Rethinking multiple importance sampling for general and efficient Monte Carlo rendering

Dissertation zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

vorgelegt von
Pascal Grittmann

Saarbrücken, 2023

Tag des Kolloquiums:

Dienstag, 06. Februar 2024

Dekan:

Prof. Dr. Jürgen Steimle

Vorsitzende:

Prof. Dr. Verena Wolf

Berichterstatter:

Prof. Dr. Philipp Slusallek

Prof. Dr. Hans-Peter Seidel

Prof. Dr. Wenzel Jakob

Akademischer Mitarbeiter:

Dr. Gurprit Singh

Abstract

Computer generated images are essential for many applications from art to engineering. Unfortunately, rendering such images is costly, with render times easily in the hours, days, or even weeks. On top of that, the demands regarding complexity and visual fidelity are ever rising. Consequently, there is an insatiable need for faster rendering.

Efficient render times are often achieved through user intervention. For example, modifying the scene and removing difficult lighting effects can keep render times below an acceptable threshold. Also, algorithm parameters can be tuned manually. For instance, diffuse outdoor scenes are best rendered by unidirectional path tracing, while interiors featuring caustics benefit greatly from bidirectional sampling. Such manual tuning, however, is unfortunate as it puts much burden on the user and poses a hurdle for novices.

In this thesis, we pave the way for more universal rendering algorithms with less need of user intervention. For that, we revisit multiple importance sampling (MIS), an essential tool to universalize rendering algorithms by combining diverse sampling techniques. We identify hitherto unknown shortcomings of MIS and propose practical solutions and improvements. As a tangible result, we achieve adaptive bidirectional rendering with performance never worse than unidirectional path tracing.

Kurzfassung

Computergenerierte Bilder sind essentiell für zahlreiche Anwendungen. Leider ist das Rendering solcher Bilder extrem teuer, mit Renderzeiten in den Studen, Tagen, oder gar Wochen. Darüberhinaus steigen die Ansprüche an Komplexität und Qualität stetig und schier unaufhaltsam. Daher gibt es einen scheinbar niemals endenden Bedarf nach immer schnelleren Rendering Algorithmen.

Schnelle Renderzeiten werden oft durch Nutzerintervention erreicht. Zum Beispiel können Szenen durch das Entfernen von teuren Effekten manuell vereinfacht werden. Auch die geschickte manuelle Wahl des besten Algorithmus und der zugehörigen Parameter ist hilfreich. Zum Beispiel werden diffuse Außenumgebungen am effizientesten mit unidirektionalem Path Tracing simuliert, während Innenumgebungen mit komplexen Kaustiken sehr von bidirektionalen Algorithmen profitieren. Solch manuelle Einstellungen sind allerdings unvorteilhaft, da sie viel Last auf den Nutzer packen und eine Hürde für Anfänger darstellen.

Diese Arbeit zielt darauf ab, den Weg freizuräumen für universellere Rendering Algorithmen mit geringerem Bedarf nach manueller Kontrolle. Eine essentielle Komponente dafür ist die Kombination mehrerer Methoden durch Multiple Importance Sampling (MIS). Wir identifizieren bislang unbekannte Mängel von MIS und beheben diese mit praktischen Lösungen. Als greifbares Ergebnis erzielen wir einen adaptiven bidirektionalen Algorithmus der stets schneller ist als reines unidirektionales Path Tracing.

Acknowledgements

This thesis reports (a select subset of) the results of multiple years of hard work, fascinating experiences – and great fun. But both, the quality of this work and the splendid time I had conducting it, would not have been possible without the support of many great people.

For starters, I would like to thank my advisor, Philipp Slusallek. He first encouraged me to embark on the insightful journey to obtain a PhD. Without his exceptional support, the contacts he introduced me to, the opportunities he found or created for me, and the great freedom he gave me to pursue my own ideas, none of this would have ever happened.

Arguably the most important person that Philipp introduced me to was Jaroslav Křivánek. Jaroslav, who sadly passed away in 2019, had the strongest influence on the direction and quality of my scientific career. He taught me most, maybe even all, the skills I needed to successfully embark on a PhD and beyond, and imparted much great – and, as he jokingly used to call it "fatherly" – advice. I will forever fondly remember the many great brainstorming sessions and other (not always work-related) amazing moments we shared.

The best part of doing a PhD was the highly collaborative nature of the work. For that, I want to thank the many wonderful people I got to work with, but also the many other practitioners in the field with whom it was always fun to exchange thoughts and ideas at conferences and other meetings. Naming all these amazing people would be all but impossible, but I would like to highlight two particularly great ones: Iliyan Georgiev, who I could always count on to refine and perfect an idea (or fix the most annoying LaTeX bugs); and Sebastian Herholz, who did an exceptional job to identify and help resolve any and (almost) all flaws in my ideas, writing, or presentations.

Another, often overlooked, invaluable group of people are the heroes without capes that fight the bureaucratic fight for us. So I would like to take this opportunity to also thank Sabine Nermerich (Saarland University), Léa Yvonne Basters (DFKI), and Radka Hacklova (Charles University) who were always eager to help with any and all problems.

Finally, I must thank my amazing wife, Mira Niemann. Not only did she provide unwavering emotional support and endured me during (too) many stressful deadline crunches, she even directly helped with my research by developing a tool that I have been using extensively ever since: the figure generator (https://github.com/Mira-13/figure-gen).

Contents

Chapter 1

Introduction

Can we find the one algorithm to render them all?

A vast range of applications benefit greatly from computer generated images. From architecture to video games, from visual effects in movies to medical applications, rendered images are almost everywhere nowadays. For example, the design process of buildings and products benefits from replacing slow and expensive physical prototypes by virtual ones. Another example are machine learning methods, where availability of training data is essential to achieve satisfactory results. There, synthetic training data is used to reduce the cost or the need to obtain data in the real world. And finally, of course, rendered images are the heart and soul of movies and video games. Worlds, things, and creatures that do not exist in reality can be brought to life through rendering.

The demands of these applications vary considerably. For artistic applications such as movies and games, visual appeal is paramount and accuracy is secondary. Real-time settings like games and interactive visualizations put render time above everything else, sacrificing both accuracy and quality if need be. For uses like product design, speed and visual appeal can be secondary, while accuracy is essential. In a perfect world, we would have a single renderer that works well for all these applications, exposing only high-level controls for the user to express the application-specific priorities. Sadly, this appears to be a forlorn dream, as the renderers today do not even work perfectly for all types of illumination encountered in the application they are specialized for.

As the applications are diverse, so is the distribution and scattering of light itself. Sometimes, illumination is diffuse; for example, when rendering distant hills on a cloudy autumn day. Other times, light is focused strongly, like the caustics at the bottom of a pool. Ideally, a renderer should capture all those effects accurately and efficiently. In this thesis, we show how to move further towards that ideal.

Our foundation are Monte Carlo rendering algorithms. These compute the light transport in a scene by sampling random paths connecting the sensor to a light source. Monte Carlo methods are pretty much the sole approach taken by offline renderers these days [Fascione et al. 2018] and see increasing use in real-time applications, as hardware capability increases.

The most basic Monte Carlo algorithm is forward path tracing. It generates paths by sampling a direction from the camera, tracing a ray along that direction, and then sampling a new direction at the hit point to continue the path. This process is repeated until a termination criterion is met (e.g., maximum depth, random termination, or full absorption). In theory, this algorithm can render every scene[1]. But, in practice, satisfactory performance is often only achieved in a subset of all possible scenes – namely, ones with reasonably uniform illumination and predominantly short paths.

More elaborate techniques are required to achieve satisfactory performance in general scenes. Such methods range from the simplest tricks, like next event estimation, to elaborate adaptive sampling schemes. Years of active research have resulted in an enormous set of techniques and variations of techniques that can be employed to optimize rendering performance for all sorts of lighting phenomena.

This vast amount of efficiency-increasing methods poses a difficult challenge. For every scene, and every lighting effect within a scene, only a handful of these techniques perform well. But this set of the best techniques differs from scene to scene – or even from effect to effect within the same scene. Efficient rendering requires us to identify and use the best – and only the best – of these techniques.

One solution to that problem is to ask the user to identify and configure the best techniques. That, however, requires deep understanding on the part of the user, and also a large amount of trial and error. Expert users can achieve astounding rendering performance by navigating the jungle of renderer parameters – or by adjusting their scenes to eliminate problematic lighting effects. But the required skills take years of hard work to acquire, and the tedious optimization process also limits artistic freedom.

An alternative is to always utilize as many techniques as possible. This is the philosophy behind bidirectional rendering algorithms like VCM [Georgiev et al. 2012a]. A combination of numerous techniques ensures that for every possible scene and every possible effect, there is at least one technique that can render it in acceptable time. Such a combination can be achieved through multiple importance sampling (MIS) [Veach and Guibas 1995b], an essential ingredient to basically every rendering algorithm. Unfortunately, while the result is a general algorithm that can render anything in *acceptable* time, the performance is almost never great, because computation time is wasted on redundant or unnecessary techniques, and because the combination through MIS does not always perform perfectly.

The overarching goal of this thesis is to solve this problem through automation. Our hypothesis is that an ideal rendering algorithm should adapt *itself* to the scene at hand, resulting in a general and efficient method that requires little or, preferably, no user control.

The research discussed in this thesis focuses on the essential ingredient: MIS. We reveal and ameliorate flaws in the common MIS weighting heuristics that can result in severe performance reduction. Further, we show how the set of techniques can be adapted automatically in a practical approach that is applicable even to involved bidirectional approaches.

The motivation of our work originates in our earlier research on efficient caustic rendering [Grittmann et al. 2018]. There, the goal was to enhance forward path tracing by the minimum amount of bidirectional samples required to efficiently capture caustics. That work showed that automatic adaptation of the techniques is possible and worthwhile, and

[1]Ignoring some non-physical corner cases like point lights, or lighting effects that cannot be described by geometric optics.

also practical [Šik and Křivánek 2019]. Despite these very encouraging results, we also identified problems and challenges. Namely, we observed that MIS weighting can perform surprisingly poorly in some cases, and that adaptation of the number of light paths has a huge impact on performance, but is a nontrivial task. Therefore, in our subsequent research, that is, the work reported in this thesis, we set out to address these challenges.

1.1 Contributions

This thesis proposes different approaches to improve the efficiency and generality of Monte Carlo rendering algorithms through better multiple importance sampling (MIS). Our work focuses on bidirectional rendering algorithms, which are among the most complex applications of MIS. But the ideas, and most of the methods, are generic and can be applied to arbitrary Monte Carlo integration problems. The main contributions in this thesis are two methods to improve the MIS weights and a method to automatically adapt the set of sampling techniques and corresponding sample counts.

Variance-aware MIS. The classic balance heuristic [Veach and Guibas 1995b] can perform surprisingly poorly for bidirectional algorithms. Our first approach to fixing this problem was to incorporate variance estimates into the weights. We discuss it in Chapter 5. This work has been previously published in our SIGGRAPH Asia 2019 paper [Grittmann et al. 2019]. I was the main author of that paper, contributing the original idea, the implementation and evaluation, and the majority of the text in the paper.

Correlation-aware MIS. A major failure case of MIS in bidirectional rendering is path correlation due to shared prefixes. This occurs in particular with the popular VCM algorithm, where millions of samples are formed by merging a single, possibly high-variance, camera subpath with millions of photons. We can alleviate the problem with a simple-to-compute heuristic. The approach is discussed in Chapter 6. It has been published in our Eurographics 2021 paper [Grittmann et al. 2021]. I was the main author of that paper, contributing the original idea, the implementation and evaluation, and the majority of the text in the paper.

Efficiency-aware MIS. A common criticism of bidirectional rendering algorithms, and most other advanced methods, is that they add a huge overhead when they are not needed. In Chapter 7, we discuss how to tackle that problem by optimizing the sample counts in MIS combinations, using a practical method that can be applied to the high-dimensional sampling techniques that are omnipresent in bidirectional methods. The work has been published in our SIGGRAPH 2022 paper [Grittmann et al. 2022]. I was the main author of that paper, contributing the original idea, the initial implementation, some evaluations, and the majority of the text in the paper.

Source code. Implementations of all three methods are available on GitHub:

- Variance-aware MIS: https://github.com/pgrit/var-aware-mis-pbrt
- Correlation-aware MIS: https://github.com/pgrit/MisForCorrelatedBidir
- Efficiency-aware MIS: https://github.com/pgrit/EfficiencyAwareMIS

The public source code for variance-aware and efficiency-aware MIS are cleaned reimplementations that focus on conveying the key concepts of implementing the respective method. The public code for correlation-aware MIS is the exact code used by the original paper and allows full reconstruction of all results, as shown in the paper and reprinted in this thesis.

Since recycling is great for the environment, parts of this thesis (specifically, most of Chapters 5 to 7) reuse figures and passages of text from these three original publications.

1.2 Outline

Chapter 2 reviews the mathematical formulation of rendering, and provides an introduction to general Monte Carlo integration and its application to light transport simulation.

Chapter 3 introduces multiple importance sampling (MIS), discusses its properties, strengths, and weaknesses, reviews previous work on the topic, and outlines open problems.

Chapter 4 defines the sampling techniques used by our prime application, the VCM algorithm [Georgiev et al. 2012a], and provides a discussion of the challenges this algorithm faces in terms of an effective application of MIS.

The subsequent three chapters present and discuss our three methods. Chapter 5 presents a practical approach to improve MIS weights in failure cases, by injecting estimates of the variance into the weights. Chapter 6 introduces an ad-hoc solution for a major shortcoming of MIS weights in the presence of correlated samples. Finally, Chapter 7 introduces a method to automatically adjust the set of techniques and corresponding sample counts, to optimize the efficiency of complex MIS combinations.

Lastly, Chapter 8 summarizes our findings and outlines interesting directions for future work.

Chapter 2

Monte Carlo Light Transport

Images of virtual scenes can be rendered by simulating the propagation of light through that scene. This light transport is described by an infinite-dimensional integral with many discontinuities that can only be computed via numerical methods; the numerical method of choice is Monte Carlo integration.

This chapter introduces the light transport equation, defines Monte Carlo integration and discusses its properties, provides an overview of essential variance reduction methods, and concludes by reviewing core rendering algorithms that apply Monte Carlo methods. A basic knowledge of calculus and probability theory is assumed.

2.1 Problem formulation

Our goal is to render an image of a virtual scene. This process can be made almost arbitrarily complex by including, for instance, volumetric scattering, wave optical effects, or elaborate material models. For the sake of conciseness, we limit the discussions in this thesis to a basic model of surface light transport without wavelength dependencies. Information on more elaborate models and how to implement them can be found in Pharr et al. [2016].

The value of a pixel is computed by combining a camera model with the rendering equation [Kajiya 1986]. The former describes how the image sensor responds to light arriving at the aperture of the lens and the latter models the scattering of light in the scene. This setup is illustrated in Figure 2.1.

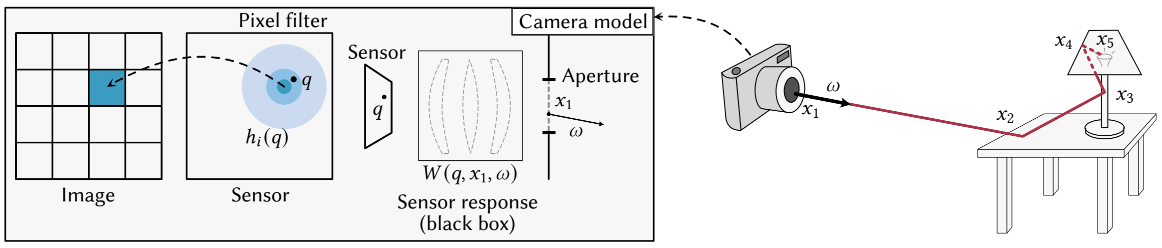


Figure 2.1: *The basic mathematical model of rendering we operate with. The value of the ith pixel is computed through a series of integrals over the sensor, aperture, and scene geometry.*

2.1.1 Camera model

The camera model describes how light that reaches the aperture is captured by the camera's sensor, and how that sensor's response is mapped to the rendered image. The former is modeled by the sensor response W, and the latter by the pixel filter h_i.

The sensor response $W(q, x_1, \omega)$ describes what fraction of incident light at the aperture point x_1 from direction ω reaches a point q on the sensor. Similarly, the pixel filter $h_i(q)$ models how much the sensor position q contributes to the ith pixel. For brevity, we denote the product of the two as

$$W_i(q, x_1, \omega) = h_i(q) W(q, x_1, \omega). \tag{2.1}$$

This W_i is often referred to as the *importance* [Veach 1997], as it measures how 'important' light arriving at x_1 from direction ω is to the image.

With these definitions, the pixel value can be computed by integrating the product of pixel filter, sensor response, and incident light over the sensor and aperture,

$$I_i = \int_{\mathcal{S}} \int_{\mathcal{L}} W_i(q, x_1, \omega) L_\mathrm{i}(x_1, \omega) \, \mathrm{d}x_1 \mathrm{d}q. \tag{2.2}$$

Here, $\mathcal{S}$ denotes the set of all points q on the sensor; $\mathcal{L}$ similarly denotes the (virtual) aperture surface. We assume that the direction ω is uniquely defined by the combination of sensor and aperture point. Otherwise, an additional integral over all directions arises.

2.1.2 Rendering equation

The incident light L_i is measured in terms of *radiance*, that is, incident differential power $\mathrm{d}\Phi$ per differential solid angle $\mathrm{d}\omega$ and differential projected surface area $\mathrm{d}x^\perp$,

$$L_\mathrm{i}(x, \omega) = \frac{\mathrm{d}\Phi}{\mathrm{d}\omega \mathrm{d}x^\perp}. \tag{2.3}$$

Note that here and throughout, we assume no volumetric scattering occurs and further simplify by ignoring wavelength dependencies. In an ideal optical system, the incident radiance $L_\mathrm{i}(x_1, \omega)$ at the aperture equals the outgoing radiance at the visible point x_2 (see Figure 2.1),

$$L_\mathrm{i}(x_1, \omega) = L_\mathrm{o}(x_2, -\omega). \tag{2.4}$$

The outgoing radiance at any point x, in turn, is defined recursively as the sum of the emitted radiance L_e and the reflected radiance [Kajiya 1986],

$$L_\mathrm{o}(x, \omega_\mathrm{o}) = L_\mathrm{e}(x, \omega_\mathrm{o}) + \int_\Omega \rho(\omega_\mathrm{i}, x, \omega_\mathrm{o}) L_\mathrm{i}(\omega_\mathrm{i}, x) \cos\theta_x \mathrm{d}\omega_\mathrm{i}. \tag{2.5}$$

Figure 2.2 visualizes the conventions used to define the directions and angles in this equation. The reflected radiance is computed by integrating over all directions and computing the product of the incident radiance and the bidirectional scattering distribution function (BSDF) ρ. The latter describes the optical properties of the surface, namely how much light from ω_i is scattered in direction ω_o at point x. The incident radiance $L_\mathrm{i}(\omega_\mathrm{i}, x)$ is obtained by recursively applying the same equation. Hence, light transport is described by an *infinite-dimensional* recursive integral.

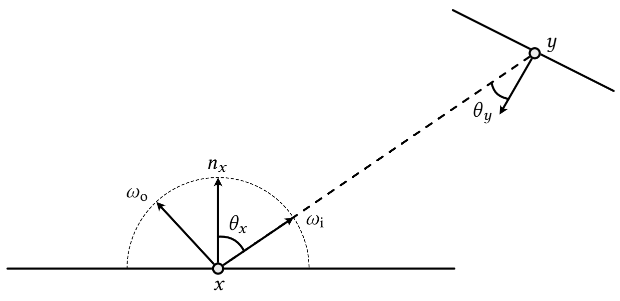


Figure 2.2: *The notation used to define the rendering equation.*

2.1.3 Surface integral

The fundamental challenge of rendering is to efficiently compute the reflected radiance integrals. To aid computation, a change of variables can be performed, rewriting the integral over directions as one over a different domain. An important equivalent formulation is the surface integral. It is the mathematical foundation for essential techniques like next event estimation or bidirectional sampling.

Instead of integrating over all directions at the point x, we can integrate over all points y on other surfaces that are visible from x,

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\mathcal{V}(x)} \rho(\omega_i, x, \omega_o) L_i(\omega_i, x) \cos\theta_x \frac{\cos\theta_y}{\|y - x\|^2} \,\mathrm{d}y. \tag{2.6}$$

$\mathcal{V}(x)$ denotes the set of all surface points y in the scene that are visible from x and

$$\frac{\mathrm{d}\omega_i}{\mathrm{d}y} = \frac{\cos\theta_y}{\|y - x\|^2} \tag{2.7}$$

is the Jacobian of the mapping from surface points to directions. It requires the cosine of the angle θ_y between $-\omega_i$ and the surface normal at y, as sketched in Figure 2.2. Appendix B provides a geometrical derivation of this Jacobian, and Appendix A provides a primer on integration by substitution in general.

In the literature, the Jacobian is often combined with the cosine term from the rendering equation, forming the *geometry term*

$$G(x, y) = \frac{\cos\theta_y \cos\theta_x}{\|x - y\|^2} = \cos\theta_x \frac{\mathrm{d}\omega_i}{\mathrm{d}y}. \tag{2.8}$$

Also, the integration domain is commonly written as the set $\mathcal{A}$ of all surface points in the scene, using the characteristic function of $\mathcal{V}(x)$, the *visibility term*

$$V(x, y) = \begin{cases} 1 & \text{if } y \text{ is visible from } x \\ 0 & \text{else,} \end{cases} \tag{2.9}$$

to equivalently rewrite the integral

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\mathcal{A}} V(x, y) G(x, y) \rho(\omega_i, x, \omega_o) L_i(\omega_i, x) \,\mathrm{d}y \tag{2.10}$$

as one over all surface points $y \in \mathcal{A}$ in the entire scene.

2.1.4 Path integral

The path integral offers a concise and convenient way to express the entire rendering operation. For that, we use the surface integral formulation for all integrals, including the ones from the camera. For the latter, we assume that the mapping from sensor position q to direction ω at the aperture point x_1 is bijective and hence invertible. Then, the integral for the camera model can be written as

$$I_i = \int_{\mathcal{A}} \int_{\mathcal{A}} V(x_1, x_2) W_i(x_1, \omega_\mathrm{i}) L_\mathrm{i}(x_1, \omega_\mathrm{i}) |J_\mathrm{cam}(x_2)| \mathrm{d}x_1 \mathrm{d}x_2. \tag{2.11}$$

Here, $|J_\mathrm{cam}(x_2)|$ denotes the Jacobian of the mapping from surface points x_2 to image plane positions q. The exact definition of that Jacobian differs between camera models; Appendix C provides a derivation for simple perspective cameras.

With this definition, we can expand the recursive surface integrals into an iterative form, by summing over all possible path lengths k and integrating over all paths of that length,

$$I_i = \sum_{k=2}^{\infty} \int_{\mathcal{A}^k} f(\overline{\mathrm{x}}) \mathrm{d}\overline{\mathrm{x}}. \tag{2.12}$$

Here, $\overline{\mathrm{x}}$ denotes a path of length k, $\mathrm{d}\overline{\mathrm{x}} = \mathrm{d}x_1 \cdots \mathrm{d}x_k$ is the product of the surface differentials, and

$$\begin{aligned} f_i(\overline{\mathrm{x}}) = & \underbrace{W_i(x_1 \to x_2) |J_\mathrm{cam}(x_2)| V(x_1, x_2)}_{\text{pixel response}} \\ & \underbrace{\left(\prod_{j=2}^{k-1} V(x_j, x_{j+1}) \rho(x_{j-1}, x_j, x_{j+1}) \cos\theta(x_j \to x_{j+1}) \frac{\cos\theta(x_{j+1} \to x_j)}{\|x_j - x_{j+1}\|^2} \right)}_{\text{throughput}} \underbrace{L_\mathrm{e}(x_k \to x_{k-1})}_{\text{emission}} \end{aligned} \tag{2.13}$$

is the path space integrand. We denote the direction from x_a to x_b

$$x_a \to x_b = \frac{x_b - x_a}{\|x_b - x_a\|}, \tag{2.14}$$

and

$$\cos\theta(x_a \to x_b) = \langle x_a \to x_b, n_a \rangle \tag{2.15}$$

is the cosine of the angle formed by that direction and the surface normal n_a at point x_a. The product of BSDFs, visibilities, and geometry terms is commonly referred to as the *throughput* of the path, as it measures the attenuation of the light along the path.

Rendering an image means computing the integral I_i for every single pixel. Unfortunately, that integral is infinite-dimensional, has scene-dependent discontinuities, and contains arbitrary black-box functions like the BSDF ρ, the sensor response W, and the emission profile L_e. Therefore, analytical integration is impossible and numerical methods are required.

2.2 Numerical integration

“Numerical integration” means computing the numerical value of a definite integral. Only two things are required for it to work: The value of the integral must be finite, and the

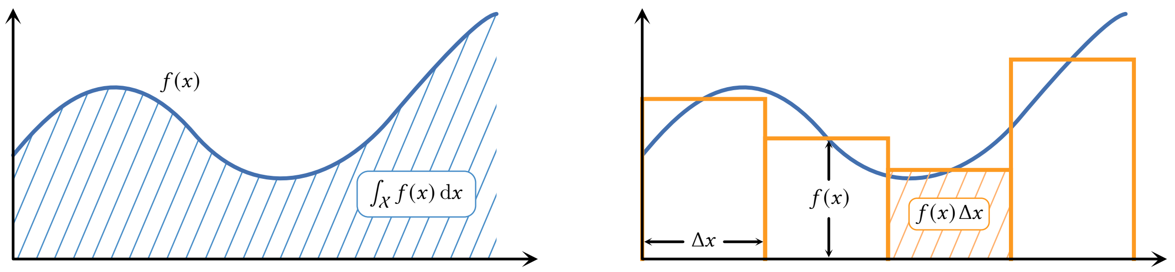


Figure 2.3: *Numerical integration via a Riemann sum. The integral is the area – or volume – under the curve of the function $f(x)$, as depicted on the left. Its numerical value can be approximated by fitting a series of simpler shapes to the function, like the rectangles shown on the right. The sum of areas of these rectangles approximates the area under the curve and hence the integral.*

function value must be computable for arbitrary points.

A range of different methods are available. While the details and performance aspects differ, these methods all utilize the same underlying principle: A series of *samples* is taken by evaluating the function at different positions. These positions can be deterministic, random, or a mix of both. The values of those samples are then used to approximate the integral. The various methods only differ in where the samples are placed and what basic shapes are used to approximate the integral.

Careful sample placement is the key to success of any numerical integration method. Deterministic placement can ensure high accuracy, but can get hampered by the curse of dimensionality. In contrast, random placement of samples – through Monte Carlo – sacrifices some accuracy but gains great flexibility and is much less affected by the dimensionality.

2.2.1 Deterministic quadrature

A straightforward approach to compute integrals is to directly apply their definition: the Riemann sum. Recall that the integral is the area under the curve – or the volume, for more than one dimension. A Riemann sum approximates this area or volume through discretization. The idea is illustrated in Figure 2.3: The domain is partitioned, the function is evaluated at a point within each part, and the integral over the part is approximated as a box.

The sum of rectangle areas / box volumes as sketched in Figure 2.3 converges to the integral as the width / area / volume Δx of the rectangle / box diminishes,

$$\lim_{\Delta x \to 0} \sum_i f(x_i)\Delta x =: \int_X f(x)\,\mathrm{d}x. \tag{2.16}$$

This is the well-known definition of the Riemann integral. Numerical integration with deterministic quadrature simply computes a finite approximation of this sum.

The downside of a deterministic subdivision approach is that sample locations are chosen in a rigid deterministic pattern. In a 1D example, that is not a problem and Riemann sums work perfectly. However, as dimensionality increases, distributing the samples regularly over the domain becomes more involved: The number of required samples to achieve the same resolution increases exponentially with the number of dimensions (see also Veach

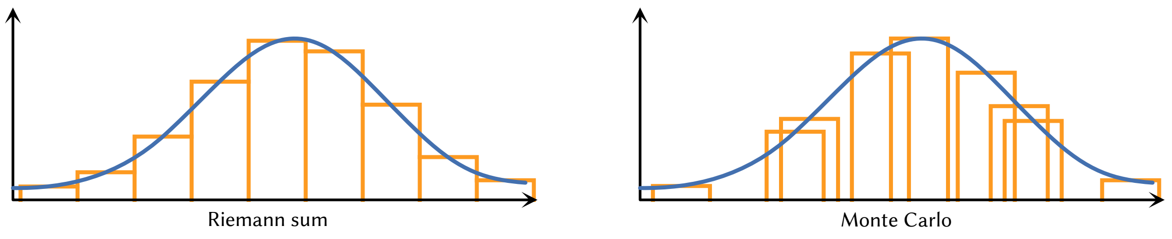


Figure 2.4: *Comparison of Monte Carlo integration to the underlying deterministic Riemann sum. The sole difference is that the approximating boxes are placed randomly instead of deterministically. This causes overlap and gaps, but also simplicity and flexibility.*

[1997, Section 2.2]). Further, growing dimensionality also hinders adaptive sample placement, where computation is focused on the most important regions, and it complicates progressive estimation, where early, low-sample estimates can be used, for example, to preview a rendered image before wasting too much computation time.

2.2.2 Monte Carlo integration

Monte Carlo methods, in general, add randomization to deterministic algorithms to simplify the implementation or facilitate performance improvements. In the context of numerical integration, Monte Carlo can be used to randomize the Riemann sum. This randomization breaks up the rigid structure, avoids the curse of dimensionality, and replaces alias with noise.

2.2.2.1 Intuition

The idea behind Monte Carlo integration is very simple. Instead of partitioning the domain and evaluating the integrand in a regular pattern, the sample positions are chosen randomly, as illustrated in Figure 2.4. On the one hand, this causes overlap and gaps between the boxes, increasing the error. On the other hand, samples can be generated and evaluated independently and in arbitrary numbers, greatly simplifying the implemenation and enabling a wide range of improvements.

The correctness of this Monte Carlo estimator can be unintuitive at first. How can we still get a valid approximation of the integral, if the approximating boxes overlap all the time? The answer is, that the expected error due to the overlap vanishes as the number of samples grows to infinity. This happens because two conditions are fulfilled: 1) the width of the approximating boxes vanishes, and 2) no two sample positions will be exactly the same, since the probability of that is zero. More formally, the correctness follows because the desired integral is the expected value of the randomized Monte Carlo estimator, and the law of large numbers guarantees convergence with growing sample counts.

2.2.2.2 Definition and convergence

Monte Carlo integration is based on a key theorem of probability theory: the law of large numbers. It states that when performing the same stochastic experiment many times, the average of the results will converge to the expected value [Ross 2014, p. 73]. Figure 2.5

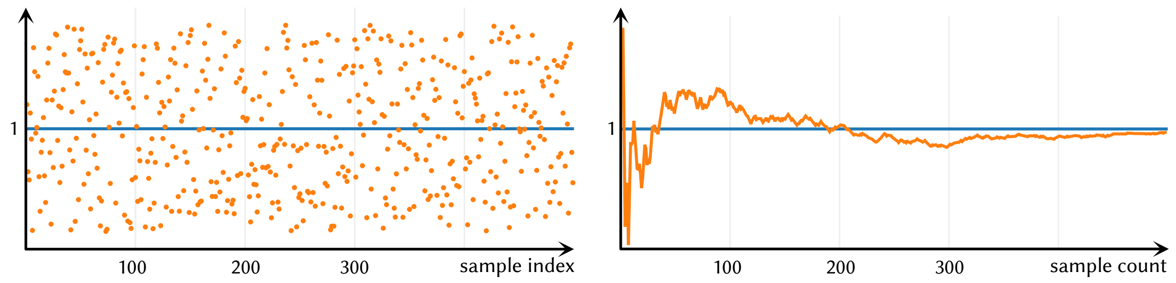


Figure 2.5: *The sample values of multiple independent runs (left) fluctuate around the expected value. Because of the law of large numbers, the average of many such runs (right) converges to the expectation as the sample count increases. Monte Carlo integration exploits that property and computes arbitrary integrals by formulating them as an expected value and simulating many samples.*

illustrates this. Individual simulations of a random variable fluctuate around the expected value; averaging these values converges to the expectation as the sample count increases.

But what does that have to do with integration? The expected value of a continuous random variable X is the integral over all possible values x multiplied by their respective probability density $p(x)$ [Ross 2014, p. 37]:

$$\mathrm{E}[X] = \int_{\mathcal{X}} x\, p(x)\, \mathrm{d}x. \tag{2.17}$$

The probability density function (PDF) $p(x)$ defines the distribution of the realizations x of the random variable. It is the continuous analog of a discrete probability.

We can compute this expected value by simulating n samples of the random variable and averaging their values X_i:

$$\mathrm{E}[X] \approx \frac{1}{n}\sum_{i=1}^{n} X_i, \quad \text{with } X_i \sim p(x). \tag{2.18}$$

The law of large numbers states that, with sufficient samples, this estimation yields a close approximation of the expected value. For example, to determine the expected amount of money lost when playing a slot machine in a casino, we can simply play very often and average the outcomes.

But we do not want to compute an expected value. Rather, we strive to compute an arbitrary integral

$$F = \int_{\mathcal{X}} f(x)\, \mathrm{d}x \tag{2.19}$$

of some function $f(x)$. Luckily, we can rewrite any integral as an expected value, by simply multiplying and dividing by a probability density $p(x)$:

$$F = \int_{\mathcal{X}} \frac{f(x)}{p(x)} p(x)\, \mathrm{d}x = \mathrm{E}\left[\frac{f(x)}{p(x)}\right]. \tag{2.20}$$

Therefore, we can estimate the integral F by generating many random points x following a probability density $p(x)$ and averaging the ratios of integrand and PDF values:

$$\langle F \rangle = \frac{1}{n} \sum_{i=1}^{n} \frac{f(x_i)}{p(x_i)}. \tag{2.21}$$

The law of large numbers guarantees that, for large enough n, this *Monte Carlo estimator* will be sufficiently close to its expected value, the desired integral F,

$$\lim_{n\to\infty} \langle F \rangle = F. \tag{2.22}$$

2.2.3 Benefits of Monte Carlo integration

A Monte Carlo estimator computes a randomized Riemann sum. This randomization offers many benefits, chiefly among which are the ease of implementation, the support of arbitrary sample counts, and the vast potential for adaptation. Also, the error distribution is more desirable, as stochastic noise is more amenable to denoising than deterministic alias.

2.2.3.1 Simplicity and flexibility

An important benefit of Monte Carlo integration is its sheer simplicity. Estimation requires only a random number generator and a means to evaluate the function. Samples can be generated sequentially in a loop or even in parallel, and the estimate can be accumulated in a single floating point value such that every intermediate result is a valid – though potentially inaccurate – approximation of the integral. Algorithm 1 shows this in pseudocode.

This simplicity also procures great flexibility. With a determinisic Riemann sum, the number of samples has to be fixed in advance. A valid approximation is achieved only once all samples have been computed. This is especially limiting at high dimensionality, where the number of samples required to achieve the same resolution grows exponentially. Figure 2.6 compares this visually on a 2D domain. Each dot marks a sample location; the rectangles visualize the approximating boxes. The columns show the first 10, 20, and 30 samples, respectively, out of a total budget of 100 samples; the top row shows a deterministic approach, the bottom a Monte Carlo method. The Monte Carlo estimate can be progressively refined, samples can be added in any number and at any time, and each intermediate result is a valid approximation of the integral.

The flipside of this flexibility is that sampling tends to be uneven: Samples clump together and leave wide, unexplored gaps in the domain. These irregularities cause higher estimation error at equal sample count than a determinisitic approach, especially for low-dimensional integrals.

2.2.3.2 Adaptivity

The flexibility of the Monte Carlo approach is important for adaptive computation. Integrals in rendering typically feature sparse, strong signals such as glossy reflections or small light sources. It is highly beneficial to adaptively focus computation in the vicinity of such sparse signals once they have been found.

With Monte Carlo integration, such adaptivity can be achieved easily. After a batch of sam-

Algorithm 1: *Pseudocode for a simple Monte Carlo estimator. Samples are added in a streaming fashion and the estimate is accumulated in a single number. At each update, we multiply by the old and divide by the new sample count to achieve progressive estimation.*

```
function MonteCarlo($f$, $p$, $n$)          ← Given an integrand, a PDF, and a number of samples
  $\tilde{F} = 0$
  for $i \in 1..n$ do
    $x = \text{Sample}(p)$                    ← Generate a random position with probability density $p$.
    $\tilde{F} \mathrel{+}= \frac{f(x)}{p(x)} \frac{i-1}{i}$     ← Correct sample count in each iteration for progressive estimation
  return $\tilde{F}$
```

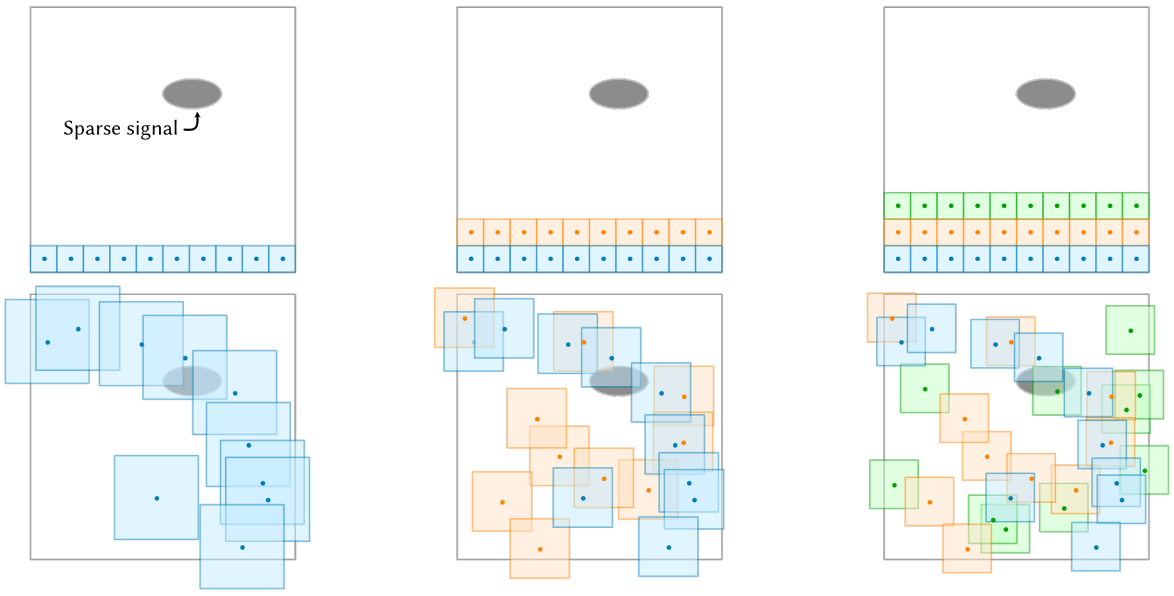


Figure 2.6: *Sample placement of a deterministic Riemann sum (top row) and a Monte Carlo estimator (bottom row). The dots mark the positions in the 2D domain where the integrand is evaluated, the rectangles visualize Δx, the base of the approximating box. The columns show the first 10, 20, and 30 samples, respectively, of a total budget of 90 samples. The deterministic approach is very rigid and initially lacks information on most of the domain, while random placement explores the whole domain from the start, refining the approximation over time.*

ples has been evaluated, the distribution of number of subsequent samples can be adapted. For example, in the scenario sketched in Figure 2.6, the sparse signal was finally found after 30 samples. At this point, we can update the sample density to more densely explore the vicinity of this first high-contribution sample.

Section 2.4.4 provides an overview of how such adaptation is achieved in rendering methods. The methods presented in Chapters 5 and 7 also fall into this category of adaptive solutions.

2.2.3.3 Noise

An added bonus is that the error of Monte Carlo integration manifests as noise, not alias. Figure 2.7 shows this on a simple example, where the integrand in each pixel of an image is a 1D normal distribution with shifting mean, as illustrated on the right. With deterministic quadrature, all pixels use the same sample positions. At low sample counts, this can

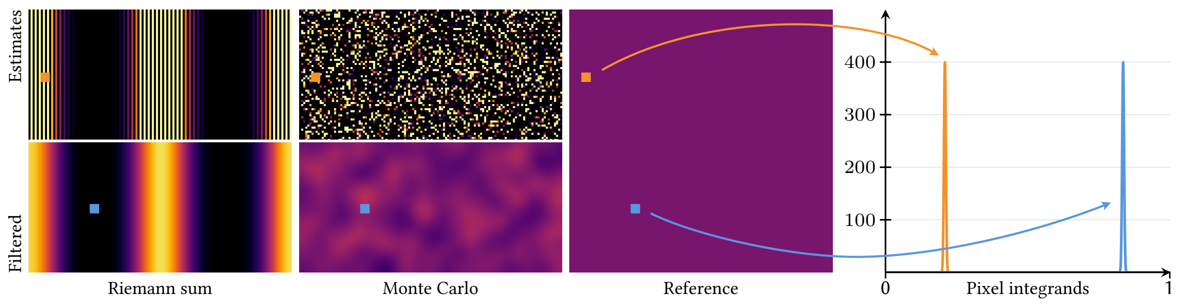


Figure 2.7: *Monte Carlo estimators produce noise instead of alias. Here, each pixel is given as the integral of a narrow 1D normal distribution shifted over the image, as shown for two example pixels on the right. That is, the value of the pixel marked in orange (top left) is the integral of the orange function; the value of the blue pixel (bottom right) that of the blue function. For both, the integration domain is the unit interval* $[0, 1]$. *The left two images are the rendered result with a deterministic Riemann sum and with Monte Carlo. In each case, the top half is the original image, the bottom half applies simple denoising (Gaussian filter). All images are false-color mapped for better visibility. Error with the Riemann sum manifests as stripes, due to the shifting integrand. Error in the Monte Carlo estimate manifests as stochastic noise. Filtering reduces the Monte Carlo error, but can amplify alias in the Riemann sum.*

manifest as severe alias, as shown in the first column. Due to the randomization of the sample positions, Monte Carlo integration does not suffer from that problem. Instead, the error manifests as noise.

Such stochastic noise can be more easily distinguished from the signal by a human observer. Further, because neighboring pixels use different sampling positions, their estimates can be blended together – for example, via a simple Gaussian blur as shown in the bottom halves – increasing the effective sample size in each pixel. Such *denoising* is extremely common in rendering applications, especially those that operate on a particularly tight budget. In contrast, the alias in deterministic quadrature cannot be removed through filtering, as shown in the bottom left.

2.2.3.4 Quasi-Monte Carlo.

The main limitation of Monte Carlo integration is that the random placement of sample points can increase the error. Intuitively, this can be observed on the illustration in Figure 2.4. With pure Monte Carlo, the boxes of the approximation can overlap and leave gaps, resulting in missing information about the integral in potentially important regions. A vast body of research on quasi-Monte Carlo methods tackles this problem. Instead of operating with random samples, these methods utilize deterministic sample sets or sequences [Keller 2013; Keller et al. 2019]. The evaluation positions are deterministic, but ordered such that the amount of information added by each sample is maximized. Thereby, benefits of both worlds can be combined.

2.3 Quantifying the efficiency

The challenge in rendering is not to construct *any* Monte Carlo estimator that can compute the pixel values. If that were the case, rendering would be a solved problem for decades now, since a correct solution has already been introduced in the 1980s [Kajiya 1986], along with the rendering equation itself. Instead, the goal is to construct an *efficient* estimator that computes accurate values within an acceptable time budget.

Efficiency is a pressing concern throughout all applications of Monte Carlo rendering. In real-time applications (e.g., visualizations and games), higher Monte Carlo efficiency directly translates to higher frame rates. For offline applications (e.g., movies), more efficient Monte Carlo rendering means lower cost, lower energy consumption, and leftover resources to invest towards higher fidelity.

Achieving higher efficiency is the goal of a vast body of research work, including the work reported in this thesis. Mathematically, efficiency can be quantified as the inverse product of cost C (aka render time) and variance V (aka expected squared error) [Hammersley and Handscomb 1968; Veach 1997],

$$\epsilon\left[\langle F\rangle\right] = \left(\mathrm{C}\left[\langle F\rangle\right]\ \mathrm{V}\left[\langle F\rangle\right]\right)^{-1}. \tag{2.23}$$

Maximum efficiency is achieved if the denominator is minimal. This product of cost and variance,

$$\mathrm{W}\left[\langle F\rangle\right] = \mathrm{C}\left[\langle F\rangle\right]\ \mathrm{V}\left[\langle F\rangle\right], \tag{2.24}$$

is sometimes referred to as the *work-normalized variance* (e.g., by Glynn and Whitt [1992]).

Methods aiming to improve efficiency can do so by reducing the cost, reducing the variance, or jointly reducing both. Often, different factors have to be weighed against each other. For example, more involved sampling methods like bidirectional rendering algorithms incur higher cost but reduce the variance. In that case, maximum efficiency is achieved by finding the right trade-off.

2.3.1 Cost

The cost is generally a function of the sample count, $\mathrm{C}\left[\langle F\rangle\right] = \mathrm{C}(n)$. Often, but not always, it is a linear function, $\mathrm{C}(n) = cn + c_0$, where c is the per-sample cost and c_0 a fixed overhead. Notable exceptions in rendering are sample reuse methods that build and traverse acceleration structures over samples and hence have $O(n \log n)$ cost complexity, like photon mapping [Jensen 1996].

Minimization of cost is usually done through low-level optimizations; for example, by reducing the cost of each ray-tracing operation, or the cost of material evaluations and texture lookups. Such optimizations increase the efficiency, because a higher number of samples can be created within the same time budget, resulting in a lower final error.

Overall, the cost is governed by many details of the algorithm and its implementation; it cannot be studied in a generalized fashion. Nevertheless, for specific applications, it is always important to keep the cost in mind. An elaborate adaptive sampling scheme might yield incredible variance reduction, but do so at prohibitive cost. Such a seemingly great algorithm is then easily outperformed by a simpler method.

There are many complex low-variance rendering algorithms, yet production often resorts

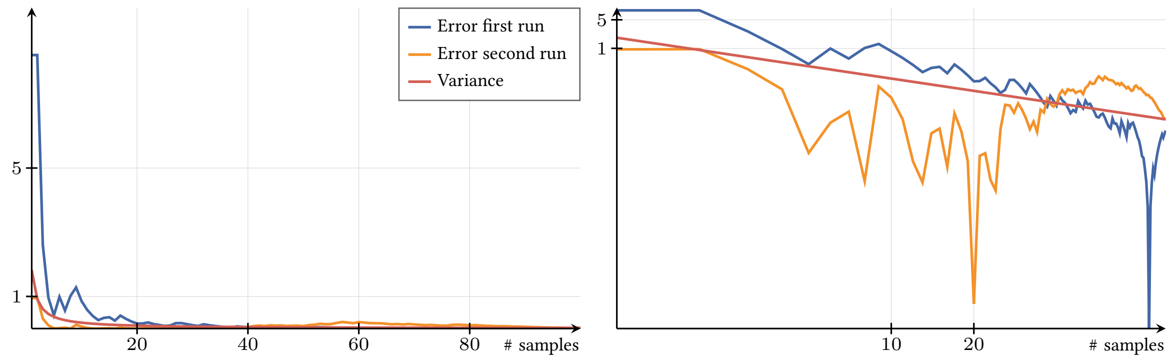


Figure 2.8: *The squared error of a Monte Carlo estimator as a function of sample count. The blue and orange curves are the actual error of two runs of the estimator, the red line marks the expected error, that is, the variance. Both plots show the same values, but the right-hand one uses a log-log scale for better visibility. The logarithmic scale counters the hyperbolic shape and helps to better distinguish different error curves. Here, we estimate the integral of a 1D Gaussian via uniform Monte Carlo sampling, though results will be similar for* any *integral and Monte Carlo estimator.*

to simple unidirectional path tracing [Fascione et al. 2018]. Often, this is because the higher cost of such involved solutions does not pay off in every scene. Chapter 7 tackles this problem by adapting the algorithm to the scene, avoiding overhead from costly techniques that are not required for the current scene.

2.3.2 Variance

Monte Carlo integration is a random process. Hence, its *variance* is a natural way to quantify the error. The variance is the expected value of the squared error,

$$\mathrm{V}\left[\langle F\rangle\right] = \mathrm{E}\left[(\langle F\rangle - F)^2\right]. \tag{2.25}$$

If the n samples of a Monte Carlo estimator are mutually independent and identically distributed, the variance

$$\mathrm{V}\left[\langle F\rangle\right] = \mathrm{V}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{p(x_i)}\right] = \frac{1}{n}\mathrm{V}\left[\frac{f(x)}{p(x)}\right] \tag{2.26}$$

is $1/n$ times that of an estimate with just a single sample. This provides us with the convergence rate of a basic Monte Carlo estimator. Figure 2.8 visualizes this convergence rate on an example. The blue and orange lines denote the acutal squared error of two independent runs of the same Monte Carlo estimator. Each run computes an estimate from the same number of samples, but seeds the (pseudo) random number generator differently. We compute and plot the error after each added sample. The red line marks the expected error, that is, the variance. This example plot provides two important insights. First, care has to be taken when evaluating the performance of different Monte Carlo estimators based on the error they achieved after a single run. Such evaluation is common in rendering, where repeatedly running the same algorithm can be expensive. Especially at low sample counts, this error value can be misleading. Second, the hyperbolic shape of the error can be cumbersome to interpret. Visualization on a logarithmic scale is helpful for that.

Convergence can be faster or slower than this baseline if the samples are correlated. With correlated samples, the variance can still be written in terms of the single-sample variance, but an additional covariance term arises,

$$\mathrm{V}\left[\langle F\rangle\right] = \frac{1}{n}\mathrm{V}\left[\frac{f(x)}{p(x)}\right] + \frac{1}{n^2}\sum_{i\neq j}\mathrm{Cov}\left(\frac{f(x_i)}{p(x_i)}, \frac{f(x_j)}{p(x_j)}\right), \tag{2.27}$$

where the covariance

$$\mathrm{Cov}\left(\frac{f(x_i)}{p(x_i)}, \frac{f(x_j)}{p(x_j)}\right) = \mathrm{E}\left[\frac{f(x_i)}{p(x_i)}\frac{f(x_j)}{p(x_j)}\right] - \mathrm{E}\left[\frac{f(x_i)}{p(x_i)}\right]\mathrm{E}\left[\frac{f(x_j)}{p(x_j)}\right] \tag{2.28}$$

is the difference between the expectation of the product and the product of expectations. The exact form of this covariance depends on the nature of the correlation (see Section 2.3.4).

Overall, the variance comprises three terms: the second moment, the squared mean, and the covariance,

$$\boxed{\mathrm{V}\left[\langle F\rangle\right] = \underbrace{\frac{1}{n}\int_{\mathcal{X}}\frac{f^2(x)}{p(x)}\,\mathrm{d}x}_{\text{second moment}} - \underbrace{\frac{1}{n}F^2}_{\text{squared mean}} + \underbrace{\frac{1}{n^2}\sum_{i\neq j}\mathrm{Cov}\left(\frac{f(x_i)}{p(x_i)}, \frac{f(x_j)}{p(x_j)}\right)}_{\text{covariance}}. \quad (2.29)}$$

Among these three terms, the second moment is generally easiest to compute, while the covariance can be particularly difficult, especially for high-dimensional problems.

2.3.3 Second moment

The first term in the variance, the second moment, is often used as an approximation of the full variance. It is straightforward to compute through Monte Carlo estimation, as it is itself an integral.

Definition. The second moment (about zero) of a single-sample Monte Carlo estimator (aka a primary estimator) is the expectation of its squared value,

$$\mathrm{M}[\langle F\rangle_1] = \mathrm{E}[\langle F\rangle_1^2] = \int_{\mathcal{X}}\frac{f^2(x)}{p(x)}\,\mathrm{d}x. \tag{2.30}$$

In most of the rendering research literature, the term "second moment" for n-sample estimators refers to this primary second moment divided by the sample count,

$$\mathrm{M}[\langle F\rangle_n] := \frac{1}{n}\mathrm{M}[\langle F\rangle_1] = \int_{\mathcal{X}}\frac{f^2(x)}{np(x)}\,\mathrm{d}x, \tag{2.31}$$

because the actual second moment about zero of an n-sample estimator,

$$\mathrm{E}[\langle F\rangle_n^2] = \mathrm{M}[\langle F\rangle_n] + \frac{1}{n^2}\sum_{i\neq j}\mathrm{Cov}\left(\frac{f(x_i)}{p(x_i)}, \frac{f(x_j)}{p(x_j)}\right) + \left(1-\frac{1}{n}\right)F^2, \tag{2.32}$$

is not typically a quantity of interest, as it involves the same terms as the full variance. We follow this nomenclature.

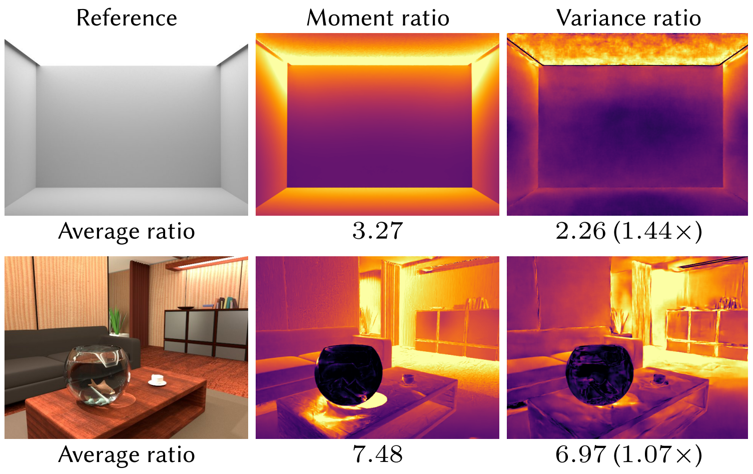


Figure 2.9: *Approximation error when operating with second moments in lieu of the full variances. The false color images show the ratio between second moments and variances of two different Monte Carlo samplers. Ideally, the center and right images should be identical, indicating that the approximation is perfect. The numbers below each image provide the average ratio. The second moment approximation can be rather poor for simple scenes like the one in the first row, where the overall variance is low. In more realistic scenes, like the one in the second row, the approximation works well almost everywhere.*

Approximation. The second moment can be used as an approximation of the variance,

$$\mathrm{V}[\langle F\rangle_n] \approx \mathrm{M}[\langle F\rangle_n]. \tag{2.33}$$

The approximation error is governed by the remaining two terms in the full variance (2.29): the squared mean and any covariance between the samples,

$$\mathrm{V}[\langle F\rangle_n] - \mathrm{M}[\langle F\rangle_n] = -\frac{1}{n}F^2 + \mathrm{Cov}. \tag{2.34}$$

This approximation assumes that the covariance is low. Then, the accuracy is highest, if the overall variance is much larger than the squared mean,

$$\mathrm{V}[\langle F\rangle_n] \gg \frac{1}{n}F^2. \tag{2.35}$$

In other words, the second moment is well suited to approximate the error of bad, that is, high-variance, estimators. It is an unfortunate fact that most Monte Carlo estimators in rendering have high variance. The silver lining is that these can be effectively optimized based solely on the second moment. Figure 2.9 demonstrates the effectiveness – and limitations – of this approximation on rendering examples. The false-color images show the ratio of second moments and variances, respectively, for two different Monte Carlo estimators. Ideally, the two images should be identical. In the first example, of a simple diffuse box with a single large light source, the variance is low. The approximation through the second moment fares badly in this case. In more realistic scenes, like the second example, variance will typically be high almost everywhere. There, the approximation works well except for a small region in the bottom left. Chapter 5 shows how understanding this approximation error can help improve methods that rely on the second moment approximation.

Use-cases. The second moment approximation has been successfully applied to optimize the efficiency of Monte Carlo rendering algorithms. A ubiquitous example is the classic balance heuristic for multiple importance sampling [Veach and Guibas 1995b]. It arises as the analytic solution when optimizing the second moment (see Section 3.2.2.1). Another example is the method presented in Chapter 7: It uses numerical optimization, estimating the second moment of different algorithm configurations on-the-fly to adapt to the scene.

2.3.4 Correlation and covariance

The samples of a Monte Carlo estimator do not have to be statistically independent. In fact, there are many good reasons why they might be correlated. Sometimes, such correlation is introduced deliberately. Other times, it is a byproduct of another efficiency-enhancing technique.

Correlation can be positive, which means it increases the variance, or negative, which means the variance is reduced. But, when used wisely, even positive correlation can increase the efficiency. For example, sample reuse introduces positive correlation but simultaneously reduces the sampling cost.

Mathematically, correlation can be quantified via the Pearson correlation coefficient

$$\rho(X_i, X_j) = \frac{\mathrm{Cov}(X_i, X_j)}{\sqrt{\mathbb{V}[X_i]\mathbb{V}[X_j]}} \in [-1, 1], \tag{2.36}$$

namely, the covariance divided by the product of standard deviations. A non-zero correlation means that the variance of an n-sample estimator (2.29) has a non-zero covariance term. If correlation is positive, the variance is higher than with independent samples. If it is negative, the variance is lower.

In rendering applications, there are two main types of correlations: correlation between pixels and correlation between samples of the same pixel. Understanding the sources, magnitudes, and associated trade-offs of these correlations helps design more efficient algorithms. An example is the method presented in Chapter 6: It uses an understanding of the covariance in the photon mapping technique to rectify an unnecessary increase in variance.

2.3.4.1 Sample correlation

Samples used for the same estimator $\langle F \rangle$ can be correlated. Negative correlation is commonly introduced deliberately as a variance reduction strategy, while positive correlation is a common side-effect of methods that amortize cost to increase efficiency. Both types of correlations are ubiquitious in rendering algorithms. For example, variance reduction techniques like stratified sampling and antithetic variates introduce negative correlation to reduce the variance, while splitting and the closely related sample reuse techniques are common examples for positive correlation as a byproduct of cost amortization.

Intuitively, correlation measures how much the samples cluster together. If the values of two samples X_i and X_j are always very similar, then they are positively correlated. Conversely, if the sample values are always far apart, then they are negatively correlated. If the values are sometimes similar and sometimes far apart, then there is no correlation.

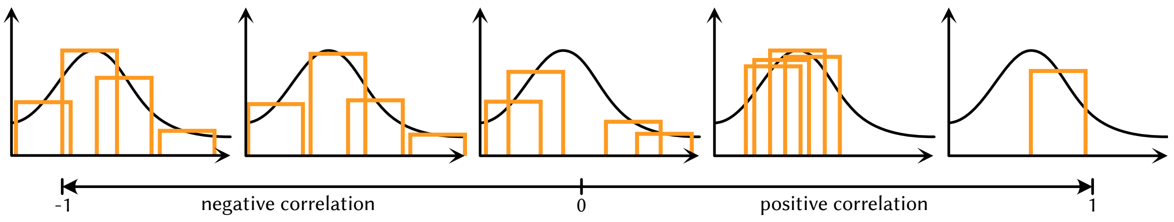


Figure 2.10: *Visual interpretation of sample correlation for 1D integration. Each plot shows possible sample locations and values with different levels of correlation. With maximum negative correlation (left) sample values (i.e., rectangle heights) are far apart from each other. With maximum positive correlation (right) sample values are identical.*

Monte Carlo estimators with negatively correlated samples converge faster than estimators with independent samples. Intuitively, this can be understood by recalling a main drawback of Monte Carlo integration: The approximating boxes can overlap and leave gaps. With high positive correlation, these boxes are almost identical – the worst case approximation. But if the samples are negatively correlated, then the boxes have minimal overlap. Figure 2.10 illustrates this on a simple example.

Effective sample size. Correlation has a direct impact on the convergence rate. This effect can be quantified through the notion of *effective sample size*. It measures the number of independent samples required to achieve the same variance,

$$\mathrm{V}[\langle F\rangle] = \frac{1}{n_{\mathrm{eff}}}\mathrm{V}\left[\frac{f(x)}{p(x)}\right] \Leftrightarrow n_{\mathrm{eff}} = \frac{\mathrm{V}\left[\frac{f(x)}{p(x)}\right]}{\mathrm{V}[\langle F\rangle]}. \tag{2.37}$$

With independent samples, the effective sample size is simply the number of samples, $n = n_{\mathrm{eff}}$. If the samples are positively correlated, then the effective sample size diminishes,

$$\rho\left(\frac{f(x_i)}{p(x_i)}, \frac{f(x_j)}{p(x_j)}\right) > 0 \Rightarrow n_{\mathrm{eff}} < n. \tag{2.38}$$

In the worst case, if $\rho = 1$, the effective sample size will be $n_{\mathrm{eff}} = 1$. Conversely, if the correlation is negative, the effective sample size increases. In theory, if $\rho = -1$, then the variance will be zero, and the effective sample size will be infinite. Variance reduction techniques like stratification and antithetic sampling increase the effective sample size, while cost amortization methods such as splitting and reuse reduce it to simultaneously reduce cost.

Stratification. Stratification is a common variance reduction technique that produces negative sample correlation. It operates by first partitioning the domain into so-called strata. Then, a fixed number of samples is taken from each stratum, as illustrated in Figure 2.11. Forcing samples to lie in different strata causes negative covariance and results in a convergence rate faster than that of independent samples, as plotted on the right. Unfortunately, much like deterministic quadrature, stratification suffers from the curse of dimensionality. Nevertheless, it can prove useful for low-dimensional integrals or when applied only along a subset of all dimensions.

Antithetic sampling. Antithetic variates are another variance reduction technique, especially suited for symmetric integrands. The idea is illustrated in Figure 2.12. Whenever a

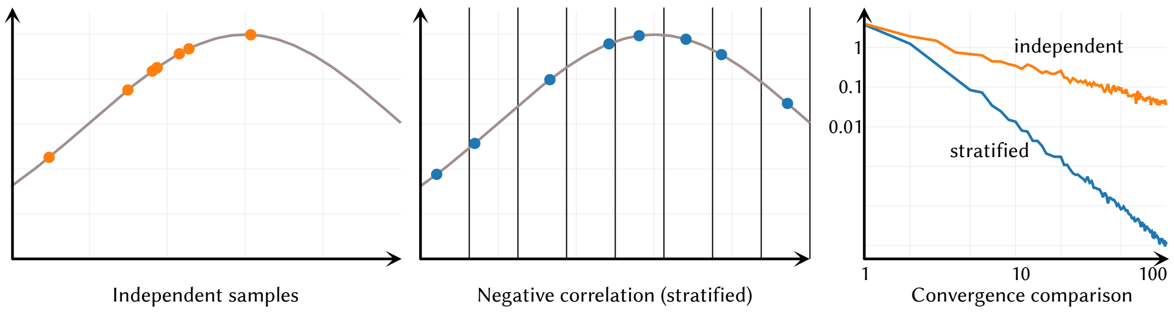


Figure 2.11: *Stratified sampling yields negatively correlated samples and thus a faster convergence rate. Independent samples, shown on the left, tend to clump together and leave gaps. Stratification improves that by partitioning the domain and taking exactly one sample in each part. The resulting negative correlation yields a convergence rate faster than the* $1/n$ *of independent sampling, as shown on the right.*

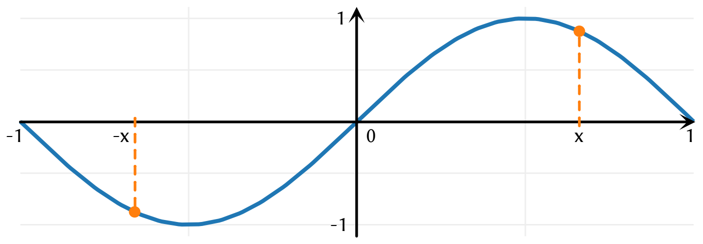


Figure 2.12: *Antithetic sampling takes one sample* x *and also evaluates its antithetic position* $-x$*. In this idealized example of a point-symmectric integrand, the result is a zero-variance estimator due to maximum negative correlation. This holds* no matter what density *the points* x *have.*

sample x is taken, its antithetic value $-x$ is also evaluated. The variance of this sample pair contains a negative covariance term, hence it is lower than the variance of two independent samples. In fact, in our example here, the integrand is point symmetric, $f(-x) = -f(x)$, and the variance with just two antithetic samples is zero. This is because the covariance

$$\operatorname{Cov}\left(\frac{f(x)}{p(x)}, \frac{f(-x)}{p(x)}\right) = -\mathrm{E}\left[\left(\frac{f(x)}{p(x)}\right)^2\right] + F^2 = -\mathrm{V}\left[\frac{f(x)}{p(x)}\right] \tag{2.39}$$

is minus the variance. Hence, the samples in this example have maximum negative correlation. Intuitively, this can be explained by observing that the integral here is zero, and that the two samples in each pair always cancel each other out, so the estimate computed from every possible pair is always exactly zero. Unfortunately, antithetic sampling is not easily applied to arbitrary integration problems, as it relies on exploiting symmetry in the integrand. In rendering it is scarcely applied. One example is the computation of signed integrals in differentiable rendering [Zeltner et al. 2021; Zhang et al. 2021].

Splitting. Splitting increases variance through positive correlation, but trades this for reduced computation cost. In rendering applications – and other Monte Carlo simulations – it can be beneficial to continue a prefix subpath with multiple suffixes [J. R. Arvo and Kirk 1990; Rath et al. 2022; Vorba and Křivánek 2016]. This process is known as splitting and illustrated in Figure 2.13. A prefix $\overline{\mathrm{y}}$ is continued with n suffixes $\overline{\mathrm{z}}_i$ to form n full paths $\overline{\mathrm{y}\mathrm{z}}_i$. These full samples are positively correlated, because they all share the same prefix $\overline{\mathrm{y}}$. Therefore, the variance of an estimator with these n samples will be higher than if n independent samples had been used. However, because only a portion of the full path is

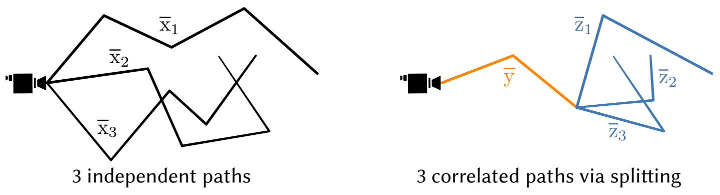


Figure 2.13: *Splitting generates multiple (here three) full paths $\overline{x}$ by continuing a single prefix $\overline{y}$ with multiple suffixes $\overline{z}$. The resulting full samples are correlated, because they all share the same prefix. Splitting is beneficial if the prefix $\overline{y}$ is costly to sample but has low variance.*

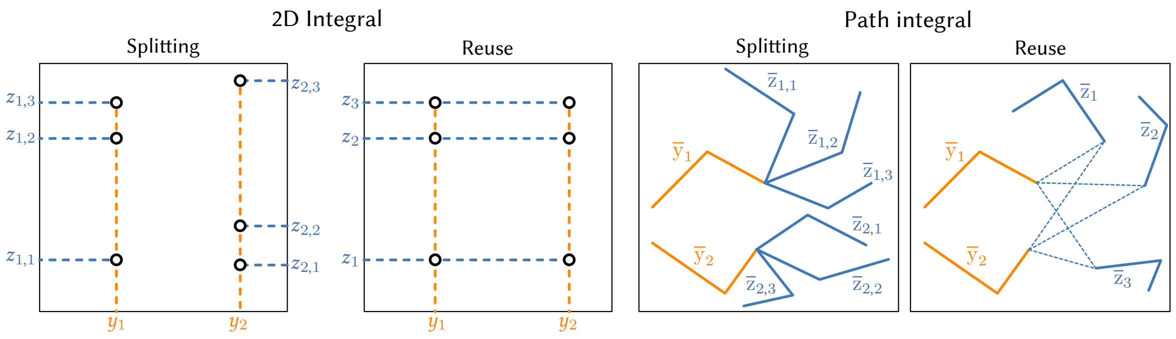


Figure 2.14: *The cost of splitting is often amortized through reuse. Here we compare pure splitting and splitting with reuse on a 2D domain (left) and in path space (right). Splitting continues each prefix with a new set of suffixes, reuse additionally shares the suffixes across all prefixes.*

sampled n times, the cost of the correlated samples is lower than the cost of independent ones. Thus, if done well, splitting can increase efficiency. Unfortunately, it is not a trivial task to perform splitting efficiently [Bolin and Meyer 1997; Rath et al. 2022; Vorba and Křivánek 2016].

Reuse and resampling. Sample reuse combines many different prefixes $\overline{y}$ with the same suffixes $\overline{z}$. This is illustrated in Figure 2.14. A simple splitting estimator would generate a new set of suffixes for each prefix sample. In contrast, a reuse estimator uses the same set of suffixes for all prefixes. This introduces positive correlation between the full paths $\overline{x}$ since they now all share the same suffixes. This correlation can be small, if the set of suffixes is vast and only a small portion is chosen randomly to continue each prefix. This is typically the case in practical applications, because sample reuse estimators typically generate thousands to millions of suffix paths and use filtering (e.g., photon mapping [Jensen 1996] or path caching [Keller et al. 2014; West et al. 2020]) or resampling (e.g., light path resampling for virtual point lights [Georgiev et al. 2012b] or bidirectional path tracing [Su et al. 2022], or ReSTIR [Bitterli et al. 2020] where (sub)paths in nearby pixels are resampled spatio-temporally) to select a subset of those. The benefit of such reuse is that the cost is ammortized over all prefix samples, resulting in a lower variance than a single independent path without increasing the cost. These suffix samples are usually also shared between multiple pixels, hence sample reuse can also introduce pixel correlation.

2.3.4.2 Pixel correlation

Pixel correlation can be a great asset or a major problem. But why is it relevant? After all, our goal is to compute the individual pixel integrals, and the variance of those is unaffected

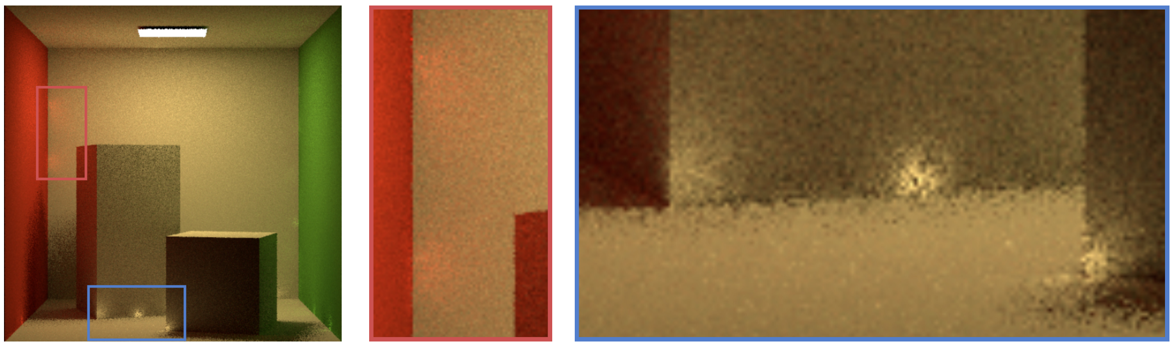


Figure 2.15: *Example for positive correlation between pixels. The path resampling method of Popov et al. [2015] shares a small set of light subpaths across all pixels. Further, nearby pixels share the same resampling probability distribution. The result is positive pixel correlation, manifesting as splotches.*

by correlations between pixels. The answer is twofold: human perception and denoising.

Denoising is a post-processing step that reduces the variance by blurring the image. There are many different methods [Zwicker et al. 2015] that differ in many details. However, the core principle is always the same: Pixel estimates are filtered with a low-pass kernel. The simplest form of denoising would be a blur with constant radius applied over the entire image. Blurring reduces the noise in each pixel by sharing samples between neighboring pixels.

Denoising is more effective if nearby pixels have more information to share. In one extreme, if nearby pixels all generated the same samples, there is no information to share and denoising cannot improve the estimates. In the other extreme, if the samples of nearby pixels are as far apart as possible, they each add valuable information that denoising can exploit. The first case arises if the pixel value estimates are positively correlated, the second if they are negatively correlated.

Noisy images with negative pixel correlation are also better suited for human perception. Human perception of noisy images can be modelled via a low-pass filter [Chizhov et al. 2022]. Hence, when viewed directly, noisy images with negative correlation are percieved as less noisy than equal-error images with no – or positive – correlation.

Positive correlation. Positive correlation occurs if two pixels share some of their sampling information. Popular examples in rendering are bidirectional algorithms like photon mapping [Jensen 1996], virtual point light methods [Keller 1997], and Markov chain Monte Carlo methods [Šik and Křivánek 2018; Veach and Guibas 1997]. Photon mapping and virtual point lights use the same samples in neighboring pixels, and Markov chain Monte Carlo methods use slightly mutated versions of the same samples. As a consequence, the pixel estimates are always similar, resulting in splotchy artifacts, as shown in Figure 2.15. This is akin to the alias with deterministic quadrature (see Figure 2.7): Naïve deterministic sampling uses the exact same sample positions in all pixels, resulting in maximum positive correlation. Positive correlation between pixels is best avoided. The splotchy artifacts are unappealing to human observers, and reconstruction through filtering is impossible, since neighboring pixels do not have mutually-beneficial information they could share with each other. Still, algorithms with positive pixel correlation might converge more rapidly and can hence be beneficial for long-running render tasks. Often, it is possible to reduce positive pixel correlation through additional randomization while keeping some of the performance benefits. For example, Estevez and Kulla [2020] use only a random subset of all nearby photons for photon mapping. This randomization removes the splotchy artifacts and gives control to the denoiser to decide what amount of blur is acceptable.

Negative correlation. Negative correlation between pixels is a great asset for reconstruction. If the samples of nearby pixels are as different as possible, the total amount of information obtained by the estimates of these pixels is maximized. Unfortunately, achieving such negative correlation is not a trivial task. There are multiple approaches, and finding the best solution is still an open research problem. One option is to utilize precomputed blue noise patterns [Georgiev and Fajardo 2016; Heitz et al. 2019; Wolfe et al. 2022] or to modify a quasi-Monte Carlo sample sequence [Ahmed and Wonka 2020]. Another approach is to adapt the sample positions based on previous iterations [Heitz and Belcour 2019], or to perform error distribution in a post-process [Chizhov et al. 2022]. Negative pixel correlation is especially useful for low sample count renderings, such as real-time or interactive applications, where strong filtering is required for acceptable image quality.

Quantifying pixel correlation. Mathematically, correlation between pixels can be quantified by looking at the error of the reconstructed image. Image reconstruction applies a (low-pass) kernel $k(i, j)$ and reconstructs the value of the ith pixel as the weighted sum of all pixels within the kernel footprint,

$$\tilde{F}_i = \sum_j k(i, j)\langle F_j \rangle. \tag{2.40}$$

The variance of this reconstruction is the weighted sum of variances and covariances,

$$\mathrm{V}[\tilde{F}_i] = \sum_j k^2(i, j)\mathrm{V}[\langle F_j \rangle] + 2\sum_{j \neq k} k(i, j)k(i, k)\mathrm{Cov}(\langle F_j \rangle, \langle F_k \rangle). \tag{2.41}$$

If the covariance is negative, the reconstruction has a lower variance than a reconstruction with independent samples. If the covariance is postive, reconstruction will be worse. Previous work has optimized this reconstructed error by, for example, imposing a kernel and optimizing the sampling pattern for that kernel [Chizhov et al. 2022].

2.4 Improving the efficiency

Much work has been done on improving the efficiency of rendering algorithms. In this thesis, we focus on high-level algorithmic tools that either reduce the variance or achieve a better trade-off between variance and cost. Importance sampling and control variates are examples for the former; adaptive sample counts an example for the latter. These techniques are made possible through the flexibility procured by Monte Carlo integration.

Importance sampling (Section 2.4.1) is maybe the best-known technique to reduce variance. It operates by using a well-chosen sample distribution rather than a uniform one. From a calculus perspective, importance sampling can be seen as an application of integration by substitution. Consequently, importance sampling is deeply tied to the various integral formulations that are used for rendering applications, such as surface or direction integrals.

The method of control variates (Section 2.4.3) is another variance reduction technique with uses in rendering applications. Instead of computing the full integral, we can subtract a

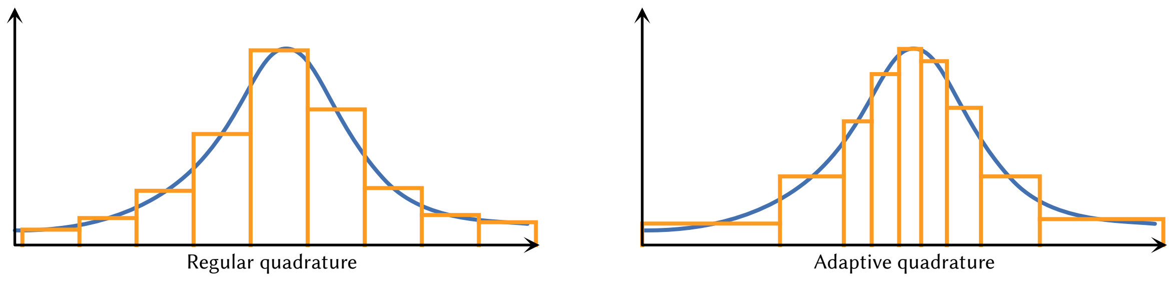


Figure 2.16: *Comparison of regular and adaptive quadrature on the example of a deterministic Riemann sum. Sampling the integrand more densely in important regions, as done on the right, produces more accurate estimates with the same computation budget.*

known integral and compute the difference. If that difference is small, then it is easier to compute than the original integral, hence variance will be lower. In rendering, this has, for example, been used for relighting applications [Rousselle et al. 2016], where the same image is rendered again under different illumination, so the previous illumination can be used as a control variate.

The integrals computed in rendering applications are far too diverse and complex to find a single a-priori method that achieves perfect efficiency. Therefore, a lot of work has been done on adaptive methods (Section 2.4.4). These aim to improve the sampling distributions, control variates, or sample counts on-the-fly during rendering.

2.4.1 Importance sampling

Adaptive computation is paramount for efficient numerical integration. Uniform sampling of the integration domain leads to wasted effort in unimportant regions or insufficient precision in important ones. This is illustrated in Figure 2.16, which compares quadrature approximations of the same function using uniform sampling on the left and adaptive sampling on the right. The integration error is reduced by sampling important regions – namely, those with high integrand values – more densely. In the context of Monte Carlo integration, such adaptation is achieved by using a non-uniform PDF $p(x)$. This is known as *importance sampling*.

Optimal importance sampling can, in theory, provide a perfect result with just a single sample. But for that, it requires prior knowledge of the desired integral, so optimal sampling is generally impossible. Nevertheless, it is insightful to understand the optimal density, as it can inform the choice of good approximations for variance reduction in practice.

Given a reasonable importance sampling PDF, the final step is to synthesize the samples. To that end, uniform random numbers in primary sample space are transformed to the integration domain with specially designed sample transformations. This step can be very involved and expensive.

2.4.1.1 Optimal importance sampling

Intuitively, a good importance sampling PDF should be high where the integrand is large, and low where the integrand is small. So it is not surprising that the ideal importance sampling density is proportional to the integrand, $p(x) \propto f(x)$. The following provides a formal derivation of this optimum.

The best importance sampling PDF is the one that minimizes the variance of the estimator,

$$\hat{p}(x) = \operatorname*{arg\,min}_{p(x)} \left(\mathrm{V}\left[\frac{f(x)}{p(x)}\right] + \lambda \left(\int_{\mathcal{X}} p(x)\, \mathrm{d}x - 1 \right) \right) \quad \text{subject to } p(x) > 0, \tag{2.42}$$

where a Lagrange multiplier is used to ensure the equality constraint

$$\int_{\mathcal{X}} p(x)\, \mathrm{d}x = 1, \tag{2.43}$$

that is, that p is a valid probability density. This is a convex optimization problem, so the minimizer can be easily found by setting the functional derivatives w.r.t. $p(x)$ to zero,

$$-\frac{f^2(x)}{p^2(x)} + \lambda = 0 \tag{2.44}$$

$$\int_{\mathcal{X}} p(x)\, \mathrm{d}x - 1 = 0. \tag{2.45}$$

The first equation tells us that the lowest variance is achieved if the PDF is proportional to the absolute value of the integrand,

$$p(x) = \frac{|f(x)|}{\sqrt{\lambda}} \propto |f(x)|. \tag{2.46}$$

The normalization factor $\sqrt{\lambda}$ is found by substituting this result into the second equation,

$$\sqrt{\lambda} = \int_{\mathcal{X}} |f(x)|\, \mathrm{d}x. \tag{2.47}$$

For positive integrands, $\sqrt{\lambda} = F$ is the integral we strive to compute. Therefore, the optimal

$$\boxed{\hat{p}(x) = \frac{|f(x)|}{\int_{\mathcal{X}} |f(x)|\, \mathrm{d}x},} \tag{2.48}$$

can only be computed if we already know the desired integral. So importance sampling with the optimal PDF is not possible in practice. However, substituting approximations of f with known integral has proven effective: Even crude approximations can provide notable variance reduction compared to uniform sampling.

2.4.1.2 Transforming samples

Our goal is to importance sample with some density $p(x)$. But how can we achieve that? Typical (pseudo-) random number generators only produce numbers between zero and one that follow a uniform distribution. That is, we can easily generate *primary samples*

$$u \in \mathcal{U} = [0, 1]^d, \qquad p(u) = 1, \tag{2.49}$$

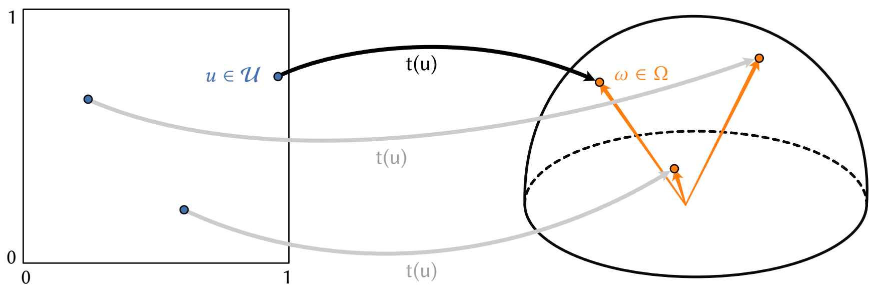


Figure 2.17: *In practice, importance sampling is achieved by transforming uniform samples $u \in \mathcal{U}$ from the unit hypercube to the integration domain, using a transformation $t(u)$. Here, directions $\omega = t(u)$ are sampled by transforming uniform samples from the 2D unit square $(0,1) \times (0,1)$ to the surface of the unit hemisphere Ω, where they implicitly define directions in 3D space.*

by using one uniform random value for each of d dimensions. This unit hypercube $\mathcal{U}$ is called the *primary sample space* [Kelemen et al. 2002]. For importance sampling, the primary samples must be transformed to the integration domain $\mathcal{X}$ such that they have the desired density. Figure 2.17 visualizes this on the example of sampling a cosine-weighted direction from the hemisphere.

To obtain samples from the target domain $\mathcal{X}$, we must design a transformation $t : \mathcal{U} \to \mathcal{X}$. This transformation must ensure that uniformly distributed points in u yield points in $\mathcal{X}$ that are distributed according to the importance sampling density $p(x)$.

There are multiple methods to find such a transformation; the preferred method in rendering is *inversion sampling*. Inversion sampling constructs a transformation $x = t(u)$ by inverting the cumulative distribution function (CDF) of the importance sampling PDF [Pharr et al. 2016, Chapter 13.3]. This inversion is not always a trivial task [Heitz 2020], but offers important benefits compared to alternatives such as rejection sampling or tabulated densities: It has low memory requirements, does not waste primary samples, and preserves stratification. Inversion sampling can be derived through integration by substitution.

2.4.2 Integration by substitution

Integration by substitution (aka change of variables) uses an injective mapping $t : \mathcal{U} \to \mathcal{X}$ to write an integral over $\mathcal{X}$ as one over $\mathcal{U}$,

$$\int_{\mathcal{X}} f(x)\,\mathrm{d}x = \int_{\mathcal{U}} f(t(u)) \frac{\mathrm{d}x}{\mathrm{d}u}\,\mathrm{d}u, \tag{2.50}$$

where the integrand is scaled by the Jacobian

$$|J_t(u)| = \frac{\mathrm{d}x}{\mathrm{d}u} \tag{2.51}$$

of the mapping. Appendix A provides a primer on the topic.

In the context of Monte Carlo rendering, substitution is applied for two main purposes. First, the rendering integral can be written as one over directions, surfaces, or spherical coordinates, and each formulation motivates different algorithms [Kajiya 1986] (see Section 2.1.2). Second, importance sampling via sample transformation is achieved through substitution. Understanding the calculus foundations of these operations allows us to unify and combine them, which is essential for multiple importance sampling (discussed in Section 3.1.3).

2.4.2.1 Deriving sample transformations

How can sample transformations achieve a desired PDF? Assume we aim for a target PDF $p(x)$ and transform uniform samples, that is, $p(u) = 1$, from primary space via a mapping $x = t(u)$. How do we need to design this mapping such that the target PDF is attained?

This can be answered by observing the effect that a given $t(u)$ has on the PDF. A sample transformation scales the density by the reciprocal Jacobian,

$$p(x) = p(u)|J_t(u)|^{-1}. \tag{2.52}$$

Intuitively speaking, this relationship holds because the Jacobian measures how the mapping t locally affects the size of the domain. If t maps a large portion of $\mathcal{U}$ to a small subset of $\mathcal{X}$, then the density is high, as many primary points are compacted into this small region. Conversely, the density is low if a small portion of $\mathcal{U}$ maps to a large subset of $\mathcal{X}$.

More formally, we can derive this relationship through integration by substitution. For that, we assume that t is an injective mapping. Then, the probability that $x = t(u)$ lies in some subset $S \subset \mathcal{X}$ is, by construction, equal to the probability that the corresponding u lies in the inverse image $S^{-1} \subset \mathcal{U}$ of that subset,

$$P(x \in S) = P(u \in S^{-1}), \tag{2.53}$$

for any subset S of any size. By definition of the PDF, this implies that

$$\int_S p(x)\,\mathrm{d}x = \int_{S^{-1}} p(u)\,\mathrm{d}u. \tag{2.54}$$

The left hand integral can be written as one over primary space by substituting $x = t(u)$:

$$\int_{S^{-1}} p(t(u))|J_t(u)|\,\mathrm{d}u = \int_{S^{-1}} p(u)\,\mathrm{d}u. \tag{2.55}$$

Since this must hold for any subset of any size, the relationship must also hold point-wise, showing that

$$p(t(u))|J_t(u)| = p(u) \tag{2.56}$$

and proving the above relationship of the two PDFs. This equation is absolutely essential for (multiple) importance sampling in rendering.

With knowledge of this relationship, we also obtain a general recipe for finding such a transformation t. Specifically, Equation (2.52) defines a differential equation that we can solve for t. One solution is given by the inverse of the CDF, $t(u) = P^{-1}(u)$, since (1) the Jacobian of an inverse function is the reciprocal of the Jacobian of the original function,

$$|J_{t^{-1}}(x)| = |J_t(u)|^{-1} \tag{2.57}$$

and (2) the CDF is the antiderivative of the PDF,

$$P(u) = \int_0^u p(u')\mathrm{d}u'. \tag{2.58}$$

Unfortunately, it is not always easy – or even possible – to invert $P(u)$ [Heitz 2020].

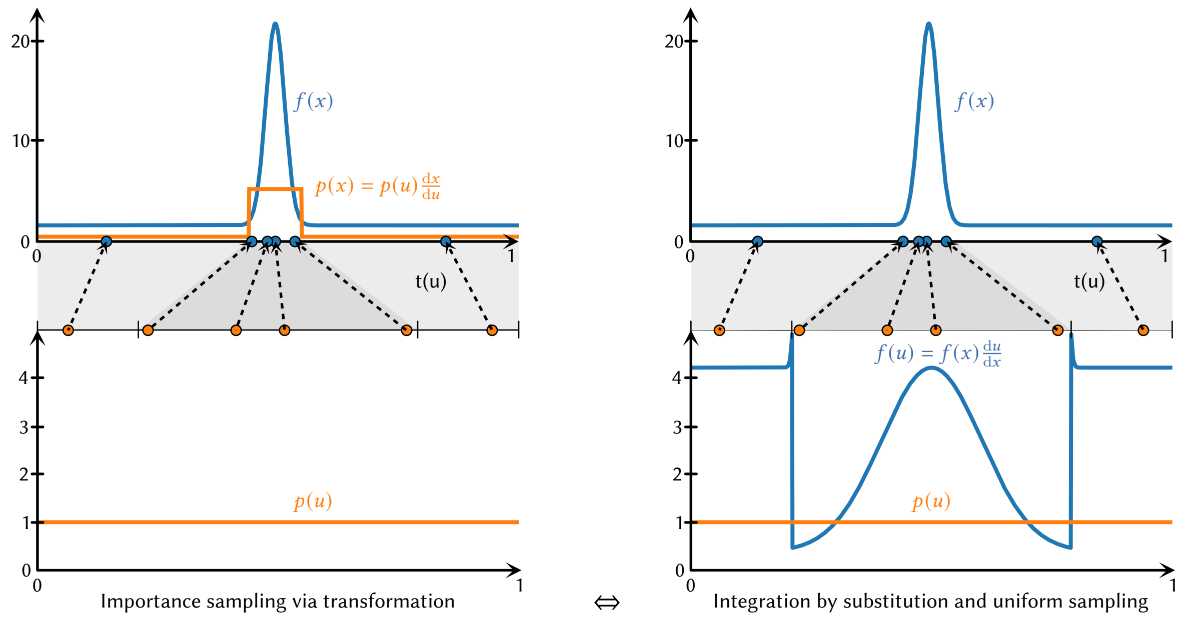


Figure 2.18: *Importance sampling is integration by substitution. In this 1D example, a narrow, vertically-shifted, and truncated Gaussian (depicted in blue) is integrated via Monte Carlo. For that, an approximate PDF (orange) is used that increases density around the peak. The plot in the top left shows this setup, and the plot below shows how uniform samples in primary space are mapped to the integration domain to achieve this importance sampling. The plots on the right instead show how integration by substitution maps the integration domain to primary space – through the same mapping – and integrates the distorted integrand there. While derived differently, both formulations yield the same estimator that performs the same operations to compute the same result.*

2.4.2.2 Importance sampling is integration by substitution

Importance sampling via sample transformation defines a change of variables $x = t(u)$ and thereby an equivalent integral over primary sample space,

$$\int_{\mathcal{X}} f(x)\,\mathrm{d}x = \int_{\mathcal{U}} f(t(u))|J_t(u)|\,\mathrm{d}u. \tag{2.59}$$

By construction, the Jacobian

$$|J_t(u)| = \frac{p(u)}{p(x)} = \frac{1}{p(x)} \tag{2.60}$$

is the reciprocal of the importance sampling PDF. So the integral can be written as

$$\int_{\mathcal{U}} f(t(u))|J_t(u)|\,\mathrm{d}u = \int_{\mathcal{U}} \frac{f(t(u))}{p(t(u))}\,\mathrm{d}u. \tag{2.61}$$

Therefore, importance sampling can also be interpreted as computing this primary space integral via uniform sampling.

Figure 2.18 shows this on a simple 1D example. The left plot shows the integrand and importance sampling PDF. A non-uniform density is achieved through a sample transformation from primary sample space. The right plot shows the equivalent interpretation where the integrand is instead mapped to primary space, using the same transformation, and estimation proceeds there with uniform sampling.

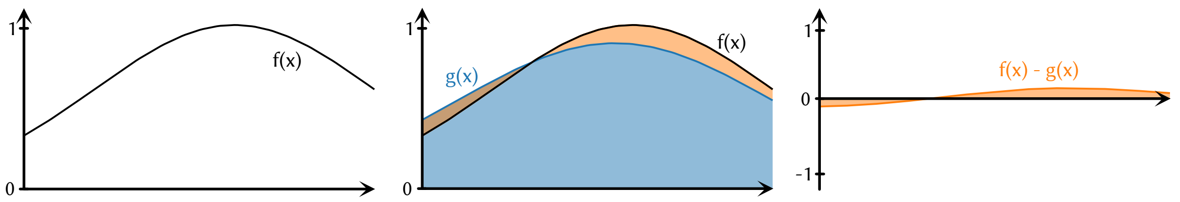


Figure 2.19: *The idea behind control variates, illustrated on a simple 1D integral. Instead of directly computing F, a function $g(x)$ with known integral G, depicted in blue, is used as a starting point. Then, the difference, shown in orange, is estimated and added to the known G to obtain F. Low variance is achieved if the difference is easier to integrate than the original function. This is the case here, where $f(x) - g(x)$, plotted on the right, is a much flatter function.*

Importance sampling and the equivalent change of variables achieve the same goal via the same math operations. Only the justification of these operations differs. On the one hand, importance sampling generates a random x by first sampling a uniform u and then computing the corresponding $x = t(u)$. The integral is then estimated as

$$\langle F \rangle = \frac{f(x)}{p(x)} = \frac{f(t(u))}{p(u)\frac{1}{|J_t(u)|}}. \tag{2.62}$$

On the other hand, the change of variables mindset also starts by sampling a random u, but then uses that u to estimate the primary space integral,

$$\langle F \rangle = \frac{f(t(u))|J_t(u)|}{p(u)}. \tag{2.63}$$

That is, both estimators are equivalent, and differ only in whether the Jacobian arises from the definition of the PDF or the definition of the integrand.

The converse of this observation is also true: every valid change of variables also defines an importance sampling estimator. This connection allows us to interpret algorithms operating with different integral formulations as different importance sampling schemes of the same formulation. That, in turn, enables combinations of these methods through, for example, multiple importance sampling (see Chapter 3).

2.4.3 Control variates

Control variates are another powerful variance reduction technique. They can be used on their own, or combined with importance sampling. The idea is fairly simple: A known integral G is subtracted from the desired F, and Monte Carlo integration is used to estimate the difference between the two.

Control variates have received less attention in rendering than importance sampling, but there are successful applications. A promising approach seems to be to learn a suitable control variate from initial samples and use it in conjunction with (learned or fixed) importance sampling [Crespo et al. 2021; Müller et al. 2020; Salaün et al. 2022; Vévoda et al. 2018]. Control variates can also further reduce the variance of multiple importance sampling estimators [Kondapaneni et al. 2019; Owen and Zhou 2000], as we will discuss in Section 3.3.2.

2.4.3.1 Definition

A control variate can be formed if we know a function $g(x) \approx f(x)$ that approximates the integrand, and if we can easily compute the integral of this approximation,

$$G = \int_X g(x)\,\mathrm{d}x. \tag{2.64}$$

Then, we can write the integral F as the sum of the known G and the difference integral,

$$\begin{aligned} F &= G + F - G && (2.65)\\ &= G + \int_X f(x)\,\mathrm{d}x - \int_X g(x)\,\mathrm{d}x && (2.66)\\ &= G + \int_X \left(f(x) - g(x)\right)\,\mathrm{d}x. && (2.67) \end{aligned}$$

Figure 2.19 illustrates this idea. The area under the curve of $f(x)$ is computed by computing the area between $g(x)$ and $f(x)$, marked in orange, and adding it to the known area under the curve of $g(x)$, marked in blue. If the area between the curves, that is, the difference integral, is easier to compute than the original integral, then the resulting estimator will be more efficient.

The difference integral can be computed via Monte Carlo integration, leading to the control variate estimator:

$$\langle F\rangle_{\mathrm{CV}} = G + \sum_{i=1}^{n} \frac{f(x) - g(x)}{p(x)}. \tag{2.68}$$

This estimator can be improved via importance sampling, by finding a PDF $p(x)$ that approximates the difference $f(x) - g(x)$. Therefore, control variates can, at the same time, act as a replacement for and an orthogonal improvement to importance sampling.

2.4.3.2 Comparison to importance sampling

An optimal control variate estimator is achieved if $g(x) = f(x)$ exactly matches the integral[1]. This is conceptually similar to optimal importance sampling. As is the case there, the optimum has a purely theoretical value, since it requires knowledge of the solution to the original integration problem.

Using $g(x)$ as a control variate, rather than for importance sampling, offers two key benefits: First, importance sampling $p(x) = g(x)G^{-1}$ requires a sample transformation from primary space. This transformation is the solution of a differential equation that may be non-trivial, prohibitively expensive, or simply impossible to find. Second, $g(x)$ can be real-valued, that is, it is not constrained to be positive.

While most integrands in rendering are non-negative, there are cases where a signed function needs to be integrated. For example in differentiable rendering [Zeltner et al. 2021] or for the simulation of wave-optical effects like interferences. A simple example is shown in

[1]There are infinitely many optimal control variates. Namely, if $g(x)$ is optimal, so is every shifted $g'(x) = g(x) + Cp(x)$ for any constant C [Hua et al. 2023, Section 3.2]. Here, we consider the most intuitive choice of $g(x) = f(x)$.

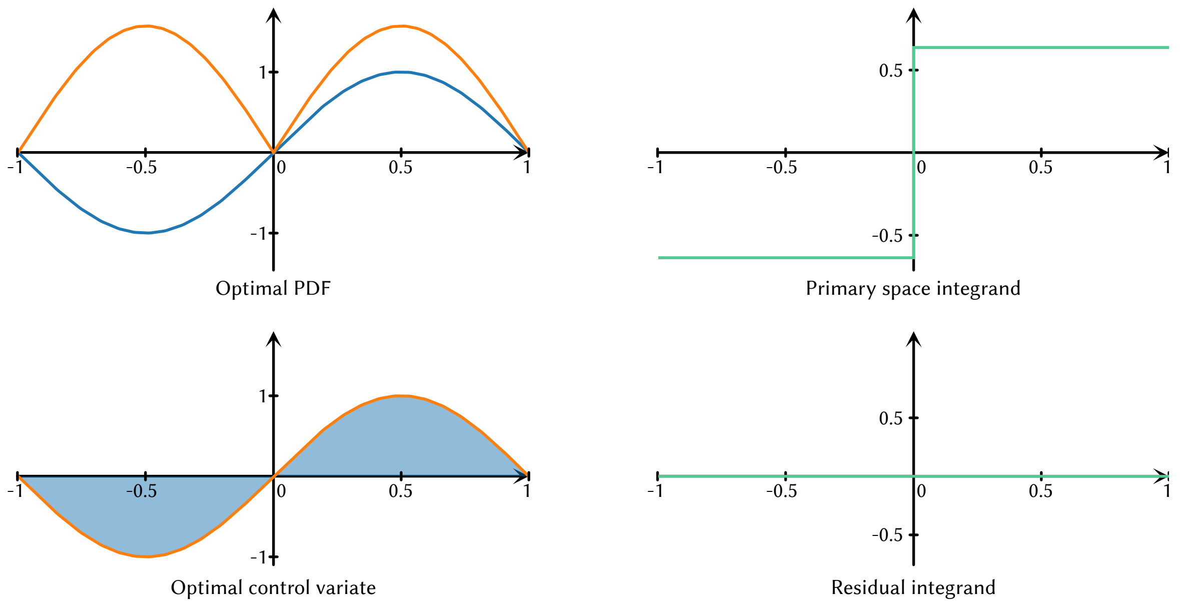


Figure 2.20: *Comparison of optimal control variates and optimal importance sampling on a signed integrand. The blue line and area on the left plot the integrand and integral, respectively. The orange line at the top is the optimal PDF, the orange line at the bottom is an optimal control variate. The green lines on the right plot the corresponding effective integrand that is estimated with uniform samples. Importance sampling will always retain the sign-related variance – due to the discontinuity in the primary space integrand – while control variates can achieve zero variance – due to the constant residual integrand.*

Figure 2.20. There, the integrand is a sine function, $f(x) = \sin(2\pi x)$, with a trivial analytical integral $F = 0$. Importance sampling cannot achieve a zero-variance estimator for this integral (without additional tricks [Owen and Zhou 2000]) but control variates can.

The optimal importance sampling PDF is proportional to the absolute value of the integrand (2.48),

$$p(x) = \frac{|\sin(2\pi x)|}{2\int_0^{0.5} \sin(2\pi x)\,\mathrm{d}x}. \tag{2.69}$$

Unlike with positive integrands, however, the variance with such optimal sampling is not zero. This can be easily seen by observing that the value of a single sample is not constant,

$$\frac{\sin(2\pi x)}{p(x)} = \operatorname{sgn}\left(\sin(2\pi x)\right) 2\int_0^{0.5} \sin(2\pi x)\,\mathrm{d}x, \tag{2.70}$$

as it depends on the sign of the integrand at the sampled position.

In comparison, the optimal control variate, $g(x) = \sin(2\pi x)$, in combination with uniform sampling, does produce a zero-variance estimator, since $G = F$. The key to success is, that control variates can be arbitrary (integrable) functions and are not constrained to be positive.

There are, however, also techniques to reduce this sign-related variance in importance sampling. *Positivization* [Owen and Zhou 2000] separates the integral into two parts, the positive and the negative regions, and computes those independently. However, doing so requires knowledge of the roots of the integrand, a sampling density that is defined only over each region, and at least two samples. Especially the latter constraint, requiring more than one sample, is problematic when applied to high-dimensional problems, where it would cause exponential growth. Antithetic sampling (see Section 2.3.4.1) can be used as a special case of positivization, additionally introducing negative correlation to reduce the variance further.

2.4.4 Adaptive sampling

The efficiency of a Monte Carlo estimator can be greatly increased by automatically adapting it to the problem at hand. The design space for such adaptation is huge, and is still actively explored by a vast body of research. Existing methods can be loosely grouped into three categories: adaptive sample counts, adaptive PDFs or control variates, and sample mutation and resampling.

The methods presented in Chapters 5 and 7 also fall into this category of adaptive methods: The former adapts the MIS weighting function based on the variance, the latter adapts the sample allocation for combined algorithms.

2.4.4.1 Adaptive sample counts

Rendering applications compute an integral in every pixel. The corresponding estimation errors can differ greatly between pixels. For example, some pixels may feature a complex reflected caustic, while others show only simple direct illumination. If the same number of samples is used in all pixels, the result would be either residual noise in difficult pixels or a waste of computation time on simpler ones. Therefore, a very common type of adaptive sampling in rendering aims to allocate samples to each pixel such that a uniform error distribution is achieved [Hasselgren et al. 2020; Kuznetsov et al. 2018; Zwicker et al. 2015]. Similarly, path splitting factors can be optimized on-the-fly [Bolin and Meyer 1997; Rath et al. 2022; Vorba and Křivánek 2016]. The method presented in Chapter 7 applies this idea to determine the number of samples to allocate to different techniques.

2.4.4.2 Adaptive importance sampling or control variates

Importance sampling and control variates both use approximations of the integrand to reduce variance. Such an approximation can be generated or adapted on-the-fly during the integration process. To that end, an initial batch of samples can be used to construct an approximation of the integrand. This approximation can then be used as an importance sampling density or a control variate for subsequent samples. A popular example for adaptive PDFs in rendering are path guiding methods [Bashford-Rogers et al. 2012; Grittmann et al. 2018; Herholz et al. 2016, 2019; Hey and Purgathofer 2002; Jensen 1995; Müller et al. 2017, 2019; Rath et al. 2020; Reibold et al. 2018; Ruppert et al. 2020; Schüßler et al. 2022; Vorba et al. 2014] or learned discrete distributions for light source selection [Donikian et al. 2006; Vévoda et al. 2018; Wang et al. 2021]. Similarly, selection of virtual point lights or bidirectional connections can be improved by constructing an adaptive distribution [Georgiev et al. 2012b; Popov et al. 2015; Su et al. 2022]. Adaptive construction of control variates is also possible [Crespo et al. 2021; Fan et al. 2006; Müller et al. 2020; Salaün et al. 2022; Vévoda et al. 2018].

Adaptive importance sampling can be a powerful tool for rendering difficult scenes. Therefore, it has found many uses in the industry [Vorba et al. 2019]. Despite these successes, there is still much more to be done. For example, finding the best representations, data structures, or subdivision rules still warrants much exploration.

2.4.4.3 Markov chains and resampling

Explicitly adapting or constructing a PDF or control variate from sample data can incur a hefty memory cost and computation overhead. This is a main reason why path guiding has been proposed decades ago [Hey and Purgathofer 2002; Jensen 1995] but only became popular relatively recently [Vorba et al. 2019]. Markov chain methods and resampling offer more memory-friendly alternatives.

Markov chain Monte Carlo (MCMC) [Šik and Křivánek 2018; Veach and Guibas 1997] methods mutate samples and randomly accept or reject these mutations. The acceptance probability is designed such that the expected sample density, after infinitly many mutations, is the desired PDF. Markov chain methods often exhibit strong correlation artifacts, making them less popular in rendering practice. Still, successful applications exist, especially for bidirectional methods [Šik and Křivánek 2019; Šik et al. 2016].

Resampling methods [Talbot et al. 2005] are based on the same idea but apply it differently. An initial set of samples is generated, and a subset of these samples is selected. That is, they also utilize acceptance and rejection, but make do without mutations. Rendering applications typically employ resampling in combination with path reuse, such that the cost of the additional samples is amortized [Bitterli et al. 2020; Georgiev et al. 2012b; Su et al. 2022]. The sampling quality of such resampling approaches is generally lower than explicitly modeled densities[2]. However, the much lower overhead can be attractive for low-budget applications such as real-time rendering [Bitterli et al. 2020].

2.5 Rendering algorithms

Modern rendering algorithms use Monte Carlo integration to compute the rendering equation. That is, images are rendered by sampling random paths that connect the camera to a light source.

The prevalent algorithm, forward path tracing, extends paths from the camera to the lights. For scenes with difficult transport, this does not always perform well. There, bidirectional methods can help. These additionally trace paths from the lights and combine them with the camera paths.

[2]The variance of resampling 1 out of M samples is lower-bounded by the variance of an ordinary M-sample estimator. In contrast, the variance of a PDF-learning-based method has no such bound – in theory, a single training sample could recover the zero-variance density.

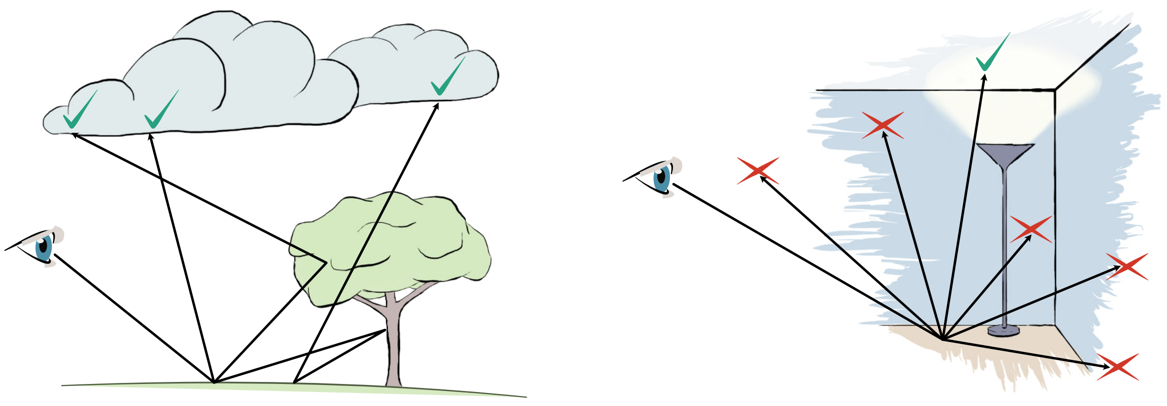


Figure 2.21: *Some scenes are easy to render with unidirectional path tracing, others almost impossible. An outdoor environment on a cloudy day (left) works well, while focused indirect illumination (right) remains difficult. Here, green ticks mark the paths that have found the light, and red crosses the ones that did not.*

2.5.1 Forward path tracing

The default method of most production renderers is unidirectional forward path tracing [Fascione et al. 2018]. Paths are formed by first tracing a ray from the camera into the scene. The path is then continued by repeatedly sampling directions from the intersected surfaces. Path construction is terminated if the path is fully absorbed, leaves the scene, is terminated randomly with Russian roulette [J. R. Arvo and Kirk 1990; Rath et al. 2022; Vorba and Křivánek 2016], or reaches a fixed maximum depth. Typically, the vertices along the path are also directly connected to points on the light sources ("next event estimation"), to boost the probability of forming a path with non-zero contribution.

The appeal of the unidirectional method is twofold. First, by starting paths from the camera, the exact number of samples per pixel can be controlled easily. This provides noise reduction through stratification and enables great flexibility for adaptive sampling. Second, the process is trivial to parallelize and only a small constant amount of memory is required for sample storage. In other words, forward path tracing is easy to adapt and trivial to implement efficiently.

Unidirectional path construction works best if the illumination in the scene is diffuse. That, of course, is often not the case. Consequently, many scenes will require a vast number of samples to converge to an acceptable error level. Figure 2.21 sketches two example cases, one where forward path tracing works well, and one where it does not. In a cloudy outdoor setting, shown on the left, most, if not all, paths yield a non-zero contribution, because similar levels of light are coming from all directions. But results are less rosy in an interior setting with focused indirect illumination, as depicted on the right. There, forward path tracing must randomly sample a direction that finds the small, brightly illuminated spot around the lamp. This is unlikely to happen, so most paths do not connect to the light.

Forward path tracing can be enhanced by sophisticated adaptive sampling schemes to handle such more difficult cases; for example, through path guiding [Vorba et al. 2019]. Unfortunately, these methods sacrifice some of the benefits of forward path tracing: They introduce significant overhead, require synchronisation, and greatly increase the implementation difficulty. Also, such solutions can be wasteful in simpler scenes where they are

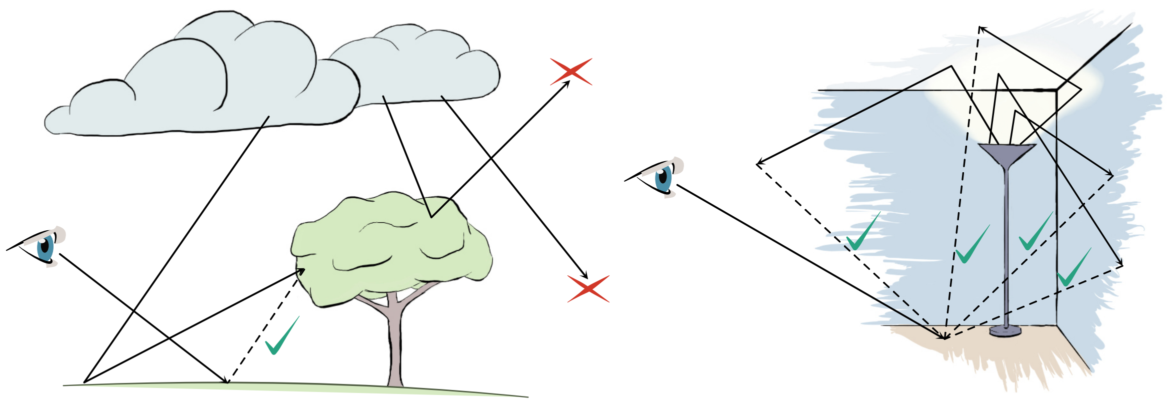


Figure 2.22: *Bidirectional sampling techniques excel at focused indirect illumination in small scenes. The sketch on the right illustrates one such example. There, connecting a short camera path to a light path is almost guaranteed to yield a valid sample connecting the camera to the (difficult to find unidirectionally) light source. On the flipside, bidirectional sampling fares poorly in large scenes or for uniform illumination, as sketched on the left. There, most light paths will never find the visible region.*

not required. Furthermore, learning-based methods such as guiding frequently encounter a chicken-and-egg problem: We cannot learn how to sample difficult paths without having found at least one similar path through pure forward path tracing. Adaptation can be slow if these initial samples are unlikely to be found. Therefore, guided forward path tracing still struggles with effects such as complex caustics.

2.5.2 Bidirectional path tracing

Bidirectional sampling offers a way to efficiently find some paths that are extremely hard for forward path tracing. In addition to tracing paths from the camera, these methods also trace paths from the light sources. The two sets of paths are then combined to form full paths.

There are many ways how that combination can be done. For example, the classic bidirectional path tracing algorithm [Veach and Guibas 1995a] traces pairs of camera and light paths and connects each vertex of one to every vertex of the other. Another option is to trace a bunch of light paths and connect every camera path vertex to all of them [Keller 1997]. These shared light paths vertices are commonly called virtual point lights (VPLs). Instead of connecting each camera vertex to all VPLs, resampling can be used to improve efficiency [Georgiev et al. 2012b; Popov et al. 2015; Su et al. 2022].

Bidirectional sampling nicely complements forward path tracing. This is illustrated on Figure 2.22, using the same representative examples as before. The figure sketches one bidirectional technique: tracing a camera prefix of just one edge, and connecting it to many light paths. In large scenes, such as the cloudy exterior, bidirectional sampling is not great, because most light paths will never find the visible region. However, interior scenes with focused indirect illumination are handled exceptionally well. Because the illumination is focused, the light subpaths easily find the visible region.

The poor performance of bidirectional methods in large scenes can be remedied via adap-

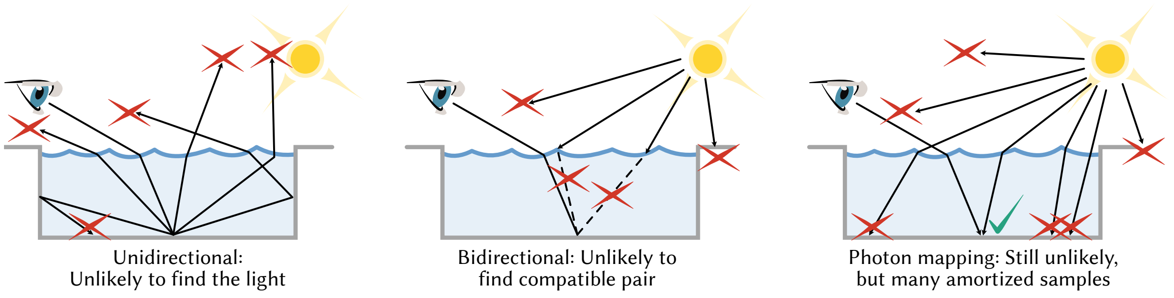

Unidirectional:
Unlikely to find the light
Bidirectional: Unlikely to
find compatible pair
Photon mapping: Still unlikely,
but many amortized samples

Figure 2.23: *Photon mapping can efficienctly render caustics, because it effectively uses millions of samples in each pixel, while ammortizing the cost over all pixels.*

tive methods [Grittmann et al. 2018; Šik et al. 2016]. However, these incur additional overhead and are thus unlikely to perform on-par with simple forward path tracing in diffusely lit scenes such as a cloudy exterior.

2.5.3 Photon mapping

Between unidirectional and bidirectional path tracing, many lighting effects can be captured efficiently. There is, however, one important exception: reflected caustics. Caustics are the focused patterns of light due to interaction with dielectrics such as glass or water. In the real world, they are present almost everywhere. When viewed directly, bidirectional methods can capture such caustics efficiently, but reflections of caustics remain difficult. Unfortunately, whenever there is a caustic in the real world, there usually is also a reflection or refraction of that caustic visible somewhere.

A popular example are the caustics at the bottom of a pool when viewed from above the water surface. Figure 2.23 sketches such a setup. Neither bidirectional nor unidirectional methods can easily find a path that connects the camera to the light through both interactions with the water surface. From both directions, it is equally unlikely to sample a direction that intersects the water surface at precisely the right angle to find the light or the camera. This is sketched in the first two pictures of Figure 2.23, where neither forward nor bidirectional sampling produced a single non-zero path.

Again, one remedy for this problem is adaptation. For instance, path guiding methods can gradually learn to sample such caustics reasonably well. However, because caustics can be extremely difficult to find in the first place, even state-of-the-art path guiding approaches can only render a few simple caustic effects efficiently.

The best method known to date to handle such reflected caustics is photon mapping [Jensen 1996]. Photon mapping is also a bidirectional technique. It starts by tracing a large number of light subpaths through the scene and recording their vertices in a so-called photon map. Then, the camera subpaths are traced, and each camera path vertex is *merged* with the nearby vertices in the photon map. This is achieved by running a nearest neighbor search and then pretending that the camera and light vertices where exactly identical.

Of course, it is statistically impossible that any nearby pair of camera and light vertices is actually identical. Therefore, photon mapping is a biased algorithm; it does not converge to the correct image. This bias manifests as blurring, splotchy artifacts, or light leaks, which can be very unappealing and difficult to remove in a post-process. Luckily, it is possible to reduce this bias by reducing the radius of the nearest neighbor search. Gradually reducing the radius over time provides a consistent estimator that converges to the reference [Hachisuka et al. 2008], and locally adapting the radius can achieve better trade-offs between noise and bias [Kaplanyan and Dachsbacher 2013a; Lin et al. 2020].

Because photon mapping is still just a bidirectional sampling technique at heart, it does not actually increase the likelihood of finding a difficult reflected caustic path. This is sketched on the right of Figure 2.23. Instead, the advantage of photon mapping stems from path reuse with ammortized cost. Millions of light paths are traced throughout the scene, and each individual camera path is combined with *all* of those through the nearest neighbor search. But the same set of light paths is used for all pixels in the image, so the cost is ammortized. Photon mapping effectively uses millions of bidirectional samples in each pixel, but only pays for a single one.

Photon mapping is a particularly expensive technique, due to the nearest neighbor search that is run at every hit point. Because of that and the bias artifacts from the merging operation, it is best used sparingly only where absolutely necessary [Grittmann et al. 2018; Šik and Křivánek 2019]. Path guiding can sometimes replace photon mapping for very simple caustics, and there are also elaborate sampling schemes based on manifold walks [Hanika et al. 2015; Jakob and Marschner 2012; Zeltner et al. 2020]. The latter strike a different trade-off than photon mapping, as they can be even more expensive and less general, but do not (necessarily) introduce bias and can handle caustics on highly-glossy surfaces better [Zeltner et al. 2020].

2.6 Summary

Monte Carlo rendering algorithms compute images of virtual scenes by sampling random paths between the camera and the light sources. This path sampling can be done unidirectionally or bidirectionally. Further, many techniques are available to increase efficiency, such as (adaptive) importance sampling. Therefore, the design space for possible algorithms is vast, and the best algorithm has yet to be found.

For every lighting effect, there is a path sampling technique that captures it best. For instance, if we are rendering a field on a cloudy day, forward path tracing is our best bet. For an interior illuminated by artificial light sources, bidirectional path tracing is an excellent choice. For caustics, there is often no way around photon mapping. Therefore, efficient rendering in practice requires us to identify the optimal subset (and variations of) the best techniques from this vast amount of candidates. And we have not even touched upon the rendering of volumetric effects [Křivánek et al. 2014], which further add to this pool of techniques.

A general, albeit inefficient, algorithm can be achieved by simply combining as many of these methods as possible [Georgiev et al. 2012a; Hachisuka et al. 2012; Křivánek et al. 2014; Veach 1997]. Thereby, any lighting effect can be rendered in acceptable time without manual parameter tuning. Such a combination can be achieved through multiple importance sampling (MIS) [Veach and Guibas 1995b] (see Chapter 3), an essential ingredient in any rendering algorithm. However, such an exhaustive combination will not be very efficient: In most scenes, a lot of the techniques are either underperforming or redundant, and the computation time invested in them is wasted.

In the following chapters, we first introduce MIS, show how it is used to combine forward path tracing, bidirectional sampling, and photon mapping, identify the challenges and problems in that combination, and propose practical solutions.

Chapter 3

Multiple Importance Sampling (MIS)

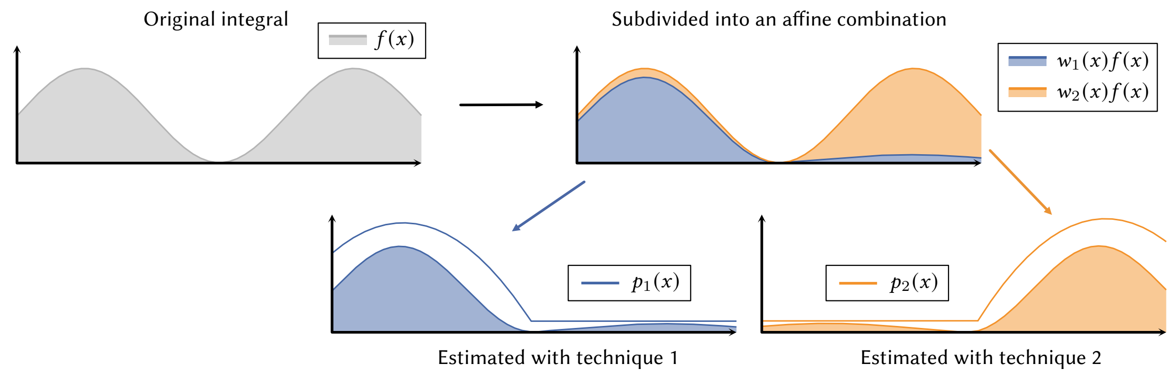


Figure 3.1: *MIS is a divide and conquer approach. An integral with difficult to sample shape (shown in gray in the top left) is divided into a sum of integrals (plotted as stacked areas in the top right). The gray area (i.e., integral) is equal to the sum of the blue and orange areas (i.e., integrals). This subdivision is achieved by multiplying the integrand $f(x)$ by a weighting function $w_i(x)$, one per technique. The component integrals are then estimated via Monte Carlo integration (bottom) using a different, sampling technique for each. Here, the PDFs are plotted as a blue and orange line, respectively.*

The integrals we compute have a complicated shape influenced by many factors. Specifically, the light transport equation integrates over a high-dimensional product of emission profiles and scattering distributions with discontinuities due to object boundaries. Finding a single sampling technique that can form an efficient Monte Carlo estimator for such an integral is definitely difficult and possibly impossible.

In computer science, such difficult problems are often efficiently solved by divide and conquer methods. Instead of attempting to solve the entire problem at once, it is divided into multiple sub-problems. When done well, the sub-problems are easier to solve and their solutions can be efficiently combined into a solution for the whole problem.

Multiple importance sampling (MIS) [Veach and Guibas 1995b] is such a divide and conquer method, applied to Monte Carlo integration. The concept is illustrated on a 1D integral in Figure 3.1. Instead of trying to find a good importance sampling PDF that works well for the entire integral, MIS first divides the integral into a sum of integrals. Then, a Monte Carlo estimator with a specialized sampling density for each part is used to estimate each sub-integral. The sum of those sub-estimators is an unbiased estimate of the full integral.

The remainder of this chapter reviews the mathematical formulation of the MIS estimator, provides a variance analysis, and reviews weighting functions and other enhancements proposed in previous work. We conclude by summarizing the open research questions.

3.1 The MIS estimator

MIS was introduced decades ago [Veach and Guibas 1995b] and has been used widely ever since. In a nutshell, it combines a set of sampling techniques $t \in \mathcal{T}$ by taking n_t samples with probability density $p_t(x)$ from each technique t. An unbiased estimator is achieved by summing over all these samples and multiplying additionally by an MIS weighting function $w_t(x)$:

$$\langle F \rangle_{\text{MIS}} = \sum_{t \in \mathcal{T}} \frac{1}{n_t} \sum_{i=1}^{n_t} \frac{w_t(x_{i,t}) f(x_{i,t})}{p_t(x_{i,t})}. \tag{3.1}$$

Here, $x_{i,t}$ is the ith sample generated by technique t. Unbiasedness is achieved as long as the weights sum to one,

$$\forall x : \sum_{t \in \mathcal{T}} w_t(x) = 1, \tag{3.2}$$

and are zero whenever a technique does not sample a point,

$$\forall x : p_t(x) = 0 \implies w_t(x) = 0. \tag{3.3}$$

The remainder of this section provides a more detailed explanation of how and why this estimator works. Section 3.2 discusses how choosing the right weights, techniques, and sample counts can improve efficiency.

3.1.1 Affine combination

MIS writes the integrand $f(x)$ as an affine combination of weighted integrands,

$$f(x) = \sum_{t \in \mathcal{T}} w_t(x) f(x), \tag{3.4}$$

thus replacing the original integral F by a sum of weighted integrals,

$$F = \int_X f(x)\, \mathrm{d}x = \sum_{t \in \mathcal{T}} \int_X w_t(x) f(x)\, \mathrm{d}x. \tag{3.5}$$

The weighting functions $w_t(x)$ can be chosen arbitrarily. The only condition is that they must sum to one so they form an affine combination. Most weighting schemes, like the popular balance heuristic, use positive weights, hence forming a convex combination. But positivity is not required, and allowing negative weights can yield further variance reduction [Kondapaneni et al. 2019].

Figure 3.1 shows how this can be interpreted visually. In 1D, the integral is the area under the curve of $f(x)$ (gray line / area). This area can be divided into multiple parts via an affine combination, as shown in the top right. MIS computes each sub-area individually and then sums them up to obtain the full area.

MIS estimates each weighted integral via ordinary Monte Carlo integration, using n_t samples from only the corresponding technique t:

$$\int_X w_t(x) f(x)\, \mathrm{d}x \approx \frac{1}{n_t} \sum_{i=1}^{n_t} \frac{w_t(x_{i,t}) f(x_{i,t})}{p_t(x_{i,t})}. \tag{3.6}$$

Algorithm 2: *Pseudocode for a simple MIS estimator. Given a generic Monte Carlo estimator implementation (Algorithm 1), MIS can be implemented by simply invoking that estimator for each technique to compute the weighted integrals.*

```
function MIS($f$, $\mathcal{T}$)                          ← Given an integrand and a set of techniques
  estimate = 0
  for $t \in \mathcal{T}$ do
    estimate += MonteCarlo($w_t f$, $p_t$, $n_t$)   ← Estimate the weighted integral of this technique
  return estimate
```

The sum of these individual estimators then yields the full MIS estimator (3.1).

The key to success is that the sampling densities $p_t(x)$ are better importance sampling distributions for their weighted integrand $w_t(x)f(x)$ than they are for the full $f(x)$. This is the case in the example depicted in Figure 3.1. Neither the orange nor the blue density perform well on the full integration problem. On their respective sub-problems, however, they each perform almost perfectly.

An idealized example of how an MIS estimator can be implemented is shown in Algorithm 2. Given a generic Monte Carlo estimator (see Algorithm 1), MIS can be implemented by repeatedly invoking this estimator, once for each technique. Unfortunately, implementations of MIS in rendering applications are rarely this simple. Often, the computations for different sampling techniques are heavily intertwined with each other and scattered throughout a large code base.

3.1.2 Designing an MIS estimator

How can we design an effective MIS estimator? Sadly, there is no definite answer to that question. The performance of an MIS estimator is controlled by three parameters: the set of sampling techniques t with their PDFs $p_t(x)$, the weighting functions $w_t(x)$, and the sample counts n_t. A perfect estimator is one that uses an optimal combination of all three.

A common approach is to decide the PDFs and the sample counts in advance. Then, the weights $w_t(x)$ are chosen conditionally on those, such that they produce low variance estimates. Intuitively speaking, this allows us to shape the integrals computed with each technique such that its (given and fixed) PDF is a good match to the weighted integral.

For that, a provably good, albeit not optimal, choice for the weighting function is the balance heuristic [Veach and Guibas 1995b],

$$w_t^{\text{bal}}(x) = \frac{n_t p_t(x)}{\sum_{t'} n_{t'} p_{t'}(x)}, \tag{3.7}$$

where the weights are set to be proportional to the *effective sampling densities* $n_t p_t(x)$ of each technique. Intuitively, the balance heuristic bounds the variance by ensuring that low-density PDFs are not used to sample high-value regions:

$$p_t(x) \approx 0 \Rightarrow w_t^{\text{bal}}(x) f(x) \approx 0. \tag{3.8}$$

Provided, of course, there is a better technique t' that samples this point x with higher density. The next section discusses these variance guarantees – and their limitations – in more detail.

For computation, it is often convenient to write the balance heuristic as the reciprocal of the sum of PDF ratios,

$$w_t^{\mathrm{bal}}(x) = \left(1 + \sum_{t' \neq t} \frac{n_{t'} p_{t'}(x)}{n_t p_t(x)}\right)^{-1}, \tag{3.9}$$

by dividing the numerator and denominator in (3.7) by $n_t p_t(x)$. This benefits numerical stability and affords efficient computation if the number of techniques is large.

3.1.3 Integral formulations and MIS

Not all sampling techniques in rendering are based on the same integral formulation. For example, consider the simple application of direct illumination computation. Most renderers estimate direct illumination through an MIS combination of BSDF sampling and light source sampling. The former is defined via the direction integral; the latter via the surface integral (see Section 2.1.3). But for MIS to work, all techniques must be defined with respect to the same integral formulation.

In the direct illumination example, we can pick either of the two formulations – directions or surfaces – as the common ground for MIS. The choice is actually irrelevant, since the resulting computations will be the same either way. For instance, BSDF importance sampling can be reformulated based on the surface integral by mapping the directions ω_i to surface points (via ray tracing). This mapping results in a surface PDF

$$p_\rho(y) = p_\rho(\omega_\mathrm{i}) \frac{\mathrm{d}\omega_\mathrm{i}}{\mathrm{d}y} = p_\rho(\omega_\mathrm{i}) \frac{\cos\theta(y \to x)}{\|x - y\|^2} \tag{3.10}$$

that is scaled by the Jacobian of the mapping from surface points to directions, as derived in Appendix B.

This reformulation has no effect on the actual estimator of the technique. BSDF importance sampling, in its original formulation, estimates the direction integral as

$$\langle L_\mathrm{r} \rangle = \frac{\rho L_\mathrm{e} \cos\theta_\mathrm{i}}{p_\rho(\omega_\mathrm{i})}. \tag{3.11}$$

After mapping the sampled directions to surface points – via the ray tracing operator $y(\omega_\mathrm{i})$ – the estimator,

$$\frac{\rho L_\mathrm{e} \cos\theta_\mathrm{i} \frac{\cos\theta(y \to x)}{\|x-y\|^2}}{p_\rho(\omega_\mathrm{i}) \frac{\cos\theta(y \to x)}{\|x-y\|^2}} = \frac{\rho L_\mathrm{e} \cos\theta_\mathrm{i}}{p_\rho(\omega_\mathrm{i})}, \tag{3.12}$$

is still the same, because the Jacobian in the integrand cancels out with the one in the PDF.

If the MIS weights are defined in the surface domain, the BSDF sampling technique estimates the weighted surface integral

$$\int_{\mathcal{V}(x)} w_\rho(y) \rho L_\mathrm{e} \cos\theta_\mathrm{i} \frac{\cos\theta(y \to x)}{\|x - y\|^2} \, \mathrm{d}y = \mathrm{E}\left[w_\rho(y) \frac{\rho L_\mathrm{e} \cos\theta_\mathrm{i}}{p_\rho(\omega_\mathrm{i})} \right]. \tag{3.13}$$

Since the Jacobian cancels out with the one in the PDF, the resulting MIS weighted estimator is simply the original estimator, as derived from the direction integral fromulation, but multiplied by the *surface-domain* MIS weights. For instance, if the balance heuristic is used, the weight for BSDF sampling in our example is

$$w_\rho(y) = \left(1 + \frac{p_{L_\mathrm{e}}(y)}{p_\rho(y)}\right)^{-1} = \left(1 + \frac{p_{L_\mathrm{e}}(y)}{p_\rho(\omega_\mathrm{i})\frac{\cos\theta(y\to x)}{\|x-y\|^2}}\right)^{-1}. \tag{3.14}$$

So, as long as the MIS weights are computed based on PDFs defined in the same domain, the combination is valid. And these PDFs can be computed by simply multiplying the original per-technique PDF with the corresponding Jacobian.

3.2 Efficiency

Section 2.3 defined the efficiency of a simple Monte Carlo estimator as the inverse of the product of variance and cost. As MIS is merely a sum of multiple Monte Carlo estimators, the same definition can be applied,

$$\epsilon\left[\langle F\rangle_\mathrm{MIS}\right] = \left(\mathrm{V}\left[\langle F\rangle_\mathrm{MIS}\right]\mathrm{C}\left[\langle F\rangle_\mathrm{MIS}\right]\right)^{-1}. \tag{3.15}$$

The cost of an MIS estimator is simply the sum of the costs of the individual techniques,

$$\mathrm{C}\left[\langle F\rangle_\mathrm{MIS}\right] = \sum_{t\in\mathcal{T}} \mathrm{C}\left[\langle F\rangle_t\right], \tag{3.16}$$

possibly increased by the cost of the MIS weight computations, if that cost is significant. This cost is therefore highly application-specific. The variance, however, can be reasoned about in an application-agnostic way. In the following, we first discuss the variance of an MIS estimator and then review existing methods that improve efficiency by optimizing the weights, sample counts, or PDFs.

3.2.1 Variance

The variance of an MIS estimator can be derived from the variances of the individual technique estimators. If the techniques are statistically independent from each other, the full variance is the sum of the technique variances:

$$\begin{aligned}
\mathrm{V}[\langle F\rangle_\mathrm{MIS}] &= V\left[\sum_{t\in\mathcal{T}}\sum_{i=1}^{n_t}\frac{w_t(x_{i,t})f(x_{i,t})}{n_t p_t(x_{i,t})}\right] && \text{by definition} && (3.17)\\
&= \sum_{t\in\mathcal{T}} V\left[\sum_{i=1}^{n_t}\frac{w_t(x_{i,t})f(x_{i,t})}{n_t p_t(x_{i,t})}\right] && \text{with independent techniques} && (3.18)\\
&= \sum_{t\in\mathcal{T}}\frac{1}{n_t}V\left[\frac{w_t(x)f(x)}{p_t(x)}\right] && \text{with independent samples.} && (3.19)
\end{aligned}$$

If the samples are correlated within techniques or between techniques, an additional covariance term arises (see Section 2.3.2),

$$\mathrm{V}[\langle F\rangle_\mathrm{MIS}] = \sum_{t\in\mathcal{T}}\frac{1}{n_t}V\left[\frac{w_t(x)f(x)}{p_t(x)}\right] + \mathrm{Cov}. \tag{3.20}$$

As with a basic Monte Carlo estimator, the expanded form of the variance comprises three terms: the second moment, the sum of squared means, and the covariance,

$$\mathrm{V}[\langle F\rangle_{\mathrm{MIS}}] = \sum_{t\in\mathcal{T}} \frac{1}{n_t} \int_{\mathcal{X}} \frac{w_t^2(x) f^2(x)}{p_t(x)} \,\mathrm{d}x - \sum_{t\in\mathcal{T}} \frac{1}{n_t} \left(\int_{\mathcal{X}} w_t(x) f(x) \,\mathrm{d}x \right)^2 + \mathrm{Cov}. \tag{3.21}$$

Also as with single-technique estimators, the term "second moment" is commonly used to refer to the first term in this equation; that is, the sum of the second moments of the single-sample per-technique estimators, divided by their respective sample counts,

$$\mathrm{M}[\langle F\rangle_{\mathrm{MIS}}] := \sum_{t\in\mathcal{T}} \frac{1}{n_t} \int_{\mathcal{X}} \frac{w_t^2(x) f^2(x)}{p_t(x)} \,\mathrm{d}x. \tag{3.22}$$

This second moment can be used as an approximation of the variance,

$$\mathrm{V}[\langle F\rangle_{\mathrm{MIS}}] \approx \mathrm{M}[\langle F\rangle_{\mathrm{MIS}}], \tag{3.23}$$

as discussed in Section 2.3.3. The approximation error,

$$\mathrm{V}[\langle F\rangle_{\mathrm{MIS}}] - \mathrm{M}[\langle F\rangle_{\mathrm{MIS}}] = -\sum_{t\in\mathcal{T}} \frac{1}{n_t} \left(\int_{\mathcal{X}} w_t(x) f(x) \,\mathrm{d}x \right)^2 + \mathrm{Cov}, \tag{3.24}$$

depends on the residual terms; that is, the sum of squared means and the covariance. It is most accurate if the variances are high and the correlation is low.

The second moment approximation to the variance has proven especially helpful for MIS. The second moment is a convex functional of the MIS weighting function w_t with a well-known optimum – the balance heuristic [Veach and Guibas 1995b]. If the balance heuristic is used for the weighting function, the second moment is also a convex functional of the PDFs p_t and a convex function of the sample counts n_t. The former has been exploited for *MIS compensation* [Karlík et al. 2019] and the latter for sample count optimization [Lu et al. 2013; Sbert et al. 2019]. Further, the second moment is also easily computable; for example, by simply squaring all sample values,

$$\mathrm{M}[\langle F\rangle_{\mathrm{MIS}}] \approx \sum_{t\in\mathcal{T}} \sum_{i=1}^{n_t} \left(\frac{w_t(x_i) f(x_i)}{n_t p_t(x_i)} \right)^2. \tag{3.25}$$

This ease of computation is exploited by our method to optimize the sample counts, presented in Chapter 7.

3.2.2 Weighting functions

One way to increase the efficiency of an MIS estimator is by optimizing the weighting functions $w_t(x)$. Given a fixed set of sampling techniques with fixed PDFs and sample counts, previous work has derived heuristic weighting functions with provable error bounds and the optimal weights for techniques with independent samples.

3.2.2.1 The balance heuristic

The balance heuristic is a provably good, although not optimal, choice for the weighting function. It sets the weight of technique t proportional to the product of sample count and density [Veach and Guibas 1995b]:

$$w_t^{\mathrm{bal}}(x) \propto n_t p_t(x). \tag{3.26}$$

That is, the normalized weight is:

$$w_t^{\mathrm{bal}}(x) = \frac{n_t p_t(x)}{\sum_{t'} n_{t'} p_{t'}(x)}. \tag{3.27}$$

The balance heuristic approximately minimizes the variance by minimizing the second moment. More precisely, it solves a constrained optimization problem,

$$w_t^{\mathrm{bal}} = \arg\min_{w_t} \left[\sum_{t \in \mathcal{T}} \int_{\mathcal{X}} \frac{(w_t(x_i) f(x_i))^2}{n_t p_t(x_i)} \, \mathrm{d}x + \int_{\mathcal{X}} \lambda(x) \left(\sum_{t \in \mathcal{T}} w_t(x) - 1 \right) \mathrm{d}x \right], \tag{3.28}$$

where a Lagrange multiplier (second integral) is used to ensure normalization of the weights.

Intuitively, the balance heuristic performs well when the approximation error in Equation (3.24) is small; that is, when all techniques have high variance and little or no covariance. Veach and Guibas [1995b] originally proved that the variance with the balance heuristic is never much worse than that of any other weighting function. Specifically, they derived that

$$\mathrm{V}[\langle F \rangle_{\mathrm{bal}}] - \mathrm{V}[\langle F \rangle_{\mathrm{MIS}}] \leq \left(\frac{1}{\min_t n_t} - \frac{1}{\sum_t n_t} \right) F^2, \tag{3.29}$$

where $\mathrm{V}[\langle F \rangle_{\mathrm{MIS}}]$ denotes the variance of an MIS estimator with any other weighting function. However, these bounds assume that the weighting functions must be positive, which is not required to form a valid estimator. Further, they also assumed that the covariance is zero. Unfortunately, there are applications in rendering where the covariance can be arbitrarily high, resulting in arbitrarily poor performance of the balance heuristic, as discussed in Chapter 6.

3.2.2.2 The power, maximum, and cut-off heuristics

Veach and Guibas [1995b] have proposed alternative weighting heuristics for cases where techniques can have low variance. Or, in other words, for cases where the second moment is a poor approximation of the full variance. The power, maximum, and cut-off heuristic amplify the weighting of the balance heuristic by further increasing the weights of techniques that have high weight. For example, the power heuristic with exponent 2 reads:

$$w_t^{\mathrm{pow}}(x) \propto (n_t p_t(x))^2 . \tag{3.30}$$

Such amplification may or may not reduce variance when combining low-variance sampling techniques. Success is not guaranteed and, on average, the power, maximum, and cut-off heuristics perform worse than the balance heuristic [Veach and Guibas 1995b].

Figure 3.2 shows two examples for cases where the balance heuristic performs poorly due to low variance. In the first case, the power heuristic can improve the result by increasing

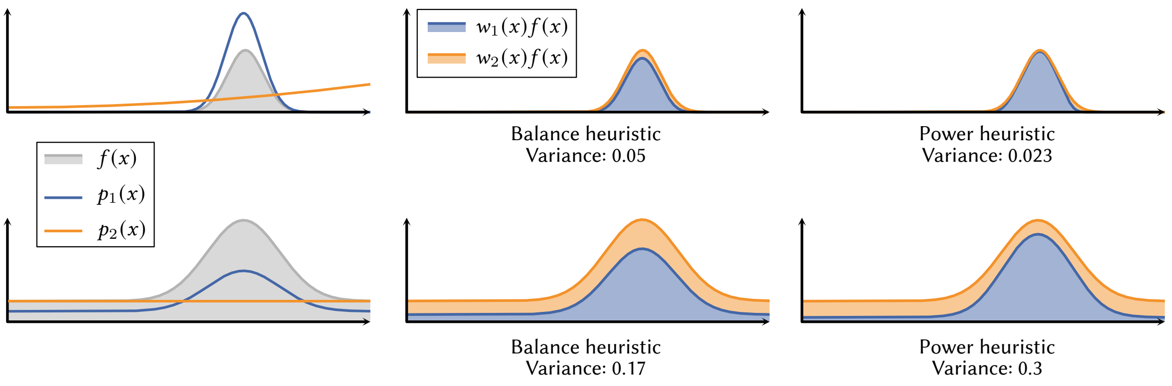


Figure 3.2: *Two simple failure cases of the balance heuristic. The plots on the left show the integrand (gray) and PDFs (blue and orange). The second and third column show the stacked area plots of the weighted contributions assigned to the orange and blue techniques. Both rows integrate a normal distribution, and the blue PDF is proportional to the integrand, $p_1 \propto f$. In the first row, the orange PDF is a smoothly varying polynomial, in the second row it is uniform. In both examples, the balance heuristic performs poorly, assigning a non-zero weight to the orange technique. In the first case, the power heuristic improves matters, in the second case it makes them worse – it is not a reliable solution to the problem.*

the weight of the better technique. However, in the second case, the power heuristic further amplifies the poor weighting, making matters even worse. In rendering, the first case is common when computing direct illumination on glossy surfaces – the original motivation behind the power heuristic – while the second case can occur, for example, with bidirectional sampling. Chapter 5 proposes an alternative solution, injecting variance estimates into the balance heuristic to remedy the problem in both scenarios.

In the context of importance sampling for virtual point light selection, Georgiev et al. [2012b] suggest to sidestep the low-variance weighting issue by incorporating prior knowledge. Specifically, they alter the maximum heuristic

$$w_t^{\max}(x) = \begin{cases} 1 & \text{if } \forall t' : n_t p_t(x) > n_t p_{t'}(x) \\ 0 & \text{else,} \end{cases} \tag{3.31}$$

which assigns unit weight to the technique with highest density, by including a scaling factor α_t,

$$w_t^{\max}(x) = \begin{cases} 1 & \text{if } \forall t' : \alpha_t n_t p_t(x) > \alpha_{t'} n_{t'} p_{t'}(x) \\ 0 & \text{else.} \end{cases} \tag{3.32}$$

This scaling factor is chosen to be large if our prior knowledge indicates that the technique often has low variance. This adaptation greatly benefits defensive sampling scenarios, where a close approximation is combined with regularizing densities to avoid overfitting and unbounded variance in corner cases. There, we know that the close approximation most of the time achieves low variance and hence can safely distort weighting in its favor.

3.2.2.3 Optimal MIS weights

It is possible to derive the optimal MIS weighting functions for cases involving independent samples, that is, zero covariance [Kondapaneni et al. 2019]. In the following, we review the core aspects of these optimal MIS weights. More details and insights can be found in our original paper [Kondapaneni et al. 2019] and our follow-up work [Hua et al. 2023].

Idea. As discussed above (Section 3.2.2.1), the balance heuristic is the MIS weighting function that minimizes the second moment of the MIS estimator. Hence, it is *approximately* optimal as it minimizes an upper bound of the variance. Truly optimal MIS weights can be found by instead minimizing the full variance $\mathrm{V}[\langle F\rangle_{\mathrm{MIS}}]$ (see Section 3.2.1). That is, optimal MIS weights are the solution to this constrained minimization problem:

$$w_t^{\mathrm{opt}}(x) = \arg\min_{w_t} \underbrace{\left[\mathrm{V}[\langle F\rangle_{\mathrm{MIS}}] + \int_{\mathcal{X}} \lambda(x) \left(\sum_{t\in\mathcal{T}} w_t(x) - 1\right) \mathrm{d}x\right]}_{\mathcal{L}}. \tag{3.33}$$

Again, a Lagrange multiplier is used to guarantee unbiasedness. Luckily, this is a convex optimization problem, so a direct solution can be found.

Theoretical solution. Kondapaneni et al. [2019] find the solution to the above minimization problem by computing derivatives, setting them to zero, and applying a substitution to turn the optimization into a simple linear system that can be solved. For that, we first compute the partial derivatives. Assuming *zero covariance*, these are [Kondapaneni et al. 2019, Appendix B]:

$$\frac{\mathrm{d}}{\mathrm{d}w_t(x)}\mathcal{L} = 2\frac{w_t(x) f^2(x)}{n_t p_t(x)} - 2\frac{f(x)}{n_t}\int_{\mathcal{X}} w_t(x') f(x')\,\mathrm{d}x' - \lambda(x), \text{ and} \tag{3.34}$$

$$\frac{\mathrm{d}}{\mathrm{d}\lambda(x)}\mathcal{L} = \sum_{t\in\mathcal{T}} w_t(x) - 1. \tag{3.35}$$

Then, we must set the above expressions equal to zero and solve the resulting linear system for the weighting functions $w_t(x)$. To facilitate that, Kondapaneni et al. [2019] first eliminate the integral by substituting

$$\alpha_t = \int_{\mathcal{X}} w_t(x) f(x)\,\mathrm{d}x. \tag{3.36}$$

With that, the first set of equations can be simplified to

$$\frac{\mathrm{d}}{\mathrm{d}w_t(x)}\mathcal{L} = 0 \Leftrightarrow w_t(x) = \alpha_t \frac{p_t(x)}{f(x)} + \frac{n_t p_t(x)}{2f^2(x)}\lambda(x). \tag{3.37}$$

Substituting that into the constraint derivative gives an expression for $\lambda(x)$,

$$\frac{\mathrm{d}}{\mathrm{d}\lambda(x)}\mathcal{L} = 0 \Rightarrow \lambda(x) = 2\frac{f^2(x) - f(x)\sum_{t\in\mathcal{T}} \alpha_t p_t(x)}{\sum_{t\in\mathcal{T}} n_t p_t(x)} \tag{3.38}$$

The final solution can now be found by first substituting (3.38) into (3.37), and then substituting the result into (3.36) (see Kondapaneni et al. [2019, Appendix B]). This can then be rewritten as a linear system

$$\boxed{\forall t : \sum_{t'} \alpha_{t'} \int_{\mathcal{X}_t} w_t^{\mathrm{bal}}(x) p_{t'}(x)\,\mathrm{d}x = \int_{\mathcal{X}} w_t^{\mathrm{bal}}(x) f(x)\,\mathrm{d}x,} \tag{3.39}$$

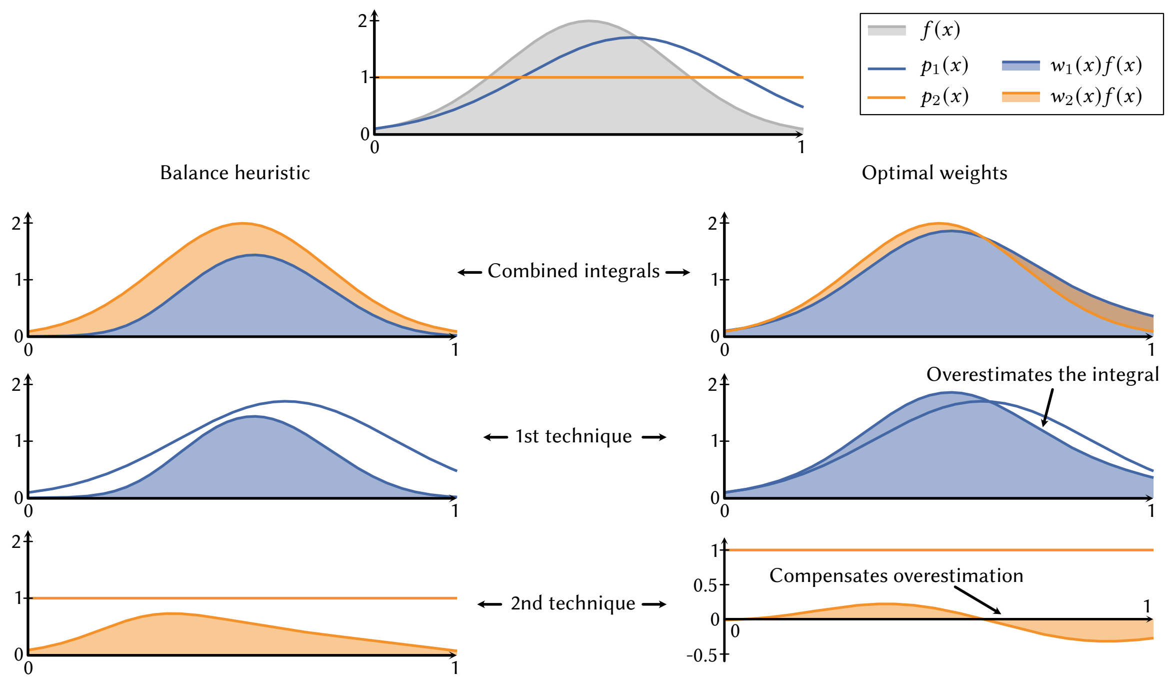


Figure 3.3: *Optimal MIS weights on a simple 1D example. Negative weights allow more expressive affine instead of convex combinations. The optimal MIS estimator has a ten times lower variance.*

that can be solved for the α_t. Here, $\mathcal{X}_t$ denotes the domain of $p_t(x)$, that is,

$$\mathcal{X}_t = \{x \mid p_t(x) \neq 0\}. \tag{3.40}$$

It is important to note that the first integral above is over this domain of p_t, and *not* over $\mathcal{X}$, that is, the domain of the original integrand f. In practice, this means that computing the optimal MIS weights requires tracking samples that would otherwise be ignored because their image contribution is zero. The system comprises one such equation for each techniqe t, leading to a unique solution. Note also that the coefficients of the system are themselves integrals that need to be computed; namely, the balance heuristic weighted integrals of the PDFs p_t and of the integrand f. As a final step, the optimal MIS weighting function [Kondapaneni et al. 2019]

$$\boxed{w_t^{\text{opt}}(x) = \alpha_t \frac{p_t(x)}{f(x)} + \frac{n_t p_t(x)}{\sum_{t' \in \mathcal{T}} n_{t'} p_{t'}(x)} \left(1 - \frac{\sum_{t' \in \mathcal{T}} \alpha_{t'} p_{t'}(x)}{f(x)} \right)} \tag{3.41}$$

is obtained by substituting the found α_t into (3.37). This solution has a subtle caveat: the $f(x)$ in the denominator requires special care. In dividing by $f(x)$, the implicit assumption was made that $f(x) \neq 0$ for all x *sampled by any technique t*. This is usually *not true* in rendering practice.

Negative weights. An interesting insight provided by Kondapaneni et al. [2019] is that better MIS weights can improve upon the balance heuristic almost arbitrarily much. This is because MIS weights do not have to be constrained to be positive. The variance reduction potential through negative weights can be understood visually. Recall that MIS operates by subdividing an integral into an affine combination of technique integrals. An effective weighting function w_t produces an affine combination where each technique has low variance. The chances to find the best such configuration are considerably higher if we do not constrain ourselves to positive weights. Figure 3.3 illustrates this on a simple example. The integrand is estimated with an MIS combination of a crude approximation (blue technique) and a defensive uniform technique (orange). The optimal weights yield ten times lower variance because they allow negative signs. With the optimal weights, the blue technique overestimates the integral, and the orange one compensates for this overestimation. Each has a much lower variance than the corresponding technique estimator with the balance heuristic. The result is an unbiased estimate with ten times lower error.

Practical application. Kondapaneni et al. [2019] demonstrate two ways to leverage optimal MIS weights in practice: An unbiased *progressive estimator*, and a biased *direct estimator*. The *progressive estimator* starts by rendering an initial iteration with the classic balance heuristic. In each pixel, the required integrals for the linear system (3.39) are estimated and stored. Then, a solver is run to obtain initial approximations of the optimal α_t. Subsequent iterations then use these approximate α_t to compute the weighting function (3.41). The linear system is continuously updated and solved. Thus, the MIS weighting converges to the optimal weights over time. Correct use hinges on careful handling of samples where $f(x) = 0$. The integrals in the linear system in (3.39) require these samples, but also the MIS estimator. This can be seen by substituting the optimal MIS weights into the definition of the MIS estimator,

$$\sum_t \sum_{i=1}^{n_t} \frac{w_t^{\text{opt}}(x_{t,i}) f(x_{t,i})}{n_t p_t(x_{t,i})} = \sum_t \alpha_t + \sum_t \sum_{i=1}^{n_t} \frac{f(x_{t,i}) - \sum_{t'} \alpha_{t'} p_{t'}(x_{t,i})}{\sum_{t'} n_{t'} p_{t'}(x_{t,i})}, \tag{3.42}$$

which shows how that a non-zero contribution must now be computed for samples where $f(x) = 0$. The *direct estimator* is a simpler but biased alternative. It also starts by computing Monte Carlo estimates of the coefficients in the linear system (3.39) and solving the system. But, instead of computing the optimal weights, the direct estimator directly computes a biased approximation of the optimally weighted estimate simply as the sum $F \approx \sum_{t \in \mathcal{T}} \alpha_t$. This works, since the α_t are defined as the optimal-MIS-weighted integrals (c.f., Equation (3.36)). For both, the progressive and the direct variant, practical applicability is very limited. Asides from the added overhead and the careful handling of zero-valued samples, the formulation assumes that each pixel computes only a single integral using a set of independent sampling techniques. Even for direct illumination, this is already violated if anti-aliasing is enabled or lens effects are sampled. However, practical use in full global illumination path tracing is possible by exploiting an analogy with control variates [Hua et al. 2023], further discussed in Section 3.3.2.

3.2.3 Sample counts

There have been a few attempts at improving the sample allocation for MIS combinations. Efficiency can be improved by adapting the number of samples that are invested in each technique. For that, three types of methods have been proposed: Domain-specific heuristics, approximate solutions based on variance estimates, and convex optimization methods.

Heuristics. Pajot et al. [2010] were the first to attack the problem of optimal sample allocation for estimating reflected radiance. They designed per-technique heuristics that can measure how “relevant” a technique is for a given configuration. For example, their heuristics for path guiding [Jensen 1995] assess whether the learned incident radiance distribution or the BSDF are more relevant at each point in the scene. However, this formulation cannot easily be generalized to other applications. A conceptually somewhat similar approach was used by Grittmann et al. [2018], where we determined an appropriate number of light paths for photon mapping based on the number of pixels that have a signficiant contribution from photon mapping.

Variance estimates. As an alternative to a heuristic approach, it is possible to set the sample counts based on variance estimates. For instance, Havran and Sbert [2014], Sbert and Havran [2017], and Sbert et al. [2018a,b] suggest to set the number of samples proportional to the inverse efficiency of the sampling techniques. While that can sometimes work, it is not guaranteed to perform better than uniform sample allocation [Sbert et al. 2018a]. Similarly, Sbert et al. [2016] set the sample counts proportional to the variance divided by sample cost, but modify the MIS weights to ignore these sample counts. They prove that this produces higher efficiency than using equal sample counts, but the margin of improvement they achieve in their tests is small. The same authors have later shown that a direct optimization of the combined MIS variance yields better results [Sbert et al. 2019].

Convex optimization. When the balance heuristic is used, the second moment is a convex function of the sample counts [Sbert et al. 2019]. Previous work has exploited that to approximately optimize efficiency, additionally assuming that all techniques have equal cost. Then, second order approximation [Lu et al. 2013], iterative Newton-Raphson optimization [Murray et al. 2020; Sbert et al. 2019], or a gradient descent [Müller 2019] can be used to find approximately optimal sample counts. In Chapter 7 we introduce a method that extends this approach to settings with uneven cost, weighting functions other than the balance heuristic, binary decisions, and optimizations considering the full variance.

3.2.4 Sampling techniques

Finally, the third option to improve the efficiency of an MIS estimator is to adapt the sampling techniques. In previous work, this has been used for learning-based importance sampling, Markov chain Monte Carlo, and environment map sampling.

In the context of tabulated densities – for example, environment map sampling – Karlík et al. [2019] show that a simple contrast enhancement can produce significant performance improvements. They propose to combine BSDF sampling with a modified light sampling technique that covers only the brightest parts of the environment map. Doing so minimizes redundancy between the sampling techniques and increases the odds that a beneficial affine combination can be formed.

In the context of photon mapping, Grittmann et al. [2018] apply an ad-hoc heuristic to focus photon sampling on caustic effects. The reasoning behind that is similar to that of Karlík et al. [2019]: to focus each technique on the effects it is best at.

Similarly, work on path guiding has shown that it is beneficial if the learned PDF has minimal redundancy with the BSDF or next event techniques that it is combined with [Rath et al. 2020; Ruppert et al. 2020]. And for learned light selection, Vévoda et al. [2018] have reported that selection probabilities that heed the MIS weights of the combination with BSDF sampling perform better.

These findings in previous work demonstrate, that it is very beneficial if the techniques in an MIS combination have minimal redundancy and complement each other well. Unfortunately, there is not yet a general solution how to accomplish this goal.

3.3 Related methods

Multiple importance sampling has strong similarities to two other variance reduction methods: mixture densities and control variates. MIS with the balance heuristic can be regarded as stratified sampling of an equivalent mixture density, and the optimal MIS weights yield an estimator identical to an optimal control variate formed from the sampling densities [Kondapaneni et al. 2019].

3.3.1 Relationship to mixture sampling

Mixture sampling constructs a probability density

$$p_{\text{mix}}(x) = \sum_{t \in \mathcal{T}} c_t p_t(x) \tag{3.43}$$

as a convex combination of other densities $p_t(x)$. For example, Gaussian mixture models combine multiple normal distributions with different means and variances. Importance sampling such a mixture density is trivial, provided the individual components can be sampled: First, a component is selected with probability proportionally to its weight c_t. Then, a sample is generated from the component.

Mixture sampling is quite common in rendering. The ubiquitious use-case is importance sampling of material models composed of multiple BSDFs. Since these models are themselves constructed as mixtures, the most intuitive approach to sampling them is to sample from a mixture where each component is (approximately) proportional to one component of the BSDF [Pharr et al. 2016, Chapter 14.1.6].

Mixture sampling is similar to multiple importance sampling in that both approaches combine multiple densities $p_t(x)$. But where MIS divides the integral into subproblems, each estimated with a different density, mixture sampling constructs only a single sampling density. The advantage of mixture sampling is that it can be used recursively without incurring exponential growth. The disadvantage is that it has lower variance reduction potential.

In terms of variance mixture sampling is always worse than a similar MIS estimator with the balance heuristic. This is illustrated in Figure 3.4. The example is modelled after a common use case of mixture sampling: path guiding. The integrand is a product of two functions. Each factor is importance sampled perfectly by the blue and orange densities, respectively. In rendering, these would correspond to the incident radiance L_i and the BSDF ρ. A naïve mixture with equal weights, $c_1 = c_2 = 0.5$ performs badly: The tail on the right is drastically oversampled because the orange density is high there. MIS with the balance heuristic, using the same total number of samples, achieves a two times lower error, as shown in the second column. MIS divides the integral, uses the blue density for almost everything, and only estimates a small residual with the other density. With optimal weights, shown on the right, the error can be reduced even further, by only using

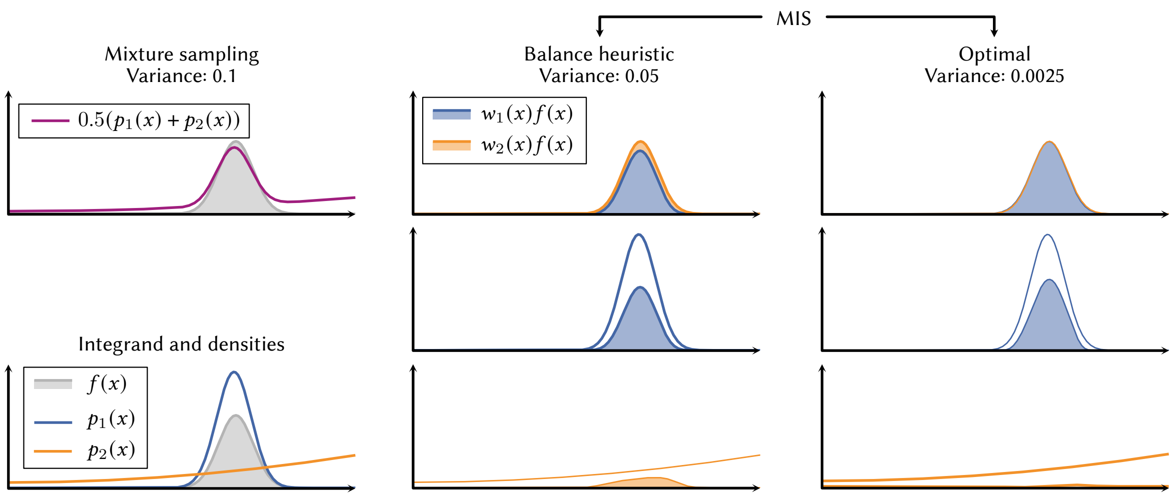


Figure 3.4: *Comparison of mixture sampling and MIS on a simple example. All estimators use exactly two samples in total. The integrand (gray) is a product of two functions. Each factor can be importance sampled (blue and orange). In this extreme case, one factor dominates. Constructing a mixture of both factors performs two times worse than MIS with the balance heuristic (center) and four times worse than MIS with optimal weights (right).*

the orange density for the tiny region where it has a noticable effect on the integral. When done well, MIS can guarantee never to be worse than any of the individual techniques alone. This is not the case with mixture sampling, which can be almost arbitrarily worse by mixing in very bad techniques.

On the flipside, MIS requires us to take at least one sample from every technique. This can be wasteful if only one technique performs well. In the example from Figure 3.4, efficiency could be doubled by simply not using the orange density. Chapter 7 proposes a method to automatically adjust the sample counts as a remedy. The methodology used for this adaptation is closely related to optimization of the mixture coefficients c_t for mixture sampling.

Veach and Guibas [1995b] discussed the connection between mixture sampling and MIS by constructing a hypothetical one-sample model of MIS. Like mixture sampling, this model does not take a fixed number of samples from each technique but rather picks techniques at random. Veach and Guibas [1995b] showed that the balance heuristic is the optimal MIS weight for such a hypothetical one-sample estimator, meaning that it is identical to mixture sampling. Further, they showed that the variance of this mixture sampling estimator is an upper bound to the variance of an MIS estimator with the balance heuristic.

3.3.2 Relationship to control variates

Control variates are an alternative variance reduction technique to importance sampling, as discussed in Section 2.4.3. The idea behind MIS, of combining multiple integrand approximations, can also be applied to control variates. Instead of forming a control variate with a single function $g(x)$, a mixture of functions,

$$g(x) = \sum_t \alpha_t g_t(x), \tag{3.44}$$

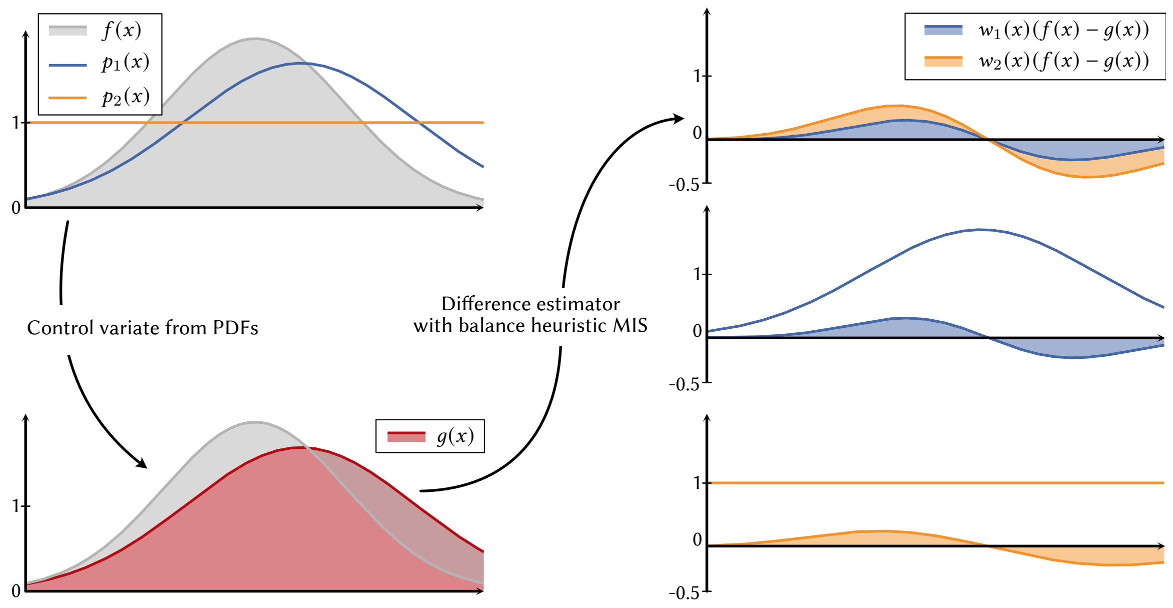


Figure 3.5: *The control variate estimator that is equivalent to an optimal MIS estimator. A linear combination of the sampling techniques is used as the control variate (shown in red). The difference integral is estimated using the same set of techniques with a balance heuristic estimator.*

can be used.

One possible choice for the individual functions g_t is to use sampling densities [Owen and Zhou 2000],

$$g_t(x) = p_t(x). \tag{3.45}$$

This has the advantage that the integral is not only known, it is simply one,

$$G_t = \int_X g_t(x)\,\mathrm{d}x = 1. \tag{3.46}$$

Resulting in a control variate estimator of the form

$$\langle F \rangle_{\mathrm{CV}} = \sum_t \alpha_t + \left\langle \int_X \left(f(x) - \textstyle\sum_t \alpha_t p_t(x)\right)\,\mathrm{d}x \right\rangle. \tag{3.47}$$

How well this estimator performs depends on how the residual term is estimated. One option is to use the same set of sampling techniques and estimate the residual using MIS with the balance heuristic:

$$\langle F \rangle_{\mathrm{CV}} = \sum_t \alpha_t + \sum_t \sum_{i=1}^{n_t} \frac{f(x_{t,i}) - \sum_{t'} \alpha_{t'} p_{t'}(x)}{\sum_{t'} n_{t'} p_{t'}(x)}. \tag{3.48}$$

This formulation has been suggested by Owen and Zhou [2000]. Recently, Kondapaneni et al. [2019] have shown that the above control variate estimator is *identical* to an MIS estimator using the optimal MIS weighting functions, shown in Equation (3.42).

Figure 3.5 visualizes this equivalent control variate for the same example used in Figure 3.3. Since the two estimators are equivalent, they have the same variance. But the intuitive reason why the variance is low differs between the two formulations. Interpreted as MIS weights (see Section 3.2.2.3, Figure 3.3), the variance is low because one technique can easily compute an overestimation of the integral, and the other technique can easily compensate for that. Interpreted as control variates, the variance is low because the difference integral is small.

We have used this equivalence between optimal MIS and optimized, PDF-based control variates to find a (somewhat) practical method that simultaneously achieves optimal MIS and reduced variance of mixture sampling [Hua et al. 2023]. For that, we optimized coefficients for Equation (3.48) and showed that the result is optimal for both, MIS and mixture sampling, even when used concurrently.

3.4 Summary

MIS is a divide and conquer scheme to reduce variance in a Monte Carlo estimator. It separates an integral into a sum of easier to compute integrals, estimates each with a different importance sampling density, and yields an unbiased estimate of the original integral as the sum of all sub-integral estimators.

MIS is an essential building block in modern rendering algorithms. Virtually every single algorithm makes use of it to combine multiple sampling techniques. But, while conceptually simple, MIS is not always easy to apply effectively. Finding the right set of sampling techniques, the best possible weights, or the most efficient sample count allocations are still open research questions.

The work in the following chapters constitutes one step forward towards a perfect MIS algorithm that can achieve (close to) maximum efficiency at all times. To that end, Chapters 5 and 6 identify and rectify robustness problems caused by the balance heuristic and Chapter 7 proposes a practical method to enable or disable techniques on-demand and to automatically adapt their sample counts. Our applications focus on bidirectional rendering algorithms, but the underlying ideas are general and readily applicable to arbitrary Monte Carlo integration problems.

Chapter 4

MIS in the VCM algorithm

The VCM algorithm [Georgiev et al. 2012a; Hachisuka et al. 2012] uses MIS to combine all major surface rendering techniques into a single algorithm. The motivation behind such a holistic combination is robustness: By including pretty much every known sampling technique, we ensure that no scene is rendered extremely slowly. Unfortunately, this robustness comes at a price. For starters, the complexity of the algorithm makes it difficult to implement; especially the correct computation of MIS weights is challenging. But after one has mastered the implementation, bigger problems arise: First, MIS with the classic balance heuristic can perform very badly for VCM, for example because of significant sample correlation. Second, the majority of sampling techniques will be useless for some or all parts of any given scene. Therefore, efficiently using VCM without manual parameter tuning by an expert user is a major challenge.

This chapter defines the sampling techniques that constitute the VCM algorithm, sketches how the MIS weights of these techniques can be computed in practice, and reviews the key challenges for robustness and efficiency.

To keep our discussions to the point, we ignore volumetric scattering and specialized techniques for volumes [Křivánek et al. 2014], assume only area light sources are used, and ignore shading normals and BSDFs with delta distributions, which require special case handling [Veach 1997, Chapter 5]. Further implementation details, such as sample storage or acceleration structures, can be found in the literature [Davidovič et al. 2014; Georgiev 2012; Georgiev et al. 2012a; Veach and Guibas 1995a] and in public implementations [Davidovič and Georgiev 2012; Grittmann 2020].

4.1 The VCM algorithm

VCM is an iterative algorithm. Each iteration first traces a set of light paths[1] and stores them. An acceleration structure for nearest neighbor search is built over the vertices of those paths. Then, a set of camera paths is traced; for example, one per pixel. The camera paths are combined with the light paths to form full path samples. Multiple sampling techniques are used to achieve these combinations; namely,

- unidirectional path tracing,
- next event estimation,
- light tracing,

[1]It is also possible to invert the process, tracing and storing a set of camera paths first.

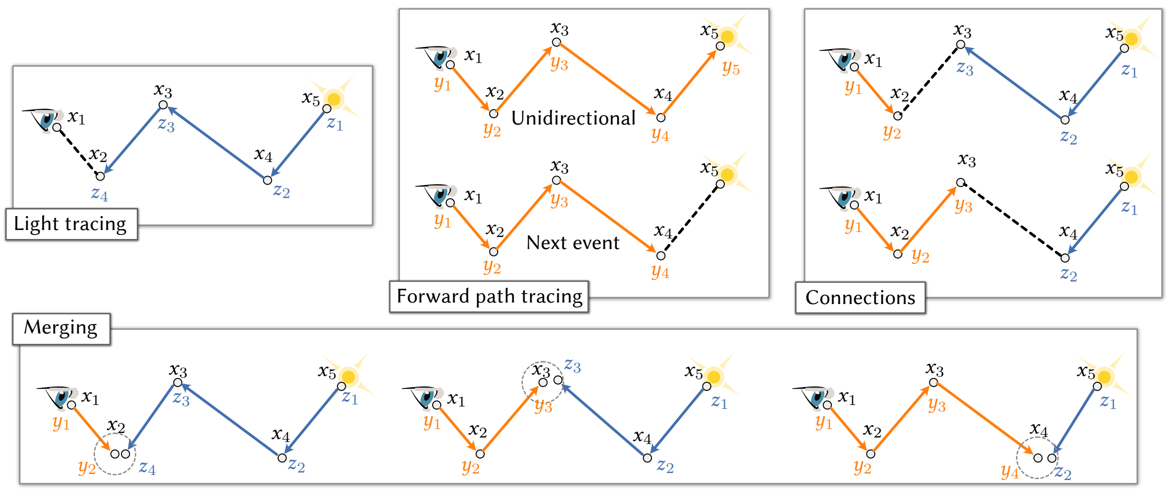


Figure 4.1: *The sampling techniques used by the VCM algorithm. Arrows represent sampled directions, dashed lines are connections, and dashed circles merging (aka photon mapping) operations.*

- connections, and
- merging.

Figure 4.1 illustrates these techniques for paths of length $k = 5$. All techniques generate full paths

$$\overline{x} = x_1 x_2 \ldots x_k \tag{4.1}$$

comprising a series of vertices x_i on surfaces in the scene, where x_1 lies on the aperture of the camera and x_k on an emitter. These paths are a composition $\overline{x} = \overline{yz}$ of a camera prefix

$$\overline{y} = y_1 y_2 \ldots y_t \tag{4.2}$$

of length t, where y_1 lies on the aperture, and a light suffix

$$\overline{z} = z_s z_{s-1} \ldots z_1 \tag{4.3}$$

of length s, where z_1 lies on the surface of an emitter. Path construction is based on three primitive operations: sampling a direction to continue the path (arrows), connecting two subpaths via a visibility test (dashed lines), and merging two subpaths via a nearest neighbor search (dashed circles). Additionally, the origin points y_1 and z_1 are sampled according to the camera and light source model, respectively.

4.1.1 Bidirectional path tracing

VCM is composed of two sets of techniques: the classic bidirectional path tracing techniques [Veach and Guibas 1995a] and the merging techniques. The former – namely, unidirectional path tracing, next event, light tracing, and connections – are unbiased estimators of the path integral. As such, they can be defined via their path PDFs.

Unidirectional path tracing. Unidirectional path tracing generates a path by sampling a position q on the image plane (see Section 2.1) and a position x_1 on the aperture. From these two, the camera model then determines the direction $x_1 \to x_2$. The vertex x_2 is therefore found by tracing a ray in that direction, starting at x_1. From there, the path is extended by repeatedly sampling directions and tracing rays to find the next vertex in that direction. The path space PDF is the product of these local densities, multiplied by their respective Jacobians (see Section 2.1.4 and Section 2.4.2.2):

$$p_{\text{unidir}}(\overline{\mathrm{x}}) = p(x_1)p(q)|J_{\text{cam}}| \prod_{i=2}^{k-1} p(x_i \to x_{i+1}) \frac{\cos\theta(x_{i+i} \to x_i)}{\|x_i - x_{i+1}\|^2}. \tag{4.4}$$

The correctness of this PDF can be verified by plugging it into the path integral estimator,

$$\langle F\rangle_{\text{unidir}} = \frac{f(\overline{\mathrm{x}})}{p_{\text{unidir}}(\overline{\mathrm{x}})} \tag{4.5}$$

$$= \frac{W_i(x_1 \to x_2)\left(\prod_{j=2}^{k-1} \rho(x_{j-1}, x_j, x_{j+1}) \cos\theta(x_j \to x_{j+1})\right) L_{\text{e}}(x_k \to x_{k-1})}{p(x_1)p(q)\prod_{i=2}^{k-1} p(x_i \to x_{i+1})}. \tag{4.6}$$

The Jacobians in the PDF cancel out with the identical Jacobians in the integrand, and the visibility terms are always one, since direction sampling cannot sample occluded points. This leaves us with the exact same estimator that plain forward path tracing computes, indicating that we have indeed used the correct Jacobian factors inside the PDF computation (see also Section 3.1.3).

Next event. Next event estimation also generates paths via forward tracing from the camera. Only the last vertex x_k differs from unidirectional path tracing: It is sampled directly on the surface of a light source. The full PDF in path space is hence almost identical,

$$p_{\text{next}}(\overline{\mathrm{x}}) = p(x_1)p(q)|J_{\text{cam}}| \left(\prod_{i=2}^{k-2} p(x_i \to x_{i+1}) \frac{\cos\theta(x_{i+i} \to x_i)}{\|x_i - x_{i+1}\|^2}\right) p(x_k), \tag{4.7}$$

except that the PDF of the last vertex, $p(x_k)$, is already a surface area density in its natural definition.

Light tracing. Light tracing is just like next event estimation, but in reverse. Path sampling starts by sampling a point $x_k = z_1$ on the surface of a light source. From there, the path is constructed in reverse by sampling a direction and tracing a ray, analogously to forward path tracing. The full PDF, in path space, is again computed by multiplying each directional PDF with the respective Jacobian,

$$p_{\text{light}}(\overline{\mathrm{x}}) = p(x_k) \left(\prod_{i=2}^{k} p(x_i \to x_{i-1}) \frac{\cos\theta(x_{i-i} \to x_i)}{\|x_i - x_{i-1}\|^2}\right) p(x_1) \tag{4.8}$$

The vertex x_1 on the aperture is sampled explicitly, but the image position q is implicitly given by the combination of x_1 and x_2, according to the camera model. Note that the Jacobians use the opposite cosine compared to forward path tracing. In the corresponding estimator, this cosine will therefore cancel out[2] with the cosine in the integrand,

$$\langle F\rangle_{\text{light}} = \frac{f(\overline{\mathrm{x}})}{p_{\text{light}}(\overline{\mathrm{x}})} \tag{4.9}$$

$$= \frac{W_i(x_1 \to x_2)|J_{\text{cam}}|V(x_1, x_2)\left(\prod_{j=2}^{k-1} \rho(x_{j-1}, x_j, x_{j+1}) \cos\theta(x_{j+1} \to x_j)\right) L_{\text{e}}(x_k \to x_{k-1})}{p(x_1)\left(\prod_{i=2}^{k} p(x_i \to x_{i-1})\right) p(x_k)}, \tag{4.10}$$

[2]Unless shading normals are used, as discussed by Veach [1997, Section 5.3].

which is why, when sampling light subpaths, the opposite cosine is multiplied on the integrand compared to forward sampling.

Connections. Connections combine a camera prefix $\overline{\mathrm{y}}$ of length t with a light suffix $\overline{\mathrm{z}}$ of length $s = k - t$ to form a full path of length k. The two subpaths are traced independently, analogously to forward path tracing and light tracing. The endpoints y_t and z_s are connected via a shadow ray, to evaluate the visibility term in the integrand. The resulting path space PDFs are computed following the same steps as unidirectional path tracing and light tracing on the respective subpaths:

$$p_{\mathrm{c},t}(\overline{\mathrm{x}}) = p(\overline{\mathrm{y}}_t)p(\overline{\mathrm{z}}_s) \tag{4.11}$$

$$p(\overline{\mathrm{y}}_t) = p(y_1)p(q)|J_{\mathrm{cam}}|\left(\prod_{i=2}^{t-1} p(y_i \to y_{i+1})\frac{\cos\theta(y_{i+i} \to y_i)}{\|y_i - y_{i+1}\|^2}\right) \tag{4.12}$$

$$p(\overline{\mathrm{z}}_s) = p(z_1)\left(\prod_{i=1}^{s-1} p(z_i \to z_{i+1})\frac{\cos\theta(z_{i+1} \to z_i)}{\|z_i - z_{i+1}\|^2}\right). \tag{4.13}$$

For paths of length k (in terms of vertices), there are $k - 3$ such connection techniques, one for every possible value of $t \in [2, k-2]$.

4.1.2 Merging

The second set of techniques – namely, the merging techniques – are based on photon mapping [Jensen 2001]. These differ from the classic bidirectional path tracing techniques in one crucial aspect: They are biased. To be precise, merging techniques compute an approximation of a higher-dimensional integral that, in turn, is an approximation of the rendering equation [Georgiev et al. 2012a; Hachisuka et al. 2012]. These differences make it harder, but fortunately not impossible, to define a meaningful MIS estimator.

In the following, for the sake of brevity, we introduce the mathematical formulation of the merging technique on the simple example of direct illumination. Extension to the full path integral is straightforward.

4.1.2.1 Merge integral

Merging computes the average reflected radiance on a disk of radius r around the shading point x. This averaging enables caching and reuse of samples, which is the key benefit of merging. Figure 4.2 sketches the approximation. Instead of evaluating the incident light at x from sample points y, merging evaluates the incident light at other points x' and then averages these estimates to obtain an approximation of the reflected radiance at x.

Concretely, merging integrates the reflected radiance over the disk $\mathcal{D}_r(x)$, multiplying by a kernel $k(x', x)$ for normalization,

$$L_{\mathrm{r}}(x, \omega_{\mathrm{o}}) \approx \int_{\mathcal{D}_r(x)} k(x', x)L_{\mathrm{r}}(x', \omega_{\mathrm{o}})\,\mathrm{d}x'. \tag{4.14}$$

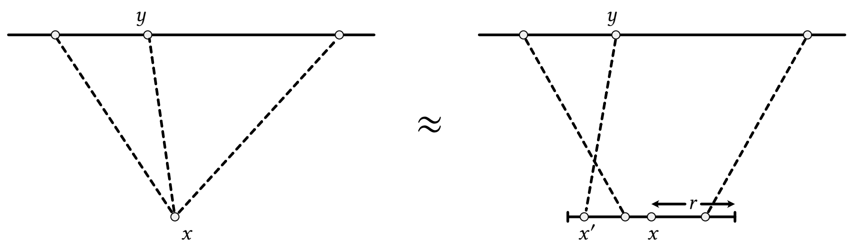


Figure 4.2: *Comparison of the surface integral, on the left, and the merge integral, on the right. The former integrates over all points y visible from x. The latter additionally integrates over all points x′ within the disk of radius r around x. The two formulations are identical if r = 0.*

The choice of kernel function is of secondary importance. Indeed, even a simple box filter,

$$k(x', x) = \begin{cases} \frac{1}{\pi r^2} & \text{if } x' \in \mathcal{D}_r(x) \\ 0 & \text{else} \end{cases} \tag{4.15}$$

can work well.

The approximation error can be considerably reduced by evaluating the BSDF and cosine at the actual surface point x. That is, merging typically computes

$$L_{\mathrm{r}}(x, \omega_{\mathrm{o}}) \approx \int_{\mathcal{D}_r(x)} k(x', x) \int_{\mathcal{A}} L_{\mathrm{i}}(x', y) \rho(\omega_{\mathrm{o}}, x, y) \cos\theta(x \to y) \frac{\cos\theta(y \to x')}{\|x' - y\|^2} \,\mathrm{d}y\mathrm{d}x'. \tag{4.16}$$

Naturally, this approximation will be more accurate the smaller $\mathcal{D}_r$ is. However, a too small radius will result in an inefficient estimator. A good trade-off can be found by locally adapting the radius based on sample statistics [Kaplanyan and Dachsbacher 2013b; Lin et al. 2020]. Further, consistent estimates can be computed by progessively reducing the radius [Hachisuka et al. 2008], starting with a large radius to obtain an efficient but inaccurate approximation and then iterating with smaller and smaller radii to gradually eliminate the approximation error without sacrificing too much efficiency.

4.1.2.2 Merge estimator

The first step in the merge estimator is to precompute a set of n pairs (y, x'). Analogously to classic bidirectional techniques, these are generated by first sampling a point y on the light and then sampling a direction to find x' through ray tracing. The corresponding PDF,

$$p(y, x') = p(y)p(y \to x') \frac{\cos\theta(x' \to y)}{\|x' - y\|^2}, \tag{4.17}$$

is hence identical to that of the bidirectional path tracing techniques.

The reflected radiances at many shading points x are then approximated using the same set of precomputed merge samples for all of them. This computation is facilitated through an additional approximation: In general, the merge disk is a hypothetical construct that does not exist as actual scene geometry. To sidestep this, the points x' are gathered from the spherical neighborhood in 3D and projected onto the merge disk through rotation. Figure 4.3 illustrates this. The actual sample positions x', marked in blue, are rotated such

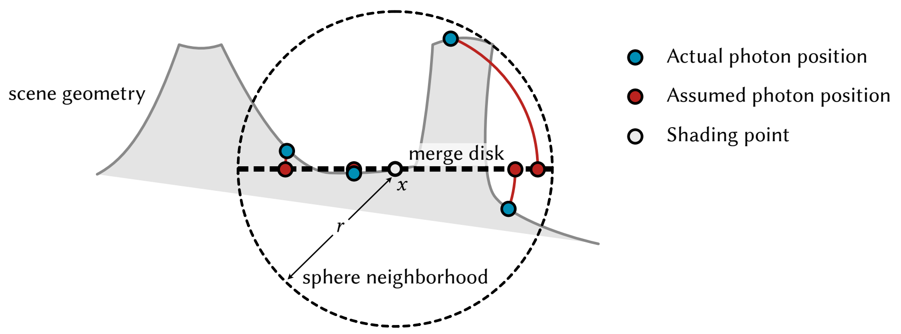


Figure 4.3: *The merge disk $\mathcal{D}_r$ generally does not exist as actual scene geometry. Therefore, an additional approximation is employed, projecting the nearby points onto the hypothetical disk to compute the kernel value. This introduces additional approximation error, the magnitude of which depends on the amount variation in the scene geometry within the merge radius.*

that they lie on the merge disk, as marked in red. In other words, for the sake of the kernel evaluation, the distance between x and x' is computed in 3D space, and we simply pretend that x' lies on the merge disk at this distance. Hence, the kernel $k(\|x' - x\|)$ is a function of the distance only.

The result is the merge estimator for direct illumination,

$$\langle L_\mathrm{r} \rangle \approx \frac{1}{n} \sum_{i=1}^{n} k(\|x' - x\|) \rho(\omega_\mathrm{o}, x, y_i) \frac{\cos\theta(x \to y_i)}{\cos\theta(x'_i \to y_i)} \underbrace{\frac{L_\mathrm{e}(y_i \to x'_i)\cos\theta(y_i \to x'_i)}{p(y_i)p(y_i \to x'_i)}}_{\text{cached}} . \tag{4.18}$$

Here, the ratio of cosines arises because the cosine at the neighbor point, $\cos\theta(x' \to y)$, is part of the PDF of the cached sample. We multiply by the ratio of cosines to ensure that the estimate contains the cosine at the shading point[3]. This reduces the error under strong normal variation.

4.1.2.3 Approximation error

The merge estimator employs two approximations to facilitate efficient computation: First, the reflected radiance is averaged over the disk neighborhood, and, second, all points within the sphere of radius r are assumed to lie on that disk.

The resulting approximation error is more severe in some cases than in others. While the overall approximation error vanishes as the radius reduces, the exact error, with the same radius, can vary hugely depending on the scene geometry. Figure 4.4 sketches example scenarios for common sources of error.

The averaging over the disk neighborhood, of course, performs poorly if the lighting varies within the radius. For example, shadow boundaries are blurred, and light can "leak" through thin surfaces. This is sketched in the first three examples in Figure 4.4.

The error due to the planar assumption can be less obvious at first. The key problem is

[3]If shading normals are used, it is important to keep in mind that cosines that are part of Jacobians, such as $\cos\theta(x' \to y)$, are with respect to the actual normal of the geometry, while the desired $\cos\theta(x \to y_i)$ is with respect to the shading normal.

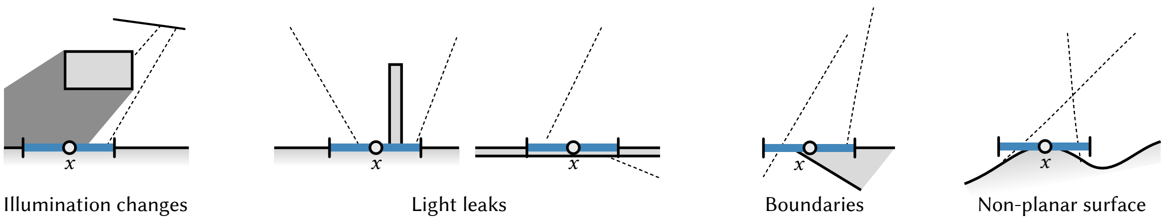


Figure 4.4: *Sources of approximation error in merging techniques. Shadow boundaries and light leaks are prominent examples where visible artifacts arise because the merge integral only approximates the rendering equation. On top of that, geometric boundaries and non-planar surfaces are scenarios where the hypothetical domain of the merge kernel is not present as actual geometry.*

that the kernel is no longer normalized. For example, if x is close to a boundary, the actual area from which samples are gathered is much smaller than the disk assumed for kernel normalization. The result is an underestimation that can manifest as darkening close to boundaries, because the kernel value is smaller than it should be. Conversely, on non-planar surfaces away from boundaries, the actual area can be much larger than that of the hypothetical disk. There, the result will be overly bright.

4.1.2.4 Surrogate PDF for MIS weighting

Compared to the classic bidirectional techniques, the merge integral computes a different, higher-dimensional, integral. Light scattering along a path of length k is approximated with $k+1$ iterated surface integrals, due to the additional integration over the merge disk. So how can we define a meaningful MIS combination that involves both merging and classic techniques?

Previous work has shown that this can be done by computing a surrogate path space PDF that approximates the sampling density of the merging technique [Georgiev et al. 2012a]. For that, merging is interpreted as a resampling technique. Specifically, merging at depth t is treated like a connection at that depth, except that it uses n samples – with n being the number of light paths – and samples are stochastically rejected if the additional vertex z_{s+1} does not lie on the merge disk around y_t.

To compute this surrogate PDF, another approximation is needed: The probability that z_{s+1} lies in the merge disk,

$$P(z_{s+1} \in \mathcal{D}_r(y_t)) = \int_{\mathcal{D}_r(y_t)} p(z^*|\overline{z}_s)\mathrm{d}z^* \tag{4.19}$$

is itself an integral that cannot be computed easily. Luckily, Georgiev et al. [2012a] have shown that, for the sake of MIS weighting, a sufficiently close approximation can be computed by assuming uniformity over the disk. Then, the probability is simply the product of the PDF of any z_{s+1} in the merge disk, multiplied by the area of the disk,

$$\int_{\mathcal{D}_r(y_t)} p(z^*|\overline{z}_s)\mathrm{d}z^* \approx p(z_{s+1}|\overline{z}_s) \int_{\mathcal{D}_r(y_t)} 1\mathrm{d}z^* = p(z_{s+1}|\overline{z}_s)\pi r^2. \tag{4.20}$$

The full surrogate PDF of the merging technique is then the PDF of the corresponding connection technique, as defined above, multiplied with the approximate acceptance probability, and the number n of light paths,

$$p_{\mathrm{merge},t}(\overline{\mathrm{x}}) = p(\overline{\mathrm{y}}_t)p(\overline{\mathrm{z}}_s)p(z_{s+1}|\overline{\mathrm{z}}_s)\pi r^2 n. \tag{4.21}$$

This surrogate can be used to compute, for instance, balance heuristic MIS weights. Since the surrogate is defined in path space, this weighting will be valid and the combination will be unbiased. Except, of course, for the bias due to the merging technique itself.

4.2 Challenges and shortcomings of MIS in VCM

The VCM algorithm is a powerful method to efficiently render certain types of scenes; namely, interior scenes with focused indirect illumination or caustics. Nevertheless, it has seen only little adoption in production. The key obstacle is that VCM is only efficient on some scenes. Often, it is greatly outperformed by forward path tracing.

The most apparent flaw of a huge combination like VCM is that samples are wasted on underperforming or redundant techniques. But also MIS weighting with the balance heuristic can be far from ideal in the context of VCM. First, effects sampled by forward path tracing with low variance are weighted poorly. Consequently, the combined result can be worse than only the forward path tracing samples *that are part of the combination.* Second, sample correlation in the merging technique is problematic. Heeding that correlation during MIS weight computation can boost performance tenfold.

The following sections elaborate these problems and challenges. The work presented in Chapters 5 and 6 tackles the weighting issues encountered with the classic balance heuristic, and Chapter 7 shows how to adapt the set of techniques to reduce redundancy and the wasting of samples on underperforming techniques.

4.2.1 Low-variance effects

The first problem of MIS in VCM is closely related to the “low-variance” problem already discussed by Veach and Guibas [1995b] (see Section 3.2.2). Since the balance heuristic is based on a minimization of the second moment, rather than the full variance, it is known to perform poorly when the variance of some technique(s) is low.

We observed that this low-variance problem can be particularly severe for bidirectional algorithms [Grittmann et al. 2019] (Chapter 5). There, effects that are sampled by forward path tracing with low variance are often weighted poorly by the balance heuristic. An example is shown in Figure 4.5. The illumination in the canonical Cornell box scene is quite simple for forward path tracing to capture. There, adding bidirectional samples increases the variance, due to poor MIS weighting by the balance heuristic. Scenes dominated by such effects will therefore be rendered worse in *equal-iteration count* by VCM than by forward path tracing. Combined with the much higher cost of VCM, the consequence is a far slower rendering than with plain forward path tracing.

In our earlier work on efficient caustic rendering [Grittmann et al. 2018], we countered this problem through artificially reducing the weights of the light tracing and photon mapping techniques that are most affected by this problem. This was done by replacing the

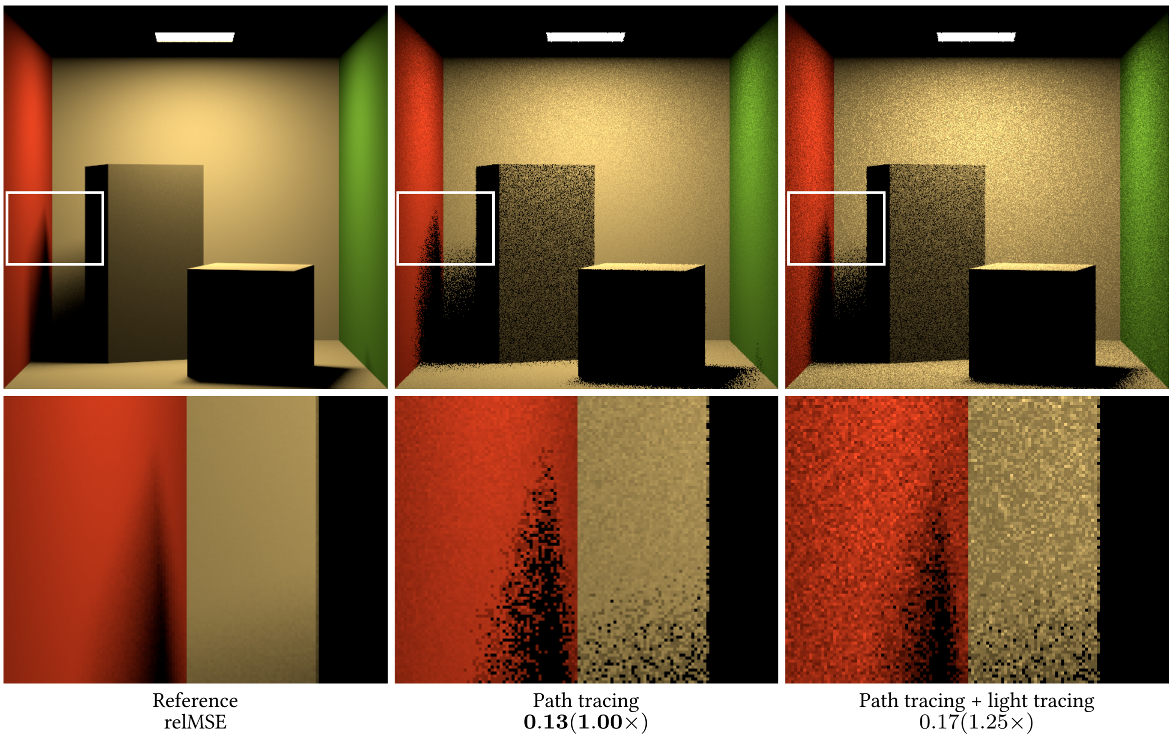


Figure 4.5: *Low-variance problem in VCM. The center image is rendered with only forward path tracing. The image on the right uses* these exact same samples *and combines them with the light tracing technique using balance heuristic MIS. Adding these extra samples increases the average image error by 25%, as measured by the relative mean squared error (relMSE) below each image.*

actual number of light paths by an arbitrary lower one, inspired by the alpha-max heuristic [Georgiev et al. 2012b] and the approximation of Popov et al. [2015]. However, that ad-hoc workaround only helped identify difficult paths but did not yield a robust weighting heuristic.

In the original work, Veach and Guibas [1995b] proposed the alternative power, maximum, and cut-off heuristics to try and combat such low-variance problems (see Section 3.2.2). Unfortunately, we found that, in bidirectional applications, these heuristics actually *worsen* the problem more often than not. As an alternative, we devised a scheme to incorporate variance estimates into the MIS weights. This method is described in Chapter 5.

Of course, in a perfect world, we could solve the problem of weighting low-variance techniques by "simply" using the optimal MIS weights [Kondapaneni et al. 2019] (see Section 3.2.2.3). Unfortunately, since the number of techniques in VCM is huge, these would be extremely costly to apply. Further, approximations have to be made to compute the optimal weights in the first place, and its implementation is greatly complicated by necessities such as handling of zero-valued samples.

4.2.2 Correlation

The second problem encountered by MIS in the VCM algorithm is that of correlations. There are three general types of correlation in the algorithm; namely, pixel correlation, intra-technique sample correlation and inter-technique sample correlation. None of the three have been discussed extensively by previous work.

The problem with correlation is that few previous methods account for it. The original weighting heuristics [Veach and Guibas 1995b] operate on the second moment alone and the optimal weights [Kondapaneni et al. 2019] operate on the variance of independent samples. Popov et al. [2015, supplemental document] derived the optimal MIS weights for cases with intra-technique correlation. The result is an even more complex – and hence more expensive and difficult to compute – expression than the optimal weights for independent samples. Hence, it has not been directly used in rendering so far. Popov et al. [2015] used it to motivate a heuristic for their use case, for which we propose an alternative in Chapter 6 that also supports the VCM algorithm.

4.2.2.1 Pixel correlation

Many bidirectional methods share one set of light paths across all pixels in the image. Doing so enables efficient implementation [Davidovič et al. 2014] and variance reduction through resampling [Nabata et al. 2020; Popov et al. 2015; Su et al. 2022]. Due to this sharing, the pixel estimates are positively correlated (see Section 2.3.4). The magnitude of that correlation is governed chiefly by two factors: (1) the number of light paths and (2) how these paths are allocated to each pixel.

One source of pixel correlation are the merging techniques. For nearby pixels, merging at the primary hit point x_2 computes an average of the same set of light paths. Therefore, it yields correlated estimates, manifesting as low-frequency splotches in the image.

Noticable pixel correlations can also arise from the connection techniques. That happens only if the implementation combines each camera prefix with a randomly selected light subpath [Davidovič et al. 2014; Popov et al. 2015; Su et al. 2022], rather than always paring two unique paths [Veach and Guibas 1995a]. In that case, it is possible for nearby pixels to select the same subpaths for connection. The amount of correlation depends on the probability with which neighboring pixels pick the same light subpath. For example, if path selection is done uniformly, and the number of light paths is the same as the number of pixels, then the probability for any two pixels selecting the same light path is very low. Consequently, pixel correlation will be low. However, as the number of light paths reduces, the probability to use the same path grows, and so does the pixel correlation. Similarly, if two pixels use the same non-uniform probability distribution to select light paths, they will also have high positive correlation.

An extreme case is the method of Popov et al. [2015]. First, they use a very small number of light paths – only a few thousand – to bound the overhead of their resampling method. Second, they build probability distributions over light paths that are stored in spatial caches and interpolated between nearby points. Consequently, nearby pixels will use the same distribution. The combination of both design decisions can yield huge pixel correlation. For example, the connections at the primary hit point x_2 of nearby pixels will generate almost identical samples most of the time. This positive correlation manifests as artifacts in the form of splotches in the image.

Figure 4.6 demonstrates both the effect of pixel correlations and their cause. Here, we use the method of Popov et al. [2015] as an example, and vary the two key parameters: the

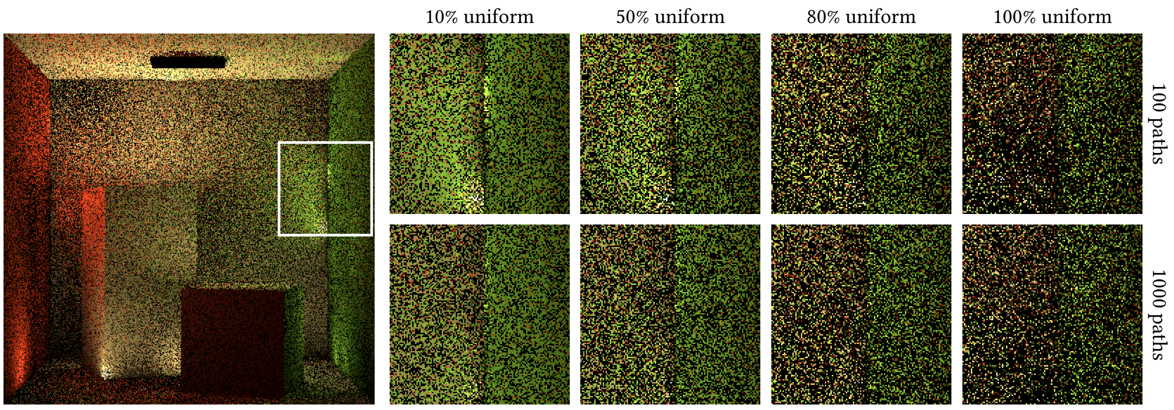


Figure 4.6: *Positive pixel correlation manifests as low-frequency artifacts. Here, we render single-bounce indirect illumination with the method of Popov et al. [2015], using only a single sample per pixel and a single technique (connection at x_2). Thereby, we isolate the effect of pixel correlation. The zoom-ins from left to right show how correlation evolves when interpolating between importance resampling and uniform sampling. The first row uses only 100 light paths, the second ten times as many. Correlation is reduced by reducing the probability that nearby pixels generate the same path. Here, this can be achieved by either using a more uniform probability, or by using more light paths. Unfortunately, the former increases the per-pixel error and the latter increases the cost.*

number of light paths and the selection probability. In the first row of zoom-ins, only 100 light paths are used. The second row uses ten times as many. Similarly, the first column uses mostly the learned selection weights, while the last uses only uniform selection. Uniform selection from a large number of light paths has no noticable correlation artifacts, while learned selection from a small number of paths has severe problems.

Such splotchy artifacts are problematic, because they are impossible for a denoiser to remove. Denoisers are, at their core, low-pass filters, and applying a low-pass filter on a low-frequency artifact has no effect. Also, adaptive sampling in the presence of positive pixel correlation is challenging, because the positive correlation might be mistaken for low variance, depending on the approximations used by the adaptive sampler.

In their method, Popov et al. [2015] suggest to ameliorate the problem by artificially reducing the MIS weights of the connection techniques. That, of course, is only a heuristic workaround to the problem. A more rigorous option to reduce pixel correlation is to randomize the estimate. The amount of correlation depends on the probability of two pixels to generate the same sample. Hence, if we add additional randomization, like mixing in a uniform distribution, this probability will be reduced. This trick has been used, for example, in the context of photon mapping to aid with denoising and adaptive sampling [Estevez and Kulla 2020].

An ideal solution to the problem of pixel correlations is yet to be found.

4.2.2.2 Technique correlation

Correlations also frequently arise within the same pixel. One example is inter-technique correlation, that is, correlation between the samples used by different techniques. In bidirectional algorithms that correlation is ubiquitious. For example, in classic bidirectional

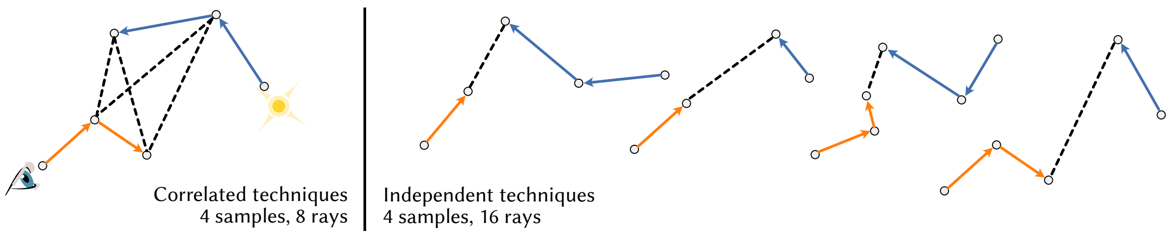


Figure 4.7: *The sampling techniques in VCM are typically correlated to avoid quadratic growth in sampling cost. The illustration on the right shows how path samples of length four would be constructed independently in a perfect world. The illustration on the left shows how correlating them – as is commonly done – reduces the sampling cost.*

path tracing [Veach and Guibas 1995a], all connection techniques operate on the same light and camera subpaths. Therefore, they are positively correlated. So far, this correlation has not been identified as a major problem. However, we are also not aware of any thorough analysis proving that these correlations are definitely never a problem.

Figure 4.7 visualizes this correlation by sketching how independent connection techniques would construct paths, compared to how typical implementations actually operate. In a perfect, correlation-free world, each connection would use its own camera prefix and light suffix. Unfortunately, the number of connection techniques grows linearly with the path length, so the total number of ray tracing operations would grow quadratically if we wanted to enforce independent samples. Therefore, connections typically share the same underlying paths. This reuse of subpaths allows us to use a large number of techniques at little extra cost. The resulting correlation does not seem to be a problem. At least, no one has of yet shown it to be.

In our experiments, we found one case where correlation between techniques is problematic; namely, the combination of light tracing and merging at the primary hit point. These two techniques operate on the exact same set of samples and compute almost identical estimates. The main difference is that merging additionally blurs these estimates across pixels. This is sketched in Figure 4.8. The figure shows, from left to right, the images of just the light tracing technique, just the merging technique, VCM including both, VCM without merging, and VCM without light tracing. There is no extra value from using both techniques, since they are essentially the same. On the contrary, using both techniques only serves to distort the MIS weights, effectively doubling the weight that these samples should be assigned. In the example in Figure 4.8, this further amplifies the low-variance weighting problem because the already too high balance heuristic weight is effectively doubled.

In our experiments, we always disable merging at the primary hit point, to minimize the bias. Alternatively, we could also disable the light tracing technique. The choice which technique to keep depends on the application and the code base of the renderer. Merging introduces bias and positive pixel correlation, but allows for a more streamlined camera sampling implementation, since all paths of all techniques start by sampling a direction from the camera. Further, light tracing is the only technique that requires random access writes to pixels in the image, which may be problematic for some applications. Šik and Křivánek [2019] therefore decided to disable light tracing instead. Since the two techniques are virtually identical in terms of their computed estimates and corresponding error, the

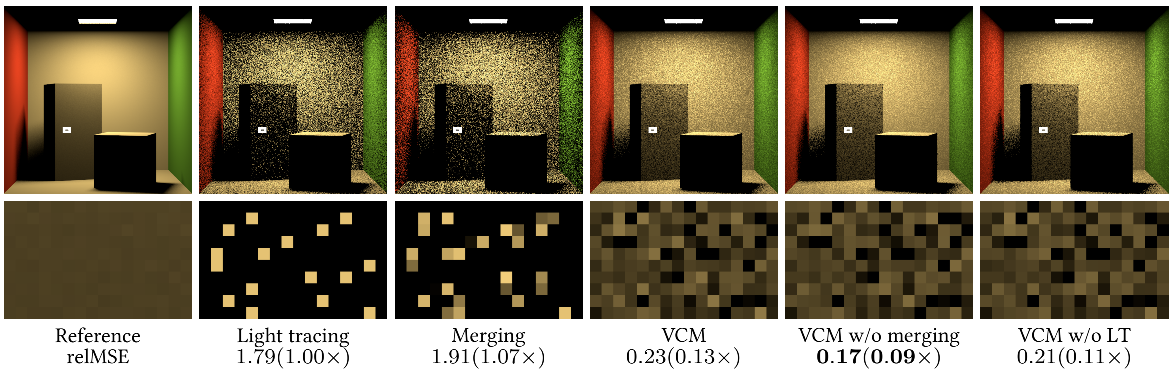


Figure 4.8: *Merging at the first hit point, x_2, is almost perfectly correlated with the light tracing technique. The first two images compare the renderings of the individual techniques. In the zoomed-in region, the two produce exactly the same noise pattern except for the blur introduced by the merging kernel. Having both enabled in the same algorithm offers no advantage and only distorts the MIS weights. Therefore, as shown on the right, results are improved by disabling either of the two.*

choice depends solely on such implementation aspects.

4.2.2.3 Sample correlation

The most problematic type of correlations we found in VCM are sample correlations within the same technique. Specifically, correlations in the merging technique that arise because the same camera prefix is continued with many suffixes. Photon mapping is a splitting estimator: each camera prefix subpath branches out into multiple (photon) suffix subpaths. The difference to the classical splitting estimator is that the suffixes are sampled from the emitters, cached in the scene, and reused over multiple prefixes.

It only takes a simple scene to demonstrate the catastrophic failure of MIS with correlated samples. Figure 4.9 shows a diffuse box illuminated by a small area light source at two different positions: near the floor (top row) and near the ceiling (bottom row). We consider length-4 paths $\overline{\mathrm{x}} = x_1x_2x_3x_4$ (that is, one-bounce indirect illumination) and vertex merging techniques only. For each path there are thus two possible techniques: merging at x_2 and x_3, respectively. We compare three variants of the balance heuristic for combining these two techniques: classical [Veach and Guibas 1995b], variance-aware [Grittmann et al. 2019] (Chapter 5), and correlation-aware [Grittmann et al. 2021] (Chapter 6). As the light source moves closer to the ceiling, the variance of merging at x_3 explodes as the camera subpath is less likely to find the shrinking, brightly illuminated spot on the ceiling. However, this is not reflected in the weights of the classical balance heuristic, which ends up producing an extremely noisy image.

The two techniques differ only in the direction in which the edge x_2x_3 is sampled. And the corresponding path densities are equal when the vertices x_2 and x_3 are both diffuse, since in that case $p(x_2 \to x_3) = p(x_3 \to x_2) = G(x_2 \leftrightarrow x_3)/\pi$, where G is the geometry term, which is symmetric w.r.t. x_2 and x_3. The classical balance heuristic thus assigns equal weights to the techniques, regardless of the geometry of the actual path.

While the variance-aware weights (Chapter 5) can solve this problem in this simple case, they are not practical. They require computing variance estimates of every single merging

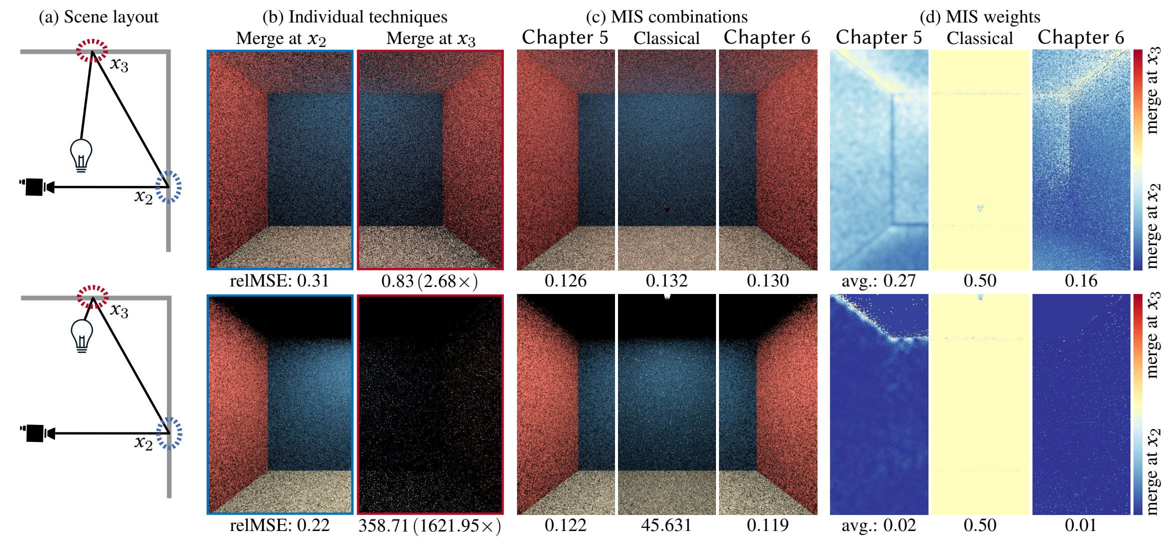


Figure 4.9: *A box with diffuse materials and a small area light (shining upwards) positioned far from the ceiling (top) and close to it (bottom). We render length-4 paths and consider vertex merging techniques only. The variance of merging at x_3 increases as the light moves closer to the ceiling, resulting in a dark image with all energy focused in a few outliers. The classical balance heuristic does not capture that increased variance and assigns equal weights to the two merging techniques in both scene configurations, producing a poor combination in the latter.*

technique, for every path length. Crucially, these estimates need to be accurate, which is expensive for techniques that produce nothing but outliers. (We have used 128 samples/pixel for the experiment in Figure 4.9, yet the image appears black as all energy is focused in just a few severe outliers.) Hence, variance-aware weighting is not efficient in this setting.

The approach of Popov et al. [Popov et al. 2015], which sets $n = 1$ in the classical heuristic, would not work. For one, n cancels out when only merging techniques are being combined. Worse, when vertex connection techniques are included, setting $n = 1$ effectively disables all merging techniques, reverting VCM to bidirectional path tracing. The heuristics of Jendersie et al. [Jendersie 2019; Jendersie and Grosch 2018] address this specific failure case but use unintuitive parameters and can sometimes perform worse than the classical balance heuristic.

In Chapter 6 we introduce a heuristic remedy for this problem. Our correlation-aware balance heuristic yields consistently better estimates than the classic balance heuristic, while being simple and cheap to compute. All scenes with light sources close to surfaces benefit tremendously from this method.

4.2.3 Efficiency

Asides from the original MIS heuristics performing poorly, there is another major obstacle towards making VCM a practical and general rendering solution: the wastefulness of the method. In any given scene – or, in fact, in any given part of a scene – most of the samples taken by the vast amount of techniques that constitute VCM are utterly wasted. Each technique in the mix only performs well for a narrow subset of all possible effects.

An extreme case occurs in most outdoor scenes. There, the majority of light paths will never find the visible region. Consequently, a lot of computation time is wasted on tracing those irrelevant paths. For those cases, one solution is to adapt the sample density of the light paths to increase the probability of finding the visible region [Grittmann et al. 2018; Šik et al. 2016]. Another option is to disable bidirectional techniques altogether. The latter is preferrable if, even with better importance sampling, the bidirectional techniques add little value over forward path tracing.

The wastefulness of the method is amplified by the fact that the individual techniques in VCM are quite expensive. For example, enabling the merging techniques can easily double the render time, depending on the implementation specifics of the nearest neighbor search and associated material evaluations. Therefore, it should be used sparingly. At the same time, merging only benefits small parts of a scene; specifically, caustics and reflected caustics. Everywhere else, merging is wasted and its contribution either weighted down by MIS, or incorrectly assigned a high weight due to the correlation issues mentioned above.

To achieve efficient rendering in any scene, the set of techniques must be adapted to the scene. For example, when rendering a vast outdoor environment without caustics, only forward path tracing should be used. Conversely, when rendering an indoor scene with caustics and focused indirect illumination, almost all techniques in VCM should be enabled, possibly with an increased sample count. Of course, such adaptation can be done manually, but that would require the user to posess extensive knowledge of rendering algorithms and their properties. Further, adaptation on a more local scale, for example, on a per-pixel level, would be difficult to achieve manually, and finding exact sample counts will always require some amount of trial and error, even from an expert user. To overcome these challenges, Chapter 7 shows how to automatically decide the set of techniques and associated sample counts on-the-fly during rendering.

4.3 Summary

Realistic scenes – even if constrained to just basic surface scattering – exhibit a wide range of illumination effects. Robust rendering of those is a problem to which the VCM algorithm offers a solution. By combining as many sampling techniques as possible, a rendering algorithm is achieved that can render any scene in *acceptable* time.

Mathematically, this combination is achieved through multiple importance sampling (MIS). For that, the individual techniques are formulated in a common domain, the path space. For the unbiased bidirectional techniques, this is done by a simple change of variables. The path space probability densities are computed by multiplying the original densities of the technique's natural domain by the appropriate Jacobians. Matters are less simple for the merging technique. Merging computes a higher-dimensional approximation of the path integral and therefore requires special case handling. For that, surrogate path-space PDFs can be defined, treating the additional dimension as a discrete resampling step.

The resulting algorithm in its basic form is far from perfect. Asides from the fact that implementation is non-trivial, the performance can be subpar in many scenes. We identify three main causes for this unsatisfactory performance. First, MIS weighting with the classic balance heuristic can perform poorly when forward path tracing has low variance. Second, when merging samples are highly correlated, MIS with the balance heuristic simply fails altogether. Third, because only a subset of techniques is beneficial for any given lighting effect, performance is severely encumbered by the superfluous techniques. The following three chapters each propose a practical solution for one of these three problems.

Chapter 5

Variance-aware MIS

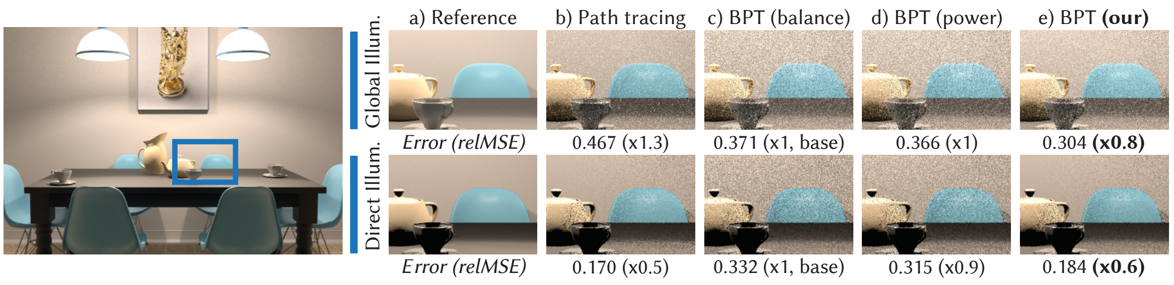


Figure 5.1: *A scene rendered with bidirectional path tracing where the balance heuristic performs poorly. For the low-variance direct illumination effects, the unidirectional samples alone produce better results than the exact same samples combined with additional bidirectional ones, due to poor MIS weighting via the balance heuristic. Our variance-aware balance heuristic provides a practical solution to this problem.*

Robustness is an important goal for MIS estimators. Ideally, an MIS combination should never be worse than using only the samples of the best technique. That way, the overall efficiency is, in the worst case, only reduced by the computation time wasted on underperforming techniques. Unfortunately, this robustness is not always achieved when using the balance heuristic.

Figure 5.1 shows a practical example. It compares a rendering with bidirectional path tracing to the result of using just the unidirectional samples that are part of that MIS combination. The direct illumination component is visibly more noisy with the full MIS combination than with just the forward path tracing subset. The balance heuristic does not achieve a robust estimator in this example.

Our method enhances the balance heuristic with variance estimates. The resulting weights increase the robustness in failure cases, as shown in Figure 5.1, without any negative impact in other cases. In this chapter, we introduce our *variance-aware* balance heuristic. We discuss how our method can improve robustness in failure cases via illustrative 1D examples and in a rendering application. The method has been previously published [Grittmann et al. 2019], I was the main author of that paper. The source code of our implementation in the PBRT renderer [Pharr et al. 2016] is available on GitHub[1].

[1]https://github.com/pgrit/var-aware-mis-pbrt

5.1 Variance-aware balance heuristic

MIS is not the only means of combining multiple Monte Carlo estimators. An alternative is to weight each estimator by its (estimated) reciprocal variance. In the following, we first review this alternative approach and then show how its benefits can be combined with those of the balance heuristic.

5.1.1 Variance-based weighting

Robust combinations can also be achieved by weighting a set of estimators by their (estimated or approximated) reciprocal variances σ_t^{-2} [Hammersley and Handscomb 1968],

$$\langle F \rangle_{\text{const}} = \sum_t \frac{(\sigma_t)^{-2}}{\sum_{t'} (\sigma_{t'})^{-2}} \langle F \rangle_t . \tag{5.1}$$

The resulting estimator is consistent, that is, it will converge to F as the sample count grows to infinity. It is, however, only unbiased if the samples used to estimate the variances are statistically independent of those used by the "actual" estimators [Kirk and J. Arvo 1991].

Variance-based weighting has two main drawbacks when compared to MIS. First, it relies on the generally unknown variances of the sampling techniques. But it is possible to estimate these on the fly and still achieve unbiased, or at least consistent, combined estimates. Second, while MIS can divide a challenging integral into multiple easier ones (see Section 3.1.1), variance-based weighting only works on a global scale. Hence, it has a lower potential for variance reduction.

There is, however, one major advantage of variance-based weighting. As discussed in Section 3.2.2.1, the balance heuristic only minimizes a portion of the variance – the second moment – and can hence perform poorly. For example, if there is significant covariance. Variance-based weighting, in contrast, guarantees that the result will always be at least as good as the best sampling technique alone. Provided, of course, the variances are estimated or approximated with sufficient accuracy.

5.1.2 Reaping the benefits of both

Our idea is simple: We combine variance-based weighting and the balance heuristic, to reap the benefits of both approaches. For that, we inject additional *variance factors* into the balance heuristic, accounting for the mismatch between second moment and full variance.

Recall that the balance heuristic is the MIS weighting function that minimizes the second moment of the combined estimator (see Section 3.2.2.1). Intuitively, the balance heuristic therefore performs well when the second moment is a good approximation of the variance, that is, when

$$\mathrm{V}[\langle F \rangle_{\text{MIS}}] \approx \sum_{t \in \mathcal{T}} \int_{X} \frac{(w_t(x) f(x))^2}{n_t p_t(x)} \, \mathrm{d}x . \tag{5.2}$$

Conversely, when the remaining terms of the variance, namely the covariance and the sum of squared means, have a significant impact, the balance heuristic estimator will perform poorly.

The ratio between the second moment and the full variance of a technique, referred to as the *variance factor* in the following,

$$v_t := \sigma_t^{-2} \int_{\mathcal{X}} \frac{f^2(x)}{n_t p_t(x)} \, \mathrm{d}x, \tag{5.3}$$

is an indicator of how well the balance heuristic will perform for combinations involving this technique t. If $v_t \approx 1$, the technique's variance is dominated by its second moment and the balance heuristic will handle it well. If $v_t > 1$, the second moment of this technique is much higher than the full variance. The balance heuristic will assign it a too low weight. For example, a zero-variance technique will have an infinitely higher second moment,

$$\sigma_t^2 \to 0 \Rightarrow v_t \to \infty, \tag{5.4}$$

and should receive an MIS weight $w_t(x) = 1$, while the balance heuristic will generally produce $w_t(x) < 1$ for such zero-variance techniques. If $v_t < 1$, the second moment is smaller than the variance. This happens only if the samples are positively correlated. In that case, the balance heuristic will assign a too high weight.

Our heuristic simply multiplies the balance heuristic weights by these variance factors:

$$w_t^{\text{var}}(x) \propto v_t n_t p_t(x). \tag{5.5}$$

So, when the balance heuristic works well, that is, when $v_t \approx 1$, the weights remain unchanged. But, when the balance heuristic does not perform well, we increase (or decrease) its weight proportionally to the ratio between second moment and full variance.

Section 5.2 analyses the performance of this heuristic in an idealized 1D setting where variances are known. Section 5.3 proposes a practical implementation in bidirectional path tracing that achieves consistent performance improvements over the balance heuristic.

5.2 Discussion in 1D

This section provides an empirical evaluation of our heuristic on low-dimensional integration problems. These are modeled to mimic scenarios commonly encountered in rendering, while being simpler to visualize and reason about. We discuss high-variance and low-variance combinations, stratification, and sample reuse. Figure 5.2 shows the integration problems and compares the performances of the different weighting schemes.

Product sampling. The first example (first row in Figure 5.2) is the common setting where the integrand is a product of two factors, and the sampling techniques are each (almost) proportional to one of them. In rendering, this corresponds, for example, to combining BSDF sampling and light sampling for direct illumination. The balance heuristic (c) is almost identical to the optimal weights (e). Variance-based weighting (b) relies almost exclusively on one technique and performs far worse than the balance heuristic. Nevertheless, the result with our variance-aware heuristic (d) is almost the same as the balance heuristic. That is because the variances of both techniques are high and thus well approximated by the second moment. The power heuristic (with power 2) (d), in comparison, doubles the error of the balance heuristic.

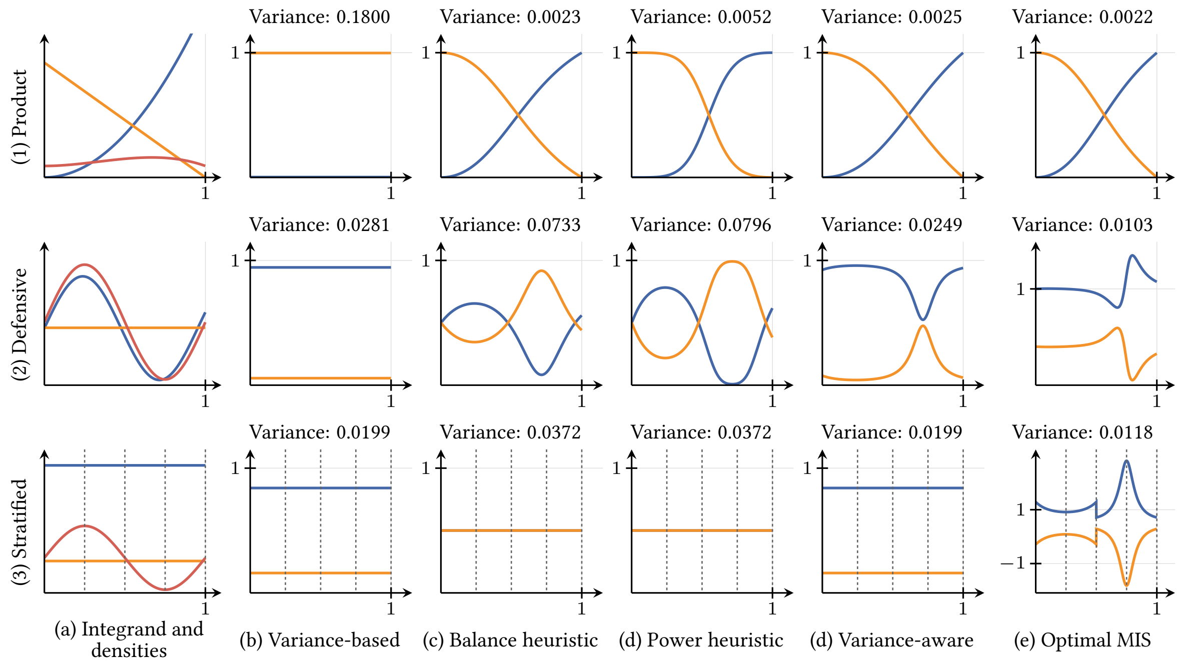


Figure 5.2: *Different MIS weighting schemes applied to 1D integration problems. The first column shows the integrand (red) and the two sampling densities (orange and blue). The following columns are the weights and resulting variances used by different combinations. Our method combines the robustness of variance-based weighting with the variance reduction potential of the balance heuristic.*

Defensive sampling. Another common example (second row in Figure 5.2) is combining an almost perfect importance sampling technique with a defensive one. For example, learned distributions for path guiding are combined with BSDF sampling to avoid bias and severe noise from learning errors. In this scenario, the blue technique has very little variance. The balance heuristic, however, assigns high weight to the poorly performing orange technique, with the power heuristic amplifying the problem even further. Variance-based weighting performs much better, and our heuristic yields results that are better still. Optimal MIS weighting [Kondapaneni et al. 2019] (see Section 3.2.2.3) could, in theory, cut this variance in half again, but remains difficult to apply in practice.

Stratification. Lastly, we inspect the setting that originally motivated our variance-aware heuristic: combining stratified and unstratified sampling techniques. The setup is shown in the last row of Figure 5.2. Both sampling techniques use a uniform density and take four samples from it. While the blue technique distributes the samples uniformly over the whole domain, the orange one employs *stratification*. That is, it subdivides the domain into four bins and samples uniformly within each one. This configuration occurs in bidirectional path tracing, where light tracing splats contributions over the whole image, while camera paths are stratified, tracing exactly one path per pixel.

It is a well known fact that such stratification reduces the variance (see, for example, Veach [1997], pp. 50-52). However, it does not have any impact on the balance heuristic. The unstratified technique takes $n_u = 4$ samples with density $p_u(x) = 1$, while the stratified one takes $n_s = 1$ sample in each bin with density $p_s(x) = 4$. Therefore, in this example, the balance heuristic assigns equal weight to both techniques. Our variance-aware heuristic, by design, is equivalent to variance-based weighting in this case.

5.3 Discussion in rendering applications

In this section we evaluate the performance of our method in two rendering applications: bidirectional path tracing and defensive sampling for light source selection. The full-size images of all results can be found in the supplemental material of the original publication [Grittmann et al. 2019], along with the source code and scripts to reproduce them. The latter are also available on GitHub: https://github.com/pgrit/var-aware-mis-pbrt. The tests were performed on a workstation with an Intel i7-4790 processor and 32 GB of RAM.

We compare our method to the optimal MIS weights of Kondapaneni et al. [2019] (see Section 3.2.2.3) only for the simpler defensive sampling application. Incorporating these weights into a bidirectional path tracer is a non-trivial task. Aside from the cost of maintaining multiple large matrices for every pixel, the optimal weights also have to consider paths with zero contribution, which requires major incisions into the light transport and material logic. We therefore do not attempt to compare this method to ours on the full bidirectional path tracer. Such a comparison would be interesting to assess the margin for further improvements, but it is beyond the scope of this work.

5.3.1 Implementation

To show its suitability to light transport problems, we have implemented our proposed weighting heuristic in two MIS applications: bidirectional path tracing, which combines techniques with and without image plane stratification, and a defensive sampling application for direct illumination. Both have been implemented in PBRT [Pharr et al. 2016], sharing the code for estimating and utilizing the variance factors v_t. The source code can be found on GitHub or in the supplemental materials of the original publication [Grittmann et al. 2019].

Procedure. Our method is easily implemented on top of progressive rendering algorithms. To utilize variance estimates, rendering is split into two main stages. In the first stage, we estimate the image with a few samples per pixel (spp), using the standard balance or power heuristics in all MIS calculations. Our implementation takes 1 spp in this stage. Based on these samples, we estimate the variance of each technique and compute v_t. In the second stage, any further rendering iterations use our variance-aware heuristic. By estimating the v_t factors in the first stage, and only using them in the second stage, we ensure that the result is unbiased [Kirk and J. Arvo 1991].

Handling initial samples. Samples from the first stage are not wasted. There are three options to proceed with the rendering result of the first stage: Simply average with the second stage (noisy if the balance heuristic performs poorly), average but weigh based on the v_t factors (biased [Kirk and J. Arvo 1991]), or keep only those pixels where no technique has a variance factor above a certain threshold (unbiased). We chose the last approach, with a threshold of $\max_t v_t < 2$.

Estimating v_t. Theoretically, we would like to compute a v_t factor per technique in every pixel. To allow our method to work with a small number of samples, even just one per pixel, we instead divide the image into equal-sized tiles (8×8 in our tests). For each tile, we compute the sample variance and sample mean of each technique. The result are per-tile

v_t factors that are used for all pixels within the tile. This approach is simpler and cheaper than approximating v_t for every pixel. Its downside is that the sample variance increases if the tile contains a discontinuity, effectively reverting to the balance heuristic for such tiles.

5.3.2 Results

To compare results numerically, we use the relative mean squared error (relMSE) metric. The relMSE of an image is computed by dividing the squared error of each pixel by the reference pixel value, then taking the average of the result among all pixels. This avoids error values being dominated by bright pixels, which the commonly used (root) mean squared error metric is susceptible to.

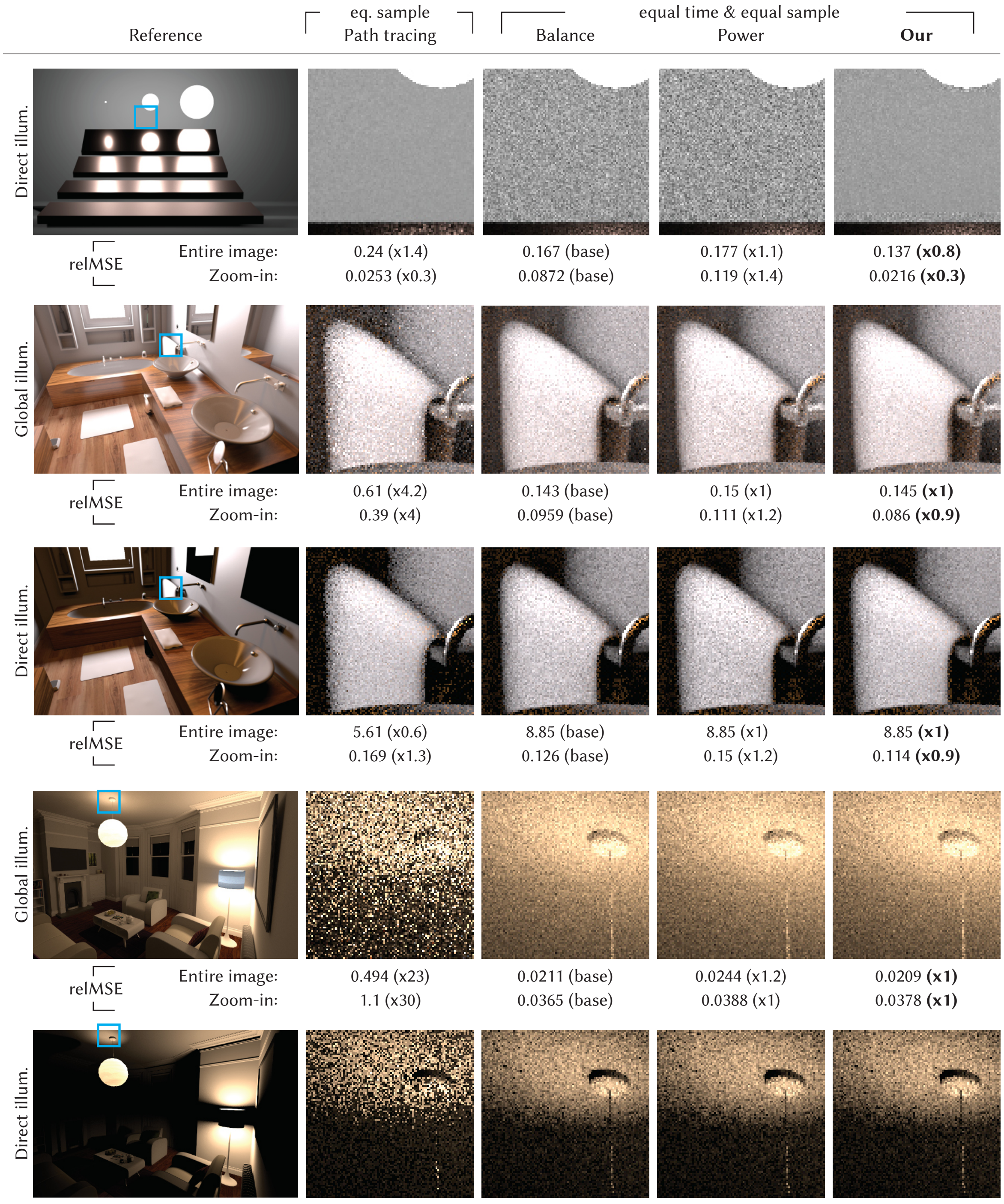
5.3.2.1 Bidirectional path tracing

Most techniques in bidirectional path tracing are stratified over the image plane, with one exception – light tracing, which connects light paths to the camera via shadow rays. Not being restricted to sampling within a given pixel is the very reason why light tracing is efficient at rendering caustics, focused indirect illumination, and even some types of direct illumination (e.g., lights close to surfaces, inside volumes, or with peaked emission profiles). As we observed before, however, the classical MIS heuristics cannot capture the variance reduction due to stratification in the other techniques, in this case over the image plane. Figure 5.1 shows how this can result in excessive noise due to the unstratified light tracer in image regions where the variance of the stratified techniques is low.

Figure 5.3 compares our heuristic to the balance and power heuristics on three scenes. We show results for full global illumination and for direct illumination alone. The numbers under each row provide the relMSE and the ratio of that error to the balance-heuristic error (in parentheses), across both the entire image and the corresponding zoom-in. Note that the comparisons are not only equal-time but also equal-sample, since our method does not introduce measurable computational overhead. The 'path tracing' images have been produced from the subset of next-event estimation samples in BPT (without the need to apply MIS). Therefore, ideally, the noise in the full MIS combinations should never be higher than those samples alone.

The first row in Figure 5.3 shows an extreme case: The error with the balance heuristic is four times larger than the path-tracing samples alone. The MIS-weight comparison in Figure 5.4 shows the reason for this behavior: The figure compares the variance of the three techniques (a) for direct illumination (BSDF samples, next-event estimation, and light tracing) to the per-pixel average MIS weights of the three heuristics. Next-event estimation (middle row) has close to zero variance on the wall behind the light sources. This low variance is partially due to the stratification on the image plane, which is ignored by the balance and power heuristics when combining with the unstratified light tracer. Therefore, the balance heuristic assigns an excessive weight to the light tracer. The power heuristic amplifies the issue further. Our approach accounts for the variance reduction due to stratification and maintains the lower error of path tracing.

The second scene in Figure 5.3, the Bathroom, is a case where no sampling technique has particularly low variance. We achieve small local improvements, as seen in the zoom-ins,



			eq. sample	equal time & equal sample		
	Reference		Path tracing	Balance	Power	**Our**
Direct illum.	relMSE	Entire image:	0.24 (x1.4)	0.167 (base)	0.177 (x1.1)	0.137 **(x0.8)**
		Zoom-in:	0.0253 (x0.3)	0.0872 (base)	0.119 (x1.4)	0.0216 **(x0.3)**
Global illum.	relMSE	Entire image:	0.61 (x4.2)	0.143 (base)	0.15 (x1)	0.145 **(x1)**
		Zoom-in:	0.39 (x4)	0.0959 (base)	0.111 (x1.2)	0.086 **(x0.9)**
Direct illum.	relMSE	Entire image:	5.61 (x0.6)	8.85 (base)	8.85 (x1)	8.85 **(x1)**
		Zoom-in:	0.169 (x1.3)	0.126 (base)	0.15 (x1.2)	0.114 **(x0.9)**
Global illum.	relMSE	Entire image:	0.494 (x23)	0.0211 (base)	0.0244 (x1.2)	0.0209 **(x1)**
		Zoom-in:	1.1 (x30)	0.0365 (base)	0.0388 (x1)	0.0378 **(x1)**
Direct illum.	relMSE	Entire image:	0.0366 (x2.1)	0.0172 (base)	0.0206 (x1.2)	0.0159 **(x0.9)**
		Zoom-in:	0.214 (x5.7)	0.0377 (base)	0.0413 (x1.1)	0.0391 **(x1)**

Figure 5.3: *Equal-time (and also equal-sample) comparisons of BPT using three different MIS heuristics. Our heuristic improves low-variance cases while, in contrast to the power heuristic, never being significantly worse than the balance heuristic overall.*

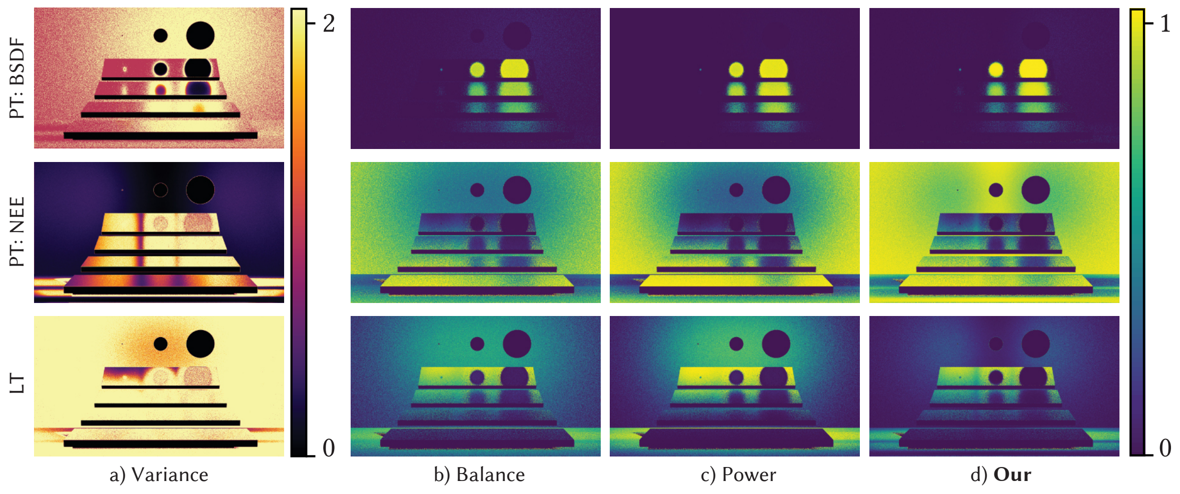


Figure 5.4: *Comparison of the average per-pixel MIS weights (b-d) to the relative variance of three techniques (a): path tracing with BSDF sampling (PT: BSDF), with next-event estimation (PT: NEE), and light tracing (LT).*

while the power heuristic again worsens the results. Importantly, our overall error is not (significantly) worse than that of the balance heuristic.

Finally, the Living room scene shows a case where the unstratified light tracer performs best. Here, our method also achieves small local improvements while not performing worse than the balance heuristic overall. In contrast, the power heuristic produces 20% larger error than the balance heuristic.

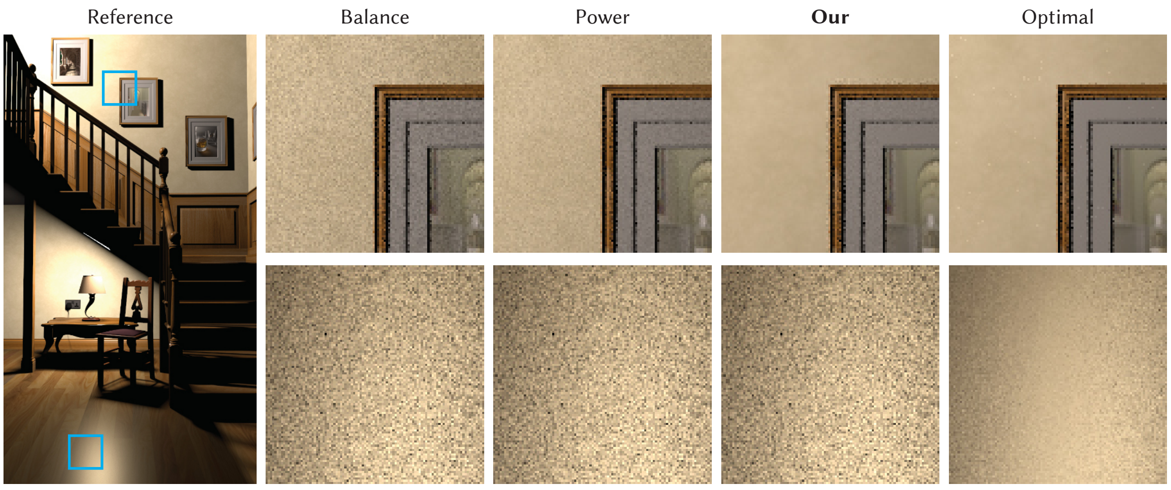
Even with coarse 1 spp variance estimates (without any filtering, denoising, or regularization), our method has never performed significantly worse than the balance heuristic in our experiments. The shape of the v_t factors, which quickly fall off to one (i.e., yielding the balance heuristic) as the variance increases, plays an important role in achieving this robustness.

5.3.2.2 Low-variance direct illumination

Kondapaneni et al. [2019] illustrated their optimal weights on a simple defensive sampling example: They combine two techniques for light selection in direct illumination computation: selecting the light based on estimates of the unoccluded contribution cached in a regular grid (the 'almost proportional' one) [Pharr et al. 2016], and uniformly (the defensive technique). We implemented our heuristic for the same application so as to compare it to the optimal weights.

The results for the Staircase scene are shown in Figure 5.5; the supplemental materials of the original paper [Grittmann et al. 2019] contain additional results. The structure of the figure is the same as in Figure 5.3, where again the error numbers in the parentheses are relative to the balance heuristic.

For the comparison to the approach by Kondapaneni et al., we used their favored 'direct' estimator, which has lower variance and less overhead but also a small amount of bias. The comparisons in this case are equal-sample, using just eight samples per technique



	Balance	Power	Our	Optimal
Entire image:	0.0342 (base)	0.0333 (x1)	0.0342 **(x1)**	0.0122 (x0.4)
Top zoom-in:	0.00803 (base)	0.00388 (x0.5)	0.00196 **(x0.2)**	0.00165 (x0.2)
Bottom zoom-in:	0.0249 (base)	0.0247 (x1)	0.0248 **(x1)**	0.00302 (x0.1)

Figure 5.5: *Equal-sample comparison (8 per technique) for defensive sampling. Our results are close to optimal MIS weights [Kondapaneni et al. 2019] whenever one technique is significantly better than the rest (top zoom-in).*

(and per pixel). For our heuristic, the comparison is also equal-time to the balance and power heuristics. The approach by Kondapaneni et al. is $5-10\%$ slower per sample on our test scenes.

The STAIRCASE scene contains two different cases where significant improvements over the balance and power heuristics are possible: A low-variance problem (top row) and an example where the *negativity* of the optimal weights can reduce the error, despite all techniques having high variance (bottom row). For the low-variance problem, our result is close to that of Kondapaneni et al.'s direct estimator. However, our heuristic cannot be negative and therefore cannot improve on the balance heuristic in the second case (where the optimal weights are negative).

In conclusion, our method and the approach of Kondapaneni et al. complement each other. Their method is provably optimal, yet challenging to apply in practice. Our method is simpler to implement, but only ensures that the combined estimator is never worse than the best technique alone, in an equal-sample comparison.

5.3.3 Comparison to weighting with variance estimates

Figure 5.6 compares our variance-aware MIS weights (top row) to classical variance-based weighting (bottom row) on zoom-ins from various scenes. In some cases, especially the simpler defensive sampling application, using variance estimates alone can produce results similar to our weights. Noise in the variance estimates, however, will then manifest as visible artifacts in the image. Our method prevents these artifacts due to the injection into the balance heuristic and the lower bound of one (i.e., $v_t \geq 1$). While the variance estimates are far too noisy to be usable on their own, incorporating them into the balance heuristic, as we propose, yields improvements without any visible artifacts.

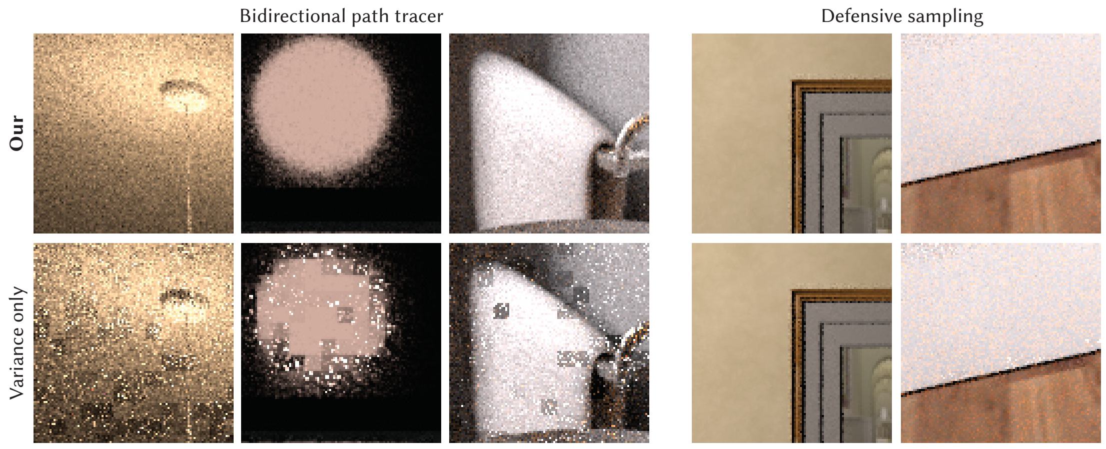
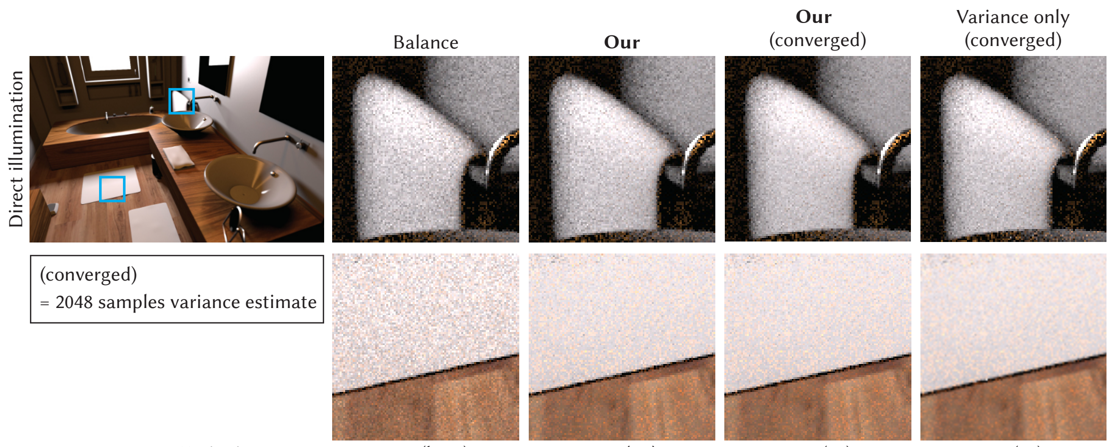


Figure 5.6: *Incorporating the variance estimates into the balance heuristic (top row), as we propose, greatly improves the robustness compared to using these same estimates on their own (bottom row).*



	Balance	Our	Our (converged)	Variance only (converged)
Entire image:	8.85 (base)	8.85 **(x1)**	8.84 (x1)	8.9 (x1)
Top zoom-in:	0.126 (base)	0.114 **(x0.9)**	0.0953 (x0.8)	0.107 (x0.9)
Bottom zoom-in:	0.0566 (base)	0.0316 **(x0.6)**	0.0275 (x0.5)	0.0741 (x1.3)

Figure 5.7: *Comparison to using variance estimates computed with 2048 samples per pixel. Computing more accurate v_t factors can improve the results further, but at higher cost. Even the converged variances alone perform worse than our approach with coarse estimates.*

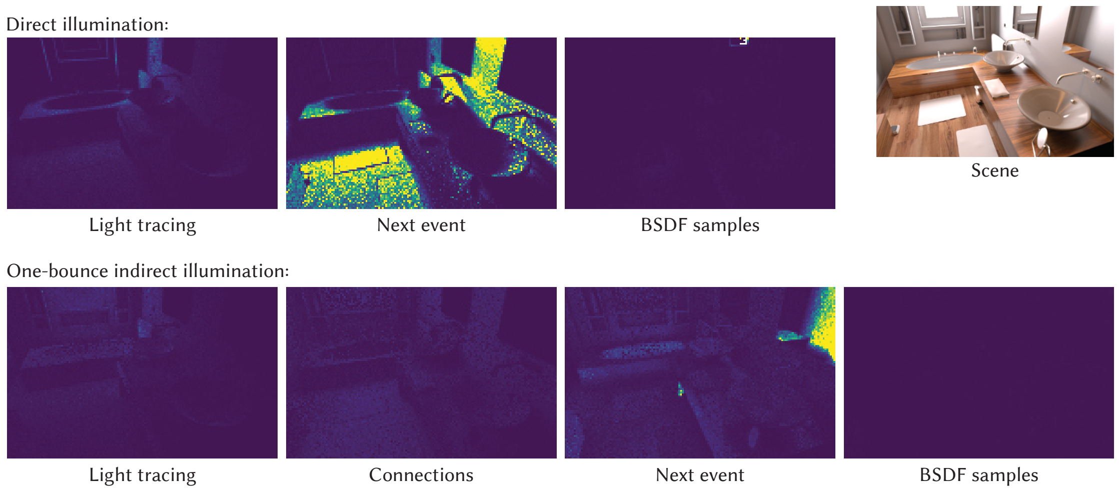


Figure 5.8: *The estimated v_t factors for the bidirectional path tracer techniques in the* Bathroom *scene. The majority of pixels receive a factor close to one.*

We also compare the performance when using accurate variance estimates (computed from 2048 samples). The results for direct illumination on the Bathroom scene are shown in Figure 5.7. Using the accurate variance estimates alone performs worse than using those same estimates with our approach.

5.3.4 Overhead

Computation-wise, the overhead of our method is negligible. However, storing the v_t factors can require a significant amount of memory if the number of combined techniques is very large.

In the bidirectional path tracer, our implementation computes the v_t factors for path lengths up to five – a total of 25 techniques. Since we store v_t every $8{\times}8$ pixels, this requires roughly two bytes per pixel on average – not much in the context of a typical renderer.

The memory footprint can be reduced further. Figure 5.8 shows the v_t factors for the direct and one-bounce indirect illumination in the Bathroom scene. A large fraction of these v_t factors are almost one. Therefore, applying compression would significantly reduce the memory requirements. We leave such optimizations for future work as in our benchmarks the overhead has been insignificant.

5.4 Limitations and future work

In addition to the downside of relying on estimated quantities, some care has to be taken when applying our heuristic to biased techniques. In this section, we discuss such limitations and outline possible improvements. Lastly, we introduce other promising applications for our heuristic.

5.4.1 Limitations

Error in variance estimates. Weighting using estimated quantities can worsen the results when these estimates are highly inaccurate. We have not encountered any such issues in our experiments, despite basing our estimates on a single sample per pixel. There could, however, be applications or particularly challenging scenes where errors in the v_t estimates cause issues.

There are several options to ameliorate such potential issues. The images storing the v_t factors can be filtered or denoised and potential outliers could be clamped. Another approach, naturally, is to increase the number of v_t estimation samples, or to accumulate better estimates progressively.

Narrow-support sampling pdfs. So far we have only considered the case where the pdf of each individual technique is non-zero over the entire integration domain. If this is not the case, weighting based on the variance no longer makes sense. In that case, the problem can be made recursive: A set of techniques, whose pdfs together cover the entire domain, can be combined via the regular balance heuristic, which is equivalent to sampling from

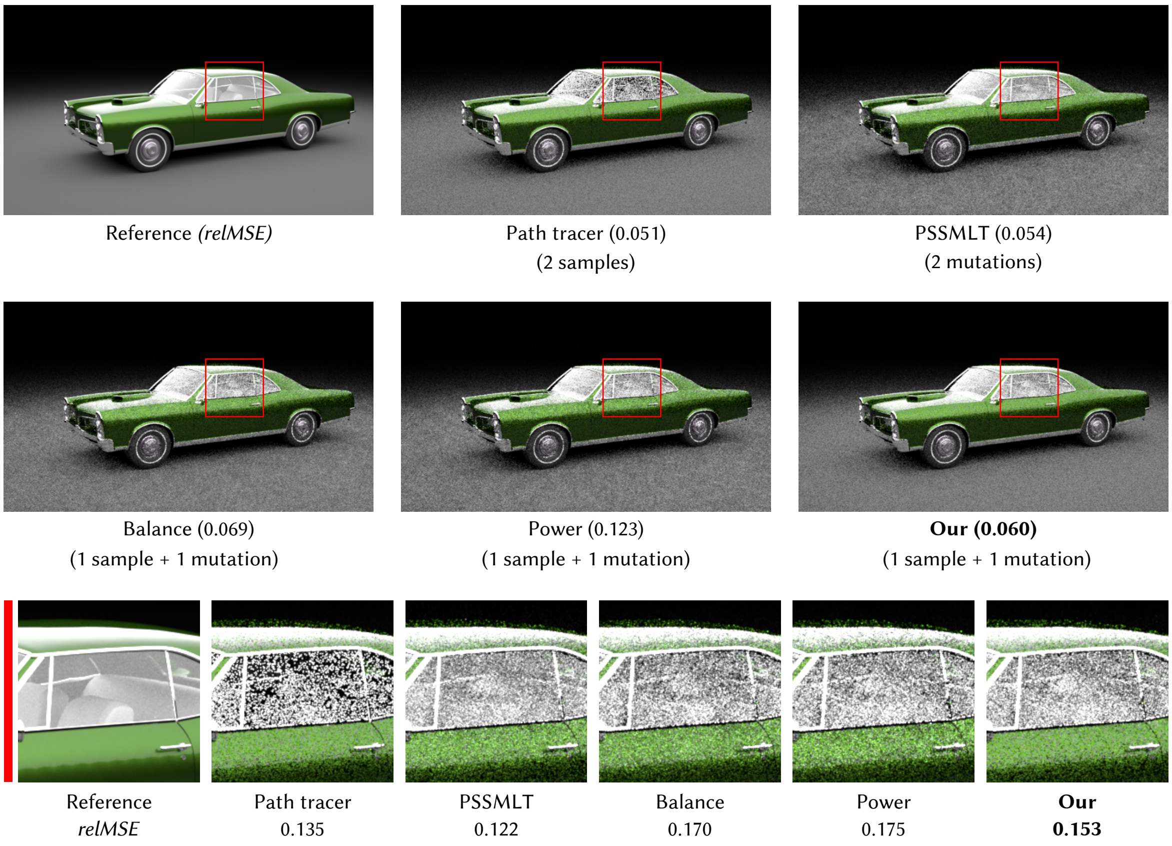


Figure 5.9: *Equal-sample comparison for the proof-of-concept Metropolis combination. The combination with our heuristic is the most robust: it retains the lower variance of both the Metropolis and the MC approach.*

the mixture of these pdfs [Veach 1997]. This mixture can then be combined with other sampling techniques using our variance-aware heuristic. An alternative is to simply set the variance factors to one for all narrow-support techniques.

5.4.2 Other applications

Efficiency. Our MIS weights help achieve robustness by preventing the noise from poorly performing techniques deteriorate the quality that could be achieved by using only the samples from the better techniques. However, for an algorithm to be truly efficient, better MIS weights are only the first step: Ideally, we also want to ensure that most samples are taken from the best-performing techniques. There have been attempts to improve the sample allocation based on the MIS-weighted contribution of previously taken samples [Grittmann et al. 2018; Hachisuka et al. 2014; Šik et al. 2016]. In combination with such approaches, our MIS weights could achieve a considerable increase in the efficiency of algorithms like vertex connection and merging (VCM) [Georgiev et al. 2012a].

VCM. Bidirectional path tracing, on top of which we apply our method, is a subset of the more powerful VCM algorithm [Georgiev et al. 2012a; Hachisuka et al. 2012]. Therefore, we can expect that the full VCM algorithm will benefit from similar improvements. Specifically, since photon mapping and light tracing for direct illumination are almost identical techniques, the direct-illumination results with VCM will be identical to those of BPT. It is possible, that our approach will improve on the VCM algorithm even further: Recent work [Jendersie 2019; Jendersie and Grosch 2018] has shown that the MIS weights for the vertex merging technique in VCM can perform poorly in some cases. Our method could also help there.

Metropolis light transport. Our observation that existing MIS heuristics neglect stratification has inspired us to experiment with a novel MIS combination: Combining a Metropolized path tracer with a regular Monte Carlo path tracer. Rendering methods based on the Metropolis algorithm are often criticized for their lack of image plane stratification [Cline et al. 2005; Šik and Křivánek 2018]. In a proof-of-concept experiment, we therefore tried combining a primary sample space Metropolis path tracer (PSSMLT) [Kelemen et al. 2002] with a stratified MC path tracer, using MIS.

An equal-time comparison is shown in Figure 5.9. The MC path tracer, thanks to the stratification, can easily handle the direct illumination in that scene, but struggles with the indirectly lit car interior. PSSMLT easily resolves that more challenging interior, but due to the lack of stratification exhibits much stronger noise in the direct illumination (e.g., on the car exterior and the floor). While the balance- and power-heuristic combinations retain the better performance on the car interior, their unawareness of image plane stratification results in higher levels of noise in the simpler direct illumination. Our approach retains the better performance of both techniques, achieving a more efficient combination.

The optimal weights [Kondapaneni et al. 2019] are not applicable to such a combination. With Metropolis methods, the exact sampling densities are unknown and require approximations for use in MIS [Kelemen et al. 2002; Šik et al. 2016]. Therefore, computing the optimal MIS weights is impossible – a heuristic like ours is needed for such a combination to perform well.

5.5 Conclusion

We propose a novel weighting heuristic for multiple importance sampling that incorporates variance estimates. We show that existing MIS heuristics neglect the impact of stratification, correlation, and other effects on the variance – a shortcoming that can be addressed by incorporating variance estimates. We apply our theory to bidirectional path tracing, defensive sampling, and a proof-of-concept combination involving Metropolis sampling. Throughout all our tests, even coarse estimates of the variance have been sufficient to achieve significant improvements over the balance heuristic in some cases, while – most importantly – never performing worse.

Chapter 6

Correlation-aware MIS

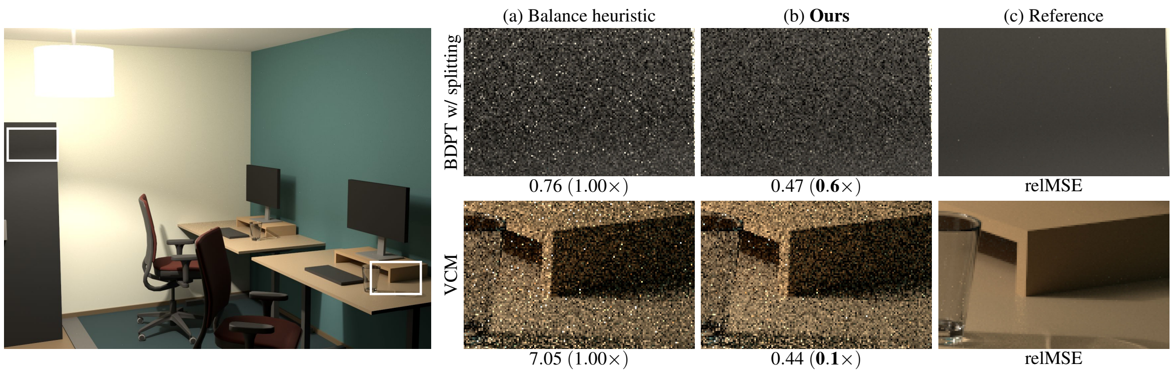
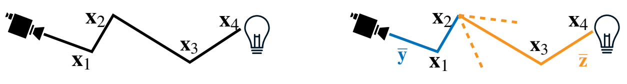


Figure 6.1: *A scene featuring complex indirect illumination (lamp shade) and caustics (glass) – a prime use-case of bidirectional algorithms. We show two such methods: bidirectional path tracing with splitting (top row), and vertex connection and merging (bottom row). (a) Both exhibit problems with MIS in this scenario, due to correlation by shared path prefixes. (b) Our simple heuristic solves these problems.*

Some rendering algorithms achieve efficiency by generating correlated paths. They reduce the sampling cost by constructing paths that share a common prefix. This is the overarching idea behind photon mapping [Jensen 1996], path splitting [J. R. Arvo and Kirk 1990; Rath et al. 2022] or distribution ray tracing [Cook et al. 1984], and path reuse methods [Keller et al. 2014; West et al. 2020]. Many of these approaches also make use of MIS to increase robustness, for example, vertex connection and merging (VCM) [Georgiev et al. 2012a; Hachisuka et al. 2012], or the method of Popov et al. [2015] which utilizes splitting in bidirectional path tracing (BDPT) [Veach and Guibas 1995a] by tracing multiple shadow rays from the same point. However, MIS and the popular balance heuristic operate under the assumption of independent sampling, that is, no correlation between individual paths. Lacking a practical alternative, the aforementioned algorithms still rely on the balance heuristic. This is problematic as illustrated in Figure 6.1: The heuristic produces poor combination weights, creating an overly dark image with strong outliers. Given a splitting technique that concatenates a single prefix with n suffix paths, the balance heuristic treats these as n mutually independent full paths; it is oblivious to the increased variance resulting from using a shared prefix.

The impact of correlation on MIS has received little attention in prior work. Even the recently derived optimal weights [Kondapaneni et al. 2019] (see Section 3.2.2.3) assume mutually independent paths and samples. For the two examples in Figure 6.1, ad-hoc solutions have been proposed [Jendersie 2019; Jendersie and Grosch 2018; Popov et al. 2015].



(a) A single path $\overline{\mathbf{x}}$ (b) One of 3 paths formed via splitting

Figure 6.2: *Without splitting (a), we trace independent full paths. Splitting (b) generates multiple full paths $\overline{\mathrm{x}}_i = \overline{\mathrm{y}\mathrm{z}}_i$ that share a prefix $\overline{\mathrm{y}}$ (in blue) and have mutually independent suffixes $\overline{\mathrm{z}}_i$ (in orange).*

Besides being specific to one of the two cases, these approaches can produce unsatisfactory results or rely on unintuitive parameters. A more general and effective solution would be to incorporate variance estimates (see Chapter 5), but as we will discuss later, that approach is neither robust nor efficient enough to be practical in the two cases of Figure 6.1.

In this chapter, we propose a simple and effective heuristic as a remedy that is based solely on sampling densities. Even without proof of optimality, the empirical evaluation of our heuristic shows consistent improvement over the balance heuristic and prior work across all our test scenes. Implementing our heuristic is straightforward as it relies on the same quantities as the balance heuristic. It also adds no noteworthy computation or memory overhead.

The method has been previously published [Grittmann et al. 2021], I was the main author of that paper. The source code can be found on GitHub[1]. It allows exact reproduction of the figures throughout this chapter.

6.1 The problem: path correlation through splitting

The sampling techniques in bidirectional algorithms each generate full paths. Specifically, for the path integral of length k, each technique t samples n_t paths $\overline{\mathrm{x}} = x_1 \dots x_k$ distributed according to $p_t(\overline{\mathrm{x}})$. As discussed in Section 4.2.2, these full paths are generally not sampled independently of each other. In this chapter, we focus on ameliorating the adverse effects of one particularly strong kind of correlation: the correlation due to splitting and path reuse.

Splitting, and the conceptually identical reuse and merging operations, forms n full samples $\overline{\mathrm{x}}$ by combining a single prefix $\overline{\mathrm{y}}$ with n suffixes $\overline{\mathrm{z}}$, as illustrated in Figure 6.2. Splitting estimators take the form

$$\langle F \rangle_{\text{split}} = \sum_{i=1}^{n} \frac{f(\overline{\mathrm{y}\mathrm{z}}_i)}{np(\overline{\mathrm{y}\mathrm{z}}_i)}. \tag{6.1}$$

Due to the shared prefix $\overline{\mathrm{y}}$, the paths $\overline{\mathrm{x}}_i$ are no longer mutually independent – they are correlated.

The variance $\mathrm{V}[\langle F \rangle_{\text{split}}]$ of the splitting estimator can be expressed as a weighted sum of the variance $V_{\overline{\mathrm{y}}}$ due to the prefix and the variance $V_{\overline{\mathrm{z}}}$ due to the suffix [Bolin and Meyer 1997]. The prefix variance,

$$V_{\overline{\mathrm{y}}} := \mathrm{V}\left[\frac{F_{\overline{\mathrm{z}}}(\overline{\mathrm{y}})}{p(\overline{\mathrm{y}})}\right] = \int_X \frac{F_{\overline{\mathrm{z}}}^2(\overline{\mathrm{y}})}{p(\overline{\mathrm{y}})} \,\mathrm{d}\overline{\mathrm{y}} - F^2, \tag{6.2}$$

[1]https://github.com/pgrit/MisForCorrelatedBidir

is the variance of a hypothetical estimator that knows the exact integral

$$F_{\overline{\mathrm{z}}}(\overline{\mathrm{y}}) = \int_{\mathcal{X}} f(\overline{\mathrm{y}\mathrm{z}})\, \mathrm{d}\overline{\mathrm{z}} \tag{6.3}$$

over all suffix paths. The suffix variance is the expectation of the variance of a primary estimator $\langle F \rangle_1$ over all possible prefixes $\overline{\mathrm{y}}$:

$$V_{\overline{\mathrm{z}}} := \mathrm{E}\left[\mathrm{V}\left[\langle F \rangle_1 \mid \overline{\mathrm{y}}\right]\right] = \int_{\mathcal{X}} \frac{f^2(\overline{\mathrm{x}})}{p(\overline{\mathrm{x}})}\, \mathrm{d}\overline{\mathrm{x}} - \int_{\mathcal{X}} \frac{F_{\overline{\mathrm{z}}}^2(\overline{\mathrm{y}})}{p(\overline{\mathrm{y}})}\, \mathrm{d}\overline{\mathrm{y}}. \tag{6.4}$$

While with independent sampling the total variance is inversely proportional to the sample count, that is, $\mathrm{V}[\langle F \rangle_n] = \frac{1}{n}\left(V_{\overline{\mathrm{y}}} + V_{\overline{\mathrm{z}}}\right)$, splitting reduces only the suffix variance,

$$\mathrm{V}[\langle F \rangle_{\mathrm{split}}] = V_{\overline{\mathrm{y}}} + \frac{1}{n} V_{\overline{\mathrm{z}}}. \tag{6.5}$$

Splitting can still be an efficient strategy, particularly when $V_{\overline{\mathrm{y}}}$ is low, for example, due to highly glossy bounces in the prefix $\overline{\mathrm{y}}$, or if the cost of sampling the prefix is high [Bolin and Meyer 1997; Rath et al. 2022].

Expanding the terms in Equation (6.5), we can write the variance in a form similar to that of independent sampling (see Section 2.3.2):

$$V[\langle F \rangle_{\mathrm{split}}] = \int_{\mathcal{X}} \frac{f^2(\overline{\mathrm{x}})}{n p(\overline{\mathrm{x}})}\, \mathrm{d}\overline{\mathrm{x}} + \frac{n-1}{n} \int_{\mathcal{X}} \frac{F_{\overline{\mathrm{z}}}^2(\overline{\mathrm{y}})}{p(\overline{\mathrm{y}})}\, \mathrm{d}\overline{\mathrm{y}} - F^2. \tag{6.6}$$

Compared to the variance of independent sampling, the value is increased by the second integral term – the covariance. Intuitively, the impact of the correlation depends on the value of $p(\overline{\mathrm{y}})$ compared to the full-path density $p(\overline{\mathrm{x}})$. This insight is a key motivation behind our heuristic.

6.2 Correlation-aware balance heuristic

The core idea of our heuristic is the same as the variance-aware weights: We modify the balance heuristic by multiplying each term with an extra correction factor c_t,

$$w_{\mathrm{correl},t}(\overline{\mathrm{x}}) = \frac{c_t n_t p_t(\overline{\mathrm{x}})}{\sum_{t'} c_{t'} n_{t'} p_{t'}(\overline{\mathrm{x}})}. \tag{6.7}$$

This correction factor is constructed such that $c_t \approx 1$ in all cases where the balance heuristic performs satisfactorily.

Mathematically, our goal is to correct the actual sample count n_t to the effective sample count $n_{\mathrm{eff},t}$ (see Section 2.3.4.1),

$$c_t = \frac{n_{\mathrm{eff},t}}{n_t} = \frac{1}{n_t} \frac{V_{\overline{\mathrm{y}}} + V_{\overline{\mathrm{z}}}}{V_{\overline{\mathrm{y}}} + \frac{1}{n_t} V_{\overline{\mathrm{z}}}}. \tag{6.8}$$

The variance factors from the previous chapter would satisfy this objective, since

$$v_t = \frac{\int \frac{f(x)}{n_t p(x)}\, \mathrm{d}x}{V_{\overline{\mathrm{y}}} + \frac{1}{n} V_{\overline{\mathrm{z}}}} \approx \frac{1}{n_t} \frac{V_{\overline{\mathrm{y}}} + V_{\overline{\mathrm{z}}}}{V_{\overline{\mathrm{y}}} + \frac{1}{n_t} V_{\overline{\mathrm{z}}}}. \tag{6.9}$$

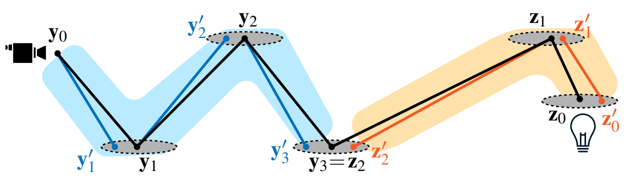


Figure 6.3: *Given a path $\overline{\mathrm{x}} = \overline{\mathrm{yz}}$ (in black), we want to compute the probability of sampling a similar path, i.e., one falling within the shaded region. We compute the probability that each edge would produce a vertex x_i' that falls within a disc around the actual vertex x_i. The blue and orange lines visualize how these are computed for the camera prefix and the light suffix, respectively.*

However, unlike the low-variance examples discussed in the previous chapter, these ratios no longer have the convenient lower bound of one. Therefore, they need to be computed for all (correlated) techniques at all path lengths. At unbounded path lengths, this would not even be possible at all, but even for moderate maximum lengths such as $k < 10$, the memory cost would be severe, as the number of techniques is quadratic in the path length.

Our solution is to construct a heuristic that approximates the above ratio without resorting to auxiliary storage. For that, we compare the sampling densities of the prefix $p(\overline{\mathrm{y}})$ and the suffix $p(\overline{\mathrm{z}})$. While this may sound simple, such a comparison has to be done with great care, as the two densities are generally with respect to two different measures, as the prefix and suffix paths do not have the same number of vertices. A direct comparison would not be meaningful, so we have to first compute a representative unitless quantity for each.

6.2.1 Computing a unitless path probability

The idea is illustrated in Figure 6.3. The figure depicts a full path $\overline{\mathrm{x}}$, composed of a prefix $\overline{\mathrm{y}}$ and a suffix $\overline{\mathrm{z}}$. We approximate the probability that another prefix $\overline{\mathrm{y}}'$ or suffix $\overline{\mathrm{z}}'$, sampled from the same PDF, would wind up within the shaded blue / orange region. For that, we approximate the probability that each vertex y_i' lands within some disc of radius r around the current y_i.

Let $P(y_{i-1} \to y_i' \in D_r(y_i))$ denote the probability that another vertex y_i', sampled from y_{i-1}, lands within the disc neighborhood $D_r(y_i)$ around the current vertex y_i. Then, we define our unitless probability measure to be the product of these probabilities, for all edges along the prefix:

$$P(\overline{\mathrm{y}}) = \prod_i P(y_{i-1} \to y_i' \in D_r(y_i)) \tag{6.10}$$

Similarly, we define the suffix probability,

$$P(\overline{\mathrm{z}}) = P(z_0' \in D_r(z_0)) \prod_i P(z_{i-1} \to z_i' \in D_r(z_i)), \tag{6.11}$$

with the only difference that we also consider the probability of sampling an initial vertex on the light source within the disc neighborhood. We did not consider the initial vertex along the camera prefix, assuming that sampling the lens of the camera does not add significant correlation. For complex lens systems, that might have to be adapted.

Computing the exact values for $P(\overline{\mathrm{y}})$ or $P(\overline{\mathrm{z}})$ above would require costly integration and defeat the purpose of our heuristic. Instead, we use the cheap approximation used to define the surrogate PDF for the merging technique during MIS weighting [Georgiev et al. 2012a] (see Section 4.1.2.4):

$$P(y_{i-1} \to y'_i \in D_r(y_i)) = \int_{D_r} p(y_{i-1} \to y'_i)\,\mathrm{d}y'_i \tag{6.12}$$

$$\approx \min\left\{p(y_{i-1} \to y_i)\pi r^2, 1\right\} \tag{6.13}$$

Where equality holds if the PDF $p(y_{i-1} \to y_i)$ is constant within the disc neighborhood D_r. Clamping the result to one ensures that we obtain a valid discrete probability $P(y_{i-1} \to y'_i \in D_r(y_i)) \in [0, 1]$. The clamping is required whenever the majority of the mass of the PDF is focused on a smaller region than the disc D_r. For example, for near-specular scattering with BSDF importance sampling. Such clamping was not needed for the MIS weighting surrogate, as the merging radius is typically much smaller than the radius we use for our path probability approximation.

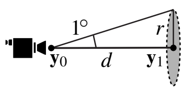
In essence, we compute a unitless path probability by multiplying each edge's sampling PDF by πr^2 and clamping to one. The product of these is then the full sub-path probability.

6.2.2 Choosing a radius

What remains is setting the radius r in Equation (6.12). The problem is similar to choosing a good radius for photon mapping: Finding the best possible value is difficult, but simple heuristics can go a long way. While in photon mapping the radius trades bias for variance, we have different considerations to make. If the radius is too large, the probabilities will all be close to one. This would make $P(\overline{\mathrm{y}}) \approx P(\overline{\mathrm{z}}) \approx 1$, and we revert to the balance heuristic. If, on the other hand, the radius is too small, we penalize glossy bounces too much.

We could set the radius as a fraction of the scene extent; however, this approach is not robust, for example, when the camera sees only a small part of a large scene. An alternative is to use pixel footprints [Šik et al. 2016]. The resulting radius then depends on the image resolution and the camera field of view. But increasing the image resolution should not alter $P(\overline{\mathrm{x}})$ as it has little effect on the path sampling variance. Therefore, we replace the pixel footprint by a related quantity independent of the resolution or field of view – the footprint of a one-degree viewing angle:

$$r := d \tan \frac{\pi}{180} \approx 0.0175d, \tag{6.14}$$



where $d = \|\mathrm{y}_1 - \mathrm{y}_0\|$ is the distance to the hit point along the camera ray, as sketched on the right above.

6.2.3 Constructing our heuristic

With the definitions above, we can now construct our final heuristic. Again, the goal is to reduce the weight, whenever the density of the prefix is much lower than the density of the full path. To achieve that goal, we compute

$$\boxed{c_t(\overline{x}) = \max\left\{ \frac{P(\overline{y}_t)}{P(\overline{y}_t) + P(\overline{z}_t) - P(\overline{y}_t)P(\overline{z}_t)}, \frac{1}{n_t} \right\}.} \tag{6.15}$$

That is, we compare the probability of sampling a similar prefix to the union probability of sampling either a similar prefix or a similar suffix. We clamp the result to the known lower bound of $1/n_t$ to increase accuracy for cases with small sample counts. For the merging technique, where n_t is in the millions, this clamping has no noticable effect, but it avoids overshooting in the application to multiple connections, where n_t is relatively small.

This ratio of probabilities was carefully chosen to satisfy the limiting behaviours of the correction factor that we desire. First, if the prefix probability is high, then correlation is likely small and the weight should not be changed, no matter what the suffix probability is. Indeed, in that case the limit

$$\lim_{P(\overline{y}_t)\to 1} c_t(\overline{x}) = 1, \tag{6.16}$$

and we retain the balance heuristic weight. Second, if the prefix probability is low, then correlation is likely strong and the weight should be reduced. There, our heuristic approaches zero, which will be clamped to the known lower bound of $1/n_t$

$$\lim_{P(\overline{y}_t)\to 0} c_t(\overline{x}) = \frac{1}{n_t}, \tag{6.17}$$

unless the third scenario occurs, where both subpaths have low probability. This is an indicator that the overall variance is high, but not dominated by the covariance. Then,

$$\lim_{\substack{P(\overline{y}_t)\to 0 \\ P(\overline{z}_t)\to 0}} c_t(\overline{x}) = 1 \tag{6.18}$$

and the balance heuristic weighting is still retained.

The second limit is required to achieve the desired improvements, while the first and last limits ensure that the balance heuristic is not adversely affected in cases where correlation is low. Other combinations of these probabilities are possible, like the much more aggressive $c_t = P(\overline{y}_t)$ that only fulfills the first two limits. These alternatives achieve similar speed-ups in the failure cases, but can result in poorer weighting when correlation is low and hence did not yield consistent improvements over the balance heuristic in our tests.

6.3 Evaluation

We have applied our correlation-aware balance heuristic (6.7) to two rendering algorithms that sample correlated paths: vertex connection and merging (VCM) and bidirectional path tracing (BDPT) with splitting. In both methods, evaluating the correction factor (6.15) is straightforward as it relies only on densities that the classical balance heuristic already requires. We implemented the methods using the SeeSharp rendering framework. The source code of our experiments is available on GitHub.

Our heuristic does not add any measurable overhead over the balance heuristic. In the results presented below, we use the same set of path samples when comparing the various MIS combinations on each scene. Thus, any differences between the images are solely due to differences in the weighting.

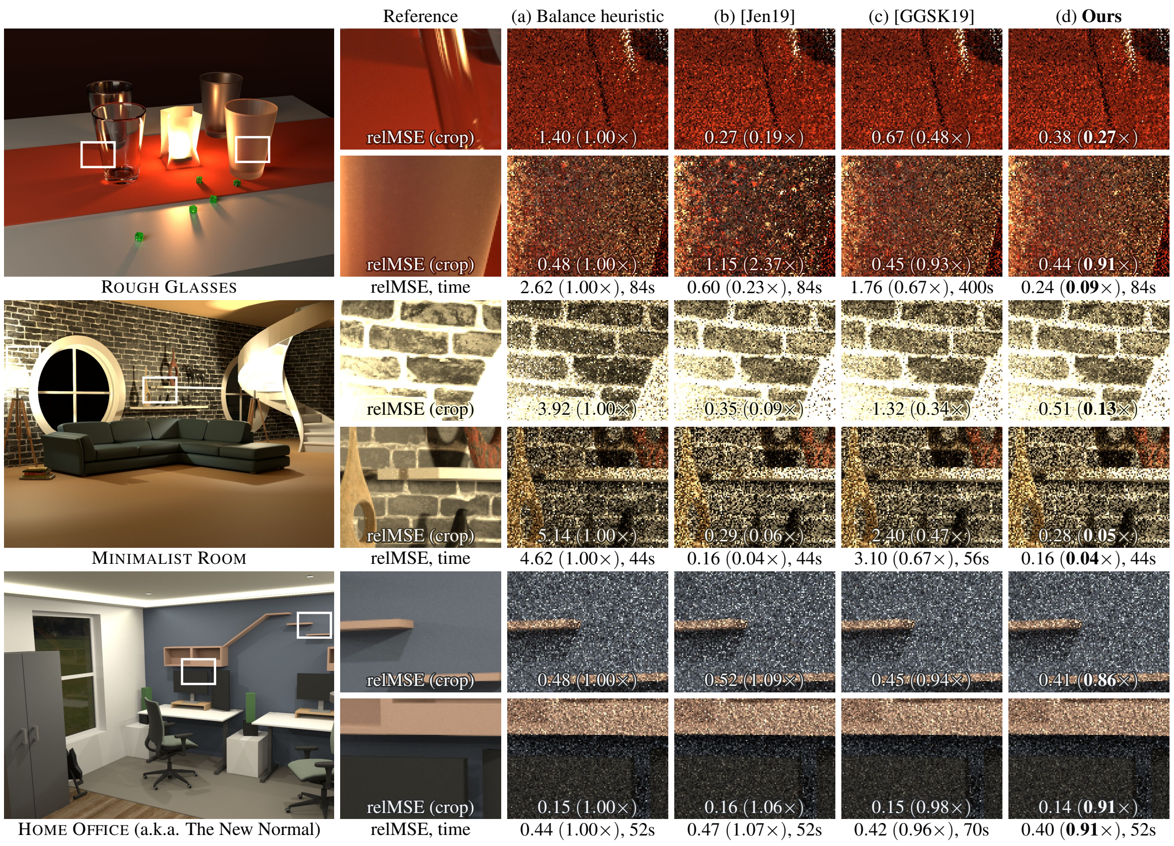


Figure 6.4: *Equal-sample comparison of different MIS heuristics for VCM (ours in bold). We provide error values (relMSE) per crop and over the entire image. The values in parentheses are relative to the balance heuristic (lower is better). In contrast to previous work, our heuristic is consistently better than the balance heuristic, providing significant error reduction in failure cases and slight improvement otherwise.*

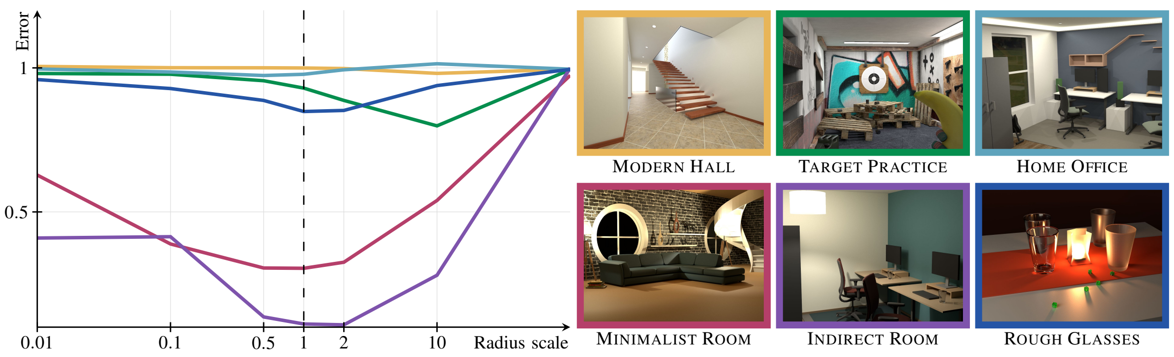


Figure 6.5: *Error relative to the balance heuristic (lower is better) when using different radii for our heuristic. Each line corresponds to one scene. The values on the x-axis are scaling factors applied to the radius computed using Equation* (6.14)*. Our default choice (dashed line) is close to the optimal for all scenes except* TARGET PRACTICE*. That scene features uniform illumination where diffuse bounces from the camera add little variance. A larger radius penalizes such bounces less and improves that specific result, but performs notably worse on all other scenes.*

6.3.1 Vertex connection and merging

Our VCM implementation follows the original formulation [Georgiev et al. 2012a; Hachisuka et al. 2012], with the exception that we forego merging at the second camera vertex, y_1. This technique produces an image identical to light tracing but with added bias due to blurring. We compare our heuristic (6.7) to the classical balance heuristic, the variance-aware balance heuristic [Grittmann et al. 2019] (see Chapter 5), as well as the approach of Jendersie [2019] (which supersedes that of Jendersie and Grosch [2018]).

We show results on three scenes in Figure 6.4. On Rough Glasses and Minimalist Room, the balance heuristic (a) suffers from the same failure as in Figure 4.9; namely, merging on the surface close to the light receives far too high weight. On the Home Office scene, it works well despite the indirect illumination around the walls close to the ceiling: the light source is large, so the variance is high, the density is low, and merging on the ceiling is beneficial.

The weighting scheme of Jendersie [2019] (b) eliminates the outliers in the top two scenes. Unfortunately, on the Rough Glasses scene, which features surfaces with varying roughness, it also deteriorates the quality of glossy reflections compared to the balance heuristic (bottom zoom-ins). This limits the utility of that weighting scheme, since glossy reflections of caustics are a key strength of VCM. The scheme also performs somewhat worse than the balance heuristic on the Home Office scene.

More consistent improvements over the balance heuristic can be achieved with variance-aware weighting (c) [Grittmann et al. 2019]. However, the variance estimates this method uses are inaccurate, so it struggles at removing outliers completely, as seen in the Minimalist Room scene. Additionally, the high computational overhead of variance estimation becomes a concern at larger path lengths. For the Minimalist Room and Home Office scenes we cap the path length to five, which is somewhat expensive but still manageable for the variance-aware scheme. However, highly glossy scenes require simulating much longer paths. With up to ten bounces on the Rough Glasses scene, variance-aware weighting takes roughly $4.7\times$ longer to render than the balance heuristic or our approach.

Our heuristic (d) successfully eliminates outliers and retains glossy reflections at all roughness levels. It never performs worse than the balance heuristic in any of our test scenes. The only case where it does not deliver the best variance reduction is in the region shown in the top zoom-ins of the Minimalist Room scene. There, the distances and angles between the wall, lamp shade, and light source are similar, and they are all diffuse. The path density does not fully capture the differences in variance.

In Figure 6.5 we explore different choices for the radius parameter in our heuristic. We have found our default choice (6.14) to be close to optimal on most scenes. In very uniformly lit scenes, e.g. Target Practice, a larger radius produces slightly better results. There, diffuse bounces from the camera add little variance and should be penalized less. This is a limitation of our heuristic as in those cases a low prefix density does not indicate high variance.

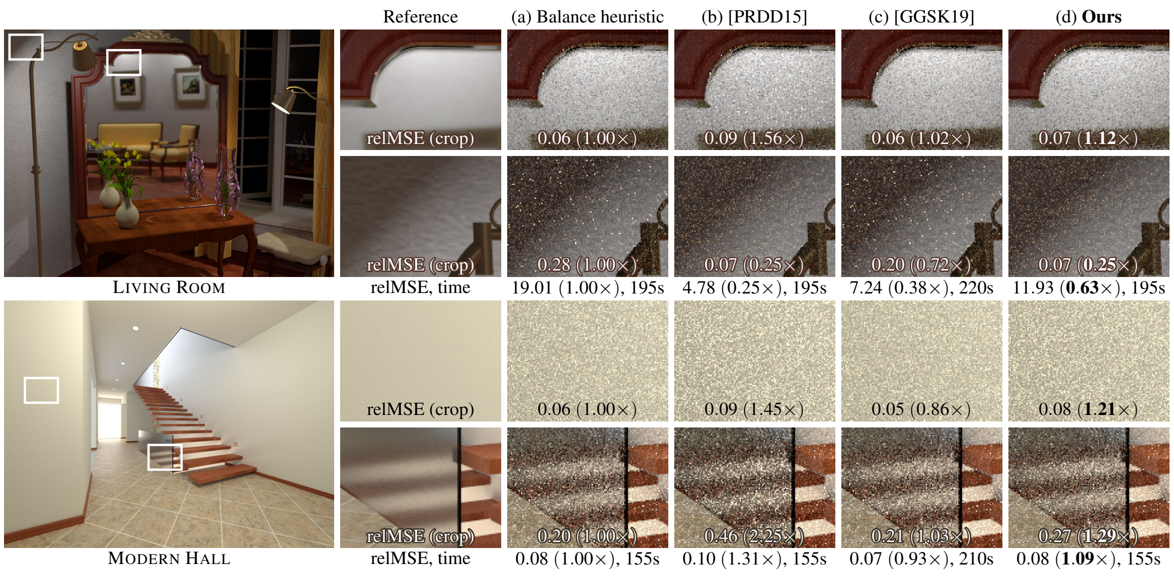


Figure 6.6: *Equal-sample comparison of different MIS heuristics for BDPT with 100 shadow rays for next-event estimation. Although the improvement is smaller in this case compared to VCM in Figure 6.4, our heuristic still performs best on average.*

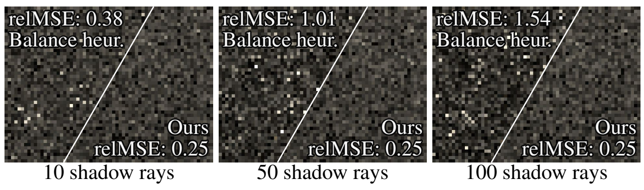


Figure 6.7: *Zoom-ins of the Indirect Room scene from Figure 6.1 rendered with different shadow-ray counts. With the balance heuristic, variance increases when more rays are used.*

6.3.2 Bidirectional path tracing

For our second application, we reproduce the correlation problem of BDPT with multiple connections per camera vertex [Popov et al. 2015] in a simple setting: we use multiple shadow rays for next-event estimation. In Figure 6.6 we compare our heuristic against setting $n = 1$ in the balance heuristic [Popov et al. 2015] and the variance-aware weights [Grittmann et al. 2019]. Again, we feed the same set of path samples to all methods. To highlight the problem and make noise more visible, we use 100 shadow rays per camera vertex. Lower counts produce similar results, only less pronounced. Figure 6.1 shows an example with 10 shadow rays, and Figure 6.7 compares different splitting factors. All full-size images are included in the supplemental materials of the original paper [Grittmann et al. 2021].

The balance heuristic produces outliers in the Living Room scene due to paths splitting on the lamp shade after bouncing off the diffuse wall (bottom zoom-ins). To eliminate these outliers, Popov et al. [Popov et al. 2015] set $n = 1$ in the balance heuristic. Unfortunately, doing so introduces new outliers in the mirror reflection (top zoom-ins). Camera prefixes that bounce off the highly glossy mirror do not increase variance, hence the weight of such paths should not be reduced. The presence of outliers makes the variance estimates in the variance-aware weighting unreliable, preventing it from improving noticeably over the balance heuristic. Our method consistently improves on the balance heuristic across the entire image.

The Modern Hall scene poses an interesting challenge. Illuminated by numerous light sources from various directions, the variance in the camera prefix is low despite the diffuse interactions. Our heuristic captures most, though not all of that effect. This is the only scene where our heuristic is outperformed by the balance heuristic. Our result is still closer to the balance heuristic than the overly conservative approach of Popov et al. In this scene, the only method that manages to outperform the balance-heuristic are the more costly variance-aware weights.

6.4 Limitations and future work

The BDPT application shows the limitations of our simple heuristic. If a scene is dominated by uniform and diffuse illumination, as is the case in the Modern Hall, the sampling density is a poor indicator of the variance, as zero variance would be achieved with a rather uniform density. In other words, a low density can sometimes still yield low variance. By design, our heuristic does not reflect that. Nevertheless, it performs consistently better than the more aggressive solution of Popov et al., while remaining cheap to evaluate.

We have restricted our discussion to surface scattering from finite light sources; however, extending our heuristic to infinite lights or volumetric scattering [Křivánek et al. 2014] should be straightforward. All that is needed is an analogy of the disc approximation used to make the sampling densities unitless and comparable. Applying our ideas to integration problems beyond rendering could also be interesting. However, due to the complex ways correlation affects variance, it is questionable whether a general *and* practical solution is possible.

We do not have hard proofs about the optimality of our heuristic. The result is bounded by the upper bound of Popov et al. [2015], and we only deviate from that if we have reason to believe the correlation does not increase variance. While our empirical results might be sufficient evidence to warrant the use in practice, it is still worthwhile to look for additional guarantees, or alternative heuristics based on a more rigorous mathematical derivation.

Multiple importance sampling in the context of VCM also ignores another important aspect: the bias due to merging. Accounting for this bias in the MIS weights could help reduce artifacts. In that direction, it would be interesting to augment the variance-aware weights by bias estimates [Hachisuka et al. 2010], or to make our simpler heuristic bias-aware.

Lastly, having more robust MIS weights opens up possibilities for more creative correlated rendering algorithms. Some prior work has avoided correlation, to also avoid issues with MIS [Nabata et al. 2020]. Allowing correlation and utilizing our heuristic could provide additional improvements to such methods. Furthermore, there has been some work on guiding photons based on MIS-weighted contributions [Grittmann et al. 2018; Šik and Křivánek 2019]. The efficiency and/or robustness of such applications could also benefit from our heuristic. The results in the next chapter are encouraging for that: There, we show that using correlation-aware MIS weights helps to optimize the number of light paths based solely on the far cheaper second moment rather than the full variance (see Section 7.5.1.3).

6.5 Conclusion

We propose a simple heuristic to account for correlation due to path splitting in multiple importance sampling for light transport simulation. Our heuristic is efficient and relies on quantities that are already required by the balance heuristic. In contrast to prior work, our approach has no overly harmful effects on the technique combination in cases where the correlation has little or no impact on the variance. Implementing our heuristic takes little effort and introduces negligible overhead.

Chapter 7

Efficiency-aware MIS

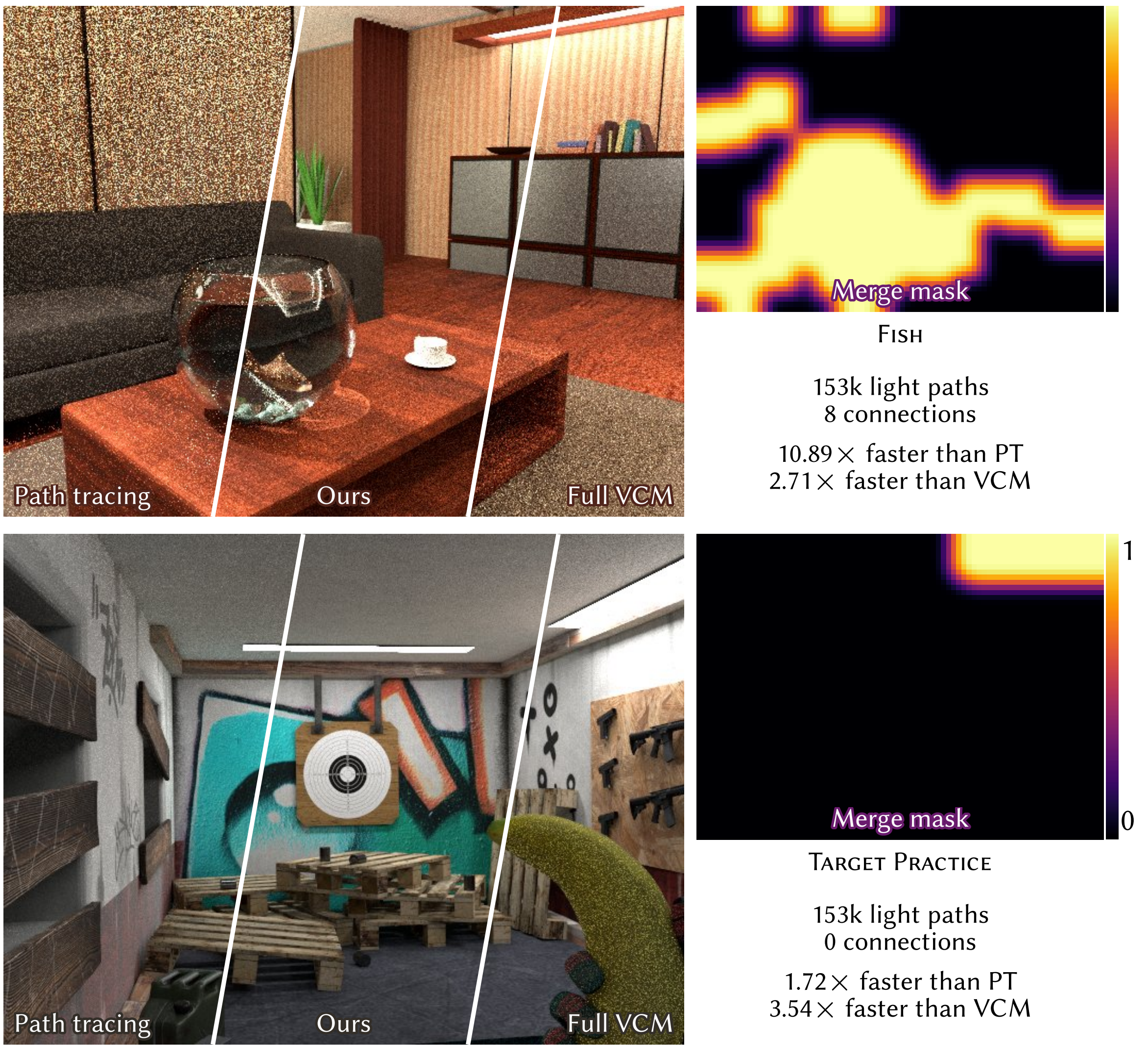


Figure 7.1: *Two scenes that are challenging to render efficiently without user guidance: The caustics and strong indirect illumination in the* Fish *scene (left) resolve much faster with a bidirectional method such as VCM. But the simpler* Target Practice *scene (right) renders 2× slower with VCM than with forward path tracing. Our method renders both scenes efficiently by automatically setting the number of light subpaths to trace, the number of bidirectional connections to make, and in which pixels to perform photon density estimation (the 'merge mask' specifies a probability for performing a photon lookup in a pixel).*

Comprehensive MIS combinations like the VCM algorithm [Georgiev et al. 2012a; Hachisuka et al. 2012] are robust but not efficient. They are robust, because they can render almost any scene in acceptable time. But they are not efficient, because the render time for any specific scene is not always the lowest it could be.

Robustness is achieved if every effect potentially present in a scene can be sampled well by at least one technique. The easiest way to achieve this robustness is by combining as many sampling techniques as possible. The VCM algorithm, for example, is robust because it combines all basic Monte Carlo path tracing techniques. Thus, for every illumination effect there is at least one technique in VCM that can sample it reasonably well.

However, typical scenes usually do not contain every possible illumination effect. For example, the Fish scene in Figure 7.1 is dominated by indirect illumination and features some reflected caustics, while the Target Practice scene is mostly directly illuminated. In both cases, many of the sampling techniques that comprise VCM are wasted. The Fish, on the one hand, only benefits from merging in some regions of the image and requires many bidirectional connections but fewer light subpaths or unidirectional samples. The Target Practice scene, on the other hand, only benefits from bidirectional samples in a small region around a light close to a surface. The most robust algorithms are often not the most efficient when tasked with one specific scene.

How can we achieve a combination that is both robust and efficient? Our approach is to adaptively select the set of sampling techniques and corresponding sample counts based on statistics of the scene at hand. We start rendering a scene in a pilot iteration with the cheapest strategy we have at our disposal: unidirectional path tracing. With the samples from this pilot iteration, we can estimate the efficiency of different MIS strategies and select the best one.

In this chapter, we present an efficient and numerically robust estimation scheme to compute the variances, second moments, or derivatives required to optimize the technique selection and sample allocation in an MIS combination. Further, we introduce a simple brute-force optimization scheme that can find good strategies even for complex algorithms such as VCM. The work presented in this chapter has been published before [Grittmann et al. 2022] and some text in the following is based on that paper, of which I am the main author. Source code of our implementation is available on GitHub[1].

Previous methods tackling the problem of sample allocation in MIS are discussed in Section 3.2.3. These fall short in the context of VCM, because they either cannot incorporate the vast cost differences or do not allow for weighting functions other than the balance heuristic. Further, numerically robust and efficient computation of the required terms for optimization remains challenging.

In the following, we first formulate the optimization objective in Section 7.1 and, in Section 7.3, show how to estimate the required terms. Then, Section 7.2.3 introduces our brute-force scheme. We conclude by discussing the results in a simple direct illumination context (Section 7.4) and for the VCM algorithm (Section 7.5).

[1]https://github.com/pgrit/EfficiencyAwareMIS

7.1 Problem statement

The discussion in the following focuses on multi-sample MIS estimators (see Chapter 3) that combine T sampling techniques:

$$\langle F \rangle_{\mathrm{n}} = \sum_{t=1}^{T} \sum_{i=1}^{n_t} w_t(x_{t,i}) \frac{f(x_{t,i})}{n_t p_t(x_{t,i})}. \tag{7.1}$$

Here, $x_{t,i}$ is the ith sample from technique t. We assume that the MIS weights are given by the extended balance heuristic (see Chapters 5 and 6),

$$w_t(x) = \frac{c_t(x) n_t p_t(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)}, \tag{7.2}$$

of which the classic balance, power, maximum, and cut-off heuristics [Veach and Guibas 1995b] are special cases. The vector of sample counts

$$\mathrm{n} = (n_1, \ldots, n_T) \tag{7.3}$$

is called the *sample-allocation strategy*, or strategy for short.

Our goal is to find the best such strategy. First, we quantify this based on the notion of *efficiency* and discuss the differences between optimizing per-pixel and per-image efficiency. Then, we derive the optimization objective, as well as a simplified version that replaces variances by second moments.

7.1.1 Efficiency

The efficiency ϵ of a multi-sample MIS estimator $\langle F \rangle_{\mathrm{n}}$ for an integral F is the reciprocal product of expected error and expected cost:

$$\epsilon(\mathrm{n}) = \left(\mathrm{V}[\langle F \rangle_{\mathrm{n}}] \mathrm{C}(\mathrm{n})\right)^{-1}. \tag{7.4}$$

Here, we assume unbiasedness, so the error is given by the variance $\mathrm{V}[\langle F \rangle_{\mathrm{n}}]$. The cost is typically a random variable, since it depends on, for example, the length of the sampled path. That is, $\mathrm{C}(\mathrm{n})$ denotes the expected value of the cost of all samples consumed by strategy n. Its exact definition is application specific, but in many common cases the cost is a linear function of the sample counts,

$$\mathrm{C}(\mathrm{n}) = \sum_{t=1}^{T} \gamma_t n_t, \tag{7.5}$$

where γ_t is the expected cost of a single sample from technique t compared to all other techniques. Note that for our purposes, the absolute value of the cost is irrelevant and $\mathrm{C}(\mathrm{n})$ can be modelled by a heuristic that is (approximately) proportional to the actual cost.

7.1.2 Per-pixel and per-image efficiency

Rendering applications estimate many integrals at the same time. The value of each pixel j is given by an integral F_j and the rendered image is a set of P such integrals, with P being the number of pixels. The efficiency in that case can be quantified in two different ways. One is the sum of per-pixel efficiencies,

$$\epsilon_{\text{pixel}}(\mathrm{n}) = \sum_{j=1}^{P} \left(\frac{\mathrm{V}[\langle F_j \rangle_{\mathrm{n}_j}]}{F_j^2} \mathrm{C}_j(\mathrm{n}_j) \right)^{-1}, \tag{7.6}$$

where n_j is the sample-allocation strategy used by the jth pixel. Dividing the pixel variance by the squared ground truth F_j is optional; the resulting relative variance accounts for the varying magnitudes of the individual integrals (i.e., bright and dark pixels). The alternative is the efficiency of the sum,

$$\epsilon_{\text{image}}(\mathrm{n}) = \left(\sum_{j=1}^{P} \frac{\mathrm{V}[\langle F_j \rangle_{\mathrm{n}_j}]}{F_j^2} \sum_{j=1}^{P} \mathrm{C}_j(\mathrm{n}_j) \right)^{-1}, \tag{7.7}$$

which is the same as the expected relative mean-squared error (relMSE) multiplied by the expected render time of the entire image. The latter is generally the more desirable objective as it is the metric typically used when evaluating rendering algorithms. The former, however, can be a better choice if an orthogonal method, like image space adaptive sampling, is already optimizing the distribution of error.

Maximizing the per-pixel efficiency ϵ_{pixel} is an easier objective, as the optimization can be done independently per-pixel. There is, however, no guarantee that the error will be uniformly distributed over the image. This can be problematic when used without an adaptive sampler in image space. Maximizing the per-image efficiency ϵ_{image} ensures a more uniform convergence of the image as a whole and does not rely on an adaptive sampler.

7.1.3 Per-pixel and per-image sample counts

The granularity at which the sample counts can be set differs between rendering algorithms. For instance, the number of light subpaths in bidirectional algorithms is necessarily a global constant over the whole image, while the number of shadow rays can be tuned on a per-pixel basis, or even spatially, depending on the position within the scene. In our application, we consider combinations of per-pixel and per-image sample counts.

We express the sample strategy n_j in pixel j as the concatenation of two vectors: the per-pixel sample counts p_j and the per-image counts i:

$$\mathrm{n}_j = (\mathrm{p}_j, \mathrm{i}). \tag{7.8}$$

Consequently, the per-pixel cost

$$\mathrm{C}_j(\mathrm{n}_j) = \mathrm{C}_j(\mathrm{p}_j) + \frac{1}{P}\mathrm{C}_j(\mathrm{i}) \tag{7.9}$$

is the sum of the cost of all per-pixel samples, $\mathrm{C}_j(\mathrm{p}_j)$, and the cost of the per-image samples $\mathrm{C}_j(\mathrm{i})$. The latter is divided by the number of pixels P to account for the amortization of the cost across all pixels.

7.1.4 Second moment

All previous methods have simplified the problem by replacing the variance $\mathrm{V}[\langle F\rangle_{\mathrm{n}}]$ by the second moment $\mathrm{M}[\langle F\rangle_{\mathrm{n}}]$. While our brute-force optimization supports the full variance, the second moment is *much* cheaper to estimate.

The variance of a multi-sample MIS estimator has the form

$$\mathrm{V}[\langle F\rangle_{\mathrm{n}}] = \mathrm{M}[\langle F\rangle_{\mathrm{n}}] - r_{\mathrm{n}}, \tag{7.10}$$

where r_{n} is a residual term (see Section 2.3.3). The second moment, when using the extended balance heuristic is:

$$\mathrm{M}[\langle F\rangle_{\mathrm{n}}] = \int_{\mathcal{X}} \frac{f^2(x)\sum_t w_{\mathrm{n},t}(x)c_t(x)}{\sum_t c_t(x)n_t p_t(x)}\,\mathrm{d}x. \tag{7.11}$$

As discussed in Section 2.3.3, the second moment is a close approximation of the variance in many cases, with sample correlation causing an important exception. We found that restricting ourselves to the second moment produces good results even with correlated techniques like photon mapping, as long as the correlation-aware or variance-aware MIS weights are used at the same time. Our brute-force optimizer, introduced in the next section, can trivially support covariance or full variance, but finding an efficient estimation or approximation scheme for the required terms remains an open problem.

7.2 Optimization

The strategy n that maximizes the efficiency can be found in two ways: via convex optimization, as done by previous work, or via an (exhaustive) search. In the following, we first formulate the optimization objective and discuss how to handle combinations of per-pixel and per-image sample counts. Based on this formulation, we then show how to extend previous convex optimization approaches to support diverse sample cost and per-image sample counts. Lastly, we introduce our brute-force search scheme that supports the extended balance heuristic (7.2), full variance, and arbitrary cost functions.

7.2.1 Objective and algorithm

In general terms, we set out to find the sample allocation strategy n minimizing the work-normalized second moment,

$$\mathrm{n} = \arg\min_{\mathrm{n}} \mathrm{M}[\langle F\rangle_{\mathrm{n}}]\mathrm{C}(\mathrm{n}) \approx \arg\max_{\mathrm{n}} \epsilon(\mathrm{n}), \tag{7.12}$$

which approximately maximizes the efficiency. The approximation error depends on how closely the second moment M approximates the variance.

We handle per-pixel and per-image techniques by a two-step approach. First, per-pixel counts p_j are found for each pixel, by approximately maximizing the sum of efficiencies

$$\mathrm{p}_j = \arg\min_{\mathrm{p}_j} \mathrm{M}[\langle F\rangle_{(\mathrm{p}_j,\mathrm{i})}]\mathrm{C}(\mathrm{p}_j,\mathrm{i}) \approx \arg\max_{\mathrm{p}_j} \epsilon_{\mathrm{pixel}}(\mathrm{n}_j), \tag{7.13}$$

which can be done independently on a pixel-by-pixel basis. The per-image counts i are left as a free variable, that is, each pixel picks an optimal combination of per-pixel and per-image counts, but the per-image values are ignored.

With the per-pixel counts fixed, the second step is to find the per-image counts by approximately maximizing the efficiency of the sum

$$\mathrm{i} = \arg\min_{\mathrm{i}} \sum_{j=1}^{P} \frac{\mathrm{M}[\langle F_j \rangle_{\mathrm{n}_j}]}{F_j^2} \sum_{j=1}^{P} \mathrm{C}_j(\mathrm{n}_j) \approx \arg\max_{\mathrm{i}} \epsilon_{\mathrm{image}}(\mathrm{n}). \tag{7.14}$$

The second step heeds the per-pixel counts p_j determined by the first step.

In principle, it is possible to directly maximize $\epsilon_{\mathrm{image}}$ and find the optimal per-pixel and per-image counts. By separating the problem into two steps, we reduce the size of the search space and make the optimization more efficient. However, the local optimum that is found by this approach is not necessarily a global optimum. In practice, this can be compensated by combining our method with image-space adaptive sampling.

In the following two sections, we show how the individual optimization problems in Equations (7.13) and (7.14) can be solved via convex optimization or searching.

7.2.2 Convex optimization

Convex optimization is only possible if we impose additional assumptions. Naturally, the objective has to be convex, which only holds for linear cost functions and the classic balance heuristic. Furthermore, the range of possible values for the individual sample counts has to be large enough such that quantization error is negligible. In the following, we explain the procedure on the example of Equation (7.14), the per-image sample count optimization.

If the cost is linear in the sample counts, $\mathrm{C}(\mathrm{n}) = \sum_t \gamma_t n_t$, we can perform a change of variables to turn the generally non-convex objective (7.14) into a convex one. To that end, we define the relative computation time invested in technique t

$$r_t = \frac{\gamma_t n_t}{\sum_{t'} \gamma_{t'} n_{t'}} \in [0, 1] \tag{7.15}$$

and substitute it into (7.14), together with our assumption of a linear cost and the classic balance heuristic, to obtain a convex objective:

$$\sum_{j=1}^{P} \frac{\mathrm{M}[\langle F_j \rangle_{\mathrm{n}}]}{F_j^2} \sum_{j=1}^{P} \mathrm{C}_j(\mathrm{n}) = \sum_{j=1}^{P} \frac{\sum_t \gamma_t n_t}{F_j^2} \int_X \frac{f^2(x)}{\sum_t n_t p_t(x)} \, \mathrm{d}x \tag{7.16a}$$

$$= \sum_{j=1}^{P} \frac{1}{F_j^2} \int_X \frac{f^2(x)}{\sum_t r_t \frac{p_t(x)}{\gamma_t}} \, \mathrm{d}x =: W(\mathrm{r}). \tag{7.16b}$$

For simplicity, but without loss of generality, we assume that all sample counts are per-image. The per-pixel counts would appear as additional constants that do not change the structure of the problem.

The resulting $W(\mathrm{r})$ is a convex function of the sampling ratio vector $\mathrm{r} = (r_1, \dots, r_T)$ over the domain $[0,1]^T$ [Sbert et al. 2019]. The constraint that the ratios must sum to one, $\sum_t r_t = 1$, can be easily incorporated by replacing the last ratio $r_T = 1 - \sum_{t<T-1} r_t$ by one minus the sum of all others [Murray et al. 2020; Sbert et al. 2019]. Then, the partial derivatives are:

$$\frac{\mathrm{d}}{\mathrm{d}r_t} W(\mathrm{r}) = \sum_{j=1}^{P} \frac{1}{F_j^2} \int_{\mathcal{X}} \frac{f^2(x)}{\left(\sum_t r_t \frac{p_t(x)}{\gamma_t}\right)^2} \left(\frac{p_T(x)}{\gamma_T} - \frac{p_t(x)}{\gamma_t}\right) \mathrm{d}x \tag{7.17a}$$

$$\frac{\mathrm{d}^2}{\mathrm{d}{r_t}^2} W(\mathrm{r}) = 2 \sum_{j=1}^{P} \frac{1}{F_j^2} \int_{\mathcal{X}} \frac{f^2(x)}{\left(\sum_t r_t \frac{p_t(x)}{\gamma_t}\right)^3} \left(\frac{p_T(x)}{\gamma_T} - \frac{p_t(x)}{\gamma_t}\right)^2 \mathrm{d}x. \tag{7.17b}$$

We can apply the same Newton-Raphson root-finding used by previous work [Murray et al. 2020; Sbert et al. 2019] and find the optimal r by iteratively updating

$$\mathrm{r}_n = \mathrm{r}_{n-1} - \mathcal{H}_r^{-1}[W](\mathrm{r}_{n-1}) \nabla_{\mathrm{r}} W(\mathrm{r}_{n-1}), \tag{7.18}$$

where $\mathcal{H}_{\mathrm{r}}^{-1}[W]$ and $\nabla_{\mathrm{r}} W$ are the inverse Hessian and the gradient, respectively. Alternatively, we can also use a slower but simpler first-order gradient descent [Müller 2019; Rath et al. 2020], or use a diagonal approximation of the Hessian to avoid costly matrix inversion and reduce the number of required integrals.

Given the optimized sampling rates r_t, the last step is to find corresponding n_t. This can be achieved by first substituting the optimized r_t into Equation (7.15) and solving for n_t:

$$r_t = \frac{\gamma_t n_t}{\sum_{t'} \gamma_{t'} n_{t'}} \Leftrightarrow n_t = \frac{r_t}{\gamma_t} \left(\gamma_t n_t + \sum_{t' \neq t} \gamma_{t'} n_{t'}\right) \Leftrightarrow n_t - r_t n_t = \sum_{t' \neq t} \gamma_{t'} n_{t'} \tag{7.19a}$$

$$\Leftrightarrow n_t = \frac{\sum_{t' \neq t} \gamma_{t'} n_{t'}}{1 - r_t}. \tag{7.19b}$$

The resulting system of equations has infinitely many solutions. This is by construction, as the r_t only define a ratio between the different sample counts. If $\mathrm{n} = (n_1, \dots, n_T)$ is a solution, then so is any multiple like $(10n_1, \dots, 10n_T)$. Consequently, additional constraints have to be imposed to find an exact solution. Either by imposing a total budget $B = \sum_t n_t$ or by fixing the sample count of one technique to a constant, for example $n_1 = 1$. In the context of bidirectional algorithms, the latter is a good choice as it can enforce stratification by fixing the number of camera subpaths per pixel.

Section 7.3 shows how the required partial derivatives can be efficiently estimated from arbitrary MIS strategies. Thereby, the initial guess of the Newton-Raphson optimizer can be completely decoupled from the actual sample allocation strategy used to gather the statistics.

7.2.3 Brute-force optimization

Convex optimization is not possible if MIS weights other than the classic balance heuristic are used. As an alternative, we employ a simple brute-force search, illustrated in Figure 7.2.

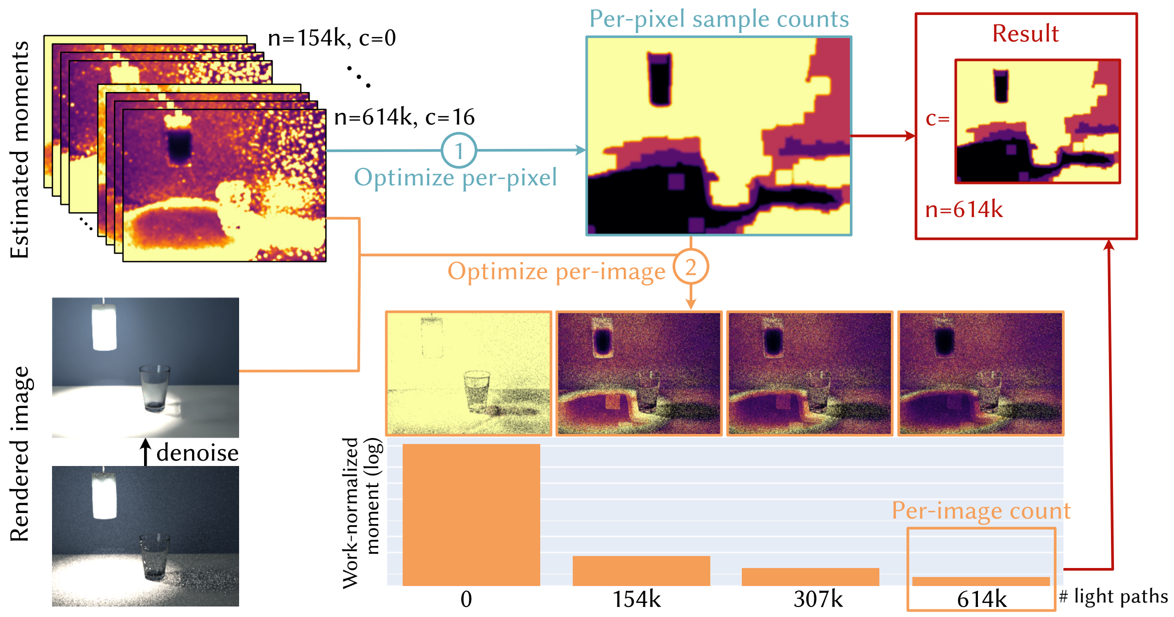


Figure 7.2: *Our optimization applied to a subset of bidirectional path tracing. In this example, we control the number of connections per pixel, while the number of light paths is set per image. Decisions are based on cheaply estimated second moments (shown in false color in the top left) and the denoised rendering (bottom left). The per-pixel counts are determined first (1), then we pick the best per-image counts (2) conditionally on those. The false color images in the bottom right depict the gathered relative moments for different numbers of light paths, given the fixed numbers of connections.*

First, we render an image and estimate the second moments of different candidate sample allocation strategies. We then pick the best per-pixel counts (1) via a simple search. Based on these decisions, we pick the conditionally best per-image sample counts (2). In the following, we describe these steps in more detail.

We start by identifying a set of N candidate sample-allocation strategies $\{\mathrm{n}^1, \dots \mathrm{n}^N\}$. This can be done either by (regularly) sampling the range of allowed sample counts, or by applying prior knowledge and domain-specific assumptions to identify the most promising candidates. During rendering, we estimate the second moments $M[\langle F\rangle_{\mathrm{n}}]$ of all candidate strategies $\mathrm{n} \in \{\mathrm{n}^1, \dots \mathrm{n}^N\}$ and simply pick the best one.

There are two essential tricks to make this approach practical. First, an efficient estimation scheme for the second moments is needed, that does not require samples from all candidates n. The next section shows how to accomplish that. Second, careful filtering should be applied to the second moment estimates, such that robust decisions can be made even with very few samples.

The pseudocode in Algorithm 3 provides an overview. Given a set of candidate strategies n, a cost function C, and a pilot strategy $\mathrm{m} = (m_1, \dots, m_T)$, the FindBestStrategy function (lines 1 - 5) determines the best per-pixel and per-image sample counts.

First, we render an image $\mathrm{I} = \{\langle I_1\rangle, \dots, \langle I_P\rangle\}$ with one set of samples from the pilot strategy. The per-pixel second moments of all candidates are estimated on-the-fly and accumulated in a moment image $\mathrm{M} = \{\langle M_1\rangle_1, \dots, \langle M_P\rangle_N\}$ storing one second moment per pixel and candidate.

Algorithm 3: *Pseudo-code for our optimization. Given a pilot strategy* m*, a set of candidate strategies* $\mathrm{n}^1, \dots, \mathrm{n}^N$*, and a cost function* C*, we render an image and estimate the second moments. Then, we first optimize all per-pixel sample counts* p_j*, followed by all per-image sample counts* i.

```
1:  function FindBestStrategy($\mathrm{m}, \mathrm{n}^1, \dots, \mathrm{n}^N, C$)
2:    $\mathrm{M}, \mathrm{I}$ = EstimateMomentsImage($\mathrm{m}, \mathrm{n}^1, \dots, \mathrm{n}^N$)
3:    $\{\mathrm{p}_j\}$ = OptPerPixel($\mathrm{M}, \mathrm{n}^1, \dots, \mathrm{n}^N, C$)
4:    $\mathrm{i}$ = OptPerImage($\mathrm{M}, \mathrm{I}, \mathrm{n}^1, \dots, \mathrm{n}^N, C, \{\mathrm{p}_j\}$)
5:    return $\{\mathrm{p}_j\}, \mathrm{i}$          ← Set of per-pixel sample counts, per-image sample counts
6:  function OptPerPixel($\mathrm{M}, \mathrm{n}^1, \dots, \mathrm{n}^N, C$)
7:    $\{\widetilde{M}_{1,1}, \dots, \widetilde{M}_{P,N}\}$ = FilterImage($\{\langle M_1 \rangle_1 \dots \langle M_P \rangle_N\}$)          ← Reduce noise in moments
8:    for pixel index $j = 1..P$ do
9:      $(\mathrm{p}_j, \_) = \arg\min \widetilde{M}_{j,\mathrm{n}} C_j(\mathrm{n})$          ← Via linear search
10:   return FilterImage($\{\mathrm{p}_1, \dots, \mathrm{p}_P\}$)          ← Avoid abrupt changes in sample counts
11: function OptPerImage($\mathrm{M}, \mathrm{I}, \mathrm{n}^1, \dots, \mathrm{n}^N, C, \{\mathrm{p}_j\}$)
12:   $\{\tilde{I}_1, \dots, \tilde{I}_P\}$ = DenoiseImage($\langle I_1 \rangle, \dots, \langle I_P \rangle$)
13:   $\{R_\mathrm{i}\} = \{0, \dots, 0\}$          ← Initialize total relative moment per image-level candidate
14:   $\{C_\mathrm{i}\} = \{0, \dots, 0\}$          ← Initialize total cost per image-level candidate
15:   for pixel index $j = 1..P$ do
16:     for all candidate strategy index $k = 1..N$ do
17:       if $\mathrm{p}_j \neq \mathrm{p}^k$ then
18:         continue          ← Skip candidates with different local counts
19:       $R_{\mathrm{i}^k}$ += $\langle M_j \rangle_k \cdot \tilde{I}_j^{-2}$          ← Accumulate relative moments
20:       $C_{\mathrm{i}^k}$ += $C_j(\mathrm{n}^k)$          ← Accumulate cost
21:   return $\arg\min\{R_\mathrm{i} \cdot C_\mathrm{i}\}$          ← Pick best per-image counts via simple search
```

Second, the per-pixel decisions (7.13) are made with a simple search. First, the second moment image **M** is filtered (line 7) to reduce noise. Then, in a loop over all pixels (line 8) the best per-pixel sample values are found by a simple linear search for the smallest work-normalized second moment (line 9). We also filter the resulting “image” of per-pixel sample counts to increase robustness (line 10). To that end, we found that a combination of dilation (to fill gaps) and blur (to avoid abrupt changes) effectively prevents visible artifacts.

Finally, the per-image decisions (7.14) are made. Since the ground truth pixel values are unknown, we run a denoiser on the rendered image I from the pilot iteration (line 12). For each per-image candidate i, we compute the marginalized relative second moment and the total cost in a loop over all pixels (lines 13 - 20). In each pixel, we iterate over all candidates, and disregard those that have different per-pixel settings (lines 17, 18). The best per-image sample counts are then again found via a simple linear search over the accumulated relative work-normalized second moments (line 21).

7.3 Computing the moments, means, or derivatives

Convex optimization and brute-force search both require estimates of the second moments or their derivatives. In simple low-dimensional applications such as direct illumination, these can be easily obtained via direct Monte Carlo estimation [Sbert et al. 2019]. However, for high-dimensional cases like bidirectional path tracing, numerical robustness is an issue.

In the following, we present a numerically robust estimation scheme to compute second moments, MIS weighted means, or derivatives of arbitrary strategies n given the samples of a different strategy m.

Consider the second moment of an MIS estimator using the candidate strategy n:

$$\mathrm{M}[\langle F\rangle_\mathrm{n}] = \int_\mathcal{X} \frac{f^2(x) \sum_t w_{\mathrm{n},t}(x) c_t(x)}{\sum_t c_t(x) n_t p_t(x)} \,\mathrm{d}x. \tag{7.20}$$

In theory, it is trivial to construct a naive MIS estimator using a pilot strategy m:

$$\langle \mathrm{M}[\langle F\rangle_\mathrm{n}]\rangle_\mathrm{m} = \sum_{t=1}^{T} \sum_{i=1}^{m_t} \frac{w_{\mathrm{m},t}(x_{t,i})}{m_t p_t(x_{t,i})} \frac{f^2(x_{t,i}) \sum_t w_{\mathrm{n},t}(x_{t,i}) c_t(x_{t,i})}{\sum_t c_t(x_{t,i}) n_t p_t(x_{t,i})}. \tag{7.21}$$

Like any other MIS estimator, we take some samples $x_{t,i}$, evaluate the integrand, divide by the PDF $p_t(x_{t,i})$ and sample count m_t, and multiply by the MIS weight $w_{\mathrm{m},t}(x_{t,i})$. However, there is a problem with this formulation: The sum in the denominator contains the PDFs $p_t(x)$. While that is not problematic for simple low-dimensional cases, it is a major problem for bidirectional algorithms. There, the PDF is a high-dimensional product of many local sampling densities, each of which can be arbitrarily high. The full product can frequently not be represented by floating point arithmetic, especially if it involves highly glossy surfaces or even Dirac deltas. On top of that, the summation can cause catastrophic cancellation due to vastly different magnitudes.

A numerically stable but impractical alternative is to estimate the second moment of n by using samples from this exact strategy. In that case, a numerically stable estimator is trivial to obtain, by simply squaring the sample weights:

$$\langle \mathrm{M}[\langle F\rangle_\mathrm{n}]\rangle_\mathrm{n} = \sum_{t=1}^{T} \sum_{i=1}^{n_t} \left(\frac{w_{\mathrm{n},t}(x_{t,i}) f(x_{t,i})}{n_t p_t(x_{t,i})} \right)^2. \tag{7.22}$$

That, however, would defeat the purpose of our optimization, as we would have to generate a full rendering iteration with every possible sample allocation strategy.

7.3.1 Our second moment estimator

Our solution is to square the samples of the pilot strategy, and multiply them by a correction factor δ to obtain the desired second moment:

$$\langle \mathrm{M}[\langle F\rangle_\mathrm{n}]\rangle_\mathrm{m} = \sum_{t=1}^{T} \sum_{i=1}^{m_t} \left(\frac{w_{\mathrm{m},t}(x_{t,i}) f(x_{t,i})}{m_t p_t(x_{t,i})} \right)^2 \delta_{\mathrm{n},\mathrm{m}}(x_{t,i}). \tag{7.23}$$

The correction factor is derived in Section 7.A. It is a ratio of the weighted sums of MIS weights w_a of a proxy strategy a:

$$\delta_{\mathrm{n},\mathrm{m}}(x) = \frac{\left(\sum_t \frac{m_t}{a_t} w_{\mathrm{a},t}(x)\right)^2 \sum_t c_t(x) \frac{n_t}{a_t} w_{\mathrm{a},t}(x)}{\left(\sum_t \frac{n_t}{a_t} w_{\mathrm{a},t}(x)\right)^2 \sum_t c_t(x) \frac{m_t}{a_t} w_{\mathrm{a},t}(x)}. \tag{7.24}$$

The proxy strategy $\mathrm{a} = (\mathrm{a}_1, \dots, \mathrm{a}_T)$ can be chosen arbitrarily. It serves two practical purposes: (1) it enables numerically stable computation and (2) it separates out terms not

Algorithm 4: *Pseudo-code for our second moment estimation. Given a pilot strategy and a set of candidate strategies, we render an image and compute the per-pixel second moments of all candidates on-the-fly in the inner loop.*

```
1:  function EstimateMomentsImage($\mathrm{m}, \mathrm{n}^1, \ldots, \mathrm{n}^N$)
2:    $\mathrm{I} = \{\langle I_1 \rangle, \ldots, \langle I_P \rangle\} = \{0, \ldots, 0\}$    ← Initialize pixel estimates
3:    $\mathrm{M} = \{\langle M_1 \rangle_1, \ldots, \langle M_P \rangle_N\} = \{0, \ldots, 0\}$    ← $N$ moments per pixel
4:    $\mathrm{a} = \{\sum_{k=1}^{N} n_k^1/N, \ldots, \sum_{k=1}^{N} n_k^T/N\}$    ← Set proxy strategy (arbitrary)
5:    for pixel index $j = 1..P$ do
6:      for technique index $t = 1..T$ do
7:        for sample index $i = 1..m_t$ do
8:          $y = \frac{w_{\mathrm{m},t}(x_{t,i}) f_j(x_{t,i})}{m_t p_t(x_{t,i})}$    ← MIS-weighted pixel $j$ estimate
9:          $\langle I_j \rangle$ += $y$    ↙ Precompute terms in $\delta_{\mathrm{n},\mathrm{m}}(x_{t,i})$ independent of $\mathrm{n}$
10:         Precompute$\left(\frac{(\sum_t \frac{m_t}{a_t} w_{\mathrm{a},t}(x_{t,i}))^2}{\sum_t c_t(x_{t,i}) \frac{m_t}{a_t} w_{\mathrm{a},t}(x_{t,i})}, \frac{w_{\mathrm{a},1}(x_{t,i})}{a_1}, \ldots, \frac{w_{\mathrm{a},T}(x_{t,i})}{a_T}\right)$
11:         for candidate strategy index $k = 1..N$ do
12:           $\langle M_j \rangle_k$ += $y^2 \cdot \delta_{\mathrm{n}^k,\mathrm{m}}(x_{t,i})$    ← Moment estimate (7.23)
13:   return M, I
```

involving n_t so that they can be pre-computed and reused for all candidate moments. For numerical stability, it is beneficial for n_t and a_t to have similar orders. Thus, we set each a_t to the average of n_t across all candidates.

For the classical balance heuristic (i.e., $c_t(x) = 1$) the correction factor simplifies to

$$\delta_{\mathrm{n},\mathrm{m}}^{\mathrm{bal}}(x) = \frac{\sum_t \frac{m_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_t \frac{n_t}{a_t} w_{\mathrm{a},t}(x)}. \tag{7.25}$$

The process is outlined in Algorithm 4. As usual, we render an image by iterating over all pixels and taking samples from all techniques. The number of samples is determined by the pilot strategy m. Each sample is added to the image (lines 8 and 9). Additionally, we compute the MIS weights of the proxy strategy (line 10). With that per-sample data, we iterate over all candidates and accumulate the second moment estimates using our correction factor (lines 11 and 12).

7.3.2 Our squared mean estimator

The moment estimation can be easily extended to full variance computations of uncorrelated techniques[2]. In that case, we additionally need the squared MIS weighted means:

$$\mu_{\mathrm{n},t}^2 = \left(\int_{\mathcal{X}} w_{\mathrm{n},t}(x) f(x) \, \mathrm{d}x \right)^2. \tag{7.26}$$

Due to the squaring, these integrals have to be estimated individually. Hence, operating with full variances is not feasible for involved combinations like bidirectional path tracing.

We can adopt the same approach as for the second moments and estimate the MIS weighted means via samples from the pilot strategy m. Therefor, we replace the MIS weight of the

[2]Estimating the correlation is more challenging and requires some form of splitting or merging to happen also in the pilot strategy.

candidate strategy n by a ratio of MIS weights involving the proxy strategy a (7.40d):

$$\int_{\mathcal{X}} w_{\mathrm{n},t}(x) f(x)\,\mathrm{d}x = \int_{\mathcal{X}} \frac{\frac{n_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_{t'} \frac{n_{t'}}{a_{t'}} w_{\mathrm{a},t'}(x)} f(x)\,\mathrm{d}x. \tag{7.27}$$

The corresponding multi-sample MIS estimator using the pilot strategy m,

$$\langle \mu_{\mathrm{n},t} \rangle_{\mathrm{m}} = \sum_{t=1}^{T} \sum_{i=1}^{m_t} \left(\frac{w_{\mathrm{m},t}(x_{t,i})\, f(x_{t,i})}{m_t p_t(x_{t,i})} \right) \frac{\frac{n_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_{t'} \frac{n_{t'}}{a_{t'}} w_{\mathrm{a},t'}(x)}, \tag{7.28}$$

again simply multiplies a correction factor on the sample weights, but this time without squaring them.

7.3.3 Our derivative estimator

The derivatives of the convex objective, Equations (7.17a) and (7.17b), can be estimated in the same way. To demonstrate, we rewrite the integral in the first derivative as follows:

$$\begin{aligned}
&\int_{\mathcal{X}} \frac{f^2(x)}{\left(\sum_{t'} r_{t'} \frac{p_{t'}(x)}{\gamma_{t'}}\right)^2} \left(\frac{p_T(x)}{\gamma_T} - \frac{p_t(x)}{\gamma_t} \right) \mathrm{d}x && (7.29a)\\
&= C(\mathrm{n})^2 \int_{\mathcal{X}} \frac{f^2(x)}{\left(\sum_{t'} n_{t'} p_{t'}(x)\right)^2} \left(\frac{p_T(x)}{\gamma_T} - \frac{p_t(x)}{\gamma_t} \right) \mathrm{d}x && (7.29b)\\
&= C(\mathrm{n})^2 \int_{\mathcal{X}} \frac{f^2(x)}{\sum_{t'} n_{t'} p_{t'}(x)} \left(\frac{w_{\mathrm{n},T}(x)}{n_T \gamma_T} - \frac{w_{\mathrm{n},t}(x)}{n_t \gamma_t} \right) \mathrm{d}x =: C(\mathrm{n})^2 D_{\mathrm{n},t} && (7.29c)
\end{aligned}$$

That is, we first re-introduce the cost function, by computing an n that corresponds to the ratio r. Then, we split the squared sum in the denominator and use it to replace the difference of PDFs by a difference of MIS weights.

The corresponding MIS estimator, using the pilot strategy m, is then obtained by combining the two correction factors used for the second moment and the mean,

$$\langle D_{\mathrm{n},t} \rangle_{\mathrm{m}} = \sum_{t=1}^{T} \sum_{i=1}^{m_t} \left(\frac{w_{\mathrm{m},t}(x_{t,i})\, f(x_{t,i})}{m_t p_t(x_{t,i})} \right)^2 \delta_{\mathrm{n},\mathrm{m}}(x_{t,i}) \frac{\frac{1}{a_T \gamma_T} w_{\mathrm{a},T}(x) - \frac{1}{a_t \gamma_t} w_{\mathrm{a},t}(x)}{\sum_{t'} \frac{n_{t'}}{a_{t'}} w_{\mathrm{a},t'}(x)}, \tag{7.30}$$

and multiplying them on the squared sample weights, as done for the second moments.

The second derivative can be computed in the same manner, applying a binomial expansion to convert the squared difference of PDFs into MIS weights:

$$
\begin{aligned}
&\int_{\mathcal{X}} \frac{f^2(x)}{\left(\sum_{t'} r_{t'} \frac{p_{t'}(x)}{\gamma_{t'}}\right)^3} \left(\frac{p_T(x)}{\gamma_T} - \frac{p_t(x)}{\gamma_t}\right)^2 \mathrm{d}x && \text{(7.31a)}\\
&= C(\mathrm{n})^3 \int_{\mathcal{X}} \frac{f^2(x)}{\left(\sum_{t'} n_{t'} p_{t'}(x)\right)^3} \left(\frac{p_T(x)}{\gamma_T} - \frac{p_t(x)}{\gamma_t}\right)^2 \mathrm{d}x && \text{(7.31b)}\\
&= C(\mathrm{n})^3 \int_{\mathcal{X}} \frac{f^2(x)}{\left(\sum_{t'} n_{t'} p_{t'}(x)\right)^3} \left(\frac{p_T^2(x)}{\gamma_T^2} + \frac{p_t^2(x)}{\gamma_t^2} - 2\frac{p_T(x)p_t(x)}{\gamma_T\gamma_t}\right) \mathrm{d}x && \text{(7.31c)}\\
&= C(\mathrm{n})^3 \int_{\mathcal{X}} \frac{f^2(x)}{\sum_{t'} n_{t'} p_{t'}(x)} \left(\frac{w_{\mathrm{n},T}^2(x)}{n_T^2\gamma_T^2} + \frac{w_{\mathrm{n},t}^2(x)}{n_t^2\gamma_t^2} - 2\frac{w_{\mathrm{n},T}(x)w_{\mathrm{n},t}(x)}{n_T n_t \gamma_T \gamma_t}\right) \mathrm{d}x && \text{(7.31d)}\\
&=: C(\mathrm{n})^3 D_{\mathrm{n},t}^2 && \text{(7.31e)}
\end{aligned}
$$

The correction factor for the corresponding MIS estimator is:

$$
\delta_{\mathrm{n,m}}(x_{t,i}) = \frac{\left(\frac{1}{a_T\gamma_T} w_{\mathrm{a},T}(x)\right)^2 + \left(\frac{1}{a_t\gamma_t} w_{\mathrm{a},t}(x)\right)^2 - 2\frac{1}{a_T\gamma_T} w_{\mathrm{a},T}(x)\frac{1}{a_t\gamma_t} w_{\mathrm{a},t}(x)}{\left(\sum_{t'} \frac{n_{t'}}{a_{t'}} w_{\mathrm{a},t'}(x)\right)^2}. \qquad (7.32)
$$

Derivatives, second moments, and variances can be computed by accumulating the same basic per-sample quantities: the sample weights of the pilot and the MIS weights of the proxy strategy. Hence, more elaborate optimization schemes are easily facilitated by combining, for example, moments and derivatives.

7.4 Application: Direct illumination

One possible application of our theory is ray-traced direct illumination. There, BSDF importance sampling and light source sampling (aka next event estimation) are combined via MIS. In our application, rendering is done in iterations; each iteration traces a single primary ray per pixel and then estimates direct illumination with a fixed budget of 32 samples, distributed across the two techniques. The goal is to determine the optimal percentage of BSDF samples per pixel.

We chose this application not because it is of high practical relevance, but because it is a context where a comparison to previous work is possible. There is little potential for improvement in this application: There are just two techniques, the sampling costs of which, while not identical, are at least on the same order of magnitude, and the problem is low-dimensional, i.e., PDFs can be computed directly. Unsurprisingly, the results, summarized in Figure 7.3, show only marginal improvement from our method over previous work.

The figure depicts the BSDF sampling percentages as false-color images, and provides equal-sample error comparisons (in terms of relMSE), and equal-time speedups. The equal-time speed-ups where computed by multiplying the equal-sample errors with the respective render times.

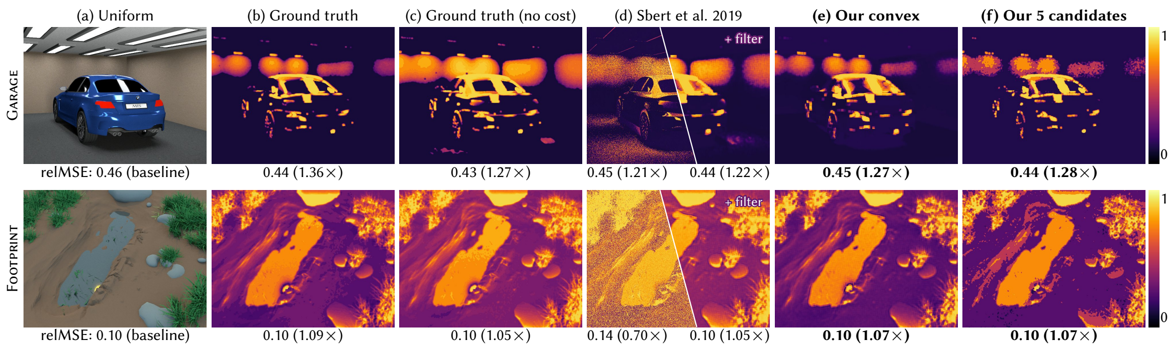


Figure 7.3: *Results in a very simple direct illumination application. The false color images show the ratio of BSDF samples per pixel. The numbers below compare the equal-sample error (relMSE, lower is better) and the speed-up (w.r.t. uniform allocation, higher is better). The speed-ups are computed by multiplying the equal-sample errors with the respective render times. We compute ground-truth ratios with (b) and without (c) accounting for sampling cost. The convex optimization of Sbert et al. [2019] (d) converges to the ground truth without sampling cost. Applying our insights produces minor speed-ups by accounting for the cost in convex optimization (e). Lastly, our brute-force search with just 5 candidates (e) provides results on par with the iteratively refined convex optimization.*

The baseline is uniform allocation, i.e., 16 samples are taken from each technique. The ground truth (b) has been computed by running our brute-force approach with all possible allocations as candidates in a pre-pass. We also computed the same ground truth but assuming equal sampling cost (c). In our implementation, BSDF samples are, on average, 60% more expensive than light samples. Not accounting for this cost produces lower equal-sample error, but worse equal-time results. However, since the cost difference is rather small, the achievable speed-up is only about 5% on average.

We compare the results to the method of Sbert et al. [2019] (d). We found that applying a low-pass filter in image space to the estimated derivatives significantly improves results. With the additional filtering, their method converges nicely to the ground truth without cost shown in (c). With some residual noise from sub-optimal early iterations.

Applying the convex variant of our more general objective (e) produces consistent, albeit marginal, improvements by accounting for the cost differences. Finally, the brute-force approach (f) works surprisingly well here. With just 5 candidates (0.1, 0.25, 0.5, 0.75, and 0.9) and the moments estimated from a single iteration, it provides results on par with the iteratively refined convex optimization (e).

7.5 Application: Vertex connection and merging

Our main application is the vertex connection and merging (VCM) algorithm [Georgiev et al. 2012a; Hachisuka et al. 2012]. It combines bidirectional path tracing (BDPT) [Lafortune and Willems 1993; Veach and Guibas 1995a] with photon mapping [Jensen 1996]. Paths are traced from the camera and the lights. Each ray along these paths might hit a light (or the camera), and next event estimation is done at every vertex to connect to a light (or the lens). On top of that, each vertex on a camera subpath is connected to some number of light-subpath vertices. Lastly, merging (a.k.a. photon density estimation) is performed at each camera-subpath vertex, except the first on the lens and the directly visible point.

Optimizing all parameters of VCM involves, e.g., setting the number of connections for each pixel, point in space, and camera-subpath length, which is prohibitively expensive. To keep the problem feasible, we reduce the degrees of freedom by only controlling

1. n, the number of light paths traced,
2. c, the number of connections as a global constant, and
3. χ_j, whether to perform merging in each pixel j.

For the set of candidate strategies, we consider all possible combinations of (with P being the number of pixels)

$$n \in \left\{\frac{1}{4}P, \frac{1}{2}P, \frac{3}{4}P, P, 2P\right\}, \quad c \in \{0, 1, 2, 4, 8, 16\}, \quad \chi_j \in \{0, 1\}, \tag{7.33}$$

as well as the special case ($n = 0, c = 0, \chi_j = 0\,\forall j$), i.e., unidirectional path tracing. Note that the number of light paths n is only allowed to be zero if all other bidirectional techniques are disabled.

7.5.1 Implementation

We have implemented our method on top of the public code of Grittmann et al. [2021]. All experiments were run on a 16-core AMD Ryzen 9 3950X processor with 64 GB of memory. The results shown in the following, unless stated otherwise, are equal-time renderings after 60s at a resolution of 640×480 (we also rendered some scenes for 25 min at 4K resolution, with similar results). We use the relative mean squared error (relMSE) as an error metric, which is an estimate of our optimization objective. Since the (rel)MSE is not robust to outliers, we ignore the 0.01% of pixels (i.e., 30 in total at our typical resolution) with highest error.

The optimization is done in up to two iterations, each rendering one sample per pixel. We start with unidirectional path tracing as the pilot strategy. If the outcome is to switch to a bidirectional technique, we optimize one more time with the samples of the bidirectional technique, which gives higher-quality second-moment estimates. If the pilot decides not to enable bidirectional sampling, no further optimization is done.

7.5.1.1 Cost heuristic

The cost of a strategy n is the expected time it takes to render one iteration with that strategy. We approximate that cost with a heuristic that roughly corresponds to the number of ray-tracing and shading operations:

$$C(n, c, \chi) = C_{\mathrm{light}} \tilde{n}_{\mathrm{l}} n + P \tilde{n}_{\mathrm{c}} \big(C_{\mathrm{cam}} + C_{\mathrm{con}} c + C_{\mathrm{m}} \tilde{n}_{\mathrm{l}} n \tilde{n}_{\mathrm{m}} \textstyle\sum_j \chi_j\big), \tag{7.34}$$

which is the sum of the cost (incl. next-event estimation) of tracing n light paths of average length $\tilde{n}_{\mathrm{l}}$, and the cost of tracing P camera paths (i.e., one per pixel) of average length $\tilde{n}_{\mathrm{c}}$ performing at each vertex next-event, connections, and merges. Here, $\tilde{n}_{\mathrm{m}}$ is the average number of photons found by each density estimation, as a fraction of the total number of photons $\tilde{n}_{\mathrm{l}} n$ in the scene.

The relative costs of the different techniques are controlled by four hyperparameters. C_{light} and C_{cam} are the combined costs of continuing the respective path at each vertex and

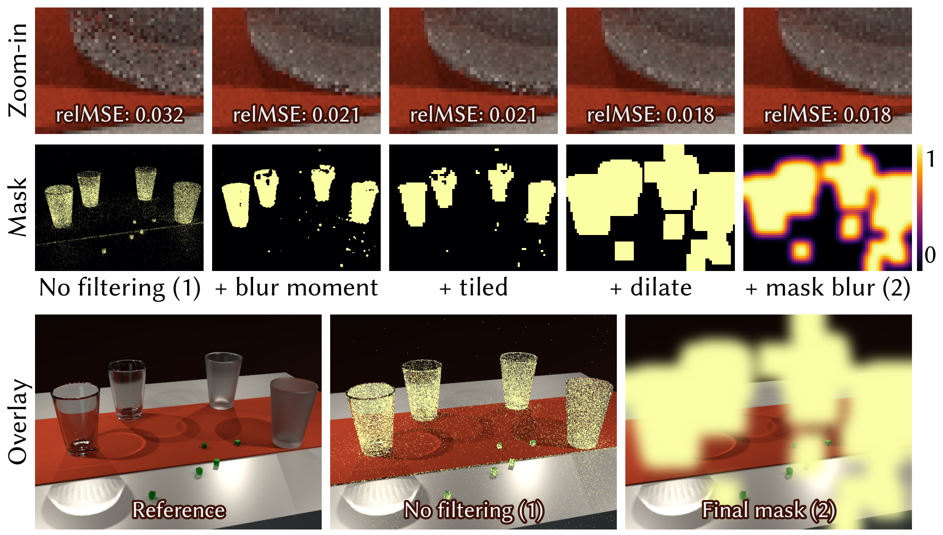


Figure 7.4: *When optimizing the pixel-level merging decisions χ_j, we apply a simple filtering scheme to increase robustness. The top row shows the effect of the different operations on the merge mask and the equal-time error. The bottom row shows an overlay of the mask over the reference image, without filtering and after applying our filtering.*

performing next-event estimation. C_{con} is the cost of a single connection and C_{m} is the cost of a single merge operation. We determined the best values for our implementation by fitting the cost heuristic to brute-force measured render times across 25 test scenes: $C_{\text{cam}} = C_{\text{light}} = 1$, $C_{\text{con}} = 0.4$, and $C_{\text{m}} = 0.5$. These numbers are implementation-specific and likely differ between renderers. The average path lengths are determined on-the-fly from rendering statistics. When using a unidirectional pilot strategy, only $\tilde{n}_{\text{c}}$ is available, in which case we initialize the remaining statistics with an initial guess: $\tilde{n}_{\text{l}} = \tilde{n}_{\text{c}}$, $\tilde{n}_{\text{m}} = 10^{-7}$.

The supplemental document shows that the above simple cost heuristic closely matches the actual render time in our tests. It also provides empirical evidence indicating that the accuracy of the cost heuristic is secondary compared to the other sources of error – using moments instead of variance and noise in the estimates.

7.5.1.2 Merge mask

Merging only benefits a specific type of effect, reflected or refracted caustics, that is often limited to small regions in the image. Hence, a global decision is apt to neglect these small regions to increase efficiency everywhere else. Therefore, we control the binary decision whether to perform merging on the pixel-level, by computing a *merge mask*.

Care has to be taken to avoid visible artifacts, since we only have a single sample per pixel. We apply a simple filtering scheme, shown in Figure 7.4. First, the second-moment images are blurred (Gaussian filter, radius 8) to reduce noise and downsampled (averaged into 8×8-pixel tiles) to reduce overhead. Then, we run our pixel-level optimization. The resulting mask can still contain gaps from missing data, where merging in important regions may be incorrectly disabled. To combat that, we dilate the mask (box, radius 4). Finally, to remove discontinuties in the noise pattern from abruptly changing sample counts, which could cause visible artifacts, we blur the dilated mask (Gaussian, radius 4). The resulting mask after the blur contains floating point values between zero and one, which we take as the

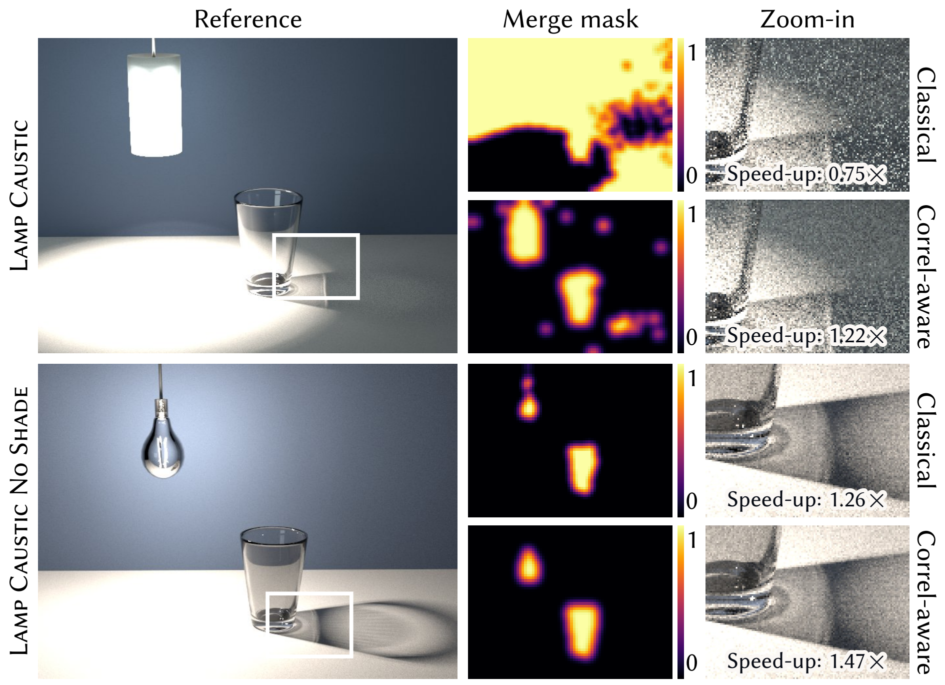


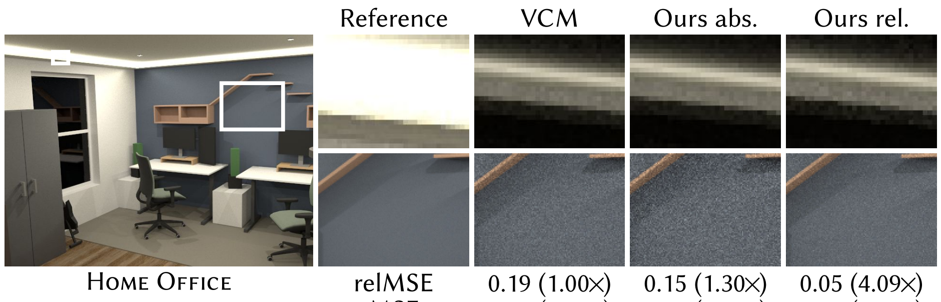
Figure 7.5: *A scene rendered with and without a lamp shade. We show the merge masks and the rendering speed-up due to our optimization when using the classical balance heuristic and Grittmann et al.'s correlation-aware weights. The lamp shade causes severe covariance in the merging techniques, and our optimization can further worsen the already poor performance of the classical balance heuristic. Using the correlation-aware weights avoids this problem by assigning low MIS weights to the problematic samples. Hence our optimization does not enable merges because they would not contribute to the combined estimate.*

probability to perform merging at each vertex.

This simple filtering can be improved further with adaptive kernels and other ideas commonly used by denoising methods. The supplemental document discusses the impact of the different filters and their parameters.

7.5.1.3 Sample correlation

Sample correlation in the merging techniques can be a problem in VCM [Grittmann et al. 2021]. The second moments can grossly *underestimate* the actual variance of merging. Hence, optimizing the sample counts based on second moments can produce suboptimal results. To circumnavigate this, we utilize the correlation-aware MIS weights of Grittmann et al. [2021] (see Chapter 6). The effect of optimizing the sample allocation w.r.t. these weights is shown in Figure 7.5. It shows the merge masks and renderings with and without correlation-aware weights. We report speed-ups over vanilla VCM (i.e., $n = P$, $c = 1$, $\chi_j = 1$) using both the classical balance heuristic and the correlation-aware weights. Applying our moment-based optimization further amplifies the correlation problem when using the balance heuristic which already struggles with poor approximation from second moments. Applying it w.r.t. the correlation-aware weights produces consistent speed-ups. All results in the following use the correlation-aware weights for both the baseline and our method.



	VCM	Ours abs.	Ours rel.
relMSE	0.19 (1.00×)	0.15 (1.30×)	0.05 (4.09×)
MSE	0.11 (1.00×)	0.13 (0.83×)	0.29 (0.38×)
n, c	307k, 1	614k, 0	614k, 16

Figure 7.6: *In this scene, light tracing is the sole technique that can render the bright strip around the ceiling (top row, reduced exposure of all methods other than the reference). Optimizing for absolute moments overfits on this bright region and disables all other techniques for the entire image. Using relative moments resolves the problem. Note that "Ours rel." yields lower relMSE but higher MSE than vanilla VCM.*

7.5.1.4 Relative error

We found that using the relative moments (rather than absolute moments, see Section 7.1.2) is essential when optimizing the number of light paths n and global number of connections c. Figure 7.6 shows an example where the scene is dominated by indirect illumination from a bright strip of light along the ceiling. The direct illumination in the strip is best handled by light tracing. But all other illumination in the scene is best handled by bidirectional connections. Optimizing the absolute moments (i.e., omitting the pixel-value normalization) results in overfitting on the bright direct illumination and disables all connections. Optimizing with relative moments instead yields $c = 16$ connections and four times faster rendering. The caveat is that this approach requires good estimates/approximations of the ground-truth pixel values. To obtain those from a one-sample-per-pixel rendering, we denoise the image using Intel's Open Image Denoise [Áfra 2019].

7.5.1.5 Choosing the pilot strategy

We consider two options for the pilot strategy: forward path tracing (PT) or vanilla VCM ($n = P$, $c = 1$, $\chi_j = 1$). Figure 7.7 compares these options on two extreme cases, rendered for only 10s. In simple scenes that are best rendered unidirectionally, such as Modern Living Room, using vanilla VCM as the pilot reduces performance for shorter render times, due to the wasted bidirectional samples in the first iteration. With PT as the pilot, the overhead is limited to that of the optimization. For scenes like Sponge, which is illuminated solely by a caustic, starting with VCM provides the best performance because there the unidirectional samples from a PT pilot are wasted. However, these unidirectional samples are much cheaper than bidirectional ones, and they provide sufficient information for our optimization.

An advantage of a vanilla VCM pilot is the lower noise in the second moment estimates used by our optimization. We found that PT and VCM pilots produce very similar results when optimizing image-level parameters, e.g., number of light paths n and number of connections c. However, for pixel-level optimization, a single sample per pixel from a PT pilot is insufficient.

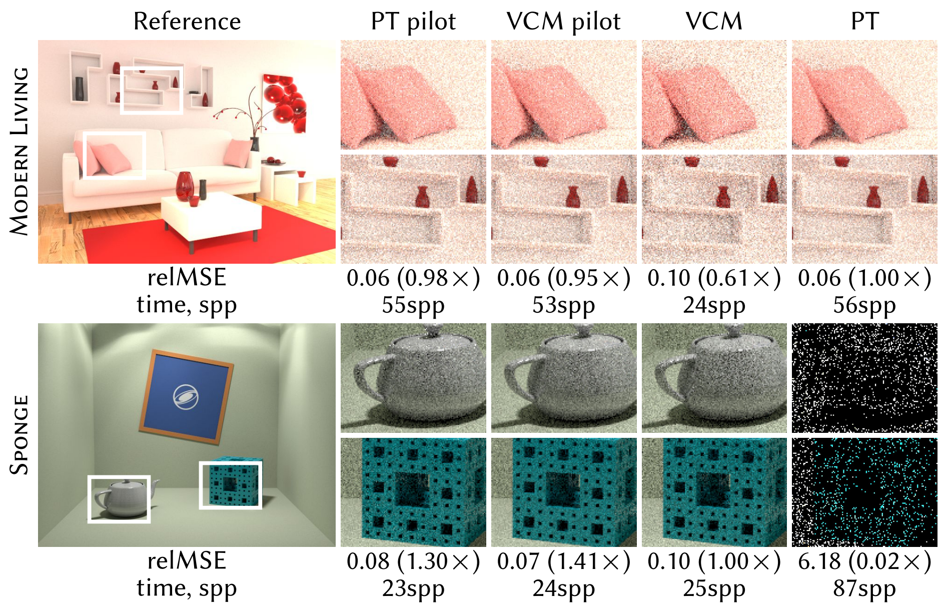


Figure 7.7: *Impact of the pilot strategy in short renderings. We show the relMSE after 10s (lower is better) and the speed-up in parentheses compared to the baseline (higher is better). Modern Living Room does not benefit from bidirectional methods. Using forward path tracing (PT) as the pilot limits the overhead to that of the optimizer (2%). Starting with VCM incurs additional overhead from (wasted) bidirectional samples. The solely caustic illumination in the Sponge scene is difficult to render with PT. Nevertheless, using PT as a pilot gives enough information to update the sampling strategy and still achieve faster rendering than vanilla VCM.*

Our solution is to optimize in two stages. We start with a PT pilot and only image-level optimization. If that optimization enables bidirectional sampling, we render one iteration with the optimized bidirectional strategy. The second moments estimated from that iteration are then used to perform a full optimization, including the pixel-level merge mask. This hybrid pilot minimizes overhead in simple scenes and produces the same optimization as a VCM pilot in difficult scenes.

The rendered image of the pilot can be averaged into the final result. However, the PT pilot may have excessive noise that will not vanish quickly. Thus, if bidirectional sampling is enabled by the PT pilot, we discard both the rendered image and the second moments from the PT pilot, and start from scratch with our optimized VCM pilot. The rendered image of the VCM pilot is averaged with the subsequent iterations, as it has a similar level of noise.

7.5.2 Results

We tested our method on bidirectional path tracing (BDPT) and full VCM. An overview of the results across our 25 test scenes (22 for BDPT) is shown in Table 7.1. Our optimized BDPT achieves 18% faster rendering on average than vanilla BDPT ($n = P$, $c = 1$). Our optimized VCM achieves consistent speed-ups of up to 5× over vanilla VCM ($n = P$, $c = 1$, $\chi_j = 1$) across all scenes; the worst case is still 12% faster. This is because merging is expensive and often not beneficial. In both variants, our method performs consistently faster than unidirectional path tracing (PT), with a worst-case slowdown of 2% and a best-case speed-up of 600×.

Table 7.1: *Statistics of the speed-up (higher is better) of our method across the 25 test scenes of the VCM application and the 22 scenes of the BDPT application. Computed after 60s rendering, averaged across 5 runs, using the relMSE error metric with outlier removal.*

Speed-up	**vs. path tracing**		**vs. vanilla**	
	BDPT	VCM	BDPT	VCM
Average	3.14×	5.00×	1.18×	1.68×
Worst	0.99×	0.98×	0.95×	1.12 ×
Best	97.09×	600.77×	1.60×	5.32×

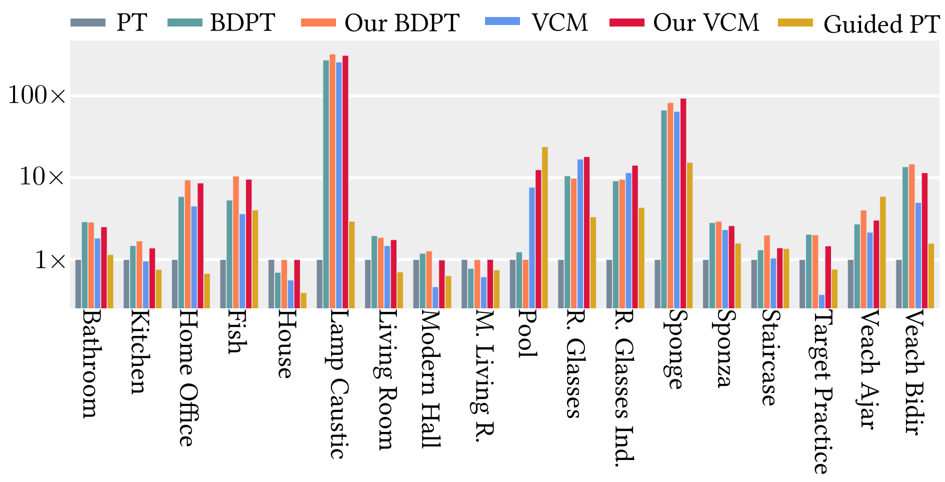


Figure 7.8: *Speed-up in terms of relMSE of different methods over unidirectional path tracing. 'Guided PT' is the path guiding method of Ruppert et al. [2020]. Our method consistently outperforms unidirectional PT and vanilla VCM.*

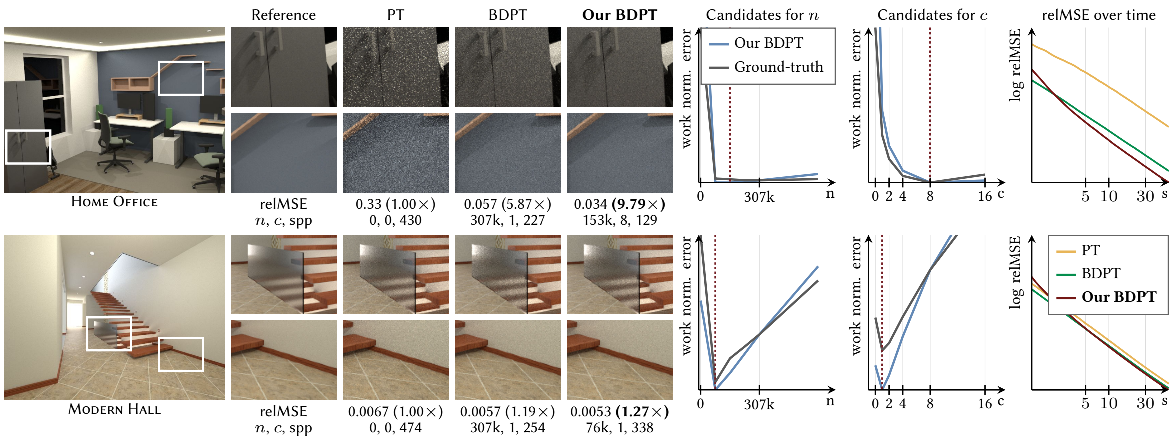


Figure 7.9: *Equal-time (60s) comparison between our optimized BDPT and two baselines. n and c are the number of light subpaths and bidirectional connections, respectively. The numbers in parentheses are the speed-ups (higher is better) over forward path tracing (PT). The plots compare our estimated work-normalized moments to ground truth work-normalized variances for different choices of n and c. The dashed red line marks the sample count chosen by our optimization. Basing our optimization on the second moments produces similar sample counts as the much more expensive full variances and yields consistent equal-time speed-ups compared to both baselines.*

Figure 7.8 plots the speed-ups over PT of all methods for multiple test scenes. It also compares the performance to that of guided forward path tracing [Ruppert et al. 2020]. The supplemental materials of the original paper [Grittmann et al. 2022] provide an interactive viewer with all rendered images.

7.5.2.1 Bidirectional path tracing

Figure 7.9 shows two scenes rendered with forward path tracing, vanilla BDPT, and our adaptive BDPT. Home Office is dominated by diffuse indirect illumination and benefits from many bidirectional connections. Modern Hall is overall well-handled by forward path tracing. By reducing the number of light paths, our method finds a sweet-spot providing a minor increase in performance. In both scenes, the overhead due to the unidirectional pilot iteration reduces performance initially, but for longer renderings our method performs consistently better than both baselines.

The first two columns of plots in Figure 7.9 compare our estimated work-normalized second moments to ground-truth work-normalized variances. The ground-truth is obtained by brute-force rendering with different sample counts and computing the product of relMSE and render time. Error values are plotted for different n given our decision for c (first column) and for different c, given our decision for n (second column). Our estimates do not perfectly match the ground truth, though they yield the same or very similar minima, which suffices for our optimization.

7.5.2.2 Full VCM

Adding merging into the mix of techniques enables robust rendering of reflected and refracted caustics. In Figure 7.10 we compare the performance of our optimized VCM to different baselines. In both VCM variants we use correlation-aware MIS weighting [Grittmann et al. 2021] (see Chapter 6).

The exterior, environment-map lit Pool scene contains a caustic that is very challenging for unidirectional path tracing to sample. VCM performs much better but also struggles with the very low photon density. Our optimization improves efficiency by using the maximum allowed number of light paths, disabling connections, and limiting merges to the region containing refracted caustics.

The Veach Bidir scene [Veach and Guibas 1995a] was originally modelled to showcase BDPT. Naturally, it benefits from many light paths and connections, but also features a caustic that benefits from merging. The merge mask generated by our method restricts the costly merges to the caustic image region. The error over time (last column) in this scene reveals the impact of starting with a PT pilot. This first unidirectional iteration is wasted. Hence, our method performs worse than vanilla VCM for short renderings of less than three seconds. Note that our optimized VCM has a numerically higher error than vanilla BDPT, but produces a qualitatively better image. The reason is that the reflected caustic in the glass egg manifests in BDPT as two outlier pixels that are ignored by our error metric. Our VCM captures this caustic almost perfectly, at the cost of slightly increased error elsewhere.

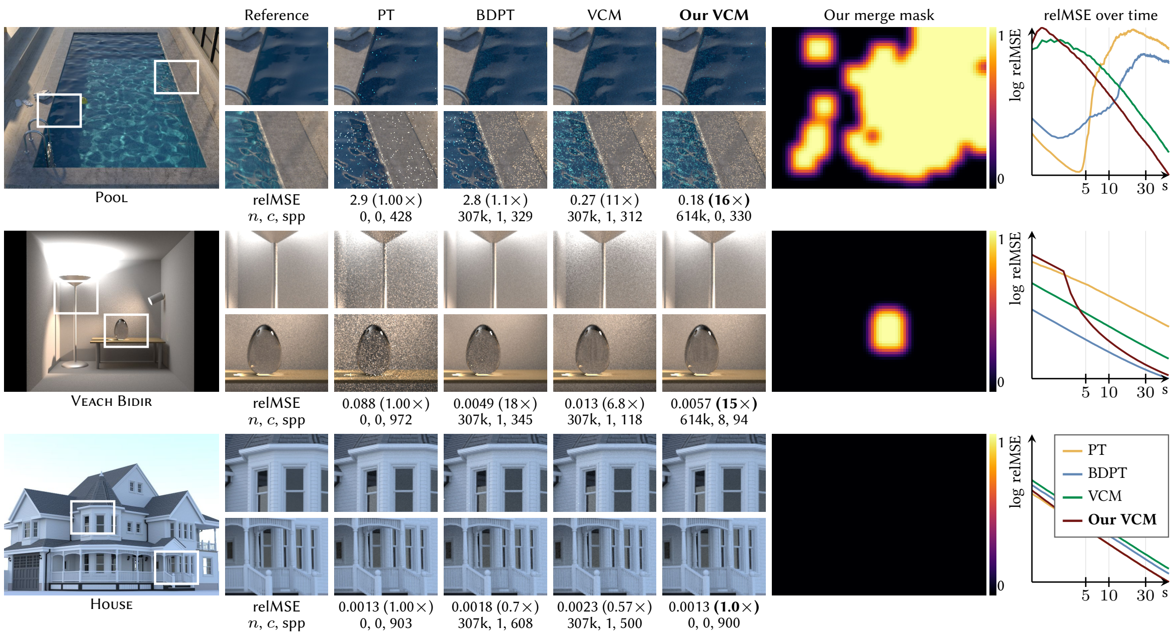


Figure 7.10: *Equal-time (60s) comparison between our optimized VCM and three baselines. The numbers in parentheses are the speed-up (higher is better) over forward path tracing (PT). The false-color image visualizes our per-pixel decision whether to perform merging. The* Pool *scene is dominated by caustics and benefits mostly from merging and light tracing.* Veach Bidir *is mostly indirectly illuminated and features a small glass egg, benefiting from bidirectional connections and local merging.* House *is best rendered unidirectionally, and our method sticks to PT, incurring less than 1% overhead due to the optimization.*

Lastly, the diffuse exterior House scene is a case where bidirectional sampling is wasteful. Vanilla VCM and BDPT are much slower than forward path tracing in this case. Our optimized VCM completely avoids tracing paths from the lights and only incurs a tiny ($< 1\%$) overhead from the optimization. The result is visually indistinguishable from the PT image.

7.5.2.3 Merge masks

Obtaining a reliable merge mask from a few (or even one) samples per pixel requires filtering that noisy data. Figure 7.11 compares our masks to ones obtained from ground-truth second moments computed from 4096 iterations of vanilla VCM. Our filtered 1-spp mask covers all crucial regions but can, of course, be improved further. An interesting insight

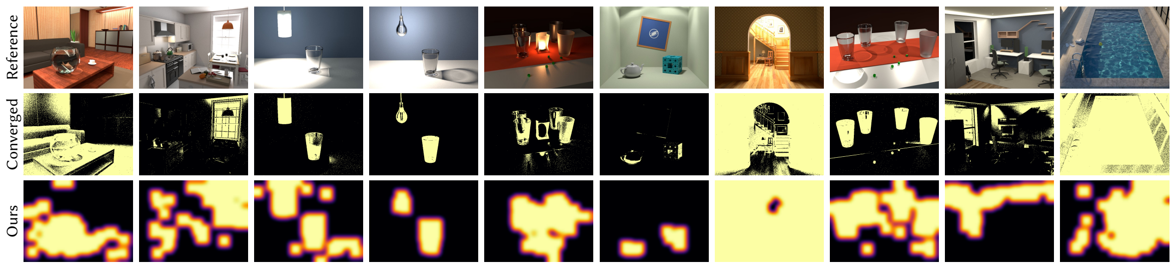


Figure 7.11: *Comparison of our aggressively filtered merge masks against ground-truth masks. Our masks are computed from 1-spp VCM with image-level optimization. The converged masks are based on second-moment estimates obtained from 4096 iterations of vanilla VCM.*

is that merges are mostly (though not always) useful in pixels that see highly glossy surfaces. A heuristic that limits merging to such pixels, based directly on material properties, could eliminate the overhead of our merge-mask computation. Such a heuristic would be reflected in the MIS weights, as part of the effective density $np(x)$, hence it trivially integrates into our optimization.

7.5.2.4 Overhead

The main computational cost in our approach is the large number of moments that are estimated and processed. The VCM application computes 61 second moments per pixel (one for each candidate from Equation (7.33)), which requires ~500 MB of memory at full-HD image resolution. This cost can be reduced by computing per-tile instead of per-pixel moments (an 8×8 tiling reduces the full-HD memory consumption to ~8 MB).

The overhead of our method also depends on the outcome of the optimization after the initial path-tracing (PT) pilot run. If the decision is to stick to PT, the overhead of our implementation is on average 82% (~141 ms) of the cost of tracing one path per pixel. This comprises the cost of denoising the image (the main bottleneck) and accumulating and processing the per-pixel moments for the image-level decisions.

If the decision after the PT pilot is to discard the rendering and switch to VCM, the overhead increases by the cost of the discarded PT iteration and the cost of constructing the merge mask. In our VCM implementation it amounts on average to 3.3× (~1.3 s) the cost of a single iteration of our optimized VCM. Most of that overhead is due to the filtering applied when constructing the merge mask. Hence, in our BDPT implementation, the overhead is only 1.5× (~498ms) the cost of a single iteration of our optimized BDPT.

Note that we have not spent much effort on optimizing our code. The overhead can potentially be reduced significantly: Our optimization consists solely of trivially parallelizable image processing operations that could, e.g., be run on a GPU. We leave such engineering to future work, since the results are already consistently faster than unidirectional path tracing, despite the overhead.

7.6 Moments versus variances

Our practical application benefits greatly from approximating the costly variances by the much more tractable second moments. To understand the adverse effect of that approximation, we performed tests with pre-computed moments and full variances from long renderings.

Figure 7.12 compares the result of our optimization when using second moments and when using full variances. In both cases, we pre-compute the required estimates. We use 128 iterations of VCM with our estimation scheme for the second moments, and a brute-force approach with 128 iterations of each candidate strategy to estimate the variances. The brute-force computation for the full variance is necessary because we did not find an efficient estimation scheme for the covariance, which is the dominant factor in the variance as we discuss below. Because 128 iterations is still far from enough for converged variance

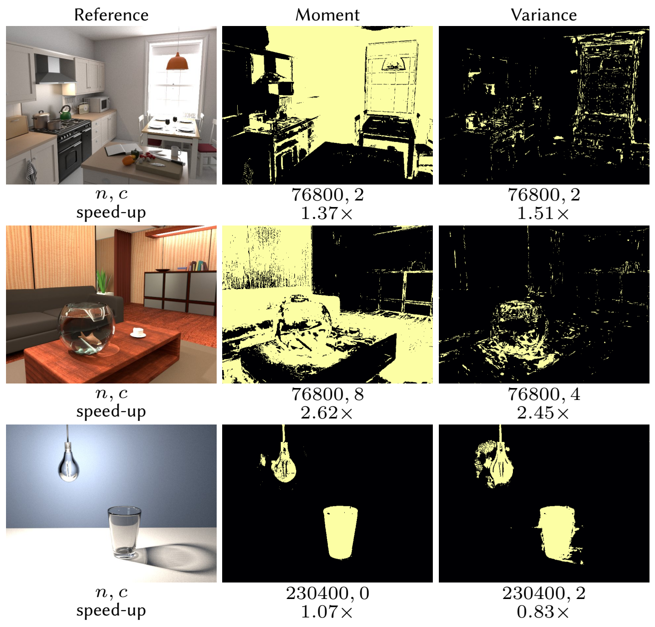


Figure 7.12: *Sample counts and resulting speed-ups compared to vanilla VCM when using pre-computed estimates of second moments or full variances. Speed-ups have to be taken with a grain of salt due to residual noise, especially for the full variance.*

estimates, we denoise the variance and moment images using OIDN [Áfra 2019].

Optimizing with the full variance improves results with high correlation. Examples can be seen in the Kitchen and Fish scenes, where the merge mask computed from the full variances is much sparser. Optimization based on second moments tends to overvalue the merit of merging. In the Fish scene, the number of connections is also lower (4 instead of 8) when using the full variance, again because of the covariance. In scenes with little correlation, like the Lamp Caustic, the results are almost identical. Note that even after 3 hours of rendering the full variances are still far from converged in this scene, so the speed-ups have to be taken with a grain of salt.

7.7 Discussion and future work

The three main limitations of our method are the overhead, error due to noisy estimates, and error due to approximations. Also, the quality of the results and the range of possible applications can be further improved by making the decisions more local, accounting for the effect of changing sampling densities (e.g., guiding), or even jointly optimizing the MIS weights [Kondapaneni et al. 2019], sampling densities [Karlík et al. 2019], and sample allocations.

Divide-and-conquer optimization. The overhead of our optimization can be reduced by reducing the number of candidate sample-allocation strategies. Additionally, the set of candidates could be altered between iterations. For example, a divide and conquer approach could start with just two candidates in the first iteration. The second iteration would then generate new candidates, e.g. by perturbing the better of the initial two. Iteratively refining the decision in this way drastically reduces the overhead, but it can also take longer to find the best strategy.

Noisy estimates. Optimizing the per-pixel merging decisions requires careful filtering, since the optimization is based on the samples of a single iteration. The estimation noise has been the biggest problem in our evaluation, with the filtering having a significant impact on the result quality. Very noisy input samples can lead to poor pixel-level decisions, which in turn could produce visible artifacts. This problem is similar to that in adaptive sampling methods, which decide on the number of samples per pixel based on variance estimates [Zwicker et al. 2015]. Future work could make our method more robust by transferring advances from adaptive sampling and reconstruction to our context. An interesting idea would be to replace our simple filtering pipeline by a learning approach, i.e., training a specialized denoising network to construct a merge mask. A simpler option is to use multiple pilot iterations to estimate the second moments before optimizing the sample counts, potentially driving that process by adaptive sampling too.

Approximation error. A key reason why our brute-force scheme works well in practice is that the full variance is approximated by the second moments. However, that is also the main source of approximation error, as we detail in the supplemental document. Only considering the second moments can lead to suboptimal sample allocations, e.g., in the presence of sample correlation. More accurate approximations, or even unbiased estimates of the full variance are apt to improve the results further. Also, we have ignored the fact that vertex merging (i.e., photon mapping) is biased. Estimating that bias and incorporating it into our objective, ideally without having to perform merging, may also improve results.

Optimization granularity. Our sample-allocation optimization is carried out in screen space, but it is possible to consider a finer granularity. For example, the number of bidirectional connections could be optimized within spatial regions in a scene. While increasing the accuracy, making decisions more localized also hazards the robustness, as fewer samples are available to compute the required quantities.

Unknown sampling densities. Optimizing the MIS sample allocation requires knowledge of the sampling densities of the different techniques. However, these densities might themselves be subject to change, or completely unknown. For example, photon emission guiding [Grittmann et al. 2018; Vorba et al. 2014] produces densities that change over time, while Markov chain Monte Carlo approaches [Šik and Křivánek 2019, 2018; Šik et al. 2016; Veach and Guibas 1997] are unable to compute the exact densities in the first place. Future work could look into approximations that predict the densities of such techniques. For example, by pretending that they are proportional to the target function [Kelemen et al. 2002].

7.8 Conclusion

We propose a method to automatically adapt the set of sampling techniques in MIS, and their sample counts, to a given input. Our application focuses on bidirectional rendering algorithms, but our method is general and applicable to any MIS combination. The key ingredient is a numerically robust and computationally efficient scheme for estimating the second moments of different sample-allocation strategies.

In practice, our adaptive VCM implementation never performs significantly worse than plain unidirectional path tracing, even on simple scenes. At the same time, complex scenes with strong indirect lighting and reflected caustics are rendered more efficiently than vanilla VCM with fixed parameters.

Rendering efficiency often relies on manual per-scene parameter tuning. The consistent speed-ups achieved by our method show that it is possible, and beneficial, to automate this tedious process.

7.A Correction factors

We are interested in estimating the second moment $M[\langle I \rangle_\mathrm{n}]$ of a candidate strategy n using the samples of another strategy m. Simply squaring and summing up the contributions of those samples yields an unbiased estimate of the second moment of m:

$$\sum_t \sum_{i=1}^{m_t} \left(\frac{w_{\mathrm{m},t}(x_{t,i}) f(x_{t,i})}{m_t p_t(x_{t,i})} \right)^2 \approx \int_\mathcal{X} \frac{f^2(x) \sum_t c_t(x) w_{\mathrm{m},t}(x)}{\sum_t c_t(x) m_t p_t(x)} \,\mathrm{d}x = M[\langle I \rangle_\mathrm{m}]. \tag{7.35}$$

To arrive at an estimator for the desired second moment $M[\langle I \rangle_\mathrm{n}]$, we write $M[\langle I \rangle_\mathrm{n}]$ in terms of the above integral but with an additional correction factor:

$$M[\langle I \rangle_\mathrm{n}] = \int_\mathcal{X} \frac{f^2(x) \sum_t c_t(x) w_{\mathrm{m},t}(x)}{\sum_t c_t(x) m_t p_t(x)} \delta_{\mathrm{n},\mathrm{m}}(x) \,\mathrm{d}x, \tag{7.36}$$

where

$$\delta_{\mathrm{n},\mathrm{m}}(x) = \frac{\sum_t c_t(x) m_t p_t(x)}{\sum_t c_t(x) n_t p_t(x)} \frac{\sum_t c_t(x) w_{\mathrm{n},t}(x)}{\sum_t c_t(x) w_{\mathrm{m},t}(x)} \tag{7.37}$$

simply replaces the sums in the numerator and the denominator with the desired ones. To obtain the desired estimator, we only need to additionally multiply each squared sample contribution by $\delta_{\mathrm{n},\mathrm{m}}(x_{t,i})$, as we do in Equation (7.23). Unfortunately, severe loss of numerical precision can be incurred when computing the involved sums of PDFs whose magnitudes can vary wildly. To that end, we can rewrite these sums in terms of MIS weights which can be evaluated in a numerically robust manner:

$$\frac{\sum_t c_t(x) m_t p_t(x)}{\sum_t c_t(x) n_t p_t(x)} = \sum_t \frac{m_t}{n_t} \frac{c_t(x) n_t p_t(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)} = \sum_t \frac{m_t}{n_t} w_{\mathrm{n},t}(x). \tag{7.38}$$

Substituting this result into Equation (7.37) yields a numerically robust expression for the correction factor:

$$\delta_{\mathrm{n},\mathrm{m}}(x) = \left(\sum_t \frac{m_t}{n_t} w_{\mathrm{n},t}(x) \right) \frac{\sum_t c_t(x) w_{\mathrm{n},t}(x)}{\sum_t c_t(x) w_{\mathrm{m},t}(x)}. \tag{7.39}$$

However, the evaluation of this expression can be inefficient: It would require computing (and storing) the weights $w_{\mathrm{n},t}$ and $w_{\mathrm{m},t}$ for every sample $x_{t,i}$, technique t, and candidate strategy n. For complex applications like bidirectional path tracing, this will quickly become expensive as each weight computation requires a full sweep over the vertices of the path represented by $x_{t,i}$. To that end, we express the MIS weights of any strategy in terms of the weights of hypothetical strategy $\mathrm{a} = (a_1, \ldots, a_T)$, with $a_t > 1 \,\forall t$, using similar manipulations as above:

$$w_{\mathrm{n},t}(x) = \frac{c_t(x) n_t p_t(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)} \tag{7.40a}$$

$$= \frac{n_t}{a_t} \frac{c_t(x) a_t p_t(x)}{\sum_{t'} c_{t'}(x) a_{t'} p_{t'}(x)} \frac{\sum_{t'} c_{t'}(x) a_{t'} p_{t'}(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)} \tag{7.40b}$$

$$= \frac{n_t}{a_t} \frac{c_t(x) a_t p_t(x)}{\sum_{t'} c_{t'}(x) a_{t'} p_{t'}(x)} \left(\sum_{t'} \frac{c_{t'}(x) n_{t'} p_{t'}(x)}{\sum_k c_k(x) a_k p_k(x)} \right)^{-1} \tag{7.40c}$$

$$= \frac{\frac{n_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_{t'} \frac{n_{t'}}{a_{t'}} w_{\mathrm{a},t'}(x)}. \tag{7.40d}$$

We substitute this result three times into Equation (7.39) to obtain

$$\delta_{\mathrm{n,m}}(x) = \left(\sum_t \frac{m_t}{n_t} \frac{\frac{n_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_{t'} \frac{n_{t'}}{a_{t'}} w_{\mathrm{a},t'}(x)}\right) \frac{\sum_t c_t(x) w_{\mathrm{n},t}(x)}{\sum_t c_t(x) w_{\mathrm{m},t}(x)} \tag{7.41a}$$

$$= \frac{\sum_t \frac{m_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_t \frac{n_t}{a_t} w_{\mathrm{a},t}(x)} \frac{\sum_t c_t(x) w_{\mathrm{n},t}(x)}{1} \frac{1}{\sum_t c_t(x) w_{\mathrm{m},t}(x)} \tag{7.41b}$$

$$= \frac{\sum_t \frac{m_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_t \frac{n_t}{a_t} w_{\mathrm{a},t}(x)} \frac{\sum_t c_t(x) \frac{n_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_t \frac{n_t}{a_t} w_{\mathrm{a},t}(x)} \frac{\sum_t \frac{m_t}{a_t} w_{\mathrm{a},t}(x)}{\sum_t c_t(x) \frac{m_t}{a_t} w_{\mathrm{a},t}(x)}, \tag{7.41c}$$

which is identical to the expression in Equation (7.24).

Chapter 8

Conclusion

Light interacts with matter in fascinating ways. These interactions cause all sorts of interesting visual phenomena, and realistic scenes contain a diverse set of many such effects. Devising a rendering algorithm that can render them all is not an easy task.

A great rendering algorithm should be efficient and general. It should be able to render any scene and any illumination effect at the highest possible performance. Preferably, such general efficiency should be attained without user intervention.

We have found that the most promising route to such an ideal rendering algorithm requires *combination* and *adaptation*. To ensure generality, it seems necessary to combine many sampling techniques; for instance, via MIS. But such a naive combination will almost never be efficient. Adaptation is paramount for efficiency. The sampling PDFs, sample counts, and even the MIS weighting functions must be adapted automatically to the scene at hand – using statistics from initial rendering samples – to achieve high efficiency.

In this thesis, we have focused on resolving issues in the MIS combination, and on adapting the sample counts. Through improved combination and adaptation, our prime application – the vertex connection and merging algorithm – achieves consistently more efficient rendering results than the current default choice, forward path tracing.

While these results are promising, there is still much to be done to find the one algorithm to render them all. Many interesting questions are yet to be answered, such as: How to best apply path guiding to a bidirectional algorithm? How to best adapt the PDFs such that they complement each other well when combined through MIS? How to decide if adaptation itself is worthwhile or too expensive? How to ensure robustness?

Appendix A

A primer on integration by substitution

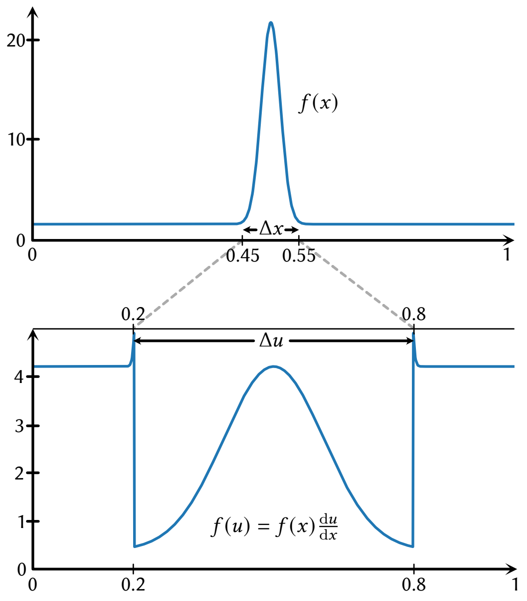


Figure A.1: *A change of variables can ease numerical integration. The top plot shows the original function, the bottom plot shows a function with identical integral but simpler shape. The change of variables distorts the domain, expanding high-value regions and compressing low-value ones. The result is a more uniform function that is significantly easier to compute numerically.*

Integration by substitution is absolutely essential for Monte Carlo rendering. It is used to define different integral formulations that motivate different types of rendering algorithms, forms the mathematical foundation of importance sampling, and is essential for multiple importance sampling.

Numerical integration is more efficient if the integrand is flat. Recall that numerical integration, be it through a deterministic Riemann sum or via randomized Monte Carlo, operates by approximating the integral through a set of boxes (rectangles in 1D). Thus, the closer the integral is to having a box shape itself, the fewer boxes are required for an accurate approximation.

A change of variables can be used to achieve a flat integrand by expanding and compressing the domain. Figure A.1 illustrates the idea on a simple example. Numerically integrating the narrow normal distribution, shown at the top, requires a large number of samples to accurately capture the peak. A change of variables can be used to expand the peak over a wider range, while compressing the tails into smaller regions. For that, the interval Δx

containing the peak is stretched to a much wider interval Δu, while the tail intervals are compressed to narrower ones. The result is a function with the same integral value, but without the difficult peak, as depicted at the bottom.

In this simplified example, the change of variables is achieved by partitioning the domain and scaling the parts based on the integrand value within. This is equivalent to importance sampling with tabulated densities, as it is used for environment map sampling [Pharr et al. 2016, Chapter 14.2.4] or path guiding [Müller et al. 2017]. But a change of variables does not have to be such a discrete partitioning. The expansion and compression of the domain can be done on a per-point basis, by growing and shrinking the differentials.

Transformation. A change of variables is defined via a transformation $t : \mathcal{U} \rightarrow \mathcal{X}$. It maps the points in the new, distorted domain $\mathcal{U}$ to points in the original domain. For example, the transformation used in Figure A.1 is a piecewise constant scaling,

$$t(u) = \begin{cases} \frac{0.45}{0.2} u & \text{if } u \in [0, 0.2] \\ 0.45 + \frac{0.1}{0.6}(u - 0.2) & \text{if } u \in (0.2, 0.8) \\ 0.55 + \frac{0.45}{0.2}(u - 0.8) & \text{if } u \in [0.8, 1] \end{cases}, \tag{A.1}$$

where the numbers reflect the change in proportions: $\frac{0.45}{0.2}$ is the scaling factor applied to the interval containing the tails and $\frac{0.1}{0.6}$ the scaling applied to the interval with the peak.

Preserving the area. The key to a correct change of variables is to retain the same integral value. The naïve transformation,

$$\int_{\mathcal{U}} f(t(u))\,\mathrm{d}u \neq \int_{\mathcal{X}} f(x)\,\mathrm{d}x, \tag{A.2}$$

would not achieve that. Figure A.2 visualizes why on a simple constant function. There, the interval $\mathcal{U} = [0, 2]$ is mapped to the half as wide interval $\mathcal{X} = [0, 1]$ via the transformation $t(u) = 0.5y$. Without altering the integrand, the area under the curve is doubled. This holds for any function $f(x)$, not just the simple constant depicted here. Fortunately, the solution is quite simple: To retain the same integral value, we must shrink the height proportionally, as shown on the right. This is achieved by multiplying the integrand by the scaling applied to the domain,

$$\int_0^2 f\left(\frac{u}{2}\right)\frac{1}{2}\,\mathrm{d}u = \int_0^1 f(x)\,\mathrm{d}x. \tag{A.3}$$

Note that this scaling is independent of the function shape and holds for non-constant functions, too.

Non-uniform transformations. Naturally, a change of variables that meerly stretches the whole domain uniformly is not very useful[1]. But the reasoning to retain the integral value can be extended to arbitrary distortions. As a first step, we can look at scaling applied to some interval Δx. If this interval is mapped to another interval Δu through a uniform scale within this portion of the domain – like in the example in Figure A.1 – then the integrand over that interval must be scaled by the ratio $\frac{\Delta x}{\Delta u}$. In our simple example, this

[1]Although it is encountered frequently in rendering; for example, when mapping uniform random numbers from primary space to a uniform distribution in another domain.

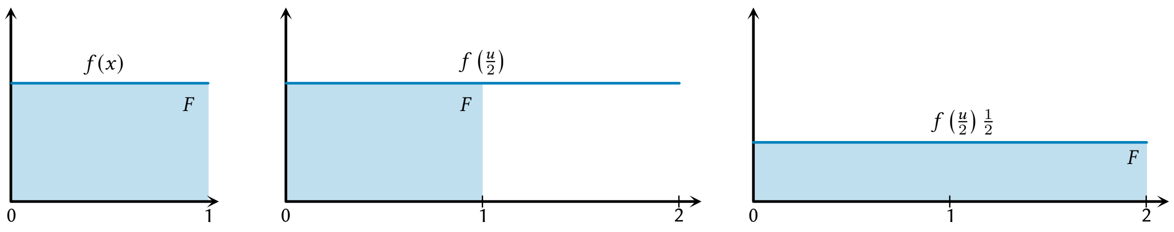


Figure A.2: *Why the integrand must be scaled if a change of variables is performed. In this simple example, a constant function is integrated, and the change of variables maps a twice as wide interval to the original domain. Without altering the integrand, the area under the curve, shown in blue, would be doubled. Scaling the integrand by the Jacobian (here, 0.5) retains the correct area, that is, the same integral value.*

means we must scale by

$$\frac{\Delta x}{\Delta u} = \begin{cases} \frac{0.45}{0.2} & \text{if } u \in [0, 0.2] \\ \frac{0.1}{0.6} & \text{if } u \in (0.2, 0.8) \\ \frac{0.45}{0.2} & \text{if } u \in [0.8, 1] \end{cases}, \tag{A.4}$$

that is, we compute

$$\int_{\mathcal{U}} f(t(u)) \frac{\Delta x}{\Delta u} \, \mathrm{d}u = \int_{\mathcal{X}} f(x) \, \mathrm{d}x \tag{A.5}$$

This logic can be extended to continuous scaling by considering an infinitely small Δx; namely, a differential $\mathrm{d}x$. Following the same argument, the integrand at every point must be scaled by the ratio of differentials

$$\int_{\mathcal{U}} f(t(u)) \frac{\mathrm{d}x}{\mathrm{d}u} \, \mathrm{d}u = \int_{\mathcal{X}} f(x) \, \mathrm{d}x. \tag{A.6}$$

Intuitively, this makes sense because we must "cancel out" the $\mathrm{d}u$ and replace it by the original $\mathrm{d}x$ to obtain the same integral.

Jacobian. For a 1D function, the ratio of differentials

$$\frac{\mathrm{d}x}{\mathrm{d}u} = t'(u) \tag{A.7}$$

is simply the derivative of the mapping. In the general multivariate setting, the ratio is given by the determinant of the Jacobian matrix (aka the Jacobian),

$$\frac{\mathrm{d}x}{\mathrm{d}u} = |J_t(u)|. \tag{A.8}$$

Appendix B

Deriving the geometry term

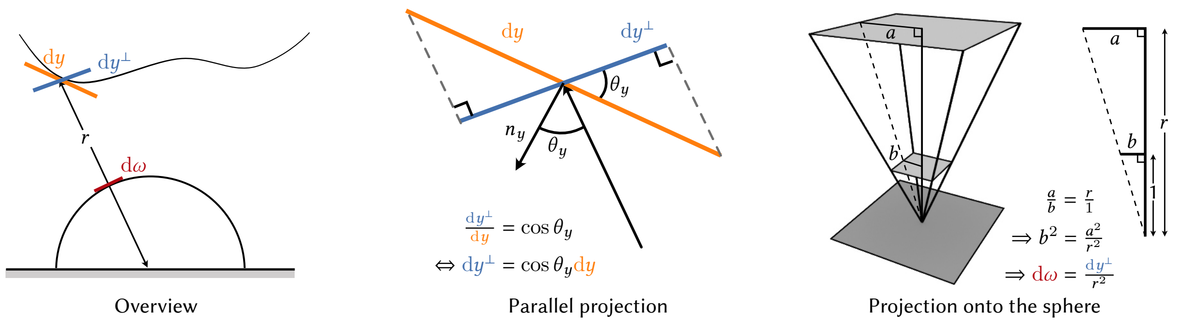
An essential transformation in rendering algorithms is the change of variables from surfaces to directions, or vice versa. For example, forward path tracing makes extensive use of this change of variables to facilitate next event estimation. Further, transforming between directions and surface points is the very foundation of bidirectional algorithms. Therefore, the corresponding Jacobian,

$$\frac{\mathrm{d}\omega}{\mathrm{d}y} = \frac{\cos\theta_y}{\|y - x\|^2}, \tag{B.1}$$

can be found in countless places. At the same time, it is surprisingly hard to find a derivation of this crucial term in the literature.

Figure B.1 shows how this Jacobian can be derived geometrically. The image on the left depicts the setup. A point $y \in \mathcal{A}$ is given on a surface in the scene. This point is transformed to a direction $\omega = \frac{y-x}{\|x-y\|}$ at a reference point x. For a change of variables – or importance sampling – we need to know how the corresponding differentials $\mathrm{d}y$ and $\mathrm{d}\omega$ relate to each other. That is, how much a small change in the position y changes the direction ω. Geometrically, this can be viewed as computing the ratio between the area of an infinitely small rectangle $\mathrm{d}y$ around y, and the area of its projection $\mathrm{d}\omega$ onto the tangent plane on the sphere of directions. Note that the differentials can be viewed as rectangles because the surface $\mathcal{A}$ and the sphere of directions are manifolds; that is, they are locally planar.

The first step is to project the tangent plane (shown in orange), of which $\mathrm{d}y$ is an infinitely small subset, onto the plane perpendicular to the direction ω (shown in blue). The area of the projection can be derived with some basic trigonometry, as sketched in the center



$\mathrm{d}y$
$\mathrm{d}y^\perp$
r
$\mathrm{d}\omega$
Overview
$\mathrm{d}y$
$\mathrm{d}y^\perp$
θ_y
n_y
θ_y
$\frac{\mathrm{d}y^\perp}{\mathrm{d}y} = \cos\theta_y$
$\Leftrightarrow \mathrm{d}y^\perp = \cos\theta_y \mathrm{d}y$
Parallel projection
a
b
a
b
r
1
$\frac{a}{b} = \frac{r}{1}$
$\Rightarrow b^2 = \frac{a^2}{r^2}$
$\Rightarrow \mathrm{d}\omega = \frac{\mathrm{d}y^\perp}{r^2}$
Projection onto the sphere

Figure B.1: *Derivation of the geometry term. The surface differential* $\mathrm{d}y$ *is defined on the tangent plane of the scene geometry (orange). First, we project it to the plane perpendicular to the ray direction (blue). Then, we project that down to the unit sphere (red).*

of Figure B.1. For that, we compute the angle θ_y between the surface normal n_y and the direction ω. The projected differential area is then scaled by the cosine of that angle,

$$\mathrm{d}y^{\perp} = \cos\theta_y \mathrm{d}y. \tag{B.2}$$

Next, the corresponding area of the differential on the sphere of directions can be computed. The perpendicular plane is parallel to the tangent plane on the sphere of directions at ω. Therefore, the projection of $\mathrm{d}y^{\perp}$ onto the sphere can also be found with basic trigonometry, as sketched on the right of Figure B.1. To compute the change in size, we can identify the similar triangles sketched on the very right of the figure. These show that the width and height of the two rectangles are each scaled by the distance $r = ||x - y||$ between x and y. Therefore, the differential solid angle is the differential perpendicular area divided by the squared distance,

$$\mathrm{d}\omega = \frac{1}{||x - y||^2}\mathrm{d}y^{\perp}. \tag{B.3}$$

Finally, combining both equations, we see that the differentials relate as

$$\mathrm{d}\omega = \frac{1}{||x - y||^2}\cos\theta_y \mathrm{d}y, \tag{B.4}$$

providing us with the desired ratio, that is, the Jacobian of the projection.

Appendix C

Jacobian for perspective cameras

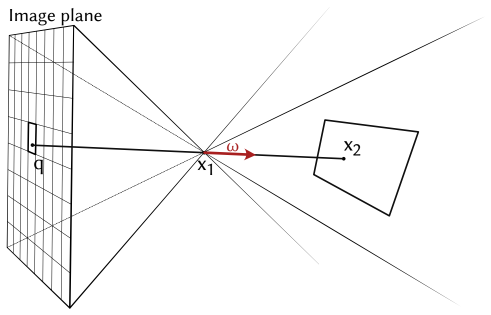
The light tracing technique connects light paths directly to the camera. For that, it requires an additional quantity: the Jacobian of the mapping from surface positions to image plane positions. Here, we look at an example of how this Jacobian can be derived for the simple pinhole camera model.

The pinhole camera model is illustrated on the left of Figure C.1. The model assumes an extreme case of a *camera obscura* where the hole that receives light is an infinitely small point – the aperture position x_1 in our notation. The light traveling through that hole is received by a rectangular image positioned behind the hole.

This camera model can be parameterized through the (horizontal and vertical) opening angle γ and the focal length f, that is, the distance between the image and the hole. This is illustrated on a cross-section in the center of the figure. The size of the image plane in this parameterization is then computed via the tangent of the opening angle, $\tan\gamma$.

The value of each pixel in the rendered image is generally defined through some response function over this image plane. For example, each pixel could measure the light received by a corresponding small rectangle on the image plane, as sketched in Figure C.1. This is a common model used in rendering. Note that this is quite far away from an actual camera sensor in the real world, where different colors are measured at different positions and combined to form the values of individual pixels through elaborate post-processing.

Forward path tracing starts a path by first sampling a position q on the image plane that contributes to the current pixel; for example, by importance sampling the pixel filter. Then, a position x_1 is sampled on the aperture. For the ideal pinhole camera, this position is



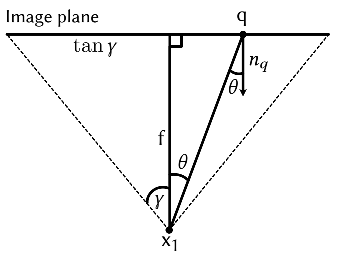


Figure C.1: *Schematic of the pinhole camera model, showing all quantities and relationships required to compute the Jacobian.*

deterministic and always the same. Together, the image position and aperture position define a direction ω from x_1 into the scene. With the pinhole camera, this direction is simply

$$\omega = \frac{x_1 - q}{\|x_1 - q\|}. \tag{C.1}$$

The visible surface point x_2 is found by tracing a ray from x_1 in direction ω.

Given an image position q with probability density $p(q)$, our goal is to find the corresponding surface density $p(x_2)$ due to the above mapping. The corresponding Jacobian, that is, the ratio of differentials, can be written as

$$\frac{\mathrm{d}q}{\mathrm{d}x_2} = \frac{\mathrm{d}q}{\mathrm{d}\omega}\frac{\mathrm{d}\omega}{\mathrm{d}x_2}, \tag{C.2}$$

that is, the product of the Jacobian for the mapping from image position to direction, and the Jacobian for the mapping from directions to surface points.

The former can be computed from the quantities and relations depicted on the right of Figure C.1. For that, we denote the surface normal of the image plane, also known as the view direction of the camera, as n_q. The angle between that normal and the direction ω is denoted as θ. Then, the Jacobian is simply a special case of the surface-to-direction Jacobian derived in Appendix B,

$$\frac{\mathrm{d}q}{\mathrm{d}\omega} = \frac{\cos\theta}{\|q - x_1\|^2}. \tag{C.3}$$

We can specialize this equation further by noting that

$$\cos\theta = \frac{f}{\|q - x_1\|} \Leftrightarrow \|q - x_1\| = \frac{f}{\cos\theta}, \tag{C.4}$$

since f is the length of the adjacent edge and $\|q - x_1\|$ that of the hypotenuse in the right triangle shown in Figure C.1. Substituting (C.4) into (C.3), we can simplify this to:

$$\frac{\mathrm{d}q}{\mathrm{d}\omega} = \frac{\cos^3\theta}{f}. \tag{C.5}$$

Therefore, the full Jacobian is

$$\boxed{|J_{\mathrm{cam}}| = \frac{\cos^3\theta}{f}\frac{\|x_2 - x_1\|^2}{\cos\theta(x_2 \to x_1)}.} \tag{C.6}$$

Bibliography

Attila T. Áfra. 2019. *Intel® Open Image Denoise.* (2019). https://www.openimagedenoise.org/.

Abdalla GM Ahmed and Peter Wonka. 2020. “Screen-space blue-noise diffusion of Monte Carlo sampling error via hierarchical ordering of pixels”. *ACM Trans. Graph. (SIGGRAPH Asia 2020)*, 39, 6, Article 244, 15 pages.

James Richard Arvo and David Kirk. 1990. “Particle Transport and Image Synthesis”. *Computer Graphics (SIGGRAPH 1990)*, 24, 4, 63–66.

Thomas Bashford-Rogers, Kurt Debattista, and Alan Chalmers. 2012. “A significance cache for accelerating global illumination”. *Comput. Graph. Forum*, 31, 6, 1837–1851.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. “Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting”. *ACM Trans. Graph. (SIGGRAPH 2020)*, 39, 4, Article 148, 17 pages.

Mark R Bolin and Gary W Meyer. 1997. “An error metric for Monte Carlo ray tracing”. In: *Rendering Techniques (Eurographics Workshop on Rendering)*. Springer, 57–68.

Vassillen Chizhov, Iliyan Georgiev, Karol Myszkowski, and Gurprit Singh. 2022. “Perceptual error optimization for Monte Carlo rendering”. *ACM Trans. Graph.*, 41, 3, Article 26, 17 pages.

David Cline, Justin Talbot, and Parris Egbert. 2005. “Energy redistribution path tracing”. *ACM Trans. Graph. (SIGGRAPH 2005)*, 24, 3, 1186–1195.

Robert L Cook, Thomas Porter, and Loren Carpenter. 1984. “Distributed ray tracing”. 18, 3, 137–145.

Miguel Crespo, Adrian Jarabo, and Adolfo Muñoz. 2021. “Primary-space adaptive control variates using piecewise-polynomial approximations”. *ACM Trans. Graph.*, 40, 3, Article 25, 15 pages.

Tomáš Davidovič and Iliyan Georgiev. 2012. *SmallVCM.* GitHub. (2012). https://github.com/SmallVCM/SmallVCM.

Tomáš Davidovič, Jaroslav Křivánek, Miloš Hašan, and Philipp Slusallek. 2014. “Progressive light transport simulation on the GPU: Survey and improvements”. *ACM Trans. Graph.*, 33, 3, Article 29, 19 pages.

Michael Donikian, Bruce Walter, Kavita Bala, Sebastian Fernandez, and Donald P Greenberg. 2006. “Accurate direct illumination using iterative adaptive sampling”. *IEEE Trans. Vis. Comput. Graph*, 12, 3, 353–364.

Alejandro Conty Estevez and Christopher Kulla. 2020. “Practical Caustics Rendering with Adaptive Photon Guiding”. In: *SIGGRAPH 2020 Talks* Article 17, 2 pages.

Shaohua Fan, Stephen Chenney, Bo Hu, Kam-Wah Tsui, and Yu-chi Lai. 2006. “Optimizing control variate estimators for rendering”. In: *Comput. Graph. Forum* 3. Vol. 25. Wiley Online Library, 351–357.

Luca Fascione, Johannes Hanika, Rob Pieké, Ryusuke Villemin, Christophe Hery, Manuel Gamito, Luke Emrose, and André Mazzone. 2018. “Path Tracing in Production”. In: *SIGGRAPH 2018 Courses* (SIGGRAPH ’18) Article 15. ACM, Vancouver, British Columbia, Canada, 79 pages.

Iliyan Georgiev. 2012. *Implementing Vertex Connection and Merging*. Tech. rep. Saarland University. http://www.iliyan.com/publications/ImplementingVCM.

Iliyan Georgiev and Marcos Fajardo. 2016. “Blue-noise Dithered Sampling”. *ACM SIGGRAPH 2016 Talks*, Article 35, 1 pages.

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012a. “Light transport simulation with vertex connection and merging.” *ACM Trans. Graph. (SIGGRAPH Asia 2012)*, 31, 6, Article 192, 10 pages.

Iliyan Georgiev, Jaroslav Křivánek, Stefan Popov, and Philipp Slusallek. 2012b. “Importance caching for complex illumination”. *Comput. Graph. Forum (EG 2012)*, 31, 2pt3, 701–710.

Peter W Glynn and Ward Whitt. 1992. “The asymptotic efficiency of simulation estimators”. *Operations research*, 40, 3, 505–520.

Pascal Grittmann. 2020. *SeeSharp*. GitHub. (2020). https://github.com/pgrit/SeeSharp.

Pascal Grittmann, Iliyan Georgiev, and Philipp Slusallek. 2021. “Correlation-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms”. *Comput. Graph. Forum (EG 2021)*, 40, 2, 231–238.

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. 2019. “Variance-Aware Multiple Importance Sampling”. *ACM Trans. Graph. (SIGGRAPH Asia 2019)*, 38, 6, Article 152, 9 pages.

Pascal Grittmann, Arsène Pérard-Gayot, Philipp Slusallek, and Jaroslav Křivánek. 2018. “Efficient Caustic Rendering with Lightweight Photon Mapping”. *Comput. Graph. Forum (EGSR ’18)*, 37, 4, 133–142.

Pascal Grittmann, Ömercan Yazici, Iliyan Georgiev, and Philipp Slusallek. 2022. “Efficiency-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms”. *ACM Trans. Graph. (SIGGRAPH 2022)*, 41, 4, Article 80, 12 pages.

Toshiya Hachisuka, Wojciech Jarosz, and Henrik Wann Jensen. 2010. “A progressive error estimation framework for photon density estimation”. *ACM Trans. Graph. (SIGGRAPH 2010)*, 29, 6, Article 144, 12 pages.

Toshiya Hachisuka, Anton S Kaplanyan, and Carsten Dachsbacher. 2014. “Multiplexed metropolis light transport”. *ACM Trans. Graph. (SIGGRAPH 2014)*, 33, 4, Article 100, 10 pages.

Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. 2008. “Progressive photon mapping”. In: *ACM Trans. Graph. (SIGGRAPH Asia 2008)* 5, Article 130. Vol. 27. ACM, 8 pages.

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. “A path space extension for robust light transport simulation”. *ACM Trans. Graph. (SIGGRAPH Asia 2012)*, 31, 6, Article 191, 10 pages.

J.M. Hammersley and D.C. Handscomb. 1968. *Monte Carlo Methods*. Springer.

Johannes Hanika, Marc Droske, and Luca Fascione. 2015. “Manifold next event estimation”. 34, 4, 87–97.

Jon Hasselgren, Jacob Munkberg, Marco Salvi, Anjul Patney, and Aaron Lefohn. 2020. “Neural temporal adaptive sampling and denoising”. In: *Comput. Graph. Forum (EG 2020)* 2. Vol. 39. Wiley Online Library, 147–155.

Vlastimil Havran and Mateu Sbert. 2014. “Optimal Combination of Techniques in Multiple Importance Sampling”. In: *Proc. VRCAI ’14*. ACM, Shenzhen, China, 141–150.

Eric Heitz. 2020. “Can’t Invert the CDF? The Triangle-Cut Parameterization of the Region under the Curve”. *Comput. Graph. Forum (EGSR 2020)*, 39, 4, 121–132.

Eric Heitz and Laurent Belcour. 2019. “Distributing monte carlo errors as a blue noise in screen space by permuting pixel seeds between frames”. *Comput. Graph. Forum (EGSR 2019)*, 38, 4, 149–158.

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”. In: *ACM SIGGRAPH 2019 Talks*, 1–2.

Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik P. A. Lensch, and Jaroslav Křivánek. 2016. “Product Importance Sampling for Light Transport Path Guiding”. *Comput. Graph. Forum (EGSR 2016)*, 35, 67–77.

Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik P. A. Lensch, and Jaroslav Křivánek. 2019. “Volume Path Guiding Based on Zero-Variance Random Walk Theory”. *ACM Trans. Graph.*, 38, 3, Article 25, 19 pages.

Heinrich Hey and Werner Purgathofer. 2002. “Importance Sampling with Hemispherical Particle Footprints”. In: *Proceedings of the 18th Spring Conference on Computer Graphics* (SCCG ’02). ACM, Budmerice, Slovakia, 107–114.

Qingqin Hua, Pascal Grittmann, and Philipp Slusallek. 2023. “Revisiting Controlled Mixture Sampling for Rendering Applications”. *ACM Trans. Graph. (SIGGRAPH 2023)*, 42, 4.

Wenzel Jakob and Steve Marschner. 2012. “Manifold exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport”. *ACM Trans. Graph. (SIGGRAPH 2012)*, 31, 4, Article 58, 13 pages.

Johannes Jendersie. 2019. “Variance Reduction via Footprint Estimation in the Presence of Path Reuse”. In: *Ray Tracing Gems*. Springer, 557–569.

Johannes Jendersie and Thorsten Grosch. 2018. “An Improved Multiple Importance Sampling Heuristic for Density Estimates in Light Transport Simulations.” In: *EGSR 2018 (EI&I)*, 65–72.

Henrik Wann Jensen. 1996. “Global illumination using photon maps”. In: *Rendering Techniques (Eurographics Workshop on Rendering)*. Springer, 21–30.

Henrik Wann Jensen. 1995. “Importance Driven Path Tracing using the Photon Map”. In: *Rendering Techniques (Eurographics Workshop on Rendering)*, 326–335.

Henrik Wann Jensen. 2001. *Realistic image synthesis using photon mapping*. Ak Peters Natick.

James T. Kajiya. 1986. “The Rendering Equation”. *Computer Graphics (SIGGRAPH 1986)*, 20, 4, 143–150.

Anton S Kaplanyan and Carsten Dachsbacher. 2013a. “Adaptive progressive photon mapping”. *ACM Trans. Graph.*, 32, 2, Article 16, 13 pages.

Anton S Kaplanyan and Carsten Dachsbacher. 2013b. “Path space regularization for holistic and robust light transport”. In: *Comput. Graph. Forum (EG 2013)* 2pt1. Vol. 32. Wiley Online Library, 63–72.

Ondřej Karlík, Martin Šik, Petr Vévoda, Tomáš Skřivan, and Jaroslav Křivánek. 2019. “MIS Compensation: Optimizing Sampling Techniques in Multiple Importance Sampling”. *ACM Trans. Graph. (SIGGRAPH Asia 2019)*, 38, 6, Article 151, 12 pages.

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. “A simple and robust mutation strategy for the metropolis light transport algorithm”. In: *Comput. Graph. Forum (EG 2002)* 3. Vol. 21. Wiley Online Library, 531–540.

Alexander Keller. 1997. “Instant radiosity”. In: *Annual Conference Series (SIGGRAPH 1997)*, 49–56.

Alexander Keller. 2013. “Quasi-Monte Carlo image synthesis in a nutshell”. In: *Monte Carlo and Quasi-Monte Carlo Methods 2012*. Springer, 213–249.

Alexander Keller, Ken Dahm, and Nikolaus Binder. 2014. “Path space filtering”. In: *ACM SIGGRAPH 2014 Talks*, 1 pages.

Alexander Keller, Iliyan Georgiev, Abdalla Ahmed, Per Christensen, and Matt Pharr. 2019. “My Favorite Samples”. In: *ACM SIGGRAPH 2019 Courses.* ACM, Los Angeles, California, USA.

David Kirk and James Arvo. 1991. “Unbiased Sampling Techniques for Image Synthesis”. *ACM Trans. Graph. (SIGGRAPH 1991)*, 25, 4, 153–156.

Ivo Kondapaneni, Petr Vévoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 2019. “Optimal Multiple Importance Sampling”. *ACM Trans. Graph. (SIGGRAPH 2019)*, 38, 4, Article 37, 14 pages.

Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. 2014. “Unifying Points, Beams, and Paths in Volumetric Light Transport Simulation”. *ACM Trans. Graph. (SIGGRAPH 2014)*, 33, 4, Article 103.

Alexandr Kuznetsov, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2018. “Deep adaptive sampling for low sample count rendering”. *Comput. Graph. Forum (EGSR 2018)*, 37, 4, 35–44.

Eric P. Lafortune and Yves D. Willems. 1993. “Bi-Directional Path Tracing”. 93, 145–153.

Zehui Lin, Sheng Li, Xinlu Zeng, Congyi Zhang, Jinzhu Jia, Guoping Wang, and Dinesh Manocha. 2020. “CPPM: chi-squared progressive photon mapping”. *ACM Trans. Graph. (SIGGRAPH Asia 2020)*, 39, 6, Article 240, 12 pages.

Heqi Lu, Romain Pacanowski, and Xavier Granier. 2013. “Second-Order Approximation for Variance Reduction in Multiple Importance Sampling”. *Comput. Graph. Forum (Pacific Graphics 2013)*, 32, 7, 131–136.

Thomas Müller. 2019. ““Practical Path Guiding” in Production”. In: *ACM SIGGRAPH Courses: Path Guiding in Production, Chapter 10.* ACM, Los Angeles, California, 18:35–18:48.

Thomas Müller, Markus H. Gross, and Jan Novák. 2017. “Practical Path Guiding for Efficient Light-Transport Simulation”. *Comput. Graph. Forum (EGSR 2017)*, 36, 91–100.

Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. “Neural Importance Sampling”. *ACM Trans. Graph.*, 38, 5, Article 145, 19 pages.

Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. 2020. “Neural control variates”. *ACM Trans. Graph.*, 39, 6, Article 243, 19 pages.

David Murray, Sofiane Benzait, Romain Pacanowski, and Xavier Granier. 2020. “On Learning the Best Balancing Strategy”. In: *Eurographics 2020.* Vol. 20, 4 pages.

Kosuke Nabata, Kei Iwasaki, and Yoshinori Dobashi. 2020. “Resampling-aware Weighting Functions for Bidirectional Path Tracing Using Multiple Light Sub-Paths”. *ACM Trans. Graph.*, 39, 2, Article 15, 11 pages.

Art Owen and Yi Zhou. 2000. “Safe and Effective Importance Sampling”. *Journal of the American Statistical Association*, 95, 449, 135–143.

Anthony Pajot, Loic Barthe, Mathias Paulin, and Pierre Poulin. 2010. “Representativity for robust and adaptive multiple importance sampling”. *IEEE Trans. Vis. Comput. Graph*, 17, 8, 1108–1121.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically based rendering: From theory to implementation.* Morgan Kaufmann. https://www.pbr-book.org/3ed-2018/.

Stefan Popov, Ravi Ramamoorthi, Fredo Durand, and George Drettakis. 2015. “Probabilistic connections for bidirectional path tracing”. In: *Comput. Graph. Forum (EGSR 2015)* 4. Vol. 34. Wiley Online Library, 75–86.

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. 2020. “Variance-Aware Path Guiding”. *ACM Trans. Graph. (SIGGRAPH 2020)*, 39, 4, Article 151, 12 pages.

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Philippe Weier, and Philipp Slusallek. 2022. “EARS: Efficiency-Aware Russian Roulette and Splitting”. *ACM Trans. Graph. (SIGGRAPH 2022)*, 41, 4, Article 81, 14 pages.

Florian Reibold, Johannes Hanika, Alisa Jung, and Carsten Dachsbacher. 2018. “Selective guided sampling with complete light transport paths”. *ACM Trans. Graph. (SIGGRAPH Asia 2018)*, 37, 6, Article 223, 14 pages.

Sheldon M Ross. 2014. *Introduction to probability models.* Academic press.

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. “Image-space control variates for rendering”. *ACM Trans. Graph. (SIGGRAPH Asia 2016)*, 35, 6, Article 169, 12 pages.

Lukas Ruppert, Sebastian Herholz, and Hendrik PA Lensch. 2020. “Robust fitting of parallax-aware mixtures for path guiding”. *ACM Trans. Graph. (SIGGRAPH 2020)*, 39, 4, Article 147, 15 pages.

Corentin Salaün, Adrien Gruson, Binh-Son Hua, Toshiya Hachisuka, and Gurprit Singh. 2022. “Regression-based Monte Carlo integration”. *ACM Trans. Graph. (SIGGRAPH 2022)*, 41, 4, Article 79, 14 pages.

Mateu Sbert and Vlastimil Havran. 2017. “Adaptive multiple importance sampling for general functions”. *The Visual Computer*, 33, 6, 845–855.

Mateu Sbert, Vlastimil Havran, and László Szirmay-Kalos. 2019. “Optimal Deterministic Mixture Sampling.” In: *Eurographics (Short Papers)*, 73–76.

Mateu Sbert, Vlastimil Havran, and László Szirmay-Kalos. 2016. “Variance Analysis of Multi-sample and One-sample Multiple Importance Sampling”. In: *Comput. Graph. Forum* 7. Vol. 35. Wiley Online Library, 451–460.

Mateu Sbert, Vlastimil Havran, and Laszlo Szirmay-Kalos. 2018a. “Multiple importance sampling revisited: breaking the bounds”. *EURASIP Journal on Advances in Signal Processing*, 2018, 1, 1–15.

Mateu Sbert, Vlastimil Havran, László Szirmay-Kalos, and Víctor Elvira. 2018b. “Multiple importance sampling characterization by weighted mean invariance”. *The Visual Computer*, 34, 6, 843–852.

Vincent Schüßler, Johannes Hanika, Alisa Jung, and Carsten Dachsbacher. 2022. “Path Guiding with Vertex Triplet Distributions”. *Comput. Graph. Forum (EGSR 2022)*, 41, 4, 1–15.

Martin Šik and Jaroslav Křivánek. 2019. “Implementing One-Click Caustics in Corona Renderer”. In: *EGSR 2019 Industry Papers.* The Eurographics Association.

Martin Šik and Jaroslav Křivánek. 2018. “Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation”. *IEEE Trans. Vis. Comput. Graph*, 26, 4, 1821–1840.

Martin Šik, Hisanari Otsu, Toshiya Hachisuka, and Jaroslav Křivánek. 2016. “Robust light transport simulation via metropolised bidirectional estimators”. *ACM Trans. Graph. (SIGGRAPH Asia 2016)*, 35, 6, Article 245, 12 pages.

Fujia Su, Sheng Li, and Guoping Wang. 2022. “SPCBPT: subspace-based probabilistic connections for bidirectional path tracing”. *ACM Trans. Graph. (SIGGRAPH 2022)*, 41, 4, Article 77, 14 pages.

Justin F. Talbot, David Cline, and Parris Egbert. 2005. “Importance Resampling for Global Illumination”. In: *Rendering Techniques* (EGSR ’05). Eurographics Association, 139–146.

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation.* Stanford University PhD thesis.

Eric Veach and Leonidas Guibas. 1995a. “Bidirectional estimators for light transport”. In: *Photorealistic Rendering Techniques.* Springer, 145–167.

Eric Veach and Leonidas Guibas. 1997. “Metropolis light transport”. In: *SIGGRAPH 1997.* ACM, 65–76.

Eric Veach and Leonidas Guibas. 1995b. “Optimally Combining Sampling Techniques for Monte Carlo Rendering”. In: *SIGGRAPH 1995.* ACM, 419–428.

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. “Bayesian online regression for adaptive direct illumination sampling”. *ACM Trans. Graph. (SIGGRAPH 2028)*, 37, 4, Article 125, 12 pages.

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. “Path Guiding in Production”. In: *ACM SIGGRAPH 2019 Courses* (SIGGRAPH 2019) Article 18. ACM, Los Angeles, California, 77 pages.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. “On-line Learning of Parametric Mixture Models for Light Transport Simulation”. *ACM Trans. Graph. (SIGGRAPH 2014)*, 33, 4, Article 101, 11 pages.

Jiří Vorba and Jaroslav Křivánek. 2016. “Adjoint-driven Russian Roulette and Splitting in Light Transport Simulation”. *ACM Trans. Graph.*, 35, 4, Article 42, 11 pages.

Yu-Chen Wang, Yu-Ting Wu, Tzu-Mao Li, and Yung-Yu Chuang. 2021. “Learning to cluster for rendering with many lights”. *ACM Trans. Graph. (SIGGRAPH Asia 2021)*, 40, 6, Article 277, 10 pages.

Rex West, Iliyan Georgiev, Adrien Gruson, and Toshiya Hachisuka. 2020. “Continuous multiple importance sampling”. *ACM Trans. Graph. (SIGGRAPH 2020)*, 39, 4, Article 136, 12 pages.

Alan Wolfe, Nathan Morrical, Tomas Akenine-Möller, and Ravi Ramamoorthi. 2022. “Spatiotemporal Blue Noise Masks”. In: *EGSR 2022 Conference.*

Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. 2020. “Specular manifold sampling for rendering high-frequency caustics and glints”. *ACM Trans. Graph. (SIGGRAPH 2020)*, 39, 4, Article 149, 15 pages.

Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. 2021. “Monte Carlo estimators for differential light transport”. *ACM Trans. Graph. (SIGGRAPH 2021)*, 40, 4, Article 78, 16 pages.

Cheng Zhang, Zhao Dong, Michael Doggett, and Shuang Zhao. 2021. “Antithetic sampling for Monte Carlo differentiable rendering”. *ACM Trans. Graph. (SIGGRAPH 2021)*, 40, 4, Article 77, 12 pages.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. 2015. “Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering”. In: *Comput. Graph. Forum* 2. Vol. 34. Wiley Online Library, 667–681.