x_2, \ldots, x_N\}$ in the domain $[0, 1]^d$ can be computed using the following expression:

$$\hat{P}(\mathbf{k}) = \sum_{j=1}^{N} \exp(-2\pi i \mathbf{k} \cdot \mathbf{x}_j), \tag{2.22}$$

where $\mathbf{k}$ is a vector in the frequency domain. A point set with blue noise properties will have a Fourier transform that exhibits a low magnitude for low-frequency components, indicating minimal clustering. Additionally the power spectrum of the point set should show a characteristic peak, a dominant frequency, further confirming the blue noise property. Figure 2.8 shows a comparison between a Progressive Multi-Jittered sequence with on low discrepancy and three different blue noise sampling methods. For each method, the corresponding power spectrum is displayed. The Progressive Multi-Jittered sequence exhibits a cross-shaped pattern in its power spectrum, indicating structured anisotropy. In contrast, all blue noise methods display a characteristic central void in the power spectrum, reflecting the absence of low-frequency components in all directions. An exception can be made for SOT that still shows some anisotropic properties outside the central hole but is still considered as a blue noise sampler.

Another interesting metric used to assess the uniformity of sample distribution is the Wasserstein distance. It provides a measure for the distance between two distributions. These distributions can be continuous or discrete, such as a set of samples. Formally, the Wasserstein distance between two distributions P and Q is defined as:

$$W(P, Q) = \left(\inf_{\gamma \in \Gamma(P,Q)} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\|^p \, d\gamma(x, y) \right)^{1/p}, \tag{2.23}$$

where $\Gamma(P, Q)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are P and Q, respectively. The connection between the Wasserstein distance and blue noise sampling lies in their shared goal of achieving a low distance between a discrete distribution (the sample set) and a given target distribution. To achieve such a close distance, the sample set must efficiently cover the domain according to the target distribution. Blue noise properties ensure that points are evenly spaced and avoid clustering, which corresponds to having a low Wasserstein distance. A sample set with a low Wasserstein distance ensure that the samples are distributed in a manner that closely approximates the target distribution, thereby maintaining the desirable properties of blue noise. This measure provides a quantitative way to assess how well a sampling method achieves blue noise properties, even for non-uniform distribution.

Blue noise sampling methods can be broadly categorized into two types: progressive methods and optimization-based methods. Progressive methods, such as Poisson Disk Sampling (McCool and Fiume, 1992), incrementally generate samples while ensuring that each new sample maintains a minimum distance from existing samples.

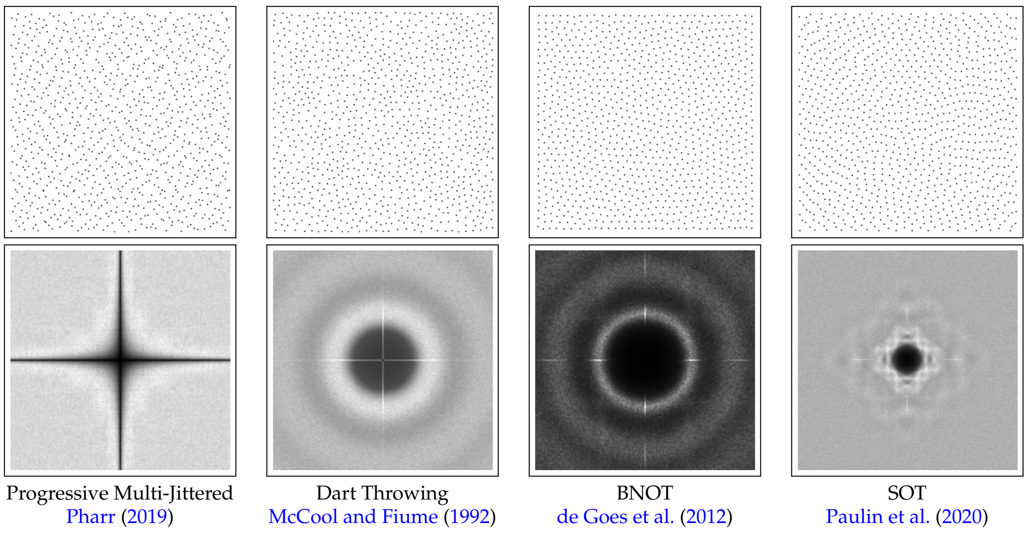


Figure 2.8: Visual comparison between a low-discrepancy sampler (Progressive Multi-Jittered Pharr (2019)) and three blue noise sampling methods: Dart Throwing (McCool and Fiume, 1992), BNOT (de Goes et al., 2012), and SOT (Paulin et al., 2020). For all four sample sets, the corresponding power spectra are shown in the second row. The low-discrepancy sampler exhibits a characteristic cross pattern in the power spectrum, indicating strong stratification along the two axes of the domain. In contrast, the blue noise samplers display a central void reflecting the absence of low-frequency components in all directions. This makes blue noise sampling isotropic in frequency space, whereas low-discrepancy samplers tend to introduce directional properties due to their axis-aligned structure.

Optimization-based methods, such as Wasserstein blue noise sampling (Qin et al., 2017), use optimization techniques to arrange samples. These methods aim to minimize an objective function that measures the uniformity and spacing of the samples, often resulting in even better blue noise properties. For example, Wasserstein blue noise sampling (Qin et al., 2017) optimizes the Wasserstein distance to achieve a uniformly distributed point set.

The interest in blue noise sampling for Monte Carlo integration lies in its ability to produce sample sets that are sparsely distributed without clustering. This results in more accurate and reliable integral estimates, as it avoids overestimation of parts of the function with clusters of samples that would act as double or triple counting of similar parts of the function. It also avoids underestimation of other parts of the function where too few samples would be used. Overall, the even sampling of the domain tends to maximize the information gathered by each sample.

The limitations of blue noise sampling methods often arise when scaling to higher dimensions. In high-dimensional spaces, the concept of distance becomes less meaningful as points tend to be far apart from each other, naturally creating voids in the sample distribution. The high-frequency components of blue noise are more significant in low dimensions, making these methods particularly suitable for many geometric applications in computer graphics, where lower-dimensional domains are common. In these settings,

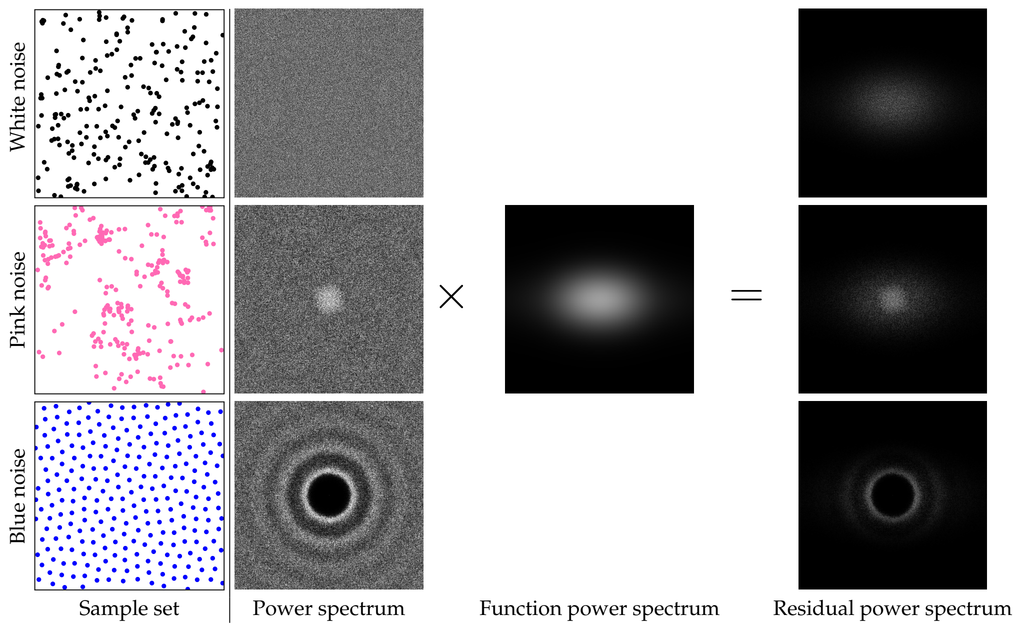


Figure 2.9: Visualization of the spectral properties of three sampling distributions and the product of their power spectra with a low-frequency function. The residual power spectrum shows a lower overlap between the sampling power spectrum and the function for the blue noise sample set, which can be explained by the absence of low frequencies. The result of a low residual power spectrum is a lower integration error in a Monte Carlo integration.

blue noise sampling excels in providing uniform coverage and reducing clustering.

Similar to low-discrepancy sequences, blue noise sampling can also be understood through a mathematical point of view to explain its effectiveness in reducing integration error. Formally, Monte Carlo integration error can be analyzed in the Fourier domain, where it is expressed as the integral of the product between the power Fourier transforms of the sampling pattern and the integrated function. This implies that the integration error is minimized when the spectral content of the sampling distribution and the integrand do not significantly overlap.

In practice, most rendering-related functions contain predominantly low-frequency components. Therefore, using a sampling pattern that suppresses low frequencies naturally reduces this spectral overlap, leading to lower error. Figure 2.9 illustrates this principle by showing the spectral product between a fixed low-frequency function and three sampling patterns with different spectral characteristics. Among them, the blue noise sampling exhibits the lowest residual power spectrum, and consequently, the smallest integration error.

Another early link between blue noise and computer graphics comes from the work of Cook (1986), who demonstrated the utility of blue noise distributions in addressing aliasing artifacts. Their insight was grounded not only in signal processing but also in human visual perception: blue noise patterns align well with the spatial frequency sensitivity of the human eye. Specifically, the eye's photoreceptor distribution and contrast sensitivity function are less responsive to high-frequency components, which are dominant in blue noise. As a result, blue noise patterns tend to mask visual artifacts more effectively than other sampling strategies, making them particularly well-suited for perceptually optimized sampling in rendering and anti-aliasing.

2.3 Physically Based Rendering

Physically based light transport simulation stands as a central application of Monte Carlo integration in computer graphics. The primary objective of this method is to produce physically accurate images by rendering a virtual scene, a process that simulates the complex interactions between light and scene surfaces, as captured by a virtual camera.

Rendering a scene in this manner is particularly complex because of the behavior of light particles. When light interacts with a surface, it can scatter in multiple directions, reflecting, refracting, or being absorbed depending on the material properties. Each light particle can potentially bounce numerous times before either being absorbed by a surface or reaching the camera sensor. This high degree of complexity, with light bouncing in unpredictable ways, necessitates an enormous number of light particles to be simulated for an accurate representation of the light flow.

Simulating every single light particle in such a scenario is computationally infeasible due to the particularly high number of interactions required. To manage this complexity, we employ Monte Carlo method to estimate light quantities. Monte Carlo methods, by leveraging random sampling and statistical averaging, provide a practical solution to approximating these interactions efficiently. This approach enables us to achieve high-quality, realistic images without the need to simulate every possible light interaction exhaustively.

2.3.1 Light And Surface Interaction

At the heart of physically based rendering lies the rendering equation, an integral equation that models the interaction of light with surfaces. Formulated by Kajiya (1986), the rendering equation encapsulates all possible light paths contributing to the color perceived at any point on a surface. The rendering equation is expressed as:

$$\mathrm{L}(x,\omega_o) = \mathrm{L_e}(x,\omega_o) + \int_\Omega f_r(x,\omega_i,\omega_o)\cdot \mathrm{L_i}(x,\omega_i)\cdot cos(\theta_i)\mathrm{d}\omega_i \tag{2.24}$$

In this equation, $\mathrm{L}(x, w_o)$ represents the outgoing radiance at point x in direction w_o which is the light leaving the surface. $\mathrm{L_e}(x, w_o)$ is the emitted radiance at point x in direction w_o, accounting for any light emitted by the surface itself. The term $f_r(x, w_i, w_o)$ is the bidirectional reflectance distribution function (BRDF), describing how light is reflected at the surface, depending on the incoming direction w_i and the outgoing direction w_o. The incident radiance $\mathrm{L_i}(x, w_i)$ represents the light arriving at the surface from direction w_i. The term w_i refers to the incoming light direction, and Ω is the hemisphere above

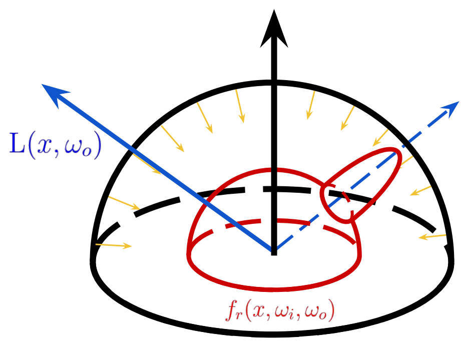


Figure 2.10: Visualization of the rendering equation. The black hemisphere represents integrating light arriving from all directions. The red hemisphere represents the Bidirectional Reflectance Distribution Function (BRDF) at a specific surface point. The blue arrow shows the outgoing radiance, and the black arrow indicates the surface normal.

point x over which the integral is computed. $cos(\theta_i)$ is the cosine between the surface normal at the point x and the direction w_i, influencing the amount of light contributing to the point. Figure 2.10 visualizes this equation at a given surface point x. It represents the different components of the rendering equation: the outgoing radiance in blue, the hemisphere integration in black, and the BRDF function in red.

Using this rendering equation 2.24, it is possible to compute the amount of light reaching a camera. For a given camera position and viewing direction, the equation allows us to calculate the amount of light arriving from the direction corresponding to the closest point on the surface in that direction. This assumes that light only scatters on surfaces. More complex formulations exist for the case of volumetric scattering, where light interacts with participating media such as fog or smoke, but we will not cover those cases here.

The rendering equation 2.24 is recursive because the incident radiance $\mathrm{L_i}(x, w_i)$ at any point x depends on the outgoing radiance from other points in the scene. This recursive nature means that solving the rendering equation at one point requires considering light contributions from potentially all other points in the scene. This interconnection is the essence of global illumination, which includes both direct and indirect lighting.

Direct lighting accounts for light traveling directly from a light source to a surface. In contrast, global illumination encompasses all light interactions, including multiple bounces, reflections, refractions, and scattering events. This comprehensive approach results in more realistic images, capturing subtleties like color bleeding, caustics, and soft shadows that simple direct lighting models do not account for. Figure 2.11 shows the Cornel box scene with a split of direct and indirect lightning on the left and the combination in a global illumination rendering on the right.

Monte Carlo integration enables the approximation of complex integrals in the rendering equation, making it possible to simulate global illumination and achieve photorealistic images. However, the stochastic nature of Monte Carlo methods introduces noise into the rendered images, manifesting as grainy artifacts that detract from visual quality.

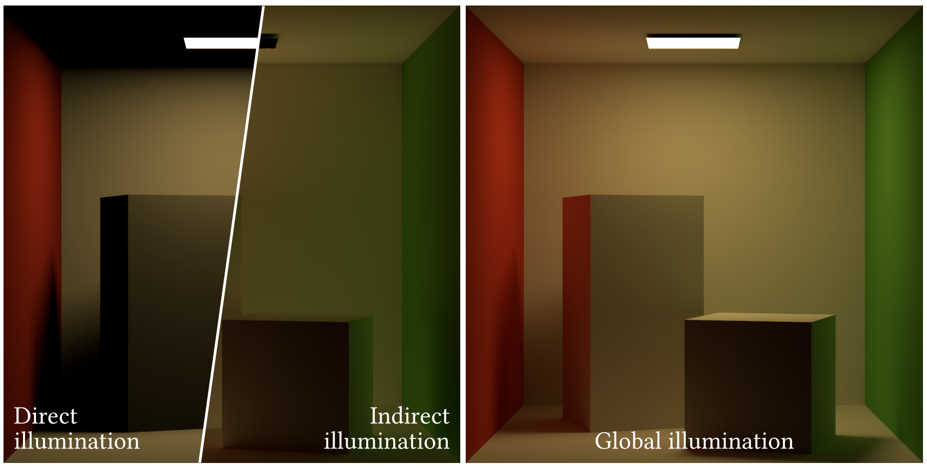


Figure 2.11: Comparison of direct and indirect illumination in a Cornell box scene. The left side illustrates the separation of direct lighting (left) and indirect lighting (right), highlighting the individual contributions. The right side shows the combined effect in a global illumination rendering.

Reducing this noise is crucial for producing high-quality images. One straightforward approach to mitigating noise is to increase the number of samples per pixel. As shown in fig. 2.12, the same scene rendered with progressively higher sample counts exhibits noticeably improved visual quality, with noise diminishing as the sample budget increases. While increasing the number of samples is effective, it also comes at the cost of longer rendering times, highlighting the trade-off between noise reduction and computational efficiency.

2.3.2 Path Space Rendering Formulation

The path space rendering formulation (Veach, 1997) offers a powerful alternative to the traditional rendering equation 2.24, reformulating the problem in terms of light paths rather than radiance at surface points. This approach simplifies the formulation of the rendering integration task in advanced rendering algorithm for global illumination and complex light effects.

The traditional rendering equation involves integrating over the hemisphere above a surface point, which can be challenging due to the recursive nature of light interactions. By reformulating the problem in terms of light paths, we can directly account for multiple scattering events. This reformulation allows for more flexible sampling strategies when using Monte Carlo methods.

Path Integral Formulation In the path space formulation, we consider all possible paths that light can take from the light sources to the camera. A light path is a sequence of vertices $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k$, where $\mathbf{x}_0$ is a point on a light source, and $\mathbf{x}_k$ is a point on the camera sensor. The contribution of a path to the final image is determined by the product of emission, scattering, and geometry terms along the path. Figure 2.13 illustrates a path with 5 vertices and displays the corresponding BRDF function on vertices along the path. The amount of radiance reaching the camera from all possible paths is defined as

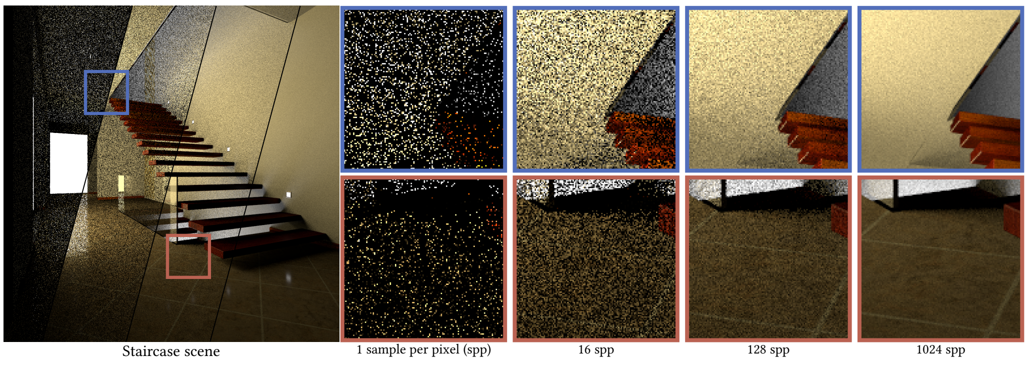


Figure 2.12: Comparison of rendered images of the same scene at different sample counts per pixel (1, 16, 128 and 1024 samples per pixel), demonstrating the effect of increasing the sample budget on visual quality. The images show a clear reduction in noise as the number of samples increases, leading to smoother, more photorealistic results.

following:

$$I = \sum_{k=2}^{\infty} \int_{\mathcal{P}} L_e(\mathbf{x}_0 \to \mathbf{x}_1) \left(\prod_{i=1}^{k-1} f_r(\mathbf{x}_i, \mathbf{x}_{i-1} \to \mathbf{x}_i, \mathbf{x}_i \to \mathbf{x}_{i+1}) G(\mathbf{x}_i, \mathbf{x}_{i+1}) \right) L_s(\mathbf{x}_k \to \mathbf{x}_{k-1})\, d\mu(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k) \tag{2.25}$$

In this equation, I represents the final image intensity. The set $\mathcal{P}$ denotes the space of all possible light paths. The term $L_e(\mathbf{x}_0 \to \mathbf{x}_1)$ is the emitted radiance from the light source at $\mathbf{x}_0$ in the direction of $\mathbf{x}_1$. The bidirectional reflectance distribution function (BRDF) $f_r(\mathbf{x}_i, \mathbf{x}_{i-1} \to \mathbf{x}_i, \mathbf{x}_i \to \mathbf{x}_{i+1})$ describes how light is reflected at the surface at $\mathbf{x}_i$, depending on the incoming direction from $\mathbf{x}_{i-1}$ and the outgoing direction towards $\mathbf{x}_{i+1}$. The term $L_s(\mathbf{x}_k \to \mathbf{x}_{k-1})$ represents the camera response function from the direction of $\mathbf{x}_{k-1}$ and $G(\mathbf{x}_i, \mathbf{x}_{i+1})$ a geometric term. The measure over the path space is denoted by $d\mu(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k)$.

Probability Measures And Computation In the path space formulation, the integral is computed using Monte Carlo estimator by sampling paths according to a probability distribution $p(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k)$. The contribution of each sampled path is weighted by the PDF of the path:

$$I \approx \frac{1}{N} \sum_{i=1}^{N} \frac{L_e(\mathbf{x}_0^i \to \mathbf{x}_1^i) \left(\prod_{j=1}^{k_i-1} f_r(\mathbf{x}_j^i, \mathbf{x}_{j-1}^i \to \mathbf{x}_j^i, \mathbf{x}_j^i \to \mathbf{x}_{j+1}^i) G(\mathbf{x}_j^i, \mathbf{x}_{j+1}^i) \right) L_s(\mathbf{x}_{k_i}^i \to \mathbf{x}_{k_i-1}^i)}{p(\mathbf{x}_0^i, \mathbf{x}_1^i, \dots, \mathbf{x}_{k_i}^i)} \tag{2.26}$$

where N is the number of sampled paths, and $\mathbf{x}_0^i, \mathbf{x}_1^i, \dots, \mathbf{x}_{k_i}^i$ are the vertices of the i-th sampled path. The probability measure $p(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k)$ must be designed to efficiently

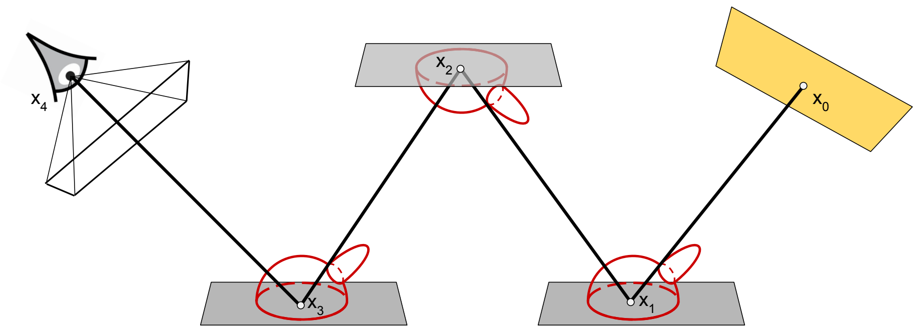


Figure 2.13: Visualization of the path space rendering formulation. The schema represent a single path composed of 5 vertices $x_0, \ldots, x_4$ connecting the light source to the camera. At each surface point the BRDF measurement is computed and the outgoing radiance is measured with the product of the different BRDF.

sample paths that contribute significantly to the final image.

Paths can be generated using two main approaches. One common method involves a random walk algorithm, where directions are randomly generated from a given point, and the nearest intersection point is found. This process is repeated from the new point, generating a path step-by-step. Another method constructs paths by directly connecting vertices sampled on surfaces and verifying visibility between them. These two approaches sample points in different manners and units. To combine them, the sampled points must be converted to a common unit, typically surface area measure. Once all vertices are expressed in the same unit, the path space PDF of the entire path is the product of the individual PDFs of each vertex:

$$p(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k) = p(\mathbf{x}_0) \cdot p(\mathbf{x}_1) \ldots p(\mathbf{x}_k) \tag{2.27}$$

By leveraging the path space formulation, we can devise more efficient Monte Carlo integration strategies by efficiently combining the two approaches for vertex sampling, enabling the computation of complex global illumination effects with greater accuracy and efficiency.

Change Of Unit In eqs. (2.25) and (2.26) a geometry term is introduced. This term $G(\mathbf{x}_i, \mathbf{x}_{i+1}) = V(\mathbf{x}_i, \mathbf{x}_{i+1}) \cdot \frac{\cos(\theta_{\mathbf{x}_{i+1}})}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2}$ accounts for the visibility $V(\mathbf{x}_i, \mathbf{x}_{i+1})$ between the vertices $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, and the Jacobian of the change of density from solid angle to surface area, $\frac{\cos(\theta_{\mathbf{x}_{i+1}})}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2}$.

Figure 2.14 propose a visualization of the Jacobian between solid angle and surface area unit. A point $\mathbf{y}$ sampled on the red surface can be re-expressed in solid angle units by first considering the area reduction between the red surface, with normal $\theta_{\mathbf{y}}$, and a plane normal to the direction $\mathbf{y} \rightarrow \mathbf{x}$. This reduction factor is given by $\cos(\theta_{\mathbf{y}})$. Next, the projection of this blue line onto the green line on the hemisphere results in a surface reduction by a factor of $\frac{1}{(\|\mathbf{x}-\mathbf{y}\|^2)}$. The square compensates for the dimensions of the 2D surfaces even if they are represented in 1D in the diagram.

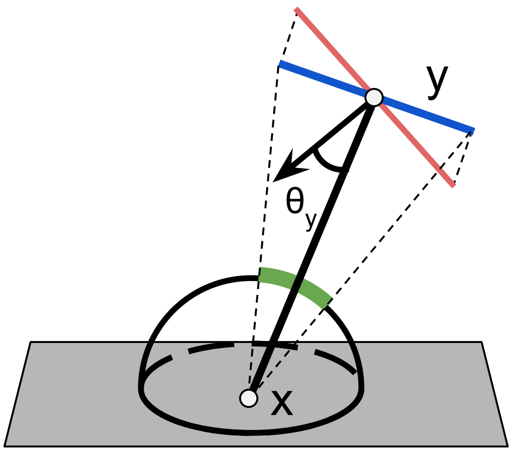


Figure 2.14: Illustration of the Jacobian transformation from surface area to solid angle in the rendering equation. The surface point $\mathbf{y}$ is mapped to solid angle as seen from point $\mathbf{x}$. The visualization highlights how the projected area diminishes due to the cosine foreshortening angle $\theta_{\mathbf{y}}$ (transition from red to blue), and how the contribution further decreases with increasing distance between $\mathbf{x}$ and $\mathbf{y}$ (transition from blue to green).

2.3.3 Rendering Algorithms

Rendering algorithms are methods used to produce light paths for evaluating the light reaching a given pixel. These algorithms fundamentally differ in their approach to generating paths, directly influencing the probability distribution of the paths they create. Algorithms that can generate contributing paths with a higher frequency are more efficient, but this efficiency is highly dependent on the specific scene being rendered. Different scenes may require different path generation strategies to achieve optimal performance.

From a practical standpoint, rendering algorithms transform uniformly distributed random numbers in $[0, 1]^d$, generated by a computer's random number generator, into complex light paths. The manner in which these random numbers are utilized determines the efficiency of the light path generation process.

Direct Lighting Algorithm The direct lighting algorithm is a method used in rendering to estimate the direct component of the rendering (only one bounce in the scene). This algorithm begins at the camera, sending a ray from the camera center through the chosen pixel that is being estimated. The ray travels into the scene until it intersects with a surface. At the intersection point, the rendering equation 2.24 is evaluated with a specific simplification: the recursive nature of the rendering equation is omitted. In direct lighting, only the incoming radiance from a light source is considered, meaning that indirect lighting is not accounted for (light that bounces off one or more surfaces before reaching the point of interest).

To estimate the incoming radiance at the surface intersection, two main methods can be employed. The first method involves sampling the hemisphere of directions around the intersection point, with the random direction occasionally coinciding with the direction

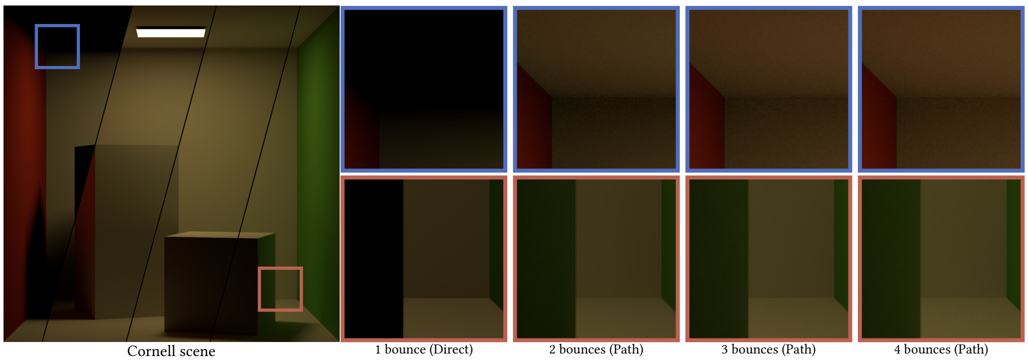


Figure 2.15: Visualization of rendering with path tracing at various bounce counts. A significant difference occurs between a single bounce and two bounces, but the change is minimal between three and four bounces. This emphasizes the significance of the initial bounces in the final image and the reduced impact of later bounces.

of a light source. This approach relies on the probability of randomly selecting the direction towards a light source, which can be inefficient and result in high variance. The second, generally more effective method is to sample a point on a random light source in the scene and create a direct connection between this point on the light source and the point of interest on the surface. This technique is known as next event estimation. This second approach significantly often reduces variance in the Monte Carlo process by ensuring that the sampled directions are more likely to result in contributing path at the intersection point.

Path Tracing Algorithm Path tracing (Kajiya, 1986) is considered the state-of-the-art method for simple global illumination rendering, encompassing both direct and indirect lighting. The algorithm starts by sending a ray from the camera through the pixel being estimated and tracing it until it intersects with the first surface in the scene. At this intersection point, direct lighting is computed similarly to the direct lighting method described previously. Up to this point, no indirect lighting is taken into account.

The key difference between path tracing and direct lighting is that path tracing continues to generate paths beyond the first intersection point. After computing the direct lighting at the first point, a random direction is sampled, and a new ray is cast from this point in the chosen direction. The ray is traced to a new intersection point, where the direct lighting is again computed. This process of sampling a direction, finding the new intersection point, and computing the direct lighting is repeated multiple times.

At each surface interaction point, the incoming radiance can, in theory, be estimated by sampling multiple directions and averaging their contributions—this approach is known as *distributed path tracing* or *distribution path tracing*. In contrast, standard path tracing limits the estimation to a single randomly sampled direction per interaction, resulting in just one path being extended at each bounce.

While sampling multiple directions (distributed path tracing) generally reduces variance and produces more stable results, it comes with a significant computational cost as the number of rays grows exponentially with each additional bounce. The motivation behind using standard path tracing that estimates incoming indirect light from a unique direction lies in a key observation: the first few bounces in a light path contribute the most to the overall image luminance. Due to energy loss at each interaction, subsequent bounces have diminishing impact on the final pixel value. This is illustrated in fig. 2.15, where the visual difference between single-bounce (direct lighting) and two-bounce rendering is substantial, while further bounces result in increasingly subtle changes that often fall below perceptual thresholds. This effect is further reinforced by the non-linear nature of human visual perception: we perceive luminance approximately on a logarithmic scale. As a result, even measurable increases in light energy from later bounces may not produce noticeable visual differences, making the computational cost of sampling them less important for the overall looking of the image.

As a result, standard path tracing offers a more efficient allocation of computational resources by prioritizing early bounces, where error reduction is most impactful. This trade-off is visualized in fig. 2.16, highlighting the contrast between generating multiple new directions per bounce in distributed path tracing and generating multiple independent paths in standard path tracing.

Path tracing is particularly effective for simple global illumination scenarios, such as outdoor scenes, and is especially suitable when there are no complex specular reflection or refraction effects present in the scene. Additionally, path tracing is well-suited for parallel computing architectures like GPUs, which can accelerate the computation of paths and improve overall rendering performance. This makes path tracing a powerful and efficient method for producing high-quality renderings with global illumination.

Other Rendering Algorithms While path tracing is highly effective for many global illumination scenarios, it struggles with certain lighting effects such as glass refraction or finding contributing paths when only a few paths can actually contribute, such as light entering through the underside of a door or a keyhole. For these types of problems, other rendering algorithms have been developed with more specialized path construction methods. From the extensive list of rendering algorithms, it is worth mentioning a few notable ones.

Light tracing (Arvo et al., 1986) mimics path tracing but starts rays from the light source, attempting to connect to the camera at each intersection. This method can effectively handle some caustic effects that path tracing struggles with such as light reflected by a mirror.

Bidirectional path tracing (Lafortune, 1996) combines the strengths of both path tracing and light tracing. It generates paths from both the camera and the light sources and attempts to connect these paths. This combination can significantly improve rendering performance for simple caustic effects by capturing complex light interactions more effectively.

Photon mapping (Jensen, 1996) is another powerful technique, especially for complex caustics involving multiple refractions. In this method, photons are traced from the light sources and stored in a photon map. During rendering, this map is used to estimate radiance at surface points, providing efficient estimation of caustics involving multiple

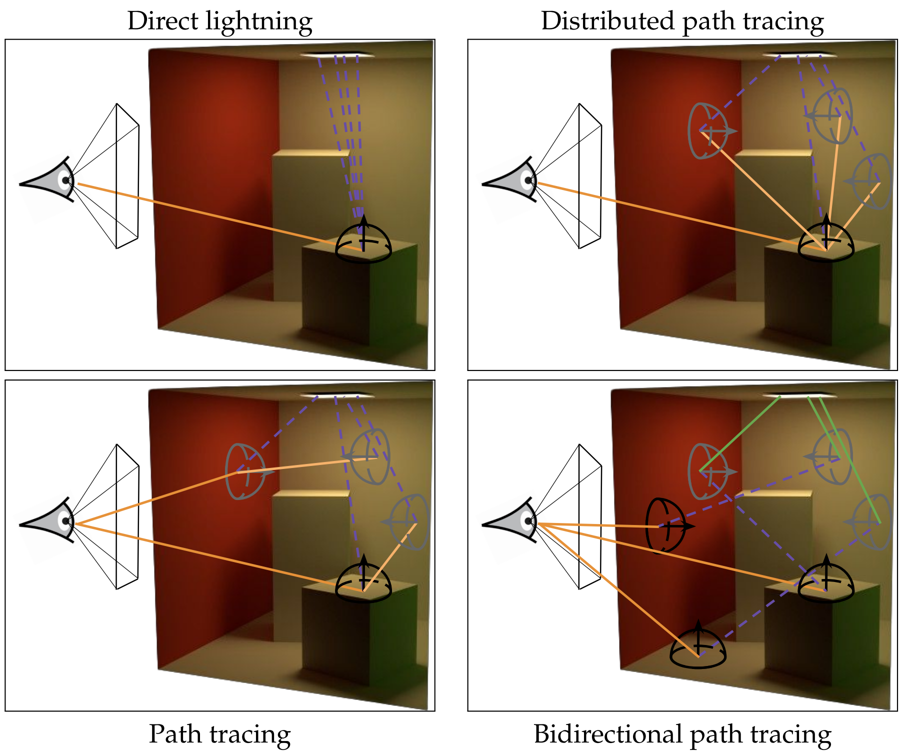


Figure 2.16: Visualization of the difference between Direct lightning, distributed Path tracing, Path tracing and Bidirectional path tracing. Paths constructed from the camera are shown in orange line, explicit connections between 2 vertices in dashed purple lines and paths generated starting from the light in green.

reflections and refraction.

Metropolis light transport (Veach and Guibas, 1997) is a more advanced algorithm that uses a Markov chain Monte Carlo approach to explore the path space. MLT is particularly useful in scenarios where only a few paths contribute significantly to the final image. It adaptively mutates paths, focusing on those that contribute the most to the overall illumination, thus improving the efficiency of the rendering process.

Each rendering algorithm has its own benefits, but no single algorithm provides the best performance in every scenario. More complex algorithms that can construct intricate paths are generally more computationally expensive. However, this extra cost can be offset by the significant error reduction they provide, particularly in scenes with complex lighting effects. The trade-off between performance and the quality of path construction is crucial and highly scene-dependent. Selecting the appropriate rendering algorithm involves balancing computational resources against the specific lighting challenges presented by the scene to achieve optimal rendering quality and efficiency.

Russian Roulette In Monte Carlo rendering algorithms, one challenge is dealing with the potentially infinite dimensional integrals due to the recursive nature of light paths. To transform this problem into a finite one, the Russian roulette technique is employed.

Russian roulette probabilistically terminates a path after each bounce based on a termination probability, thereby avoiding the computational overhead of tracing infinitely long paths.

Consider a path tracing scenario where, after each bounce, a random number $r \in [0, 1]$ is generated. If r exceeds a certain threshold q, the path is terminated. The threshold q is typically based on the accumulated path weight or radiance contribution. For instance, if $q = 0.2$, there is a 80% chance that the path will be terminated at any given bounce. If the path continues, the contribution of the path is scaled by the inverse of the survival probability $1 - q$ to ensure unbiased results.

Mathematically, if L is the radiance contribution of a path without Russian roulette, the expected value with Russian roulette is:

$$\hat{L} = \begin{cases} 0 & \text{if } r > q \\ \frac{L}{1-q} & \text{if } r \leq q \end{cases} \tag{2.28}$$

This ensures that the expected value $\mathbb{E}[\hat{L}]$ remains unbiased:

$$\mathbb{E}[\hat{L}] = (1 - q) \cdot \frac{L}{1 - q} = L \tag{2.29}$$

In rendering, the more bounces a light path undergoes, the more light is absorbed by materials, resulting in a lower final contribution of the path to the integrand. This phenomenon means that shorter paths generally have a more significant contribution than longer paths. Consequently, focusing computational resources on shorter paths can be beneficial, as they are more likely to contribute meaningfully to the final rendered image.

This method strikes a balance between the need to trace long light paths for accurate global illumination and the practical constraints of limited computation time. Russian roulette achieves this by probabilistically terminating light paths, allowing the renderer to avoid tracing infinitely recursive bounces. While this introduces some additional variance due to the stochastic nature of the termination, it preserves the unbiasedness of the estimator and provides an elegant solution to managing the recursive structure of the rendering equation.

2.3.4 Rendering Specific Variance Reduction Techniques

Monte Carlo rendering algorithms often suffer from high variance, leading to noisy images. To address this, various variance reduction techniques (Section 2.2.1) can be applied to improve the efficiency and accuracy of these algorithms. This section covers several key variance reduction tools and their application to rendering: Importance sampling, Multiple Importance Sampling and Correlated sampling. Many other variance reduction methods can be applied, but this stand as the most commune one.

Importance Sampling On Direction Sampling In the rendering equation 2.24, we need to evaluate the integration over the hemisphere. A straightforward approach would

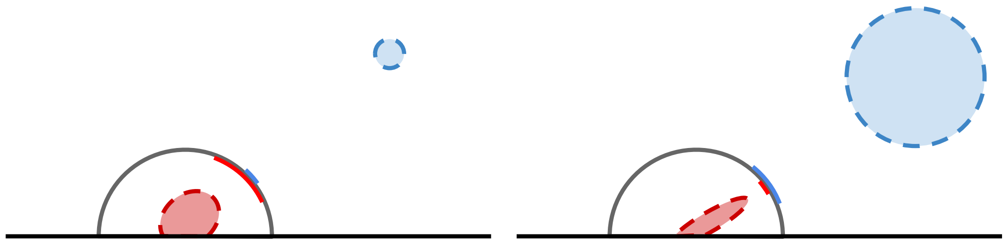

Diffuse surface and small light
Secular surface and large light

Figure 2.17: Visualization of two opposite illumination scenarios. On the left, the surface material is diffuse and illuminated by a small light source. In that case, the support of the BSDF is larger than the projected area of the light. On the right, the material is specular and illuminated by a large light source. In this case, the support of the BSDF is smaller than the projection of the light, meaning not all samples from the light would contribute to the final illumination.

be to sample directions uniformly over the hemisphere, but this is highly inefficient. The actual integral involves the product of three terms: the BRDF function, the incident luminance, and a cosine term. The incident luminance is a complex function and often unknown. However, the other two terms, the BRDF and the cosine term, are known elements.

Instead of sampling directions uniformly, it is possible to sample directions proportional to the product of these two known elements, or at least to the cosine term when the BRDF is too complex to handle directly. Even when the BRDF is complex, approximations can be used for importance sampling.

This applied importance sampling is crucial for reducing variance by focusing ray computation in directions that are likely to contribute significantly to the integral, rather than randomly sampling directions, many of which may not contribute at all. This is particularly important for specular surfaces, where only one or a small range of directions can contribute. Tracing rays in directions where the BRDF is low would result in a low contribution, regardless of the amount of incident luminance.

Multiple Importance Sampling For Next Event Estimation Multiple Importance Sampling (Section 2.2.2) is a powerful technique to reduce variance by combining different sampling strategies. One important use of MIS is to combine BRDF sampling and light sampling in next event estimation.

When estimating the contribution of direct lighting at a point $\mathbf{x}$, we can sample directions either according to the BRDF or the light source distribution. For BRDF sampling, we sample directions from the hemisphere oriented by the surface normal at $\mathbf{x}$ with importance sampling proportional to the BSDF component and evaluate the incoming radiance from the light source. For light sampling, we directly sample points on the light source and evaluate the BRDF for the direction from $\mathbf{x}$ to the sampled light point.

The key idea of MIS is to combine these two estimates using weights that reduce the

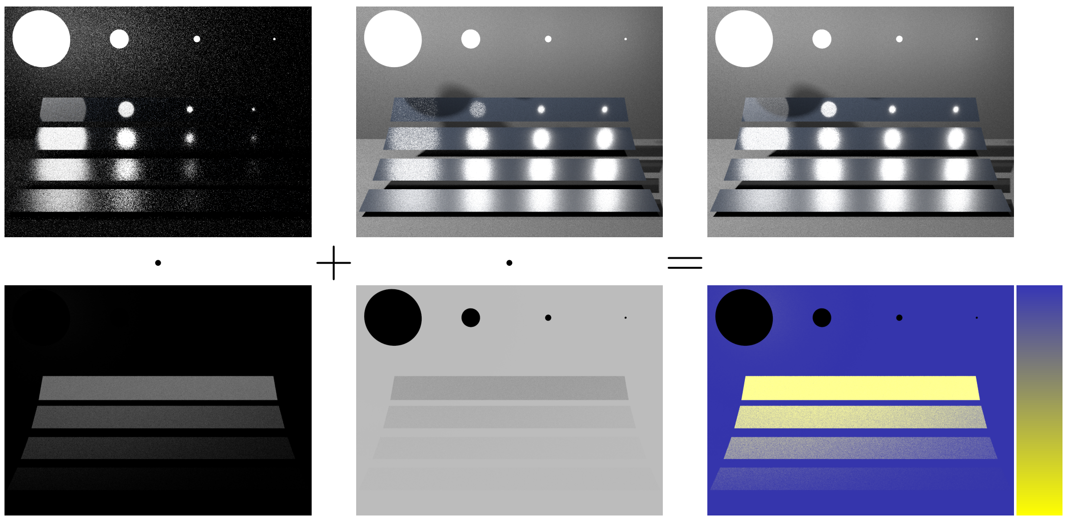


Figure 2.18: Visualization of how multiple importance sampling impacts next event estimation, with renderings using BRDF sampling, light sampling, and their MIS combination on top. The lower part shows sampling probabilities and MIS weight distribution, where yellow regions indicate higher importance of BSDF sampling, and blue regions indicate higher importance of light sampling.

variance of the combined estimator. Using MIS, the estimation of 2.24 becomes:

$$I \approx \sum_{i=1}^{N_{\text{BRDF}}} \frac{w_{\text{BRDF}}(\omega_i)}{p_{\text{BRDF}}(\omega_i)} L_i(\mathbf{x}, \omega_i) f_r(\mathbf{x}, \omega_i, \omega_o)(\mathbf{n} \cdot \omega_i) + \sum_{i=1}^{N_{\text{Light}}} \frac{w_{\text{Light}}(\omega_i)}{p_{\text{Light}}(\omega_i)} L_i(\mathbf{x}, \omega_i) f_r(\mathbf{x}, \omega_i, \omega_o)(\mathbf{n} \cdot \omega_i) \tag{2.30}$$

where $w_{\text{BRDF}}(\omega)$ and $w_{\text{Light}}(\omega)$ are the MIS weights for the two sampling techniques and $p_{\text{BRDF}(\omega)}$ and $p_{\text{Light}}(\omega)$ the PDF of the of the sampling strategies. This approach ensures that the estimator benefits from the strengths of both BRDF sampling and light sampling, reducing variance effectively.

The benefits of using MIS for BRDF and light sampling are visualized in fig. 2.17. These diagrams illustrate two different scenarios. On the left, we have a small light source and a diffuse BRDF. When projecting the importance distribution of each sampling method onto the hemisphere, we see a small blue area representing the light source and a large area for the BRDF projection. In this scenario, sampling directions based on the BRDF has a high probability of missing the light source, whereas sampling the light source always intersects the support of the BRDF, thus contributing to the integral. Therefore, sampling the light source is the most effective strategy. In the second scenario, the BRDF is sharp, indicating specular properties, and the light source is large. Here, the light source's projection covers a significantly larger area on the hemisphere than the BRDF's projection. Sampling the light source in this case often results in low or even zero values for the BRDF, whereas sampling directions based on the BRDF always intersect the light

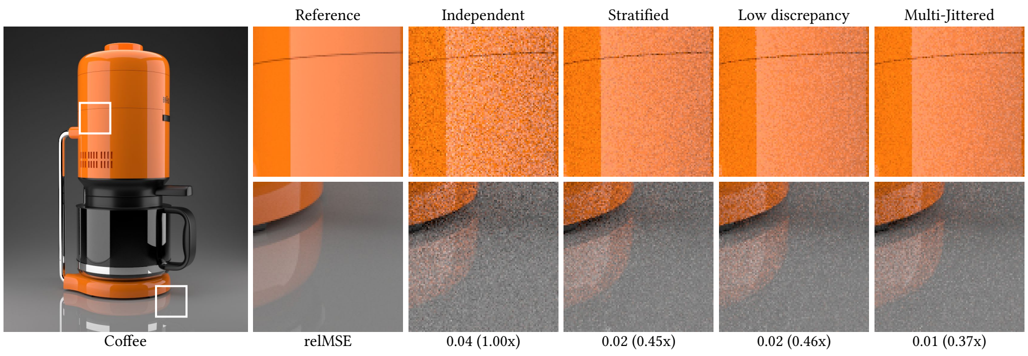


Figure 2.19: Visualization of rendering with different sampling methods. The figure compare Independent sampler, Stratified sampling, Low discrepancy sampling Kollig and Keller (2002) and Correlated Multi-Jittered Kensler (1967)

source. Thus, BRDF sampling is more effective in this scenario, yielding a significantly lower error.

Figure 2.18 demonstrates this effect in a simple rendering scene. The figure shows the renderings obtained with BRDF sampling and light sampling, along with the appropriate MIS weights computed using the balance heuristic (Veach and Guibas, 1995). The last column presents the MIS combination of both techniques. By combining the two sampling strategies, the result has low error where each method performs well and avoids noisy regions where one method might fail while the other succeeds. The false color visualization highlights the contribution of each sampling method, with BRDF sampling shown in yellow and light sampling in blue. In particular BRDF sampling shines on specular surfaces and light sampling in diffuse area.

Low Discrepancy Sampling Low discrepancy sampling (Section 2.2.3), also known as quasi-random sampling, is an effective technique to reduce variance in Monte Carlo rendering by generating samples that are not independently sampled but offer a more even distribution compared to purely random samples. This even distribution leads to better coverage of the integration domain, which in turn improves the accuracy and stability of the rendered results.

In rendering, low-discrepancy sequences such as Halton, Sobol, and Hammersley sequences are commonly used for their simple construction properties. When using Monte Carlo integration to estimate the rendering equation 2.24, low discrepancy sequences can be employed to sample the directions ω_i. For instance, in path tracing, each bounce of a light path requires sampling a new direction according to the BSDF. By using low discrepancy sequences for these directions, we achieve a more uniform coverage of the hemisphere, which reduces variance in the path tracing estimator. This results in fewer samples being required to achieve a given level of image quality compared to random sampling.

At the path level, using low discrepancy samples for sampling direction also improves the uniform coverage of the path space. It can also be beneficial for next even estimation to ensure that all lights are sampled frequently reducing the risk of possible under-estimation of some light contribution at low sample count. Single stratification can achieve this, but a low discrepancy sampler also allows for stratification inside each light in addition.

The effectiveness of low discrepancy sampling is particularly evident with smooth and continuous integrals where the uniformity of sample distribution significantly impacts the convergence rate. In such cases, low discrepancy sequences can outperform random sampling with improved convergence rate of the integration error. Figure 2.19 shows an example of rendering with different sampling methods. Low discrepancy and Correlated Multi-Jittered achieve a significant error reduction compared to the Independent sampler. In a simple rendering scenario, stratified sampling can also be sufficient but does not offer good scaling with the number of dimensions of rendering.

2.3.5 Quality Metrics

Evaluating the quality of rendered images produced by Monte Carlo methods is crucial for assessing and improving rendering algorithms. Various metrics are used to measure the quality, primarily focusing on per-pixel error metrics and perceptual metrics. These metrics help in understanding the effectiveness of variance reduction techniques and the perceptual impact of rendering errors.

Per-pixel error metrics quantify the difference between the rendered image and a reference image. These metrics provide a direct measure of the error introduced by the Monte Carlo integration. The most common metric is the Mean Squared Error (MSE). Given a rendered image I and a reference image R, both of size $W \times H$, the MSE is defined as:

$$\text{MSE} = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} (I_{ij} - R_{ij})^2 \tag{2.31}$$

MSE measures the average squared difference between the rendered and reference images. A lower MSE indicates a closer match to the reference image, implying higher rendering quality.

Relative Mean Squared Error (RMSE) is a normalized version of MSE that accounts for the varying intensity levels in the image. It is defined as:

$$\text{RMSE} = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} \frac{(I_{ij} - R_{ij})^2}{(R_{ij})^2 + \epsilon} \tag{2.32}$$

where ϵ is a small constant to avoid division by zero. RMSE provides a relative measure of error, making it more robust to variations in image brightness. This aligns better with the characteristics of the human visual system, which does not perceive luminance linearly but more closely follows a logarithmic response. As a result, expressing error as a ratio with respect to the reference value better correlates with perceived visual differences—though it remains an imperfect approximation. Moreover, this relative formulation prevents the metric from being dominated by errors in high-luminance regions, providing a more balanced evaluation across the full dynamic range of the image.

Another important per-pixel metric is the variance, which measures the spread of the pixel values around the mean. In the context of Monte Carlo rendering, variance quantifies the noise in the pixel values due to the stochastic nature of the sampling process. For a pixel I_{ij}, the variance $\text{Var}(I_{ij})$ is defined as:

$$\text{Var}(I_{ij}) = \mathbb{E}[(I_{ij} - \mathbb{E}[I_{ij}])^2] \tag{2.33}$$

In practice, the sample variance can be computed as:

$$\text{Var}(I_{ij}) = \frac{1}{N-1}\sum_{k=1}^{N}(I_{ij}^k - \bar{I}_{ij})^2 \tag{2.34}$$

where N is the number of samples, I_{ij}^k is the k-th sample value for pixel (i, j), and $\bar{I}_{ij}$ is the mean value of the samples. Lower variance indicates less noise and higher quality in the rendered image. Reducing variance in the Monte Carlo estimator decreases the pixel-wise error, thereby lowering MSE and RMSE. Therefore, these metrics are essential for evaluating the effectiveness of variance reduction techniques in rendering.

2.4 Gradient Estimation For Machine Learning Optimization

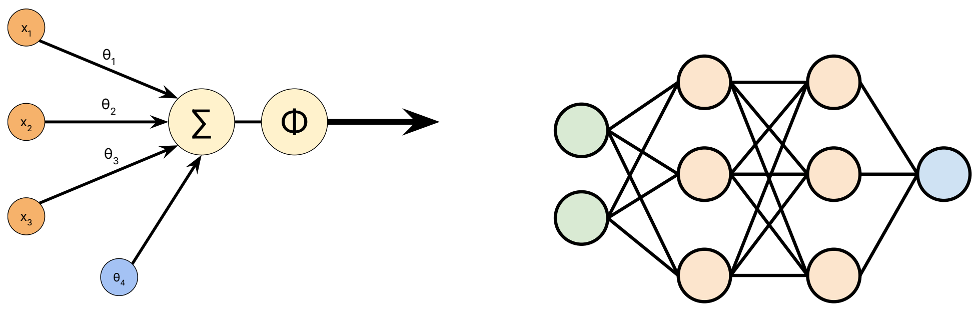
Machine learning is another computer graphic application that relies on Monte Carlo integration. It involves optimizing a model function, typically a neural network, using example data to minimize an objective function. This is generally accomplished through iterative optimization due to the complexity of the task and the nature of available data. The iterative process, commonly known as stochastic gradient descent (SGD), includes a Monte Carlo estimation of the partial derivatives of the model function with respect to a set of parameters. This estimation is inherently noisy, which can reduce the speed of convergence and potentially hinder the ability to reach the optimal solution. Therefore, reducing the noise in the Monte Carlo estimation is crucial for efficient and effective optimization.

2.4.1 Supervised Optimization

In supervised machine learning, the primary objective is to learn a mapping from inputs to outputs using a model function m, parameterized by a set of coefficients θ. The model function can be expressed as:

$$m(x, \theta) = y \tag{2.35}$$

where x represents the input data, y is the output generated by the model, and θ denotes the parameters of the model.



Perceptron Visualization MLP Visualization

Figure 2.20: Illustration of a perceptron (left) and a multi-layer perceptron (right).

A common choice for the model function in machine learning is a neural network. Neural networks are a class of functions that consist of layers of interconnected neurons. Each neuron applies a linear transformation to its inputs followed by a non-linear activation function. These networks can approximate complex functions, adapt through learning, and handle high-dimensional data.

Perceptron The perceptron is the simplest type of neural network and serves as the building block for more complex architectures. A single perceptron can be defined as:

$$\text{Perceptron}(x) = \Phi\left(\sum_{k=1}^{K} \theta_k \cdot x_k + b\right) \tag{2.36}$$

where θ_k represents the weights, b is the bias term, x is the inputs, and Φ is the activation function. Common activation functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU). Figure 2.20 left side represent a perceptron with 3 inputs x_1, x_2, x_3, associated parameters θ_k, and bias b and the activation function.

Multi-layer perceptrons (MLP) extend the concept of a single perceptron by stacking multiple layers of neurons, allowing the network to model more complex functions. Each layer is composed of several perceptrons, and the output of one layer serves as the input to the next.

Neural Network Architectures Building on the concept of Multi-Layer Perceptrons, various neural network architectures have been developed to address different types of data and tasks more effectively.

Fully Connected Networks (FCN) are neural networks where every neuron in one layer is connected to every neuron in the next layer. These networks are versatile and can be used for various tasks, but they can be computationally expensive and prone to overfitting.

Convolutional Neural Networks (CNN) are designed to process grid-like data such as images. They use convolutional layers to automatically and adaptively learn spatial hierarchies of features, making them particularly effective for image and video recognition tasks.

U-Net is a type of convolutional network particularly effective for image segmentation tasks. It features an encoder-decoder architecture that concatenate feature maps from the encoder to the corresponding decoder layers, allowing for precise localization and segmentation of images.

Transformers are primarily used for sequential data and natural language processing tasks. They utilize self-attention mechanisms to weigh the influence of different parts of the input data, enabling them to capture long-range dependencies and relationships within the data. Recently, their usage to image related task has show potential when attention information inside a same image can be beneficial.

Loss Function The performance of the model is measured using a loss function $\mathcal{L}$, which quantifies the difference between the predicted output $m(x;\theta)$ and the target value y. For a given input-target pair (x, y), the loss can be expressed as:

$$\mathcal{L}(m(x;\theta), y) \tag{2.37}$$

Different types of loss functions are used depending on the task. The L2 Loss (Mean Squared Error) is commonly used for regression tasks and measures the average squared difference between the predicted and actual values. The Cross-Entropy Loss is used for classification tasks and measures the difference between the predicted probability distribution and the actual distribution. The Perceptual Loss is often used in image processing tasks and compares high-level features extracted from a pre-trained neural network rather than pixel-wise differences.

Regularization terms can be added to the loss function to penalize certain properties of the model parameters, such as their magnitude, to ensure smoothness and prevent overfitting. The generalized loss function is:

$$\mathcal{L}(m(x;\theta), y) + \lambda R(\theta) \tag{2.38}$$

where λ is a regularization parameter, and $R(\theta)$ is the regularization term. A classical regularization term is the norm of the model parameters θ. It ensure that the parameters remain all small avoiding high frequencies in the function $m(x;\theta)$.

Dataset The dataset Ω consists of input-target pairs (x, y) and is crucial for defining the loss function and training the model. The task is to find the optimal set of parameters θ^* that minimize the loss function over the dataset:

$$\theta^* = \arg\min_{\theta} \int_{\Omega} \mathcal{L}(m(x;\theta), y) \mathrm{d}(x, y) \tag{2.39}$$

The objective is to adjust the parameters θ so that the model function m produces outputs as close as possible to the target values y for the given inputs x.

A key requirement for effective machine learning is that the training dataset accurately reflects the distribution of data encountered during inference. Nevertheless, in many practical scenarios, the dataset employed to define the optimization objective (eq. (2.39)) may not fully capture the characteristics of real-world data due to the inherent limitations in obtaining exhaustive datasets. The ability of a learned model to perform well on unseen real-world data is termed generalization, and it is a critical attribute. To promote robust generalization, the training dataset Ω should be as statistically similar as possible to the true distribution of the real-world data.

2.4.2 Gradient-Based Optimization

Finding the optimal set of parameters for the model function is a complex task. This complexity arises from the non-linearity of the model function, especially in the case of neural networks. Due to this non-linearity, no direct method can provide a solution to the problem eq. (2.39). Consequently, solving this task generally requires running an optimization process, wherein the parameters are iteratively refined to reduce the loss over the dataset.

The classical approach to this iterative refinement is known as gradient descent. In gradient descent, the parameters are updated in the direction that reduces the loss function. This is achieved by computing the derivative of the loss function with respect to the model parameters and adjusting the parameters in the opposite direction of this derivative. Mathematically, this can be expressed as:

$$\theta \leftarrow \theta - \eta \int_{\Omega} \nabla_{\theta} \mathcal{L}(m(x;\theta), y) \, \mathrm{d}(x, y) \tag{2.40}$$

where η is the learning rate scaling down the gradient, and $\int_{\Omega} \nabla_{\theta} \mathcal{L}(m(x;\theta), y) \, \mathrm{d}(x, y)$ is the gradient of the loss function with respect to the parameters. In practice, however, applying gradient descent is not feasible. The gradient is an integral over the derivatives of the loss with respect to each example in the dataset (x, y). Computing this gradient exactly is intractable, even for finite datasets, because the number of data points can be extremely large. Furthermore, the optimization process may require computing this gradient millions of times, making the exact computation impractical.

Instead, most machine learning optimizations rely on stochastic gradient descent (SGD). The key difference between gradient descent and SGD is that, in SGD, the integral of the gradient over the data samples is approximated using Monte Carlo integration. Specifically, a subset (or mini-batch) of the dataset is used to compute an approximate gradient:

$$\nabla L_{\theta} = \int_{\Omega} \nabla_{\theta} \mathcal{L}(m(x;\theta), y) \, \mathrm{d}(x, y) \approx \frac{1}{B} \sum_{j=1}^{B} \nabla_{\theta} \mathcal{L}(m(x_j;\theta), y_j) \tag{2.41}$$

where B is the number of examples in the mini-batch, which is typically much smaller than the dataset size.

This approximation is significantly cheaper to compute and allows the optimization process to proceed efficiently. However, the Monte Carlo estimation introduces error due to its stochastic nature. This integration error can reduce the convergence speed and potentially affect the quality of the resulting set of parameters. Despite these drawbacks, SGD and its variants are widely used as they provide a practical means of optimizing complex model functions in machine learning.

Gradient Computation And Backpropagation The gradient computation in neural networks, particularly the backpropagation algorithm, is central to efficiently optimizing these models. Backpropagation uses the chain rule to compute the gradient of the loss function with respect to each parameter in the network. For a neural network with layers $L_1, L_2, \ldots, L_n$, the gradient of the loss with respect to the parameters in a given layer is influenced by the gradients from subsequent layers.

Consider a simple neural network model $m(x, \theta) = \text{Perceptron}_1(\text{Perceptron}_0(x))$, composed of two perceptron layers. The parameters of each perceptron (including the bias) are denoted by θ_L, where L indicates the layer index.

The derivative of the loss $\mathcal{L}$ with respect to a parameter in the last layer can be computed by applying the chain rule, separating the derivative of the loss with respect to the network output and the derivative of the output with respect to the parameters of the last layer:

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{\partial \mathcal{L}}{\partial m(x, \theta)} \cdot \frac{\partial m(x, \theta)}{\partial \theta_1} \tag{2.42}$$

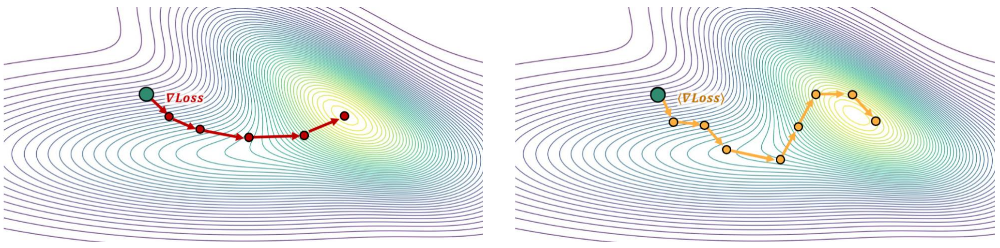
This gradient computation can then be propagated backward to earlier layers, reusing parts of the intermediate results:

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{\partial \mathcal{L}}{\partial m(x, \theta)} \cdot \frac{\partial m(x, \theta)}{\partial \text{Perceptron}_1} \cdot \frac{\partial \text{Perceptron}_1}{\partial \theta_0} \tag{2.43}$$

The power of the chain rule lies in its ability to decompose the derivative of a complex function into a product of derivatives of simpler sub-functions. This structure enables efficient gradient computation through backpropagation, where intermediate derivatives are reused across layers, drastically reducing computational overhead.

In summary, backpropagation iteratively applies the chain rule from the output layer to the input layer, propagating the error gradients backward through the network. This process ensures that the gradients are efficiently computed, allowing for effective parameter updates.

In practice the chain rule and the different derivatives involved can be computed on the fly using automatic differentiation (auto-diff) libraries, such as TensorFlow and PyTorch. These libraries build computational graphs dynamically and apply backpropagation to compute gradients, simplifying the implementation of complex neural networks.



Gradient descent Stochastic gradient descent

Figure 2.21: Visualization of the convergence process over a loss landscape. The path illustrates the iterative updates of SGD navigating through local minima and saddle points.

Limits Of Stochastic Gradient Descent While stochastic gradient descent is a powerful optimization method, it has several limitations. One key issue is its convergence rate compared to pure gradient descent. Pure gradient descent uses the full dataset to compute the gradient, ensuring a more direct path to the minimum. This is illustrated in fig. 2.21 with a straighter path for pure gradient descent compared to an SGD, reaching the minimum in fewer steps. However, it is computationally expensive for large datasets. In contrast, SGD approximates the gradient using mini-batches, which introduces noise into the gradient estimates. This noise can slow down convergence, leading to a zigzagging path rather than a smooth descent.

Overfitting is a practical concern in SGD, particularly when the dataset is a subset of a more general dataset. Overfitting occurs when the model learns to perform well on the training data but fails to generalize to new, unseen data. This issue is exacerbated by the noise introduced in SGD, which can cause the model to capture spurious patterns in the training data.

The batch size also significantly impacts SGD's performance. Smaller batch sizes provide more frequent updates but with noisier gradients, which can hinder convergence. Larger batch sizes yield more accurate gradient estimates but reduce the number of updates per epoch, potentially slowing down learning. The trade-off between fast gradient estimation with a small batch size and a larger batch size for more accurate estimation can also be adapted during optimisation. At the start of learning, a smaller batch size is often sufficient than when the model is already close to minima and noise has a greater impact.

The learning rate is another critical factor. A high learning rate can cause the optimization to overshoot the minimum, while a low learning rate can result in slow convergence and getting stuck in local minima. Finding an optimal learning rate is challenging and often requires extensive tuning.

The shape and number of model parameters further complicate the optimization process. High-dimensional parameter spaces can exacerbate the difficulties in navigating the loss landscape, making it easier to get trapped in local minima or saddle points. On the other hand, a reduced number of parameters can lead to sub-optimal convergence with a compromise in terms of quality.

Finally, the approximation inherent in SGD means that it often cannot reach the true minimum of the loss function. Instead, SGD tends to oscillate around a minimum due to the stochastic nature of the gradient estimates. This limitation can prevent the model from achieving the best possible performance.

2.4.3 Variance Reduction In Gradient Estimation

As mentioned earlier, the Monte Carlo estimation introduces noise in the gradient, which can adversely affect the optimization process by increasing the loss due to this noise. This reduction in convergence speed can be mathematically expressed for convex tasks. Therefore, reducing the estimation error and variance is crucial for achieving faster and more robust optimization. While methods used for variance reduction in Monte Carlo rendering do not all apply directly to machine learning optimization due to differences in problem domains and technical challenges, several techniques have been proposed to enhance stochastic gradient descent (SGD).

One classical approach is the moment-based estimation of the gradient. By reusing previous gradient estimates, it is possible to smooth the gradient over multiple iterations using an exponential moving average. This method, requiring minimal memory as only the previous gradient needs to be stored between iterations, can be expressed as follows:

$$g_t = \beta g_{t-1} + (1-\gamma)\nabla L_\theta \tag{2.44}$$

where g_t is the moving average of the gradient at iteration t, and γ is a smoothing parameter typically close to 1. This method is improved in algorithms like Adam Kingma and Ba (2014), which adjusts the learning rate based on the moments of the gradient, thereby accelerating or decelerating the optimization:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L_\theta \tag{2.45}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla L_\theta)^2 \tag{2.46}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{2.47}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.48}$$

$$\theta \leftarrow \theta - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.49}$$

In these equations, m_t and v_t are the first and second moments of the gradient, β_1 and β_2 are parameters controlling the decay rates, and ϵ is a small constant for numerical stability. This method can be linked to Temporal Anti-Aliasing in rendering, which reuses previous temporal information to reduce noise in consecutive frames. Another variance reduction tool is importance sampling (Section 2.2.2). By choosing samples based on an optimized probability distribution, it is possible to reduce gradient noise. The challenge with this approach is that the optimal importance sampling distribution is not fixed for an optimization task; it varies depending on the state of the model function during optimization. Therefore, dynamically adapting the sampling distribution is crucial but challenging.

Finally, data stratification (Section 2.2.3) can also reduce noise. This method involves selecting data samples for a mini-batch that are as different as possible. For example, in a classification task with 10 equally represented class, it is possible to construct a mini-batch with 1 data from each class at each training step. This ensures some uniformity of the data in each mini-batch. The difficulty lies in the high dimensionality of the samples and the challenge of measuring data differences effectively. Moreover, different data samples may not always correlate with significantly different gradients, which is the desired property to reduce estimation noise through stratification.

2.4.4 Special Properties Of Monte Carlo Gradient Estimation

Monte Carlo gradient estimation for machine learning optimization presents unique challenges due to the nature of input data and the requirements of the optimization process. One key specificity is that input data are generally very high-dimensional and exhibit a highly non-uniform distribution. For example, images have a dimension as large as the number of pixels, and the set of natural-looking images represents a tiny subset of the image space. This makes the optimization of stratification in a mini-batch particularly difficult, as effectively sampling diverse and representative data is challenging.

Another fundamental limitation is the simultaneous estimation of Monte Carlo estimators for each partial derivative of the model function. In SGD, all these estimations are performed using the same data samples in the mini-batch. This joint estimation complicates the application of importance sampling. The difficulty arises because the gradient with respect to one parameter can differ significantly from the gradient with respect to another. Consequently, a good importance sampling distribution for one parameter may be a poor choice for another, potentially increasing the overall estimation error. Thus, a well-designed importance sampling strategy must achieve a trade-off in terms of quality across all parameters.

Moreover, gradient estimation is performed numerous times during optimization, making overhead and memory footprint crucial factors. Even if a method reduces estimation error, its impact on execution time and memory usage must be considered. Some methods developed for other Monte Carlo integration tasks, such as light transport simulation, do not perform well in the context of machine learning due to these technical limitations. For example, the control variate technique is difficult to apply to gradient estimation because it requires storing a control variate function for each trained parameter, possibly millions to billions of them. This creates an unsuitable memory footprint, and using a single control variate may not be sufficiently well-suited for each parameter, potentially increasing the estimation error.

In summary, the use case of machine learning optimization is closely linked with performance. Variance reduction techniques cannot come at the cost of slower execution or significant memory usage. The high dimensionality of input data, the need for joint estimation of gradients, and the practical constraints of frequent gradient estimation all contribute to the complexities of applying Monte Carlo methods in this context.

Chapter 3
Related work overview

Monte Carlo methods have been extensively studied and applied across a wide range of fields, notably in physically-based rendering and machine learning. In rendering, these methods are foundational for solving light transport integrals and simulating complex global illumination phenomena. To improve the efficiency and reduce the variance of Monte Carlo estimators, a number of advanced techniques have been developed. These include control variates, which exploit correlated auxiliary functions to reduce estimator variance, as well as optimized sampling that correlate the samples to better estimate the integrand. Recent research has also explored multi-class sampling, which aims to generate sample sets satisfying multiple distributional constraints simultaneously, a challenge that arises in rendering and in other domains requiring multi-objective sample design.

In machine learning, Monte Carlo methods are also crucial, particularly in the context of stochastic optimization techniques such as Stochastic Gradient Descent (SGD). Here, reducing the variance of gradient estimates can significantly improve training stability and convergence speed. Techniques such as importance sampling and adaptive data weighting have been proposed to prioritize training examples that are more informative, thereby enhancing the quality of the Monte Carlo gradient estimator.

In this section, we review the most relevant and recent contributions in Monte Carlo-based methods for both rendering and machine learning, with a focus on control variate techniques, advanced sampling optimization, and their applications to high-dimensional learning tasks.

3.1 Control Variate Estimator For Rendering

Control variate estimators are variance reduction techniques that improve Monte Carlo integration by incorporating a correlated auxiliary function with a known or cheaply estimated integral. By modeling the target function and subtracting this approximation, the estimator reduces the variability of the residual, leading to more accurate results.

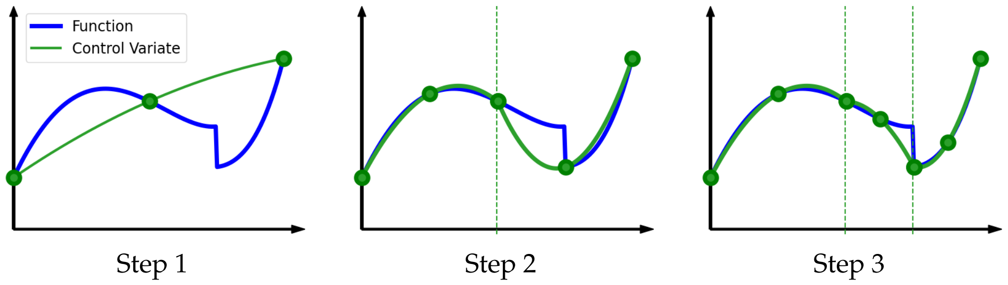


Figure 3.1: Illustration of the adaptive subdivision method from Crespo et al. (2021). The process begins with a polynomial quadrature over the entire domain (Step 1). If additional computational resources are available, the domain is divided into two equal subdomains, and quadrature is performed on each (Step 2). This iterative subdivision continues, targeting regions with higher variance to improve accuracy by refining those areas (Step 3). In this example, the right subdomain is further subdivided because the initial quadrature fit was less accurate than in the left subdomain. The final result is a piecewise polynomial function that seamlessly combines lower-order polynomials across all subdomains, effectively covering the entire domain.

The effectiveness of this method depends on the quality of the approximation and its correlation with the target function. In this section, we explore its practical use in rendering. For a theoretical introduction and derivation, see section 2.2.2.

Quadrature Based Control Variate Crespo et al. (2021) introduced a novel method for adaptively constructing a control variate function for Monte Carlo estimators, specifically in the context of rendering. Selecting an optimal control variate a priori is particularly challenging due to the complex nature of rendering functions, which result from the interplay of many local sub-functions. Instead of relying on predefined control variates, the authors proposed using a quadrature-based approximation of the integrand in the primary sample space as the control variate function. This quadrature is computed either at each rendering pixel or by using shared information between multiple pixels. This approach requires relatively few samples to construct an adaptive control variate, making it efficient for rendering tasks.

A key innovation of this method is its combination of quadrature with an adaptive domain subdivision. The domain is partitioned into non-overlapping regions $s \in \Omega$, each expected to have a similar integration error. For each region, a low-order polynomial quadrature is performed, which is favored due to its simplicity and low computational cost. The final Monte Carlo estimator is constructed by combining the control variate estimates from each subdivided region.

$$\langle I_{Crespo} \rangle_N = \sum_{s \in \Omega} \frac{|s|}{|\Omega|} \langle I_{Crespo} \rangle^s_{M_s}, \quad \sum_{s \in \Omega} M_s = N \tag{3.1}$$

More formally, the method begins with a single integration domain, where an initial quadrature is performed using a small, fixed number of samples. If the available quadrature budget permits further refinement, the domain is split into two subregions along a selected axis, and a new quadrature is computed within each subregion. This process continues iteratively: as long as the sampling budget allows, the subregion with the highest estimated error is selected for further subdivision and additional quadrature evaluation. An overview of this adaptive subdivision strategy is illustrated in fig. 3.1.

This adaptive subdivision ensures that the regions are allocated an appropriate number of samples based on their expected error, instead of uniformly sampling the entire domain. The adaptive sampling not only ensures that all regions contain samples for estimating the control variate, but it also focuses the sampling budget on higher-error regions, thereby reducing overall error and variance.

The main strength of this approach lies in the adaptive nature of both the control variate and the domain subdivision. By targeting regions with higher expected error, the method significantly reduces variance compared to traditional Monte Carlo integration. In rendering, using low-order polynomials as control variates is effective because rendering functions are generally smooth. Most function discontinuities in rendering result from shadow occlusion or geometric edges. The domain subdivision helps account for these sharp transitions or discontinuities, improving the overall robustness of the estimator compared to a single polynomial.

However, the method does have some limitations. The accuracy of the quadrature-based control variate depends on how well the quadrature captures the features of the integrand. In cases where the function has features not aligned with the fixed sampling points of the quadrature, the method may underestimate the error within certain regions. This can lead to under-sampling those regions, resulting in higher errors in the final estimator. In extreme cases, this can even cause the method to perform worse than classical Monte Carlo estimators, where no such control variates are used. Despite this potential drawback, the method represents a significant advancement in the use of adaptive control variates for rendering applications, particularly when the function being integrated exhibits smooth behavior.

Multiple Importance Sampling As Control Variate Kondapaneni et al. (2019) proposed an optimal weighting scheme for Multiple Importance Sampling, as described in eq. (6.16). This weighting strategy can be interpreted as an adaptive and optimal control variate, where the weighted combination of multiple proposal densities serves as the control variate function. Under this perspective, their method can be seen as a form of optimal control variate. However, unlike classical control variates, it is not applicable in arbitrary sampling spaces; its use is restricted to mixtures of predefined proposal distributions.

It is also important to note that although this approach fits within the control variate framework, it remains fundamentally a MIS weighting technique. As such, it operates independently of variance reduction methods applied in the primary sample space. Therefore, primary sample space techniques can still be employed in conjunction with this method, as they address variance from an orthogonal perspective.

Neural Based Control Variate Müller et al. (2020) introduced the first adaptive control variate method based on neural regression of a scene's light field. Their approach trains a neural network on-the-fly during rendering, using previously traced rays to learn an approximation of the scene's light field. This learned representation is then employed as a control variate to reduce variance in Monte Carlo integration.

One of the key strengths of this method is its adaptivity: by leveraging data from previously rendered rays, the neural network can progressively refine its approximation of the light field over time. An additional insight from the paper is the use of light field regression instead of full path space regression. This choice significantly reduces the dimensionality of the learning problem and makes the representation independent of the camera position, allowing it to remain valid even as the camera moves. A significant limitation is that while camera motion is feasible, scene animation yields poor results because the network must adapt between frames.

To ensure the control variate is analytically integrable, the authors propose modeling the light field as a normalized probability distribution, scaled by a learnable factor. By design, this construction enables the integral of the control variate to be computed analytically as the scaling factor.

The primary contribution of this method lies in its ability to learn high-quality control variates from previous frames, enabling provably unbiased rendering using neural predictions. However, the approach also faces challenges: the training and inference of the neural network at runtime introduce significant computational overhead, and achieving a high-quality approximation may require extensive training time.

3.2 Multi-Class sampling

Multi-class sampling refers to the process of generating a set of samples within a domain such that the samples collectively satisfy multiple, potentially conflicting, distributional objectives. Unlike traditional sampling where all points are drawn to fit a single target distribution, in multi-class sampling each point may be influenced by several distinct distributions or optimization criteria. In practice, this typically results in the point set being composed of overlapping subsets, where each subset follows a specific target distribution. Crucially, some points may belong to multiple subsets, thereby being required to simultaneously satisfy multiple objectives. This overlapping introduces a fundamental challenge: points must balance the constraints imposed by the different distributions they are associated with. As a result, the design of such sample sets becomes a complex multi-objective problem, where trade-offs must be managed to ensure that all distributions are adequately represented.

Solving the multi-class sampling problem typically involves formulating it as an optimization task, where the objective is to adjust the position of samples such that they collectively approximate multiple target distributions. Among the various strategies explored in this context, optimal transport (OT) theory has proven particularly powerful. In this vein, the state-of-the-art approach for sample set optimization based on optimal transport has been proposed by Paulin et al. (2020). Their method leverages the concept of sliced optimal transport to iteratively refine a point set so that it approximates a desired distribution.

The key idea is based on the Radon transform and consists in projecting the sample set and the target distribution onto randomly sampled one-dimensional directions. For each projection, the 1D Wasserstein distance is computed, effectively measuring the discrepancy between the projected sample distribution and the target. This formulation enables the analytic computation of gradients with respect to the sample positions, making it possible to perform stochastic gradient descent to iteratively update the point set.

One of the main advantages of this method lies in its scalability: the use of 1D projections significantly reduces the computational cost of optimal transport, making it suitable for high-dimensional spaces. Moreover, the approach is flexible, accommodating arbitrary sampling domains and non-uniform target densities. Still this idea is limited to single class optimization.

Multi-Class Wasserstein Barycentric Optimization Wasserstein blue noise sampling (Qin et al., 2017) utilizes Optimal Transport theory to generate point distributions that are both spatially uniform and visually irregular—hallmarks of blue noise. The central idea is to minimize the Wasserstein distance between a continuous target density and the empirical distribution defined by the sample points. Technically, the method employs iterative Bregman projections combined with Newton-based optimization to progressively shift the samples toward the target density while maintaining even spatial coverage through a relaxation process.

For multi-class sampling, the method naturally extends through the use of regularized Wasserstein barycenters. Instead of aligning to a single target density, the algorithm considers multiple target densities—one for each class—and computes a joint sample distribution that minimizes the aggregated Wasserstein distance to all class-specific densities. This approach ensures that each class maintains blue noise characteristics individually, while their union also preserves blue noise properties. By unifying the treatment of multiple classes within a single optimization framework, the method avoids common artifacts and interference seen in simpler multi-class strategies.

3.3 Perceptual Error Distribution In Rendering

When viewing rendered images, humans do not focus on individual pixels; rather, the eye processes groups of pixels together, filtering the signal from the screen. This perceptual characteristic allows us to "hide" certain rendering errors by distributing them in a way that is less noticeable to the human eye Weier et al. (2017). A technique inspired by image dithering (Roberts, 1962) achieves this by pushing rendering errors into the high-frequency components of the image. By maximizing intensity differences between adjacent pixels in a single frame, the resulting images appear smoother to human viewers, even if the overall pixel-wise error remains the same or increases. This technique creates a perceptually pleasing error distribution, minimizing the visible impact of the error.

Blue-Noise Dithered Sampling: The first practical method to achieve perceptually pleasing error distribution was introduced by Georgiev and Fajardo (2016). This approach optimizes the distribution of Monte Carlo samples over the image plane to maximize the spatial distance between samples in neighboring pixels. By ensuring that nearby pixels are sampled with significantly different sample sets, the rendering results exhibit greater variation between adjacent pixels, effectively pushing the error into the higher frequencies of the image. The method relies on a dithering mask, optimized to shift the common sample set uniquely for each pixel during rendering.

The optimization target for this dithering mask is defined through an energy function, representing the weighted sum of shift distances between pixels:

$$E = \sum_{p} \sum_{q \neq p} e^{-\frac{\|p_i - q_i\|^2}{\sigma_i^2}} \cdot e^{-\frac{\|p_s - q_s\|^{d/2}}{\sigma_s^2}} \tag{3.2}$$

In this equation, p and q represent two pixels. The term $\|p_i - q_i\|$ denotes their distance in screen space, while $\|p_s - q_s\|$ measures the distance between their associated samples in the high-dimensional sample space. To account for the dimensionality d of the sample space, the sample-space distance is scaled by an exponent of $d/2$. The parameters σ_i and σ_s control the spread of the Gaussian functions in screen space and sample space, respectively. Minimizing this equation tends to result in negative correlation of samples from neighboring pixels. This means that adjacent pixels tend to have samples that are widely spaced from each other, thereby sampling different regions of the functions in the context of Monte Carlo integration. Typically, σ_i is set to 2.1 and σ_s to 1. These values were initially proposed by the paper, but have become standard in the literature due to their empirically proven effectiveness.

A simulated annealing algorithm is used to minimize this energy. The choice of simulated annealing ensures that the initial sample distribution remains intact, while only the spatial positioning of the shifts is altered. Unlike gradient-based methods, this approach preserves the uniformity of the shift distribution, maintaining the unbiased nature of the sampling while optimizing for perceptually better error distribution.

Although the method efficiently achieves good sample distribution properties with relatively low computational overhead, high-quality convergence can be challenging due to the inherent limitations of simulated annealing. When applied to rendering, a single sequence of samples is generated, and each pixel's samples are shifted using the optimized dithering mask before being passed to the rendering engine. This results in images with smoother perceptual error, which are more pleasing to the human eye.

Despite its groundbreaking contribution, several limitations were identified with this method. First, it struggles to scale effectively with the increasing dimensionality of the rendering integration problem, a phenomenon known as the curse of dimensionality. Additionally, the method performs significantly better at lower sample counts, as the shift mapping tends to degrade distribution when a sample requires a replacement if the shift leaves the sampling domain.

Nevertheless, this method was a pioneering approach to perceptually guided error distribution in rendering. Although it has limitations, particularly in scaling and high sample counts, it remains superior to purely random samplers in most cases. At its best, it improves the perceptual quality of rendered images, and at worst, it matches the performance of traditional random sampling approaches.

Sample Optimization Through Surrogate Rendering: Building upon the previous method, Heitz et al. (2019) introduced significant improvements to overcome key limitations. Instead of relying solely on sample distances, this method focuses directly on minimizing rendering error, thereby reducing the problem's dimensionality since rendering is treated as a single-dimensional output. Given that the rendering function is often unknown, they proposed using proxy functions, such as Heaviside step functions, which can be randomly generated and serve as approximate surrogates for the actual rendering function.

A second notable advancement in this method is the shift from optimizing a dithering mask to optimizing the distribution of low-discrepancy sequences, such as Sobol sequences. This guarantees that each pixel receives a high-quality sample set, ensuring low error in the rendering even if the perceptual error distribution is not fully reduced. The modified energy term reflects this focus on rendering error rather than spatial sample distance:

$$E = \sum_{p} \sum_{q \neq p} e^{-\frac{\|p_i - q_i\|^2}{\sigma_i^2}} \cdot D(p, q) \tag{3.3}$$

Similarly to eq. (3.2), p and q denote two pixels, and $\|p_i - q_i\|$ represents their distance in screen space. The key distinction from the previous approach lies in the substitution of the sample-space distance with a rendering discrepancy term $D(p, q)$. Instead of comparing high-dimensional sample positions directly, $D(p, q)$ measures the rendering difference between the two pixels using a large set of randomized Heaviside functions. As in the previous method, simulated annealing is used to optimize the screen-space positions of the sequences, minimizing the energy function across the image plane.

This method demonstrates superior performance compared to the approach from Georgiev and Fajardo (2016), attributable both to the use of higher-quality Sobol sequences (with XOR scrambling) for each pixel and to the more direct optimization based on rendering error, which better correlates with the actual rendering process. The previous method's reliance on sample distance was not the most efficient metric for this type of optimization, while using rendering error provides a more accurate proxy for guiding the sampling process.

Additionally, this approach scales better with the sample count because it directly accounts for the entire sample set in the rendering error metric. At runtime, the method is simple to implement, requiring only a scrambling key per pixel to modify a Sobol sequence, which makes it both practical and efficient.

However, similar to the earlier approach, the convergence speed of simulated annealing remains a bottleneck. While the method quickly reaches a good state, achieving very high-quality results requires a longer time due to the nature of the simulated annealing process. This approach was later extended with Rank-1 sequences to improve per-pixel quality and optimized further, particularly in terms of parallelizing the optimization process, as demonstrated by Belcour and Heitz (2021).

Reorganizing Low Discrepancy Sampling Across Screen: Ahmed and Wonka (2020) proposed a deterministic method for reducing integration error by reorganizing a single

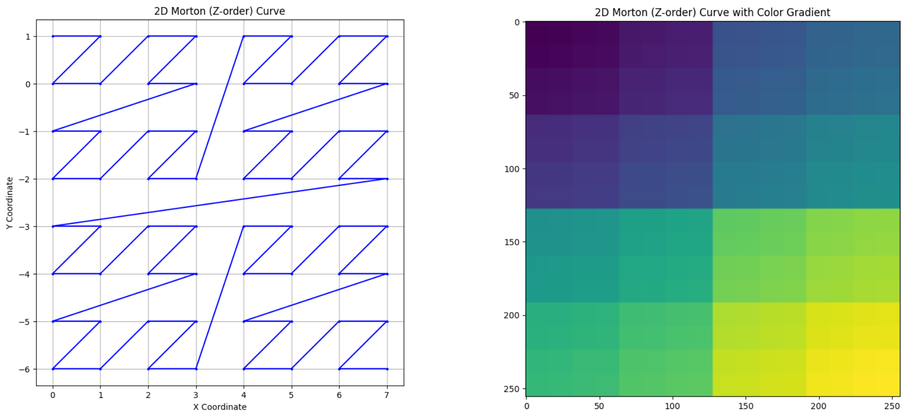


Figure 3.2: Visualization of the Z-ordering curve and the pixel ordering in color gradient. Neighboring pixels tend to use close index following the ordering.

low-discrepancy sequence—specifically a Sobol sequence—along a Z-curve in screen space. Leveraging the symmetric properties of Sobol sequences, where a sequence of length 2^n can be recursively split into two low-discrepancy subsequences, they efficiently partitioned the sequence into multiple high-quality subsequences. Each pixel is then assigned one of these subsequences according to its position along the Z-curve, leading to improved screen-space error distribution with high-frequency properties.

Figure 3.2 illustrates the Z-ordering curve and its distribution of indices across a 2D plane. This indexing, combined with multiple low-discrepancy subsequences, ensures that not only each individual pixel receives a low-discrepancy sequence, but also that aggregated local areas exhibit low-discrepancy properties.

A key advantage of this technique is its practicality: unlike previous methods, it does not require any precomputation or costly optimization. However, a notable limitation is its inability to incorporate sophisticated perceptual models, as the assignment of subsequences is based solely on a filling curve over screen space.

3.4 Importance And Adaptive Sampling For Gradient Estimation

Monte Carlo methods have found significant applications in the context of Stochastic Gradient Descent (SGD), where the gradient of the loss function is estimated using randomly selected mini-batches of training data. The accuracy of these gradient estimates plays a critical role in the efficiency and stability of the optimization process. To reduce the variance of the gradient estimate and thus improve convergence, various strategies have been developed. One prominent approach is importance sampling, which aims to construct mini-batches by sampling data points with some non-uniform distribution to focus on data with higher contribution to the gradient. Importance sampling can rely on precomputed metrics, but adaptive schemes that dynamically update the sampling distribution during training have been shown to provide more effective improvements. Another strategy, closely related to importance sampling, is adaptive data weighting, where individual gradients within a mini-batch are scaled according to heuristics designed to prioritize samples that lead to faster progress toward the optimal model parameters. Both approaches leverage Monte Carlo principles to enhance gradient estimation.

Adaptive Mini-Batch Importance Sampling Importance sampling in the context of machine learning aims to construct mini-batches by sampling training data with probabilities proportional to the magnitude of their corresponding gradients. However, computing the exact gradient norm for each data point is typically infeasible, as it would require a full backward pass per sample, incurring a prohibitive computational cost. To address this, several methods have been proposed to approximate the gradient norm efficiently. One of the simplest yet effective heuristics is to use the loss value of a data point as a proxy for its gradient norm. The underlying intuition is that a high loss generally corresponds to a large gradient, while low-loss examples tend to produce small gradients. Although this approximation does not perfectly capture the gradient norm, it has been shown to sufficiently improve the sampling distribution, resulting in faster convergence with minimal additional computational overhead compared to more sophisticated techniques.

Katharopoulos and Fleuret (2018) presented an adaptive importance sampling strategy for constructing training mini-batches. Given that the optimal importance distribution is proportional to the gradient norm, they proposed an approximation based on an upper bound derived from the pre-activation outputs of the final network layer. Specifically, the defined importance metric is the gradient of the loss function with respect to these pre-activations, demonstrating a provable relationship with the gradient norm. A significant aspect of this methodology is the dynamic nature of the importance metric, which evolves during training in response to weight updates, unlike static importance sampling techniques.

Complementing this, they used a resampling procedure. This involves an initial draw of a large mini-batch, followed by a forward pass to compute the importance metric for each instance. A smaller training mini-batch is then subsampled proportionally to the computed relative importances, effectively eliminating the need for persistent storage of importance values.

The integration of this adaptive importance sampling and the resampling mechanism enables a more accurate estimation of the stochastic gradient, resulting in accelerated training of neural networks.

Adaptive Mini-Batch Weighting The work of Santiago et al. (2021) introduces a method to accelerate the training of neural networks by adaptively weighting data gradient within each mini-batch to maximize the learning. The core idea is to identify, within a mini-batch, the samples that contribute most significantly to the learning process, measured by the magnitude of their gradient norms, and to scale their gradients to emphasize their influence during parameter updates. The following equation defines the weighting strategy:

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix} \qquad \nabla_{\theta_t} = \begin{bmatrix} \|\nabla_{\theta_t} L(m(x_1, \theta_t), y_1)\|^2 \\ \vdots \\ \|\nabla_{\theta_t} L(m(x_M, \theta_t), y_M)\|^2 \end{bmatrix}, \tag{3.4}$$

$$\arg\max_{\mathbf{w}} \quad \mathbf{w}^T \nabla_{\theta_t} - \lambda \left\| \mathbf{w} - 1 \right\|^2, \qquad \text{with } w_j \geq 0 \quad \text{and} \quad \sum_{j=1}^{M} w_j = M \tag{3.5}$$

The definition of optimal weighting includes a regularization term scaled by the parameters λ that avoids the trivial solution giving a weight of 1 to the sample with the largest gradient norm and a weight of zero for all other data.

This adaptive weighting ensures that, although the average scaling factor in the batch remains 1 as $\sum_{j=1}^{M} w_j = M$, samples with greater learning potential are prioritized effectively increasing the expected magnitude of the gradient and thus accelerating training.

However, this approach introduces a biased gradient estimate, as it breaks the uniform assumption of sampling. Additionally, because mini-batches are typically constructed randomly, there is no guarantee that any given mini-batch contains high-value samples. As a result, while the method is effective in the early stages of training, where large gradient are more likely, it becomes less suitable for late-stage optimization, where bias in the gradient can lead to suboptimal convergence. Despite these limitations, the method provides valuable insights into dynamic gradient scaling and could be further enhanced through importance sampling, which targets efficiency gains from a complementary perspective.

Chapter 4
Regression For Improved Monte Carlo Estimation

The traditional interpretation of Monte Carlo integration leans heavily on the concept of statistical estimation. It views the integral as the expected value of a random variable. To estimate this value, the method draws random samples from the integration domain and then calculates the average of the integrand evaluated at these samples. This interpretation boasts simplicity and broad applicability. It focuses solely on the values of the integrand at the randomly chosen samples within the integration domain, disregarding both the specific positions of the samples and the shape of the domain itself. This approach makes Monte Carlo integration a general-purpose tool that can be applied to a wide range of integrals without requiring any specific knowledge about the integrand properties.

However, an alternative calculus interpretation of Monte Carlo integration exists, one that frames it as a reconstruction task using a constant function. This interpretation draws inspiration from the mean value theorem for definite integrals. This theorem guarantees the existence of a constant value within the integration domain whose function value equals the integral's average value. This constant function integrates to the same solution as the entire integral. Interestingly, the theorem also implies that this constant's value can be approximated by averaging random samples of the integrand, leading to an estimator identical to the one produced by the traditional statistical approach.

This work explores the possibility of extending the interpretation of calculus beyond the use of a constant function. This exploration is only feasible within the framework of the calculus interpretation. The statistical interpretation intrinsically ties the concept of expected value to the mean of the sample values. Here, we propose a novel class of Monte Carlo estimators that utilizes exactly the same input, randomly generated samples, without making any further assumptions. Instead of simply averaging the samples to construct a constant function, we propose a more sophisticated approach. We leverage the power of least-squares regression to fit a non-constant model function (e.g., a polynomial) to these samples. By strategically selecting the model function basis, we can then compute the analytical integral of the regressed function. This analytical integral serves as the foundation for constructing a more powerful estimator compared to the traditional approach. Interestingly, this approach reveals a clear connection to control variate estimators, a well-established technique commonly employed to reduce variance in Monte Carlo simulations. We demonstrate that our regression step effectively generates control variates and can reduce integration error even when using simple control variate functions. While the use of regression within control variates isn't entirely new, we provide the first in-depth exploration of the tight coupling between regression, control variates, and Monte Carlo integration. Notably, our framework highlights that traditional Monte Carlo integration is a special case where the regression solution is simply a constant function.

Our work offers significant advantages. The approach demonstrably minimizes an upper bound of the variance for control variate estimators, providing a strong theoretical foundation. Additionally, we develop a practical algorithm based on our framework and polynomials, offering a simple drop-in improvement over traditional Monte Carlo.

4.1 Related Work

4.1.1 Regression In Rendering

The rendering process often hinges on evaluating complex functions that can be computationally expensive. To overcome this limitation, researchers have proposed various methods that leverage approximations of these functions based on regression techniques. These approximations, or proxy functions, are significantly cheaper to evaluate at runtime, leading to substantial performance improvements.

One such application is adaptive sampling and reconstruction, as demonstrated by Zwicker et al. (2015). Their approach utilizes regression to create a substitute function that captures the essence of the original, allowing for more efficient sampling strategies. Similarly Moon et al. (2014) explored the use of local linear regression on auxiliary buffers, such as depth, normal, and texture information. This approach improves image filtering by effectively reconstructing the final image from these readily available buffers. Notably, Moon et al. (2014) placed a particular emphasis on minimizing the overhead associated with the regression step. They achieved this by employing an iterative linear model that reconstructs multiple pixels simultaneously, leading to a more efficient process.

These examples showcase the potential of regression-based approximations in rendering. By creating cheaper-to-evaluate proxies for complex rendering functions, researchers are paving the way for faster and more efficient rendering techniques.

4.1.2 Control Variate

Section 2.2.2 introduced the general concept of the Control Variates (CVs) estimator. Several methods have been developed specifically for rendering applications, each employing unique strategies to construct and adapt control variates. These methods range from using simple proxies for light sources to utilizing neural approximations and applying quadrature techniques to the entire path space rendering formulation.

Glynn and Szechtman (2002); Loh (1995) introduced this variance reduction technique in Monte Carlo simulations. Constructing effective control variates for complex integrands in light transport simulations, particularly rendering, poses a significant challenge. The simplest form utilizes a constant ambient term as a proxy for incoming illumination Lafortune and Willems (1994). More sophisticated approaches leverage specific properties of the rendering process. For example, Clarberg and Akenine-Möller (2008) exploit visibility correlations to design control variates for direct illumination. Vévoda et al. (2018) employ Bayesian regression to create a light clustering, followed by a control variate function derived from the estimated contribution of each cluster at any shading point. Kutz et al. (2017) proposed a control variate extension for rendering heterogeneous participating media. Their method relies on decomposing the medium into a control component, easier to integrate, and a residual component. These parts can then be sampled separately for increased efficiency. The applicability of control variates extends beyond light transport simulations. Belcour et al. (2018) demonstrate using CVs to integrate spherical harmonic expansions over polygonal domains even for non-analytic integrands.

The concept of control variates finds further application within the framework of Multilevel Monte Carlo (MLMC) methods. Notably, Heinrich (2001); Keller (2001) demonstrated the effectiveness of using low-resolution rendering as a control variate for high-resolution rendering within the MLMC framework. This approach leverages the inherent correlation between low and high-resolution results to achieve variance reduction in the final estimate.

Neural based control variates Building upon the concept of control variates, Müller et al. (2020) proposed using a neural network to construct a scene-specific control variate function. Their approach leverages a combination of normalizing flows and scaling parameters to approximate the scene's light field. The key insight is that if the network accurately represents a probability density function (pdf), its integral will inherently equal one. This allows them to learn a single scaling parameter that directly provides the integral of the control variate function, enabling efficient analytical integration. To maximize effectiveness, they propose a method that learns the light field globally for the entire scene using previously generated samples. This eliminates the need for pre-training the control variate and allows the function to be continuously refined during rendering. Additionally, they employ a separate neural network to guide the sampling process.

Following a different perspective, Subr (2021) introduced Q-NET, a family of neural network able to give the integral of sigmoidal universal approximators networks. Such method could be used in a control variate estimator co evaluate efficiently the integral of a proxy network used as control variate function.

4.2 Regression Based Control Variate

This section delves into the connection between our regression-based Monte Carlo estimator and control variates. We begin by highlighting the key differences between the statistical and calculus interpretations of Monte Carlo integration. Following this, we introduce the formulation of our estimator and discuss its properties. We then explore the specific case where polynomial model functions are employed and detail the distinctions between our approach and the method proposed by Crespo et al. (2021). Finally, we discuss some practical implementation considerations.

4.2.1 Dual Interpretation Of Monte Carlo Integration

The Monte Carlo method offers two distinct interpretations for estimating integrals: the statistical interpretation and the reconstruction interpretation. These interpretations provide different perspectives on how the integral is approximated using random samples.

Statistical Interpretation The statistical interpretation views the integral as the expected value of a random variable. To estimate this value, we draw N independent and identically distributed (i.i.d.) samples from the integration domain. The integrand is evaluated at each sample point. The integral is then approximated by the average of these function evaluations:

$$F = \int_{\Omega} f(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} \tag{4.1}$$

Here, F define the integral of the function f. The Monte Carlo estimation is the average value of the N samples x_i.

Calculus Interpretation The calculus interpretation, on the other hand, views the integration as a two step process. The replacement of the integrand f by a function g and the succeeding analytical integration g. Let us consider a constant function $g(x) = c$ as the model function. To keep its integral G equal to the original integral F, we would like to find a value for c such that

$$F = \int_{\Omega} f(x)\mathrm{d}x = \int_{\Omega} g(x)\,\mathrm{d}x = G. \tag{4.2}$$

The integral mean value theorem in calculus shows that there exist such a constant c for a given definite integral. Since the function $G(x)$ can be analytically integrated, we have

$$F = G = \int_{\Omega} g(x)\mathrm{d}x = c|\Omega| \tag{4.3}$$

where $|\Omega|$ is the volume of the integration domain. Therefore, the constant c can be derived as the average of $f(x)$ since

$$c = \frac{1}{|\Omega|}\int_{\Omega} g(x)\mathrm{d}x = \frac{1}{|\Omega|}\int_{\Omega} f(x)\mathrm{d}x \tag{4.4}$$

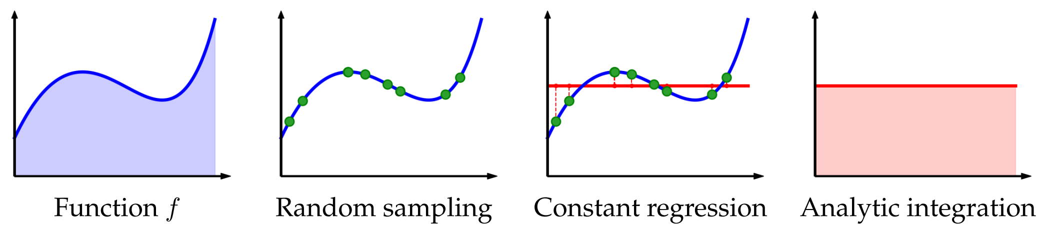


Figure 4.1: Visualization of the calculus-based interpretation of Monte Carlo integration. The process begins with random sampling of the target function. A constant regression is then performed over the sampled points. Finally, the resulting regressed function is analytically integrated, yielding the Monte Carlo estimate of the original integral.

where the last expression is the definition of the average of $f(x)$ over the integration domain. Note that there is no approximation introduced until this point. Since c is defined as the average of $f(x)$, we can estimate c by taking the average of samples:

$$c \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} = \langle F\rangle \tag{4.5}$$

This interpretation thus leads to the same estimator as the statistic-oriented interpretation since

$$F = \int_{\Omega} g(x)\mathrm{d}x = \int_{\Omega} c\,\mathrm{d}x \approx \langle F\rangle. \tag{4.6}$$

Under this interpretation, MC integration can be seen as the process of first estimating a function $g(x) = c$, and then analytically integrating (the approximation of) the function $g(x)$.

4.2.2 Regression Based Estimator

This work challenges the traditional boundaries of the calculus interpretation in Monte Carlo integration. We move beyond the use of a simple constant function and explore the potential of employing more sophisticated model functions. Our findings demonstrate that leveraging these more complex functions, in conjunction with control variate estimators and regression techniques, can lead to significant benefits. Notably, the classical Monte Carlo estimator emerges as a special case within our framework when the model function is a constant (or, more generally, when the regression process yields a constant function). Furthermore, we establish that our proposed estimator provably guarantees a lower or equivalent expected error compared to the classical approach. We have formulated this new class of estimators as regression-based Monte Carlo estimators.

Extended Calculus-Oriented Interpretation We deviate from the traditional calculus-oriented interpretation by introducing an arbitrary model function, denoted by $g(x)$. This function can be any function that can be integrated analytically, offering greater flexibility compared to a constant. Unlike the classical approach, the integral of $g(x)$ (denoted by G) is not required to be equal to the original integral we want to estimate (represented by F). This relaxation is crucial because, in general, it's not possible to find an arbitrary model function whose integral matches the original integral exactly.

However, even with this difference in integrals, we can still leverage the power of model functions by employing a control variate estimator (section 2.2.2). This estimator can compensate for the discrepancy between the integrals and ultimately provide an accurate estimate of F. To illustrate this concept, we can express the original integral as a combination of the model function and a residual term, as shown below, for any arbitrary function $g(x)$:

$$F = G + (F - G) \tag{4.7}$$

In the control variate estimator, $(F - G)$ is estimated by a classical Monte Carlo estimator.

$$F - G \approx \frac{1}{N} \sum_{i=1}^{N} \left(\frac{f(x_i) - g(x_i)}{p(x_i)} \right) = \langle R \rangle \tag{4.8}$$

The term R represents the difference between the integrals of the original function F and the model function g. By leveraging this residual, we can reformulate the integral estimation as:

$$F \approx G + \langle R \rangle = \langle I_{MC} \rangle_{CV}. \tag{4.9}$$

Restricting our choice of g to functions with analytically solvable integrals ensures the first term becomes a known constant, requiring no further estimation. While this reformulation resembles a control variate estimator, a key distinction sets our approach apart. Traditionally, control variate methods rely on a user-defined function for g. In contrast, our method leverages least-square regression to obtain this function from the available samples, leading to a data-driven approach.

Since G is deterministic, the expected squared error of the control variate estimator only depend on the residual term estimation. The variance of the estimator is given by :

$$\mathrm{E}\left[(\langle F \rangle - F)^2 \right] = \mathrm{Var}\left[\langle R \rangle \right] = \frac{1}{N} \mathrm{Var}\left[f(x) - g(x) \right]. \tag{4.10}$$

Conventional Monte Carlo integration suffers from variance. Control variates offer a solution, potentially reducing the expected squared error when the variance of the difference between the original function $f(x)$ and the control variate function $g(x)$ is lower than the variance of the original function itself. However, error reduction is not guaranteed, as acknowledged by Hickernell et al. (2005).

Our key contribution lies in demonstrating that least-squares regression not only automates the selection of $g(x)$ but also provably reduces the expected squared error when a constant function is included in the regression process. Notably, this inclusion of a constant function has been largely overlooked in traditional (adaptive) control variate methods. This work presents the first proof that combining least-squares regression with a constant function leads to guaranteed improvement, demonstrably reducing the error compared to standard Monte Carlo integration.

Least-Square Regression Of The Model Function To perform least squares regression, we first need to define a function space over which the regression will be computed. This involves specifying a function $g(x, \theta)$ with a set of parameters $\theta = (\theta_0, \cdots, \theta_M)$. The function space encompasses all possible functions $g(x, \theta)$ and defines the distance to the target function of the regression. The goal of least squares regression is to find the parameter set that minimizes this distance to the target function $f(x)$.

This process can be viewed as defining a class of functions and finding the best function within that class. The distance from a given function, defined by a set of parameters, to the target function is measured by the residual integral between the two functions:

$$R(\theta) = \int_{\Omega} \Big(f(x) - g(x, \theta) \Big)^2 \mathrm{d}x. \tag{4.11}$$

Minimizing this residual integral yields the optimal parameters for the regression function, ensuring the closest possible fit to the target function within the defined function space.

The quality of the regression, i.e., how close the regression function can approximate the target function or how small the residual can be, directly depends on the chosen function space. It depends on the number of parameters and how these parameters are utilized within the function definition. The regression process ensures finding the best set of parameters for a given function class, but the function space's expressiveness ultimately determines the regression's accuracy. Let us then consider regression of $g(x)$ to $f(x)$ such that the residual $R(\theta)$ is minimized. Since analytical solution to the integral in $R(\theta)$ is usually not available, we first consider using N random samples x_i to approximate $R(\theta)$.

$$R(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\frac{f(x_i) - g(x_i, \theta)}{p(x_i)} \right)^2 = \langle R(\theta) \rangle, \tag{4.12}$$

which is equivalent to Monte Carlo integration of the function $(f(x) - g(x, \theta))^2$ using the estimator $\langle R(\theta) \rangle$. This unbiased estimation thus converges to $R(\theta)$ as we increase the number of samples. This expression of $\langle R(\theta) \rangle$ can also be seen as the squared $\mathcal{L}_2$ norm of the differences between $f(x_i)$ and $g(x_i, \theta)$ on the samples x_i. The solution to this regression can be obtained via least-squares regression of $g(x, \theta)$ to N pairs of $(x_i, f(x_i))$. Figure 4.2 visualizes our estimator, including the least squares regression using a linear function and the control variate using analytic integration along with the residual term.

Minimal Example Listing 4.1 presents a minimal example of our estimator implemented in Python using the NumPy library. This code snippet highlights the three primary components of our method: regression, analytic integration of the polynomial via the anti-derivative, and the control variate estimator with the residual term estimation.

In the regression step, we use $np.polyfit$ to fit a polynomial to the sampled data, generating the coefficients for the polynomial function. Next, the analytic integration of the polynomial is performed by computing its anti-derivative $np.polyint$, which allows us to evaluate the integral of the polynomial function. Finally, the control variate estimator is constructed by combining the polynomial approximation with the residual term, which is estimated from the sample data. This minimal example demonstrates the core operations required to implement our estimator.

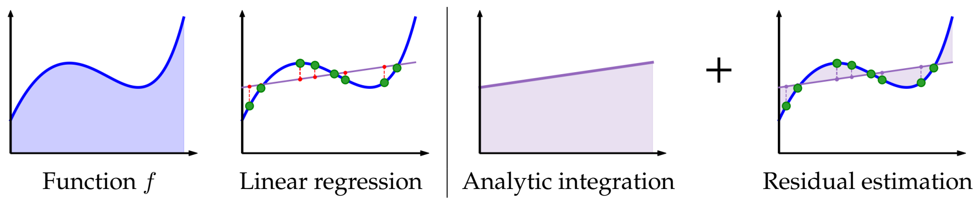


Figure 4.2: Visualization of a regression-based estimator using a linear model. On the left, the target function and its linear regression approximation are shown. On the right, the control variate estimator is illustrated, consisting of the analytic integration of the regressed function combined with a Monte Carlo estimate of the residual difference between the target function f and its regression.

```python
import numpy as np

def function(x):
    # insert your function there
    x = x*3
    return (-0.8*x*x+0.8*x + 0.3*np.exp(x))*1.7 # Example function


N = 128 # number of samples
O = 3 # Polynomial order

x = np.random.random(N)
y = function(x)

z = np.polyfit(x, y, O) # compute regression
g = np.poly1d(z)

G = np.polyint(g) # Compute the anti-derivative function

result = (G(1)-G(0)) + (y - g(x)).mean()
```

Listing 4.1: Python example of our regression estimator

4.2.3 Properties

Our estimator exhibits several important properties. Firstly, if the function space only includes the constant function, the regression task simplifies to averaging the samples to obtain a constant regression. This scenario is equivalent to using a classical Monte Carlo estimator following the calculus interpretation. Consequently, in the worst-case scenario, if the function space for regression contains the constant function, the regression result will at least be a constant, equating to a classical Monte Carlo estimator. In other cases, our regression approach converges to a more accurate fit to the target function, leading to variance reduction. Additionally, our estimator encompasses some classical control variate estimator formulation. Finally, our estimator inherently minimizes an upper bound on the variance of the control variate estimator, ensuring improved performance over traditional Monte Carlo methods.

Reduction To Monte Carlo Integration In the case where the function class only considers the constant function, meaning it has only one parameter, the regression process can be defined as:

$$g(x, \theta) = \theta_0 \tag{4.13}$$

where θ_0 is the single parameter. The least squares minimizer for this scenario has a closed-form solution, which is simply the average of the sample points:

$$\theta_0 = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \tag{4.14}$$

In this case, the resulting control variate estimator is identical to the classical Monte Carlo estimator. This demonstrates that our estimator framework is a general class of estimators, with the classical Monte Carlo estimator being a specific instance of regression within our framework.

Reduced Variance Over Monte Carlo In more complex regression settings—provided that the function space includes constant functions—our estimator converges to a lower expected squared error as the regression becomes a better approximation of the true integrated function with increasing sample count. At low sample counts, the regression can be unstable and may occasionally perform worse than the classical Monte Carlo estimator. However, as more samples are used, the regression converges toward the optimal least-squares solution, reducing the residual term and thus improving variance reduction. This follows directly from comparing residuals: because the constant function is contained in the regression space, the residual $R(\theta)$ of the learned regression must converge to a value less than or equal to the residual of the constant regression $R(c)$.

$$R(\theta) \leq R(c). \tag{4.15}$$

This inequality shows that the least squares regression process converges to a minimization of the residual error more effectively when considering a broader function space compared to only using a constant function. Consequently, the expected squared error of our estimator converges to a lower value than that of the classical Monte Carlo estimator. This inherent reduction in residual error translates directly to a reduction in the variance of our estimator compared to the Monte Carlo estimator:

$$\mathrm{E}\left[(\langle I_{MC}\rangle_{CV} - F)^2\right] \leq \frac{1}{N} R(\theta) \leq \frac{1}{N} R(c) = \mathrm{E}\left[(\langle I_{MC}\rangle - F)^2\right] \tag{4.16}$$

$$\text{therefore } \mathrm{E}\left[(\langle I_{MC}\rangle_{CV} - F)^2\right] \leq \mathrm{E}\left[(\langle I_{MC}\rangle - F)^2\right]. \tag{4.17}$$

Unlike classical control variates, our formulation is designed to naturally converge toward an estimator that outperforms standard Monte Carlo as the regression improves. However, to guarantee that the estimator never performs worse than Monte Carlo, one can still introduce the classical α safeguard, which scales the control term according to its correlation with the integrand. Since constant functions are included in the regression function space, the method is always capable of reverting to the Monte Carlo solution when needed, while the α parameter ensures strict variance reduction whenever the control variate is beneficial.

Control Variate Formulation Under Our Framework Even if the framework use user-defined classes of functions rather than specific user-defined functions, the method remains applicable to existing works. In this scenario, the function itself has no parameters θ. More interestingly, the scaling parameter α, traditionally used to maximize the effectiveness of the estimator, can be interpreted within our framework as a single scaling parameter for the user-defined control variate function.

$$g(x, \theta) = \underbrace{\theta_0}_{\alpha} \cdot h(x) \tag{4.18}$$

With $h(x)$ the user defined function and θ_0 the scaling factor obtain through regression. Consequently, our method provides a unified and flexible way to optimize control variates.

Variance Upper Bound Minimization The variance of a control variate estimator can be derived as follows:

$$\mathrm{Var}\left[\frac{f(x) - g(x,\theta)}{p(x)}\right] = \mathrm{E}\left[\left(\frac{f(x) - g(x,\theta)}{p(x)}\right)^2\right] - \mathrm{E}\left[\frac{f(x) - g(x,\theta)}{p(x)}\right]^2 \tag{4.19}$$

$$= \int_\Omega (f(x) - g(x,\theta))^2 \, \mathrm{d}x - (F - G)^2 = R(\theta) - (F - G)^2. \tag{4.20}$$

The variance of a control variate estimator is composed of two terms. The first term is the residual between the control variate function and the integrated function. The second term, subtracted from the first, is the squared bias between the integral of the two functions. Therefore, our regression minimizing the first term effectively minimizes an upper bound of the variance:

$$\mathrm{E}\left[(\langle I_{MC}\rangle_{CV} - F)^2\right] = R(\theta) - (F - G)^2 \leq R(\theta) \tag{4.21}$$

As stated at the beginning of the section, it is not possible to ensure that the bias term is zero with an arbitrary regressed function. This is primarily because the integral F is unknown and lacks a closed-form solution, unlike the special case of constant regression. Consequently, minimizing the variance further for a given function space is not feasible. The only way to achieve greater variance reduction is by enhancing the function basis to reduce the residual term more effectively. This can be accomplished by using more complex function bases. The more refined and sophisticated the basis, the better the residual reduction. Additionally, incorporating knowledge about the function's shape to better fit its particular characteristics can also help reduce the residual term.

4.2.4 Polynomial Basis

While our formulation is general and could work for any arbitrary model function g with an analytic integration, we focus on a common class of functions: polynomials. This choice is motivated by several factors. First, polynomials have a good capacity to approximate smooth functions. Second, they scale simply with dimensions and the number of parameters, making them computationally efficient. Third, the integrals of polynomial functions are straightforward to compute. The choice of polynomials also has practical implications; linear function regression is simpler to solve compared to non-linear models. Finally, polynomials inherently include constant functions, which are necessary to guarantee enhancements over Monte Carlo. Despite our focus on polynomials for the experimental results presented later, our formulation is inherently general. The choice of polynomials serves to demonstrate the efficacy of our approach, but other function classes could be used within the same framework to potentially achieve different or improved results.

A polynomial function can be expressed as a sum of weighted monomials as follows:

$$g(x,\theta) = \sum_{m=0}^{M} \theta_m \phi_m(x) \tag{4.22}$$

With $\phi_m(x)$ the monomials defining the polynomial basis and θ_m the scaling of the monomial. A monomial for a polynomial is a single term consisting of a product of constants and variables raised to non-negative integer powers. In general, a monomial in one variable x can be written as x^n, where n is a non-negative integer exponent. For multiple dimensions, a monomial takes the form $x_1^{n_1} x_2^{n_2} \cdots x_k^{n_k}$, where $n_1, n_2, \cdots, n_k$ are non-negative integer exponents for each dimension $x_1, x_2, \cdots, x_k$ of the vector x.

As the function is a linear combination of the monomials, it's integral is a weighted sum of each monomial integral time it's coefficient θ_m:

$$\int_\Omega g(x,\theta)\mathrm{d}x = \sum_{m=0}^{M} \theta_m \int_\Omega \phi_m(x)\mathrm{d}x \tag{4.23}$$

Monomials integral have close form solution and can be evaluated at low cost. In more complexes cases with linear combination this integrals can be pre-computed.

We tested the effectiveness of polynomial regression on simple integrals. Figure 4.3 illustrates the regression results using polynomials on four types of integrands (left column). The center-left column displays the regression obtained with three different polynomials (order 1, 3, and 5), with the target function shown as a dashed line. The center-right column depicts the difference, or residual, between the different regressed polynomials and the target function. The right column shows the error convergence with sample count for the regression-based estimator for each polynomial, using the Monte Carlo estimator as a baseline. These results highlight that polynomials consistently achieve lower error than the Monte Carlo method in each scenario, even when the regression does not perfectly fit the function. The smaller the residual, the greater the improvement over Monte Carlo. Additionally, higher-order polynomials yield better results because higher-order polynomials include lower-order ones, thus expanding the function space and enabling better regression and error reduction. Notably, even for non-smooth functions like the step function, this approach manages to improve upon the classical estimator.

4.2.5 Difference To Crespo et al. (2021)

The method proposed by Crespo et al. (2021) is the closest to our approach, sharing several similarities but also presenting important theoretical and practical differences.

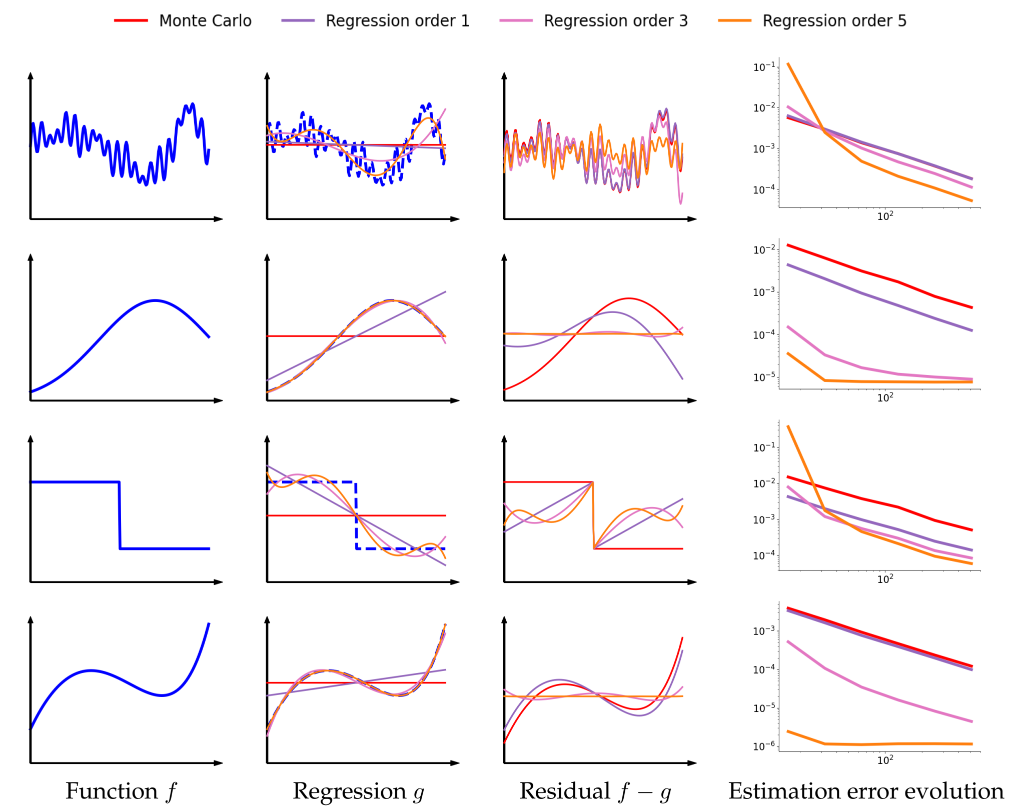


Figure 4.3: Example of 1D function integration using our regression based estimator. For each function we display the function alone (left), the associated regression using polynomial of order 1, 3 and 5 as well as the classical constant term of a Monte Carlo estimator(center left), the residual term (center right) and the error evolution with respect to the sample count (left).

Both methods employ adaptive control variates using polynomial approximations of the function. This shared foundation highlights the importance of polynomial-based strategies for variance reduction in Monte Carlo integration.

Despite these similarities, there are several key differences. First, in the construction of the control variate function, Crespo et al. (2021) use a quadrature method, while we compute the regression directly on Monte Carlo random samples. Quadrature requires fewer samples than regression, but the samples for quadrature cannot be selected randomly and thus cannot be used for residual estimation. In contrast, our method uses the same samples for both regression and residual estimation, making more efficient use of the sample budget.

In terms of sample efficiency, Crespo et al. (2021)'s control variate function is computed before the sampling of the Monte Carlo samples, consuming part of the sampling budget. Conversely, our control variate function is constructed from the randomly sampled data points. Although the quadrature-based approach can enable importance sampling based on the quadrature, our method allows all samples to contribute to both regression and residual estimation, providing more stable and uniform improvements over Monte Carlo methods.

Regarding quality and robustness, Crespo et al. (2021)'s combination of quadrature and adaptive importance sampling can achieve significantly higher quality than our estimator in the best cases. However, this adaptive sampling relies on the quadrature, and if the quadrature misses important high-variance regions, it can lead to poor importance sampling and increased error. Our method, on the other hand, ensures statistically stable and uniform improvements over Monte Carlo without the risk of increased error. While our method may not achieve the same peak quality improvement as Crespo et al. (2021), it provides consistent enhancements and cannot perform worse than classical Monte Carlo methods.

Our method offers broader theoretical properties and generalization compared to Crespo et al. (2021)'s approach, which primarily focuses on practical variance reduction using quadrature-based polynomials. Our method provides a broader theoretical framework that generalizes to various classes of regression, not limited to polynomials or multi-level piece-wise polynomials. It has more formal definitions and provable properties, providing a more comprehensive understanding of the underlying principles.

While both methods offer significant benefits for variance reduction in Monte Carlo integration, they cater to different needs and scenarios. Crespo et al. (2021)'s approach provides a practical method with high variance reduction potential, particularly in best-case scenarios with effective adaptive sampling. Our method, on the other hand, ensures stable and consistent improvements with more robust theoretical guarantees and broader applicability to various regression classes. Depending on the specific requirements and properties desired in the estimator, both approaches offer valuable tools for enhancing Monte Carlo integration quality.

4.2.6 Practical Computation

Our estimator's regression component can be computed through various methods, allowing for flexible implementation. In the context of rendering, we focused on two distinct approaches: a direct matrix solution and a gradient-based optimization method. Each approach offers unique properties and advantages.

The direct matrix solution is straightforward and computationally efficient. It provide a direct way to calculate the regression coefficients for linear model function. In contrast, the gradient-based optimization method is well-suited for handling complex and non-linear function. This approach offers adaptability and robustness but require to run many iteration to converge to the regression solution.

Direct Estimation The direct estimator is designed for linear functions, solving the regression task by addressing a linear system of equations. The regression task involves finding the set of function parameters θ that minimize the residual function. This is achieved by identifying the parameters for which the derivative of the residual function is zero for all parameters simultaneously:

$$\frac{\partial \langle R \rangle(\theta)}{\partial \theta_q} = 0 \qquad \forall q \in \{0, 1, ..., M\} \tag{4.24}$$

Given a function that can be expressed as a linear combination of basis functions:

$$g(x, \theta) = \sum_{m=0}^{M} \theta_m \phi_m(x) \tag{4.25}$$

Where $\phi_m(x)$ are the basis functions, we can derive the residual term's derivative:

$$\frac{\partial \langle R \rangle(\theta_p)}{\partial \theta_q} = \sum_{i=1}^{N} \phi_p(x_i)(g(x_i, \theta) - f(x_i)) = 0 \tag{4.26}$$

By setting the derivative to zero, we obtain a set of linear equations for the parameters θ_i. These equations can be represented in matrix form:

$$\begin{bmatrix} \Phi(0,0) & \cdots & \Phi(0,M) \\ \Phi(1,0) & \cdots & \Phi(1,M) \\ \vdots & & \\ \Phi(M,0) & \cdots & \Phi(M,M) \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_M \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} f(x_i)\phi_0(x_i) \\ \sum_{i=1}^{N} f(x_i)\phi_1(x_i) \\ \vdots \\ \sum_{i=1}^{N} f(x_i)\phi_M(x_i) \end{bmatrix} \tag{4.27}$$

with :

$$\Phi(p, q) = \sum_{i=1}^{N} \phi_p(x_i)\phi_q(x_i) \tag{4.28}$$

In this linear system the matrix represent the basis functions evaluated at sample points and the vector the target function values. Each element of the matrix and vector is itself a Monte Carlo estimator. The solution parameters θ con be obtain by inverting the matrix of basis and multiplying it with the target function vector. This matrix representation allows for an efficient solution of the regression coefficients θ, ensuring the minimization of the residual function in a computationally straightforward manner.

Algorithm 1 Direct matrix estimator

```
1: $f \leftarrow \{\}$; $u \leftarrow \{\}$; $\Phi_0 \leftarrow 0^{M\times M}$; $b_0 \leftarrow 0^{M\times 1}$
2: **for** $i \leftarrow 1$ **to** $N$ **do**
3:     $u_i \leftarrow$ random()
4:     $f_i \leftarrow f(u_i)$
5:     $(\Phi_{i+1}, b_{i+1}) \leftarrow$ update_system($\Phi_i$, $b_i$, $u_i$, $f_i$)
6: **end for**
7: $\theta_{\text{min}} \leftarrow$ solve_linear_system($\Phi_{N+1}$, $b_{N+1}$)
8: **Return** $G(\theta_{\text{min}}) + \frac{1}{N}(\sum_{i=1}^{N} f_i - g(u_i, \theta_{\text{min}}))$
```

Our regression-based estimator, utilizing the matrix-based approach, iteratively constructs the linear system by updating the basis matrix and the target vector with each new data sample. This update step takes $M^2 + M$ operations with M the number of basis function. Once all data samples are collected, the linear system is solved using matrix inversion, and the optimal set of parameters is then used to compute the control variate estimator. This procedure is detailed in algorithm 1.

Despite its effectiveness, this approach has some limitations. Since it relies on a matrix formulation of a linear system and requires matrix inversion to solve it, the matrix must be invertible and the inversion numerically stable. This necessitates having linearly independent basis elements and ensuring that the Monte Carlo estimation of each matrix element has limited noise. To achieve a correct solution for the linear system, a careful selection of basis elements and a minimum number of samples (more than the number of basis elements) are essential.

In practice, these requirements are straightforward to meet with polynomials. By construction, polynomial basis elements are linearly independent. Additionally, the number of basis elements can be adjusted based on the available sample budget for rendering. This inherent flexibility supports the choice of polynomials for regression in our approach.

Iterative Optimization An alternative approach to our regression task involves using an iterative gradient-based optimization of the control variate function to minimize the residual term. This method works by computing the derivative of the residual with respect to all parameters and applying iterative updates in a stochastic gradient descent (SGD) manner.

$$\theta_{t+1} = \theta_t - \eta \left(\dots , \frac{\partial \langle R \rangle (\theta_t)}{\partial \theta_q}, \dots \right) \tag{4.29}$$

As optimizing the parameters may require many iterations, we begin by generating the expected number of random samples. We then run the SGD over these generated samples until the parameters converge. Finally, we compute the control variate estimator using the optimized parameters. This approach is summarized in Algorithm 2.

Several factors impact the regression speed of this approach. First, the initialization of the parameters plays a crucial role in determining the number of optimization steps required. Additionally, the choice of gradient-based optimizer can significantly accelerate the process. In particular, Adam or other momentum-based optimizers are especially useful for achieving fast convergence.

This gradient-based optimization method is practical for non-linear functions and multi-frame rendering, as previous results can serve as initialization for subsequent frame renderings. It also supports the advantage of not necessarily requiring the explicit derivative of the model with respect to parameters, as automatic differentiation tools can be employed. Furthermore, this approach allows for the flexible modification of the residual formulation to accommodate other classes of regression.

Algorithm 2 Gradient descent estimator

```
Parameter $lr$ : learning rate, $T$ : number of iterations
$f \leftarrow \{\}; u \leftarrow \{\}$
for $i \leftarrow 1$ to $N$ do
    $u_i \leftarrow$ random()
    $f_i \leftarrow f(u_i)$
end for
$\theta_{\text{min}} \leftarrow 0^{M \times 1}$
for $t \leftarrow 1$ to $T$ do
    $\theta_{\text{min}} \leftarrow$ gradient_descent($\theta_{\text{min}}$, lr, $u_i$, $f_i$)
end for
Return $G(\theta_{\text{min}}) + \frac{1}{N}(\sum_{i=1}^{N} f_i - g(u_i, \theta_{\text{min}}))$
```

The gradient-based method offers a powerful approach to optimizing a polynomial, particularly due to the convex nature of the problem, which facilitates easy convergence. The straightforward computation of derivatives of the polynomial requires only one evaluation of the polynomial for each sample and the evaluation of each monomial. This efficiency makes the gradient-based method potentially faster to compute compared to the direct matrix solution, although converging only to the regression solution where the direct matrix provides an exact solution. However, the most compelling aspect of this approach lies in its applicability to non-linear model functions. Specifically, it enables the utilization of neural networks, such as those discussed in the work of Müller et al. (2020), expanding the scope of potential applications.

Bias Correction Using the same sample set for both the regression and the estimation of the residual term in the control variate estimator can introduce bias. Although this bias was not evident in our experiments due to the noise level, it could become problematic in other applications or with different sample counts. A straightforward solution to mitigate this bias involves dividing the sample budget into two sets of similar sizes and performing a separate regression for each set. The residuals for each regression are then computed using the other sample set.

This approach ensures that all data are used for both regression and residual computation without introducing bias, as each step is performed with uncorrelated samples. This method maintains the integrity of the estimator by preventing any overlap between the samples used for regression and those used for residual estimation, thus eliminating potential bias. However, implementing this method does come with certain requirements and trade-offs. Firstly, it necessitates a sufficiently large sample set to perform two separate regressions effectively. This could be a limiting factor in scenarios where the sample budget is tight. Secondly, there is an additional computational overhead associated with performing the second regression, which could impact performance, particularly in real-time applications.

Despite these challenges, the dual-set approach provides a robust solution to the potential bias problem, ensuring that our estimator remains accurate and reliable across a wider range of applications and sample counts.

Other Regression Methods We have detailed two methods to compute the regression: a direct matrix solution and an iterative gradient-based optimization approach. Both methods are straightforward and well-suited for polynomial functions and rendering setups. The direct matrix solution is computationally efficient for linear regressions, while the gradient-based method offers flexibility for more complex and non-linear regressions.

However, our approach is not limited to these methods. Other regression techniques can also be applied within our framework. For instance, ridge regression and Lasso regression can be used to handle multicollinearity and enforce sparsity in the coefficients, respectively. Support Vector Regression (SVR) provides robust performance in high-dimensional spaces and can handle non-linear relationships with kernel functions. Additionally, neural network-based regression can capture complex patterns and interactions in the data, offering powerful approximations for intricate functions. Bayesian regression offers probabilistic interpretations and can incorporate prior knowledge into the model, which can be particularly useful in scenarios with limited data.

Each of these alternative regression methods brings unique strengths and can be selected based on the specific requirements and constraints of the application. Our framework's flexibility allows for the integration of various regression techniques, enabling it to adapt to a wide range of problems beyond rendering and polynomial approximations.

4.3 Results

Our implementation uses PBRT v3 Pharr et al. (2016a) as a rendering system and Eigen Guennebaud et al. (2010) as a solver for linear systems. All the results were generated on AMD EPYC 7702 64-Core or Intel Core i9-8950HK Processor using eight threads for each. For comparison, we compare our method against conventional Monte Carlo integration and Crespo et al. (2021) using the relative mean squared error (relMSE) eq. (2.32). The reference image for each scene is computed with 65,536 samples per pixel.

To handle a vector (i.e., RGB) integrand in rendering, we first estimate the RGB estimate F_{rgb} by averaging RGB samples just as $F_{rgb} \approx \langle I_{MC}^{rgb} \rangle_N = {}^1/_N \sum_{i=1}^{N} f_{rgb}(u_i)$ where $f_{rgb}(u_i)$ is the integrand that returns an RGB sample. We then perform regression on the luminance value of the samples, resulting in the luminance valued G as the analytical integral. The reconstruction of the RGB value based on our estimator is written as $\langle I_{CV}^{rgb} \rangle_N = ({}^{\langle I_{MC}^{rgb} \rangle_N}/_{y(\langle I_{MC}^{rgb} \rangle_N)}) \langle I_{CV} \rangle_N$ where y is the luminance function and $\langle I_{CV} \rangle_N$ estimate the luminance value of $\langle I_{MC}^{rgb} \rangle_N$ based on our method. In other words, we correct the noisy luminance value of $\langle I_{MC}^{rgb} \rangle_N$ by our estimate.

4.3.1 Direct Light Comparison

We show in Figure 4.4 an equal-time comparison between the conventional Monte Carlo integration and our approach using polynomials of order 1 to 3 on three different scenes.

The KILLEROOS scene (top row, Figure 4.4) is composed of a single spherical light and motion blur, which is a 3-dimensional integration problem. Despite the fact that motion blur introduces more discontinuities in the integrand—which makes it difficult

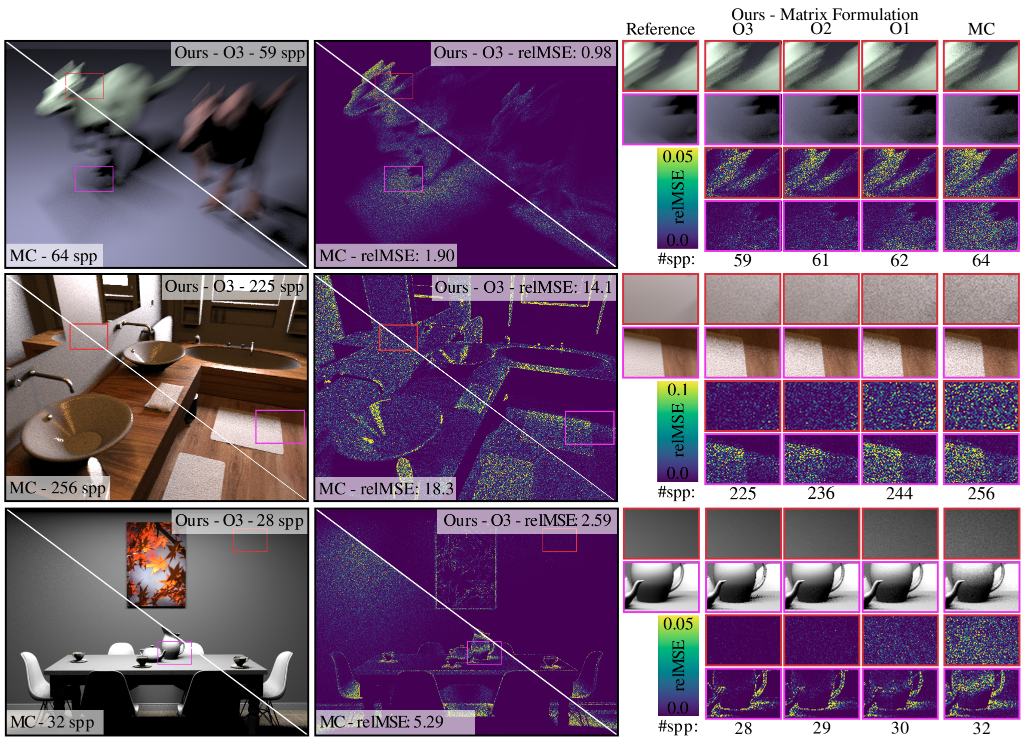


Figure 4.4: Equal-time comparisons between regression-based and conventional Monte Carlo integration. We demonstrate the effectiveness of our approach with polynomials of order 1 to 3. The rendering involves 3D integration for all scene. All RelMSE values are scaled by 10^3.

to approximate by polynomials of any order—our method can still reduce the variance in most of the regions. This result demonstrates that it is possible to achieve variance reduction with model functions that only roughly approximate the underlying integrand.

The BATHROOM scene (middle row, Figure 4.4) has multiple mesh light sources picked randomly using a random number, making it also a 3-dimensional integration problem. Moreover, each light in this scene is a mesh light with several triangles making discontinuities in the 2D area light parameterization. For this scene, the improvement in our method is limited but no worse than Monte Carlo integration.

The DINING ROOM scene (bottom row, Figure 4.4) is composed of a single triangle light source for direct illumination, making it a 2-dimensional integration problem. Our method shows significant variance reduction compared to Monte Carlo integration as predicted. Even in the penumbra region where the reduction of error is small, our method is no worse than Monte Carlo integration.

Figure 4.5 shows the error convergence plots for the scenes described above. As predicted by our theory, our method consistently outperforms Monte Carlo integration once the solution to regression is almost converged. At a fewer sample count than 10 spp, our method with higher order polynomials (order two and three) may perform worse than Monte Carlo integration due to the error in regression. This is because the matrix tends to be ill-conditioned due to noise in the sampling. Even though a linear model function

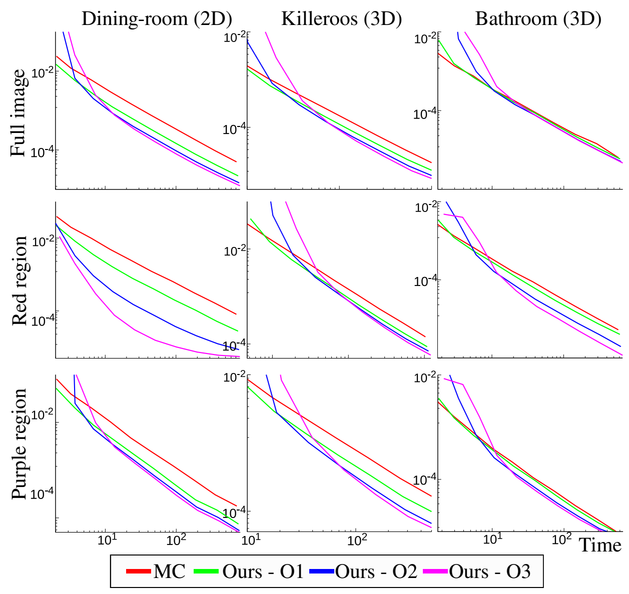


Figure 4.5: Convergence plot for equal-time comparisons in fig. 4.4. The metric is relMSE. Our method converges faster than Monte Carlo estimation in all but has similar performance in the BATHROOM scene.

(O1) is only slightly more complex than a constant model function (Monte Carlo), our method with O1 significantly outperforms Monte Carlo integration. We also observe that the maximum possible reduction in variance is limited by the complexity of the model function. There is thus a trade-off between the maximum reduction of variance and its stability at a few sample count, when one chooses a model function. The order or number of basis also seems to impact the overhead.

4.3.2 Global Illumination Comparison

We start the numerical analysis with simple analytic integrands. In fig. 4.6, we show the convergence curves over 1D, 5D and 15D space for two integrands: (a) sum of sinusoids and (b) a multi-dimensional exponential function. In both cases, our method shows significant improvements in variance reduction. Both Order 1 and Order 2 polynomials offer improvements. However, at low sample count (up to about 100 samples) in 15D, Order 2 polynomial performs poorly. This is due to the fact that, as the polynomial order increases in higher dimensions, a larger number of samples is required to obtain an accurate regression.

We also applied our method to a path tracing rendering with higher dimensional rendering scenario. Figure 4.7 shows the results of our method for a DUCK-CORNELL scene rendered in path tracing with 3 bounces. This corresponds to a 15D problem in our algorithm: a 3D random number for each next-event estimation and a 2D random number for the next-direction sampling at each bounce. Improvements are visible in both directly- and indirectly-illumniated regions. However, the improvements in the indirectly-lit regions are less due to the strong variations of the function. These variations come from the light making several bounces, which makes the control variate function less accurate

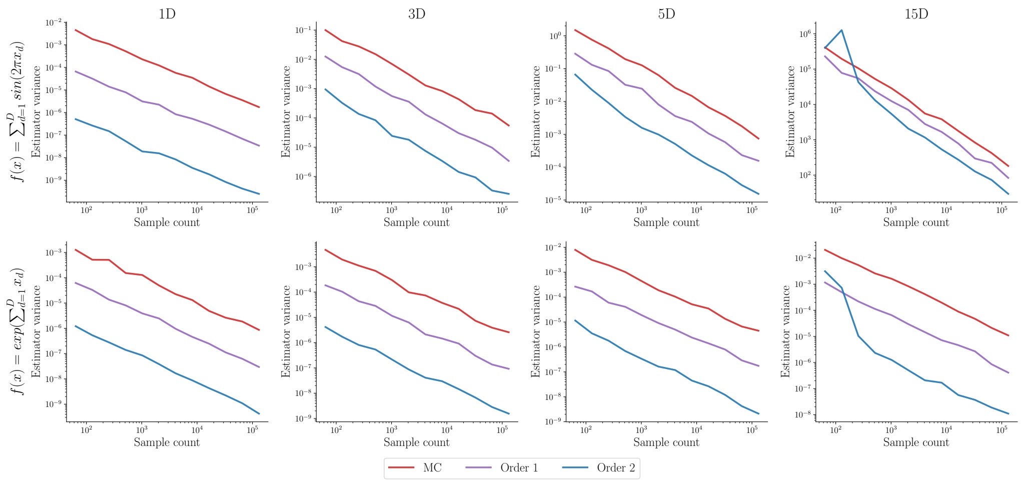


Figure 4.6: Convergence plots comparing our method using polynomial bases with a classical Monte Carlo estimator at different dimensionality. We observe that performance stays unchanged if the function complexity does not depend on the integration dimensionality (bottom row). Otherwise, our method experiences a slight decrease of improvement with the dimensionality but is still clearly advantageous compared to the Monte Carlo estimator (top row).

than in the case of direct illumination. One can of course consider having a more complex model function, but it will put an additional computational overhead on our method.

Figure 4.8 shows the improvement due to our method on illumination coming from different number of bounces. Our method provides the largest improvement on direct illumination (the top row), and the improvement generally decreases with more bounces (the middle and the bottom rows). As we discussed above, it is not directly due to the increased number of dimensions for higher number of bounces. We speculate that it is rather specific to the fact that the integrand in the primary sample space becomes more and more complex for higher number of bounces. Similar observations have been made in prior work that attempt to approximate the integrand in the primary sample space Guo et al. (2018); Zheng and Zwicker (2019); Crespo et al. (2021), so we believe that it is not specific to our regression approach. We note that our method is still not performing worse than Monte Carlo integration even for higher number of bounces.

4.3.3 Combination With Multiple Importance Sampling

In Figure 4.9, we show that our method can be used in conjunction with MIS Veach and Guibas (1995). From different sampling strategies available during MIS, we simply perform the regression on weighted samples from each sampling strategy. The figure shows the results using 64 samples per pixel of our method with two different importance sampling techniques: BRDF and light sampling. We compare the balance heuristic (Veach, 1997) and the optimal weights (Kondapaneni et al., 2019). Our method improves the quality of the result for both weighting schemes compared to Monte Carlo counterpart. We also found that the combination of our method with optimal weight (Kondapaneni et al., 2019) keeps the strength of these two techniques, leading to the best variance reduction. The error reduction brought by the optimal weights allows a closer regression to

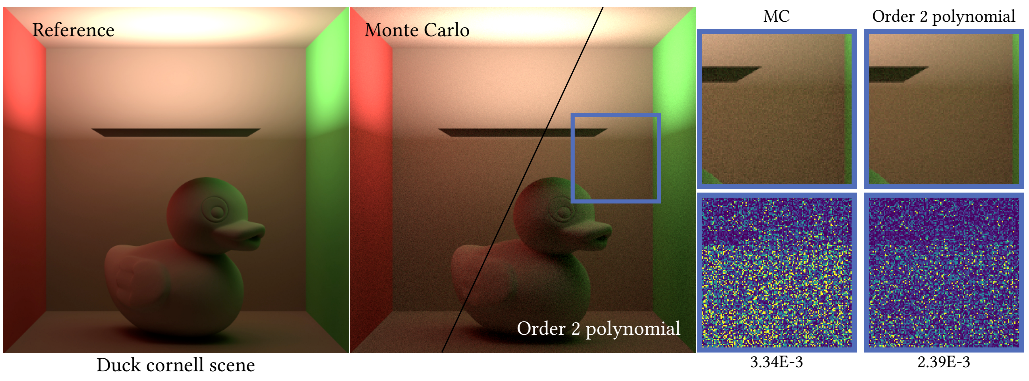


Figure 4.7: Equal sample comparison between classical Monte Carlo and our method using an Order 2 polynomial basis. In this DUCK-CORNELL scene, only the upper part is directly illuminated by the light source. The lower part has indirect-illumination. The error map (relMSE) shows clear improvements using our method. relMSE values marked at the bottom.

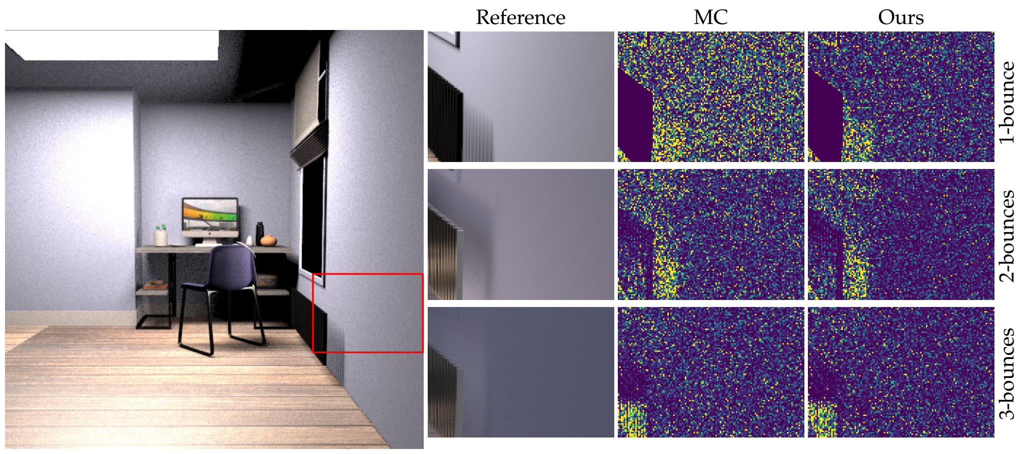


Figure 4.8: Evaluation of the impact of path length on integrand complexity and performance. We compared our method with standard Monte Carlo estimator across increasing light path lengths: 1, 2, and 3 bounces, corresponding to 3D, 5D, and 7D integration domains, respectively. At equal sample count, the benefits of our approach is more pronounced at the lowest number of rendering bounces as the complexity of the integrand is lower.

the integrand leading to a greater error reduction. This result indicates that our method can take advantage of orthogonal variance reduction methods that simplify the shape of the integrand in primary sample space.

4.3.4 Comparison To Crespo et al. (2021)

Crespo et al. (2021) proposed two adaptive control variate techniques using quadratures: one based on a quadrature per pixel and one with a global quadrature for the whole image. As we also treat each pixel independently, we have chosen to compare our method

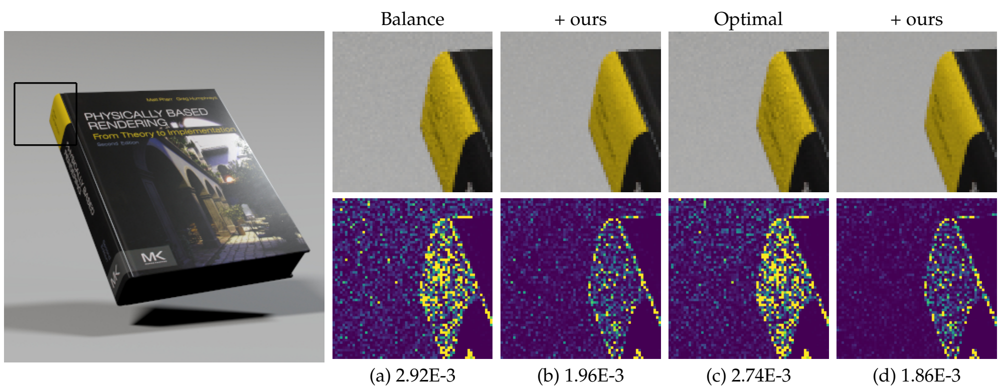


Figure 4.9: Equal-sample comparison of MIS strategies combined with our method. We compare two Multiple Importance Sampling (MIS) techniques (BRDF sampling and light sampling) using both balance heuristics Veach (1997) and optimal weights Kondapaneni et al. (2019). Our method is integrated using an order-2 polynomial basis. (a–d) The error maps and corresponding relMSE values shown below each image highlight the improvements achieved by our method in both configurations.

to the former method, which has been shown to be competitive. Specifically, Crespo et al. (2021) construct an adaptive piecewise-polynomial control variate (per pixel) in the primary sample space by recursively splitting the space based on an error heuristic built upon nested quadrature rules and regions' volume. We leave the possibility of extending our approach over a block of pixels as future work.

Figure 4.10 provides an equal-time comparison of Crespo et al.'s approach with our Order 2 polynomial basis method for a 2D direct illumination integration in two scenes. The DRAGON scene consist of a simple object lit by a high-frequency environment map. The CHOPPER-TITAN scene contains glossy complex objects lit by a low-frequency environment map. For both scenes, we show the reference control variate function, the control variate functions constructed by Crespo et al. and our method (columns (a)-(c)). We also shows on columns (d)-(f) the error images of MC and these different technique.

For functions shown in column (a), Crespo et al.'s method can produce very accurate function approximations, resulting in error reduction compared to MC method (d). However, in columns (b, c), their adaptive control variate construction misses some critical function regions, resulting in a poor approximation. Moreover, stratifying the samples across regions with importance sampling increases this problem, resulting in higher error than MC (columns e,f). In comparison, our method uses a crude approximation of control variate via a polynomial function but consistently outperforms the classical MC and is more robust compared to Crespo et al.'s method.

Figure 4.11 shows an equal-sample comparison between Crespo et al.'s method and ours. For the VW-VAN and the TEAPOT scenes, our method significantly outperforms Crespo et al.'s in difficult regions such as the bottom of the van and the reflections on the bottom side of the teapot. However, for the HOUSE scene, Crespo et al.'s method performs better especially on the house's facades. Despite such, it is noteworthy that the error reduction by our method is more uniform across the scenes. Crespo et al.'s method, on the other hand, shows significant improvement upon classical MC in some regions, e.g., directly illuminated regions of the VW-VAN and the TEAPOT, but performs worse than MC in some other regions.

Figure 4.12 demonstrates the impact of importance sampling in Crespo et al.'s method. We found that their importance sampling strategy can increase variance, which can be explained by closely investigating their adaptive subdivision scheme. In the first row of fig. 4.10, Crespo et al.'s control variate function misses important integrand features. They used this approximation for importance sampling as well which may increase variance.

Figure 4.13 shows another experiment with no adaptive subdivision in the method of Crespo et al.. Without adaptive subdivision, their method falls back to a control variate estimator using a simple quadrature for a polynomial model function. This setup looks deceivingly equivalent to ours, but they are different. Crespo et al.'s method finds a polynomial via a quadrature rule with a fixed set of points within the domain for their control variates. Consequently, their control variate cannot take advantage of any more samples than being required for the quadrature rule without splitting. Our method, on the other hand, can take any number of samples to improve the least-squares fit without any splitting. The least-squares regression further makes sure to reduce the variance bound as our formulation shows. Both the error map and the relMSE values reported in fig. 4.13 confirm our observation.

In summary, due to the lack of robustness during the quadrature construction caused by their fixed sample locations and error heuristics, Crespo et al.'s method does not consistently outperform the classical MC overall. In the case of inaccurate subdivisions of the primary sample space, their method might result in inefficient importance sampling, leading to higher errors than MC. By contrast, our method is provably never worse than the MC estimator despite that ours could result in rather coarse approximations of the function, limiting the error reduction in some regions. As future work, Crespo et al.'s adaptive model for g can potentially be useful in our formulation, but developing a practical least-squares regression algorithm for piecewise polynomials remains challenging.

4.3.5 Ablation Experiments

Choice Of Basis Functions We perform an experiment using different basis functions in the regression. The results are shown in fig. 4.14. We compare the performance of order 2 polynomial, step functions, Gaussian mixture and Sine functions. All these basis have been chosen to contain the same number of parameters for a fair comparison. The quality of the results obtained slightly varies according to the bases used and the scenes chosen. Other results for additional bases and scenes is available in the supplemental materials.

In general, while it is challenging to define a function basis robust for all situations, our experiments show that polynomial basis works relatively well in many cases. As long as a constant function is included in least-squares regression, our method would be no worse than Monte Carlo integration, no matter, what basis functions are chosen.

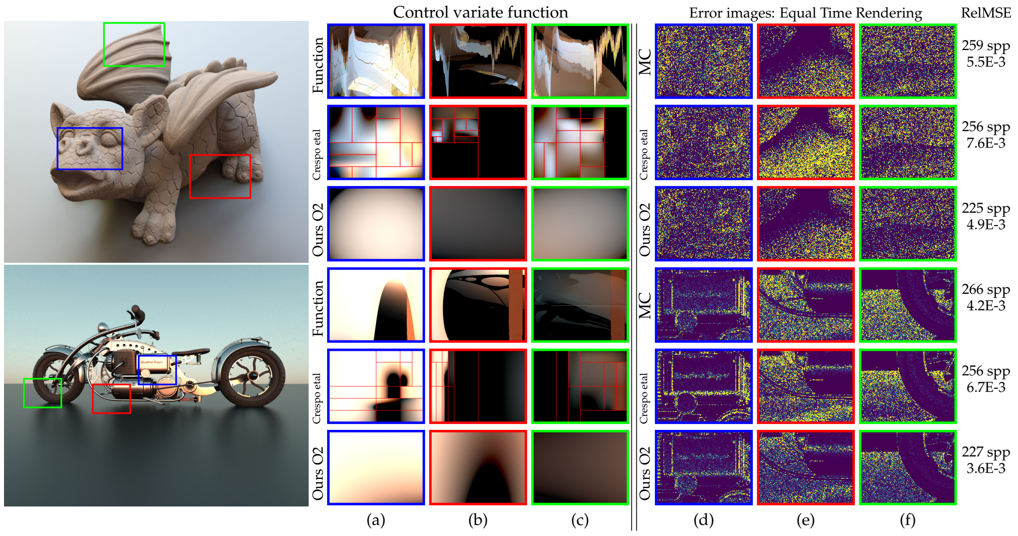


Figure 4.10: Equal-time comparison between our method and Crespo et al. (2021) where each pixel is considered independently. Columns (a)-(c) show the control variate function for a pixel centered at each inset. Columns (d)-(f) show the error images for each inset. Following Crespo et al., we ensure 30% of the samples are used for building the control variate. For both the DRAGON and the CHOPPER-TITAN scene, our method show relatively less error ((d)-(f)). Crespo et al. is efficient in some cases. For DRAGON scene, column (e), Crespo et al. gives worse error than the classical MC. The relMSE value of our method is lower than Crespo et al..

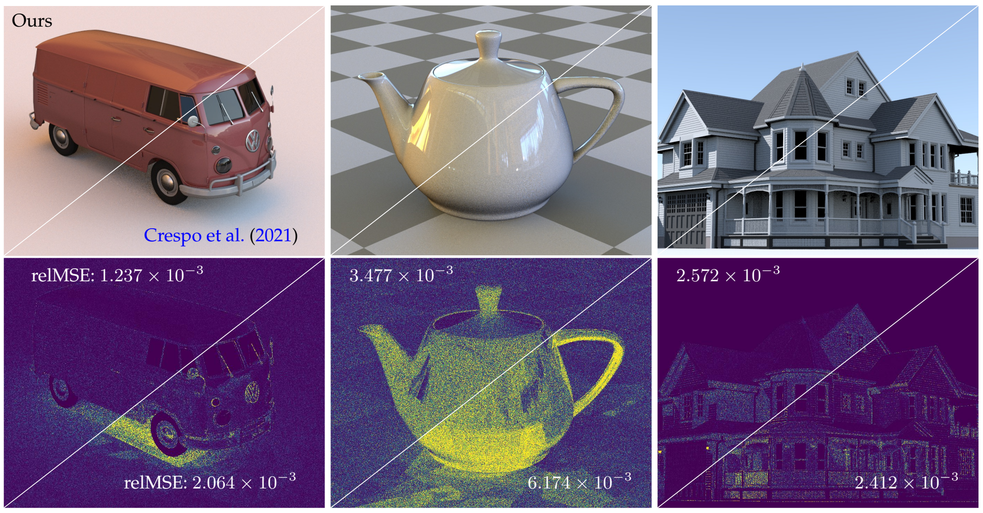


Figure 4.11: Equal-sample comparison of our method using an Order 2 polynomial function basis with Crespo et al. (2021) method.

Discontinuities impact on polynomial regression In a rendering scene, multiple light sources can be present. Typically, these multiple light sources are represented in an additional discrete dimension within the integration domain. This behavior is illustrated in fig. 4.15(b), where each slice represents the integration of one light over the surface.

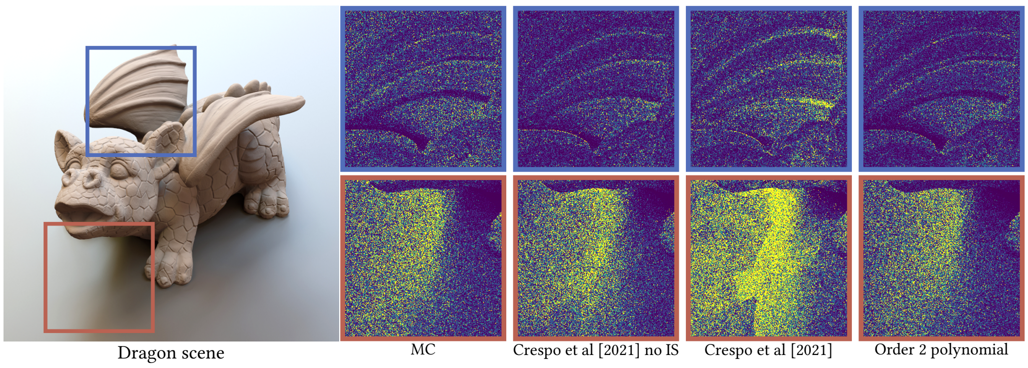


Figure 4.12: Importance sampling (IS) performed in Crespo et al.'s approach can increase the variance wrt classical MC due to bad IS strategy.

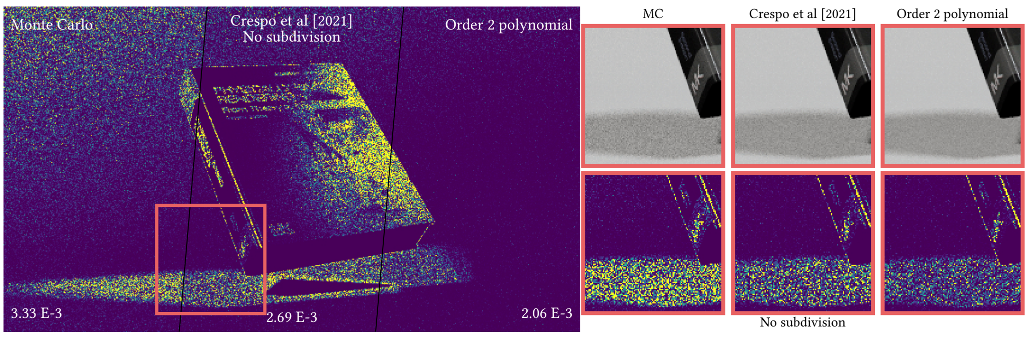


Figure 4.13: Without recursive space subdivision, Crespo et al. appears to be similar to our approach, but it is not. Our approach still performs better. All methods are rendered with 64 spp. relMSE values are at the bottom.

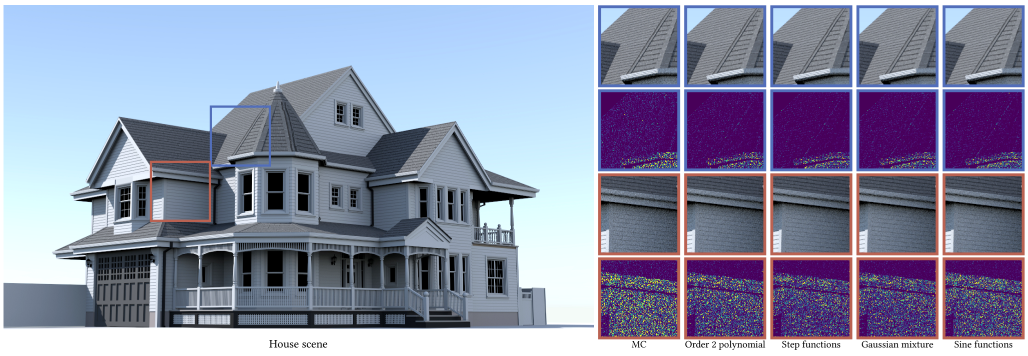
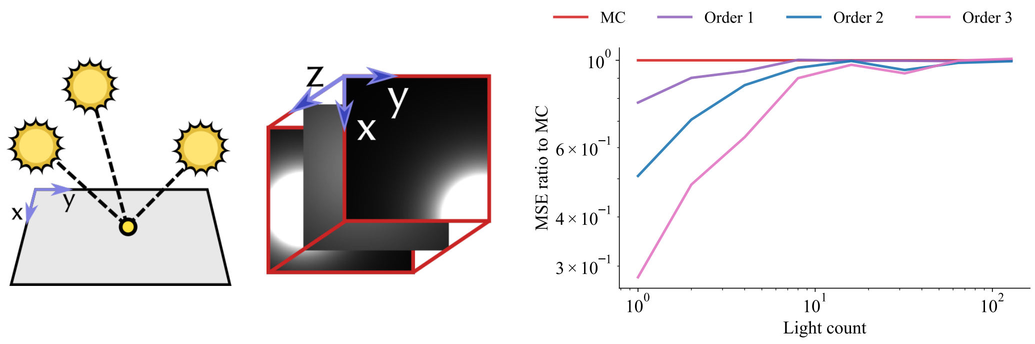


Figure 4.14: Comparison of our method using four different function bases against classical Monte Carlo. At equal sample count, all function bases lead to improved estimation quality over the standard Monte Carlo estimator. The visual results highlight how basis selection impacts reconstruction quality, yet consistently outperforms pure Monte Carlo sampling.

Polynomial bases often struggle to accurately represent functions with numerous discontinuities. In fig. 4.15, we examine the evolution of integration error with a fixed sample



a) Rendering setup b) Integration space c) Relative error to Monte Carlo estimator

Figure 4.15: Direct lighting integration with increasing light count and varying polynomial orders. This figure compares our method to classical Monte Carlo at equal sample count, showing how the effectiveness of our polynomial-based estimator diminishes as the number of lights increases. While higher-order polynomials initially offer strong improvements, the presence of many light sources introduces sharp discontinuities in the integrand that are challenging to capture with smooth polynomial regression. As a result, the benefit over Monte Carlo decreases with light count due to reduced regression accuracy.

budget of 1024 samples while integrating the contribution of multiple light sources over a diffuse plane (fig. 4.15(a)). Here, the number of discontinuities within the 3D integrand increases as the number of lights grows, owing to the random decision process for light selection (fig. 4.15(b)). We investigate the impact of employing Order 1, Order 2, and Order 3 polynomials with our method compared to the conventional Monte Carlo estimator. The graph in fig. 4.15(c) illustrates that with a small number of lights, our method notably outperforms the conventional Monte Carlo estimation. However, this advantage diminishes as the number of discontinuities increases. With a high number of lights, the error generated by our method becomes comparable to that of the Monte Carlo estimator, indicating that the polynomial basis behaves similarly to a constant model function, namely the conventional approach.

Computation Time We empirically found that the computation overhead of our method becomes significant for higher-order polynomial regression using the direct matrix approach. For example, our method with low-order polynomials has negligible overhead over Monte Carlo integration (fig. 4.4) but a polynomial of order 5 can cause 68% overhead in terms of running time. Similarly, higher-dimensional integration also requires significant overhead over Monte Carlo integration. However, we focus in demonstrating the versatility of our approach. Improving the regression computation is out of the scope for this work. Following above observation, we also investigate another approach for regression, such as gradient descent, which can be a better option for higher-order polynomials. The gradient descent approach scales well thanks to its lower computational complexity ($O(M)$ vs $O(M^2)$) where M is the number of monomials.

Matrix vs. Gradient Descent Solver Figure 4.16 shows the comparison between the gradient descent and the direct-matrix solver. The results are demonstrated with 32spp and 256spp on a BATHROOM scene taken from PBRT. We use three different polynomials of order 2, 5 and 7. We compare the direct matrix approach based on complete orthogonal decomposition and stochastic gradient descent estimator (SGD) to solve the polynomial regression. At a low sample count, SGD performs better than the direct matrix approach as the matrix tends to be ill-conditioned due to noise in sampling. At a high sample count, the direct matrix approach outperforms SGD due to its ability to find a global minimum. For a high-order polynomial (order 7), the direct matrix approach becomes computationally expensive whereas SGD runs at at least $2\times$ faster speed while giving similar relMSE.

Combination With Russian Roulette In our current implementation, we manually set all path lengths to be the same. This is because, for regression it is necessary to fix the dimensionality of the search space. Russian roulette, however may result in paths of different length/dimensionality.

One simple way to combine Russian roulette with our approach would be to perform regression up to a certain prescribed dimension and only invoke Russian roulette for paths that have dimensionality higher than the prescribed one. The high-dimensional sample components can then be projected back into the lower-dimensional space.

4.3.6 Discussions

Incremental Estimator In our implementation, we assume that the N sample estimator $\langle I_{CV}\rangle$ is constructed by first finding the solution to regression g using all the N samples, and then evaluating $\langle I_{CV}\rangle$ with the same N samples. Therefore, when N is changed to $N+1$, we need to solve regression again with $N+1$ samples first and then re-evaluate $\langle I_{CV}\rangle$ with the new g using $N+1$ samples. In some applications, such as progressive rendering, being able to incrementally update both $\langle I_{CV}\rangle$ and g as we add a new sample is desirable. Thanks to the use of control variates, our formulation allows such incremental update as follows.

Let us denote $\langle I_{CV}\rangle_N$ and g_N as the estimated value and the solution to regression with N samples. Using an online regression algorithm, one can obtain g_{N+1} based on g_N by adding a new sample u_{N+1}. We define the incremental estimate $\langle I_{CV}\rangle_{N+1}$ as

$$\langle I_{CV}\rangle_{N+1} = \frac{1}{N+1}\left(N\langle I_{CV}\rangle_N + (G_N + f(u_{N+1}) - g_N(u_{N+1}))\right) \tag{4.30}$$

where G_{N+1} is the integral of g_{N+1}. This estimator remains unbiased since

$$\begin{aligned}
\mathrm{E}\left[\langle I_{CV}\rangle_{N+1}\right] &= \mathrm{E}\left[\frac{1}{N+1}\left(N\langle I_{CV}\rangle_N + (G_N + f(u_{N+1}) - g_N(u_{N+1}))\right)\right] \\
&= \frac{1}{N+1}\left(NF + \mathrm{E}\left[(G_N + f(u_{N+1}) - g_N(u_{N+1}))\right]\right) \\
&= \frac{1}{N+1}\left(NF + F\right) = F.
\end{aligned} \tag{4.31}$$

The expected squared error of incremental estimators is likely worse than that of non-incremental counterparts; the N sample incremental estimator now combines the results of regression for all the $N-1, N-2, ..., 1$ samples which might be numerically unstable at the beginning. The incremental estimators are, however, often computationally less expensive for the same number of samples than the non-incremental counterparts. We leave the exact analysis of these differences as future work, but we show a preliminary

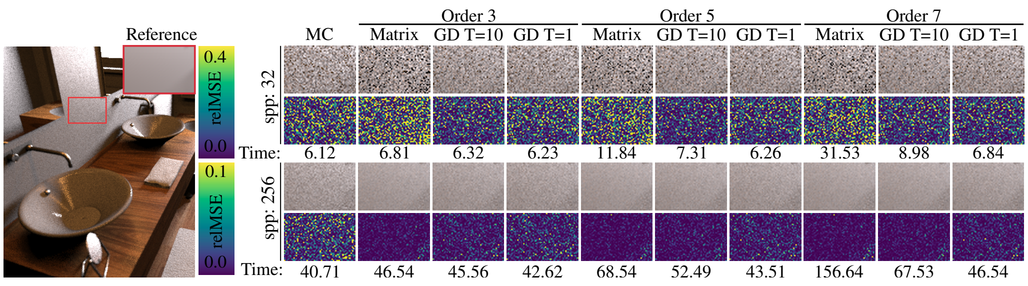


Figure 4.16: Regression using a direct matrix approach and stochastic gradient descent (SGD). At a low-sample count, SGD performs better. At a high-sample count, the direct matrix approach outperforms SGD. At high-order polynomial (order 7), the direct matrix approach has a very high overhead, making SGD an attractive alternative. Time is reported in seconds.

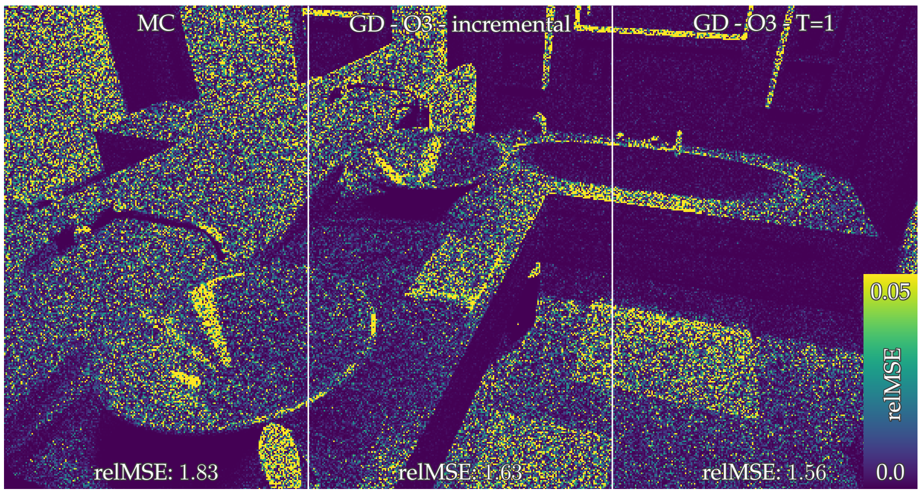


Figure 4.17: Comparison of our incremental regression-based Monte Carlo integration with the gradient descent estimator and the classical Monte Carlo estimator. We render the Bathroom scene (256 spp) and compare the error map for each method. While the incremental gradient descent has slightly higher error than its non-incremental counterpart, both achieves less error than MC integration. relMSE values are scaled by 10^{-2}.

result in fig. 4.17 where the incremental estimate is performed with the gradient descent estimator. The numerical error of the incremental estimate is slightly higher than the non-incremental version, but it still outperforms Monte Carlo integration.

Multi-Pixel Regression In the presented experiments, regression is performed on a per-pixel basis to ensure that the results are never worse than those obtained using standard Monte Carlo methods. To achieve this, each pixel is treated independently.

However, in practice, neighboring pixels in space and time often solve very similar integrals. One way to leverage this similarity is to reuse samples from adjacent pixels, which could help reduce variance during the regression step. Another approach is to perform regression for groups of pixels rather than for individual pixels, thereby reducing computational overhead by sharing the workload among multiple pixels. Additionally, the model function can be made adaptive during the rendering of a sequence of images by using data from previous frames to tailor the model function.

This adaptability could be particularly beneficial for handling discontinuities in the function. While this adaptive approach was not developed in this work to preserve the mathematical properties of the improved estimator, it would be important for more practical implementations.

Improved Sampling In this section, all presented equations and theoretical derivations are based on the assumption of independent and identically distributed sampling. While this provides a robust foundation for theoretical analysis, practical implementations often benefit from improved sampling techniques, such as those discussed in section 2.2.3. These methods distribute samples more evenly across the domain. This even distribution offers advantages in two key areas: residual integration and regression. For residual integration, improved sampling directly enhances the accuracy of the Monte Carlo estimator used to compute the difference between functions. In the regression step, it prevents the clustering of samples, which can otherwise lead to certain regions of the function being over-represented or disproportionately weighted. Unfortunately, incorporating such improved sampling methods typically breaks the theoretical guarantees underlying our derivations. Therefore, despite their clear empirical benefits, we haven't extensively explored them within this theoretical framework.

Differences From MCLS Our regression-based Monte Carlo integration bears some similarities to Monte Carlo Least-Squares MCLS) (Nakatsukasa, 2018). They reformulated Monte Carlo integration as least-squares regression followed by the analytical integration of the resulting function. MCLS also performs no worse than Monte Carlo integration as long as a constant function is included in regression. There are two major differences between MCLS and our approach. Firstly, MCLS considers a model function g which integrates exactly to F (i.e., $F = G$). Such a family of model functions is far more restrictive than our model function which is allowed to integrate to *any* value. Secondly, Nakatsukasa claims that MCLS is derived under an entirely different framework than control variates, thus its connection to control variates is unclear. In contrast, we show how control variates is closely related to the combination of Monte Carlo integration and least-squares regression. When we use a model function g such that $G = F$ (i.e., keeps the integral value), then our approach in fact reduces to MCLS. MCLS is thus can be seen as a special case of our formulation.

4.3.7 Limitations And Future Work

While our method often just works as a drop-in improvement over Monte Carlo integration in many cases, there are several limitations and future work.

Regression in our method incurs additional computation over MC integration. Regression may or may not be worth the additional cost depending on the amount of error reduction in the end. In general, regression becomes computationally more costly for a more complex model function, but a complex function typically leads to better error reduction. There is a trade-off among the reduction of error, the complexity of a model function, and the computational overhead of the regression process. Finding an optimal balance among those can potentially be achieved in an adaptive manner. Our formulation does not specify how regression should be solved in practice, so there is a large room of design choices in its implementation.

Since our formulation is based on the primary sample space, applying our method to a higher number of bounces in light transport simulation is challenging. The dimensionality of the primary sample space is directly proportional to the number of bounces, and regression in a high dimensional is not numerically stable. It is being recognized that light transport simulation has a much lower dimensional structure in the original path space, as was utilized by path guiding methods Vorba et al. (2014); Müller et al. (2017). It would be interesting to study how our regression-based formulation would benefit from such a low dimensional structure of light transport.

Depending on how the regression is implemented, our method can have a very small bias. When the same set of samples are used for both regression of the control variate and estimation of the difference, our estimator can be biased. Owen and Zhou (2000a) and Hickernell et al. (2005) have analyzed the amount of bias introduced when estimating the α parameter and evaluating the difference estimator using the same set of samples in control variates. In brief, the bias is negligible compared to the variance of the estimator and reduces at the order of $O(N^{-1})$ for N samples as opposed to $O(N^{-1/2})$ of the error of Monte Carlo integration. This makes our estimator consistent and the bias reduces faster than variance. One can also always remove this bias by separating a set of samples into two halves, performing regression only on the other half, and taking the average of the two difference estimators, in a manner akin to cross-validation. We, however, have not used this approach in the results in this paper since we have observed that bias is negligible. Our analysis is not influenced by this bias since this bias happens only when the same set of samples are used in its implementation of regression, while our analysis does not assume any specific implementation of regression.

4.4 Conclusion

In this chapter, we developed a novel control variate estimator leveraging least-squares regression. This new estimator consistently demonstrated improved integration error compared to the classical Monte Carlo estimator for physically based rendering. It advances the state of the art by providing a more robust strategy for adaptive control variates. When the regression accurately approximates the integrated function, it offers substantial error reduction. While primarily demonstrated with polynomial regression, we also showed that employing other function classes could lead to even greater improvements in integration error if adapted to the integrated function.

Chapter 5
Multi-Class Sampling For Monte Carlo Integration

Sampling is a fundamental technique in computer graphics, applied across various tasks to achieve high-quality results in rendering (Section 2.2.3), object placement, and visualization. Monte Carlo integration is a particularly prevalent method that uses random sampling to numerically estimate integrals, essential in producing realistic images by simulating complex light interactions. By averaging the contributions of many random samples, Monte Carlo methods help achieve accurate and noise-reduced images, although the process can be computationally expensive and require sophisticated strategies to manage sample distribution effectively.

In the realm of computer graphics, multi-class sampling has emerged as a method for addressing tasks that demand the simultaneous satisfaction of multiple objectives (Section 3.2). Applications such as object placement in scenes Wei (2010), visualization of multi-dimensional data Hu et al. (2020); Onzenoodt et al. (2021), and image color stippling Secord (2002); Schulz et al. (2021) benefit significantly from multi-class sampling. The primary goal of multi-class sampling is to produce a point set that fulfills several objective simultaneously. An objective is to optimize a specific subset of points to follow a given target distribution. When the subsets are mutually disjoint, the task is relatively easy since each objective can be optimized separately. The difficulty arises in applications where the subsets overlap. Overlaps introduce conflicts between optimization objectives. A classical example is multi-tone image stippling, where individual color channels and their union(s) all have different target densities (An example is visible in fig. 5.1). Such problems call for formulating a global optimization problem that can encode all objectives (15 in the case of color stippling with 4 points color) with a desired balance between them.

However, existing multi-class sampling methods, such as those proposed by Wei (2010); Jiang et al. (2015); Qin et al. (2017), face significant challenges when scaling to a large number of objectives. These methods often struggle with the complexity of specifying many objectives and optimizing them within a reasonable time frame and memory foot-

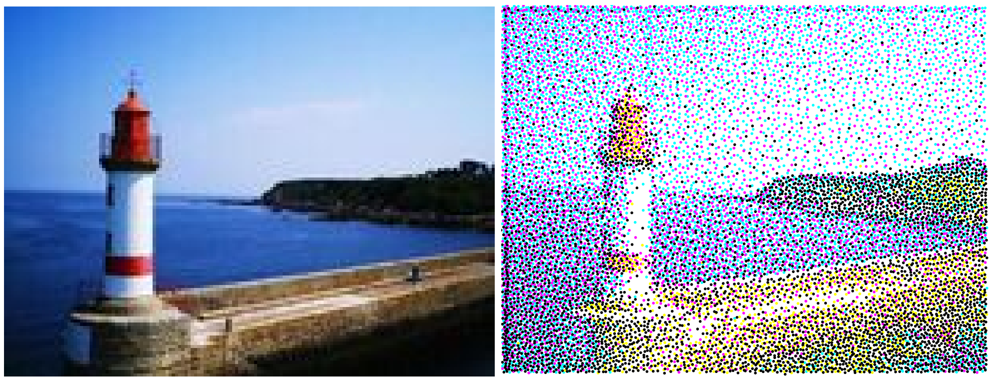


Figure 5.1: Example of Multi-class sampling application for color image stippling using CMYK (Cyan, Magenta, Yellow and black) points. The image is represented with 16000 dots.

print. The need for efficient and scalable solutions in scenarios with numerous objectives remains an open problem.

To address these challenges, this work proposes a novel formulation based on continuous Wasserstein barycenters. This approach enhances scalability by providing a straightforward way to specify multiple objectives simultaneously and balance them according to desired weights. The formulation leverages the properties of Wasserstein barycenters, which represent average distributions in a manner that minimizes transportation cost. Additionally, this method includes a gradient-descent optimization scheme that is robust to the number of objectives, ensuring efficient and scalable performance.

The effectiveness of this framework has been demonstrated in diverse applications involving numerous objectives, such as color stippling, object placement, and progressive sampling for Monte Carlo integration. These applications highlight the framework's ability to manage complex task. For instance, in progressive sampling, the method is able to generate good integration properties even with thousands of conflicting objectives created by the sample set progressiveness.

A notable application of this work is in formulating perceptual error distribution for Monte Carlo rendering (Section 3.3) as a multi-class problem. This involves deriving a perceptual error bound that depends on the sample set used during rendering, optimizing it before the rendering process to achieve perceptually pleasing error distributions. This approach marks the first theoretical formulation where the noise distribution in screen space results from minimizing perceptual error rather than being an ad-hoc goal. The method demonstrates improved results over state-of-the-art a priori error distribution techniques.

5.1 Preliminaries

This section explore the essential mathematical tools employed in this project. We will begin with a review of measure theory, then delve into the concepts of optimal transport between measures, and finally, we will discuss the Wasserstein barycenter.

Measure theory Measure theory provides a rigorous mathematical framework for handling the concept of size or measure of sets, which is essential for understanding probability, integration, and many other areas of mathematics. In the context of probability, a measure is often referred to as a probability measure, which assigns a probability to subsets of a given space $\mathcal{X}$.

A probability measure is a real-valued function ν defined on a space $\mathcal{X}$ and satisfies certain axioms. Formally, a probability measure ν on a σ-algebra $\mathcal{F}$ of subsets of $\mathcal{X}$ is a function $\nu : \mathcal{F} \to \mathbb{R}^+$ such that:

- $\nu(A) \geq 0$ for all $A \in \mathcal{F}$
- $\nu(\emptyset) = 0$
- For any collection $\{A_i\}$ of M disjoint sets in $\mathcal{F}$, $\nu\left(\bigcup_{i=1}^M A_i\right) = \sum_{i=1}^M \nu\left(A_i\right)$
- $\nu(\mathcal{X}) = 1$

A probability space $(\mathcal{X}, \mathcal{F}, \nu)$ consists of a sample space $\mathcal{X}$, a σ-algebra $\mathcal{F}$ an $\mathcal{X}$ and a probability measure ν.

In this framework, the integral of a function $f : \mathcal{X} \to \mathbb{R}$ with respect to a probability measure ν is defined, extending the classical notions of Riemann and Lebesgue integrals. The integral of f over a set $A \in \mathcal{F}$ is given by:

$$\int_A f \mathrm{d}\nu = \int_A f(x)\nu(\mathrm{d}x) \tag{5.1}$$

This integral allows to define concepts such as expectation in probability theory. The expectation $\mathrm{E}\left[f\right]$ of a function f on a probability space $(\mathcal{X}, \mathcal{F}, \nu)$ is:

$$\mathrm{E}\left[f\right] = \int_A f(x)\nu(\mathrm{d}x) \tag{5.2}$$

Optimal Transport Building on this foundation, optimal transport theory leverages the concepts of probability measures and integration to address the problem of transforming one distribution into another with minimal cost (Villani, 2008; Santambrogio, 2015; Peyré and Cuturi, 2018). The primary objective is to determine the least expensive way to move mass from one distribution ν to a target distribution μ with a minimal cost C. This minimal cost provides a notion of distance between two probability distributions, formalized through the optimal transport cost function.

Formally, let ν and μ e probability measures on measurable spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively. The cost of transporting a unit of mass from a point $x \in \mathcal{X}$ to a point $y \in \mathcal{Y}$ is given by the cost function $\mathrm{c}(x, y)$. The goal is to find a transport plan γ that minimizes the total transport cost:

$$\mathrm{C} = \inf_{\gamma \in \Gamma(\nu,\mu)} \int_{\mathcal{X}\times\mathcal{Y}} \mathrm{c}(x, y)\mathrm{d}\gamma(x, y). \tag{5.3}$$

$\Gamma(\nu, \mu)$ denotes the set of all couplings (transport plans) between ν and μ. A coupling γ is a measure on $\mathcal{X} \times \mathcal{Y}$ with marginals ν and μ.

$$\gamma(A \times \mathcal{Y}) = \nu(A) \text{ and } \gamma(\mathcal{X} \times B) = \mu(B) \quad \forall A \subseteq \mathcal{X}, B \subseteq \mathcal{Y} \tag{5.4}$$

In the classic analogy of piles of sand, this problem translates to finding the minimum amount of work needed to reshape one pile into another, where the work is quantified as the total mass displaced per unit distance. This minimum cost, known as the optimal transport cost, provides a meaningful notion of distance between two distributions.

Wasserstein Distance The Wasserstein distance Ollivier et al. (2014), also known as the Earth Mover's Distance, is a concept from optimal transport theory that quantifies the minimal cost of transforming one probability distribution into another. This distance metric is particularly useful in various applications, such as image processing, machine learning, and economics, where it is essential to measure the dissimilarity between distributions.

Formally, the Wasserstein distance of order p between two probability measures ν and μ on a metric space Ω is defined as:

$$W_p(\nu, \mu) = \left(\inf_{\gamma \in \Gamma(\nu,\mu)} \int_{\Omega^2} \|x - y\|^p \, \mathrm{d}\gamma(x, y) \right)^{1/p}. \tag{5.5}$$

In this context, $\|x - y\|$ represents the Euclidean distance within the domain Ω. Conceptually, γ is a transport plan (technically, a joint measure with marginals μ and ν) such that $\mathrm{d}\gamma(x, y)$ indicates the (differential) quantity of mass to be moved between any two points x and y. The associated cost is therefore $\|x - y\|^p \mathrm{d}\gamma(x, y)$. Among all such plans in the space $\Gamma(\nu, \mu)$, we seek the one that minimizes the transportation cost across all point pairs, i.e., the integral in eq. (5.5).

One of the significant advantages of the Wasserstein distance over other distance metrics, such as the Kullback-Leibler divergence or the total variation distance, is its sensitivity to the underlying geometry of the space Ω. This means that the Wasserstein distance takes into account the actual locations of the distributions' mass, rather than just their values at specific points, making it a more robust measure in contexts where the spatial arrangement is crucial.

Sliced Wasserstein Distance The Sliced Wasserstein distance (Pitié et al., 2005; Rabin et al., 2011; Bonneel et al., 2015) is an extension of the Wasserstein distance designed to overcome computational challenges associated with high-dimensional probability distributions. While the Wasserstein distance provides a robust measure of dissimilarity by accounting for the geometry of the distributions, its computation can be intractable for large-scale or high-dimensional data. The Sliced Wasserstein distance mitigates this issue by projecting the high-dimensional distributions onto lower-dimensional subspaces and then computing the Wasserstein distance on these lower-dimensional projections.

Formally, let ν and μ be two probability measures on a high-dimensional space $\mathbb{R}^d$. The Sliced Wasserstein distance of order p between ν and μ is defined as:

$$SW_p(\nu, \mu) = \left(\int_{\mathbb{S}^{d-1}} W_p^p\left(\nu^\theta, \mu^\theta\right) \mathrm{d}\theta \right)^{1/p} \tag{5.6}$$

Within the context of the sliced Wasserstein distance, $\theta \in \mathbb{S}^{d-1}$ represents a point on the $(d-1)$-dimensional sphere. The measures ν^θ and μ^θ are the orthogonal projections of the original measures onto the line defined by θ.

The computation of the Sliced Wasserstein distance involves the following steps:

- **Projection**: Project the high-dimensional probability measures ν and μ onto a one-dimensional subspace defined by θ.
- **Wasserstein Distance computation**: Compute the Wasserstein distance between the one-dimensional projected distributions.
- **Integration**: Integrate these distances over all possible directions θ on the unit sphere.

One of the key advantages of the Sliced Wasserstein distance is that the Wasserstein distance in one dimension can be computed efficiently using the sorting-based method. For one-dimensional discreet distributions, the Wasserstein distance W_p has a closed-form solution involving the sorted samples of the distributions. Specifically, for empirical distributions, it can be computed as:

$$W_p\left(\nu^\theta, \mu^\theta\right) = \left(\frac{1}{N} \sum_{i=1}^{N} |x_i - y_i|^p \right)^{1/p} \tag{5.7}$$

where x_i and y_i are the sorted samples of the projections of ν and μ, respectively. Furthermore, the Sliced Wasserstein distance can be used to construct an upper and a lower bound for the Wasserstein distance, establishing a theoretical connection between these two distances.

Wasserstein Barycenter The concept of the Wasserstein barycenter (Agueh and Carlier, 2011; Rabin et al., 2011) extends the idea of the Wasserstein distance to the problem of finding a central or average measure among a collection of probability distributions. Given a set of probability measures $\{\mu_i\}_{i=1}^N$, the Wasserstein barycenter ν is defined as the probability measure that minimizes the weighted sum of the Wasserstein distances to the given measures. Formally, for a set of weights $\{\lambda_i\}_{i=1}^N$ such that $\sum_{i=1}^N \lambda_i = 1$ and

$\lambda_i \geq 0$, the Wasserstein barycenter ν solves the following optimization problem:

$$\nu = \arg\min_{\nu} \sum_i \lambda_i W_p^p(\nu, \mu_i), \tag{5.8}$$

The Wasserstein barycenter can be interpreted as the "weighted average" distribution that best represents the given set of distributions under the Wasserstein metric. It is important to note that the Wasserstein barycenter result in a distribution minimizing the sum of Wasserstein distances and not a simple mixture of densities.

The Wasserstein barycenter serves as a method to balance multiple objectives, aiming to find a distribution that reduces the (weighted) distance to several targets. By substituting W_p with its sliced version SW_p, the barycenter can be practically computed through successive 1D optimizations (Rabin et al., 2011; Bonneel et al., 2015).

5.2 Related Work

5.2.1 Blue Noise Sampling

Blue Noise Blue noise is a important concept in computer graphics characterized by it's spectral properties. Specifically, the presence of high-frequency components and the absence of low-frequency components. This high-frequency attribute ensures an even distribution and the absence noticeable patterns or clusters, which is particularly desirable in various graphics applications. Blue noise has proven to be beneficial in enhancing visual quality, especially in image dithering.

In image dithering, blue noise dithering masks are employed to convert images into binary colors (black and white) while preserving the perception of detail and gradients. This high visual quality is achieved by leveraging the human visual system's tendency to be less sensitive to high frequencies compared to low frequencies. As a result, blue noise masks can produce images that retain the essential details and subtleties of the original image, even when limited to binary colors. Figure 5.2 illustrates image dithering using three different dithering masks: constant, random, and blue noise. The constant dithering loses most of the image content by binarizing the gray levels to the nearest value (0 or 255). The random dithering, while preserving more structure and visually appearing to have gray levels, introduces significant noise due to its stochastic nature. Blue noise dithering improves upon this by correlating different pixels and reducing the visibility of noise, resulting in a visually smoother image.

Multi-Class Sampling Multi-class blue noise sampling extends the concept of blue noise sampling to scenarios where multiple types or classes of samples must be considered simultaneously, with individual samples potentially belonging to multiple classes. In traditional blue noise sampling, the goal is to distribute samples within a single class as evenly as possible, avoiding clustering while maintaining randomness. However, many applications, such as rendering, geometric optimization, or object placement, require managing multiple classes of samples with distinct characteristics or properties. The primary challenge in multi-class blue noise sampling is to achieve a well-distributed pattern for each class while allowing for overlap between classes, ensuring that each

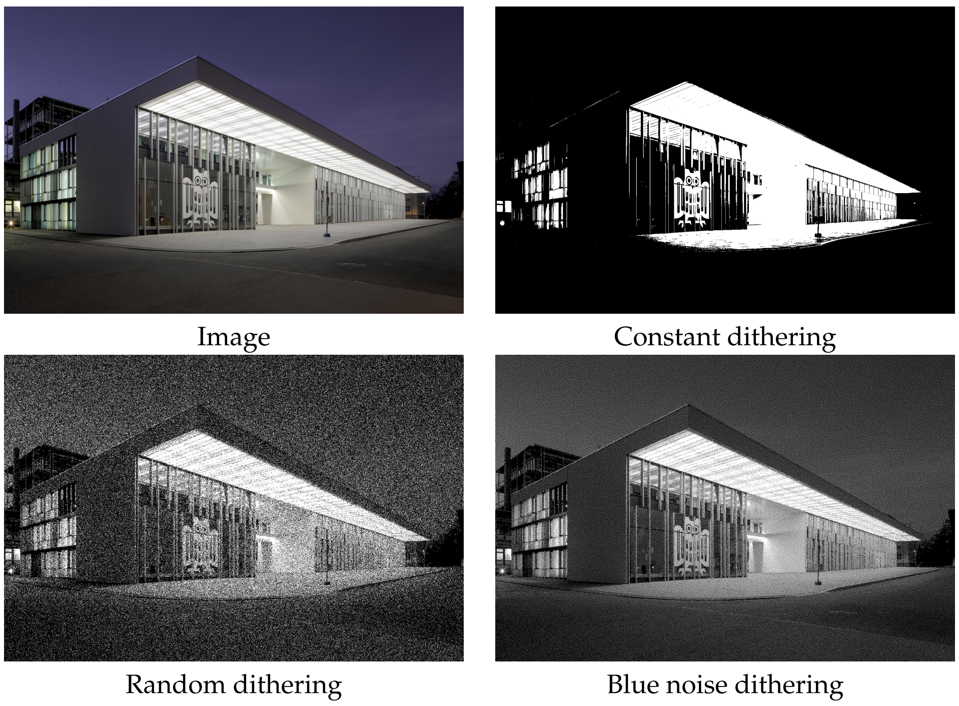


Figure 5.2: Example of image dithering using three different masks: constant, random, and blue noise. The constant dithering significantly loses image content by binarizing the gray levels to the nearest value (0 or 255). Random dithering preserves more structure but introduces noticeable noise. Blue noise dithering reduces the visibility of noise, leading to a visually smoother image by correlating different pixels.

class maintains its blue noise properties and contributes to high-quality sampling. Considering only the spatial locations of the samples could severely limit their applicability in real-world scenarios. Multi-class sampling allows incorporating non-spatial features while maintaining the blue noise property of the spatial coordinates.

Wang and Parker (1999) first demonstrated the impact of multi-class sampling on colored halftoning of images, developing a sampling algorithm that generates blue-noise quality in combinations of the R, G, and B channels. Wei (2010) proposed a dart-throwing-based method to produce multi-class sample sets, which was later improved by Qin et al. (2017) using optimal transport optimization. Schmaltz et al. (2012) introduced electrostatic halftoning, while Jiang et al. (2015) employed an SPH method to obtain multi-class samples. These methods enforce multi-class blue noise through an interaction matrix encoding the spacing between class pairs, though the matrix can exhibit discontinuous changes in the off-diagonal entries representing the coupling between different classes' distributions. Chen et al. (2012) proposed a two-step algorithm based on capacity-constrained Voronoi tessellation to achieve a multi-class property, first optimizing each class individually, then optimizing their unions. Chen et al. (2013) introduced a continuous multi-class sampling scheme limited to dart throwing and kernel-based optimization.

Qin et al. (2017) overcame these limitations via a multi-class framework based on optimal transport (Rabin et al., 2011; Rachev and Rüschendorf, 1998; Agueh and Carlier, 2011).

By optimizing for the Wasserstein barycenter of target distributions for classes and their unions, they obtained a multi-class blue-noise point set. To handle conflicts between classes and avoid regularity, they leveraged entropic regularization Cuturi (2013). While effective, this method lacks the flexibility to specify multiple objectives and does not scale well with the number of objectives. We propose a new formulation of multi-class sampling using sliced optimal transport, which overcomes these limitations and generalizes multi-class sampling to different applications.

5.2.2 Correlated Sampling For Monte Carlo Integration

Low Discrepancy Sampling Low discrepancy sampling is a fundamental technique in numerical integration, particularly within the context of Monte Carlo methods. As discussed in section 2.2.3, low discrepancy sequences aim to cover the integration domain as uniformly as possible, thereby minimizing the discrepancy, which measures deviation from perfect uniformity. This uniform coverage helps in reducing integration error by ensuring all regions of the domain are adequately sampled. Prominent examples of low discrepancy sequences include the Halton, Sobol, and Faure sequences, each with unique properties suitable for different integration tasks.

While low discrepancy sampling emphasizes uniform point coverage, blue noise sampling focuses on distributing points with specific spectral properties. Blue noise is characterized by its even distribution and lack of low-frequency components, which prevents clustering and maintains randomness on a finer scale. Both low discrepancy and blue noise sampling aim for even point distribution, but they differ in their metrics: low discrepancy measures uniformity using hyper-rectangles or other element shapes, whereas blue noise uses relative point distances and spectral properties. This distinction influences their respective applications.

In Monte Carlo integration, the Koksma–Hlawka inequality provides a theoretical link between integration error and sample point discrepancy. The inequality states that integration error is bounded by the product of the sample points' discrepancy and the integrand's variation in the Hardy-Krause sense. This underscores the importance of low discrepancy sequences in reducing integration error. Consequently, various sampling methods have been developed to achieve low discrepancy and enhance Monte Carlo integration accuracy.

Isotropic Sampler Isotropic sampling distinguishes itself from low discrepancy sampling by aiming for uniformity of distances in all directions. Low discrepancy methods, particularly those focusing on star discrepancy, use hyper-rectangles with fixed orientations to measure uniformity. This approach can lead to anisotropic properties, where distances in diagonal directions are less penalized compared to axis-aligned directions. Consequently, low discrepancy sampling may not provide uniform coverage in all directions, potentially leading to higher integration errors in some cases.

Isotropic sampling, such as blue noise, addresses the anisotropy of low discrepancy sampling by ensuring uniform distances in all directions. Blue noise sampling achieves this through its characteristic even distribution and lack of low-frequency components, maintaining randomness while avoiding clustering. This isotropic property is crucial for applications requiring uniform point distribution across all directions, such as rendering and spatial optimization.

Similar to the Koksma–Hlawka Inequality for low discrepancy sampling, Monte Carlo error can be bounded using isotropic measures like the Wasserstein distance or the Sliced Wasserstein distance. Considering a continuous function $f : \mathcal{H} \to \mathbb{R}^+$ on the hypercube $\mathcal{H}$ with Lipschitz constant L_f such that, $\forall x, y \in \mathcal{H}$,

$$|f(x) - f(y)| \leq L_f \|x - y\|. \tag{5.9}$$

Let $\gamma \in \Gamma(\nu, \mu)$ be a joint measure whose marginals ν and μ are measures on the unit hypercube $\mathcal{H}$. Integrating both sides of eq. (5.9) w.r.t. γ, and then using $\left|\int g\right| \leq \int |g|$, yields

$$\int_{\mathcal{H}^2} |f(x) - f(y)| \, \mathrm{d}\gamma(x, y) \leq L_f \int_{\mathcal{H}^2} \|x - y\| \, \mathrm{d}\gamma(x, y) \tag{5.10}$$

$$\left| \int_{\mathcal{H}^2} \left[f(x) - f(y)\right] \mathrm{d}\gamma(x, y) \right| \leq L_f \int_{\mathcal{H}^2} \|x - y\| \, \mathrm{d}\gamma(x, y). \tag{5.11}$$

Te left side of eq. (5.11) is expanded into two integrals and each is simplified by marginalizing the product measure; the bound on the right is tightened by taking the infimum over all valid joint measures γ:

$$\left| \int_{\mathcal{H}} f(x) \, \mathrm{d}\nu(x) - \int_{\mathcal{H}} f(x) \, \mathrm{d}\mu(x) \right| \leq L_f \underbrace{\inf_{\gamma \in \Gamma(\nu,\mu)} \int_{\mathcal{H}^2} \|x - y\| \, \mathrm{d}\gamma(x,y)}_{W(\nu,\mu)}, \tag{5.12}$$

where $W(\nu, \mu)$ is the Wasserstein distance between ν and μ. The resulting inequality provides a numerical integration bound when ν is a Dirac point-mass measure, i.e., a point set.

As the Sliced Wasserstein distance bound by the Wasserstein distance, it is possible to also bound Monte Carlo error with it as following:

$$\left| \int_{\mathcal{H}} f(x) \, \mathrm{d}\nu(x) - \int_{\mathcal{H}} f(x) \, \mathrm{d}\mu(x) \right| \leq L_f \, SW(\nu, \mu) \tag{5.13}$$

While similar in structure, the error bound using the Wasserstein distance remains technically distinct from the Koksma–Hlawka inequality. Both bounds consist of two terms: one capturing the variation of the integrand independently of the sampling distribution, and the other measuring the uniformity of the sampling distribution independently of the function. In the Koksma–Hlawka inequality, the first term involves the bounded variation of the function, while in the Wasserstein-based bound, it is represented by the Lipschitz constant. The uniformity of the sampling distribution in the Koksma–Hlawka bound is quantified by the discrepancy, whereas in the Wasserstein-based bound, it is measured using the Wasserstein distance. This distinction in metrics reflects the difference in approach, with the Wasserstein distance providing a more isotropic measure of sampling quality compared to the directional sensitivity of discrepancy.

Paulin et al. (2020) leveraged the bound provided by the Lipschitz inequality to optimize a sampler that reduces the error bound in Monte Carlo integration. This sampler achieves blue noise properties by ensuring uniformity in all directions, treating all directions with equal importance. By minimizing the Wasserstein distance between the empirical and target distributions, the sampler produces high-quality blue noise point sets, effectively reducing integration error.

5.2.3 Perceptual Error Distribution

Traditionally in Monte Carlo rendering, pixel values are estimated independently from one another, resulting in a white-noise distribution of error across the image. Mitchell (1991) observed that error distributions with high-frequency power spectra produce more visually pleasing noisy images. This insight leverages the characteristics of the human visual system, which is less sensitive to high-frequency noise, thereby making such distributions preferable in rendering applications.

The human visual system's sensitivity to different frequencies is described by the Contrast Sensitivity Function (CSF). The CSF indicates that humans are less sensitive to high-frequency noise compared to low-frequency noise. This can be attributed to the distribution of cones in the retina, which follows a blue noise pattern, providing a natural basis for preferring high-frequency noise in rendered images. This pattern helps the visual system to process and perceive images with greater clarity and detail, even when the image contains noise (Weier et al., 2017).

Error distribution methods in rendering can be categorized into two main types: a priori optimization and a posteriori optimization. A priori methods optimize the sampling distribution for Monte Carlo rendering without prior knowledge of the scene. This means that a single optimized sampling strategy can be applied across various scenes with no additional computational overhead during rendering. In contrast, a posteriori methods use information from the rendering process to tailor the sampling distribution to the specific scene, potentially achieving higher quality by incorporating scene-specific details.

A Priori Methods Georgiev and Fajardo (2016) introduced a dithering-inspired method to explicitly coordinate sampling across image pixels to achieve a blue-noise error distribution. This approach uses simulated annealing to optimize a shift vector for each pixel such that adjacent pixels have as different sampling patterns as possible. This shift is applied to a common stratified sampling sequence shared by all pixels, ensuring that the samples maintain high-frequency distribution properties. For smooth functions, this results in a rendering where adjacent pixels produce different outcomes, contributing to a high-frequency error distribution.

Heitz et al. (2019) and Belcour and Heitz (2021) demonstrated that optimizing the expected rendering error directly, rather than the sample positions, can enhance quality. They focused on a class of rendering functions (Heaviside functions) and used simulated annealing to optimize the rendering distribution for these functions. This method has two notable advantages: it can work with any low discrepancy sequence, ensuring it is never worse than classical Monte Carlo methods, and it performs optimization in the lower-dimensional result space rather than the high-dimensional sample space.

Later, Ahmed and Wonka (2020) proposed an optimization-free approach based on decomposing a single Sobol sequence over screen space using z-ordering. This method provides a simple and effective way to distribute samples, leveraging the properties of the Sobol sequence to achieve desirable noise characteristics.

Overall, these a priori methods aim to produce high-frequency rendering distributions. However, they do not directly link to human perception, relying instead on explicit target distributions.

The concept of perceptual optimization for a single frame has been extended to the temporal domain by Wolfe et al. (2022). Instead of optimizing individual frames independently, this work proposes optimizing the sequence in which samples are used across frames. Specifically, the method maximizes the per-pixel distance of samples in the temporal domain, in addition to the spatial 2D domain. This dual-domain optimization results in perceptually pleasing error distribution within each frame and significantly varied rendering at each pixel across frames, enhancing visual quality and stability in video rendering.

A Posteriori Methods Previously discussed methods optimize a single sample set independent of specific scenes, as the optimization is agnostic of the rendering content. In contrast, Heitz and Belcour (2019) introduced a method that incorporates pre-rendering of the scene. The core idea is to apply dithering sampling based on a local approximation of the pixel histogram of rendering estimates. This is accomplished by rearranging the positions of the sampling sequence in screen space between frames according to their rendering values, guided by a dither mask.

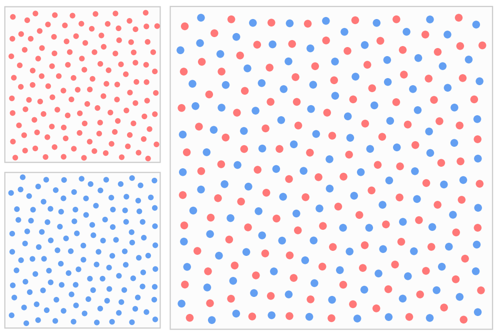
More recently, Chizhov et al. (2022) adopted theories from halftoning to formally address perceptual error in rendering. Even with a simple modeling of the HVS using a Gaussian, they managed to obtain a proper error distribution minimizing visual error. Building on their formulation, we derived a bound for perceptual error. This bound can be minimized by our optimization scheme to produce sample sets that yield blue-noise error distributions, aligning with the principles of human visual perception.

These advancements demonstrate that by leveraging perceptual considerations and sophisticated optimization techniques, both a priori and a posteriori methods can significantly enhance the visual quality of Monte Carlo rendering.

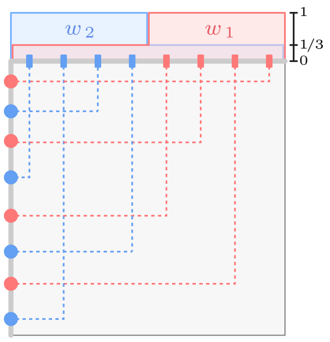
5.3 Multi-Class Optimal Transport

This section introduces the concepts of class, sub-class, and barycenter, fundamental to multi-class optimization. These definitions will form the basis for framing the multi-class problem as an optimization task with multiple Wasserstein distances.

Multi-class sampling involves generating a point set $X = \{x_i\}_{i=1}^{n} \subset \Omega$ where each subset of points is optimized according to a distinct objective. For example, in fig. 5.3a, the aim is to achieve a high-quality isotropic uniform distribution for each point color and their



(a) Classical 3-class optimization: red, blue, union



(b) Our representation

Figure 5.3: (a) A classical 3-class example, where the red and blue point subsets represent one uniform-density objective (class) each and their union is another objective, all three with equal optimization priority. (b) We represent this three-objective problem using two staircase functions, one per color, defined on an extra dimension (e.g., the point indices). The overlap between the functions implicitly specifies the third (union) objective.

union, resulting in three optimization objectives.

When the number of objectives is small, as in previous works (Wei, 2010; Jiang et al., 2015; Qin et al., 2017), it is feasible to specify each objective individually. However, this approach does not scale well when the number of objectives increases. Our goal is to manage a large number of objectives efficiently by providing a principled and convenient way to specify them in bulk and handle potential conflicts between them. Additionally, we seek a more abstract method for defining point subsets, allowing points to be grouped by various attributes beyond simple indices.

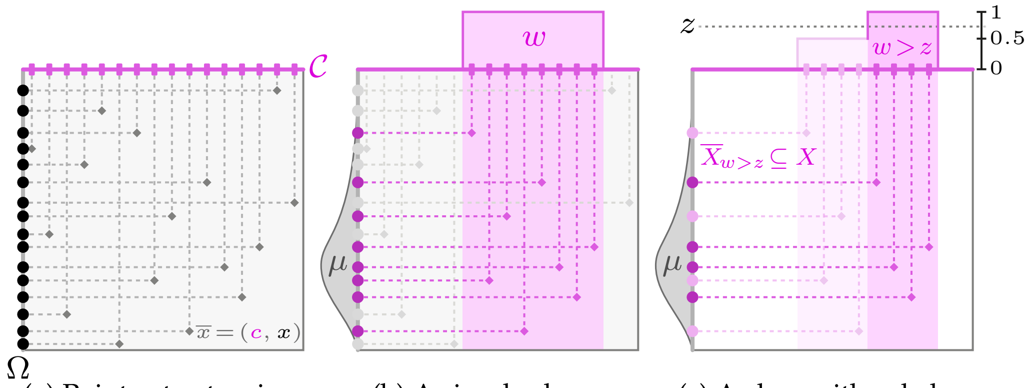
To address these challenges, we propose a novel formulation of the multi-class sampling problem based on optimal transport. This formulation enables us to achieve our objectives by operating in an extended space where an additional dimension is used for point classification while the remaining dimensions are optimized. As shown in fig. 5.3b, we represent the three optimization objectives in fig. 5.3a using two functions on this extra dimension. The following sections will introduce each component of this figure in detail.

5.3.1 Classes And Subclasses

A class is an optimization objective specified by subset of points and a target distribution. The subset is typically given as a range of indices. We begin by generalizing the index space. Specifically, we extend the optimization space Ω by a classification dimension $\mathcal{C}$. Given a point set X, a corresponding extended point set $\overline{X} = \{\overline{x}_i\}_{i=1}^{n}$ is created by distributing n class coordinates c_i uniformly in $\mathcal{C}$ and assigning them arbitrarily to the optimization points: $\overline{x}_i = (c_i, x_i)$.

Figure 5.4a illustrates this setup. The best choice for $\mathcal{C}$ depends on the application. $\mathcal{C}$ can be multi-dimensional but most often it will simply be the unit line, i.e., $\mathcal{C} = [0, 1]$, and the class coordinates will be the normalized indices in the base point set: $\overline{x}_i = (i/n, x_i)$. This normalization will allow us to define classes directly on the (fixed) extended space

$\mathcal{C} \times \Omega$, independently of (the size of) any particular point set. Sometimes a different class dimension is more natural, e.g., in rendering-error minimization $\mathcal{C}$ will be the 2D image plane (see section 5.6).



(a) Point-set extension (b) A simple class (c) A class with subclasses

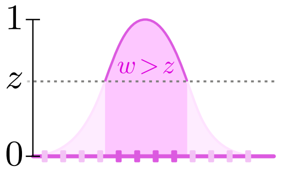
Figure 5.4: (a) We extend the dimension of a point set by assigning to each point x a unique class coordinate $c \in \mathcal{C}$ that remains fixed during optimization. This coordinate will often be the (normalized) index of the point. (b) A subregion in dimension $\mathcal{C}$ isolates a subset of points that can be optimized to follow a target distribution μ. The subregion can be given by the support of a function w on $\mathcal{C}$, here a box function. We define a class as a pair (w, μ). (c) A non-trivial class function yields multiple subclasses (sharing a target distribution) enumerated by slicing w. The staircase function here yields two subclasses, selecting all ($\overline{X}_{w>0}$) and half ($\overline{X}_{w>0.5}$) the class' points.

Classes The classification dimension $\mathcal{C}$ is not part of the optimization and is invariant to the number of points to be optimized. This allows to isolate point subsets by taking subregions of $\mathcal{C}$, as illustrated in fig. 5.4b. One way to specify a subregion is through the support of some function w on $\mathcal{C}$; the figure shows the simplest case of a box function. The isolated points can then be optimized toward a target distribution μ.

We can now give a concrete definition of a class as a pair (w, μ). For a specific point set $\overline{X}$, the optimization objective is to have all points within the support of w follow the distribution μ.

Subclasses The expressiveness of our formulation comes from the use of non-trivial (i.e., non-box) class functions w. Such a function can specify multiple optimization objectives. Figure 5.4c shows a simple staircase function. Two unique intervals on $\mathcal{C}$ can be extracted by thresholding that function, selecting all $(w > 0)$ and half $(w > 0.5)$ of the class' points, respectively. Each such interval represents a distinct optimization objective.

Extending the example to a more complex staircase—or even smooth—function w, allows us
to specify an arbitrary number of sub-objectives, or subclasses, all sharing the target distribution μ. Each subclass selects



a point subset within the support of the thresholded, or filtered, class function w at a value $z \geq 0$ (see the inline figure). Formally, the support of the filtered function is the set $\{c\colon w(c) > z\} \subseteq \mathcal{C}$. A subclass is then defined by a tuple (w, μ, z).

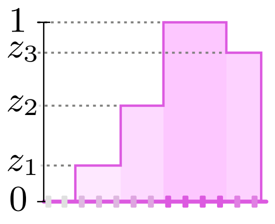
Point-set filtering A class is defined on a continuous extended space $\overline{\Omega}$. A smooth class function defined in $\overline{\Omega}$ thus specifies an entire continuum of subclasses. For a specific point set $\overline{X}$, their effective count is capped by the number of points. The subset of points selected by a subclass (w, μ, z) is obtained via a filtering operation: $\overline{X}_{w>z} \subseteq X$ contains all points $x_i \in X$ for which $w(c_i) > z$ (see fig. 5.4c). Note that this also strips the class coordinate, producing a subset of the original point set X. A threshold value $z = 0$ selects all points in a class, and larger z values yield smaller subsets. The example in fig. 5.3b demonstrates one such example with two staircase functions. For each staircase function, when $w > 1/3$ it selects half of the points, otherwise for $w > 0$ all points are selected.

5.3.2 Subclass Barycenter

The objective specified by one subclass can be satisfied by minimizing the Wasserstein distance between the corresponding point subset and the target. However, subclasses overlap, meaning that a point can be subject to multiple objectives. A compromise between all subclass objectives can be achieved via a barycenter that minimizes all involved Wasserstein distances simultaneously. For a continuous class function w, the barycenter takes an integral form:

$$\overline{X} = \underset{\overline{X}}{\arg\min} \underbrace{\int_{\mathbb{R}} W_p^p\big(\overline{X}_{w>z}, \mu_{w>z}\big)\,\mathrm{d}z}_{B_p(\overline{X}, w, \mu)}, \tag{5.14}$$

which is a continuous variant of eq. (5.8), with the difference that we have one target distribution μ and multiple optimization (point) distributions. We scale the target distribution, $\mu_{w>z}$, to match the total mass of the subclass $\overline{X}_{w>z}$. The scaling factor is the relative number of points in the subclass compared to the size of the entire point set. When the class function w is piecewise constant, with levels $0 = z_0, z_1, \ldots, z_s = 1$ (see inline figure), the integral becomes a sum, turning the problem into a discrete barycenter:



$$\overline{X} = \underset{\overline{X}}{\arg\min} \sum_{j=1}^{s} \lambda_j W_p^p\big(\overline{X}_{w>z_j}, \mu_{w>z}\big), \quad \text{with} \quad \lambda_j = z_j - z_{j-1}. \tag{5.15}$$

We enforce w to have a maximum value of one to ensure that the weights λ_j sum up to unity. For a trivial (box-function) class, the barycenter simplifies to the single objective of minimizing the Wasserstein distance between all class points and the target μ.

Note that the class function w encodes both the shape and the relative importance of each subclass (i.e., its weight λ_i in the discrete case). A class (w, μ) thus completely describes an entire optimization problem (5.14), independently of the point-set size.

5.3.3 Multi-Class Barycenter

While a single subclass barycenter can completely describe some optimization tasks, it is not sufficiently expressive to model many practical problems. For example, having overlapping point subsets follow different target distributions. Even with one target, multiple subsets can be assembled into a single class only if they are nested into one another. The example in fig. 5.3 cannot be modelled with a single class as the red and blue subsets are disjoint. Such problems require specifying and optimizing across multiple classes.

Continuous Case To specify multiple classes, we add one more dimension to our representation from fig. 5.4, illustrated in fig. 5.5a. Each point t on the $\mathcal{T}$ axis identifies a class (w_t, μ_t). $\mathcal{T}$ can be multi-dimensional but for simplicity we use the unit line: $\mathcal{T} = [0, 1]$. The different classes generally have conflicting objectives due to overlaps in their associated functions w_t. As discussed in section 5.3.2, the compromise between these objectives can be expressed as the Wasserstein barycenter

$$\overline{X} = \underset{\overline{X}}{\arg\min} \int_0^1 \underbrace{\int_0^1 W_p^p\big(\overline{X}_{w_t>z}, \mu_{t,w_t>z}\big)\,\mathrm{d}z}_{B_p(\overline{X}, w_t, \mu_t) \quad \text{eq. (5.14)}}\,\mathrm{d}t \tag{5.16}$$

across all classes (outer integral) and their subclasses (inner integral), recalling that our class functions have a maximum value of one.

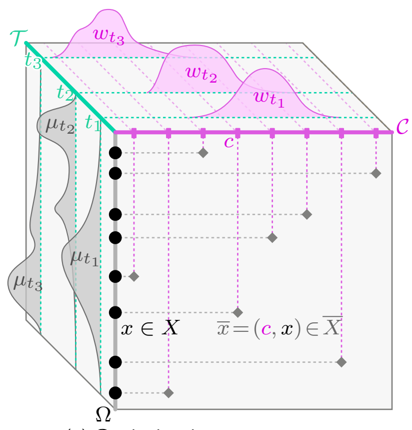
Discrete Case Not every identifier t has to map to a unique class. When the classes are a finite number n, the mapping is piecewise constant: $0 = t_0, t_1, \ldots, t_n = 1$, and every $t \in [t_{i-1}, t_i)$ maps to the class (w_i, μ_i). In the fully discrete case, where each class has a staircase-like function, the optimization problem (5.16) becomes a sum:

$$\overline{X} = \underset{\overline{X}}{\arg\min} \sum_{i=1}^{n} \sum_{j=1}^{s_i} \kappa_i \lambda_{i,j} W_p^p\big(\overline{X}_{w_i>z_{i,j}}, \mu_{i,w_i>z_{i,j}}\big), \tag{5.17}$$

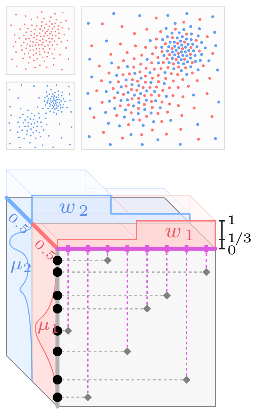
where $\kappa_i = t_i - t_{i-1}$ are the class weights, and $\lambda_{i,j} = z_{i,j} - z_{i,j-1}$ are the subclass weights as in eq. (5.15).

Figure 5.5b extends the example from fig. 5.3 to non-uniform target distributions. We formalize this optimization problem using only two classes: (w_1, μ_1) and (w_2, μ_2). Filtering the point set using these class functions give four subsets: $\{\overline{X}_{w_1>0}, \overline{X}_{w_1>1/3}, \overline{X}_{w_2>0}, \overline{X}_{w_2>1/3}\}$. Equation (5.17) aims to find the barycenter defined by the Wasserstein distance wrt each subset. It is important to note that the target distribution is only defined for the red (μ_1) and the blue points (μ_2) and not their union. The union will be aiming towards a barycenter of μ_1 and μ_2 since each class function considers all the points (the union) when $w_i > 0$.

Discussion Note that the class functions w (defined along the magenta axis in fig. 5.5) can overlap. Overlaps allow increasing the "footprints" of individual objectives to target more points than would be otherwise possible; however, they also introduce conflicts. In regions of overlap, the values of each function indicate its class' relative local optimization priority. Consequently, using functions with smooth falloffs allows us to precisely



(a) Optimization parameters

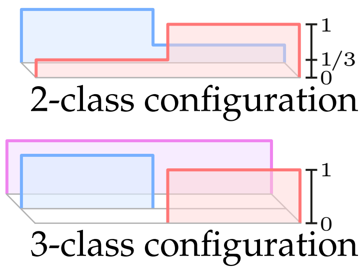


(b) Two-class example

Figure 5.5: (a) Our continuous optimization formulation yields a barycenter between classes (w_t, μ_t), each identified by point t on the unit line $\mathcal{T}$; we show three classes here. Smooth class-function falloffs allow for accurate control over conflicts resulting from overlaps. (b) A classical, fully discrete example with non-uniform target distributions. Each class function assigns equal optimization priority to each half of the points and their union. The top shows an optimized 256-point set.

control the barycentric trade-off between class objectives in such regions. In the general case of diverse targets μ_1 and μ_2, the class overlaps can make it difficult to satisfy simultaneously the objectives. Generally, we want to avoid having two classes assign high priority to the same region. Class functions should ideally be arranged to overlap only in their tails; this helps better satisfy each class' objective by minimizing conflicts and reducing optimization pressure. Class functions can be designed depending on the application.

Our more traditional-looking discrete barycenter (5.17) makes it clear that the atomic optimization objective in our framework is the subclass. A subclass is equivalent to a trivial, box-function class. The right inline figure at the bottom shows 3 such box functions representing the classical 3-class example (red, blue and their union). The formulation of Qin et al. (2017) supports only such classes. It is still as expressive as ours (2-class) but does not provide means to easily specify trade-offs between many objectives as it is not designed to scale to large number of objectives. Finally, we do not need to explicitly specify a target distribution for the union, which ends up being optimized toward a barycenter of the two targets. Figure 5.5b shows one such example point set where the target distributions are only defined for the red and blue points. The union is optimized towards their barycenter following eq. (5.17).



5.4 Stochastic Gradient-Descent For Multi-Class Optimization

In its most general form, the multi-class barycenter problem (5.16) is continuous. For a finite number of optimization points, the effective number of subclasses within any class is finite too. However, the use of continuous class functions makes this number very large, far beyond the few objectives that state-of-the-art multi-class methods (Wei, 2010; Jiang et al., 2015; Qin et al., 2017) can scale to, in terms of both memory and computation time. This is because these iterative methods optimize for all objectives at every step.

Taking cues from stochastic gradient-descent methods (Bottou, 1998), our approach is to optimize one objective at each of many iterations. Such optimization trivially scales to arbitrarily many objectives, although with potentially reduced convergence speed. Another advantage of this approach is that memory consumption does not directly depend on the objective count.

Sliced Multi-Class Barycenter Our multi-class barycenter formulation (5.16) computing optimal transport plans and minimizing Wasserstein distances, which can be very costly. For practical efficiency, we turn to sliced optimal transport, replacing the Wasserstein distance W_p by its sliced approximation SW_p (5.6). This adds another dimension to the integral in eq. (5.16), over the projection axis θ:

$$\overline{X} = \underset{\overline{X}}{\arg\min} \int_0^1\int_0^1\int_{\mathbb{S}^{d-1}} W_p^p\big(\overline{X}^{\theta}_{w_t>z}, \mu^{\theta}_{t,w_t>z}\big)\,\mathrm{d}\theta\mathrm{d}z\mathrm{d}t. \tag{5.18}$$

Since the sliced Wasserstein distance (5.6) bounds the regular Wasserstein distance (5.5) Bonnotte (2013), the resulting optimization problem (5.18) is an upper bound for the one in eq. (5.16). During optimization, we use this eq. (5.18) as our cost function which involves filtering point set for each slice θ. For brevity, we refer to this as filtered sliced optimal transport (FSOT).

Iterative Minimization The 1D subclass Wasserstein distance in eq. (5.18) has a known solution whose derivative w.r.t. an optimization point x_i (Rachev and Rüschendorf, 1998).

We start by rewriting the Wasserstien distance as :

$$W_p^p(\nu,\mu) = \int_0^{\infty} \big|\,F_\nu^{-1}(x) - F_\mu^{-1}(x)\big|^p \mathrm{d}x, \tag{5.19}$$

where F_ν^{-1} and F_μ^{-1} are the measures' inverse cumulative distribution functions (CDFs). We are specifically interested in the case where $p=2$ and one of the measures represents a 1D point set $X = \{x_i\}_{i=1}^n$. For this case we have

$$W_2^2(X,\mu) = \int_0^1 \big[F_X^{-1}(x) - F_\mu^{-1}(x)\big]^2\mathrm{d}x = \sum_{i=1}^n \int_{\frac{i-1}{n}}^{\frac{i}{n}} \big(x_i - F_\mu^{-1}(x)\big)^2\mathrm{d}x.$$

To differentiate this distance w.r.t. every point x_i, only one of the integrals depends on each x_i:

$$\frac{\mathrm{d}}{\mathrm{d}x_i} W_2^2(X,\mu) = \int_{\frac{i-1}{n}}^{\frac{i}{n}} \frac{\mathrm{d}}{\mathrm{d}x_i} \left(x_i^2 - 2x_i F_\mu^{-1}(x) + (F_\mu^{-1}(x))^2\right) \mathrm{d}x \tag{5.20}$$

$$= \int_{\frac{i-1}{n}}^{\frac{i}{n}} 2(x_i - F_\mu^{-1}(x))\, \mathrm{d}x = 2\frac{x_i}{n} - 2\int_{\frac{i-1}{n}}^{\frac{i}{n}} F_\mu^{-1}(x)\mathrm{d}x. \tag{5.21}$$

The integral of the inverse target CDF is given by the average point inside the i^{th} region of the target density with mass $1/n$, multiplied by $1/n$. The resulting derivative is thus similar to offset used by Paulin et al. (2020); the difference is the above scaling factor of $2/n$ and that they take the median target point (instead of the mean).

The derivative of the entire barycenter is then a nested integral of such 1D derivatives. This enables an iterative stochastic minimization scheme which performs repeated 1D gradient-optimization steps by randomly sampling the multi-dimensional integral in eq. (5.18).

Figure 5.6 illustrates one step of our optimization procedure. Given an extended point set $\overline{X}$ and a class configuration, we first select a class (w, μ), then threshold its function w with a random value z to choose a subclass that isolates a fraction $\overline{X}_{w>z}$ of the points. Finally, we sample an axis θ and perform one step of gradient-descent optimization on the 1D Wasserstein distance between the projected points $\overline{X}_{w>z}^{\theta}$ and the projected (scaled) target distribution $\mu_{w>z}^{\theta}$ along the axis.

Each 1D optimization step involves offsetting each projected point x_i^θ along the (negative) derivative of the Wasserstein distance w.r.t. the point's position:

$$x_i^\theta = x_i^\theta - \eta \cdot \gamma_i^\theta \cdot \underbrace{\frac{\mathrm{d}}{\mathrm{d}x_i} W_p^p(\overline{X}_{w>z}^\theta, \mu_{w>z}^\theta)}_{\Delta_i^\theta}, \tag{5.22}$$

where η is a step-size parameter (a.k.a. learning rate) and γ the offset scaling factor (section 5.4). In eq. (5.21) the derivative for the semi-discrete 2-Wasserstein distance is done for the case where both distributions are normalized. In the current case they have reduced mass due to filtering, which can be compensated by scaling the derivative by the relative number of selected optimization points $m/n = \left|\overline{X}_{w>z}\right|/n$:

$$\Delta_i^\theta = \frac{m}{n}\left[2\frac{x_i^\theta}{m} - 2\int_{\frac{i-1}{m}}^{\frac{i}{m}} F_{\mu^\theta}^{-1}(x)\mathrm{d}x\right], \tag{5.23}$$

where the semi-discrete derivative in the parentheses is computed w.r.t. normalized distributions.

We repeat this entire process multiple times to obtain many offset vectors for every point. We average these offsets, update the point's position, and begin a new iteration on the result. This is similar to the method of Paulin et al. (2020).

Since 1D projections of target distributions cannot always be analytically represented, we point-sample them at a rate 3–5× higher than the n points being optimized. The optimization still solves a balanced (i.e., discrete one-to-one) optimal-transport problem. This is done by binning the target points across n adaptive bins that follow the target distribution. Points within each bin are then averaged.

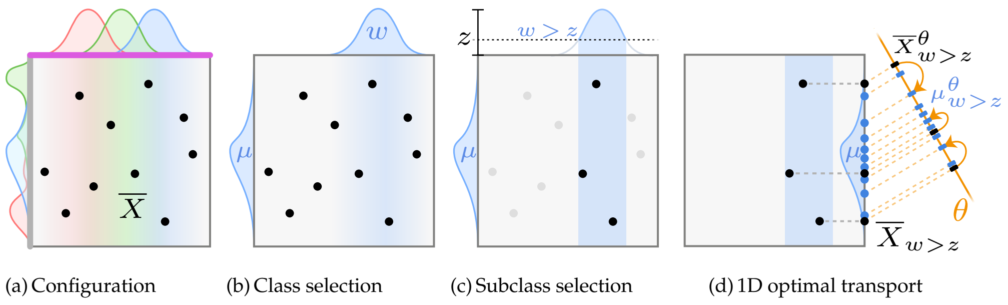


Figure 5.6: One step of our stochastic gradient-descent minimization of the sliced multi-class barycenter (5.18). Given the optimization parameters and an extended point set (a), we first select a class (b) and then randomly threshold) the class function to sample a subclass (c). Finally, we project the filtered set and the class' target distribution onto a sampled axis and perform one step of 1D gradient-descent optimization (d). We handle arbitrary target distributions by point-sampling them before projection.

Numerical Gradient Estimation The partial derivative step (5.22) requires computing the inverse CDF of the projected target distribution μ^{θ}. In practice, we use $c \times n$ points to better approximate the target distribution. First, all points are uniformly binned in n bins. The inverse CDF then adaptively changes the bin length according to the target distribution. The integral term in eq. (5.23) corresponds to the Monte Carlo estimate (i.e., average) location of the points within a certain interval of the inverse CDF: $b_i^{\theta} = {}^{1}\!/_{c} \sum_{j=(i-1)c}^{ic} y_j^{\theta}$, where the projected target samples y_j^{θ} are sorted. This gives the average location per i-th bin. Computing the offset Δ_i^{θ} then involves sorting x_i^{θ} and pairwise matching them with the bin values b_i^{θ}.

Offset Correction Projecting a target distribution μ along an arbitrary axis generally yields a different, non-uniform distribution μ^{θ} for each axis, even when the target is uniform (Paulin et al., 2020). Stochastic gradient descent on such different distributions can produce point offsets that are highly anisotropic in the (full-dimensional) optimization domain. The anisotropy is aligned with the density/domain boundaries and is susceptible to causing point alignments, as seen in fig. 5.7c. We avoid this problem by scaling the gradient of each projected point by the projected target density at that point. We estimate this density using the projected target samples.

The scaling factor γ_i (5.22) is simply the relative change in the length of the i-th bin:

$$\gamma_i^{\theta} = \frac{\text{Average bin length}}{\text{Length of bin } i} = \frac{(F_{\mu^{\theta}}^{-1}(1) - F_{\mu^{\theta}}^{-1}(0)) / m}{F_{\mu^{\theta}}^{-1}(i/m) - F_{\mu^{\theta}}^{-1}((i-1)/m)} \tag{5.24}$$

For single-class sampling, this gradient correction is the major change between our method and Paulin et al. (2020) which leads to the quality improvement shown in fig. 5.7f.

Discussion The sliced Wasserstein distance is only an approximation to the regular distance, and can yield suboptimal barycenters (Bonneel and Pfister, 2013). However, in our experience it is a practical option for optimizing many points for many targets and produces satisfactory results even with highly non-uniform target distributions. Other

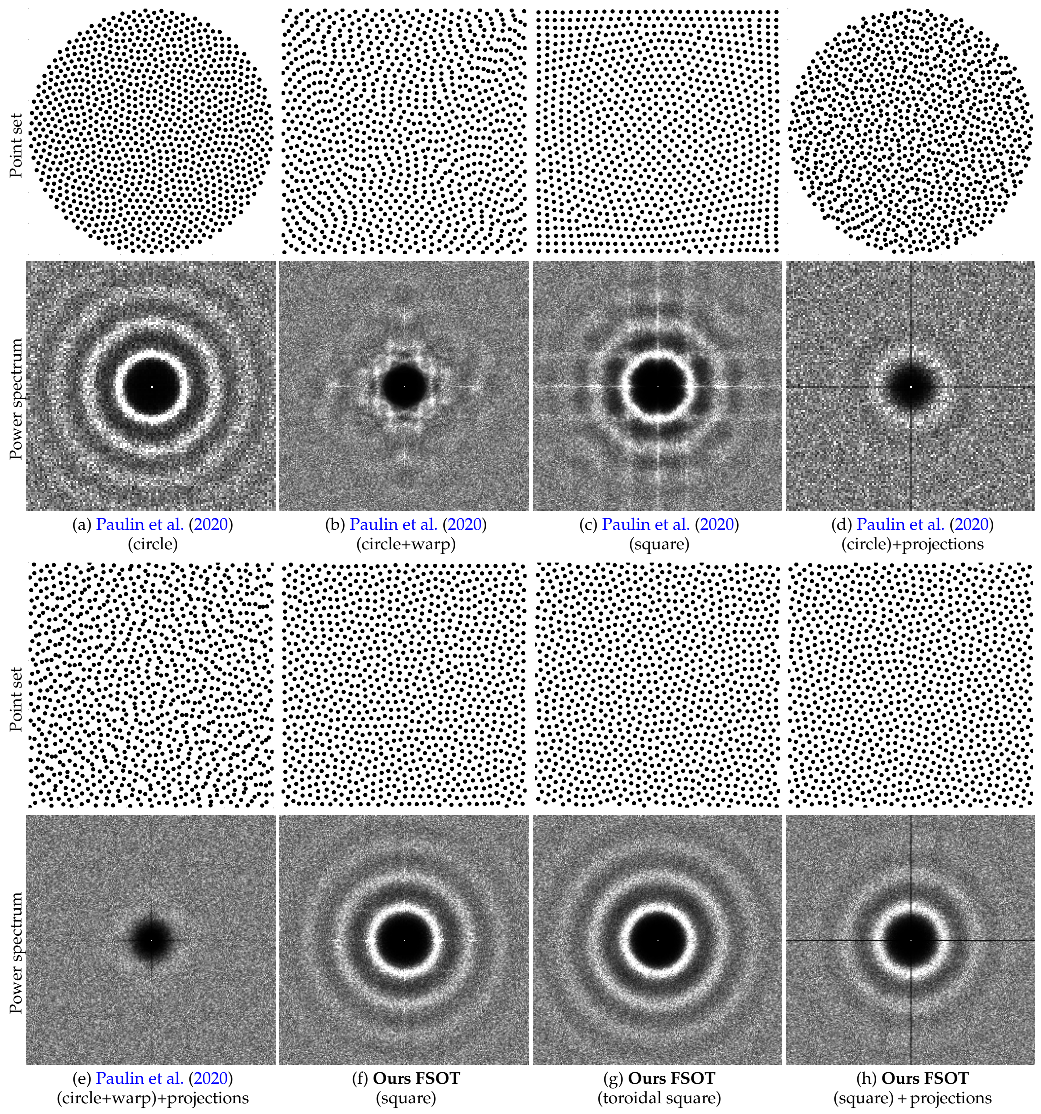


Figure 5.7: Comparison between different variants of our optimization and that of Paulin et al. (2020) which we build upon. All point sets are of size 1024 and the Fourier power spectra are averaged over 10 realizations. For our method we show realizations constructed with and without toroidality. Paulin et al. optimize on the unit circle (a) and then warp the resulting point set to the unit square (b); they also show direct unit-square optimization (c). Both their variants yield alignments that our method avoids (f-h), largely thanks to our offset correction (described in section 5.4). One can also prioritize certain projections which can be beneficial for Monte-Carlo integration (see fig. 5.16); here we choose the x- or y-axis in 30% of the optimization steps (d, e, h).

approaches such as entropic regularization (Cuturi, 2013), stochastic barycenters (Claici et al., 2018) or neural solvers (Korotin et al., 2022) can also be employed but we leave that for future work. Another consequence of using sliced optimal transport is that it increases the effective number of objectives, by adding an extra (spherical) dimension to the barycentric integral (5.18). Thankfully, the individual "sliced" 1D objectives are

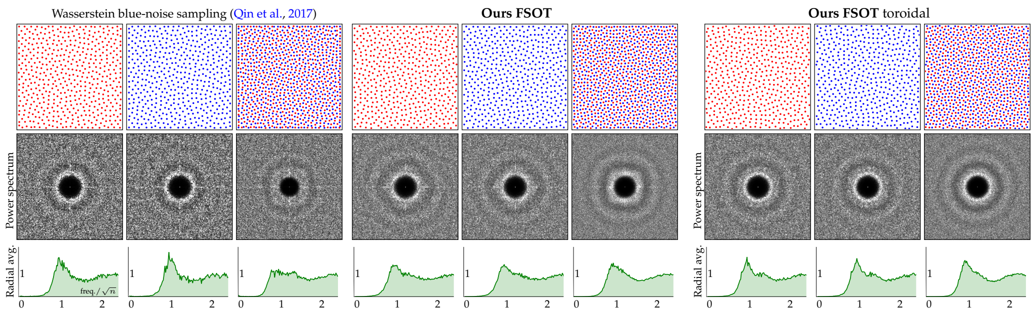


Figure 5.8: 3-class (red, blue, red & blue) optimization of 2048 points (top row), along with the corresponding expected power spectra (middle row) and their radial averages (bottom row). Our optimization (bounded and toroidal) achieves similar quality to that of Qin et al. (2017) (bounded); ours takes 38 sec on GPU and theirs takes about 1 hour on CPU. The spectral anisotropy in the left two results is due to point alignments near the boundaries.

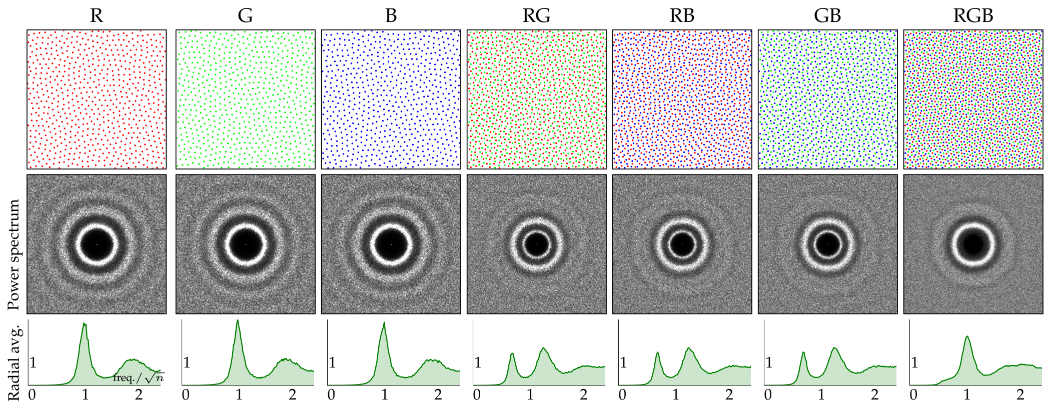


Figure 5.9: Extending the problem in fig. 5.8 to three colors (i.e., 7 classes), using 2049 points (683 points per color). Achieving uniform blue-noise quality across all classes is more difficult in this case due to higher, contention between the objectives.

simple, and stochastic optimization scales to the added complexity.

5.5 Experiments

To demonstrate the utility of our multi-class framework, we show the results of several experiments from CPU (C++) and GPU (CUDA) implementations. The C++ implementation of our stochastic gradient-descent optimization is parallelizable across the projections within each iteration. The CUDA one parallelizes over different operations (projections, sorting, averaging). The different point sets presented below have been generated on an NVIDIA Quadro RTX 8000 and Intel® Core™ i9-8950HK CPU @ 2.90GHz.

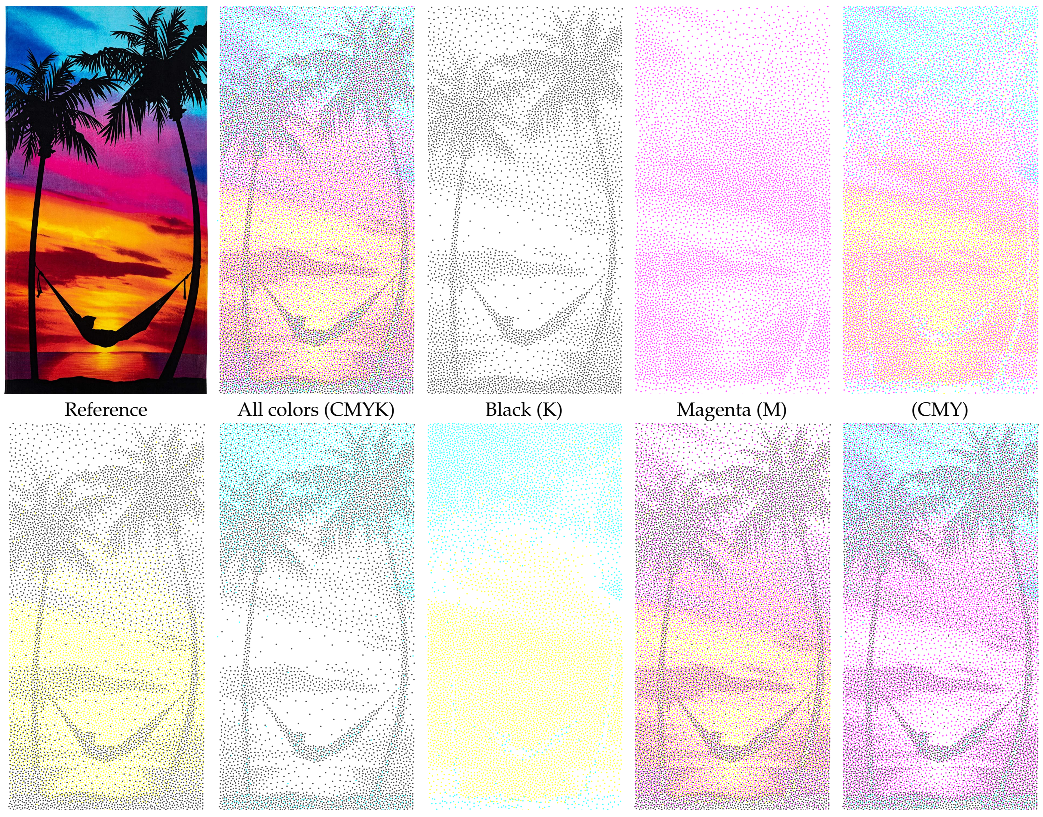


Figure 5.10: CMYK color stippling involves optimizing 15 classes—four base colors and their various 2- and 3-color combinations, each targeting a different density. In this example we use 20,000 points and show nine of these classes.

5.5.1 Blue-Noise Sampling

Single-class blue noise Figure 5.7 compares the blue-noise quality for a single-class point set and its power spectrum averaged over 10 realizations. Paulin et al. (2020) perform the optimization on the unit circle, achieving high quality (fig. 5.7a) which, however, deteriorates after warping the points to the unit square (fig. 5.7b); this is also reflected in the power spectrum. Paulin et al. also show direct optimization on the unit square, which yields strong alignments along the domain boundaries (fig. 5.7c). In contrast, our unit-square optimization produces a high-quality blue-noise distribution, without any alignments (see fig. 5.7f). This quality improvement is mostly due to our offset correction (section 5.4) which avoids alignments. Our optimization can also operate on a toroidal domain (fig. 5.7g).

Prioritizing certain projection directions can be beneficial in Monte-Carlo integration as we will demonstrate below; Figure 5.7h shows an example where we choose the x- or y-aixs with 30% probability, creating a cross in the power spectrum. While Paulin et al. (2020) can also prioritize these projections on the unit circle (fig. 5.7d), the achieved quality is not maintained after mapping the points to the unit square (fig. 5.7e).

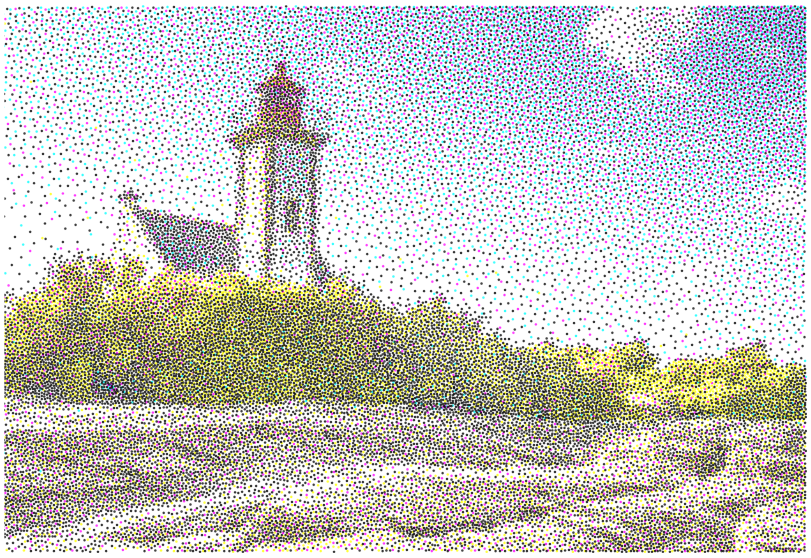
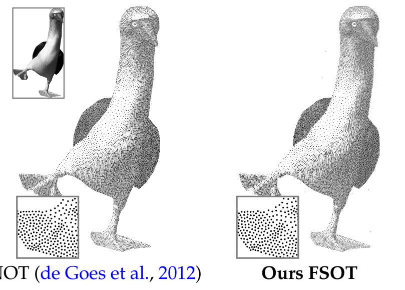
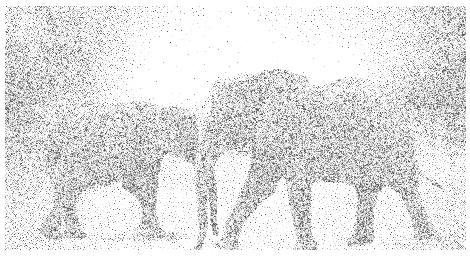


Figure 5.11: Example of a stippled image with cyan, magenta, yellow and black dots. It represents the image of a lighthouse using 40 000 points.



(a) Comparison of monochrome image stippling using 15,000 points. Our method achieves similar quality of method dedicated for single class optimization.



(b) Monochrome stippling using a large number of points (100 000 points). This example show the scaling capacity of our method to large number of points optimized at the same time.

Figure 5.12: Two examples of stippling with a single class. On the left is a comparison with the method of de Goes et al. (2012), and on the right, a use of a large point count.

Multi-Class Sampling With Uniform Density In fig. 5.8, we compare our method to that of Qin et al. (2017) on the two-color problem from fig. 5.3. The spectra obtained by Qin et al. (2017) and our method without toroidality show some artefacts due to natural point alignments near the domain boundaries. Our method shows same quality for the single colors and slightly better for the complete set. In fig. 5.9, we extend the problem to 3 colors, i.e., 7 classes. The overall distribution quality is good for all classes. The spectral distributions of the three color pairs RG, RB, GB exhibit double peaks, which has also been observed by Qin et al. (2017, Fig. 7). The reason for this double peak is that the improvement of these particular two-color classes has a strong impact on the other classes. Improving two-color classes would reduce the quality of the other classes too much.

Color Stippling Figure 5.10 shows a CMYK image stippled with 20,000 points. The four individual colors and their various 2- and 3-color combinations each represent a class with a different target density, for a total of 15 classes. We show five of these classes. The combinations have weighted-average densities based on the respective energy of the channels. Unlike prior work (Qin et al., 2017), our stochastic gradient-descent optimization scales to this many classes with a negligible memory footprint. Figure 5.11 shows another example of color stippling with 40,000 dots. This example shows the image of a lighthouse with CMYK points. This example shows the scaling capability of the color stippling method.

Figure 5.12a compares our stippling to that of de Goes et al. (2012) on a greyscale image using 15,000 points. Although our method is not tuned for single-class problems, we achieve competitive quality. Figure 5.12b shows another stippling example using a large point count of 100,000 points. Our method scales to this high point count while maintaining high quality.

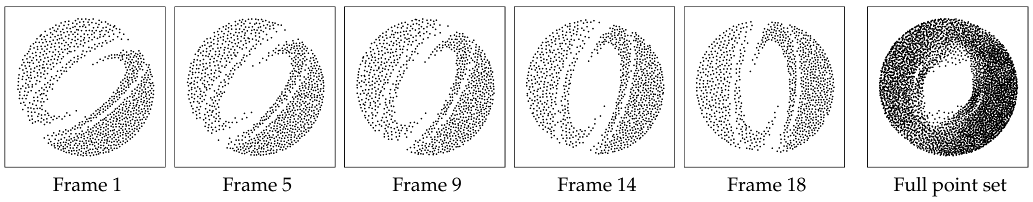
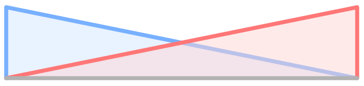


Figure 5.13: Stippling of an 18-frame animation using 5,500 points. Each frame shows the 1,000 points inside an index window that moves by 250 points between consecutive frames; every point thus participates in four frames.

Our method can also be applied to animation stippling, where consecutive frames share a subset of the points. Figure 5.13 illustrates an example of the SIGGRAPH logo rotating. Each frame consists of 1,000 points, with only 250 points being modified between any two consecutive frames. The figure shows five of these frames along with the entire point set. It is visible that the full set lacks a particular distribution, and only the selection of points within each frame gives the points their intended meaning. Despite retaining the majority of points between frames, each frame maintains a good blue noise distribution, ensuring high visual quality and coherence throughout the animation.

Continuous Class Extraction To demonstrate the scalability of our optimization to a large number of objectives, in fig. 5.14 we consider a non-traditional multi-class problem. We define two classes with linear-ramp functions on the index space of points, as illustrated in the inline figure. The target density is uniform. This construction allows us to split the optimized set at any point index, so that the subsets on the left and right of it (and their union) have good-quality distribution. We optimize 2048 points for an effective total number of 4096 targets (i.e., subclasses). In the figure we include a few example splits; the corresponding subset power spectra show reasonable blue-noise quality considering the large number of optimization objectives.



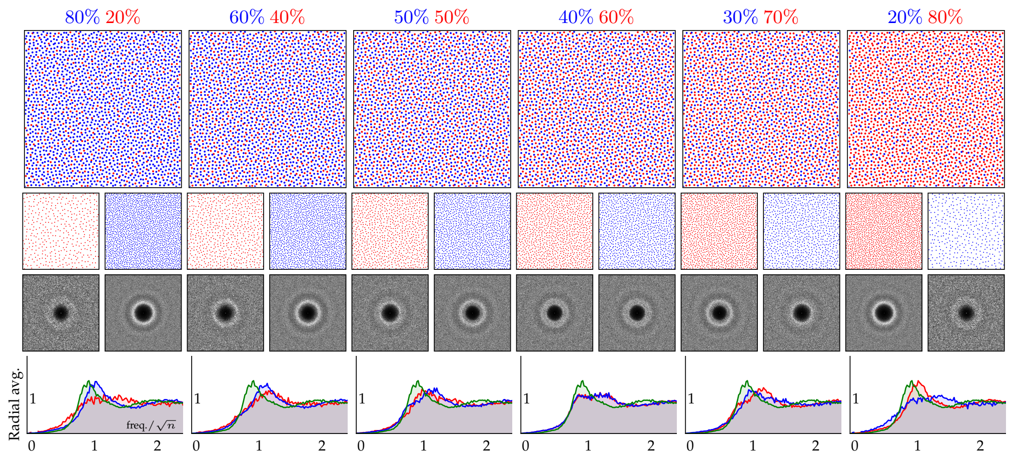
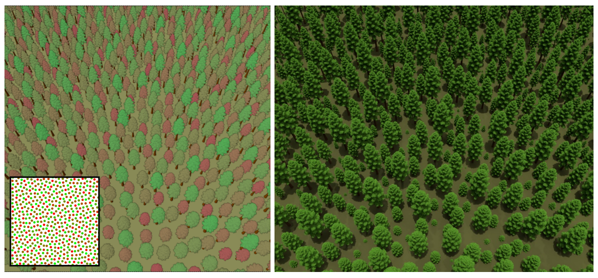
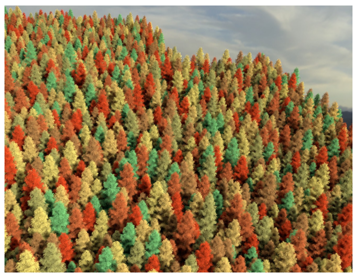


Figure 5.14: A set of 2048 points optimized according to a configuration with two linear-ramp-function classes. At every point index we can split the set into two subsets with good-quality distribution each, for an effective number of 4096 optimization targets (i.e., subclasses). We show six such (color-coded) splits and the 2D Fourier power spectra of the two extracted subsets. The last row shows the radially averaged spectra of the subsets and the entire point set (in green).



(a) Application of our continuous-class optimization to objects placement with continuous color variation (left) and size (right).



(b) Multi-class sampling for tree placement with 7 tree colors.

Figure 5.15: Application of multi-class sampling for object placement. (a) show the use of our continuous multi-class optimization for tree placement with varying size (Also visible as color on the left). Multi-class sampling can also be use to distribute trees of multiple colors in (b). In this example we use 7 colors from red to green.

Object Placement Multi-class sampling can also be used to place objects in an environment. Figure 5.15b shows an example distribution of trees, each taking one of 7 colors. We also optimize for the union, for a total of 8 classes. Two other results are displayed in fig. 5.15a. The point set used (in the lower left corner) was produced using the optimization configuration from the continuous class extraction problem presented above. In the left image, the point color guides the tree color, and in the right image it guides the tree height.

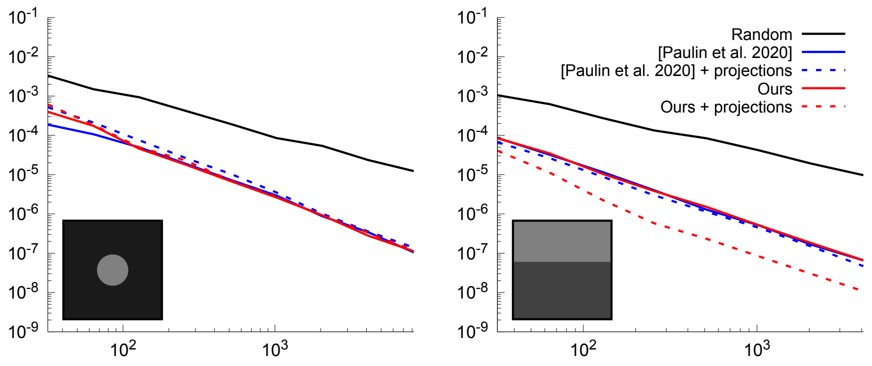


Figure 5.16: Comparison of the Monte-Carlo variance convergence of our optimized point sets against those of Paulin et al. (2020). We average variance over 10 realizations of each method and 40 variations of each function. Our axis-aligned projection prioritization is more effective than theirs.

5.5.2 Monte-Carlo Integration

We also evaluate our approach on Monte-Carlo integration. In fig. 5.16 we analyze the convergence behavior of our optimized point sets against the method of Paulin et al. (2020) on two simple integrands. We generate two types of point sets using each method: one with axis-aligned 1D projections prioritized with 30% probability (as in fig. 5.7h) and the other without prioritization (as in fig. 5.7g). For the isotropic integrand on the left the four variants give similar results. On the other hand, on the right integrand with axis-aligned variation, our projections yield lower integration error. Axis prioritization using Paulin et al.'s method is ineffective since the post-optimization point warping to the unit square ruins the point-set properties.

Progressive Sampling Our multi-class formulation allows constructing progressive point patterns with controlled granularity. We can use a single, staircase-function class where the number of steps (i.e., subclasses) dictates the number of prefix subsets (i.e., progressive levels) to optimize for. A constant class function corresponds to optimizing only the full set of points for uniformity; in the other extreme of a linear-ramp class function every prefix of points is optimized. Figure 5.17 shows progressive error-convergence plots for 5 such variants using 16,384 points. The steps have equal lengths in power-of-2 scale. The 1-subclass red curve behaves almost like a random one for all sample counts except for the strong dip at the end. Only when all samples are used is the integration error low; in fact, this is the lowest error achieved by any point set in the plot. Increasing the number of subclasses increases the number of dips but also shortens each. This result clearly illustrates that finer progressive granularity comes at the cost of increased error due to the larger number of objectives the optimization needs to balance. In the extreme case of 16,384 subclasses, the point set is fully progressive and shows uniform error behavior.

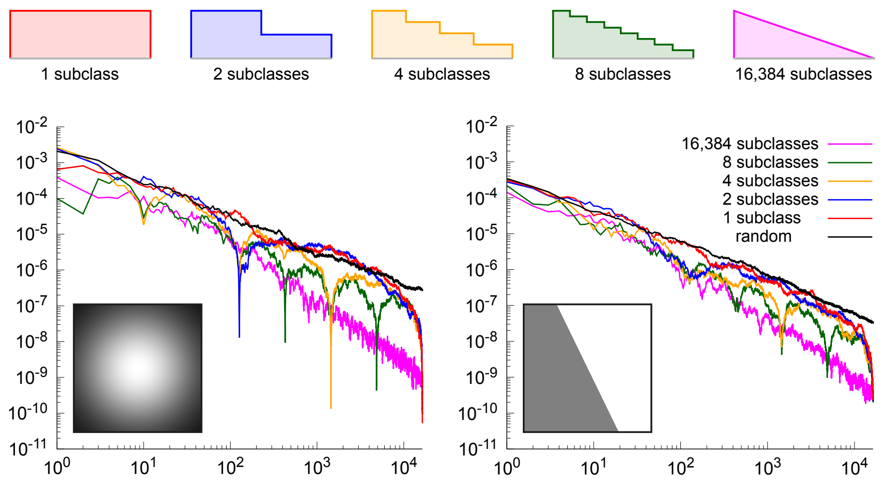


Figure 5.17: Progressive point-set optimization using a single, staircase-function class. Increasing the number of steps (with equal lengths in power-of-2 scale) increases the number of prefix subsets to optimize; we show 5 examples. The graphs plot integration-error behavior with increasing number of points taken, up to 16,384, averaged over 30 integrand variations and 20 point-set realisations. We see that finer progressive granularity yields a larger number of error dips, but each is shorter. The fully progressive (pink) point set exhibits uniform error behavior.

5.6 Perceptual Error Optimization

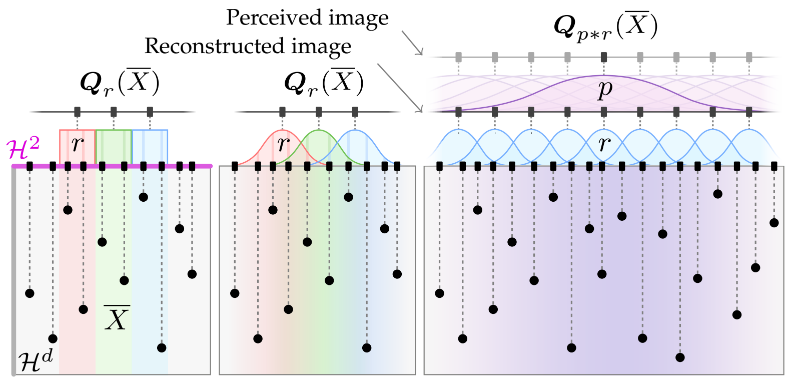
In this section we describe how to use our multi-class framework for perceptual optimization of image error in Monte-Carlo rendering.

In rendering, the value of every pixel is a light-transport integral. In practice pixel integrals are estimated via point sampling, and the resulting error manifests itself as image noise. Research efforts in sampling have traditionally focused on reducing the magnitude of the error, i.e., the accuracy of individual pixel estimates. Recently, it has been recognized that the distribution of this error over the image plays an important perceptual role, and that visual fidelity can be drastically improved when this distribution is isotropic and high-frequency (Georgiev and Fajardo, 2016). Achieving such a blue-noise distribution requires carefully coordinating the samples across pixels. We show that this problem can be cast as a multi-class optimization. We derive an image-error bound which can be minimized using our multi-class barycenter (5.16). The resulting formulation provides a principled way to minimize error in Monte Carlo rendering w.r.t. given perceptual and/or pixel-reconstruction kernels.

5.6.1 Problem Statement

Given a point set $X = \{x_i\}_{i=1}^{n}$, the value I_r of an image pixel is estimated by point-sampling its associated integral:

$$Q_r(X) = \frac{1}{n}\sum_{i=1}^{n} r(x_i)f(x_i) \approx I_r = \int_{\mathcal{H}^{2+d}} r(x)f(x)\,\mathrm{d}\rho(x) = \langle r, f\rangle. \tag{5.25}$$


Perceived image
Reconstructed image
$\boldsymbol{Q}_r(\overline{X})$
$\boldsymbol{Q}_r(\overline{X})$
$\boldsymbol{Q}_{p*r}(\overline{X})$
$\mathcal{H}^2$
r
r
r
p
$\overline{X}$
$\mathcal{H}^d$

(a) Box reconst. (b) Gaussian reconst. (c) Gaussian reconst.+percept.filter

Figure 5.18: Illustration of image synthesis where the grey box represents the sampling space, i.e. the unit hypercube $\mathcal{H}^{2+d}$; the horizontal axis represents the image subspace where reconstruction from the samples $\overline{X}$ is performed. (a) When using a box reconstruction kernel r, the sample sets estimating different pixels are disjoint. (b) A Gaussian kernel introduces overlaps, making each sample contribute to multiple pixel estimates. (c) The human visual system applies additional filtering on the reconstructed image with a generally wider kernel p. The convolution $p*r$ acts as an effective reconstruction kernel for the perceived image, and introduces even more overlaps.

Here r is a pixel-reconstruction kernel, $f(x)$ is the illumination carried by a light-transport path corresponding to the point x in the unit hypercube $\mathcal{H}^{2+d}$ with Lebesgue measure ρ. The first two dimensions are image space (where r operates), and d is the path-space dimension. The variance of an estimate $Q_r(X)$ is reduced when the samples in X within the kernel support are well-stratified.

When using box-kernel reconstruction (fig. 5.18a), every sample falls within the kernel of a single pixel, which allows stratifying the samples independently per pixel. Non-box kernels, e.g., Gaussians, generally overlap in image space, making each sample contribute to the estimates of several pixels (fig. 5.18b). This case calls for coordinating the stratification of samples across pixels.

Moreover, our eyes do not perceive individual pixels but rather process the image as a whole. One type of processing that occurs is pre-filtering the input visual signal to avoid aliasing. That is, we perceive a version of the image that is blurred by an amount dependent on the observing distance. This filtering can be modeled as a discrete convolution of the (r-reconstructed) pixels with a perceptual filter p (González et al., 2006; Näsänen, 1984) that can be well approximated by a Gaussian (Pappas and Neuhoff, 1999). Every pixel in the perceived ground-truth image thus takes the form $p * I_r = p*\langle r, f\rangle = \langle p*r, f\rangle = I_{p*r}$. Analogously, pixels in the perceived estimated image can be written as $Q_{p*r}(X)$, which we illustrate in fig. 5.18c. That image can thus be computed by convolving the samples with a combined reconstruction kernel $p * r$ centered at every pixel. The difference between the two images can be viewed as a measure of perceptual error (Chizhov et al., 2022). We can then formulate our problem as minimizing reconstruction w.r.t. a given (combined) kernel by optimizing the distribution of the samples X.

Note that in reality pixel reconstruction is performed by the renderer—to compute pixel estimates, while perceptual filtering occurs in the human visual system upon perceiving these estimates.

5.6.2 Multi-Class Image-Error Bound

Figure 5.18 illustrates graphically how image-error minimization can be viewed as a multi-class optimization problem. Mapping the problem to the language of section 5.3, the optimization domain is the d-dimensional unit hypercube, $\Omega = \mathcal{H}^d$, and, notably, the class dimension is not the unit line (e.g., as in fig. 5.3b) but the unit square, $\mathcal{C} = \mathcal{H}^2$. The regular and extended point sets are identical, $\overline{X} = X$. Every pixel has an associated reconstruction kernel and defines a distinct class, all sharing the Lebesgue measure ρ as their (uniform) target distribution. Next we show that the barycenter between these classes provides a bound for the (perceptual) error of the image.

Pixel-Error Bound The error of a pixel w.r.t. some given kernel w is the difference between the estimated value and the ground truth: $\epsilon_w(X) = \left|Q_w(X) - I_w\right|$. This becomes a perceptual error when the kernel $w := p * r$ incorporates perceptual filtering. Paulin et al. (2020) recently showed that optimal transport can provide a bound on the estimation error of pixel during Monte Carlo integration (eq. (5.13)). Unfortunately, this bound is not immediately useful: it measures the deviation of the entire point set $\overline{X}$ from uniformity and does not capture the strong effect of the narrow-support kernel w on each pixel estimate. We instead desire a bound tailored to the estimation of weighted integrals of the f form $w(x)f(x)$, where w is an analytically known function.

As in eqs. (5.12) and (5.13), our derivations use general probability measures ν and μ, but for our application we are specifically interested in the case where ν is a Dirac point-mass measure, i.e., a point set. We begin by expressing $w(x)$ and $w(y)$ in the error as integrals over corresponding indicator functions, then swap the integration order using Fubini's theorem:

$$\left|\int_{\mathcal{H}} w(x) f(x)\, \mathrm{d}\nu(x) - \int_{\mathcal{H}} w(y) f(y)\, \mathrm{d}\mu(y)\right| \tag{5.26}$$

$$\left|\int_{\mathcal{H}} \underbrace{\int_{\mathbb{R}} \mathbf{1}_{[0,w(x)]}(z) \mathrm{d}z}_{w(x)}\, f(x)\, \mathrm{d}\nu(x) - \int_{\mathcal{H}} \underbrace{\int_{\mathbb{R}} \mathbf{1}_{[0,w(y)]}(z) \mathrm{d}z}_{w(y)}\, f(y)\, \mathrm{d}\mu(y)\right|$$

$$= \left|\int_{\mathbb{R}} \left[\int_{\mathcal{H}} \mathbf{1}_{[0,w(x)]}(z) f(x)\, \mathrm{d}\nu(x) - \int_{\mathcal{H}} \mathbf{1}_{[0,w(y)]}(z) f(y)\, \mathrm{d}\mu(y)\right] \mathrm{d}z\right|. \tag{5.27}$$

Next, for any $x \in \mathcal{H}$ and $z \in \mathbb{R}$ we have:

$$\mathbf{1}_{[0,w(x)]}(z) = \mathbf{1}_{[z,\infty]}(w(x)) = \mathbf{1}_{\{x' \in \mathcal{H} | w(x') > z\}}(x) =: \mathbf{1}_{\mathcal{H}_{w>z}}(x), \tag{5.28}$$

the indicator function effectively restricts the integration to the region $\mathcal{H}_{w>z}$ where

$w(\cdot) > z$. Plugging this identity into eq. (5.27) and then using $\left|\int g\right| \leq \int |g|$, we get

$$= \left| \int_{\mathbb{R}} \left[\int_{\mathcal{H}} \mathbf{1}_{\mathcal{H}_{w>z}}(x) f(x)\, \mathrm{d}\nu(x) - \int_{\mathcal{H}} \mathbf{1}_{\mathcal{H}_{w>z}}(y) f(y)\, \mathrm{d}\mu(y) \right] \mathrm{d}z \right| \tag{5.29}$$

$$\leq \int_{\mathbb{R}} \left| \int_{\mathcal{H}} \mathbf{1}_{\mathcal{H}_{w>z}}(x) f(x)\, \mathrm{d}\nu(x) - \int_{\mathcal{H}} \mathbf{1}_{\mathcal{H}_{w>z}}(y) f(y)\, \mathrm{d}\mu(y) \right| \mathrm{d}z \tag{5.30}$$

$$= \int_{\mathbb{R}} \left| \int_{\mathcal{H}} f(x)\, \mathrm{d}\nu_{w>z}(x) - \int_{\mathcal{H}} f(y)\, \mathrm{d}\mu_{w>z}(y) \right| \mathrm{d}z, \tag{5.31}$$

where $\nu_{w>z}$ and $\mu_{w>z}$ are the measures ν and μ restricted to the region $\mathcal{H}_{w>z}$. We can now apply eq. (5.12) to the absolute error in the outer integral, obtaining a bound for the expression in eq. (5.26):

$$\left| \int_{\mathcal{H}} w(x) f(x) \mathrm{d}\nu(x) - \int_{\mathcal{H}} w(y) f(y) \mathrm{d}\mu(y) \right| \leq L_f \int_{\mathbb{R}} W(\nu_{w>z}, \mu_{w>z}) \mathrm{d}z. \tag{5.32}$$

It is important to note that for the Wasserstein distance to work, the measures $\nu_{w>z}$ and $\mu_{w>z}$ must have equal masses in the hypercube subset corresponding to each valid slicing $w > z$. In other words, we need $\nu(\mathcal{H}_{w>z}) = \mu(\mathcal{H}_{w>z})$, or equivalently, $\nu_{w>z}(\mathcal{H}) = \mu_{w>z}(\mathcal{H})$, for all $z \in [0, \max w(.)]$.

This bound can be applied to Monte Carlo error bound of a discreet pointset as:

$$\epsilon_w(\overline{X}) \;\leq\; L_f \int_{\mathbb{R}} W(\overline{X}_{w>z}, \rho_{w>z})\, \mathrm{d}z \;=\; L_f B_1(\overline{X}, w, \rho), \tag{5.33}$$

where B_1 is the minimization objective of the 1-Wasserstein subclass barycenter (5.14). Note that the kernel w has moved from the Lipschitz constant to the Wasserstein distance compared to the error bound from Paulin et al. (2020).

Image-Error Bound Our end goal is to minimize the total image error. Applying the bound from eq. (5.33) to each of M pixels yields a bound for the L_1 error:

$$\sum_{i=1}^{M} \epsilon_{w_i}(\overline{X}) \;\leq\; L_f \sum_{i=1}^{M} B_1(\overline{X}, w_i, \rho). \tag{5.34}$$

This bound is a product of the Lipschitz constant of f and a (discrete) M-class barycenter (5.16). It postulates that to reduce the image error, we need to increase the uniformity of all subsets of $\overline{X} = X$ given by the z-filtering of every kernel (i.e., class function) w_i.

Equation (5.34) is based on the 1-Wasserstein distance W_1, but in practice we use our W_2-based optimization scheme from section 5.4 to minimize a sliced variant of the bound. This works because SW_1 is bounded by SW_2. Note that we do not optimize the image-space dimensions of the points which are fixed and used for classification.

5.7 Results

For rendering applications, we optimize a point set covering 128×128 pixels that is toroidally tiled over the image. All rendering results have been generated using PBRT-v3 Pharr et al. (2016b) using optimized samples for each presented methods.

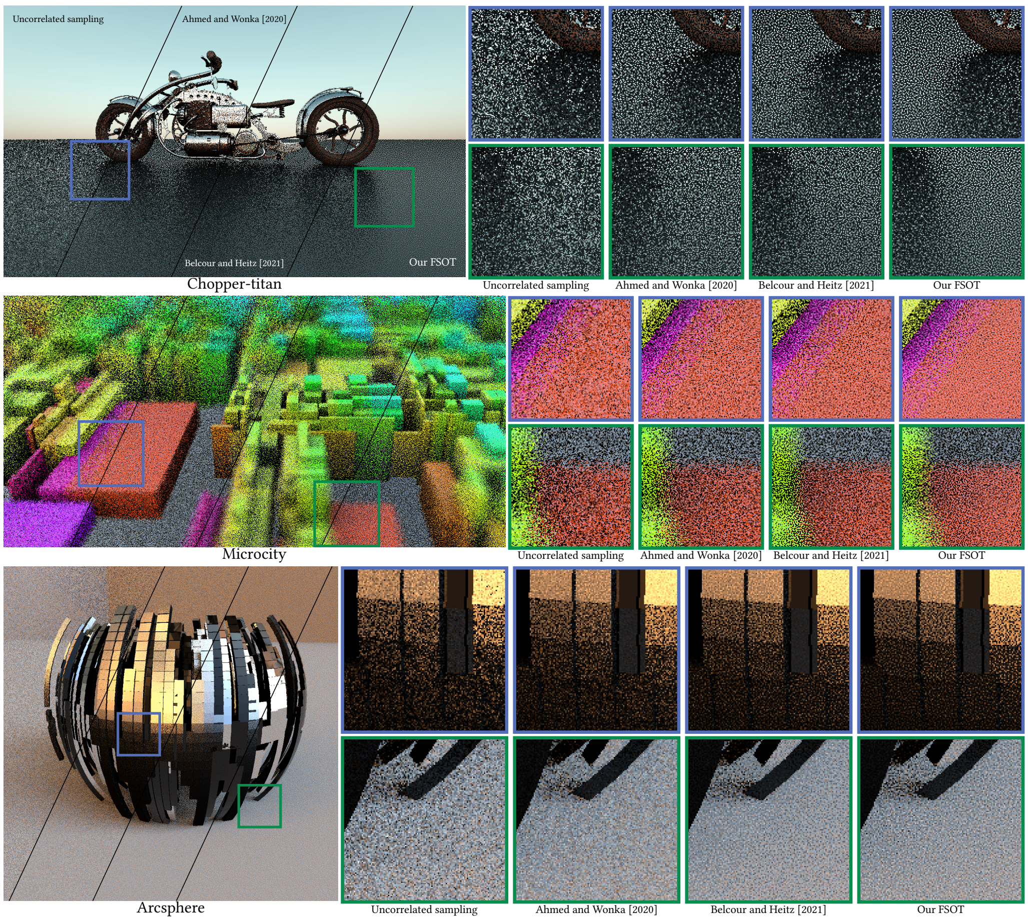


Figure 5.19: Comparison of our perceptual-error optimization against classical uncorrelated pixel sampling and state-of-the-art blue-noise error distribution methods. The top and bottom scenes are directly lit by an environment map, and the middle scene has defocus blur that increases the sampling dimensions to four.

In fig. 5.19, we compare our point sets against those from prior work on perceptual (i.e., blue-noise) error optimization (Ahmed and Wonka, 2020; Belcour and Heitz, 2021); we use box reconstruction for a fair comparison. The top 2 scenes are rendered with 1 sample per pixel under direct lighting and the bottom one with 4 samples per pixel. The benefit of our approach (rightmost column) is most visible in the top scene, where the specular regions show a much improved error distribution. In the middle row scene, we use a finite-aperture camera. The zoom-ins show the better perceived quality achieved by our method over the state of the art. The bottom scene show that our approach also scale to multiple sample per pixel while still achieving the highest distribution quality.

Figure 5.20 shows an additional rendering result for perceptual error distribution comparing an uncorrelated rendering, Ahmed and Wonka (2020) , Belcour and Heitz (2021) and our method. This result display an additional image showing a power spectrum per tile for each method. It show the spectral distribution of the noise and in particular the blue noise distribution property. Our method achieve the best blue noise characteristics

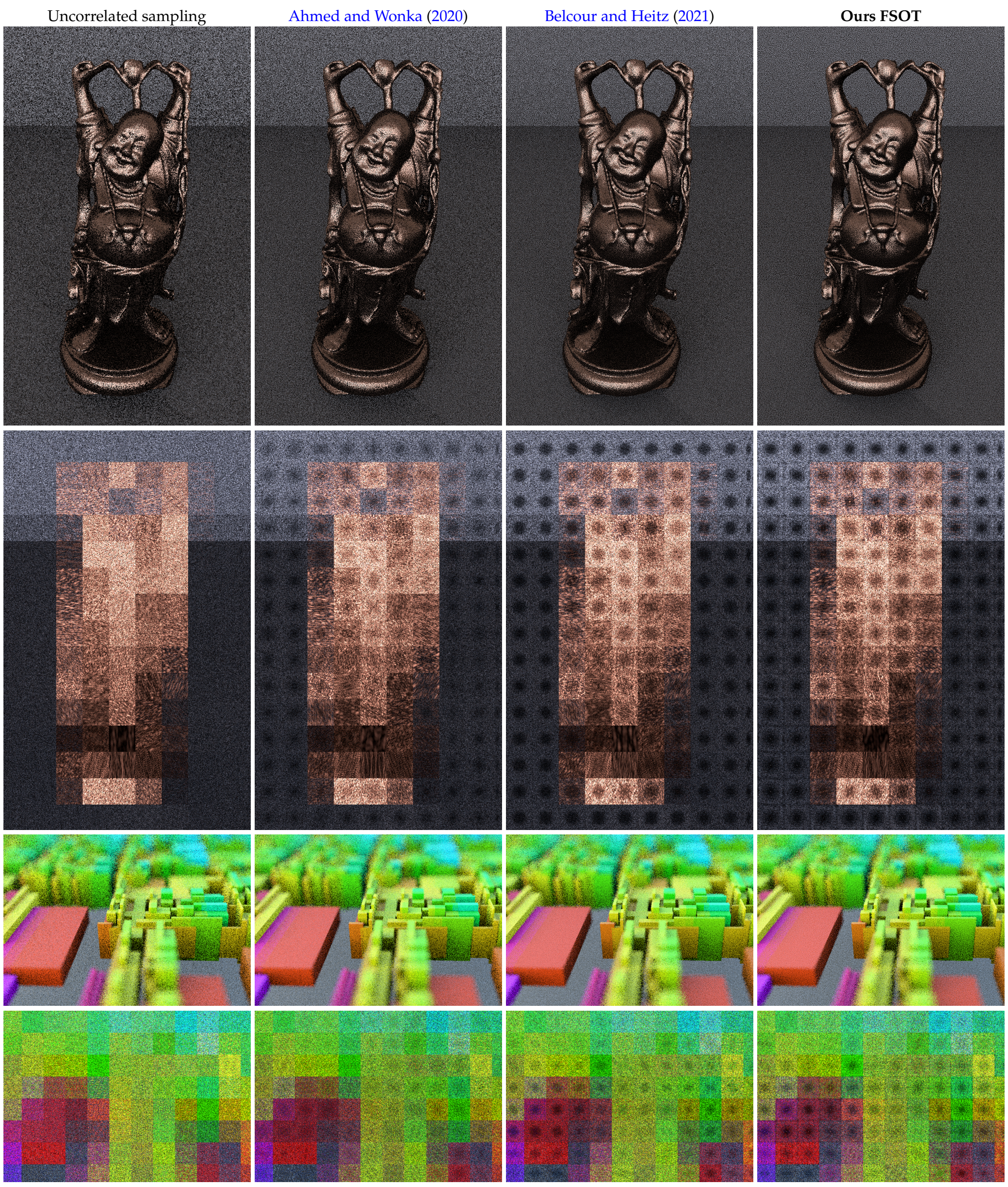


Figure 5.20: Comparison of rendering our method with Ahmed and Wonka (2020) and Belcour and Heitz (2021). The first line shows the rendering at 1 samples per pixel for the buddha scene and at 4 samples per pixel for the microcity scene. The second line shows the tilled power spectrum of the error with the reference.

with a a low frequency region of same size as Belcour and Heitz (2021) but with a slightly lower energy in this region. This better characteristics is also visible is the rendering image with a better visual quality and noise distribution.

While traditionally point sets are optimized assuming a box pixel-reconstruction kernel, our framework allows optimizing for arbitrary kernels. Figure 5.21 shows the impact on

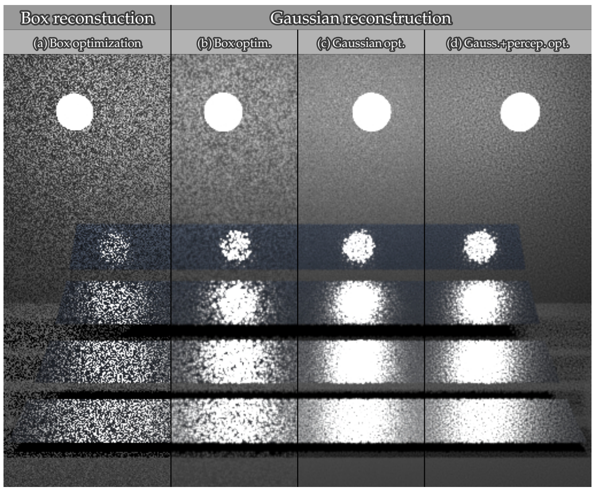


Figure 5.21: Optimizing pixel samples for different reconstruction kernels. (a) When using box reconstruction, the samples for individual pixels can be optimized separately. (b) Traditionally this optimization is used also when the reconstruction is non-box. (c) Our framework allows optimizing for the specific kernel used, taking into account overlaps between pixels and showing substantial error reduction. (d) Additionally taking into account perceptual blur achieves blue-noise error distribution over the image..

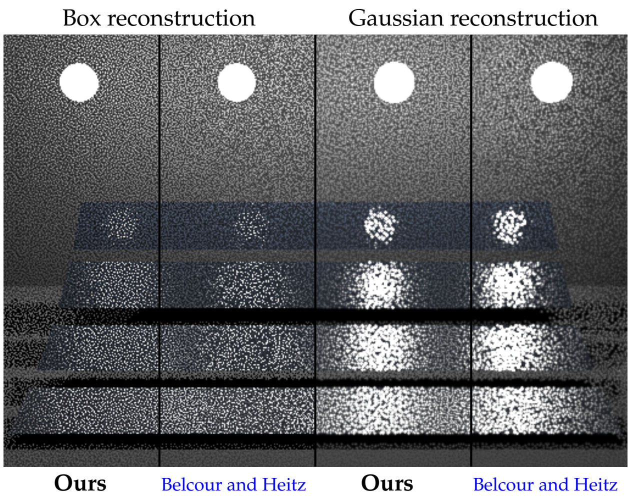


Figure 5.22: Comparison between our perceptual error optimization and the blue-noise error distribution method of Belcour and Heitz (2021) using box and Gaussian pixel-reconstruction kernels. Our method achieves better quality in both cases.

error distribution while taking into account the reconstruction kernel. On the right, note the substantial improvement in fig. 5.21c over fig. 5.21b, due to specially optimizing for the Gaussian reconstruction kernel used. Additionally accounting for perceptual blur further pushes the error distribution toward high frequencies (fig. 5.21d).

Figure 5.22 shows the comparison of our method with Belcour and Heitz (2021) for rendering at 1 sample per pixel. On the right the comparison shows box reconstruction filter and on the right gaussian filter. We can see that our method performs better in both cases but this difference is more important for the gaussian filter. Contrary to Belcour and Heitz (2021) our method can be optimized for both the perceptual and the reconstruction filter. This adaptive of our method leads to the improvement visible here and more importantly the capacity to scale to more complex pixel-reconstruction kernel.

5.8 Extension To Temporal Domain

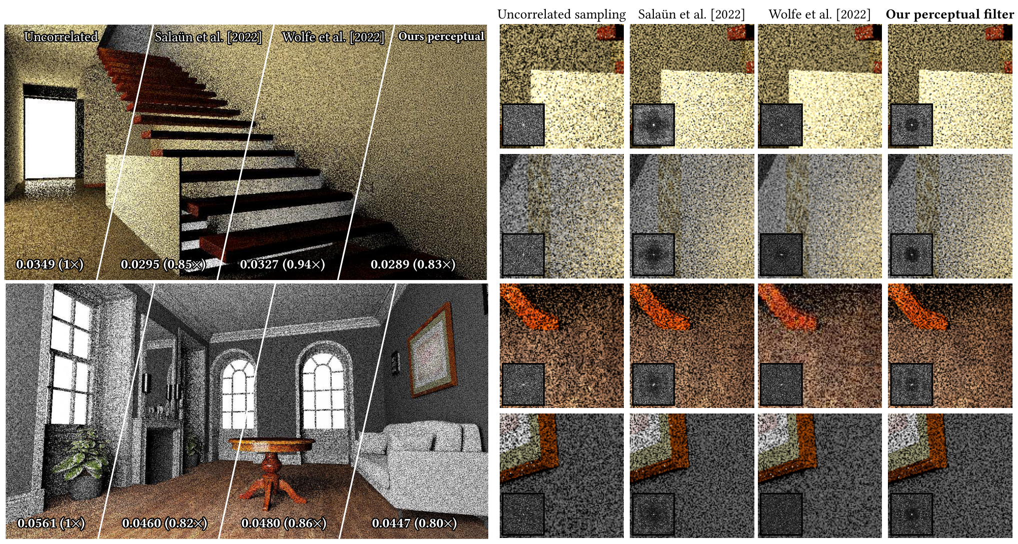


Figure 5.23: Comparison of the 16th animation frame. To mimic human perception, for display we apply the temporal filter of Mantiuk et al. (2021). It compares optimization with spatial en temporal components uncorrelated sampling, spatial only optimization and the method of Wolfe et al. (2022). The insets in each crop show the DFT of the error image, and the numbers on the left are the filtered RelMSE of each method (lower is better). Full optimization achieves visible improvements over other methods on both scenes and a more pronounced blue-noise distribution in the DFT spectrum.

The previous section described a method to optimize Monte Carlo rendering error distribution through perception-aware optimization of samples. This concept can also be extended to the temporal domain. By modeling perception using a spatial kernel, as previously discussed, and incorporating a temporal sensitivity function, it is possible to account for spatio-temporal perception. This approach allows for the optimization of a multi-frame sample set, minimizing the error across both space and time. This extension, which leverages spatio-temporal perception modeling, has been implemented and demonstrated in Korać et al. (2023).

In animation, the human visual system does not perceive each frame in isolation. Rather, temporal perception can be modeled as a low-pass filter in the temporal domain Mantiuk et al. (2021); Burbeck and Kelly (1980); Hammett and Smith (1992). In our experiments, we employ the kernel proposed by Mantiuk et al. (2021), which is a sum of two components: a sustained kernel and a transient kernel. The sustained kernel encodes the response to

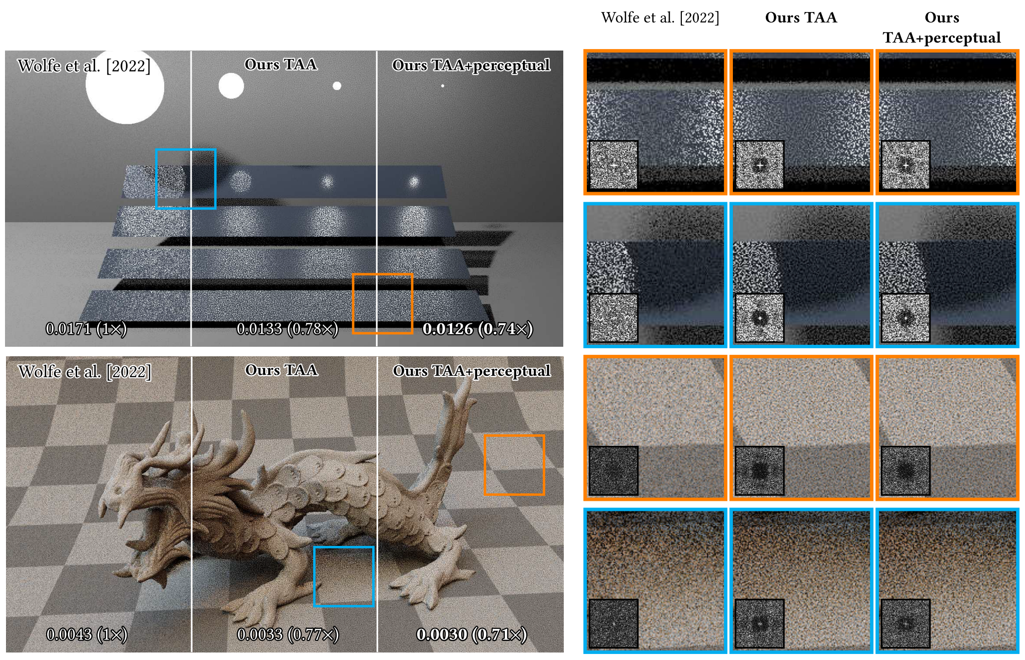


Figure 5.24: Rendering comparison on the 10th animation frame, rendered with 4 samples per pixel. To mimic human perception, for display the temporal filter of Mantiuk et al. (2021) is applied. Comparison of spatio-temporal optimization with the method of Wolfe et al. (2022) is visible. The insets in each crop show the DFT of the error image, and the numbers on the left are the filtered RelMSE of each method (lower is better). Optimization including both filters better preserves the desirable blue-noise error distribution. It also show the capacity of producing perceptually pleasing error distribution with multiple samples per pixel.

slow temporal changes, while the transient kernel responds to fast changes Hammett and Smith (1992); Burbeck and Kelly (1980). We convolve this temporal kernel with the spatial kernel described in eq. (5.25). By optimizing samples for both spatial and temporal components, we extend the previous results to animation rendering. Additionally, we show that it is possible to include temporal filtering, such as Exponential Moving Average for temporal anti-aliasing Yang et al. (2020), in the optimization.

Compared to the state-of-the-art spatio-temporal perceptual error distribution method from Wolfe et al. (2022), our approach does not separate the optimization of spatial and temporal properties. Instead, it considers their combination by convolving the two kernels, allowing for better results as it imposes fewer constraints on the optimization.

Figure 5.23 shows a comparison of spatio-temporal rendering. Since it is not possible to show a video, the image represents the temporal accumulation of images as perceived according to Mantiuk et al. (2021). This allows visualization of the accumulation of multiple frames in a single image. The result demonstrates that spatial-only optimization, as proposed in the previous sections, does not achieve a high-quality error distribution due to the lack of correlation in the temporal domain. However, it still performs better than uncorrelated noise. Importantly, this spatial-only optimization outperforms the temporal method by Wolfe et al. (2022) due to a significantly better error distribution in each frame. The best results are achieved with a joint optimization of spatial and temporal components.

Figure 5.24 shows a comparison with optimization including temporal anti-aliasing. This figure demonstrates the flexibility of our approach, which allows for the joint optimization of the temporal filter and the perceptual kernel, achieving higher quality results. Compared to Wolfe et al. (2022), our method significantly reduces error as indicated by the error metric. It is also visible that combining both TAA and the perceptual kernel does not substantially improve the result, likely due to the high correlation between the two filters and the non-adaptivity to specific scene characteristics. Nonetheless, the closer the filter matches the actual filtering, the lower the error.

This extension to the temporal domain demonstrates the method's capacity to adapt to other types of multi-class applications for rendering. Adapting this method to more complex and specialized kernels is possible but limited by the a priori optimization, which is agnostic to the specific rendering scene characteristics.

5.9 Limitations and Future Work

Algorithmic Complexity And Performance The bottleneck of our optimization is the sorting of m projected optimization points and $c \cdot m$ density-sample points (where $c =$ const), with complexity $O(m \log(m))$ per iteration. The number of classes and subclasses has no direct impact on complexity, although in practice increasing the number of optimization objectives can impact the convergence speed of gradient descent. The memory consumption of our algorithm is linear in the total number of optimization points n.

For 4096 points, single-class GPU optimization takes 40 sec, 3-class takes 59 sec, and 7-class takes 71 sec. For 262,144 points, single-class takes 3840 sec (2000 iterations), 3-class takes 4325 sec (2500 iterations), and 7-class takes 4370 sec (3000 iterations). The added cost of increasing the number of classes is moderate. The reason is that, while more classes require more optimization iterations to obtain high quality, the time per iteration is lower as fewer points are optimized at once (since one subclass it optimized per iteration). With this in mind, it is possible to imagine a more efficient optimization, e.g., utilizing a data structure to pre-order the points before projection and then using a sorting algorithm that takes advantage of this pre-ordering. One can also imagine relaxing the constraints on the Wasserstein equations to perform local rather than global optimizations. By computing several Wasserstein distances on subsets of the domain, it is possible to approximate the full distance with fewer points in each "sub-distance". Because of the complexity of these operations, reducing the number of points would speed up the computation at the cost of a looser error bound.

Limitations Our multi-class Wasserstein barycenter objective has a fully integral form, which allows us to leverage stochastic optimization and achieve scalability. However, optimizing for a single objective per iteration can yield noisy gradients and slow down convergence toward the sought barycenter. Our point-sampling of non-uniform distributions exacerbates the issue by adding more noise to the gradients.

Wherever functions of different classes overlap, points are implicitly optimized toward a barycenter of the corresponding target distributions. Some applications require a union of point subsets to follow a mixture of the targets instead. A notable example is color stippling where the base targets are the distributions of the individual color channels. Our framework requires specifying mixture targets explicitly via dedicated classes.

Future Work Our optimization can benefit from analytic target-distribution projection and informed choices of projection axes that allows tailoring application-specific samplers. A more advanced optimizer could achieve better local minima than stochastic gradient descent. While enabling efficient optimization, the sliced Wasserstein barycenter we use may not yield a good distribution interpolation (Bonneel et al., 2015; Bonneel and Pfister, 2013). Efficient optimization of the regular Wasserstein barycenter is an interesting direction for future investigation.

The Wasserstein distance provides a convenient integration-error bound, as it is amenable to gradient-based minimization. However, the tightness of that bound is not well understood, especially in relation to the discrepancy-based bound given by the Koksma-Hlawka inequality. Exploring this relation could help better understand the optimization manifolds for future sampling patterns. Another interesting investigation would be the efficient minimization of discrepancy metrics.

5.10 Conclusion

We develop a theoretical point optimization framework designed for handling large numbers of objectives. Specifying these objectives for a given application can be tedious if done manually. Prior methods Wei (2010); Qin et al. (2017) have overlooked this issue as they target applications with fewer objectives.

We devise a principled framework for point optimization that can handle large numbers of objectives. We introduce the notion of a subclass which adds a level of granularity by specifying an objective for a subset of points in a class. Our framework scales to such a large number of objectives because, theoretically, the achievable quality does not depend on the number of objectives but on the amount of overlap between them and the difference in target distributions. The memory footprint of our stochastic gradient-descent optimization is negligible as we optimize a single subclass per iteration. We demonstrate a variety of applications such as stippling, object placement or progressive sampling.

We also formalized perceptual error optimization as a multi-class problem. Under this framework Blue noise error distribution is not a target but a consequence of the visual system modeling. Most importantly, this formulation enables theoretical error bounds, allowing us to assess not only per-pixel errors but also the distribution of errors across screen space. Using our optimizer, we achieved superior visual quality compared to existing methods. This improvement stems from optimization at the sample level, something prior approaches could not effectively achieve. Furthermore, the flexibility of our method makes it easily adaptable to new perceptual models and pixel filtering strategies.

Chapter 6
Importance Sampling For Gradient Estimation In ML

Monte Carlo gradient estimation is fundamental in optimizing complex, high-dimensional models within machine learning. This method's stochastic nature enables the estimation of gradients when analytical solutions are either intractable or too costly to compute. However, the high variance inherent in Monte Carlo methods can impede convergence and increase computational load during training. Addressing this variance is crucial for improving the efficiency and speed of machine learning algorithms.

This chapter explores two advanced techniques that utilize importance sampling to enhance the efficiency of Monte Carlo gradient estimation, thereby accelerating the training process of machine learning models. Importance sampling reduces variance by drawing samples from a distribution that closely matches the target distribution, leading to more accurate gradient estimates with fewer samples.

The first technique focuses on efficient importance sampling by utilizing an upper bound of the gradient norm. This method provides theoretical and empirical benefits in variance reduction. By selecting samples based on an upper bound of the gradient norm, the method ensures that the most significant samples for gradient estimation are prioritized. This reduces variance and accelerates training without introducing significant computational overhead. The method effectively balances sampling efficiency and computational cost, making it practical for various machine learning tasks.

The second technique introduces multiple importance sampling (MIS) for vector-valued gradient estimation, a concept widely used in light transport simulation. Optimal multiple importance sampling is particularly effective for multi-dimensional gradient estimates, where each dimension can be addressed independently to minimize variance. This approach significantly reduces variance compared to simple importance sampling by allowing for the independent solution of each gradient dimension. The use of optimal MIS results in faster convergence and, in simple cases with few parameters, can approximate a perfect gradient. This technique demonstrates substantial efficiency improvements, providing a robust solution to the variance challenges in Monte Carlo gradient estimation.

6.1 Efficient Importance Sampling And Adaptive Sampling For Gradient Estimation

Efficient importance sampling is crucial for reducing variance and speeding up training in Monte Carlo gradient estimation. This method introduces two key contributions: an efficient SGD-based algorithm with adaptive sampling and a straightforward yet effective sampling metric.

The first contribution is an efficient SGD-based algorithm for importance sampling and adaptive sampling. This algorithm maintains a memory of the relative importance of each data sample and updates it whenever a sample is evaluated during training. This adaptive approach allows the importance metric to evolve as training progresses, ensuring that the sampling strategy remains adapted even as the training dynamics change. Unlike existing resampling methods, this algorithm maximizes computational efficiency by using all samples, avoiding the need to discard any for importance sampling. This ensures full utilization of computational resources, enhancing overall training efficiency.

The second contribution is a simple and efficient sampling metric based on an upper bound of the gradient norm. While the ideal importance sampling metric is the full gradient norm, this is often impractical to compute directly. Instead, this method uses the norm of the derivative of the loss with respect to the network's output. This choice approximates the gradient norm effectively and is computationally feasible. Empirical evidence supports that this proxy is closely related to the actual gradient norm, making it a practical and efficient choice for importance sampling.

The combination of the adaptive algorithm and the efficient sampling metric results in significant gradient variance reduction. By continuously updating the importance metric and using a practical approximation of the gradient norm, this method ensures that the most relevant samples are prioritized throughout training. This reduces noise in gradient estimates, leading to faster and more stable convergence. Consequently, the training process accelerates, as less noisy gradients facilitate more efficient optimization.

6.1.1 Related Work

Gradient estimation serves as a cornerstone in the realm of machine learning, underpinning the optimization of models. In practical scenarios, computing the exact gradient is often unfeasible due to the sheer volume of data, leading to a reliance on mini-batch approximations. Improving these approximations to obtain more accurate estimates is an enduring challenge in the field. The ultimate goal is to accelerate gradient descent by using more accurate gradient estimates.

Importance Sampling Importance sampling serves as a mechanism for error reduction in mini-batch gradient estimation. Each data point is assigned a probability of being selected in each mini-batch, making some data more likely to be selected than others. Bordes et al. (2005) developed an online algorithm (LASVM), which uses importance sampling to train kernelized support vector machines. Several studies, including Zhao and Zhang (2015); Needell et al. (2014); Wang et al. (2017); Alain et al. (2015), have demonstrated that importance sampling proportional to the gradient norm is indeed the optimal sampling strategy. Hanchi et al. (2022) recently proposed deriving an importance sampling metric from the gradient norm of each data point, demonstrating favorable convergence properties and provable improvements under certain convexity conditions.

Estimating the gradient for each data point can be computationally intensive. Thus, the search for more efficient sampling strategies has led to the exploration of methods approximating the gradient norm with reduced computational costs. Methods like those proposed by Loshchilov and Hutter (2015) rank data based on their loss and use this rank to create an importance sampling strategy, giving higher importance to data with higher loss. Katharopoulos and Fleuret (2017) proposed importance sampling the loss function. Additionally, Dong et al. (2021) proposed a re-sampling-based algorithm to reduce the number of backpropagation computations, selecting a subset of data based on loss. Similarly, Zhang et al. (2023) proposed re-sampling based on multiple heuristics to reduce the number of backward propagations and focus on more influential data.

Katharopoulos and Fleuret (2018) introduced an upper bound to the gradient norm that can be used as an importance function, suggesting re-sampling data based on the importance computed at the last layer. These re-sampling methods reduce unnecessary backward propagations but still involve forward computation.

Adaptive Data Weighting And Sampling Adaptive weighting/sampling dynamically adjusts the contribution of individual data samples during optimization to prioritize those with greater influence on the gradient descent process. This adjustment involves either increasing or decreasing their weight contributions to the estimator. This weight can be apply by sampling more often highly weighted data without reducing their contribution or simply by multiplying randomly selected data by a weighting factor. Unlike importance sampling, adaptive sampling aim to accelerate convergence at the cost of introducing bias into the gradient estimation.

To compute adaptive weights within a mini-batch, Santiago et al. (2021) proposed a method maximizing the mini-batch's effective gradient. This strategic allocation of weights aligns data point contributions with the optimization objective, expediting convergence despite potential bias.

6.1.2 Monte Carlo Gradient Estimation

In machine learning, the goal is to find optimal set of parameters θ for a model function $m(x, \theta)$, with x a data sample, that minimize a loss function $\mathcal{L}$ over a dataset Ω. The optimization is typically expressed as

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, L_\theta, \text{where } L_\theta = \frac{1}{|\Omega|}\int_\Omega w(x)\mathcal{L}(m(x,\theta),y)\mathrm{d}(x,y) = \mathrm{E}\left[\frac{w(x)\mathcal{L}(m(x,\theta),y)}{p(x,y)}\right]. \tag{6.1}$$

The total loss L_θ can be interpreted in two ways. The analytical interpretation views it as the integral of the loss $\mathcal{L}$ over a data space Ω, normalized by the space's volume. In machine learning, the data space typically represents the training dataset and the normalization is its size. The second, statistical interpretation defines L_θ as the expected value of the loss $\mathcal{L}$ for a randomly selected data point, divided by the probability of selecting it. These two approaches are perfectly equivalent. Here we have an additional weight term w.It's new term we introduce to distinguish importance sampling and adaptive sampling. Energy preservation requires $E[w(x)] = 1$, though the weight may not be constant.

In practice, the minimization of the total loss L_θ is tackled via iterative gradient descent. At each step t, its gradient ∇L_{θ_t} with respect to the current model parameters θ_t is computed, and the parameters are updated as

$$\theta_{t+1} = \theta_t - \lambda\nabla L_{\theta_t}, \tag{6.2}$$

where $\lambda > 0$ is the learning rate. This iterative procedure can be repeated until convergence.

Gradient Estimator The parameter update step in eq. (6.13) involves evaluating the total-loss gradient ∇L_{θ_t}. This requires processing the entire dataset Ω at each of potentially many (thousands of) steps, making the optimization computationally infeasible. In practice one has to resort to mini-batch gradient descent which estimates the gradient from a small set $\{x_i\}_{i=1}^B \subset \Omega$ of randomly chosen data points in a Monte Carlo fashion:

$$\nabla L_\theta \approx \frac{1}{B}\sum_{i=1}^{B} w(x_i)\frac{\nabla\mathcal{L}(m(x_i,\theta),y_i)}{p(x_i,y_i)} = \langle\nabla L_\theta\rangle, \quad \text{with} \quad x_i \propto p(x_i). \tag{6.3}$$

Here, $\nabla\mathcal{L}(m(x_i,\theta),y_i)$ is the gradient (w.r.t. θ) of the loss function for sample x_i selected following a probability density function (pdf) p (or probability mass function in case of a discrete dataset). Setting the weighting function to $w(x_i) = 1$ makes $\langle\nabla L_\theta\rangle$ an unbiased estimator for the total loss, i.e., $\mathrm{E}[\langle\nabla L_\theta\rangle] = \nabla L_\theta$. Mini-batch gradient descent uses $\langle\nabla L_\theta\rangle$ in place of the true gradient ∇L_θ in eq. (6.13) to update the model parameters at every iteration. The batch size B is typically much smaller than the dataset, enabling practical optimization.

Theoretical Convergence Analysis Mini-batch gradient descent is known to be affected by Monte Carlo noise due to the gradient estimation (6.14). This noise can cause the parameter optimization trajectory to be erratic and slow down convergence. It arises from the varying contributions of different samples x_i to the estimate. In certain conditions, it is possible to express the convergence rate of such methods. Gower et al. (2019) demonstrated that for an L-smooth and μ-convex function, the convergence rate of mini-batch gradient descent with constant learning rate is given by:

$$\mathrm{E}\left[\|\theta_t - \theta^*\|^2\right] \leq (1-\lambda\mu)^t\,\|\theta_0 - \theta^*\|^2 + \frac{2\lambda\sigma^2}{\mu} \tag{6.4}$$

with $\sigma^2 = \mathrm{E}\left[\|\langle\nabla L_{\theta^*}\rangle\|^2\right] - \overbrace{\mathrm{E}\left[\|\langle\nabla L_{\theta^*}\rangle\|\right]^2}^{=0}$. The expected value of the gradient norm is zero for the optimal set of parameters θ^*, as the solution of the gradient descent is reached when the gradient converges to zero. This equation underscores the significance of minimizing variance in gradient estimation to enhance the convergence rate of gradient descent methods. While not universally applicable, it provides valuable insights into expected behavior when reducing estimation errors. Hence, refining gradient estimates is crucial for optimizing various learning algorithms, facilitating more efficient convergence towards optimal solutions. Our experimental evaluation comparing different methods in section 6.1.5 further supports this notion.

Sampling And Weighting Strategies In classical mini-batch gradient descent, all samples are given equal importance, selected and weighted uniformly. To efficiently reduce estimation error it is possible to focus training resources on some more important data using a non-uniform sample selection and weighting.

Typically, the selection of samples that go into a mini-batch is done with constant probability $p(x_i) = 1$. Importance sampling is a technique for using a non-uniform pdf to strategically pick samples to reduce estimator variance. A distribution p is valid if $p(x_i) = 0 \implies \nabla\mathcal{L}(m(x_i,\theta), y_i) = 0$ for all data x_i.

Adaptive sampling represents an alternative approach to data selection, where samples are chosen based on non-uniform sampling distributions without applying normalization to their contributions (i.e., dividing by the pdf in eq. (6.14)). In this method, each data point's weight, denoted as $w(x_i)$, is equal to the pdf of the sample $p(x, y)$. This term compensate of the division by the same factor. Consequently, adaptive sampling naturally focuses on data points with high sampling probabilities, effectively modifying the problem statement eq. (6.12).

Adaptive weighting employs a uniform sampling distribution, where $p(x_i) = 1$, but introduces an adaptive weight normalization factor $w(x_i)$. In contrast to adaptive sampling, which focuses on non-uniform data selection, adaptive weighting prioritizes data samples by assigning them varying levels of importance through the weighting. Much like its counterpart, this method results in a modification of the problem statement, allowing for the emphasis of specific data samples during the optimization process. If carefully chosen, this emphasis can significantly accelerate optimization convergence.

6.1.3 Efficient Importance And Adaptive Sampling Algorithm

We propose an algorithm to efficiently perform importance and adaptive sampling for mini-batch gradient descent, outlined in Algorithm 3. Similarly to Loshchilov and Hutter (2015) and Schaul et al. (2015), it is designed to use an importance function that relies on readily available quantities for each data point, introducing only negligible memory and computational overhead over classical uniform mini-batching.

We maintain a set of persistent *un-normalized importance* scalars $q = q_i{}_{i=1}^{|\Omega|}$, continually updated during optimization. Initially, we process all data points once in the first epoch to determine their initial importance (line 3) following algorithm 4. Subsequently, at

Algorithm 3 Mini-batch importance sampling for SGD.

```
1:  $\theta \leftarrow$ random parameter initialization
2:  $B \leftarrow$ mini-batch size, $N = |\Omega|$                         $\leftarrow$ Dataset size
3:  $q, \theta \leftarrow$ Initialize$(x, y, \Omega, \theta, B)$               $\leftarrow$ algorithm 4
4:  until convergence do                                        $\leftarrow$ Loop over epochs
5:      for $t \leftarrow 1$ to $N/B$ do                                $\leftarrow$ Loop over mini-batches
6:          $p \leftarrow q/\text{sum}(q)$                              $\leftarrow$ Normalize importance to pdf
7:          $x, y \leftarrow B$ data samples $\{x_i, y_i\}_{i=1}^{B} \propto p$
8:          $\mathcal{L}(x) \leftarrow \mathcal{L}(m(x, \theta), y)$
9:          $\nabla\mathcal{L}(x) \leftarrow$ Backpropagate$(\mathcal{L}(x))$
10:         $w(x)$    $\leftarrow$    (ImportanceSampling)    ?    $1$    :    $p(x)$
    $\leftarrow$ weight for adaptive or importance sampling
11:         $\langle\nabla L_\theta\rangle \leftarrow (\nabla\mathcal{L}(x) \cdot ({}^{w(x)}/{}_{p(x)})^T)/B$          $\leftarrow$ eq. (6.14)
12:         $\theta \leftarrow \theta - \eta\,\langle\nabla L_\theta\rangle$                       $\leftarrow$ SGD step
13:         $q(x) \leftarrow \gamma \cdot q(x) + (1-\gamma) \cdot \left\| \frac{\partial\mathcal{L}(x)}{\partial m(x,\theta)} \right\|$      $\leftarrow$ Accumulate importance
14:     end for
15:     $q \leftarrow q + \epsilon$
16: Return $\theta$
```

each mini-batch optimization step t, we normalize the importance values to obtain the probability density function (PDF) p (line 6), and use it to sample B data points with replacement (line 7). We then evaluate the loss for each selected data sample (line 8) and backpropagate to compute the corresponding loss gradient (line 9). Depending on whether we aim for importance sampling or adaptive sampling, we select the per-sample weight accordingly (line 10). For importance sampling, $w(x) = 1$, while for adaptive sampling, $w(x) = p(x)$. Finally, we update the network parameters using the estimated gradient (line 12). Additionally, we compute the sample importance for each data sample from the mini-batch and update the persistent importance q (line 13). Various importance heuristics such as the gradient norm Zhao and Zhang (2015); Needell et al. (2014); Wang et al. (2017); Alain et al. (2015), the loss Loshchilov and Hutter (2015); Katharopoulos and Fleuret (2017); Dong et al. (2021) or more advanced importance Katharopoulos and Fleuret (2018) can be implemented to replace our sampling metric in this line. To enhance efficiency, our algorithm reuses the forward pass computations made during line 8 to compute importance, updating q only for the current mini-batch samples. The weighting parameter α ensures weight stability as discussed in eq. (6.11).

At the end of each epoch (line 14), we add a small value to the un-normalized weights of all data to ensure that every data point will be eventually evaluated, even if its importance is deemed low by the importance metric.

It is importance to note that the initialization epoch (algorithm 4) is done without importance sampling to initialize each sample importance. This does not create overhead as it is equivalent to a classical epoch running over all data samples. While similar schemes have been proposed in the past, they often rely on a multitude of hyperparameters, making their practical implementation challenging. This has led to the development of alternative methods like re-sampling Katharopoulos and Fleuret (2018); Dong et al. (2021); Zhang et al. (2023). Our proposed sampling strategy has only a few hyperparameters. Tracking importance across batches and epochs minimizes the computational

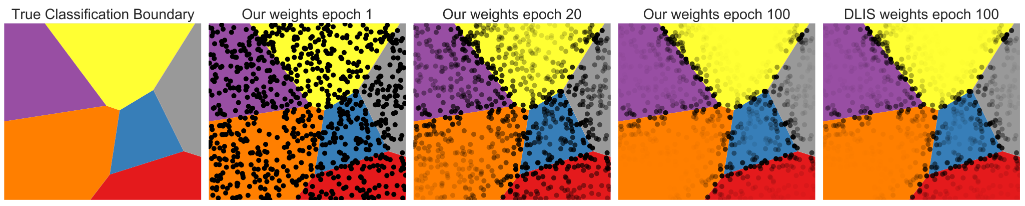


Figure 6.1: Visualization of the importance sampling at 3 different epoch and the underlying classification task. For each presented epoch, 800 data-point are presented with a transparency proportional to their weight according to our method.

overhead, further enhancing the efficiency and practicality of the approach.

Algorithm 4 Subroutine for initialization for algorithm 3

```
1:  Initialization($x$,$y$,$\Omega$,$\theta$,$B$,$q$):                ← Initialize $q$ in a classical SGD loop
2:      for $t \leftarrow 1$ to $|\Omega|/B$ do
3:          $x, y \leftarrow \{x_i, y_i\}_{i=(t-1)\cdot B+1}^{t\cdot B+1}$     ← See all samples in the first epoch
4:          $\mathcal{L}(x) \leftarrow \mathcal{L}(m(x,\theta), y)$
5:          $\nabla\mathcal{L}(x) \leftarrow$ Backpropagate$(\mathcal{L}(x))$
6:          $\langle\nabla L_\theta\rangle(x) \leftarrow \nabla\mathcal{L}(x)/B$        ← eq. (6.14)
7:          $\theta \leftarrow \theta - \eta\,\langle\nabla L_\theta\rangle(x)$            ← eq. (6.13)
8:          $q(x) \leftarrow \left\|\frac{\partial\mathcal{L}(x)}{\partial m(x,\theta)}\right\|$   ← Initialize per sample importance
9:      end for
10:     Return $q$,$\theta$
```

6.1.4 Loss-Gradient-Based Importance Function

Gradient Norm Upper Bound In combination with the presented algorithm, we propose an importance function that is efficient to evaluate. While the gradient L_2 norm has been shown to be optimal Zhao and Zhang (2015); Needell et al. (2014); Wang et al. (2017); Alain et al. (2015), calculating it can be computationally expensive as it requires full backpropagation for every data point. This choice minimizes the first term of the gradient variance, thereby bounding the convergence of eq. (6.4). Instead, we compute an upper bound of the gradient norm using the output nodes of the network: $q(x) = \left\|\frac{\partial\mathcal{L}(x)}{\partial m(x,\theta)}\right\|$. This upper bound of the gradient norm is derived from the chain rule and the Cauchy–Schwarz inequality:

$$\left\|\frac{\partial\mathcal{L}(x_i)}{\partial\theta}\right\| = \left\|\frac{\partial\mathcal{L}(x)}{\partial m(x,\theta)}\cdot\frac{\partial m(x,\theta)}{\partial\theta}\right\| \le \left\|\frac{\partial\mathcal{L}(x)}{\partial m(x,\theta)}\right\|\cdot\left\|\frac{\partial m(x,\theta)}{\partial\theta}\right\| \le \underbrace{\left\|\frac{\partial\mathcal{L}(x)}{\partial m(x,\theta)}\right\|}_{q(x)}\cdot C\,, \tag{6.5}$$

where C is the Lipschitz constant of the parameters gradient. That is, our importance function is a bound of the gradient magnitude based on the output-layer gradient norm.

For some specific task when the output layer has predictable shape, it is possible to derive a closed form definition of the proposed importance metric.

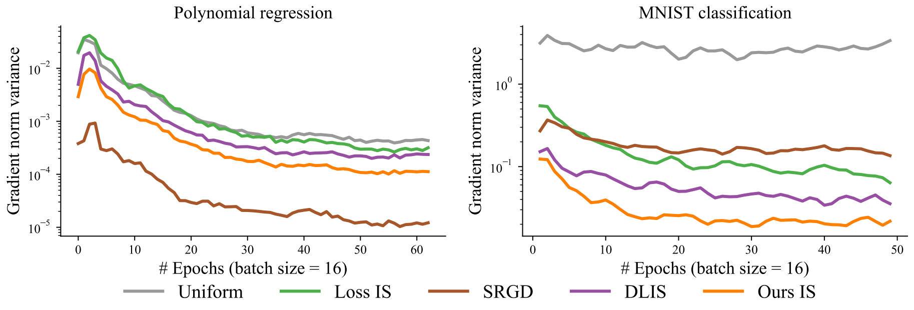


Figure 6.2: Evolution of gradient variance for various importance sampling strategies on polynomial regression and MNIST classification task. In both case the optimization is done on a 3 fully-connected layer network. Variance estimation is made of each method on the same network at each epoch. The variance is computed using a mini-batch of size 16.

Cross-Entropy Loss Gradient Cross entropy is a widely used loss function in classification tasks. It quantifies the dissimilarity between predicted probability distributions and actual class labels. Specifically, for a binary classification task, cross entropy is defined as:

$$\mathcal{L}(m(x_i,\theta)) = -\sum_{j=1}^{C} y_j \log(s_j) \text{ where } s_j = \frac{\exp(m(x_i,\theta)_j)}{\sum_{k=0}^{C}\exp(m(x_i,\theta)_k)} \tag{6.6}$$

where $m(x_i,\theta)$ is an output layer, x_i is the input data and C means the number of classes. It is possible to express the derivative of the loss $\mathcal{L}$ with respect to the network output $m(x_i,\theta)_j$ in a close form.

$$\frac{\partial \mathcal{L}_{\text{cross-ent}}}{\partial m(x_i,\theta)_j} = -\frac{\partial}{\partial m(x_i,\theta)_j}\left(\sum_i^C y_i \log s_i\right) = -\sum_i^C y_i \frac{\partial}{\partial m(x_i,\theta)_j}\log s_i \tag{6.7}$$

$$= -\sum_i^C \frac{y_i}{s_i}\frac{\partial s_i}{\partial m(x_i,\theta)_j} = -\sum_i^C \frac{y_i}{s_i} s_i \cdot (\mathbf{1}\{i == j\} - s_j) \tag{6.8}$$

$$= \sum_i^C y_i \cdot s_j - \sum_i^C y_i \cdot (\mathbf{1}\{i == j\}) = s_j \sum_i^C y_i - y_j = s_j - y_j \tag{6.9}$$

The partial derivative of the cross-entropy loss function wrt output layer parameters has the form:

$$\frac{\partial \mathcal{L}}{\partial m(x_i,\theta)_j} = s_j - y_j \tag{6.10}$$

This equation can be directly computed from the network output without any graph back-propagation. This make the computation of our importance function extremely cheap for classification tasks.

6.1.5 Convergence Analysis

Building on the theoretical bound defined in eq. (6.4), we proceed to examine the effects of various importance sampling methods on the gradient variance. Such variance influences the convergence of an optimization procedure. This equation relies on the ideal

Table 6.1: Average computation time on 3 layer fully-connected network for multiple sampling metric the task from fig. 6.2. Time is average over one epoch and computed on mini-batch of size 8.

Computation time (↓)	Loss	SRGD	DLIS	Ours autodiff	Ours analytic
Polynomial regression	$1.33\cdot10^{-4}$ $(1.\times)$	$7.17\cdot10^{-4}$ $(5.39\times)$	$4.38\cdot10^{-4}$ $(3.29\times)$	$3.23\cdot10^{-4}$ $(2.43\times)$	-
MNIST	$1.28\cdot10^{-4}$ $(1.\times)$	$5.57\cdot10^{-4}$ $(4.35\times)$	$3.27\cdot10^{-4}$ $(2.55\times)$	$3.59\cdot10^{-4}$ $(2.80\times)$	$1.40\cdot10^{-4}$ $(1.09\times)$

model parameters θ^*, but they cannot be practically calculated. Rather, we measure the gradient variance during training using a suboptimal parameter set.

Figure 6.2 displays the evolution of gradient variance using different strategies for polynomial regression and MNIST classification, both using a three-layer fully connected network. Each method is evaluated on the same network, trained using uniform sampling. This allows for a variances comparison of the gradient norm. We analyze five techniques: Uniform sampling, Loss-based importance sampling, SRGD Hanchi et al. (2022), DLIS Katharopoulos and Fleuret (2018), and our method. SRGD is an importance sampling technique using a conditioned minimization of gradient variance using memory of the gradient magnitude. This method has shown robust theoretical convergence properties in strongly convex scenarios. This variance reduction is visible on the polynomial regression task where it result in lower variance than other methods. However, for more complex tasks such as MNIST classification, SRGD underperforms all methods, suggesting scalability limitations to non-convex and complex problems. In contrast, our method consistently yields lower variance than Loss-based importance sampling and DLIS Katharopoulos and Fleuret (2018). These findings elucidate the results in section 6.1.6.

In addition, we provide evaluation times for each metric for both the polynomial regression and MNIST classification tasks in table 6.1. Clearly, SRGD Hanchi et al. (2022) demands more computational resources, even for small-scale networks comprising only three layers. This increased demand stems from its dependence on calculating the gradient norm for each individual data point. Both our metric, which employs automatic differentiation, and DLIS Katharopoulos and Fleuret (2018), incur comparable computational costs due to their reliance on derivatives from the final layers. Nonetheless, our approach proves to be the most efficient when analytical evaluations are feasible. Such differences in computational efficiency are likely to significantly influence the outcomes of comparisons made under equal-time conditions in later sections.

Zhao and Zhang (2015) have shown that importance weights w.r.t. the gradient norm gives the optimal sampling distribution. On the right inline figure, we show the difference between various weighting strategies and the gradient norm w.r.t. all parameters. In this experiment, all sampling weights are computed using the same network on an MNIST optimization task. Our proposed sampling strategies, based on the loss gradient are the closest approximation to the gradient norm.

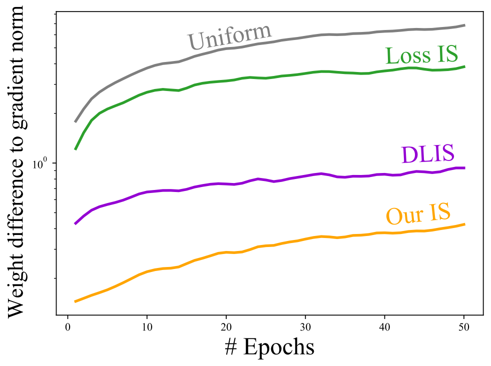


Figure 6.3: Comparison of different sampling importance compared to optimal on MNIST classifier.

6.1.6 Experiments

In this section, we delve into the experimental outcomes of our proposed algorithm and sampling strategy. Our evaluations encompass diverse classification tasks, spanning both importance sampling and adaptive sampling. We benchmarked our approach against those of Katharopoulos and Fleuret (2018) and Santiago et al. (2021), considering various variations in comparison. Distinctions in our comparisons lie in assessing performance at equal steps/epochs and equal time intervals. The results presented here demonstrate the loss and classification error, computed on test data that remained unseen during the training process.

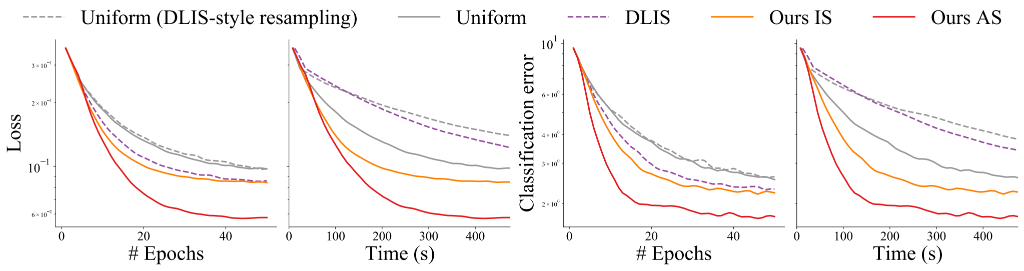


Figure 6.4: We compare loss and classification error metrics for the MNIST dataset between the resampling algorithm by Katharopoulos and Fleuret (2018) (DLIS) and our algorithm. At equal epochs, the resampling algorithm with importance sampling works better than uniform sampling for DLIS. However, at equal time, the resampling cost is too high, making DLIS even slower than standard uniform sampling. Our algorithm outperforms all existing methods.

6.1.6.1 Implementation Details

We implement our method and all baselines in a single PyTorch framework. Experiments run on a workstation with an NVIDIA GeForce RTX 2080 graphics card and an Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz. The baselines include uniform sampling, DLIS Katharopoulos and Fleuret (2018) and LOW Santiago et al. (2021). Uniform means

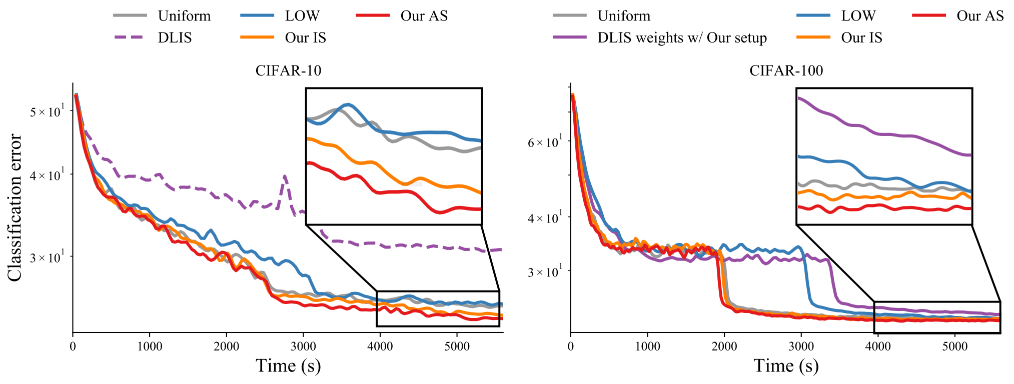


Figure 6.5: Comparison of convergence at equal time intervals on the CIFAR-10 dataset with a ViT network Dosovitskiy et al. (2020) and on the CIFAR-100 dataset with a ResNet-18 network He et al. (2016). We evaluate our adaptive and importance sampling techniques against DLIS Katharopoulos and Fleuret (2018) and Low Santiago et al. (2021). For CIFAR-10, the complete DLIS method is applied, while for CIFAR-100, our algorithm is utilized alongside their weighting approach. In both scenarios, we emphasize the differences in convergence by focusing on the latter part of the plot.

that we sample every data point from a uniform distribution. DLIS importance samples the data mainly depending on the norm of the gradient on the last output layer. We use functorch Horace He (2021) to accelerate this gradient computation. LOW is based on adaptive weighting that maximizes the effective gradient of the mini-batch using the solver from Vandenberghe (2010).

Weight Stability. Updating the persistent per-sample importance q directly sometime leads to a sudden decrease of accuracy during training. To make the training process more stable, we update q by linearly interpolating the importance at the previous and current steps:

$$q(x) = \alpha \cdot q_{prev}(x) + (1 - \alpha) \cdot q(x) \tag{6.11}$$

where α is a constant for all data samples. In practice, we use $\alpha \in \{0.0, 0.1, 0.2, 0.3\}$ as it gives the best trade-off between importance update and stability. This can be seen as a momentum evolution of the per-sample importance to avoid high variation. Utilizing an exponential moving average to update the importance metric prevents the incorporation of outlier values. This is particularly beneficial in noisy setups, like situations with a high number of class or a low total number of data.

MNIST Dataset. The MNIST database contains 60,000 training images and 10,000 testing images. We train a 3-layer fully-connected network (MLP) for image classification over 50 epochs with an Adam optimizer Kingma and Ba (2014).

CIFAR Dataset. CIFAR-10 Krizhevsky et al. (2009) contains 60,000 32x32 color images from 10 different object classes, with 6,000 images per class. CIFAR-100 Krizhevsky et al. (2009) has 100 classes containing 600 images each, with 500 training images and 100 testing images per class. For both datasets, we train the same ResNet-18 network He et al. (2016). We use the SGD optimizer with momentum 0.9, initial leaning rate 0.01, and batch size 64. We divide the initial learning rate by 10 after 70 epochs for CIFAR-10 and train the network for a total of 100 epochs. Additionally, we trained a Vision Transformer (ViT) Dosovitskiy et al. (2020) on CIFAR-10 using the Adam optimizer with an initial learning rate 0.0001 and a cosine annealing scheduler Loshchilov and Hutter (2016). For CIFAR-100, we divide the learning rate by 10 after 100, 200 epochs and train for a total of 300 epochs. For both datasets, we use random horizontal flip and random crops to augment the data on the fly.

Point-Cloud Classification Dataset. We train a PointNet Qi et al. (2017) with 3 shared-MLP layers and one fully-connected layer, on the ModelNet40 dataset Wu et al. (2015). The dataset contains point clouds from 40 categories. The data are split into 9,843 for training and 2,468 for testing. Each point cloud has 1,024 points. We use the Adam optimizer Kingma and Ba (2014), with batch size 64, weight decay 0.001, initial learning rate 0.00002 divided by 10 after 100, 200 epochs. We train for 300 epochs in total.

Oxford 102 Flower Dataset. The Oxford 102 flower dataset Nilsback and Zisserman (2008) contains flower images from 102 categories. We follow the same experiment setting of Zhang et al. (2017, 2019). We use the original test set for training (6,149 images) and the original training set for testing (1,020 images). In terms of network architecture, we use the pre-trained VGG-16 network Simonyan and Zisserman (2014) for feature extraction and only train a two-layer fully-connected network from scratch for classification. We use the Adam optimizer Kingma and Ba (2014) with a learning rate 0.001 and train the two-layer fully-connected network for 100 epochs.

6.1.6.2 Results

In fig. 6.11, we compare our algorithm against DLIS Katharopoulos and Fleuret (2018). DLIS applies resampling to both uniform and their importance sampling. This increases the overall computation overhead for their approach. Standard uniform sampling is much faster than the resampling approach. We assess cross-entropy loss and classification error in terms of epoch count and equal time. The DLIS method shows similar performance to ours at equal epochs but incurs high computational costs due to the need for a large dataset forward pass using the re-sampling algorithm.

In fig. 6.5, we show the convergence classification error at equal time for CIFAR-10 and CIFAR-100. For CIFAR-10, we compare with Uniform sampling, LOW Santiago et al. (2021), and DLIS full methods Katharopoulos and Fleuret (2018). For CIFAR-100, we consider our algorithm using DLIS weights. Zoom-ins of the curves are shown at the end of the plot. In both cases, our adaptive sampling method achieves the lowest error, followed by our importance sampling variant. LOW does not significantly outperform Uniform sampling due to its overhead. Similarly, on CIFAR-100, DLIS with our algorithm maintains a high overhead, but our importance sampling algorithm still shows a clear advantage. On CIFAR-10, the DLIS full method fails to converge. This issue is consistent across different architectures, as shown in fig. 6.6. The figure compare uniform sampling, DLIS, LOW, and our importance and adaptive sampling strategies for CIFAR-10 and point cloud classification tasks. Our adaptive sampling excels in classification error across epochs and optimization time, especially when compared at equal time due to our low overhead compared to DLIS and LOW. The poor convergence of DLIS is due to resampling, which discards many "less important" data samples, causing the training to focus on a subset of the dataset and overfit.

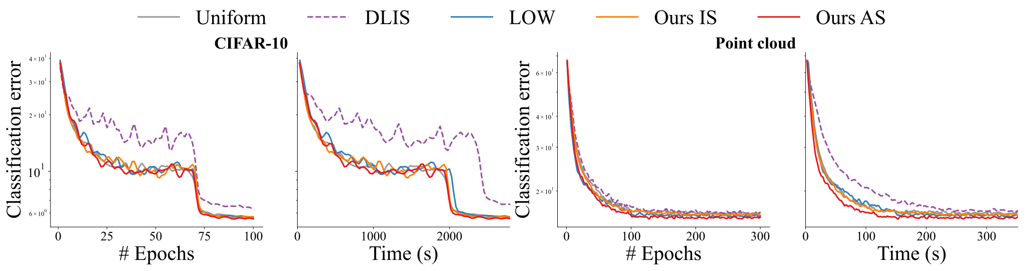


Figure 6.6: When comparing on CIFAR-10 and Point cloud ModelNet40 Wu et al. (2015) classification datasets, DLIS performs poorly at equal time due to the resampling overhead. Unlike DLIS Katharopoulos and Fleuret (2018), we use standard uniform sampling which is faster. We also compare against another adaptive scheme by Santiago et al. (2021) (LOW). Our adaptive (Ours AS) and importance sampling (Ours IS) shows improvements on the ModelNet40 dataset against other methods. Overall, our adaptive variant achieves lower classification errors with minimal overhead compared to others.

We conduct a similar experiment on the Oxford flower classification task in fig. 6.7). This dataset is known for its complexity due to a large number of classes with few images per class. Our method employing adaptive sampling again emerged as the top performer. Notably, DLIS exhibits under-performance, likely due to the challenges of re-sampling in a dataset with a high number of classes with only (10 data samples per class). With this distribution of data, re-sampling does not achieve a good estimation of the gradient. This causes visible over-fitting in the convergence curve. This highlights the robustness of our sampling metric as well as the use of memory based algorithm.

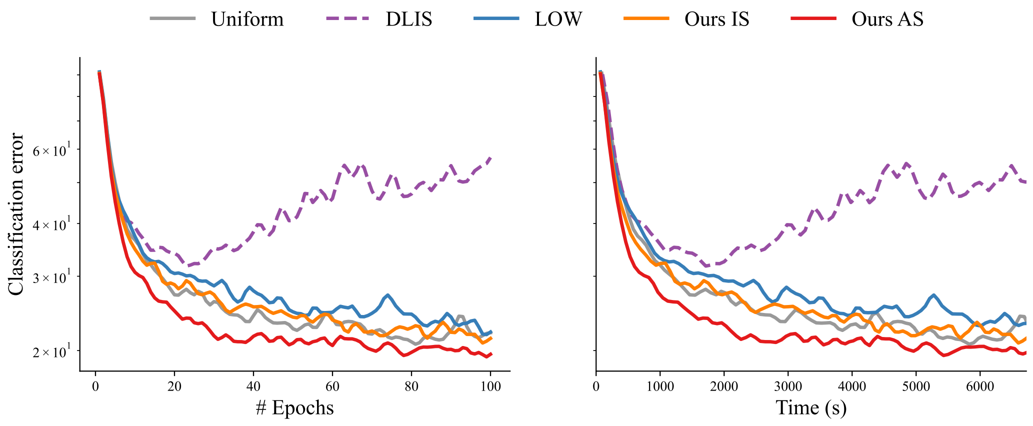


Figure 6.7: On Oxford flower 102 classification dataset Nilsback and Zisserman (2008), where the number of classes are high, our approach shows significant improvement compared to other methods. DLIS performs worse due to the sparsity in the data, which hampers their resampling strategy.

We also tested our importance strategy of regression task in fig. 6.8. In this task, we trained a 5-layer SIREN network Sitzmann et al. (2020) to predict RGB values from given

2D pixel coordinates. The left side of the figure illustrates the loss evolution wrt the training epochs. On the right, we showcase the reference image we train the network on, the error image (difference from the reference), and a crop for both Uniform sampling and our importance sampling method. To use our method in this example with an L_2 norm loss function, we employed autograd computation with respect to the output of the network.

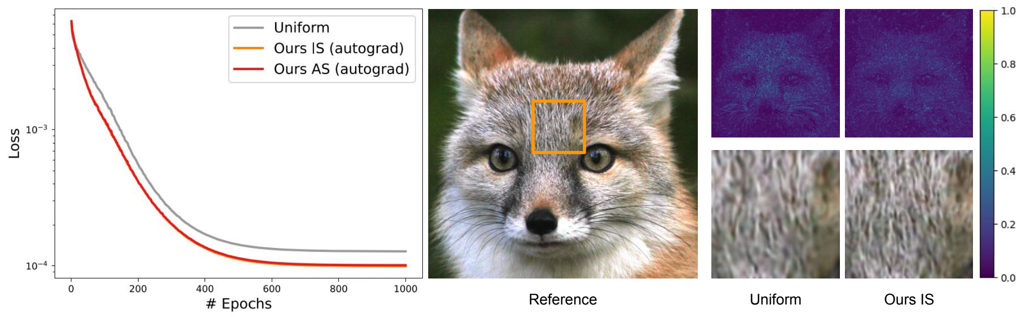


Figure 6.8: Image regression with a 5-layer SIREN network Sitzmann et al. (2020) trained to predict RGB values from 2D pixel coordinates. Our autograd-based importance and adaptive sampling strategies demonstrate clear improvements over uniform sampling in equal-epoch loss curves. Additionally, we present error maps and zoom-in results obtained using both uniform sampling and our importance sampling. Further equal-time comparisons are available in the appendix.

Discussion. Both our and DLIS importance metrics are highly correlated but ours is simpler and efficient to evaluate. Even with a slightly better importance sampling metric, most of the improvement come from the memory based algorithm instead of a resampling one. The resulting algorithm gives higher performance at equal time and has more stable convergence.

Our adaptive sampling achieve better convergence properties than LOW. Adaptive sampling generally outperforms adaptive weighting as mini-batch are composed with multiple high importance data instead of a good weighting of randomly selected data. Thus the average in eq. (6.14) is done with more contributing data. Additionally, our loss derivative-based weighting is closer to the gradient norm than loss-based importance.

Limitations. As the algorithm rely on past information to drive a non-uniform sampling of data, it requires seeing the same data multiple times. This creates a bottleneck for architectures that rely on progressive data streaming. More research is needed to design importance sampling algorithms for data streaming architectures, which is a promising future direction. Non-uniform data sampling can also create slower runtime execution. The samples selected in a mini-batch are not laid out contiguously in memory leading to a slower loading. We believe a careful implementation can mitigate this issue.

6.1.7 Conclusion

In conclusion, our work introduces an efficient sampling strategy for machine learning optimization, including both importance and adaptive sampling. This strategy, which relies on the gradient of the loss and has minimal computational overhead, was tested across various classification as well as regression tasks with promising results. Our work demonstrates that by paying more attention to samples with critical training information, we can speed up convergence without adding complexity. We hope our findings will encourage further research into simpler and more effective importance/adaptive sampling strategies for machine learning.

6.2 Multiple Importance Sampling For Gradient Estimation

In the previous section, we explored how importance sampling can be leveraged to improve gradient estimation in stochastic optimization, leading to faster convergence in training machine learning models. Crucially, we demonstrated that our method yields sampling strategies that closely approximate the optimal distribution for gradient estimation. However, a fundamental limitation of classical importance sampling is its inability to simultaneously align with all partial derivatives of a model's parameters. A single sampling distribution cannot be optimal for each component of the gradient vector simultaneously.

To address this, we propose a novel method that goes beyond the standard importance sampling framework. Our approach embraces the idea of using a weighted combination of multiple sampling distributions through the lens of *multiple importance sampling* section 2.2.2. Instead of attempting to find a single optimal distribution, we compute a set of distributions, each potentially better aligned with different components of the gradient, and adaptively combine them using MIS. This allows the estimation process to reduce the influence of distributions poorly correlated with certain gradients, while amplifying the contribution of those better aligned. Crucially, these weights must be adapted *per gradient component*, as each parameter may benefit from a different sampling strategy.

We introduce an efficient MIS-based algorithm for gradient estimation that avoids costly resampling steps, unlike earlier approaches such as Katharopoulos and Fleuret (2018). Our method uses a self-adaptive metric to handle the evolution of gradient across training stages and extends importance sampling to the *vector-valued* setting, where gradients span multiple parameters. Inspired by robust techniques from rendering in computer graphics Veach (1997), we adopt and extend the theory of *optimal multiple importance sampling* (OMIS) Kondapaneni et al. (2019) to the context of gradient estimation. By computing adaptive, per-sample weights across multiple proposal distributions, our method yields significantly improved gradient estimates and accelerates convergence.

We summarize our key contributions as follows:

- We introduce a novel multiple importance sampling estimator tailored for vector-valued gradient estimation.
- We propose a practical and efficient strategy for computing optimal MIS weights,

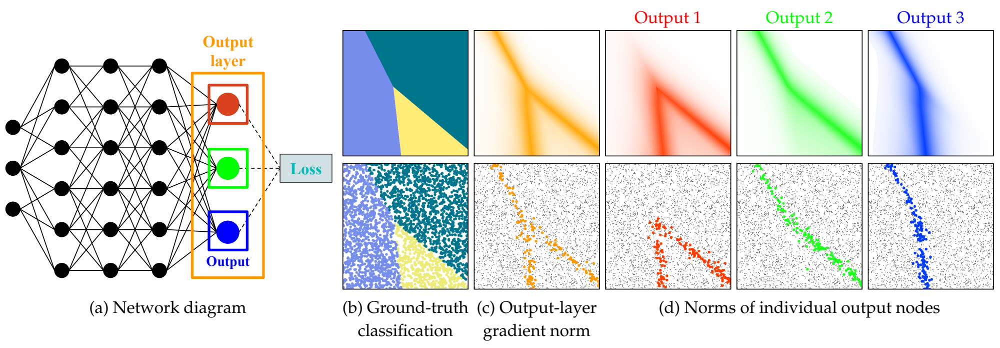

Output layer
Output
Loss
Output 1
Output 2
Output 3
(a) Network diagram
(b) Ground-truth classification
(c) Output-layer gradient norm
(d) Norms of individual output nodes

Figure 6.9: We visualize different importance sampling distributions for a simple classification task. We propose to use the output layer gradients for importance sampling, as shown in the network diagram (a). For a given ground-truth classification (top) and training dataset (bottom) shown in (b), it is possible to importance sample from the L_2 norm of the output-layer gradients (c) or from three different sampling distributions derived from the gradient norms of individual output nodes (d). The bottom row shows sample weights from each distribution.

enabling per-parameter adaptation.

- We demonstrate the effectiveness of our approach across several machine learning tasks, showing faster convergence and improved optimization performance.

6.2.1 Related Work

Importance Sampling For Gradient Estimation. Importance sampling (IS) Kahn (1950); Kahn and Marshall (1953); Owen and Zhou (2000b) has emerged as a powerful technique in high energy physics, Bayesian inference, rare event simulation for finance and insurance, and rendering in computer graphics. In the past few years, IS has also been applied in machine learning to improve the accuracy of gradient estimation and enhance the overall performance of learning algorithms Zhao and Zhang (2015).

By strategically sampling data points from a non-uniform distribution, IS effectively focuses training resources on the most informative and impactful data, leading to more accurate gradient estimates. Bordes et al. (2005) developed an online algorithm (LASVM) that uses importance sampling to train kernelized support vector machines. Loshchilov and Hutter (2015) suggested employing data rankings based on their respective loss values. This ranking is then employed to create an importance sampling strategy that assigns greater importance to data with higher loss values. Katharopoulos and Fleuret (2017) proposed importance sampling the loss function. Subsequently, Katharopoulos and Fleuret (2018) introduced an upper bound to the gradient norm that can be employed as an importance function. Their algorithm involves resampling and computing gradients with respect to the final layer. Despite the importance function demonstrating improvement over uniform sampling, their algorithm exhibits significant inefficiency.

Multiple importance sampling. The concept of Multiple Importance Sampling (MIS) emerged as a robust and efficient technique for integrating multiple sampling strategies Owen and Zhou (2000b). Its core principle lies in assigning weights to multiple importance sampling estimator, each using a different sampling distribution, allowing each data sample to utilize the most appropriate strategy. Veach (1997) introduced this concept of MIS to rendering in computer graphics and proposed the widely adopted *balance heuristic* for importance (weight) allocation. The balance heuristic determines weights based on a data sample's relative importance across all sampling approaches, effectively mitigating the influence of outliers with low probability densities. While MIS is straightforward to implement and independent of the specific function, Variance-Aware MIS Grittmann et al. (2019) advanced the concept by using variance estimates from each sampling technique for further error reduction. Moreover, Optimal MIS Kondapaneni et al. (2019) derived optimal sampling weights that minimize MIS estimator variance. Notably, these weights depend not only on probability density but also on the function values of the samples. Section 2.2.2 summarizes the theory behind multiple importance sampling. It also states the optimal MIS estimator and how to compute it.

6.2.2 Problem Statement

The primary goal of machine-learning optimization is to find the optimal parameters θ for a given model function $m(x, \theta)$ by minimizing a loss function $\mathcal{L}$ over a dataset Ω:

$$\theta^* = \operatorname*{argmin}_{\theta} \underbrace{\int_{\Omega} \mathcal{L}(m(x_i, \theta), y)\, \mathrm{d}x.}_{L_\theta} \tag{6.12}$$

The loss function $\mathcal{L}$ quantifies the dissimilarity between the model predictions $m(x, \theta)$ and observed data y. In the common case of a discrete dataset, the integral becomes a sum.

In practice, the total loss is minimized via iterative gradient descent. In each iteration t, the gradient ∇L_{θ_t} of the loss with respect to the current model parameters θ_t is computed, and the parameters are updated as

$$\theta_{t+1} = \theta_t - \lambda \underbrace{\int_{\Omega} \nabla\mathcal{L}(m(x, \theta_t), y)\, \mathrm{d}x,}_{\nabla L_{\theta_t}} \tag{6.13}$$

where $\lambda > 0$ is the learning rate. It is also possible to use an adaptive learning rate instead of a constant.

Monte Carlo gradient estimator. In practice, the parameter gradient is estimated from a small batch $\{x_i\}_{i=1}^{B}$ of randomly selected data points:

$$\langle \nabla L_\theta \rangle = \sum_{i=1}^{B} \frac{\nabla\mathcal{L}(m(x_i, \theta), y_i)}{Bp(x_i)} \approx \nabla L_\theta, \quad x_i \sim p. \tag{6.14}$$

The data points are sampled from a probability density function (pdf) p or probability mass function in discrete cases. The mini-batch gradient descent substitutes the true gradient ∇L_{θ_t} with an estimate $\langle \nabla L_{\theta_t} \rangle$ in eq. (6.13) to update the model parameters in each iteration.

We want to estimate ∇L_{θ_t} accurately and also efficiently, since the gradient-descent iteration (6.13) may require many thousands of iterations until the parameters converge. These goals can be achieved by performing the optimization in small batches whose samples are chosen according to a carefully designed distribution p. For a simple classification problem, fig. 6.9c shows an example importance sampling distribution derived from the output layer of the model. In fig. 6.9d we derive multiple distributions from the individual output nodes. Below we develop theory and practical algorithms for importance sampling using a single distribution (**??**) and for combining multiple distributions to further improve gradient estimation (section 6.2.3).

6.2.3 Multiple Importance Sampling

The parameter gradient ∇L_θ is vector with dimension equal to the number of model parameters. The individual parameter derivatives vary uniquely across the data points, and estimation using a single distribution (section 6.1) inevitably requires making a trade-off, e.g., only importance sampling the overall gradient magnitude. Truly minimizing the estimation error requires estimating each derivative using a separate importance sampling distribution tailored to its variation. However, there are two practical issues with this approach: First, it would necessitate sampling from all of these distributions, requiring "mini-batches" of size equal at least to the number of parameters. Second, it would lead to significant computation waste, since backpropagation computes all parameter derivatives but only one of them would be used per data sample. To address this issue, we propose using a small number of distributions, each tailored to the variation of a parameter subset, and combining *all* computed derivatives into a low-variance estimator, using multiple importance sampling theory. As an example, fig. 6.9d shows three sampling distributions for a simple classification task, based on the derivatives of the network's output nodes, following the boundary of each class.

MIS Gradient Estimator. Combining multiple sampling distributions into a single robust estimator has been well studied in the Monte Carlo rendering literature. The best known method is *multiple importance sampling* (MIS) Veach (1997). In our case of gradient estimation, the MIS estimator takes for form

$$\langle \nabla L_\theta \rangle_{\mathrm{MIS}} = \sum_{j=1}^{J} \sum_{i=1}^{n_j} w_j(x_{ij}) \frac{\nabla \mathcal{L}(m(x_{ij}, \theta), y_{ij})}{n_j p_j(x_{ij})}, \tag{6.15}$$

where J is the number of sampling distributions, n_j the number of samples from distribution j, and x_{ij} the i^{th} sample from the j^{th} distribution. Each sample is modulated by a weight $w_j(x_{ij})$; the estimator is unbiased as long as $\sum_{j=1}^{J} w_j(x) = 1$ for every data point x in the dataset.

Optimal Weighting. Various MIS weighting functions w_j have been proposed in literature, the most universally used one being the balance heuristic Veach (1997). In this work we use the recently derived optimal weighting scheme Kondapaneni et al. (2019) which minimizes the estimation variance for a given set of sampling distributions p_j:

$$w_j(x) = \alpha_j \frac{p_j(x)}{\nabla \mathcal{L}(m(x, \theta), y)} + \frac{n_j p_j(x)}{\sum_{k=1}^{J} n_k p_k(x)} \left(1 - \frac{\sum_{k=1}^{J} \alpha_k p_k(x)}{\nabla \mathcal{L}(m(x, \theta), y)} \right). \tag{6.16}$$

Here, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_J]$ is the solution to the linear system

$$\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{b}, \text{ with } \begin{cases} a_{j,k} = \displaystyle\int_\Omega \frac{p_j(x)p_k(x)}{\sum_i^J n_i p_i(x)} d(x,y), \\ b_j = \displaystyle\int_\Omega \frac{p_j(x)\nabla\mathcal{L}(m(x,\theta),y)}{\sum_i^J n_i p_i(x)} d(x,y), \end{cases} \tag{6.17}$$

where $a_{j,k}$ and b_j are the elements of the matrix $\boldsymbol{A} \in \mathbb{R}^{J\times J}$ and vector $\boldsymbol{b} \in \mathbb{R}^J$ respectively.

Instead of explicitly computing the optimal weights in eq. (6.16) using eq. (6.17) and plugging them into the MIS estimator (6.15), we can use a shortcut evaluation that yields the same result Kondapaneni et al. (2019):

$$\langle \nabla L_\theta \rangle_{\text{OMIS}} = \sum_{j=1}^{J} \alpha_j. \tag{6.18}$$

In section 2.2.2 we provide an overview of MIS and the aforementioned weighting schemes. Importantly for our case, the widely adopted balance heuristic does not bring practical advantage over single-distribution importance sampling (section 6.1) as it is equivalent to sampling from a mixture of the given distributions; we can easily sample from this mixture by explicitly averaging the distributions into a single one. In contrast, the optimal weights are different for each gradient dimension as they depend on the gradient value.

Practical Algorithm. Implementing the optimal-MIS estimator (6.18) amounts to drawing n_j samples from each distribution, computing $\boldsymbol{\alpha}$ for each dimension of the gradient and summing its elements. The integrals in $\boldsymbol{A}$ and $\boldsymbol{b}$ (sums in the discrete-dataset case) can be estimated as $\langle \boldsymbol{A} \rangle$ and $\langle \boldsymbol{b} \rangle$ from the drawn samples, yielding the estimate $\langle \boldsymbol{\alpha} \rangle = \langle \boldsymbol{A} \rangle^{-1} \langle \boldsymbol{b} \rangle$.

Algorithm 5 shows a complete gradient-descent algorithm. The main differences with algorithm 3 are the use of multiple importance distributions $\boldsymbol{q} = \{q_j\}_{j=1}^J$ (line 5) and the linear system used to compute the OMIS estimator (line 6). This linear system is updated (lines 15-18) using the mini-batch samples and solved to obtain the gradient estimation (line 22). Since the matrix $\langle \boldsymbol{A} \rangle$ is independent of the gradient estimation (see eq. (6.17)), its inversion can be shared across all parameter estimates.

Momentum-Based Linear-System Estimation. If the matrix estimate $\langle \boldsymbol{A} \rangle$ is inaccurate, its inversion can be unstable and yield a poor gradient estimate. The simplest way to tackle this problem is to use a large number of samples per distribution, which produces a accurate estimates of both $\boldsymbol{A}$ and $\boldsymbol{b}$ and thus a stable solution to the linear system. However, this approach is computationally expensive. Instead, we keep the sample counts low and reuse the estimates from previous mini-batches via momentum-based accumulation, shown in lines 17–18, where β is the parameter controlling the momentum; we use $\beta = 0.7$. This accumulation provides stability, yields an estimate of the momentum gradient Rumelhart et al. (1986), and allows us to use 1–4 samples per distribution in a mini-batch.

Algorithm 5 Optimal multiple importance sampling SGD.

```
1:  $\theta \leftarrow$ random parameter initialization
2:  $B \leftarrow$ mini-batch size, $J \leftarrow$ number of pdf
3:  $N = |\Omega| \leftarrow$ dataset size
4:  $n_j \leftarrow$ sample count per technique, for $j \in \{1, ..J\}$
5:  $\boldsymbol{q}, \theta \leftarrow$ InitializeMIS$(x, y, \Omega, \theta, B)$
6:  $\langle \boldsymbol{A} \rangle \leftarrow 0^{J \times J}$, $\langle \boldsymbol{b} \rangle \leftarrow 0^J$    $\leftarrow$ OMIS linear system
7:  until convergence do    $\leftarrow$ Loop over epochs
8:      for $t \leftarrow 1$ to $N/B$ do    $\leftarrow$ Loop over mini-batches
9:          $\langle \boldsymbol{A} \rangle \leftarrow \beta \langle \boldsymbol{A} \rangle$, $\langle \boldsymbol{b} \rangle \leftarrow \beta \langle \boldsymbol{b} \rangle$
10:         for $j \leftarrow 1$ to $J$ do    $\leftarrow$ Loop over distributions
11:             $p_j \leftarrow q_j / \text{sum}(q_j)$
12:             $x, y \leftarrow B$ data samples $\{x_i, y_i\}_{i=1}^{n_j} \propto p_j$
13:             $\mathcal{L}(x) \leftarrow \mathcal{L}(m(x, \theta), y)$
14:             $\nabla \mathcal{L}(x) \leftarrow$ Backpropagate$(\mathcal{L}(x))$
15:             $S(x) \leftarrow \sum_{k=1}^{J} n_k p_k(x)$
16:             $\boldsymbol{W} \leftarrow {n_i p_i(x)} / {\sum_{k=1}^{J} n_k p_k(x)}$
17:             $\langle \boldsymbol{A} \rangle \leftarrow \langle \boldsymbol{A} \rangle + (1-\beta) \sum_{i=1}^{n_j} \boldsymbol{W}_i \boldsymbol{W}_i^T$    $\leftarrow$ Momentum estim.
18:             $\langle \boldsymbol{b} \rangle \leftarrow \langle \boldsymbol{b} \rangle + (1-\beta) \sum_{i=1}^{n_j} \nabla \mathcal{L}(x_i) \boldsymbol{W}_i / S(x_i)$
19:             $\boldsymbol{q}(x) \leftarrow \gamma \boldsymbol{q}(x) + (1-\gamma) \frac{\partial \mathcal{L}(x)}{\partial m(x, \theta)}$
20:         end for
21:         $\langle \boldsymbol{\alpha} \rangle \leftarrow \langle \boldsymbol{A} \rangle^{-1} \langle \boldsymbol{b} \rangle$
22:         $\langle \nabla L_\theta \rangle_{\text{OMIS}} \leftarrow \sum_{j=1}^{J} \langle \alpha_j \rangle$
23:         $\theta \leftarrow \theta - \eta \, \langle \nabla L_\theta \rangle_{\text{OMIS}}$    $\leftarrow$ SGD step
24:     end for
25:
26: Return $\theta$
```

Importance Functions. To define our importance distributions, we expand on the approach from section 6.1. Instead of taking the norm of the entire output layer of the model, we take the different gradients separately as $\boldsymbol{q}(x) = \frac{\partial \mathcal{L}(x)}{\partial m(x,\theta)}$ (see fig. 6.9d). Similarly to algorithm 3, we apply momentum-based accumulation of the per-data importance (line 19 in algorithm 5). If the output layer has more nodes than the desired number J of distributions, we select a subset of the nodes. Many other ways exist to derive the distributions, e.g., clustering the nodes into J groups and taking the norm of each; we leave such exploration for future work.

6.2.4 Experiments

Implementation Details. We evaluate our importance sampling (IS) and optimal multiple importance sampling (OMIS) methods on a set of classification and regression tasks with different data modalities (images, point clouds). We compare them to classical SGD (which draws mini-batch samples uniformly without replacement), DLIS Katharopoulos and Fleuret (2018), and LOW Santiago et al. (2021). DLIS uses a resampling scheme that samples an initial, larger mini-batch uniformly and then selects a fraction of them for backpropagation and a gradient step. This resampling is based on an importance sampling metric computed by running a forward pass for each initial sample. LOW applies adaptive weighting to uniformly selected mini-batch samples to give importance

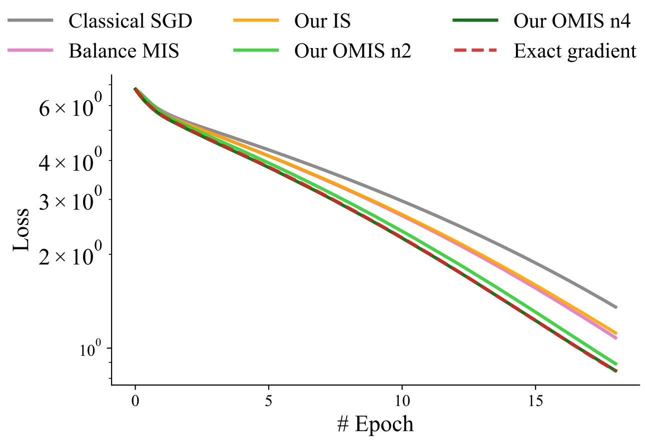


Figure 6.10: Convergence comparison of polynomial regression of order 6 using different method. Exact gradient show a gradient descent as baseline and classical SGD. For our method, we compare importance sampling and OMIS using $n = 2$ or 4 importance distributions. Balance heuristic MIS is also visible. Our method using OMIS achieve same convergence as exact gradient.

to data with high loss. All reported metrics are computed on data unseen during training, with the exception of the regression tasks. All experiments are conducted on a single NVIDIA Tesla A40 graphics card.

Classification. The classification tasks include image classification (MNIST Deng (2012), CIFAR-10/100 Krizhevsky et al. (2009) and point-cloud (ModelNet40 Wu et al. (2015)) classification.

The MNIST database contains 60,000 training images and 10,000 testing images. We train a 3-layer fully-connected network (MLP) for MNIST over 50 epochs with an Adam optimizer Kingma and Ba (2014). CIFAR-10, introduced by Krizhevsky et al. (2009), is a dataset that consists of 60,000 color images of size 32x32. These images belong to 10 different object classes, each class having 6,000 images. On the other hand, CIFAR-100 Krizhevsky et al. (2009) contains 100 classes with 600 images each. For each class, there are 500 training images and 100 testing images. In our experiments, we train the ResNet-18 network He et al. (2016) on both datasets. We apply random horizontal flip and random crops to augment the data during training. ModelNet40 contains 9,843 point clouds for training and 2,468 for testing. Each point cloud has 1,024 points. We train a PointNet Qi et al. (2017) with 3 shared MLP layers and 2 fully-connected layers for 300 epochs on point-cloud classification. We use the Adam optimizer Kingma and Ba (2014), with batch size 64, weight decay 0.001, initial learning rate 0.00002 divided by 2 after 100, 200 epochs.

Regression. Polynomial regression consists of optimizing the coefficients of a 1D polynomial of a given order to fit randomly drawn data from a reference polynomial of the same order. The reference data are generated on the interval $[-2; 2]$. Optimization is done using an Adam optimizer Kingma and Ba (2014) with a mini-batch size of 32 elements.

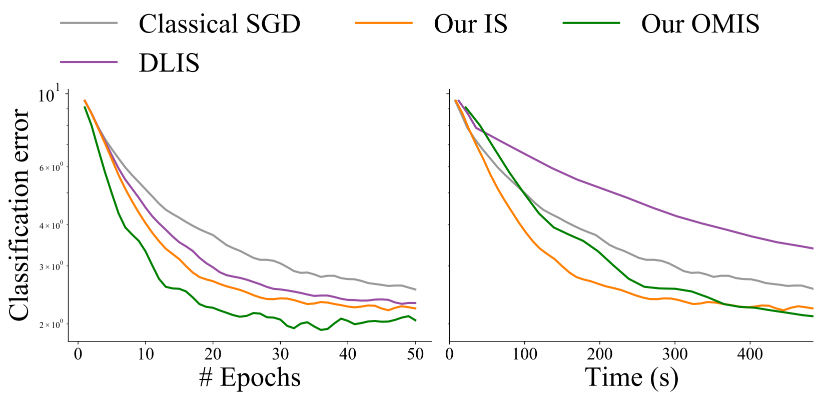


Figure 6.11: Classification error convergence for MNIST classification for various methods. Both Katharopoulos and Fleuret (2018) (DLIS) and resampling SGD approach. In comparison, our two method use the presented algorithm without resampling. It is visible that while DLIS perform similarly to our IS at equal epoch, the overhead of the method makes ours noticeably better at equal time for our IS and OMIS.

The image regression task consists in learning the mapping between a 2D coordinate input (pixel coordinate) and the 3-color output of the image for this pixel. We use a network with 5 fully-connected layers associated with positional encodings using SIREN activations Sitzmann et al. (2020). The training is done over 500 epoch using an Adam Kingma and Ba (2014) optimizer and each mini-batch is composed of 256 pixels for a 512^2 reference image.

Convex Problem. We performed a basic convergence analysis of IS and OMIS on a convex polynomial-regression problem. Figure 6.10 compares classical SGD, our IS, and three MIS techniques: balance heuristic Veach (1997) and our OMIS using two and four importance distributions. The exact gradient serves as a reference point for optimal convergence. Balance-heuristic MIS exhibits similar convergence to IS. This can be attributed to the weights depending solely on the relative importance distributions, disregarding differences in individual parameter derivatives. This underscores the unsuitability of the balance heuristic as a weighting method for vector-valued estimation. Both our OMIS variants achieve convergence similar to that of the exact gradient. The four-distribution variant achieves the same quality as the exact gradient using only 32 data samples per mini-batch. This shows the potential of OMIS to achieve low error in gradient estimation even at low mini-batch sizes.

Classification. In fig. 6.11, we compare our algorithms to the DLIS resampling algorithm of Katharopoulos and Fleuret (2018) on MNIST classification. Our IS performs slightly better than DLIS, and our OMIS does best. The differences between our methods and the rest are more pronounced when comparing equal-time performance. DLIS has a higher computational cost as it involves running a forward pass on a large mini-batch to compute resampling probabilities. Our OMIS requires access to the gradient of each mini-batch sample; obtaining these gradients in our current implementation is inefficient due to technical limitations in the optimization framework we use (PyTorch). Nevertheless, the method manages to make up for this overhead with a higher-quality gradient estimate. In fig. 6.11 we compare classification error.

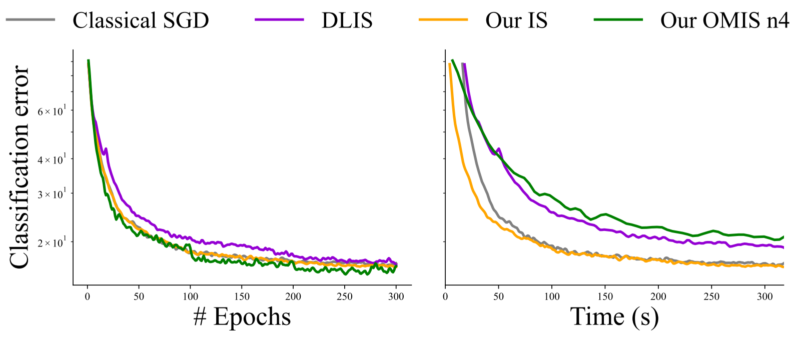


Figure 6.12: Comparison of our two methods (Our IS, Our OMIS) on point-cloud classification using PointNet Qi et al. (2017) architecture. Our OMIS achieves lower classification error at equal epochs, though it introduces computation overhead as shown at equal-time comparisons. At equal time, our method using importance sampling achieves the best performance.

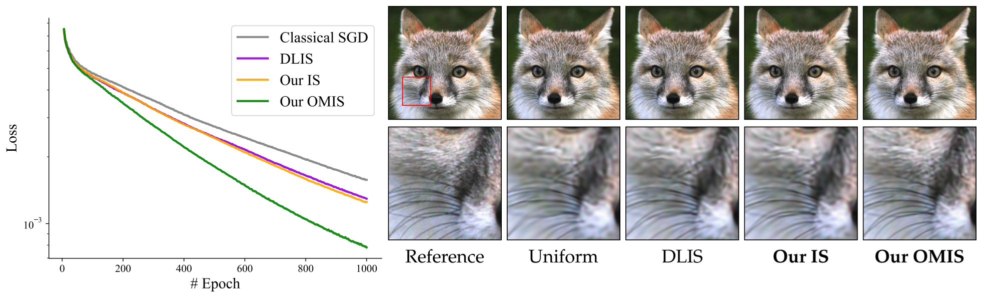


Figure 6.13: Comparison at equal step for image 2D regression. Left side show the convergence plot while the right display the result regression and a close-up view. Our method using MIS achieves the lower error on this problem while IS and DLIS perform similarly. On the images it is visible that our OMIS recover the finest details of the fur and whiskers.

Figure 6.12 shows point-cloud classification, where our IS is comparable to classical SGD and our OMIS outperforms other methods in terms of classification error at equal epochs. In complex cases where importance sampling cannot enhance convergence by providing a more accurate gradient estimator, our method is still as efficient as SGD due to minimal overhead. This means that even though importance sampling does not offer additional benefits in these scenarios, our implementation remains competitive with classical methods. In his case DLIS and our OMIS both suffer from computational overhead.

We also perform an ablation study for linear-system momentum in algorithm 5. We apply same momentum on the gradient for classical SGD, DLIS and our IS. Figure 6.14 shows this comparison. It demonstrate that classical SGD, DLIS and Our IS work similarly with and without momentum. Our OMIS still outperforms other methods for this task at equal steps.

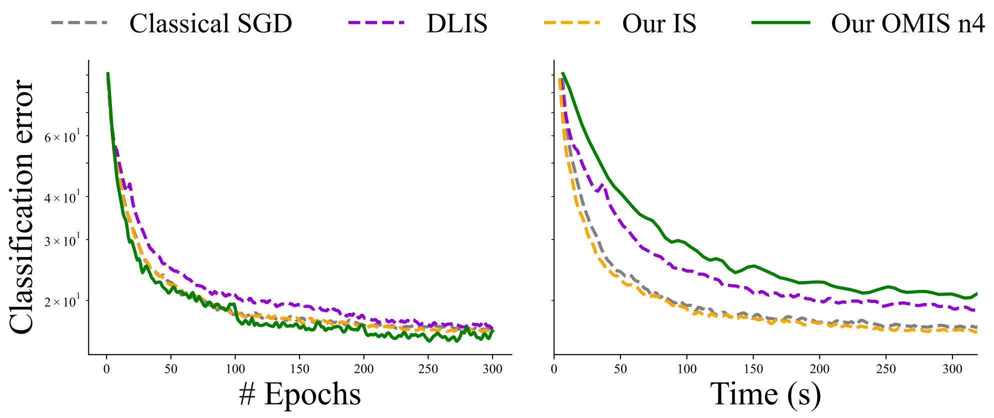


Figure 6.14: Ablation study on point-cloud classification using linear-system momentum as described in algorithm 5 for baselines represented as dashed lines. Our OMIS still outperforms other baselines at equal epochs, similar to the results shown in fig. 6.12.

Regression. Figure 6.13 shows results on image regression, comparing classical SGD, DLIS, and our IS and OMIS. Classical SGD yields a blurry image, as seen in the zoom-ins. DLIS and our IS methods achieves similar results, with increased whisker sharpness but still blurry fur, though ours has slightly lower loss and is computationally faster, as discussed above. Our OMIS employs three sampling distributions based on the network's outputs which represent the red, green and blue image channels. This method achieves the lowest error and highest image fidelity, as seen in the zoom-in.

6.2.5 Limitations And Future Work

We have showcased the effectiveness of importance sampling and optimal multiple importance sampling (OMIS) in machine-learning optimization, leading to a reduction in gradient-estimation error. Our current OMIS implementation incurs some overhead as it requires access to individual mini-batch sample gradients. Modern optimization frameworks can efficiently compute those gradients in parallel but only return their average. This is the main computational bottleneck in the method. The overhead of the linear system computation is negligible; we have tested using up to 10 distributions.

In all our experiments we allocate the same sampling budget to each distribution. Non-uniform sample distribution could potentially further reduce estimation variance, especially if it can be dynamically adjusted during the optimization process.

This work focuses on combining predefined sampling distributions, yet the optimal choice for multiple sampling distributions remains an open question. The quality and quantity of these chosen distributions significantly affect the potential enhancement of the method. Identifying the best set of sampling distributions for gradient estimation presents an intriguing and promising future research direction that could substantially enhance network optimization.

6.2.6 Conclusion

We presented a new approach to improve gradient-based optimization through efficient, adaptive importance sampling. Our method introduces a gradient-based sampling metric that evolves during training with minimal overhead, effectively prioritizing the data samples that contribute most to accurate gradient estimation.

We further extended this to vector-valued gradients using multiple importance sampling (MIS). Unlike standard importance sampling, which uses a single distribution across all parameters, our method assigns adaptive weights to multiple distributions independently for each parameter. This enables down-weighting poor sampling distributions while emphasizing the ones that best correlate with the true gradient direction. When high-quality distributions are available, this per-parameter differentiation can yield near-perfect gradient estimates at a fixed sample budget.

A key component of our approach is the use of optimal multiple importance sampling Kondapaneni et al. (2019), which gives an upper bound on the achievable error reduction from a given set of sampling strategies. While OMIS introduces extra computational cost to solve a linear system and scales poorly with the number of distributions, it provides valuable insight into the best possible weighting for minimizing variance in gradient estimates.

In machine learning, where the quality of gradient estimation directly impacts convergence speed and final model performance, this strategy offers a clear advantage. By adapting sampling to the structure of the gradient, our approach enables faster and more effective optimization, especially in compute-constrained settings.

Chapter 7
Conclusion

This thesis presents several methods to improve Monte Carlo integration across two major applications: physically based rendering and gradient estimation in machine learning. The central idea underpinning these contributions is to exploit prior knowledge about the structure and smoothness of the integrand to reduce variance. It demonstrates how such knowledge can be introduced through control variates, perceptual error distribution, and sampling.

We proposed a novel class of Monte Carlo estimators based on regression based control variates. These estimators offer provable improvements over classical Monte Carlo under minimal assumptions about the regression. While our experiments focused on polynomial regression for direct lighting, the framework is general and supports other function classes, allowing for broader applicability. Any regression that better approximates the function than a constant would improve integration.

In the context of sampling, we developed a new framework for multi-class sampling using Wasserstein barycenters. This approach introduces a filtered sliced optimal transport solver capable of handling millions of conflicting classes. Unlike prior work, which struggles to scale with class count, our method enables new possibilities, such as perceptually optimized sampling. We derived a perceptual error bound tailored to image-space Monte Carlo rendering and used it to guide sample distribution. Minimizing this bound leads to blue noise error characteristics in image space, significantly improving visual quality without increasing the sample count.

For gradient estimation in machine learning, we improved the use of importance sampling for mini-batch construction. By prioritizing data samples with higher learning potential, we demonstrated faster convergence and a lower final error. Our proposed sampling strategy is not only computationally efficient but also closely approximates the optimal importance distribution. To address the limitations of using a single distribution for vector-valued gradients, we introduced a multiple importance sampling estimator. This allows for adaptive weighting per gradient component, enabling improvements that would not be achievable with a single distribution.

Open Research Questions Further variance reduction may be achieved by incorporating even more task-specific knowledge into the estimators. This could involve problem-dependent structure, learned models of the integrand, or more advanced forms of functional approximation.

In rendering, although this thesis focuses on offline Monte Carlo integration, real-time rendering has gained significant traction in recent years, driven by both hardware acceleration and algorithmic improvements. Extending techniques such as control variates and perceptual sampling to real-time rendering scenarios presents an exciting and open area of research. Another promising direction involves leveraging learning-based tools to simplify various components of the rendering pipeline, including sampling, reconstruction, and denoising, to meet the stringent performance constraints of real-time applications. In particular, exploring the relationship between correlated sampling and neural denoising appears to be a critical area for future work. While correlated noise in screen space improves rendering, it can degrade the effectiveness of neural denoisers as it behaves differently from the uncorrelated noise typically used during training. Understanding and adapting to this correlation is essential, as neural denoisers can struggle when the noise distribution diverges from their training assumptions. Developing denoising strategies that are aware of or even optimized for correlated errors could lead to significant improvements in visual quality. Furthermore, identifying the optimal error distribution for training neural denoisers could unlock further advancements in both real-time and offline rendering contexts.

In this work, we developed a method to generate blue noise error distributions by optimizing sample positions independently of the rendering function. This scene-agnostic approach is practical as it enables the same sample set to be reused across a wide variety of scenes without the need for re-optimization. However, this generality is also its main limitation. Since the method produces a blue noise error sampling set that works across many scenes, the resulting distribution cannot be optimal for any specific one. In reality, scene-specific error distributions could better target local integration challenges and further improve image quality. This limitation is particularly relevant in real-time rendering applications, where maintaining low sample budgets is crucial. A scene-dependent optimization of the error distribution could significantly enhance visual quality without increasing rendering cost. Developing efficient, real-time methods to adapt sampling to scene content is therefore a promising and important direction for future work.

Error distribution has proven to be a critical factor in both enhancing the perceptual quality of rendered images and improving the effectiveness of denoising algorithms. This has been particularly evident in early analytic denoising approaches, such as Gaussian filtering, where the structure of the error directly impacted the quality of the final image. More recent work has shown that neural denoisers, particularly those using direct pixel prediction, struggle to handle structured or correlated noise patterns effectively. In contrast, approaches like neural kernel prediction demonstrate greater robustness but still face challenges when confronted with non-uniform or correlated error distributions. One promising direction to address these limitations is to train denoisers using correlated error distributions. This strategy could enable neural networks to better adapt to the noise characteristics of modern rendering techniques and produce higher-quality outputs. The issue is also relevant in image-space methods like ReSTIR (Salaün et al., 2025), which naturally introduce low-frequency error due to their reuse sampling strategy. If a denoiser is not specifically trained to handle such correlated noise, it may fail to remove artifacts effectively. Addressing this gap by aligning the training noise distribution with the actual rendering error structure could unlock significant improvements in denoising performance, but it requires further investigation.

Independent random sampling has solid theoretical backing, especially through the law of large numbers, but it is often not the best choice in practice. Low-discrepancy sequences and blue noise sampling have shown clear benefits in Monte Carlo integration. In most simulations where random numbers are used, blindly choosing independent sampling is likely suboptimal. There is almost always a better pattern more suited to the problem. While it is not always obvious what the ideal sampling correlation is, better alternatives exist. This challenge becomes even more important in multi-class or multi-channel tasks, where a single sampling strategy has to work across different objectives. In these cases, finding sampling patterns that balance the needs of all outputs is essential. Developing problem-specific, optimized sampling strategies, especially for new tasks, should be a key research focus.

In machine learning, the use of adaptive estimators at the per-parameter level has shown strong potential, as each parameter often corresponds to a distinct integral with unique characteristics. While our work primarily focused on using multiple importance sampling to differentiate integration across parameters, other variance reduction techniques such as per parameter control variates may offer similar benefits. A particularly promising direction for future research lies in designing estimators that are both simpler and more effective, while remaining easy to integrate into existing training frameworks. Such developments could lead to significant improvements in gradient estimation accuracy and overall training efficiency, especially in large-scale or high-dimensional models where gradients are extremely complex to estimate.

We conducted some preliminary exploration of data pruning in conjunction with our importance sampling method. The broader idea of restricting the training process to the most informative or impactful data samples is a promising direction. This concept aligns closely with curriculum learning, where training is structured by gradually introducing samples based on their difficulty or relevance. Integrating our importance sampling strategy with such curriculum-based approaches could lead to more efficient and effective training. Additionally, combining our method with multiple importance sampling techniques may further enhance performance by leveraging diverse sampling strategies in a unified framework. This synergy could provide improved convergence for learning tasks.

In this work, we focused on two primary applications: rendering and gradient estimation. However, Monte Carlo integration and sampling are fundamental techniques with broad applicability across many domains. Looking forward, an important research direction lies in identifying new tasks where task-specific information can be effectively integrated into the estimator or sampling process. By tailoring estimators to the structure or characteristics of a given problem, we can potentially achieve significant improvements in efficiency and accuracy. Such advances could extend the impact of our methods to diverse areas.

Bibliography - Own Work

Xingchang Huang, Corentin Salaun, Cristina Vasconcelos, Christian Theobalt, Cengiz Oztireli, and Gurprit Singh. 2024. Blue noise for diffusion models. In *ACM SIGGRAPH 2024 Conference Papers (SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 28, 11 pages. https://doi.org/10.1145/3641519.3657435

Miša Korać, Corentin Salaün, Iliyan Georgiev, Pascal Grittmann, Philipp Slusallek, Karol Myszkowski, and Gurprit Singh. 2023. Perceptual error optimization for Monte Carlo animation rendering. In *SIGGRAPH Asia 2023 Conference Papers (SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 89, 10 pages. https://doi.org/10.1145/3610548.3618146

Corentin Salaün, Martin Balint, Laurent Belcour, Eric Heitz, Gurprit Singh, and Karol Myszkowski. 2025. Histogram Stratification for Spatio-Temporal Reservoir Sampling. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 83, 10 pages. https://doi.org/10.1145/3721238.3730723

Corentin Salaün, Iliyan Georgiev, Hans-Peter Seidel, and Gurprit Singh. 2022a. Scalable Multi-Class Sampling via Filtered Sliced Optimal Transport. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH Asia)* 41, 6, Article 261 (2022), 14 pages. https://doi.org/10.1145/3550454.3555484

Corentin Salaün, Adrien Gruson, Binh-Son Hua, Toshiya Hachisuka, and Gurprit Singh. 2022b. Regression-based Monte Carlo integration. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)* 41, 4, Article 79 (2022), 14 pages. https://doi.org/10.1145/3528223.3530095

Corentin Salaün, Xingchang Huang, Iliyan Georgiev, Niloy Mitra, and Gurprit Singh. 2025a. Multiple Importance Sampling for Stochastic Gradient Estimation. In *Proceedings of the 14th International Conference on Pattern Recognition Applications and Methods - ICPRAM*. INSTICC, SciTePress, 401–409. https://doi.org/10.5220/0013311200003905

Corentin Salaün, Xingchang Huang, Iliyan Georgiev, Niloy Mitra, and Gurprit Singh. 2025b. Online Importance Sampling for Stochastic Gradient Optimization. In *Proceedings of the 14th International Conference on Pattern Recognition Applications and Methods - ICPRAM*. INSTICC, SciTePress, 130–140. https://doi.org/10.5220/0013311100003905

Martin Weier, Oliver Staadt, Bipul Mohanto, Colin Groth, Alexander Marquardt, Corentin Salaün, and Praneeth Chakravarthula. 2025. Second International Workshop on Perception-driven Graphics and Displays for VR and AR (PerGraVAR). In *2025 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 366–367. https://doi.org/10.1109/VRW66409.2025.00085

Bibliography

Martial Agueh and Guillaume Carlier. 2011. Barycenters in the Wasserstein Space. *SIAM Journal on Mathematical Analysis* 43, 2 (2011), 904–924. `https://doi.org/10.1137/100805741`

Abdalla G. M. Ahmed and Peter Wonka. 2020. Screen-space blue-noise diffusion of Monte Carlo sampling error via hierarchical ordering of pixels. *ACM Trans. Graph.* 39, 6 (2020), 244:1–244:15. `https://doi.org/10.1145/3414685.3417881`

Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. 2015. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481* (2015).

James Arvo et al. 1986. Backward ray tracing. In *Developments in Ray Tracing, Computer Graphics, Proc. of ACM SIGGRAPH 86 Course Notes*. 259–263.

Laurent Belcour and Eric Heitz. 2021. Lessons Learned and Improvements When Building Screen-Space Samplers with Blue-Noise Error Distribution. In *ACM SIGGRAPH 2021 Talks (SIGGRAPH '21)*. Association for Computing Machinery, New York, NY, USA, Article 9, 2 pages. `https://doi.org/10.1145/3450623.3464645`

Laurent Belcour, Guofu Xie, Christophe Hery, Mark Meyer, Wojciech Jarosz, and Derek Nowrouzezahrai. 2018. Integrating clipped spherical harmonics expansions. *ACM Transactions on Graphics (TOG)* 37, 2 (2018), 1–12.

Nicolas Bonneel and Hanspeter Pfister. 2013. *Sliced Wasserstein Barycenter of Multiple Densities*. Technical Report. Harvard Technical Report TR-02-13.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. 2015. Sliced and Radon Wasserstein Barycenters of Measures. *J. Math. Imaging Vis.* 51, 1 (2015). `https://doi.org/10.1007/s10851-014-0506-3`

Nicolas Bonnotte. 2013. *Unidimensional and evolution methods for optimal transportation*. Ph.D. Dissertation. Paris 11.

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. 2005. Fast Kernel Classifiers with Online and Active Learning. *Journal of Machine Learning Research* 6, 54 (2005), 1579–1619. `http://jmlr.org/papers/v6/bordes05a.html`

Léon Bottou. 1998. Online Algorithms and Stochastic Approximations. In *Online Learning and Neural Networks*, David Saad (Ed.). Cambridge University Press, Cambridge, UK.

Christina A. Burbeck and D. H. Kelly. 1980. Spatiotemporal characteristics of visual mechanisms: excitatory-inhibitory model. *J. Opt. Soc. Am.* 70, 9 (Sep 1980), 1121–1126. `https://doi.org/10.1364/JOSA.70.001121`

Russel E. Caflisch. 1998. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica* 7 (1998), 1–49. `https://doi.org/10.1017/S0962492900002804`

Jiating Chen, Xiaoyin Ge, Li-Yi Wei, Bin Wang, Yusu Wang, Huamin Wang, Yun Fei, Kang-Lai Qian, Jun-Hai Yong, and Wenping Wang. 2013. Bilateral Blue Noise Sampling. *ACM Trans. Graph.* 32, 6, Article 216 (nov 2013), 11 pages. https://doi.org/10.1145/2508363.2508375

Zhonggui Chen, Zhan Yuan, Yi-King Choi, Ligang Liu, and Wenping Wang. 2012. Variational Blue Noise Sampling. *IEEE Transactions on Visualization and Computer Graphics* 18, 10 (2012), 1784–1796. https://doi.org/10.1109/TVCG.2012.94

Vassillen Chizhov, Iliyan Georgiev, Karol Myszkowski, and Gurprit Singh. 2022. Perceptual Error Optimization for Monte Carlo Rendering. *ACM Trans. Graph.* 41, 3, Article 26 (mar 2022), 17 pages. https://doi.org/10.1145/3504002

Sebastian Claici, Edward Chien, and Justin Solomon. 2018. Stochastic wasserstein barycenters. In *International Conference on Machine Learning*. PMLR, 999–1008.

Petrik Clarberg and Tomas Akenine-Möller. 2008. Exploiting Visibility Correlation in Direct Illumination. 27, 4 (2008), 1125–1136. https://doi.org/10/d2gd6k

Robert L. Cook. 1986. Stochastic sampling in computer graphics. 5, 1 (1986), 51–72.

Miguel Crespo, Adrian Jarabo, and Adolfo Muñoz. 2021. Primary-Space Adaptive Control Variates Using Piecewise-Polynomial Approximations. *ACM Trans. Graph.* 40, 3, Article 25 (jul 2021), 15 pages. https://doi.org/10.1145/3450627

Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf

Fernando de Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue noise through optimal transport. *ACM Trans. Graph.* 31, 6, Article 171 (nov 2012), 11 pages. https://doi.org/10.1145/2366145.2366190

Li Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine* 29, 6 (2012), 141–142.

Chaosheng Dong, Xiaojie Jin, Weihao Gao, Yijia Wang, Hongyi Zhang, Xiang Wu, Jianchao Yang, and Xiaobing Liu. 2021. One Backward from Ten Forward, Subsampling for Large-Scale Deep Learning. *arXiv preprint arXiv:2104.13114* (2021).

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

Iliyan Georgiev and Marcos Fajardo. 2016. Blue-Noise Dithered Sampling. In *ACM SIGGRAPH 2016 Talks (SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 35, 1 pages. https://doi.org/10.1145/2897839.2927430

Paul Glasserman. 2004. *Monte Carlo methods in financial engineering*. Vol. 53. Springer.

Peter W. Glynn and Roberto Szechtman. 2002. Some New Perspectives on the Method of Control Variates. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, Kai-Tai Fang, Harald Niederreiter, and Fred J. Hickernell (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 27–49.

Alvaro J. González, Jan Bacca Rodríguez, Gonzalo R. Arce, and Daniel Leo Lau. 2006. Alpha stable human visual system models for digital halftoning. In *Electronic Imaging*.

Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. 2019. SGD: General Analysis and Improved Rates. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 5200–5209. `https://proceedings.mlr.press/v97/qian19b.html`

Pascal Grittmann, Iliyan Georgiev, and Philipp Slusallek. 2021. Correlation-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms. *Comput. Graph. Forum (EUROGRAPHICS 2021)* 40, 2 (2021).

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. 2019. Variance-Aware Multiple Importance Sampling. *ACM Trans. Graph. (SIGGRAPH Asia 2019)* 38, 6 (2019), 9. `https://doi.org/10.1145/3355089.3356515`

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Jerry Guo, Pablo Bauszat, Jacco Bikker, and Elmar Eisemann. 2018. Primary sample space path guiding. In *Eurographics Symposium on Rendering*, Vol. 2018. The Eurographics Association, 73–82.

John Hammersley. 2013. *Monte carlo methods*. Springer Science & Business Media.

S.T. Hammett and A.T. Smith. 1992. Two temporal channels or three? A re-evaluation. *Vision Research* 32, 2 (1992), 285–291. `https://doi.org/10.1016/0042-6989(92)90139-A`

Ayoub El Hanchi, David Stephens, and Chris Maddison. 2022. Stochastic Reweighted Gradient Descent. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.), Vol. 162. PMLR, 8359–8374. `https://proceedings.mlr.press/v162/hanchi22a.html`

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Stefan Heinrich. 2001. Multilevel Monte Carlo Methods. In *Proceedings of the Third International Conference on Large-Scale Scientific Computing-Revised Papers (LSSC '01)*. Springer-Verlag, Berlin, Heidelberg, 58–67.

Eric Heitz and Laurent Belcour. 2019. Distributing Monte Carlo Errors as a Blue Noise in Screen Space by Permuting Pixel Seeds Between Frames. *Computer Graphics Forum* (2019). `https://doi.org/10.1111/cgf.13778`

Eric Heitz, Laurent Belcour, Victor Ostromoukhov, David Coeurjolly, and Jean-Claude Iehl. 2019. A low-discrepancy sampler that distributes Monte Carlo errors as a blue noise in screen space. 1–2. `https://doi.org/10.1145/3306307.3328191`

Fred J. Hickernell, Christiane Lemieux, and Art B. Owen. 2005. Control Variates for Quasi-Monte Carlo. *Statist. Sci.* 20, 1 (2005), 1 – 31. `https://doi.org/10.1214/088342304000000468`

Richard Zou Horace He. 2021. functorch: JAX-like composable function transforms for PyTorch. https://github.com/pytorch/functorch.

Hu, Sha Ruizhen, van Kaick Tingkai, Deussen Oliver, Huang Oliver, and Hui. 2020. Data Sampling in Multi-view and Multi-class Scatterplots via Set Cover Optimization. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of InfoVis 2019)* 26, 1 (2020), 739–748.

Henrik Wann Jensen. 1996. Global illumination using photon maps. In *Rendering Techniques' 96: Proceedings of the Eurographics Workshop in Porto, Portugal, June 17–19, 1996 7*. Springer, 21–30.

Min Jiang, Yahan Zhou, Rui Wang, Richard Southern, and Jian Jun Zhang. 2015. Blue Noise Sampling Using an SPH-Based Method. *ACM Trans. Graph.* 34, 6, Article 211 (oct 2015), 11 pages. https://doi.org/10.1145/2816795.2818102

Herman Kahn. 1950. Random Sampling (Monte Carlo) Techniques in Neutron Attenuation Problems–I. *Nucleonics* 6, 5 (1950), 27, passim.

Herman Kahn and A. W. Marshall. 1953. Methods of Reducing Sample Size in Monte Carlo Computations. *Journal of the Operations Research Society of America* 1, 5 (1953), 263–278.

James T Kajiya. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 143–150.

Malvin H Kalos and Paula A Whitlock. 2009. *Monte carlo methods*. John Wiley & Sons.

Angelos Katharopoulos and François Fleuret. 2017. Biased Importance Sampling for Deep Neural Network Training. *ArXiv* abs/1706.00043 (2017). https://api.semanticscholar.org/CorpusID:38367260

Angelos Katharopoulos and Francois Fleuret. 2018. Not All Samples Are Created Equal: Deep Learning with Importance Sampling. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 2525–2534. https://proceedings.mlr.press/v80/katharopoulos18a.html

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. 21, 3 (Sept. 2002), 531–540. https://doi.org/10/bfrsqn

Alexander Keller. 2001. Hierarchical Monte Carlo Image Synthesis. *Math. Comput. Simul.* 55, 1–3 (Feb. 2001), 79–92. https://doi.org/10.1016/S0378-4754(00)00248-2

Alexander Keller. 2004. Stratification by rank-1 lattices. In *Monte Carlo and Quasi-Monte Carlo Methods 2002: Proceedings of a Conference held at the National University of Singapore, Republic of Singapore, November 25–28, 2002*. Springer, 299–313.

Andrew Kensler. 1967. Correlated multi-jittered sampling. *Mathematical Physics and applied mathematics* 7 (1967), 86–112.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Thomas Kollig and Alexander Keller. 2002. Efficient multidimensional sampling. In *Computer Graphics Forum*, Vol. 21. Wiley Online Library, 557–563.

Ivo Kondapaneni, Petr Vévoda, Pascal Grittmann, Tomas Skrivan, Philipp Slusallek, and Jaroslav Krivanek. 2019. Optimal Multiple Importance Sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2019)* 38, 4 (July 2019), 37:1–37:14. https://doi.org/10.1145/3306346.3323009

Miša Korać, Corentin Salaün, Iliyan Georgiev, Pascal Grittmann, Philipp Slusallek, Karol Myszkowski, and Gurprit Singh. 2023. Perceptual error optimization for Monte Carlo animation rendering. In *SIGGRAPH Asia 2023 Conference Papers (SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 89, 10 pages. https://doi.org/10.1145/3610548.3618146

Alexander Korotin, Daniil Selikhanovych, and Evgeny Burnaev. 2022. Neural optimal transport. *arXiv preprint arXiv:2201.12220* (2022).

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. 2017. Spectral and decomposition tracking for rendering heterogeneous volumes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–16.

Eric Lafortune. 1996. Mathematical models and Monte Carlo algorithms for physically based rendering. *Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven* 20, 74-79 (1996), 4.

Eric P. Lafortune and Yves D. Willems. 1994. The Ambient Term as a Variance Reducing Technique for Monte Carlo Ray Tracing. 163–171.

W. W. Loh. 1995. *On the Method of Control Variates*. Ph.D. Thesis. Stanford University.

Ilya Loshchilov and Frank Hutter. 2015. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343* (2015).

Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).

Rafał K Mantiuk, Gyorgy Denes, Alexandre Chapiro, Anton Kaplanyan, Gizem Rufo, Romain Bachy, Trisha Lian, and Anjul Patney. 2021. FovVideoVDP: A visible difference predictor for wide field-of-view video. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–19.

Michael McCool and Eugene Fiume. 1992. Hierarchical Poisson disk sampling distributions. In *Graphics interface*, Vol. 92. 94–105.

Don P. Mitchell. 1991. Spectrally Optimal Sampling for Distribution Ray Tracing. *SIGGRAPH Computer Graphics* 25, 4 (July 1991), 157–164.

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. 2014. Adaptive Rendering Based on Weighted Local Regression. 33, 5 (Sept. 2014), 170:1–170:14. https://doi.org/10/f6km7m

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. 36, 4 (June 2017), 91–100. https://doi.org/10/gbnvrs

Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. 2020. Neural Control Variates. *ACM Trans. Graph.* 39, 6, Article 243 (Nov. 2020), 19 pages. https://doi.org/10.1145/3414685.3417804

Yuji Nakatsukasa. 2018. Approximate and integrate: Variance reduction in Monte Carlo integration via function approximation. arXiv:math.NA/1806.05492

R. Näsänen. 1984. Visibility of halftone dot textures. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-14, 6 (1984), 920–924.

Deanna Needell, Rachel Ward, and Nati Srebro. 2014. Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz algorithm. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2014/file/f29c21d4897f78948b91f03172341b7b-Paper.pdf

Maria-Elena Nilsback and Andrew Zisserman. 2008. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*. IEEE, 722–729.

Yann Ollivier, Herve Pajot, and Cédric Villani (Eds.). 2014. *Optimal Transport - Theory and Applications*. London Mathematical Society lecture note series, Vol. 413. Cambridge University Press.

Christian van Onzenoodt, Gurprit Singh, Timo Ropinski, and Tobias Ritschel. 2021. Blue Noise Plots. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.142644

Art Owen and Yi Zhou. 2000a. Safe and effective importance sampling. *J. Amer. Statist. Assoc.* 95, 449 (2000), 135–143.

Art Owen and Yi Zhou. 2000b. Safe and Effective Importance Sampling. *J. Amer. Statist. Assoc.* 95, 449 (2000), 135–143. https://doi.org/10.1080/01621459.2000.10473909 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.2000.10473909

Art B Owen. 1995. Randomly permuted (t, m, s)-nets and (t, s)-sequences. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing: Proceedings of a conference at the University of Nevada, Las Vegas, Nevada, USA, June 23–25, 1994*. Springer, 299–317.

Thrasyvoulos N. Pappas and David L. Neuhoff. 1999. Least-squares model-based halftoning. *IEEE Transactions on Image Processing* 8, 8 (Aug 1999), 1102–1116. https://doi.org/10.1109/83.777090

Lois Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Antoine Webanck, Mathieu Desbrun, and Victor Ostromoukhov. 2020. Sliced optimal transport sampling. *ACM Trans. Graph.* 39, 4 (2020), 99.

Gabriel Peyré and Marco Cuturi. 2018. Computational Optimal Transport. (2018). https://doi.org/10.48550/ARXIV.1803.00567

Matt Pharr. 2019. Efficient Generation of Points that Satisfy Two-Dimensional Elementary Intervals. *Journal of Computer Graphics Techniques Vol* 8, 1 (2019).

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016a. *Physically Based Rendering: From Theory to Implementation* (3 ed.). San Francisco, CA, USA.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016b. *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

François Pitié, Anil C. Kokaram, and Rozenn Dahyot. 2005. N-Dimensional Probablility Density Function Transfer and its Application to Colour Transfer. In *10th IEEE International Conference on Computer Vision*. IEEE Computer Society. https://doi.org/10.1109/ICCV.2005.166

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

Hongxing Qin, Yi Chen, Jinlong He, and Baoquan Chen. 2017. Wasserstein Blue Noise Sampling. *ACM Trans. Graph.* 36, 5, Article 168 (Oct. 2017), 13 pages. https://doi.org/10.1145/3119910

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. 2011. Wasserstein Barycenter and Its Application to Texture Mixing. In *Scale Space and Variational Methods in Computer Vision - Third International Conference, SSVM 2011, Ein-Gedi, Israel, May 29 - June 2, 2011, Revised Selected Papers (Lecture Notes in Computer Science)*, Alfred M. Bruckstein, Bart M. ter Haar Romeny, Alexander M. Bronstein, and Michael M. Bronstein (Eds.), Vol. 6667. https://doi.org/10.1007/978-3-642-24785-9_37

Svetlozar Rachev and Ludger Rüschendorf. 1998. *Mass Transportation Problems: Volume I: Theory*. Springer.

L. Roberts. 1962. Picture coding using pseudo-random noise. *IRE Transactions on Information Theory* 8, 2 (1962), 145–154. https://doi.org/10.1109/TIT.1962.1057702

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* 323, 6088 (1986), 533–536.

Corentin Salaün, Martin Balint, Laurent Belcour, Eric Heitz, Gurprit Singh, and Karol Myszkowski. 2025. Histogram Stratification for Spatio-Temporal Reservoir Sampling. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 83, 10 pages. https://doi.org/10.1145/3721238.3730723

Corentin Salaün, Iliyan Georgiev, Hans-Peter Seidel, and Gurprit Singh. 2022a. Scalable Multi-Class Sampling via Filtered Sliced Optimal Transport. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH Asia)* 41, 6, Article 261 (2022), 14 pages. https://doi.org/10.1145/3550454.3555484

Corentin Salaün, Adrien Gruson, Binh-Son Hua, Toshiya Hachisuka, and Gurprit Singh. 2022b. Regression-based Monte Carlo integration. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)* 41, 4, Article 79 (2022), 14 pages. https://doi.org/10.1145/3528223.3530095

Corentin Salaün, Xingchang Huang, Iliyan Georgiev, Niloy Mitra, and Gurprit Singh. 2025a. Multiple Importance Sampling for Stochastic Gradient Estimation. In *Proceedings of the 14th International Conference on Pattern Recognition Applications and Methods - ICPRAM*. INSTICC, SciTePress, 401–409. `https://doi.org/10.5220/0013311200003905`

Corentin Salaün, Xingchang Huang, Iliyan Georgiev, Niloy Mitra, and Gurprit Singh. 2025b. Online Importance Sampling for Stochastic Gradient Optimization. In *Proceedings of the 14th International Conference on Pattern Recognition Applications and Methods - ICPRAM*. INSTICC, SciTePress, 130–140. `https://doi.org/10.5220/0013311100003905`

F. Santambrogio. 2015. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Springer International Publishing.

Carlos Santiago, Catarina Barata, Michele Sasdelli, Gustavo Carneiro, and Jacinto C Nascimento. 2021. LOW: Training deep neural networks by learning optimal sample weights. *Pattern Recognition* 110 (2021), 107585.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

C. Schmaltz, P. Gwosdek, and J. Weickert. 2012. Multi-Class Anisotropic Electrostatic Halftoning. *Comput. Graph. Forum* 31, 6 (sep 2012), 1924–1935. `https://doi.org/10.1111/j.1467-8659.2012.03072.x`

Christoph Schulz, Kin Chung Kwan, Michael Becher, Daniel Baumgartner, Guido Reina, Oliver Deussen, and Daniel Weiskopf. 2021. Multi-Class Inverted Stippling. *ACM Trans. Graph.* 40, 6, Article 245 (dec 2021), 12 pages. `https://doi.org/10.1145/3478513.3480534`

Adrian Secord. 2002. Weighted Voronoi stippling. In *Proc. NPAR*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems* 33 (2020), 7462–7473.

Kartic Subr. 2021. Q-NET: A Network for Low-dimensional Integrals of Neural Proxies. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 61–71.

Lieven Vandenberghe. 2010. The CVXOPT linear and quadratic cone program solvers. *Online: http://cvxopt. org/documentation/coneprog. pdf* (2010).

Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Thesis. Stanford University, United States – California.

Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering, Vol. 29. 419–428. `https://doi.org/10/d7b6n4`

Eric Veach and Leonidas J Guibas. 1997. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 65–76.

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian online regression for adaptive direct illumination sampling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

C. Villani. 2008. *Optimal Transport: Old and New*. Springer Berlin Heidelberg. https://books.google.fr/books?id=hV8o5R7_5tkC

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-Line Learning of Parametric Mixture Models for Light Transport Simulation. 33, 4 (Aug. 2014), 101:1–101:11. https://doi.org/10/f6c2cp

Linnan Wang, Yi Yang, Renqiang Min, and Srimat Chakradhar. 2017. Accelerating deep neural network training with inconsistent stochastic gradient descent. *Neural Networks* 93 (2017), 219–229.

Muge Wang and Kevin J. Parker. 1999. Properties of combined blue noise patterns. *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)* 4 (1999), 328–332 vol.4.

Li-Yi Wei. 2010. Multi-Class Blue Noise Sampling. *ACM Trans. Graph.* 29, 4, Article 79 (July 2010), 8 pages. https://doi.org/10.1145/1778765.1778816

Martin Weier, Michael Stengel, Thorsten Roth, Piotr Didyk, Elmar Eisemann, Martin Eisemann, Steve Grogorick, André Hinkenjann, Ernst Kruijff, Marcus Magnor, et al. 2017. Perception-driven accelerated rendering. In *Computer graphics forum*, Vol. 36. Wiley Online Library, 611–643.

Alan Wolfe, Nathan Morrical, Tomas Akenine-Möller, and Ravi Ramamoorthi. 2022. Spatiotemporal Blue Noise Masks.. In *EGSR (ST)*. 117–126.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

Lei Yang, Shiqiu Liu, and Marco Salvi. 2020. A Survey of Temporal Antialiasing Techniques. *Computer Graphics Forum* 39, 2 (2020), 607–621. https://doi.org/10.1111/cgf.14018 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14018

Cheng Zhang, Hedvig Kjellstrom, and Stephan Mandt. 2017. Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607* (2017).

Cheng Zhang, Cengiz Öztireli, Stephan Mandt, and Giampiero Salvi. 2019. Active mini-batch sampling using repulsive point processes. In *Proceedings of the AAAI conference on Artificial Intelligence*, Vol. 33. 5741–5748.

Minghe Zhang, Chaosheng Dong, Jinmiao Fu, Tianchen Zhou, Jia Liang, Jia Liu, Bo Liu, Michinari Momma, Bryan Wang, Yan Gao, et al. 2023. AdaSelection: Accelerating Deep Learning Training through Data Subsampling. *arXiv preprint arXiv:2306.10728* (2023).

Peilin Zhao and Tong Zhang. 2015. Stochastic Optimization with Importance Sampling for Regularized Loss Minimization. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 1–9. https://proceedings.mlr.press/v37/zhaoa15.html

Quan Zheng and Matthias Zwicker. 2019. Learning to importance sample in primary sample space. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 169–179.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. 34, 2 (May 2015), 667–681. `https://doi.org/10/f7k6kj`