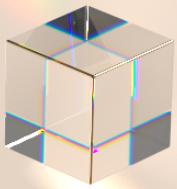
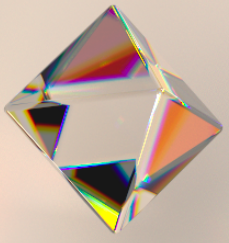
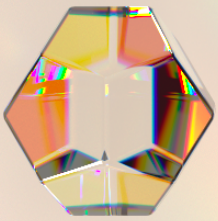
Pre-Estimated Spectral Rendering

Mark van de Ruit







TU Delft

Pre-Estimated Spectral Rendering

by

Mark van de Ruit

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday October 22, 2019 at 10:00 AM.

Student number:	4228723	
Project duration:	November 13, 2018 – October 22, 2019	
Thesis committee:	Prof. dr. E. Eisemann,	TU Delft, supervisor
	Dr. R. Marroquim,	TU Delft
	Dr. W. Ruszel,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

Spectral Monte Carlo renderers are capable of reproducing several advanced phenomena of light, such as chromatic dispersion and fluorescence. As spectral renderers must sample the spectral domain, they are typically hampered by a multitude of sampling issues leading to notably poor convergence rates, which are reinforced when realistic emission or reflectance spectra are involved in otherwise simple scenes. We propose *pre-estimated spectral rendering*, which is a simple method that iteratively builds estimates of spectral radiance distributions before rendering, and subsequently uses these to efficiently guide importance sampling of the spectral domain. Our method significantly lowers variance and reduces chromatic noise with little overhead in multiple difficult scenarios, which we demonstrate with an implementation in a conventional renderer.

Preface

How many cups of coffee does a student drink while working on their masters thesis? It is a shame that I did not keep track of this truly wonderful statistic, though I imagine the number is approximately equal to the time I spent waiting on renders. Fortunately, the coffee was not wasted, and I can honestly say I am more than satisfied with my thesis and the work assembled within.

Of course, this work would not exist without the enthousiastic support and supply of ideas provided by my supervisor: prof. dr. Elmar Eisemann, to whom I extend my sincerest thanks. I would also like to thank Jerry Guo and Victor Petitjean for their brief insights; Ricardo Marroquim and Wioletta Ruszel for their willingness to form my thesis committee; and Bart de Jonge for his rigorous late-night proofreading.

In addition, I must acknowledge Matt Pharr, Wenzel Jakob, and Greg Humphreys, whose collective work on PBRT [22] and the Mitsuba renderer [35] has likely spared hundreds of researchers thousands of hours debugging broken Halton samplers, and Pablo Bauszat, who together with Victor Petitjean extended Mitsuba for spectral rendering.

Finally, I would like to thank my family and friends from the bottom of my heart, as they were there to make sure I got some sunlight, did too much bouldering, played long tabletop games, and saw a movie from time to time.

Mark van de Ruit
Delft, October 2019

Contents

List of Symbols

λ	Wavelength (nm)
Λ	Spectral domain of wavelengths
$S(\lambda), R(\lambda)$	Spectral power distribution, spectral reflectance distribution
X	Random variable
x	Realization of random variable, alternatively point in 3D space
$P(X)$	Cumulative distribution function
$p(x)$	Probability density function
$E[X]$	Expected value
$\sigma[X]$	Standard deviation
$V[X]$	Variance
Ω	Arbitrary integration domain, alternatively path space, alternatively image filter neighborhood
$\hat{I}$	(Monte Carlo) Estimator of a value I
N	Number of samples
$\beta[\hat{I}]$	Bias of an estimator $\hat{I}$
$MSE[\hat{I}]$	Mean squared error of an estimator $\hat{I}$
$Unif[0,1]$	Uniform distribution over a domain $[0,1]$
I_j	Measured radiance for a pixel j
$\bar{x}$	Light path
$f_j(\bar{x}, \lambda)$	Measurement contribution function for a pixel j
$L_o(x,\lambda), L_i(x,\lambda), L_e(x,\lambda)$	Exitant, incident and emitted radiance
$\omega, \omega_o, \omega_i$	Arbitrary, exitant and incident direction vectors
$f_s(\omega_o, \omega_i, \lambda)$	Bidirectional scattering distribution function
S	Unit (hemi)sphere of directions
θ	Solid angle
$t(x,\omega)$	Raycasting operation
$G(x \to x')$	Geometric coupling term
$V(x \to x')$	Visibility term
$\mu(D)$	Area product measure
$p(\bar{x}, \lambda)$	Path-wavelength probability density function
$\Delta_{ij}, \Delta_{\lambda\lambda'}$	Gradient, spectral gradient
$T_{i\to j}(\bar{x}), S_{\lambda\to\lambda'}(\bar{x})$	Shift mapping, spectral shift mapping
$r(I)_j$	Reconstruction filter applied to pixel j of image I
G_σ	Centered gaussian kernel with standard deviation σ
$p_j(\bar{x}, \lambda)$	Path-wavelength probability density function for a pixel j
$\tilde{I}_j$	Pre-estimate of I_j
$\tilde{N}$	Number of samples in pre-estimate
$I_j^{\downarrow}$	Downscaled pixel j in an image I
$r_{res}, r_{def}, r_{filt}$	Resampling, defensive and filtering reconstruction functions
α	Defensive sampling factor
K	Number of layers in pre-estimate

List of Abbreviations

RGB	Red-green-blue, trichromacy
CIE	International commission on illumination
SPD	Spectral power distribution
LTE	Light transport equation
BSDF	Bidirectional scattering distribution function
MIS	Multiple importance sampling
CDF	Cumulative distribution function
PDF	Probability density function
MSE	Mean squared error
SSIM	Structural similarity index
Path	Standard path tracing
Eis	Emitter importance sampling
Hero	Hero wavelength sampling
Sgpt	Spectral gradient sampling
Pre	Pre-estimated spectral rendering
px.	Pixel(s)
spp.	Samples per pixel

1 Introduction

Physically based rendering is an area of computer graphics that focuses on mathematically modeling the physical processes of light in order to create photorealistic renders and simulations. As these renders can be immensely costly to compute, models are often approximated and simplified where feasible. Currently, *Monte Carlo light transport* algorithms are the most advanced techniques available for producing photorealistic renders such as the one demonstrated in Figure 1.1, and have in recent years seen mainstream adoption in industry rendering systems such as Autodesk Arnold [2], Disney's Hyperion [29], and Pixar's Renderman [23]. Most modern rendering systems are *trichromatic*, as they represent all light and other types of spectra as a simple combination of three values, e.g. RGB. This is unfortunate, as trichromatic rendering has long been known to be incapable of accurately reproducing all the colors the human visual system perceives, partly due to effects such as metamerism [4]. In addition, trichromatic rendering makes it profoundly difficult to correctly simulate several physical phenomena such as chromatic light dispersion, diffraction and fluorescence.

To overcome these limitations, extended *spectral* light transport algorithms have been devised which instead incorporate full light spectra in their underlying computations. Spectral renderers can accurately reproduce colors, while also incorporating many advanced light phenomena. Unfortunately, in a classic example of the *no free lunch theorem*, spectral renderers tend to greatly increase the complexity of what is already an expensive computational process. In recent years, significant efforts have been made to reduce this added overhead, most notably through techniques stemming from research contributions such as *hero wavelength spectral sampling* [38] and *spectral gradient sampling* [20]. However, while the capabilities of spectral renderers have certainly advanced, their performance is still not on par with traditional renderers. This can be partially attributed to a single flaw, as their effectiveness is severely reduced when rendering scenes containing one or more realistic or non-uniform spectral distributions, which can be used to represent different levels of reflectance, transmittance or emission. The kinds of emission spectra attributed to modern LEDs and fluorescenct bulbs, for example, tend to be highly non-uniform and spiky, but are clearly required components for highly realistic visualizations, such as architectural renderings and light simulations.

Counteracting this reduction in performance is the main focus of this masters thesis, as we develop a

Figure 1.1: **Light Transport.** A demonstration of some of the widespread capabilities of modern light transport algorithms, produced with an unbiased path tracer in the open source Mitsuba Renderer [35].

simple and robust method to sufficiently negate the increased costs associated with using non-uniform spectra, thereby making their use in spectral rendering viable. Before we provide details on our method, we step back in the rest of this introductory chapter. First, we provide a brief, abstract overview of relevant topics in Section 1.1. In Section 1.2 we briefly describe a basic form of the problems and challenges we face. We then summarize our contributions in Section 1.3. Finally, in Section 1.4, we provide details on the structural organization of the remainder of this thesis.

1.1. Spectral Rendering

Although we will briefly introduce topics central to spectral rendering, we revisit and derive many of the involved equations and principles later on in Chapter 2.

Light transport algorithms, simply put, accurately estimate the color composition of light entering a certain sensor, when given a detailed description of the surrounding environment. A sensor can be a pixel on a camera, part of a piece of film, or even the human eye. We commonly describe the radiance I as entering one of a sensor's pixels j through the *light transport equation* (LTE) [10, 11, 30], which we give in simplified form as

$$I_j = \int_\Omega f_j(\bar{x})\, d\mu(\bar{x}). \tag{1.1}$$

Here the *path space* Ω is defined as the collection of all possible light paths $\bar{x} = x_0, x_1, \ldots, x_n$ of finite length n entering our sensor, and μ is a measure on this path space. A light path can be seen as a set of surface points along which light travels, starting from an emitter or light source at x_n, passing through a scene, and finally terminating at x_0 which is located directly on the sensor. Along a path, light is scattered by interactions with the environment and the different objects, materials and media contained within. The amount of light that reaches the sensor through a specific light path is then measured by the *measurement contribution function* f_j. Although this formulation of light transport, illustrated in Figure 1.2, is highly abstract, it is incredibly useful and allows us to generate realistic renders.

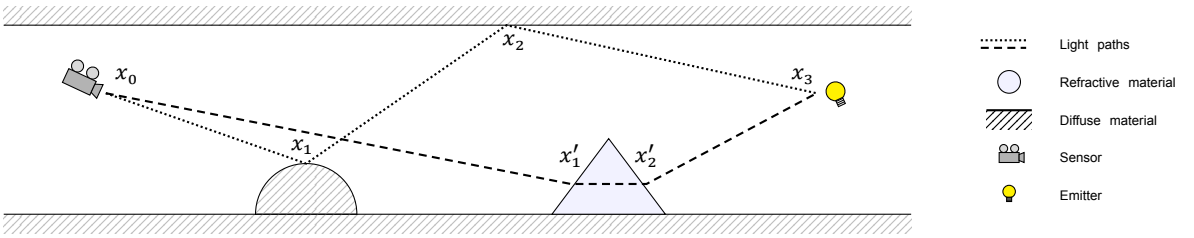


Figure 1.2: **Light Transport.** Light transport algorithms evaluate the contribution of different possible light paths passing through a scene, starting at a sensor and ending at an emitter. Along the way, paths interact differently with varying types of objects and materials. Two possible paths of finite length are shown as $\bar{x} = x_0, x_1, x_2, x_3$ and $\bar{x}' = x_0, x_1', x_2', x_3$.

Ω is a complicated space that likely contains infinitely many light paths, so we generally do not evaluate I_j analytically, nor can we simply measure all light paths. Instead, we apply an estimation technique known as *Monte Carlo integration*, estimating I_j by sampling N light paths randomly from some probability distribution, represented by its *probability density function* (PDF) p, and averaging the measured summation:

$$\hat{I}_j = \frac{1}{N} \sum_{i=1}^{N} \frac{f_j(\bar{x}_i)}{p(\bar{x}_i)}. \tag{1.2}$$

As $\hat{I}_j$ is an estimator, it suffers from a certain degree of error, but converges to a correct solution as $N \to \infty$. This error is present in renders with an insufficient number of samples, typically appearing as luminance noise which can, at times, severely impact image quality. Although this formulation of light transport describes how the composition of light can be measured, it does not account for its underlying representation. Most (trichromatic) representations are based on an important relation between the composition of light, and what we perceive as color. This relation stems from the human eye, which we briefly cover. For an in-depth overview, refer to the comprehensive textbook by Levine [16].

The human eye is sensitive to electromagnetic radiation in wavelengths ranging from approximately 380 nm to 740 nm, which is the range we consider the visible light spectrum. In order to distinguish

between different wavelengths, the eye uses a large collection of four main types (although there are many more) of photoreceptor cells: rod cells and three types of cone cells. The rod cells are primarily used for peripheral and scotopic (low-light condition) vision, and are highly sensitive to light centered at $\lambda \approx 500nm$. The three types of cone cells are used for photopic (well-lit condition) vision, and are each sensitive to different but overlapping areas of the visible light spectrum: the short or blue wavelengths *S* (centered at $\lambda \approx 420nm$), the medium or green wavelengths *M* ($\lambda \approx 530nm$), and the long or red wavelengths *L* ($\lambda \approx 560nm$). In 1931, the *International Commision on Illumination* (CIE) experimentally quantified the different sensitivity curves for these three cone cells, which led directly to the interesting property that we can describe any perceivable color with just three parameters, each corresponding to a certain amount of stimulation for one of the cone cells. This tristimulus representation of color was formalized in the CIE 1931 XYZ and RGB color spaces [28]. The CIE XYZ color space is particularly useful because it is *device-invariant*, and many different trichromatic color spaces have been derived from it since. Any spectral distribution $S(\lambda)$ can be transformed to the CIE XYZ color space by integrating it over three color matching functions $\bar{x}(\lambda)$, $\bar{y}(\lambda)$ and $\bar{z}(\lambda)$, illustrated in Figure 1.3, as

$$X = \int_{380}^{740} S(\lambda)\,\bar{x}(\lambda)\,d\lambda, \quad Y = \int_{380}^{740} S(\lambda)\,\bar{y}(\lambda)\,d\lambda, \quad Z = \int_{380}^{740} S(\lambda)\,\bar{z}(\lambda)\,d\lambda. \tag{1.3}$$

As we previously mentioned, there are unfortunate issues associated with rendering in the CIE XYZ color space, or any trichromatic color space such as RGB. Although a spectral distribution can be easily converted into a format usable in a trichromatic color space, inverting this process poses a difficult problem. As the response curves of human cone cells overlap, there are vastly different spectra that can appear as identical colors under certain lighting conditions. Such spectra are called *metamers*, and they imply that upsampling a trichromatic color value to a spectral distribution has infinitely many different results. If wavelength-specific information is required during rendering, it may be difficult to recover from a trichromatic representation.

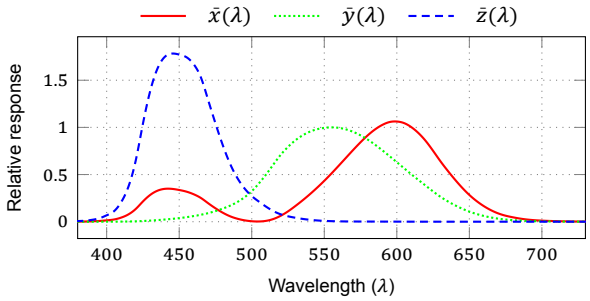


Figure 1.3: Relative response curves of the three CIE XYZ color matching functions [28].

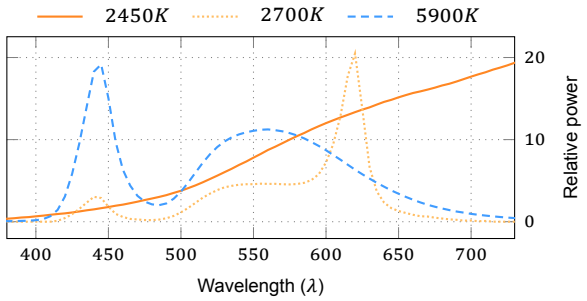


Figure 1.4: Relative emissive power of three measured SPDs of different color temperature LEDs [1], showcasing how irregular spectra can be.

To circumvent this, we can instead represent spectra explicitly during rendering, whether these are spectral power distributions (SPD) for emitters, reflectance spectra, or radiance measurements. This is not as straightforward as it may sound, given that spectra must contain data in a large $(380-740nm)$ range, and can appear in highly irregular forms, ranging from smooth curves to complicated, almost spiky functions. The SPDs illustrated in Figure 1.4 exemplify the different kinds of spectra we must consider. While representing such spectra directly by storing values for $1nm$ intervals gives correct results, the profound impact on both memory and rendering time makes such a method impractical at best. Instead, spectra are often discretized into fixed sets of bins spanning multiple nms. This approach introduces undersampling issues, i.e. aliasing, and may have trouble accurately representing peaks. A clear advantage over a simple trichromatic representation, however, is that wavelength-specific information is now available during rendering.

This is convenient, as we require this information to reproduce physical phenomena such as chromatic light dispersion. Light dispersion occurs when a beam of polychromatic light (exhibiting multiple wavelengths) passes through a refractive material such as glass, as it is then split into beams

[1] Measured SPD data, which is obtained from the *Light Spectral Power Distribution Database* [26], is licensed under a Creative Commons "Attribution-NonCommercial-NoDerivs 2.5" license.

of mostly monochromatic light (primarily exhibiting a single wavelength), each traversing the material at slightly different angles. The occuring refraction depends on a specified *index of refraction* (IOR), which describes how fast light propagates through a material, and which is defined as a function over the wavelength of light. A simple way to model light dispersion is through Sellmeier's Equation [27]:

$$n^2(\lambda) = 1 + \sum_{i=1}^{k} \frac{B_i \lambda^2}{\lambda^2 - C_i}, \tag{1.4}$$

where n is the resultant refractive index for a given wavelength λ, and $B_1, \dots, b_k$, $C_1, \dots, C_k$ are sets of empirically determined coefficients for specific materials. These coefficients are often recorded and published by manufacturers of technical glass, and as such are generally available for public use [24].

Unfortunately, although light dispersion is simple to integrate into refractive materials, it cannot efficiently be integrated into a light transport algorithm based on Equation 1.1, as this form of the LTE evaluates light paths without knowledge about their specific wavelengths. In order to accomodate wavelength-dependent phenomena, we add a second integral over a spectral domain $\Lambda \approx [380, 740]$, then evaluating paths that are restricted to a single wavelength by sampling these from a restricted path space Ω_λ. Formally, the LTE becomes

$$I_j = \int_\Lambda \int_{\Omega_\lambda} f_j(\bar{x}, \lambda)\, d\mu(\bar{x})\, d\lambda, \tag{1.5}$$

which allows us to evaluate spectral light transport problems. We illustrate one such problem in Figure 1.5. We form a Monte Carlo integrator in similar form to Equation 1.2, but explicitly sample wavelengths, as

$$\hat{I}_j = \frac{1}{N} \sum_{i=1}^{N} \frac{f_j(\bar{x}_i, \lambda_i)}{p(\bar{x}_i, \lambda_i)}, \tag{1.6}$$

in which $f_j(\bar{x}, \lambda)$ denotes the measurement contribution function for a path-wavelength pair $(\bar{x}, \lambda)$, and p describes the probability of sampling such a path-wavelength pair. As our estimator now additional samples the spectral domain, significant additional error is introduced, which typically becomes visible in resultant renders as chromatic (colored) noise.

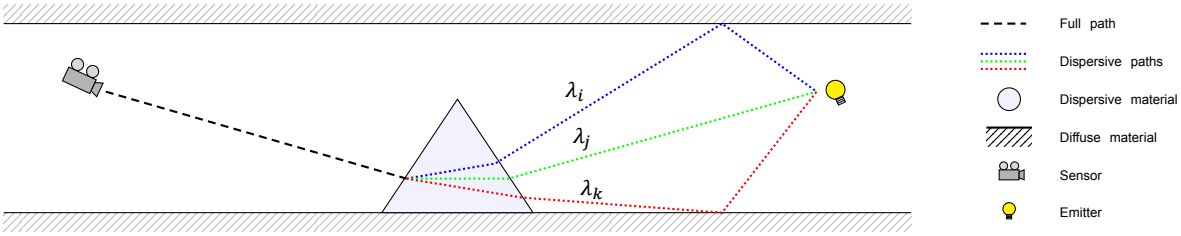


Figure 1.5: **Spectral Light Transport.** Spectral light transport algorithms allow us to evaluate the contribution of light paths that initially differ in only their sampled wavelengths λ, but eventually traverse the environment in different manners due to this wavelength.

As we mentioned, major developments have in recent years managed to improve the convergence rate of spectral renderers significantly. As the increase in error in spectral rendering stems from a number of sources, these techniques tend to focus on solving slightly different problems. Evans and McCool [8] first noted that the estimator of Equation 1.6 is inherently inefficient, as it propagates a light path for but a single wavelength, and can be easily modified to evaluate a cluster of wavelengths as long as no wavelength dependency is introduced into a path. They additionally briefly covered the impact of rendering with non-uniform emittance spectra, which is a topic we revisit in Section 1.2. An extension of their technique was developed by Radzisewski et al. [25], who developed a method for propagating multiple wavelengths through wavelength-dependent non-specular BSDFs such as rough glass or human skin. This was later refined by Wilkie et al. [38] with hero wavelength spectral sampling, generalizing the technique through a clever application of *multiple importance sampling* (MIS) [32]. More recently, Petitjean et al. [20] demonstrated gains in rendering performance through a spectral extension of gradient domain rendering techniques [12, 15], by simultaneously estimating a so-called

spectral gradient next to normal image values, and afterwards combining these in a *screened poisson reconstruction* [3]. We further explore these techniques, as well as basic spectral light transport, in Chapter 2.

1.2. Wavelength Sampling Problems

Implementations of Equation 1.6 require a suitable distribution for sampling wavelengths. A uniform distribution seems a simple and logical choice, as we do not know the distribution of power over incoming radiance beforehand. Indeed, notable spectral renderers, such as LuxCoreRender [1], Mental Ray [18], and Mitsuba [35], all sample wavelengths uniformly. Unfortunately, we can easily create scenarios where uniform sampling of wavelengths is at best sub-optimal. Unless the distribution of power over incoming radiance is uniform (i.e., varying shades of gray or white), some wavelengths contribute more energy than others, and should not be evaluated with equal probability.

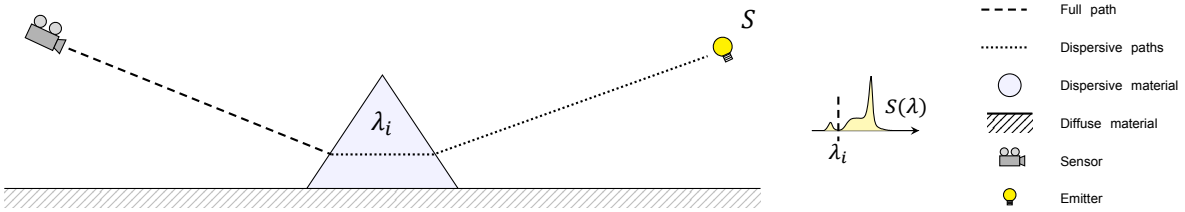


Figure 1.6: **Simple wavelength sampling problem.** A path is evaluated for a sampled wavelength λ_i, but will hardly contribute energy as it falls off major contributing parts of the encountered emitter's SPD $S(\lambda)$.

This was briefly touched on by Evans and McCool [8], who noted that if we render a scene with non-uniform SPDs such as those in Figure 1.4, wavelengths centered on an emitter's peaks tend to contribute greatly to an image, while others hardly contribute at all. In this case, uniform sampling can lead to a scenario such as Figure 1.6, where we evaluate a wavelength that, at the end of a path, turns out to contribute little energy. Although Evans and McCool showed that using a distribution similar to the emitter's SPD for wavelength sampling is highly beneficial, they could only do so for a single emitter. In fact, as we will show in Chapter 3, this technique becomes detrimental in scenarios where multiple emitters are present. We can further complicate the issue by introducing other non-uniform distributions, such as reflectance and transmittance spectra, into a scene. To our knowledge, no technique currently exists that generates a wavelength sampling distribution that can efficiently handle this issue, and performs sufficiently better than uniform sampling.

1.3. Overview of Contributions

Following on the issue we noted in Section 1.2, we focus our contributions in this thesis entirely on designing a robust and simple method to alleviate this wavelength sampling problem. The main part of these contributions consists of a method we call *pre-estimated spectral rendering*, which enables us to generate a suitable wavelength sampling distribution, and which dynamically accomodates the presence of multiple non-uniform spectral distributions throughout a scene, with little overhead compared to a simple path tracer. Our contribution allows us to improve the efficiency of spectral rendering in a number of difficult scenarios, and subsequently enables the use of complicated, realistic spectra in rendering. We provide more details on the different components of this method in Chapter 4.

1.4. Thesis Structure

We first cover notable background theory and relevant related work in Chapter 2. In Chapter 3, we explore the *wavelength sampling problem* we introduced in Section 1.2 with a number of examples. Then, in Chapter 4, we develop the main components of our method, which we afterwards thoroughly benchmark for a number of difficult scenarios in Chapter 5. We analyze obtained the results in Chapter 6, before concluding the thesis in Chapter 7. Finally, we detail an additional and related technique which we only partially developed in Appendix A.

2 Background

in Chapter 1, we provided a rather abstract introduction to spectral rendering and Monte Carlo light transport. In order to properly analyze our problem in Chapter 3, we must first obtain a formal understanding of the relevant material. As such, we revisit some previously discussed topics, and provide formal derivations for most of them. We first review basic probability concepts in Section 2.1, and afterwards use these to derive a spectral version of the LTE in Section 2.2. Here, we also extensively describe related techniques such as hero wavelength sampling [38] and spectral gradient sampling [20].

Throughout this chapter and the rest of this thesis, we loosely follow mathematical notation used by Pharr et al. in their book on physically based rendering [22], which they themselves base heavily on the groundwork done by Veach in his dissertation [30].

2.1. Probability Theory

We briefly review basic concepts stemming from probability theory, as we will use these extensively for topics such as Monte Carlo integration and importance sampling. For a more thorough introduction to probability theory, refer to the previously mentioned writings of Pharr et al. [22], who provide an extensive overview within the context of computer graphics. Note that, for consistency throughout the thesis, we denote random variables as upper case letters, i.e., X, Y, and their *sampled* realizations as corresponding lower case letters, i.e., x, y.

Recall that a random variable X is a value obtained or generated through some random process or event, and is drawn from a domain that is either discrete or continuous. In this context, we are primarily concerned with the continuous case, which can be drawn from the real numbers, among others. Applying a function f to a random variable X logically results in a new random variable $Y = f(X)$. There are two important functions that we use to describe properties of random variables. The first of these is the *cumulative distribution function* (CDF) $P_X(X)$, which describes the probability that X takes a value less to or equal to some x, i.e.

$$P_X(x) = Pr(X \leq x). \tag{2.1}$$

The second of these is the *probability density function* (PDF) $p_X(x)$, which can be interpreted as describing the relative probability of X realizing a specific value x, and is formally defined as

$$p_X(x) = \frac{d}{dx} P_X(x). \tag{2.2}$$

All values in a PDF are essentially nonnegative, and integrating a PDF over its domain always results in 1. For a domain bounded to $[0, 1]$, this is essentially the same as asking $Pr(X \leq 1.0)$.

2.1.1. Expected Value and Variance

Given some function f defined over a domain Ω, we often wish to attain a notion of its *expected value*, which is defined as its average over a collection of values distributed by some p, or formally

$$E_p[f(x)] = \int_\Omega f(x)\ p(x)\ d\mu(x). \tag{2.3}$$

The *variance* of a function then gives us a notion of how far individual samples of f are spread around its expected value. It is defined as the expected squared deviation of a function from its expected value, or

$$V[f(x)] = E[(f(x) - E[f(x)])^2] \tag{2.4}$$

Given this definition, we display a number of properties of both the expected value and variance:

$$E[af(x)] = a\ E[f(x)], \tag{2.5}$$

$$E\left[\sum_i f(X_i)\right] = \sum_i E[f(X_i)], \tag{2.6}$$

$$V[af(x)] = a^2\ V[f(x)], \tag{2.7}$$

and, given that all used random variables X_i are independent:

$$V\left[\sum_i f(X_i)\right] = \sum_i V[f(X_i)]. \tag{2.8}$$

We use these properties to expand expression of the variance into a more concise form as

$$\begin{aligned} V[f(x)] &= E[(f(x) - E[f(x)])^2] \\ &= E[f^2(x) - 2f(x)\ E[f(x)] + E[f(x)]^2] \\ &= E[f^2(x)] - 2E[f(x)]E[f(x)] + E[f(x)]^2 \\ &= E[f^2(x)] - E[f(x)]^2, \end{aligned} \tag{2.9}$$

which implies variance may be defined as the expected value of the square, minus the square of the expected value. Finally, we also define the *standard deviation* $\sigma[f(x)]$ as the square root of the variance, i.e.

$$\sigma[f(x)] = \sqrt{V[f(x)]}. \tag{2.10}$$

2.1.2. Monte Carlo Integration

We now use these concepts to provide a definition for Monte Carlo integration. This is a technique that allows us to, given a function f defined over a domain Ω, to evaluate an integral of the form

$$I = \int_\Omega f(x)\ d\mu(x), \tag{2.11}$$

which occurs surprisingly often in computer graphics. Intuitively, Monte Carlo integration functions by obtaining a set of N realized random samples $x_i \in \Omega$ from some probability distribution p, and then averaging the results of evaluating f for each sample. We define a Monte Carlo estimator for I as

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}. \tag{2.12}$$

This estimator converges to a correct result as more samples are used, i.e.

$$\lim_{N\to\infty} E[\hat{I}] = I, \tag{2.13}$$

which follows from

$$\begin{aligned} E[\hat{I}] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}E\left[\frac{f(x_i)}{p(x_i)}\right] \\ &= E\left[\frac{f(x)}{p(x)}\right] \\ &= \int_{\Omega}\frac{f(x)}{p(x)}\,p(x)\,d\mu(x) \\ &= \int_{\Omega}f(x)\,d\mu(x). \end{aligned} \tag{2.14}$$

Monte Carlo integration carries several advantages over other integration techniques. It is particularly useful for evaluating multidimensional integrals, as we have until now made no assumptions about the dimensionality or shape of the covered domain Ω. This is especially relevant for light transport problems, which tend to be (extremely) multidimensional. In addition, Monte Carlo integration is a surprisingly simple technique as it only requires us to satisfy two principles: we must be able to draw random samples from some distribution p that is (preferably) similar in shape to Ω, and we must be able to subsequently query f using these samples.

Finally, we can show that Monte Carlo integration has a predictable rate of convergence bound by $\mathcal{O}(N^{-1/2})$ in the number of samples. Assuming all random samples x_i we take are independent, we apply Equation 2.9 to derive the variance of our estimator as

$$\begin{aligned} V[\hat{I}] &= V\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right] \\ &= \frac{1}{N^2}\sum_{i=1}^{N}V\left[\frac{f(x_i)}{p(x_i)}\right] \\ &= \frac{1}{N}\,V\left[\frac{f(x)}{p(x)}\right], \end{aligned} \tag{2.15}$$

Assuming that $V\left[\frac{f(x)}{p(x)}\right]$ is finite, variance decreases linearly as N increases. The standard deviation is then derived as

$$\begin{aligned} \sigma[\hat{I}] &= \sqrt{V[\hat{I}]} \\ &= \sqrt{\frac{1}{N}\,V\left[\frac{f(x)}{p(x)}\right]} \\ &= \frac{1}{\sqrt{N}}\,\sigma\left[\frac{f(x)}{p(x)}\right], \end{aligned} \tag{2.16}$$

which shows the convergence rate of $\hat{I}$ as bounded in $\mathcal{O}(N^{-1/2})$, in terms of direct deviation from the expected value I. In practical terms, this implies that if we wish to halve the relative error of a Monte Carlo estimator, we must roughly quadruple the number of samples used.

One final property of a Monte Carlo estimator we must cover, is its bias, which is the difference between its expected value and the actual value it is defined for as an estimator, or

$$\beta[\hat{I}] = E[\hat{I}] - I. \tag{2.17}$$

Not all estimators necessarily converge to a correct result. An estimator is said to be unbiased when $\beta(\hat{I}) \equiv 0$. Both bias and lack thereof may be desirable properties depending on the circumstances, as biased estimators sometimes converge considerably faster than their unbiased counterparts.

2.1.3. Mean Squared Error

A useful metric we apply in Chapter 5 to compare both the convergence rates and eventual bias of different Monte Carlo estimators, is the *mean squared error* (MSE)

$$MSE[\hat{I}] = E[(\hat{I} - I)^2]. \tag{2.18}$$

We can expand this definition as

$$\begin{aligned} MSE[\hat{I}] &= E[(\hat{I} - I)^2] \\ &= E[(\hat{I} - E[\hat{I}] + E[\hat{I}] - I)^2] \\ &= E[(\hat{I} - E[\hat{I}])^2 + 2\,(\hat{I} - E[\hat{I}])\,(E[\hat{I}] - I) + (E[\hat{I}] - I)^2] \\ &= E[(\hat{I} - E[\hat{I}])^2] + 2\,E[(\hat{I} - E[\hat{I}])]\,E[(E[\hat{I}] - I)] + E[(E[\hat{I}] - I)^2] \\ &= E[(\hat{I} - E[\hat{I}])^2] + 2\,(E[\hat{I}] - E[\hat{I}])\,E[(E[\hat{I}] - I)] + E[(E[\hat{I}] - I)^2] \\ &= E[(\hat{I} - E[\hat{I}])^2] + E[(E[\hat{I}] - I)^2] \\ &= V[\hat{I}] + \beta[\hat{I}]^2, \end{aligned} \tag{2.19}$$

which, informally, implies the MSE is the sum of an estimator's variance and its squared bias.

2.1.4. Random Variable Sampling

We mentioned in Subsection 2.1.2 that, in order to perform Monte Carlo integration, we must be able to draw random samples from a specific probability distribution, preferably in an efficient manner. We will cover two generic methods for this.

Inversion Method The first of these is the commonly known *inverse transform* or *inversion method*, which leverages a uniform distribution $Unif[0, 1]$, and maps samples drawn from this distribution to the actual distribution we wish to sample. In the event we have some probability distribution represented by its PDF $p(x)$, we can create a random variable X to sample from p in four steps:

1. Obtain a random variable $U \in Unif[0, 1]$.
2. Compute the CDF of $p(x)$: $P(x) = \int_0^x p(x')\,dx'x$.
3. Compute the inverse of the CDF: $P^{-1}(x)$.
4. Compute $X = P^{-1}(U)$.

Any realization of the random variable X is essentially sampled from p. This technique generally has low associated cost for drawing samples. Unfortunately, it explicitly requires the ability for us to both integrate over $p(x)$, and to subsequently invert $P(x)$. Depending on underlying implementations or representations of the distributions used, this may not always be possible.

Rejection Method The second method we cover, which is preferable in this difficult scenario, is the *acceptance-rejection* or *rejection method*, that can be used to sample a PDF $p(x)$ which cannot be easily integrated over, or for which the CDF cannot be inverted analytically. The idea is to instead take a conveniently sampleable distribution q that satisfies

$$p(x) \leq c \cdot q(x) \tag{2.20}$$

for some constant c. We then draw samples from p in the following steps:

1. Obtain a random variable $X \in q$.
2. Obtain a random variable $U \in Unif[0, 1]$.
3. If $U \leq p(X)\ /\ c \cdot q(X)$, return X.
4. Else, repeat from step 1.

Although the rejection method can be used for sampling difficult distributions, it is clearly an inefficient method in more general cases where the inversion method is applicable.

2.1.5. Variance Reduction Techniques

We discussed the convergence rate of the Monte Carlo estimator in Subsection 2.1.2. A number of different techniques exist that modify the basic estimator of Equation 2.12 to improve its convergence rate in certain scenarios. We will describe variations of one of these techniques - *importance sampling* - as it is applicable in many different situations.

Importance Sampling A straightforward technique to improve the Monte Carlo estimator is importance sampling. Simply put, as the estimator realizes samples from some distribution p, it will converge more quickly if said distribution is similar in shape to the integral we are evaluating. To provide a basic example, we show what happens if we choose the best possible distribution for our estimator, which would be of the form $p(x) = c\, f(x)$, with

$$c = \frac{1}{\int_\Omega f(x')\, d\mu(x')}, \tag{2.21}$$

which leads to an estimator with zero variance, as

$$\begin{aligned} \hat{I} &= \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{c f(x_i)} \\ &= \frac{1}{c} \\ &= \int_\Omega f(x')\, d\mu(x'), \end{aligned} \tag{2.22}$$

which is the value of our integral. This is purely theoretical, as there is virtually no point to estimating an integral whose value we know beforehand. However, it serves to show that, if we can find a distribution for p such that it is highly proportional to the integrand, i.e.

$$p(x) \propto I, \tag{2.23}$$

variance will generally be reduced. Unfortunately, there are caveats to importance sampling. A considerable amount of care should be taken when selecting a distribution, as variance can increase when a distribution does not remotely fit the integral. For example, consider what happens if we compute the estimate of

$$I = \int f(x)\, dx \quad : \quad f(x) = \begin{cases} 1.5 & x \in [0, 0.5) \\ 0.5 & x \in [0.5, 1.0) \end{cases} \tag{2.24}$$

with the following estimator

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x)}{p(x)} \quad : \quad p(x) = \begin{cases} 0.1 & x \in [0, 0.5) \\ 0.9 & x \in [0.5, 1.0) \end{cases} \tag{2.25}$$

which would give terrible results. The majority of samples falls into the $[0.5, 1.0)$ range, which rapidly leads to $f(x)/p(x) \approx 0.55$, even though $I = 1.0$. This is bound to converge, but will do so at slower rate than when we had used a uniform distribution for sampling. To make matters worse, consider what happens if we use a distribution that does not cover the domain Ω entirely. This will lead to an estimator which never queries certain parts of the integral, and which is therefore biased.

Defensive Importance Sampling Given that we may not be able to measure the quality of a sampling distribution, we may inadvertedly increase variance or introduce bias. *Defensive importance sampling* [19] leverages a mixture with a safe (likely uniform) distribution to reduce potential impacts. In short, when we choose a distribution p for importance sampling, we mix it with a secondary distribution q, of which we know that it at least covers the domain Ω, as

$$r(x) = a_1 p(x) + a_2 q(x) \quad : \quad a_1 + a_2 = 1.0 \tag{2.26}$$

and consequently use the resulting distribution r for sampling instead. Depending on the choices of constants a_1, a_2, the effectiveness of a *good* distribution may be reduced, and the ineffectiveness of a *bad* distribution may be improved.

Multiple Importance Sampling Although basic importance sampling can greatly reduce variance when we are estimating singular integrals, it is of less use when working with difficult (and especially multi-dimensional) integrals. Consider, for example, the following integral

$$I = \int_{\Omega} f(x)\ g(x)\ dx. \tag{2.27}$$

We would preferably use a convenient distribution that allows us to importance sample both f and g, but this may be difficult to find. If we instead have two separate distributions p_f and p_g that are suitable only for sampling their respective integrands, applying either of these likely increases variance by incorrectly covering the other integrand. A clever technique to (mostly) circumvent this issue was developed by Veach for his dissertation as *Multiple Importance Sampling* (MIS) [32], where, intuitively, we draw a number of samples from both distributions, and then weight them accordingly in such a way that most increases in variance from incorrect sampling are cancelled out. We can apply this idea to form an estimator for Equation 2.27 as

$$\hat{I} = \frac{1}{N_f} \sum_{i=1}^{N_f} \frac{f(x_i)\ g(x_i)\ w_f(x_i)}{p_f(x_i)} + \frac{1}{N_g} \sum_{j=1}^{N_g} \frac{f(y_j)\ g(y_j)\ w_g(y_j)}{p_g(y_j)}, \tag{2.28}$$

where N_f samples are drawn from p_f and N_g samples are drawn from p_g. Here we combine both integrands, but weight results by two weighting factors w_f and w_g, which we will explain below. We first show how MIS is usually applied in a more general case, where we evaluate an integral over just one integrand, but have n different importance sampling distributions $p_0, \ldots, p_n$ that together are appropriately similar in shape to the integral, which can happen when the domain Ω is sufficiently complicated. An estimator that samples each of these distributions is formed as

$$\hat{I} = \sum_{i=1}^{n} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})}. \tag{2.29}$$

Veach derives a number of different weighting functions that are suitable for most circumstances. The most commonly used of these is the so-called *balance heuristic*

$$w_i(x) = \frac{n_i\ p_i(x)}{\sum_k n_k\ p_k(x)}, \tag{2.30}$$

which is similar to the more generalized *power heuristic*

$$w_i(x) = \frac{(n_i\ p_i(x))^{\beta}}{\sum_k (n_k\ p_k(x))^{\beta}}. \tag{2.31}$$

which, as Veach shows, can in practice be more useful for low-variance problems. It should be noted that there are many different variations on these weighting functions within existing literature, but we will not cover these as this essentially poses a topic in itself.

2.2. Spectral Light Transport

In Chapter 1, we briefly touched on an integral formulation of the *light transport equation* (LTE) that is suitable for spectral rendering, as light paths are evaluated (measured) for a specific wavelength:

$$I_j = \int_{\Lambda} \int_{\Omega_\lambda} f_j(\bar{x}, \lambda)\ d\mu(\bar{x})\ d\lambda. \tag{1.5}$$

Given that this equation is central to the topic of our thesis, we summarily show how it was derived from its non-spectral counterpart of Equation 1.1, essentially providing an overview of its advantages over other formulations. Afterwards, we will cover the related work that builds further on this formulation.

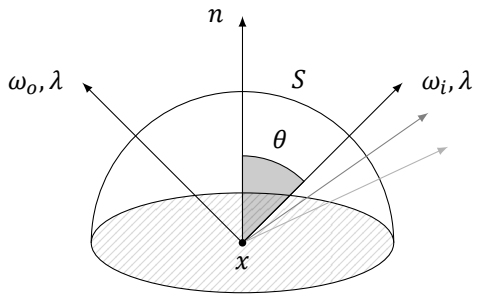


Figure 2.1: **Directional LTE.** The directional formulation of the LTE accounts for scattered light from possible incident directions ω_i, represented here by a hemisphere S. Scattered light is weighted by a cosine factor over θ, which is the angle between ω_i and a surface normal n at surface point x.

2.2.1. The Spectral Light Transport Equation

Instead of a path-integral formulation, physically based rendering algorithms have traditionally been occupied with evaluating a *directional formulation* of light transport, which was simultaneously developed by both Immel et al. [10] and Kajiya [11]. This formulation describes an equilibrium between exitant radiance L_o leaving a certain surface point x in direction ω_o, the emitted radiance L_e, and the scattering of incident radiance L_i, as

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_S f_s(x, \omega_o, \omega_i)\ L_i(x, \omega_i)\ |\cos\theta_i|\ d\omega_i. \tag{2.32}$$

The integration domain S describes the set of possible incident directions ω_i, which we generally represent as either a unit sphere or hemisphere of directions. The *bidirectional scattering distribution function* (BSDF) f_s defines the amount of light that is scattered instead of absorbed, given incident and exitant directions on a surface point. The added cosine factor accounts for a weakening of incident light, given that the relative surface area on which light is projected depends on the incident angle θ.

In his dissertation [30], Veach provides a rather extensive derivation for the familiar path-integral formulation of the LTE, which instead consists of a single integral over a path-space Ω, and which carries several immediate advantages over the directional formulation, foremost of which is that we can apply general-purpose integration techniques (such as Monte Carlo integration) to solve it. It has led directly to the development of improved rendering techniques such as bidirectional path tracing [14, 31], metropolis light transport [33] and gradient-domain path tracing [15]. In order to arrive at a spectral variant of the path integral formulation, we will make several modifications to the surface point LTE in a similar vein to Veach in his dissertation. We wil follow his derivations, albeit in a *highly* abbreviated manner. To start, we modify the LTE to incorporate most wavelength-dependent phenomena, by adding a wavelength parameter λ as

$$L_o(x, \omega_o, \lambda) = L_e(x, \omega_o, \lambda) + \int_S f(x, \omega_o, \omega_i, \lambda)\ L_i(x, \omega_i, \lambda)\ |\cos\theta_i|\ d\omega_i, \tag{2.33}$$

which we illustrate in a basic form in Figure 2.1. We now reformulate this notation further, as we can directly relate incident and exitant radiances L_i and L_o. If we assume that there are no participating media in a scene (i.e., we operate in a perfect vacuum, which makes things easier), then any radiance traveling along a ray must remain constant. In other words:

$$L_i(x, \omega, \lambda) = L_o(t(x, \omega), -\omega, \lambda), \tag{2.34}$$

where we substitute a surface point with the application of a simple raycasting operation t given by

$$t(x, \omega) = x', \tag{2.35}$$

where x' is the first other surface point encountered by a ray traveling along direction ω from x. When performing this substitution, we drop incident and exitant subscripts, and reformulate the LTE as:

$$L(x, \omega_o, \lambda) = L_e(x, \omega_o, \lambda) + \int_S f(x, \omega_o, \omega_i, \lambda)\ L(t(x, \omega_i), -\omega_i, \lambda)\ |\cos\theta|\ d\omega_i. \tag{2.36}$$

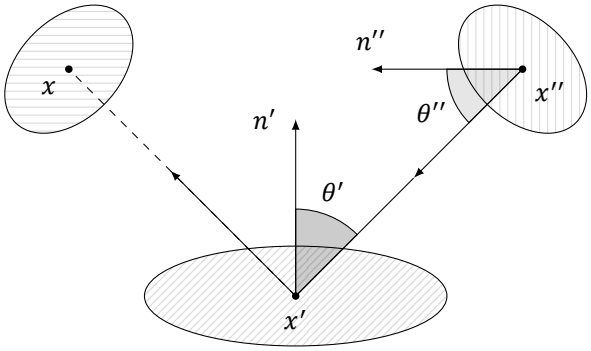


Figure 2.2: **Surface Area LTE.** The surface area formulation of the LTE accounts for scattering of light between several surface points $x, x' and x''$, instead of directions on a sphere or hemisphere.

In this formulation, light transport is a recursive problem in addition to an integration problem, which unfortunately makes it rather difficult to evaluate. The next step required is the removal of directional variables ω_i and ω_o as we slowly shift towards a formulation that leverages sets of surface points (i.e. paths). Instead of relying on a raycasting operation, we explicitly define the relation between two such surface points x and x' as $x' \rightarrow x$. We reformulate exitant radiance as

$$L(x' \rightarrow x, \lambda) = L(x', \omega, \lambda) \quad : \quad \omega = \widehat{x - x'} \tag{2.37}$$

and similarly rewrite the BSDF f_s which describes a relationship between three different points, as

$$f_s(x'' \rightarrow x' \rightarrow x, \lambda) = f(x', \omega_o, \omega_i, \lambda) \quad : \quad \omega_o = \widehat{x - x'}, \omega_i = \widehat{x'' - x'}. \tag{2.38}$$

Building on this, we reformulate the LTE into what is commonly known as a *three-point* or *surface area* form, which we additionally visualize in Figure 2.2:

$$L(x' \rightarrow x, \lambda) = L_e(x' \rightarrow x, \lambda) + \int_A f_s(x'' \rightarrow x' \rightarrow x, \lambda)\, L(x'' \rightarrow x', \lambda)\, G(x'' \rightarrow x')\, dA(x''). \tag{2.39}$$

An important distinction between these different formulations, is the change to an integration domain A, which represents the space of all surfaces (or points thereon). In order to transform the LTE from one over a solid angle θ to one over surface areas, Veach describes the geometric coupling term G as

$$G(x \rightarrow x') = V(x \rightarrow x')\, \frac{|\cos\theta|\, |\cos\theta'|}{\|x - x'\|^2} \tag{2.40}$$

which accounts for a change-of-variables between integration domains, and, as we no longer rely on a raycasting operation, adds a visibility term V which is defined as

$$V(x \rightarrow x') = \begin{cases} 1 & \text{if } x \text{ and } x' \text{ are mutually visible,} \\ 0 & \text{else.} \end{cases} \tag{2.41}$$

The next step is for us to derive the familiar path integral formulation, for which we must expand the recursive components. If we repeatedly substitute the right-hand $L(x'' \rightarrow x')$ of Equation 2.39 into the main integral, we form a continuously growing equation over an increasing number of surface points. The first two expansions of this equation, which are shown in this manner by Pharr et al. [22] in their own derivation of the path-integral formulation, are:

$$\begin{aligned} L(x_1 \rightarrow x_0, \lambda) \;=\; & L_e(x_1 \rightarrow x_0, \lambda) \\ & + \int_A L_e(x_2 \rightarrow x_1, \lambda)\, f_s(x_2 \rightarrow x_1 \rightarrow x_0, \lambda)\, G(x_2 \rightarrow x_1)\, dA(x_2) \\ & + \int_A \int_A L_e(x_3 \rightarrow x_2, \lambda)\, f_s(x_3 \rightarrow x_2 \rightarrow x_1, \lambda)\, G(x_3 \rightarrow x_2) \\ & \quad \times f_s(x_2 \rightarrow x_1 \rightarrow x_0, \lambda)\, G(x_2 \rightarrow x_1)\, dA(x_3)\, dA(x_2) \\ & + \ldots \end{aligned} \tag{2.42}$$

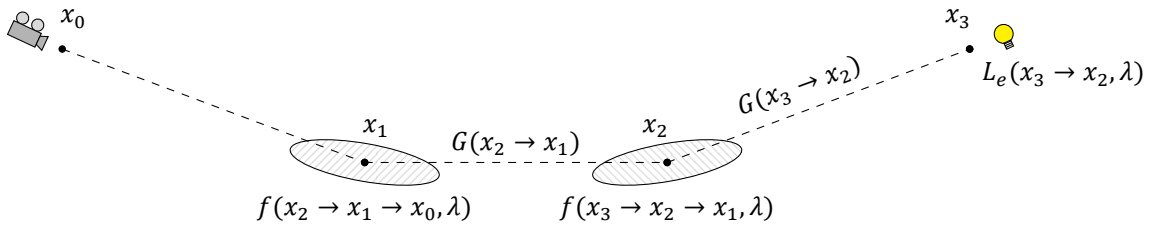


Figure 2.3: **Light path.** The formulation of light transport along a single consecutive set of surface points, starting at a sensor and ending at an emitter. The different components that form the final contribution of this *path* are marked in the image.

We now change our notation slightly, and instead of x', x'', start numbering the numerous surface points - vertices - in order of appearance from the first surface point queried at $L(x_1 \to x_0, \lambda)$. As Pharr et al. formulate, each of the terms at the right side of this equation can be described as a *path* of increasing length, towards some surface that contributes emitted radiance L_e. We visualize this expansion for a single path in Figure 2.3.

It becomes clear at this point that, although the first vertices of the path (x_0 and x_1) are predetermined by the camera's location and first ray intersection, the rest of the path varies over all surface points throughout the scene. As there are likely infinitely many surface points in a scene, there are likewise infinitely many paths similar to the one in Figure 2.3. We generalize this principle by defining the set of all such paths of a specific length n and carrying a wavelength λ, as $\Omega_{n,\lambda}$, where each path is of the form $\bar{x} = x_0, x_1, \ldots, x_n$. In order to integrate over $\Omega_{n,\lambda}$, Veach describes three required components. The first of these is the so-called *path space*, which represents the union of all these sets of paths of finite lengths as

$$\Omega_\lambda = \bigcup_{i=1}^{\infty} \Omega_{\lambda,i}, \tag{2.43}$$

and will form the integration domain from here on. Veach then defines the *area-product measure* - which he shows is required to continue accounting for the change of variable and visibility terms - as

$$\mu_n(D) = \int_D dA(x_0) \cdots dA(x_n) \quad : \quad D \subset \Omega_{n,\lambda}, \tag{2.44}$$

which can be extended to all paths as

$$\mu(D) = \sum_{i=1}^{\infty} \mu_i(D \cap \Omega_{i,\lambda}). \tag{2.45}$$

The final component we require for a path-integral formulation is the integrand, which will become the *measurement contribution function* f_j we briefly mentioned in Chapter 1. The integrand is defined for each specific path $\bar{x}$ separately, in a non-recursive manner. Given a single example path $\bar{x}$, we can expand part of f_j as

$$\begin{aligned} f_j(\bar{x}, \lambda) = \; & L_e(x_1 \to x_0, \lambda)\, f_s(x_2 \to x_1 \to x_0, \lambda)\, G(x_2 \to x_1) \\ & \cdot\, L_e(x_2 \to x_1, \lambda)\, f_s(x_3 \to x_2 \to x_1, \lambda)\, G(x_3 \to x_2) \\ & \cdot\, \ldots \end{aligned} \tag{2.46}$$

As we now have an integratable domain Ω_λ, an integrand f_j, and all other required components, this results in a path-integral formulation of light transport for a single wavelength:

$$I_{j,\lambda} = \int_{\Omega_\lambda} f_j(\bar{x}, \lambda)\, d\mu(\bar{x}). \tag{2.47}$$

As the spectral domain Λ of relevant wavelengths is comparatively simple (being one-dimensional) we can simply apply a second integral over this domain to accumulate measured light over all wavelengths, thereby forming the final formulation of the spectral LTE:

$$I_j = \int_\Lambda \int_{\Omega_\lambda} f_j(\bar{x}, \lambda)\, d\mu(\bar{x})\, d\lambda. \tag{1.5}$$

2.2.2. Monte Carlo Estimation

As we now have a non-recursive integral formulation for light transport, we apply the Monte Carlo integration technique we covered in Subsection 2.1.2 to form an estimator as

$$\hat{I}_j = \frac{1}{N}\sum_{i=1}^{N}\frac{f_j(\bar{x}_i,\lambda_i)}{p(\bar{x}_i,\lambda_i)}, \tag{1.6}$$

where $p(\bar{x},\lambda)$ is a representative PDF for a distribution for the sampling of pairs of path and wavelength samples. This is, once again, an unbiased estimator for I_j, as

$$\begin{aligned} E[\hat{I}_j] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f_j(\bar{x}_i,\lambda_i)}{p(\bar{x}_i,\lambda_i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}E\left[\frac{f_j(\bar{x}_i,\lambda_i)}{p(\bar{x}_i,\lambda_i)}\right] \\ &= \int_{\Lambda}\int_{\Omega_\lambda}\frac{f_j(\bar{x},\lambda)}{p(\bar{x},\lambda)}\,p(\bar{x},\lambda)\,d\mu(\bar{x})\,d\lambda \\ &= \int_{\Lambda}\int_{\Omega_\lambda} f_j(\bar{x},\lambda)\,d\mu(\bar{x})\,d\lambda. \end{aligned} \tag{2.48}$$

The question that remains is how to sample path-wavelength pairs $(\bar{x},\lambda)$. Sampling light paths in an efficient manner is a topic in and of itself, which we do not cover further. Many different techniques and strategies exist, some of which Veach discusses in his dissertation [30].

2.2.3. Hero Wavelength Sampling

Careful readers might note the inherent inefficiency of Equation 1.6, given that this always samples a single path-wavelength pair. If no wavelength-dependency occurs along a path, different wavelengths can be propagated in a similar manner. This was first noted by Evans and McCool [8], who propose propagating a cluster of wavelengths along a single path until wavelength-dependency does occur. At this point, they suggest either discarding all but one wavelength, or splitting into multiple paths. They necessarily keep path and wavelength sampling separate, i.e.

$$p(\bar{x},\lambda) = p(\lambda)\cdot p(\bar{x}). \tag{2.49}$$

Building on this idea, both Radziszewski et al. [25] and more recently Wilkie et al. [38] note that far from all (sub)surface scattering at wavelength-dependent phenomena is perfectly specular. Example scenarios include most forms of glass and other refractive media, which have roughened and scratched surfaces, and even human skin, which curiously exhibits non-specular wavelength dependency. Both groups of authors demonstrate methods to propagate multiple wavelengths along a single light path in these scenarios. We cover the aptly named *hero wavelength sampling* developed by Wilkie et al. [38]. They propose to randomly select an initial wavelength - the *hero wavelength* λ_h - for path propagation. A stratified set of other wavelengths is then similarly measured through the same path, using a clever combination of MIS [32], which results in the following estimator:

$$\hat{I}_j = \frac{1}{N}\frac{1}{C}\sum_{i=1}^{N}\sum_{j=1}^{C} w_{\lambda_h}(\bar{x}_i,\lambda_j)\frac{f_j(\bar{x}_i,\lambda_j)}{p(\bar{x}_i,\lambda_j)}, \tag{2.50}$$

where C denotes the number of wavelength samples used, and the weight $w_{\lambda_h}(\bar{x}_i,\lambda_j)$ is a MIS balance heuristic (Subsection 2.1.5) defined as

$$w_{\lambda_h}(\bar{x}_i,\lambda_h) = \frac{p(\bar{x}_i,\lambda_h)}{\sum_{k=1}^{C}p(\bar{x}_i,\lambda_k)}. \tag{2.51}$$

In contrast to the technique developed by Evans and McCool [8], a path's probability density again becomes dependent on the wavelength it is propagated for, i.e. the PDF becomes

$$p(\bar{x},\lambda) = p(\lambda)\cdot p(\bar{x}\mid\lambda), \tag{2.52}$$

As hero wavelength sampling allows multiple wavelengths to propagate past non-specular materials without computing additional light paths, this technique can show significant gains above naive spectral rendering. Even so, no improvement is made to the rendering of perfectly specular wavelength-dependent materials.

2.2.4. Spectral Gradient Sampling

A more recent development, only partially related to the work we have discussed so far, is based on the seminal work of *gradient-domain rendering* [12, 15], which was recently adapted for spectral rendering by Petitjean et al. [20]. We briefly cover gradient-domain rendering before we show its application to spectral rendering.

Gradient-domain rendering is a technique where, while rendering an image, we additionally build an estimate of the image's gradient. When tracing a light path $\bar{x}$ in a pixel i, most variations of the technique generate a second, correlated light path in a neighboring pixel j through a *shift mapping*

$$\bar{x}' = T_{i\to j}(\bar{x}). \tag{2.53}$$

The shift mapping is a *bijective* function that aims to keep both light paths $\bar{x}$ and $\bar{x}'$ correlated, even given their different origins. Many different shift mappings have been proposed with different advantages for different scenarios [12, 17]. A simple shift mapping defined by Kettunnen et al. [12] lets both paths make similar choices at specular materials by sharing reflection half vectors. When two consecutive diffuse scatterings occur, the paths are quickly reconnected. An estimate is then constructed for the image gradient Δ_{ij}, using the difference between the two paths' measured contributions:

$$\Delta_{ij} = \int_{\Omega_i} (f_i(\bar{x}) - f_j(T_{i\to j}(\bar{x}))) \; |T'_{i\to j}| \; d\mu(\bar{x}) \tag{2.54}$$

where $|T'_{i\to j}|$ is a Jacobian determinant to account for the change-of-variables that occurs between Ω_i and Ω_j. Afterwards, a *screened poisson reconstruction* [3] is performed over the combined rendered image and gradient estimate, likely recovering an image with significantly reduced error.

An adapted version of gradient-domain rendering was developed by Petitjean et al. [20] which instead of an image gradient, directly estimates the *spectral gradient* of incoming radiance I on a per-pixel basis. The spectral gradient is defined as the measured difference between radiances I_λ and $I_{\lambda'}$ for two wavelengths λ and $\lambda' = \lambda + \delta$ separated by a small offset δ. Application of this principle to Equation 2.55 results in the following formulation for the spectral gradient:

$$\Delta_{\lambda\lambda'} = \int_{\Omega_\lambda} (f(\bar{x}, \lambda) - f(S_{\lambda\to\lambda'}(\bar{x}), \lambda')) \; |S'_{\lambda\to\lambda'}| \; d\mu(\bar{x}) \tag{2.55}$$

where $S_{\lambda\to\lambda'}$ denotes a *spectral shift mapping* which shifts a light path with associated wavelength λ to the shifted wavelength λ'. This shift mapping is highly similar to the one used by Kettunen et al. [12], but has both light paths originate at the same point, and the shift is only performed when a wavelength-dependent phenomenom is encountered. Paths are similarly reconnected as two consecutive diffuse materials are encountered. We illustrate this shift mapping in Figure 2.4.

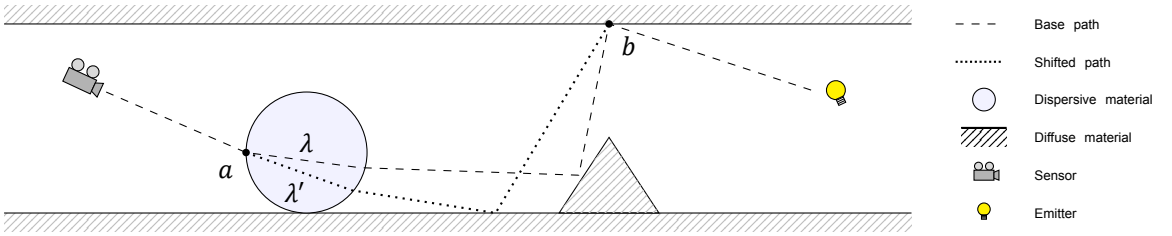
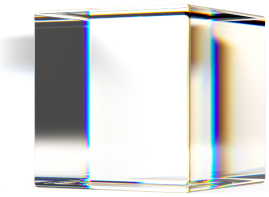


Figure 2.4: **Spectral shift mapping** On an encounter with a dispersive medium at a, the shifted path is generated for wavelength λ', which is reconnected with the base path with wavelength λ on a second consecutive diffuse scattering at b.

Although spectral gradient sampling shows significant gains for a variety of scenarios, it is explicitly noted by Petitjean et al. [20] to be an inherently inefficient technique for handling of non-uniform emission spectra. Should a main path's wavelength sample be located at an emitter's *peak*, then the shifted path likely falls off said peak, and will contribute significantly less.



3 Problem Analysis

Having covered the relevant background material in Chapter 2, we can now properly describe the potential problems our method aims to solve. Recall how we first noted in Section 1.2 that an estimator for spectral Monte Carlo light transport explicitly requires a suitable distribution for sampling the spectral domain Λ. Indeed, this distribution $p(\lambda)$ becomes visible in the standard Monte Carlo estimator if, once again, we decompose its distribution for sampling path-wavelength pairs as

$$p(\bar{x}, \lambda) = p(\lambda) \cdot p(\bar{x} \mid \lambda) \tag{2.52}$$

The choice of $p(\lambda)$ matters only if wavelength dependency occurs along a path, as we can otherwise just propagate multiple wavelengths as discussed in Subsection 2.2.3. Optimally, the chosen distribution's sampling density is proportional to the integral's value, i.e.

$$p(\lambda) \propto I_j, \tag{3.1}$$

as is generally the case with importance sampling. By necessity, however, many implementations of spectral light transport, provided in renderers such as LuxCoreRender [1], Mental Ray [18], and Mitsuba [35], leverage a uniform distribution, in which case proportionality would hold only if all wavelengths contribute equally. Such a situation is circumstantial at best, given the likely presence of non-uniform spectra in a scene. We mentioned in Section 1.2 that, in their work on wavelength clustering, Evans and McCool [8] leveragea scene distribution to alleviate this *wavelength sampling problem*. They select one of a scene's emitters, to leverage it's SPD $S(\lambda)$ for wavelength sampling. This strategy naturally works well for single-emitter scenes, as it is then likely that

$$S(\lambda) \propto\!\!\!\sim I_j. \tag{3.2}$$

Unfortunately, as we illustrate in Figure 3.1, such a strategy may not be a robust solution for rendering a scene with multiple different emitters. Evans and McCool state that, in a multiple-emitter scenario, they randomly select an emitter for sampling, which leads to the problem we discussed in Subsection 2.1.5.

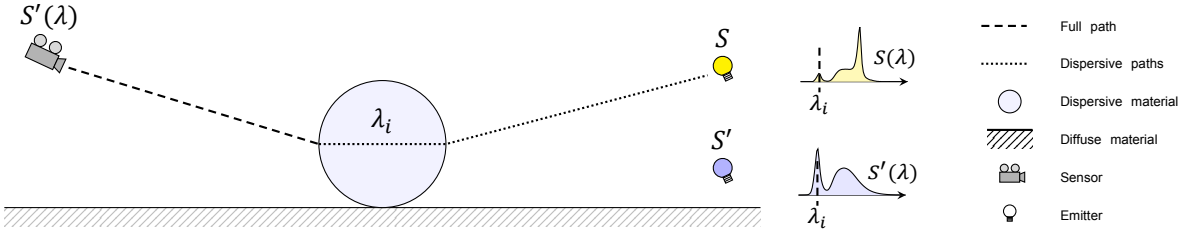


Figure 3.1: **Problematic sampling strategy.** A path is evaluated for a wavelength λ_i which was sampled from distribution $S'(\lambda)$, but instead encounters emitter S, and therefore hardly contributes energy as it falls off major contributing parts of $S(\lambda)$.

When importance sampling, a poorly chosen distribution can lead to increased variance. If a path-wavelength is sampled from one emitter's distribution, and the path encounters another emitter with an entirely different distribution, results are at best suboptimal. We do not know beforehand which emitter our path encounters, which makes selecting the right sampling distribution difficult.

This is mirrored by a secondary problem Evans and McCool did not consider, which is the presence of absorption through non-uniform reflectance or transmission along a light path, which would modify the constitution of I_j in a similar manner to non-uniform emission spectra. In the simplest scenario, where we have a uniform emitter SPD, and our wavelength-dependent light path encounters a single non-uniform reflectance spectrum $R(\lambda)$ during scattering, we can naturally apply importance sampling, given that

$$R(\lambda) \propto I_j. \tag{3.3}$$

If we introduce multiple non-uniform spectra along a path, the situation becomes all too familiar. Whichever spectrum we select for importance sampling in this case - we do not know for certain which is or are encountered beforehand - can potentially lead to a variance increase. We again illustrate a simple scenario in Figure 3.2. Here, even uniform sampling would be a poor choice, as only a small range of wavelengths will not be largely absorbed.

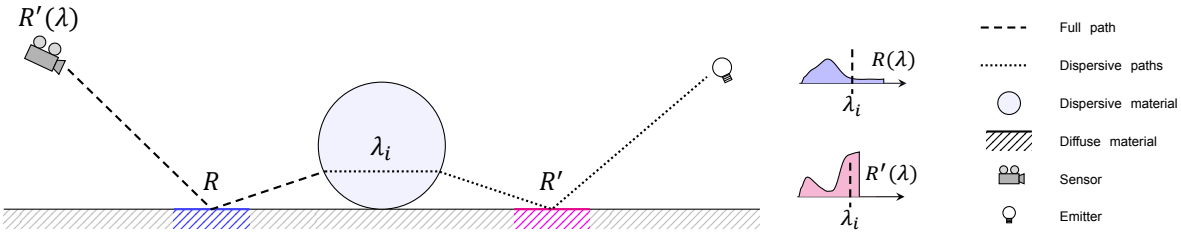


Figure 3.2: **Problematic sampling strategy.** A path is evaluated for a wavelength λ_i which was sampled from distribution $R'(\lambda)$, but encounters a surface reflectance R, and therefore hardly contributes energy as it is largely absorbed as per $R(\lambda)$.

These importance sampling strategies are indicative of a larger underlying problem. In simple situations, we can viably leverage a scene distribution that is suitable for importance sampling, but this becomes increasingly difficult as more spectra are potentially involved in a path's contribution. Both the scenarios we have described are problematic on their own, but a realistic and complicated scene likely contains many different variations and combinations of these scenarios. Furthermore, we have only considered a single pixel. If we were to base a suitable distribution $p(\lambda)$ on the spectral distributions present in a scene, this might be suited for one pixel, but not another. Formally:

$$p(\lambda) \propto I_j \quad \nrightarrow \quad p(\lambda) \propto I_k \quad : \quad j \neq k. \tag{3.4}$$

which means we have to either consider many different pixel-dependent distributions, or find a single distribution such that

$$\forall_j \; p(\lambda) \propto I_j, \tag{3.5}$$

the existence of which is highly unlikely, as it is almost trivial to create a scenario where different pixels observe different parts of a scene. Given this, and the other factors we discussed, finding a suitable wavelength sampling distribution clearly poses a difficult problem. To our knowledge, no method yet exists that selects or generates a suitable distribution or set of distributions which, in all of the aforementioned scenarios, proves sufficiently better than uniform wavelength sampling, and at the very least will not perform worse. We expect significant gains from such a method, given the improvements importance sampling is known [30] to provide.

4 Methodology

We now describe the formal basis of our method, which, as we will show, provides an efficient and unbiased solution to the *wavelength sampling problem* we described in Chapter 3. With *pre-estimated spectral rendering*, we provide a simple extension to spectral Monte Carlo light transport that first generates a sampling distribution of sufficient quality, such that we can then importance sample the spectral domain. We will focus on deriving the different components of our method by extending a basic path tracer, although our method is likely orthogonal to state-of-the-art methods such as hero wavelength sampling [38] and spectral gradient sampling [20], and may be extended to fit their respective estimators.

First, recall that a suitable distribution $p(\lambda)$ preferably has a sampling density of such a shape that

$$\forall_j \; p(\lambda) \overset{\propto}{\sim} I_j. \tag{3.5}$$

To obtain an ideal sampling distribution for even a single pixel, we would have to know the value of I_j beforehand. Although this does not sound inherently practical for obvious reasons, the concept is actually useful to us. Consider that a direct estimate may prove to be extremely convenient for importance sampling, as it incorporates the effects of different emission spectra, reflectance spectra, and wavelength-dependent phenomena, and thus perfectly accomodates the different scenarios we described in Chapter 3.

Following this, we propose to append two brief preprocessing stages to conventional light transport algorithms. In the first stage, which we describe in Section 4.1, we generate a fast and rough approximation of the image called the *pre-estimate* $\tilde{I}$. In the second stage, we pass this pre-estimate through a reconstruction function r which essentially filters and restricts the generated distributions. We discussed in Subsection 2.1.5 that importance sampling becomes detrimental if a distribution is of insufficient quality. As we wish to avoid potential bias, the reconstruction function, which we cover extensively in Section 4.2, will act as a safeguard.

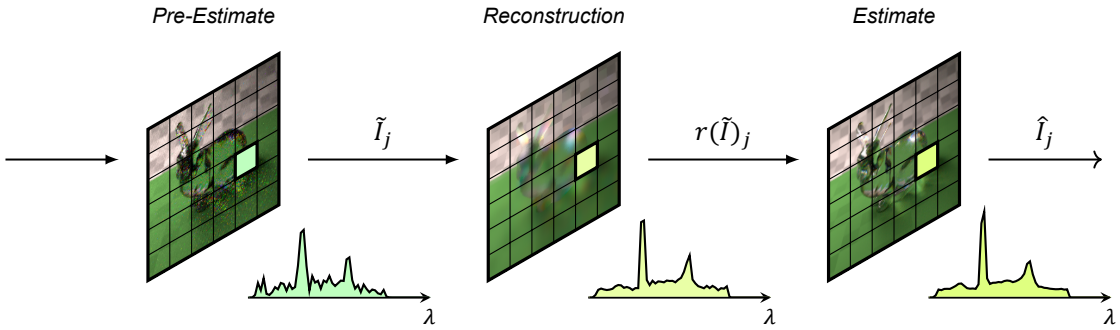


Figure 4.1: **Pre-Estimated Rendering.** For every pixel j, we compute a fast and rough approximation $\tilde{I}_j$, which we pass through a reconstruction function r for filtering, and afterwards importance sample for wavelengths, ultimately rendering $\hat{I}_j$ more efficiently.

After preprocessing, we draw distributions from the reconstructed pre-estimate $r(\tilde{I})$ for wavelength sampling. We modify a basic estimator to leverage pixel-specific distributions as

$$\hat{I}_j = \frac{1}{N} \sum_{i=1}^{N} \frac{f_j(\bar{x}_i, \lambda_i)}{p_j(\bar{x}_i, \lambda_i)}, \tag{4.1}$$

where we decompose $p_j(\bar{x}, \lambda)$, in a manner similar to Equation 2.52, as

$$p_j(\bar{x}, \lambda) = r(\tilde{I})_j(\lambda) \cdot p(\bar{x} \mid \lambda). \tag{4.2}$$

If we return to Equation 3.5, we note that, although it is unlikely that we could find a distribution that is useful across all pixels, we are instead generating a large set of distributions such that

$$\forall_j \; \exists \; p_j(\lambda) \quad : \quad p_j(\lambda) \propto\!\!\!\sim I_j. \tag{4.3}$$

We illustrate the basic concept of our method in Figure 4.1. Finally, in addition to the two preprocessing stages we have introduced, we describe an extension to this concept where we iteratively apply each stage in *layers* in Section 4.3.

4.1. A Low-Cost Pre-Estimate

Given that $\tilde{I}$ should account for all scenarios $\hat{I}$ would account for, we essentially recycle the basic Monte Carlo estimator for the pre-estimate, but must consider that this may not be inherently useful. In sufficently complicated scenes with dispersive effects, large sample rates may be required for the pre-estimate to even be usable. Unfortunately, any time spent preprocessing is bound to introduce a significant (but constant) runtime overhead into our method, which negatively impacts the improvements we can bring, especially for low sample rates. To reduce this overhead, we introduce cost-lowering optimizations, and rely heavily on the reconstruction function to recover usable distributions. To generate the pre-estimate, we query an arbitrary light transport estimator (a path tracer in our case) with a separate number of samples $\widetilde{N}$, and additionally allow this to be performed in a lower resolution. Formally, we define the pre-estimate as

$$\tilde{I}_j^{\downarrow} = \frac{1}{\widetilde{N}} \sum_{i=1}^{\widetilde{N}} \frac{f_j^{\downarrow}(\bar{x}_i, \lambda_i)}{p(\bar{x}_i, \lambda_i)}, \tag{4.4}$$

where $\frac{\downarrow}{j}$ indicates a pixel coordinate in a lower-resolution image, which will have to be resampled. We take resampling into account later on in the reconstruction function, but briefly cover the potential impact resolution has on error in the pre-estimate.

It stands to reason that considerable high-frequency detail is lost in lower resolution images, especially for scene boundaries and minute prismatic effects. We compare the different convergence rates of possible high, medium and low resolution estimators in Figure 4.2, where we show that, in essence, lower-resolution estimators are *biased*. As we briefly mentioned in Subsection 2.1.2, bias can at times be advantageous. Although the low-resolution estimators in this comparison have limited potential in high-frequency areas, the introduced bias leads to significantly (and temporarily) lower error especially for low-frequency areas, which we leverage.

This error decrease can occur as, at the same time an image's resolution is downscaled by a factor c, we can increase the sample rate $\widetilde{N}$ by the same factor, essentially keeping the total number of samples (across the entire image), and therefore render times, equal. Following on the convergence rate for an estimator we covered in Subsection 2.1.2, such a lower-resolution pre-estimate is bound by:

$$\sigma[\tilde{I}_j^{\downarrow}] = \frac{1}{\sqrt{c\widetilde{N}}} \, \sigma\left[\frac{f_j^{\downarrow}(\bar{x}, \lambda)}{p(\bar{x}, \lambda)}\right]. \tag{4.5}$$

If we assume a worst case scenario, there is some constant $d \geq 1$ such that

$$\sigma\left[\frac{f_j^{\downarrow}(\bar{x}, \lambda)}{p(\bar{x}, \lambda)}\right] = d \cdot \sigma\left[\frac{f_j(\bar{x}, \lambda)}{p(\bar{x}, \lambda)}\right], \tag{4.6}$$

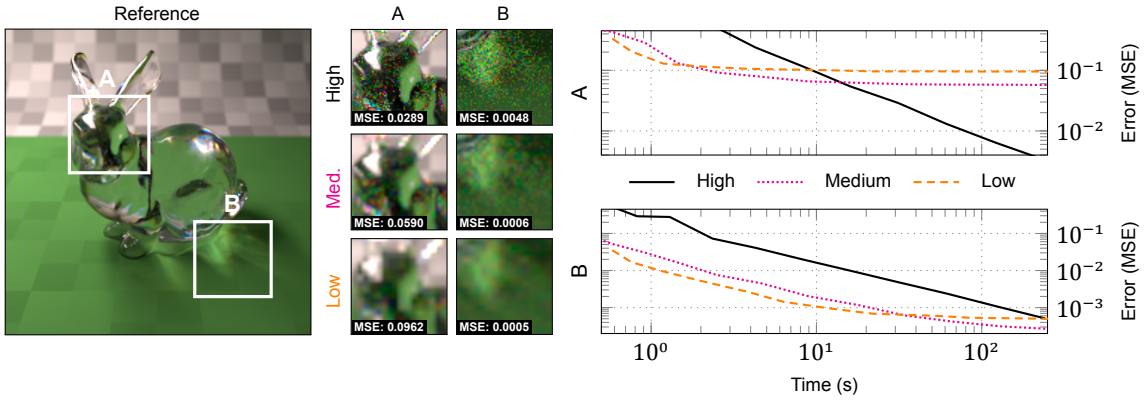


Figure 4.2: **Convergence rates.** We compare convergence rates of high (256×256), medium (128×128) and low (64×64) resolution estimators, paired with linear resampling, over two insets A and B of a high-resolution reference produced with an unbiased path tracer ($N \approx 200K$). Both Error and Time axes are logarithmic. Inset A demonstrates that the unbiased high resolution estimator is the sole estimator to converge in high-frequency areas, while inset B demonstrates that the biased low-resolution estimators provide a temporary advantage in low-frequency areas.

i.e. a lower-resolution integrand never directly exhibits a lower standard deviation than a normal-resolution integrand. Given this, a low-resolution pre-estimate that still manages to be advantageous must then be bound by the inequality

$$\begin{aligned} \frac{1}{\sqrt{\tilde{N}}}\,\sigma\left[\frac{f_j(\bar{x},\lambda)}{p(\bar{x},\lambda)}\right] &\geq \frac{1}{\sqrt{c\tilde{N}}}\,\sigma\left[\frac{f_j^{\downarrow}(\bar{x},\lambda)}{p(\bar{x},\lambda)}\right], \\ &\geq \frac{d}{\sqrt{c\tilde{N}}}\,\sigma\left[\frac{f_j(\bar{x},\lambda)}{p(\bar{x},\lambda)}\right], \end{aligned} \tag{4.7}$$

which we reformulate as

$$\begin{aligned} \frac{1}{\sqrt{\tilde{N}}} &\geq \frac{d}{\sqrt{c\tilde{N}}}, \\ \frac{\sqrt{c}}{\sqrt{\tilde{N}}} &\geq \frac{d}{\sqrt{\tilde{N}}}, \end{aligned} \tag{4.8}$$

to obtain

$$\sqrt{c} \geq d. \tag{4.9}$$

In other words, in order for a lower-resolution pre-estimate to be advantageous, this inequality must hold. Given that c and d are not linearly related, this is only the case when d is sufficiently small, as is demonstrated in a low-frequency area like inset B, in Figure 4.2. We may be able to use lower-resolution estimators to our advantage in similar scenarios, and will evaluate this in Chapter 5.

4.2. The Reconstruction Function

The reconstruction function r receives an unfiltered and erroneous pre-estimate $\tilde{I}$, which is likely too approximate for sampling directly, and is potentially of a different resolution than the intended target image. As such, we leverage r, which fulfills several purposes, for recovering suitable distributions, by decomposing it as

$$r(\tilde{I}) = r_{res}(r_{def}(r_{filt}(\tilde{I}))). \tag{4.10}$$

We can considerably reduce the estimation error through an application of image filtering techniques with r_{filt}, and afterwards apply edge-aware resampling using r_{res}. In addition, we apply a defensive sampling technique, which ensures that the final distributions are at least adequate for importance sampling, through r_{def}.

4.2.1. Filtering and Resampling

Of the selection of image noise filtering algorithms available, the most adaptable is currently *non-local means filtering* (NL-means) [5], which we thus use for the reconstruction function. This is an edge-aware filter which is considered (relatively) computationally expensive, and may introduce significant overhead, so we additionally consider the more efficient *joint bilateral filter* [7, 21]. This is an image filter that leverages a secondary *guidance image* for edge detection. Fortunately, we are in a situation where suitable guidance images, such as direct depth and normals, are readily available.

We first demonstrate the NL-means filter in the reconstruction function. NL-means filtering leverages the concept of *image patches*, which are small collections of pixels throughout an image, and compares these patches to extract information. We use A to represent the total *area* of an image (i.e. the area spanning all pixels), and define j and k as two separate pixels in A. The filter becomes

$$\begin{aligned} r_{filt}(\tilde{I}^{\downarrow})_j &= \frac{1}{w_j} \sum_{k \in A} G_\sigma(\|P_k - P_j\|^2)\ \tilde{I}^{\downarrow}_k \\ \because \quad w_j &= \frac{1}{w_j} \sum_{k \in A} G_\sigma(\|P_k - P_j\|^2), \end{aligned} \tag{4.11}$$

where G_σ is a simple (discrete) gaussian filter with standard deviation σ. The weight w_j is a normalization factor that serves to ensure the sum of all weights is equal to 1, even in a discrete filter. P_j and P_k are image patches, which are local neighborhoods of size $2f+1 \times 2f+1$ of the form

$$P_{j,k} = \begin{bmatrix} \tilde{I}^{\downarrow}_{j-f,k-f} & \cdots & \tilde{I}^{\downarrow}_{j+f,k-f} \\ \vdots & \ddots & \vdots \\ \tilde{I}^{\downarrow}_{j-f,k+f} & \cdots & \tilde{I}^{\downarrow}_{j+f,k+f} \end{bmatrix}. \tag{4.12}$$

An important detail of NL-means filtering is that, formally, as it is applied to a full image, it considers and compares patches across the entire image. Given that this may become extremely costly for even moderately sized images, implementations typically restrict the area of A around a pixel j. Buades et al. [5] mention that typical area sizes range between 15×15 and 21×21 pixels, and consider $f = 5$ and $f = 7$ to be suitable matching neighborhood sizes.

We now consider joint bilateral filtering, which was simultaneously developed by Eisemann and Durand [7] and Petschnigg et al. [21], and which may provide faster image filtering, as it only considers a limited local neighborhood Ω around a specific pixel. We define the joint bilateral filter as

$$\begin{aligned} r_{filt}(\tilde{I}^{\downarrow})_j &= \frac{1}{w_j} \sum_{k \in \Omega} G_{\sigma_s}(\|k-j\|)\ G_{\sigma_r}(\|I'_k - I'_j\|)\ \tilde{I}^{\downarrow}_k \\ \because \quad w_j &= \sum_{k \in \Omega} G_{\sigma_s}(\|k-j\|)\ G_{\sigma_r}(\|I'_k - I'_j\|), \end{aligned} \tag{4.13}$$

where G_{σ_s} and G_{σ_r} are *range* and *spatial* gaussian filters with respective standard deviations σ_s and σ_r, and w_j again serves as a normalization factor. As we previously mentioned, joint bilateral filtering leverages a secondary guidance image, which we denote as I', for which we query secondary scene information. Fortunately, most renderer implementations provide access to a variety of guidance image candidates, such as depth, normals, and reflectance values. As a single candidate may not provide an optimal result, we combine n different guidance images $I'_1, \ldots, I'_n$, such that

$$I'_j = \sum_{i=1}^{n} w_i\ (I'_i)_j \quad \because \quad \sum_{i=1}^{n} w_i = 1 \tag{4.14}$$

where $w_1, \ldots, w_n$ are specific chosen weights for the different guidance images, although an average likely suffices. We illustrate an example combination of guidance images in Figure 4.3.

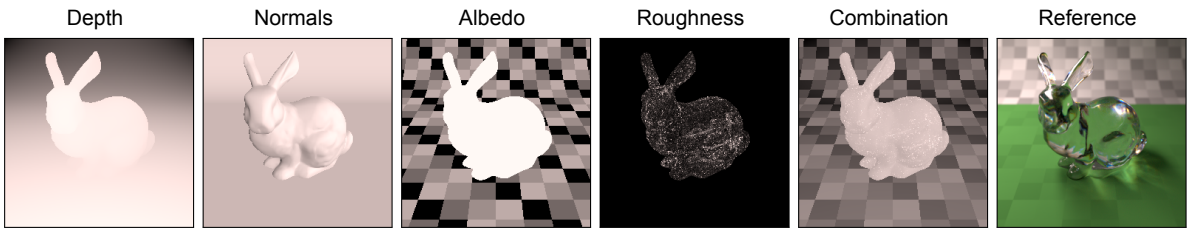


Figure 4.3: **Guidance images.** We show guidance image candidates for the joint bilateral filter. Although no individual guidance image correctly identifies all edges, these are all present in the combined guidance image.

We perform resampling in a similar manner, in cases where the pre-estimate is generated in a lowered resolution, as we opt for a slightly modified version of a edge-preserving joint bilateral upsampling [13] technique, instead of linear resampling. We define resampling as

$$r_{res}(\tilde{I}^{\downarrow})_j = \frac{1}{w_j} \sum_{k \in \Omega} G_{\sigma_s}(\|k_\downarrow - j_\downarrow\|)\; G_{\sigma_r}(\|I'_k - I'_j\|)\; r_{def}(\tilde{I}^{\downarrow})_{k_\downarrow}$$
$$\vdots \quad w_j = \sum_{k \in \Omega} G_{\sigma_s}(\|k_\downarrow - j_\downarrow\|)\; G_{\sigma_r}(\|I'_k - I'_j\|), \tag{4.15}$$

where $j_\downarrow$ and $k_\downarrow$ again denote pixel-coordinates in the lower-resolution image. For direct queries to $\tilde{I}^{\downarrow}$, we apply linear resampling.

4.2.2. Defensive Sampling

As previously mentioned, it is vital that the distribution $p_j(\lambda)$ is sufficiently proportionate to the spectral radiance we are estimating, as importance sampling may otherwise become detrimental. Unfortunately, we have no guarantees of this, especially given that we generate said distribution using a biased estimator. To address these concerns, we apply the defensive sampling technique we covered in Subsection 2.1.5, mixing recovered spectral distributions with a base distribution we consider to be safe. Although a uniform distribution might suffice, not necessarily all wavelengths are guaranteed to partake in a scene, depending on the specific configuration of emitters. Assuming we can query all of a scene's emitter SPDs or other representative spectra during preprocessing, we instead generate a base distribution b as

$$\forall_\lambda\; b(\lambda) = \begin{cases} 1 & \text{if } \exists\; S : S(\lambda) \neq 0, \\ 0 & \text{else.} \end{cases} \tag{4.16}$$

The mixture then consists of a simple interpolation

$$r_{def}(\tilde{I}^{\downarrow})_j = \alpha \cdot \frac{b}{\|b\|} + (1-\alpha) \cdot \frac{r_{filt}(\tilde{I}^{\downarrow})_j}{\|r_{filt}(\tilde{I}^{\downarrow})_j\|}. \tag{4.17}$$

Depending on the choice of α, the effectiveness of our technique may be reduced in the case of a good estimate, or improved in the case of a bad estimate. We evaluate a good choice of α in Chapter 5. We note that the safe distribution b is potentially problematic if we simulate fluorescent effects, as we could then not predict which wavelengths we should evaluate based on emitter SPDs alone. Fortunately, our focus lies on dispersive effects, and fluorescence remains outside the scope of this thesis.

4.3. Layered Pre-Estimates

Following on the three-stage defintion of our method, we define a natural extension where we iteratively generate pre-estimates of an increasing resolution. Consider that, if use of a pre-estimated considerably improves the convergence rate of our actual estimate, a second pre-estimate may logically improve the first beyond an increased sample rate $\tilde{N}$, while a third can again improve the second, and so on. We illustrate the concept of a layered pre-estimate in Figure 4.4.

While we can constrain ourselves to a fixed number of layers, we instead provide a generalized definition using a list of K consecutive pre-estimates

$$\tilde{I} = \tilde{I}^1, \ldots, \tilde{I}^K, \tag{4.18}$$

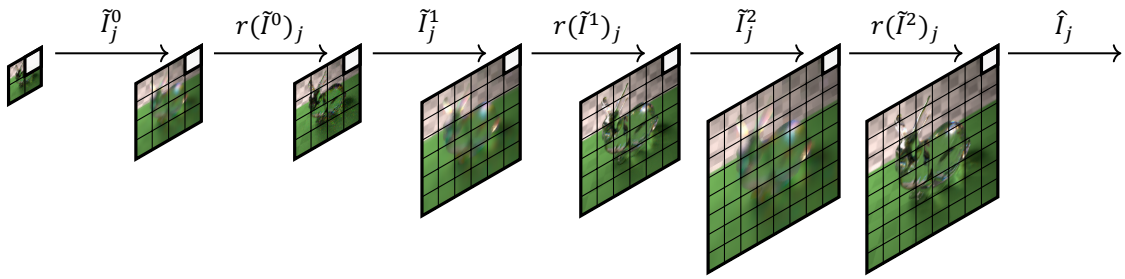


Figure 4.4: **Layered Pre-Estimate.** For every pixel j, we consecutively compute a pre-estimate, pass it through the reconstruction function, and feed the resulting distribution into the next pre-estimate.

such that the k^{th} pre-estimate uses an estimator of the form

$$\tilde{I}_j^k = \frac{1}{\tilde{N}^k} \sum_{i=1}^{\tilde{N}^k} \frac{f_j^{\downarrow}(\bar{x}_i, \lambda_i)}{p_j^k(\bar{x}_i, \lambda_i)}, \tag{4.19}$$

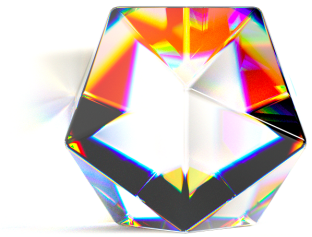
where $p_j^k(\bar{x}, \lambda)$ is another pixel-specific PDF, which in turn queries the $k-1^{\text{th}}$ pre-estimate as

$$p_j^k(\bar{x}, \lambda) = \begin{cases} r(\tilde{I}^{k-1})_j(\lambda) \cdot p(\bar{x} \mid \lambda) & k > 1, \\ p(\lambda) \cdot p(\bar{x} \mid \lambda) & k = 1. \end{cases} \tag{4.20}$$

Unfortunately, there is one potential issue that restricts the effectiveness of layering. As consecutive layers are used to estimate the layers above them, if any one layer were to result in a poor set of distributions, then following layers may accumulate this error. Although we apply defensive sampling to counteract poor distributions for a single layer, the problem will become significantly more pronounced for extremely low-resolution layers, which are unable to correctly estimate scene discontinuities. As such, we are forced to increase defensive sampling throughout lower layers, i.e.

$$\alpha^{k-1} = m \cdot \alpha^k \quad : \quad m > 1. \tag{4.21}$$

Greater use of defensive sampling will limit both negative and positive impacts of lower layers, which means errors are less likely to accumulate, but additionally means that using many layers may not be more effective than simply taking more samples. Whether this is the case, and how many layers we should viably use, is evaluated in Chapter 5.



5 Results

We are now at the point where we can evaluate our method and benchmark its effectiveness when pitted against different versions of the problems we described in Chapter 3. First, we provide brief details on our implementation. Then, in Section 5.1, we evaluate the influences of our method's different parameters. Finally, in Section 5.2, we apply our method against a variety of problem scenarios and provide a comparative benchmark that includes state-of-the-art methods.

Throughout this chapter, we provide error metrics in the form of mean squared error (MSE), which we described in Subsection 2.1.3, as this provides a good overview of both convergence rates and eventual bias (which we hope to avoid). In addition, we may at times evaluate the structural similarity index (SSIM) [34], which is a metric to establish the *perceived noise* of an image. Generally speaking, lower MSE is better, and opposite thereof, higher SSIM is better.

We implement our method in Mitsuba [35]: a conventional *C++*-based renderer commonly used within the computer graphics research community. Although Mitsuba is not in itself a true spectral renderer, we have access to a modified version that supports full spectral rendering with light dispersion, and includes state-of-the-art techniques such as hero wavelength sampling [38] and spectral gradient sampling [20]. Mitsuba employs a discrete representation for spectra, which we configure in the same manner as Petitjean et al. [20], who used 15 equally sized bins. For a comprehensive overview of Mitsuba's components, refer to the user reference manual [37] and accompanying API reference [36].

We extend Mitsuba's standard path tracer, which itself extends the base `MonteCarloIntegrator` class, as `PreEstimatedIntegrator`. As illustrated in a basic UML activity diagram in Figure 5.1, we prepend our method's components to the path tracer, by implementing the virtual `preprocess()` function. The underlying integrator provides us with direct access to Mitsuba's sophisticated multi-threading and networking layers, which we do not cover further for sobriety. We reuse the path tracer's implementation of `Li()` for radiance queries. After generating and reconstructing the pre-estimate, we deserialize and transfer the resulting image into Mitsuba's networking layer for use during rendering. We override the path tracer's implementations of `render()` and `renderBlock()` so we can obtain

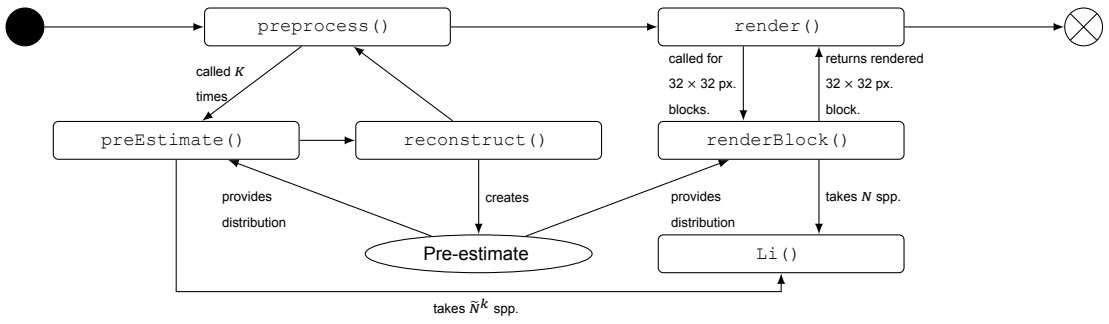


Figure 5.1: **Activity flow.** The basic activity flow of `PreEstimatedIntegrator`. Components concerned with threading, post-processing and networking are left out purposefully.

and sample spectral distributions from the pre-estimate during rendering.

5.1. Parameter Evaluation

We now carefully and systematically evaluate the effects of the different parameters of our method. In order to establish consistent results for a set of different problems, we compare convergence rates over three select insets A, B and C of a single test scene detailed in Figure 5.2.

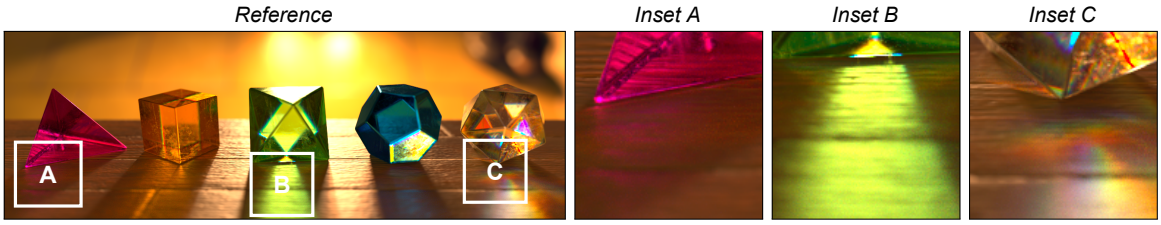


Figure 5.2: **Parameter Evaluation Scene.** This scene displays absorption inside dispersive objects, illuminated by a detailed environment map. RGB-based wood textures are obtained from FreePBR [9] and the environment map is obtained from HDRI Haven [39]. Reference produced in Mitsuba [35] using an unbiased path tracer ($N \approx 500K$). We evaluate the error over three selected insets which focus on areas of interest in the scene. Inset A and B both demonstrate significant, non-local, and non-uniform transmittance and absorption, and should be well suited to our method. Inset C demonstrates subtle prismatic effects, which may prove challenging when we use a low-resolution pre-estimate.

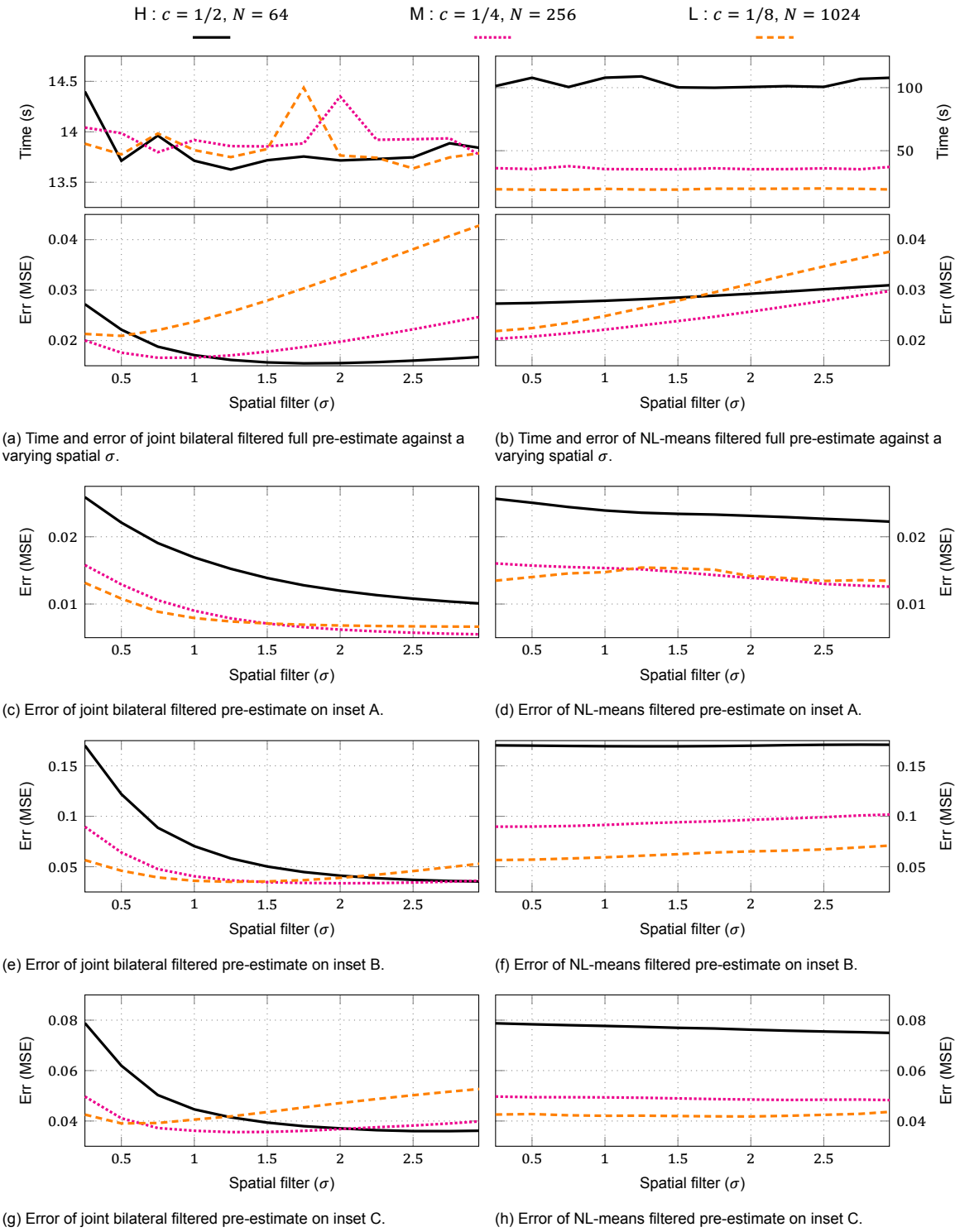
As the configuration of the different components of the reconstruction function likely depends on the configuration of the pre-estimate, we first evaluate parameters of the former using three equal-time configurations of the latter: H (scale $= 1/2$, $\tilde{N} = 64$), M (scale $= 1/4$, $\tilde{N} = 256$) and L (scale $= 1/8$, $\tilde{N} = 1024$). We found in preliminary tests that greater scaling become too costly, and smaller scaling offers reduced benefits in certain areas. For defensive sampling, we select $\alpha = 0.1$ as a suitable value based on literature [19]. For bilateral filtering and resampling, we manually select the range parameter $\sigma_r = 0.015$ as a good fit for our particular guidance images.

Configuration	H	M	L
Scale (c)	1/2	1/4	1/8
Samples ($\tilde{N}$)	64	256	1024
Filtering (σ_s)	2.5	1.5	0.75
Resampling (σ_s)	2.5	2.0	2.5

Table 5.1: **Filtering and resampling parameters.** Selected spatial filter parameters for filtering and resampling, for three equal-time configurations of the pre-estimate.

We evaluate filtering and resampling function parameters for each configuration in Figure 5.3 and Figure 5.4. At $t > 25s$, NL-means filtering adds significant overhead when compared to joint bilateral filtering ($t \approx 14s$), while providing similar error reductions. We select the joint bilateral filter going forward, and list the selected filtering and resampling σ_s parameter values in Table 5.1. We compare convergence rates of the three configurations with these parameters in Figure 5.5. Although differences are small, we select the medium-resolution configuration M, as this is the most consistent across all three insets. Using this configuration, we then show the influence of the α parameter for defensive sampling in Figure 5.6. We select $\alpha = 0.1$ as, although $\alpha = 0.05$ performs best for two out of three insets, it performs significantly worse than any higher α value in the remaining inset. In Figure 5.7, we determine that $\tilde{N} = 256$ provides a good tradeoff between overhead and good convergence rates, as any higher values show diminishing returns.

Finally, in Figure 5.8, we evaluate the layering approach we described in Section 4.3. As we noted there, we increase defensive sampling α for each layer above the starting layer with a factor $m = 2$, and similarly double the resolution downscaling. We keep sample rates consistent across layers. Based on results, we select $K = 3$ as a suitable number of layers.



(a) Time and error of joint bilateral filtered full pre-estimate against a varying spatial σ.

(b) Time and error of NL-means filtered full pre-estimate against a varying spatial σ.

(c) Error of joint bilateral filtered pre-estimate on inset A.

(d) Error of NL-means filtered pre-estimate on inset A.

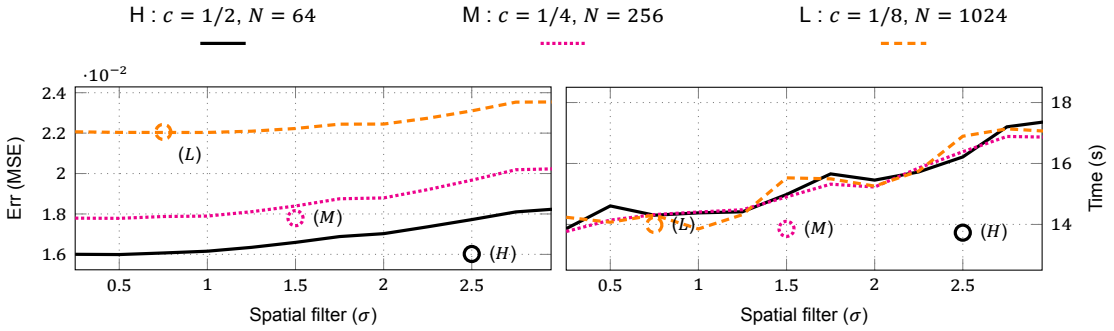
(e) Error of joint bilateral filtered pre-estimate on inset B.

(f) Error of NL-means filtered pre-estimate on inset B.

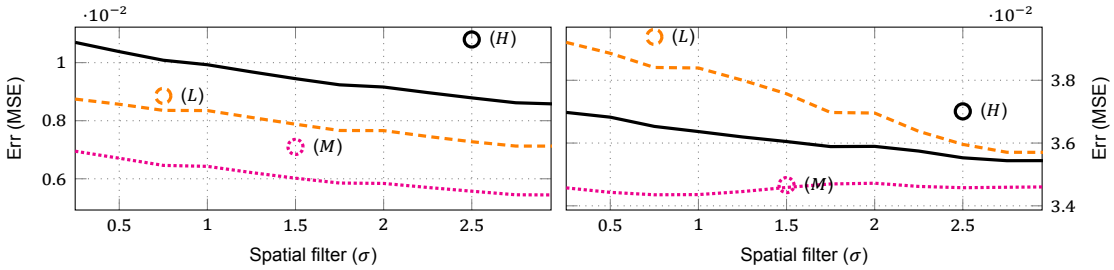
(g) Error of joint bilateral filtered pre-estimate on inset C.

(h) Error of NL-means filtered pre-estimate on inset C.

Figure 5.3: **Filtering function evaluation.** We evaluate spatial filter parameters for both joint bilateral and NL-means filtering. We manually select the bilateral filter's range parameter $\sigma_r = 0.015$, as this suits our guidance image well. We compare runtimes and errors for three different (high, medium and low-resolution) scalings of the pre-estimate. Based on results, we select joint bilateral filtering as a suitable, low-cost filter, especially given the significant overhead NL-means filtering introduces. We list the selected filtering parameters for each configuration in Table 5.1.

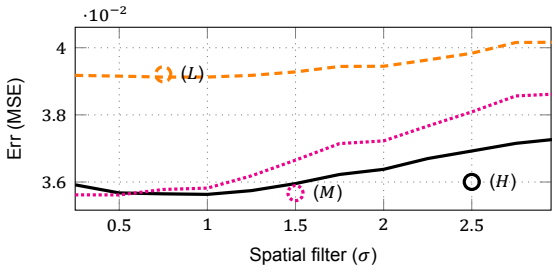


(a) Time and error of joint bilateral resampled full pre-estimate for varying spatial σ, compared to linear resampling.



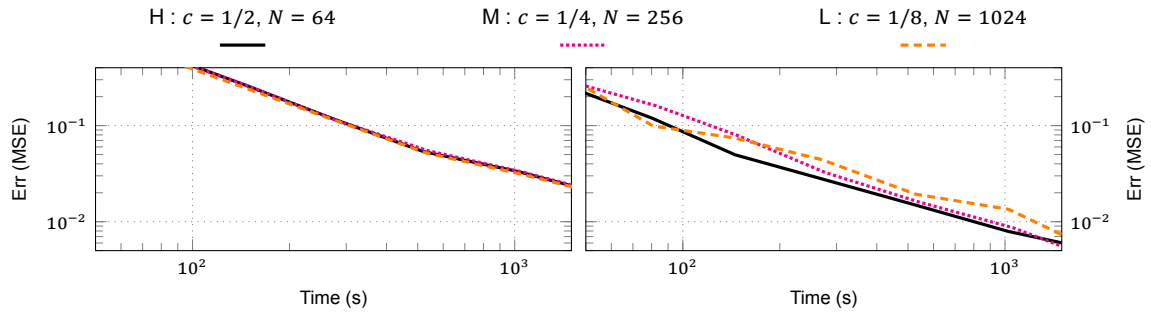
(b) Error of joint bilateral resampled pre-estimate on inset A, compared to linear resampling.

(c) Error of joint bilateral resampled pre-estimate on inset B, compared to linear resampling.



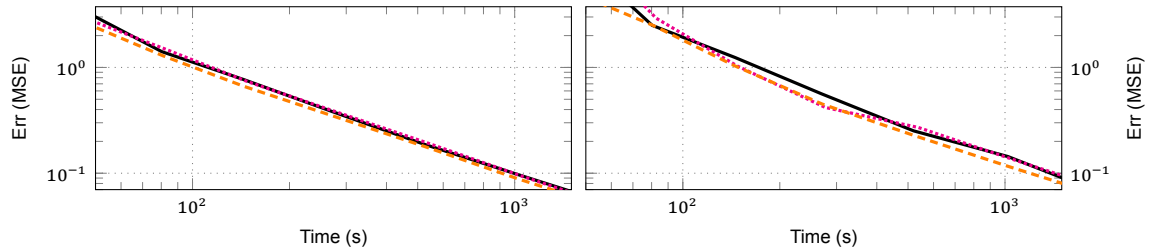
(d) Error of joint bilateral resampled pre-estimate on inset C, compared to linear resampling.

Figure 5.4: **Resampling function evaluation.** We evaluate the spatial filter parameter for joint bilateral resampling, in comparison to linear resampling, which is dot-marked for a single σ in each figure. We manually select the range parameter $\sigma_r = 0.015$, as this suits our guidance image well. We compare runtimes and errors for three different (high, medium and low-resolution) scalings of the pre-estimate. We list the selected resampling parameters for each configuration in Table 5.1.



(a) Time against error over full image, for different scalings.

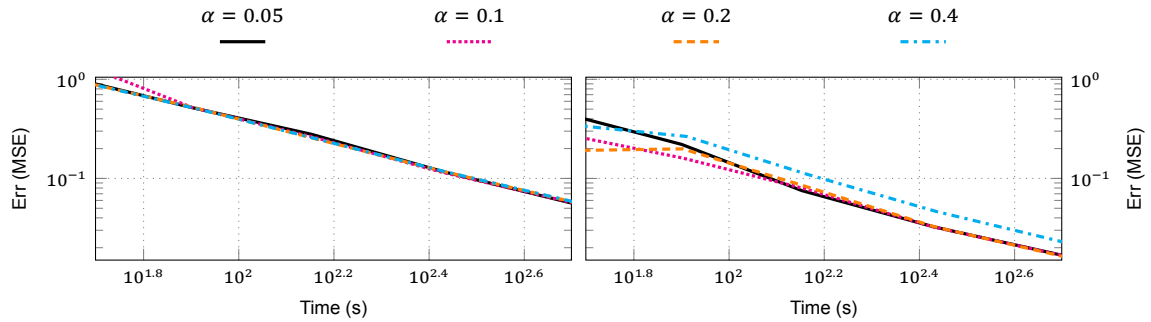
(b) Time against error over inset A, for different scalings.



(c) Time against error over inset B, for different scalings.

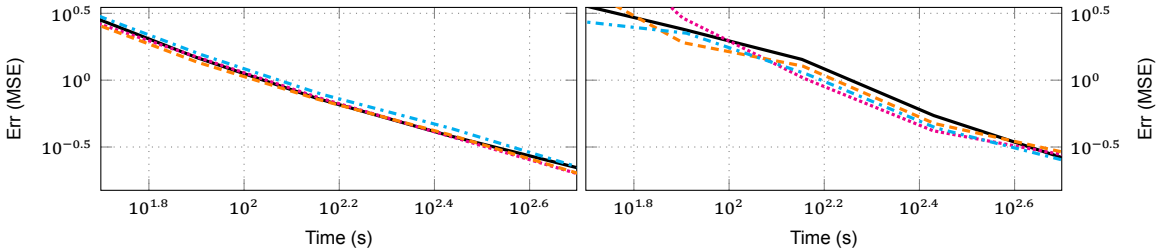
(d) Time against error over inset C, for different scalings.

Figure 5.5: **Scaling configuration evaluation.** We compare use of the different pre-estimate scaling configurations on final estimate convergence rate, given use of the reconstruction filter results listed in Table 5.1. Although differences are at best minute (and at worst indistinguishable), we select the medium configuration M as this provides consistent results across all three insets.



(a) Time against error over full image, for different α.

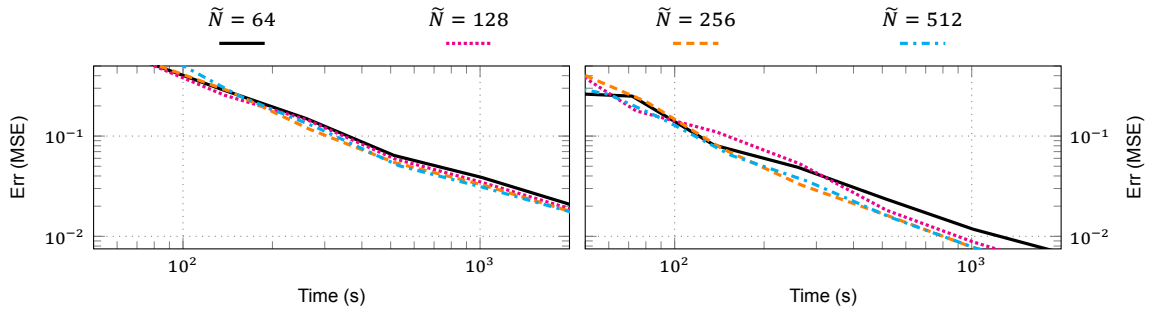
(b) Time against error over inset A, for different α.



(c) Time against error over inset B, for different α.

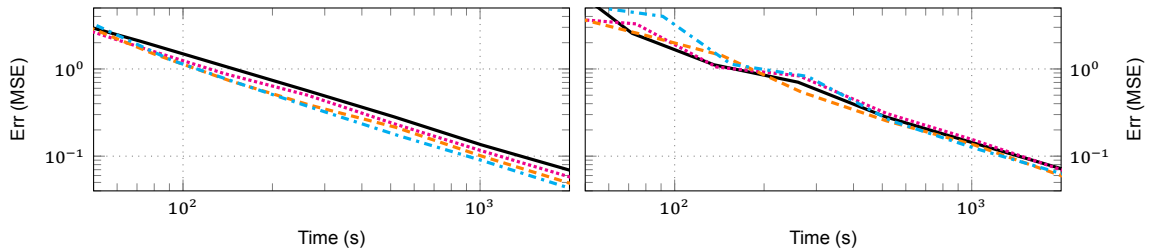
(d) Time against error over inset C, for different α.

Figure 5.6: **Defensive sampling evaluation.** We compare the influence of different levels of the defensive mixture α parameter through. We have previously used $\alpha = 0.1$, based on literature [19], and now establish this as a suitable value. Lower α increases error in inset C, while higher α increases error in insets A and B.



(a) Time against error over full image, for different N.

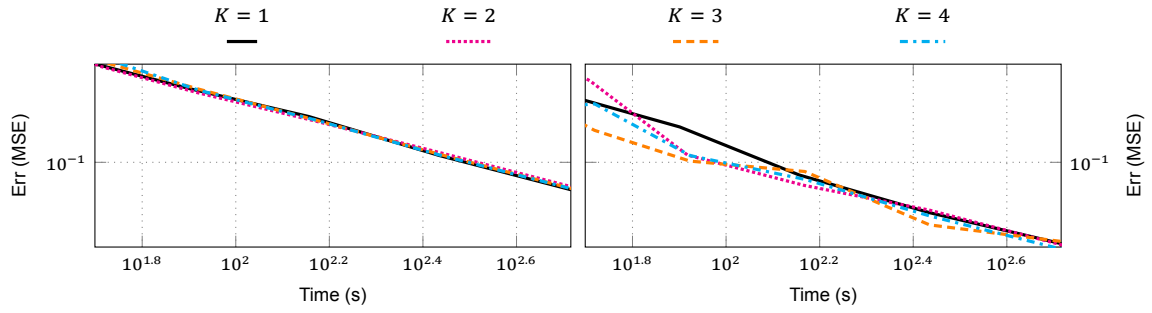
(b) Time against error over inset A, for different N.



(c) Time against error over inset B, for different N.

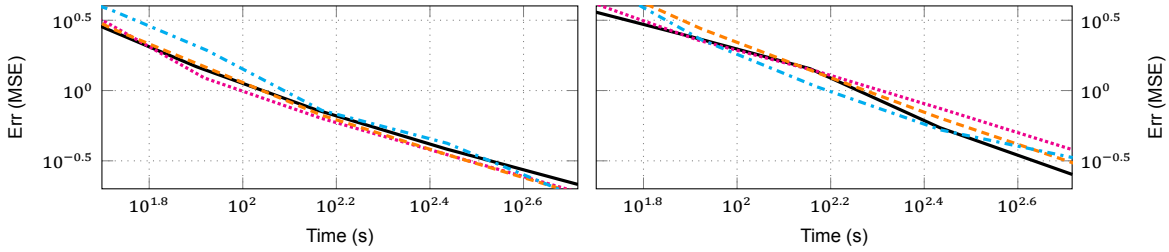
(d) Time against error over inset C, for different N.

Figure 5.7: **Sample rate parameter evaluation.** We compare the influence of different sample rates $\widetilde{N}$ for the pre-estimate. Based on results, we establish $\widetilde{N} = 256$ as a suitable value. Higher pre-estimate sample rates provide diminishing returns and introduce significant overhead, which would make our method unsuited for rendering with low sample rates.



(a) Time against error over full image, for different K.

(b) Time against error over inset A, for different K.



(c) Time against error over inset B, for different K.

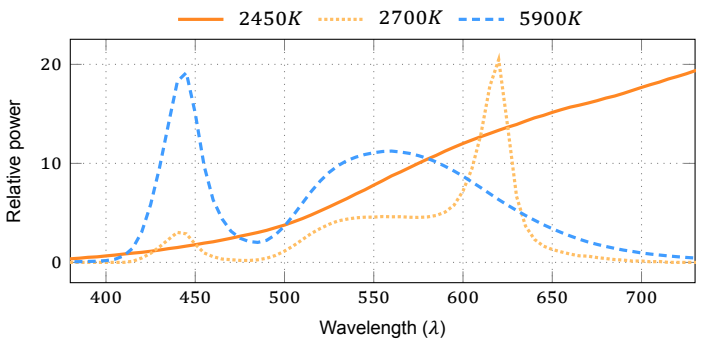
(d) Time against error over inset C, for different K.

Figure 5.8: **Layering parameter evaluation.** We compare the influence of different numbers of pre-estimate layers K. Based on results, we select $K = 3$ as a suitable number of layers. We note that, although layering has a diminished impact due to the increased defensive sampling, it garners larger improvements for low sample rates than an increased $\widetilde{N}$ would, as it impacts the initial overhead of our method less.

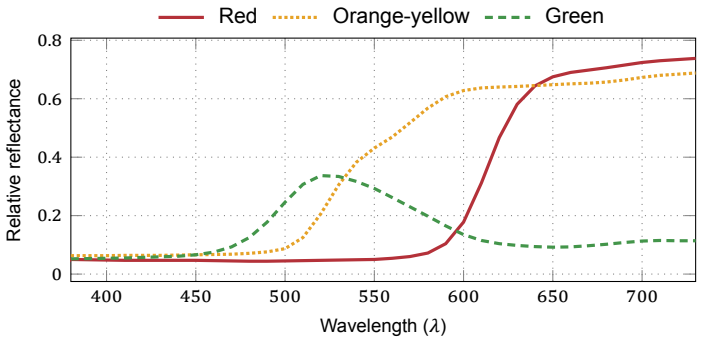
5.2. Comparative Results

We construct a set of test scenes to describe both the problems we covered in Chapter 3, as well as problems comparable methods are equipped to handle. As our method emphasises the handling of non-uniform spectra, we keep all scenes geometrically identical, but vary material and emitter properties using sets of measured non-uniform emission and reflectance spectra. We display these spectra and the accompanying constructed scenes in Figure 5.9.

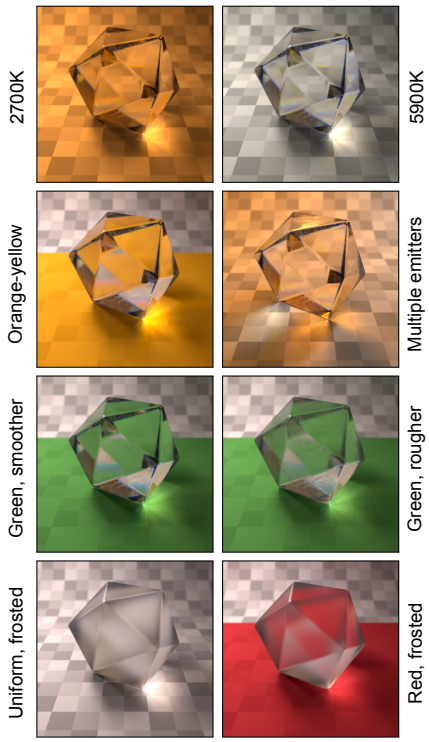
We first compare our method against standard path tracing in Figure 5.10, for three simple scenes. Afterwards, we separately compare our method against comparable methods in scenes tailored to each. For example, in Figure 5.11, we show convergence rates of our method and the emitter importance sampling technique briefly mentioned by Evans et al. [8], for single and multiple emitter scenarios, reflecting the issues we discussed in Chapter 3. Then, we compare our method with state-of-the-art techniques Hero Wavelength Sampling [38] and Spectral Gradient Sampling [20] in Figure 5.12 and Figure 5.13, respectively. We additionally provide timings for standard path tracing, hero wavelength sampling, and our method in Table 5.2. Finally, in Figure 5.14, we revisit the insets we used in Section 5.1. We reserve any discussion of obtained results for Chapter 6.



(a) Emission spectra are obtained from the Light Spectral Power Distribution Database (LSPDD) under a CC-Y-NC-ND 2.5 CA License [26].



(b) Reflectance spectra are obtained from MacBeth Color Checker data [6].



(c) All references are produced in Mitsuba [35] with an unbiased path tracer ($N \approx 200K$).

Figure 5.9: **Test scenes.** We use a set of test scenes that leverage different mixtures of the emitter and reflectance spectra displayed in Figures 5.9a-5.9b, as well as different levels of surface roughness, but are otherwise identical.

spp.	128	256	512	1024	2048	4096	8192
Path, time (s)	27.53	54.69	107.31	214.02	427.32	857.34	1699.23
Ours, time (s)	33.91	61.25	114.94	223.53	441.34	872.404	1730.78
Hero, time (s)	42.50	82.54	163.02	322.93	644.25	1294.40	2566.38
Ours, overhead (%)	+23.17	+12.00	+7.11	+4.44	+3.28	+1.76	+1.86
Hero, overhead (%)	+54.38	+51.11	+51.92	+50.89	+50.77	+50.98	+51.03

Table 5.2: **Timings.** We compare the runtimes of standard path tracing, our method, and hero wavelength sampling [38], for the `Green, rougher` scene, essentially showcasing the comparatively diminishing overhead of our method.

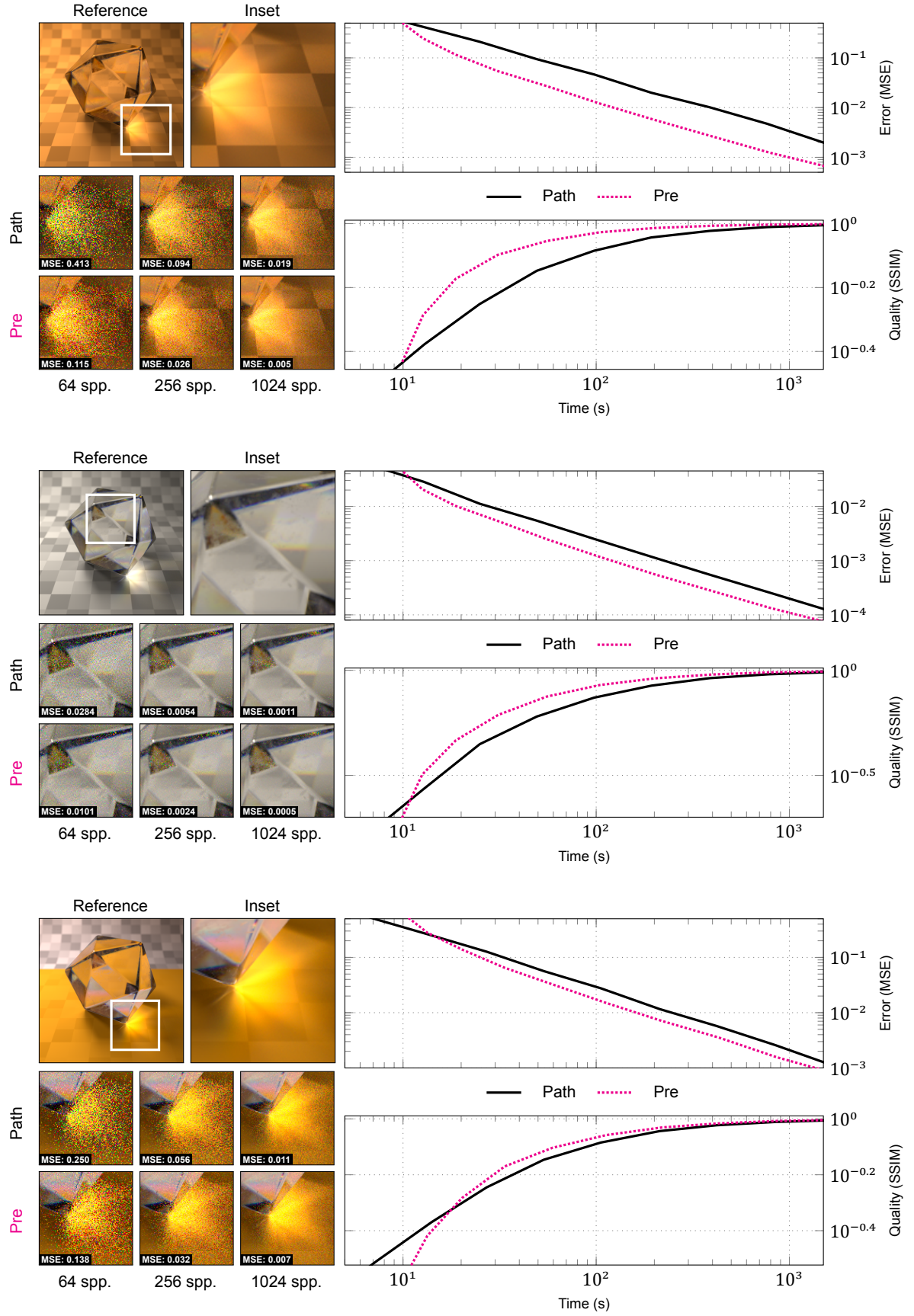


Figure 5.10: **Comparison against standard path tracing**. We compare the effectiveness of our method (*Pre*) against standard path tracing (*Path*), for three non-uniform spectra (*2700k*, *5900k*, *Orange-yellow*). We show visual comparisons of both methods, as well as convergence plots (all axes logarithmic). Our method shows significant improvements in performance over standard path tracing in all three comparisons, which involve non-uniform emission, reflectance, and transmittance.

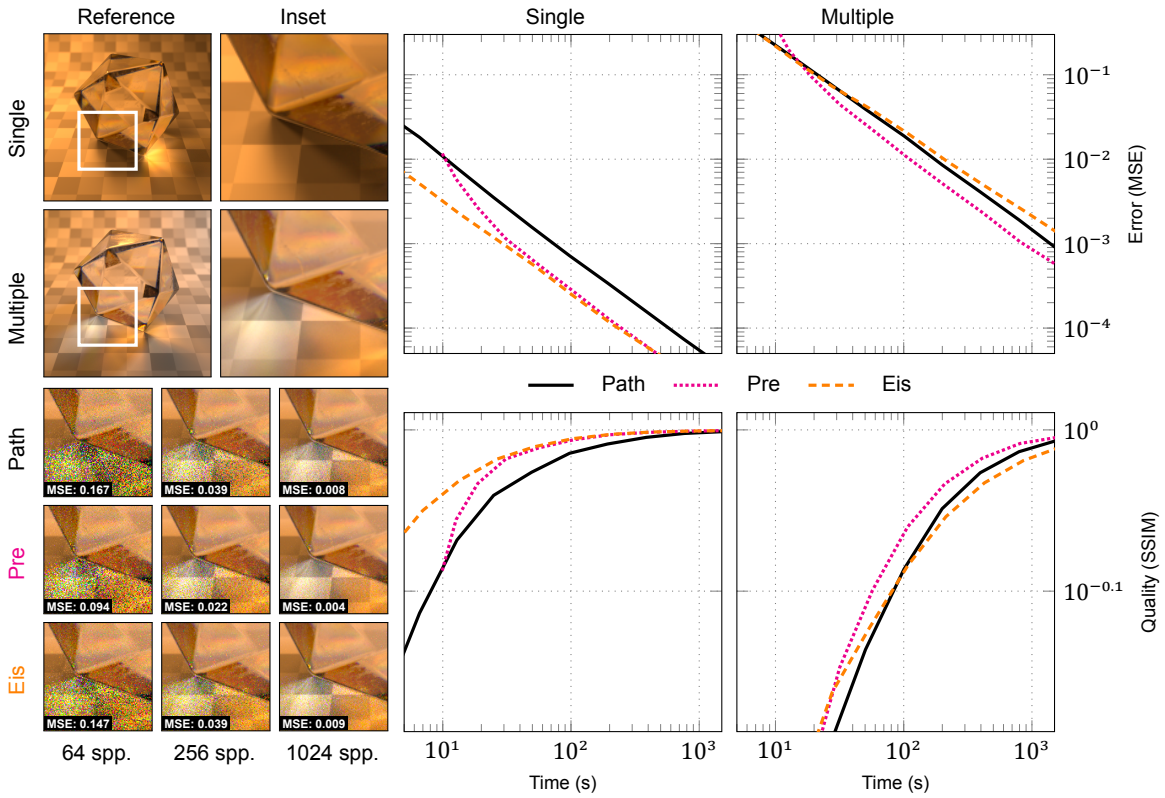


Figure 5.11: **Comparison against emitter importance sampling**. We compare the effectiveness of our method (*Pre*) against emitter importance sampling (*Eis*) [8] for single and multiple emitter scenarios. All axes are logarithmic. In the single emitter case, our method converges towards emitter importance sampling at $N \approx 1024$. Furthermore, as we predicted in Chapter 3, emitter importance sampling is not robust in a multiple emitters case, whereas our method performs well here.

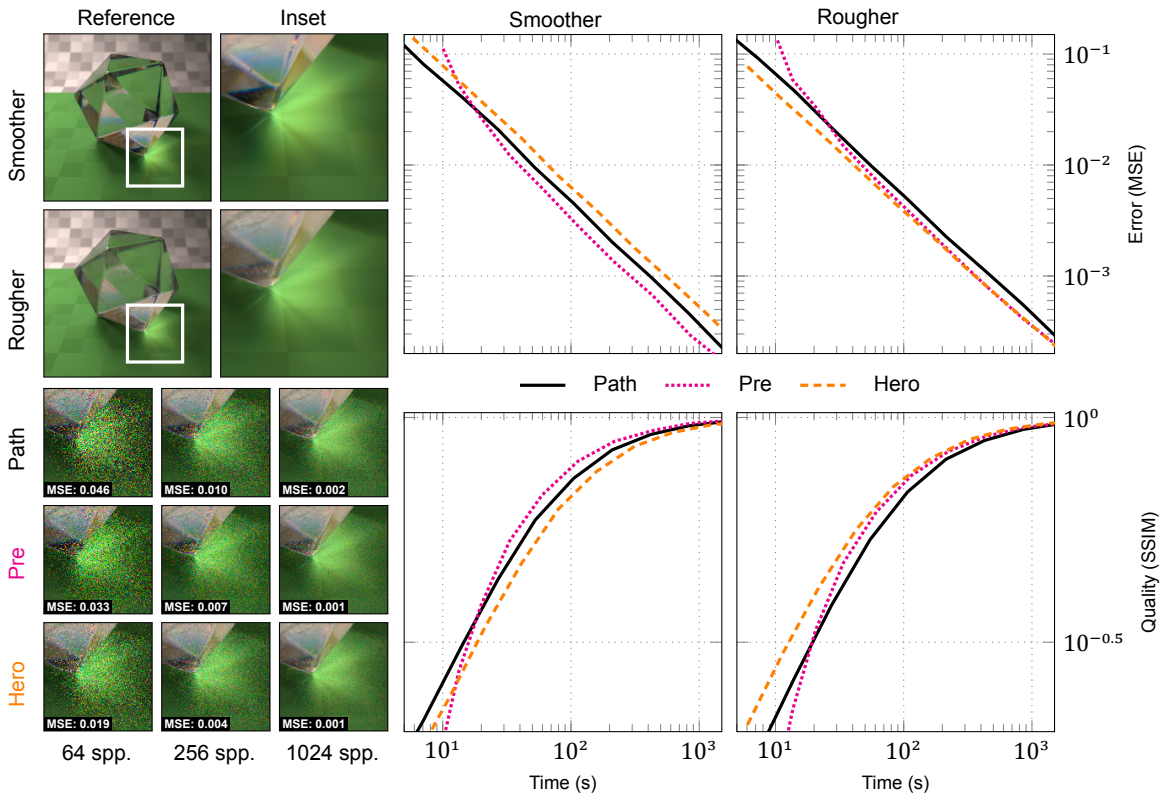


Figure 5.12: **Comparison against Hero Sampling**. We compare the effectiveness of our method (*Pre*) against hero wavelength sampling (*Hero*) [38], for smoother and rougher dielectric surfaces, the latter of which should prove optimal for Hero Sampling. All axes are logarithmic. As shown, our method converges towards a similar effectiveness as Hero Sampling, but additionally supports specular dielectrics.

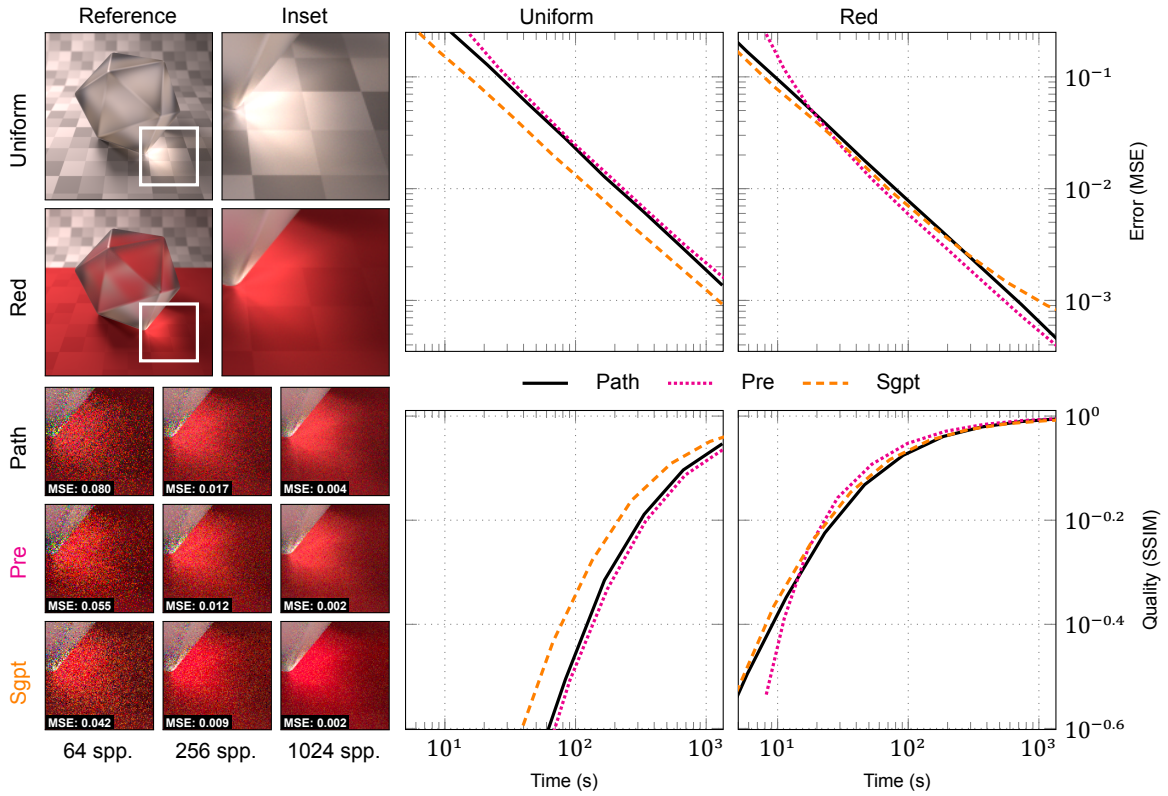


Figure 5.13: **Comparison against Spectral Gradient Sampling**. We compare the effectiveness of our method (*Pre*) against Spectral Gradient Sampling (*Sgpt*) [20] for uniform and non-uniform spectra. All axes are logarithmic. In the uniform case, our method becomes outright detrimental. In the non-uniform case, Spectral Gradient Sampling appears to diverge around $N \approx 2048$, failing to provide any advantage.

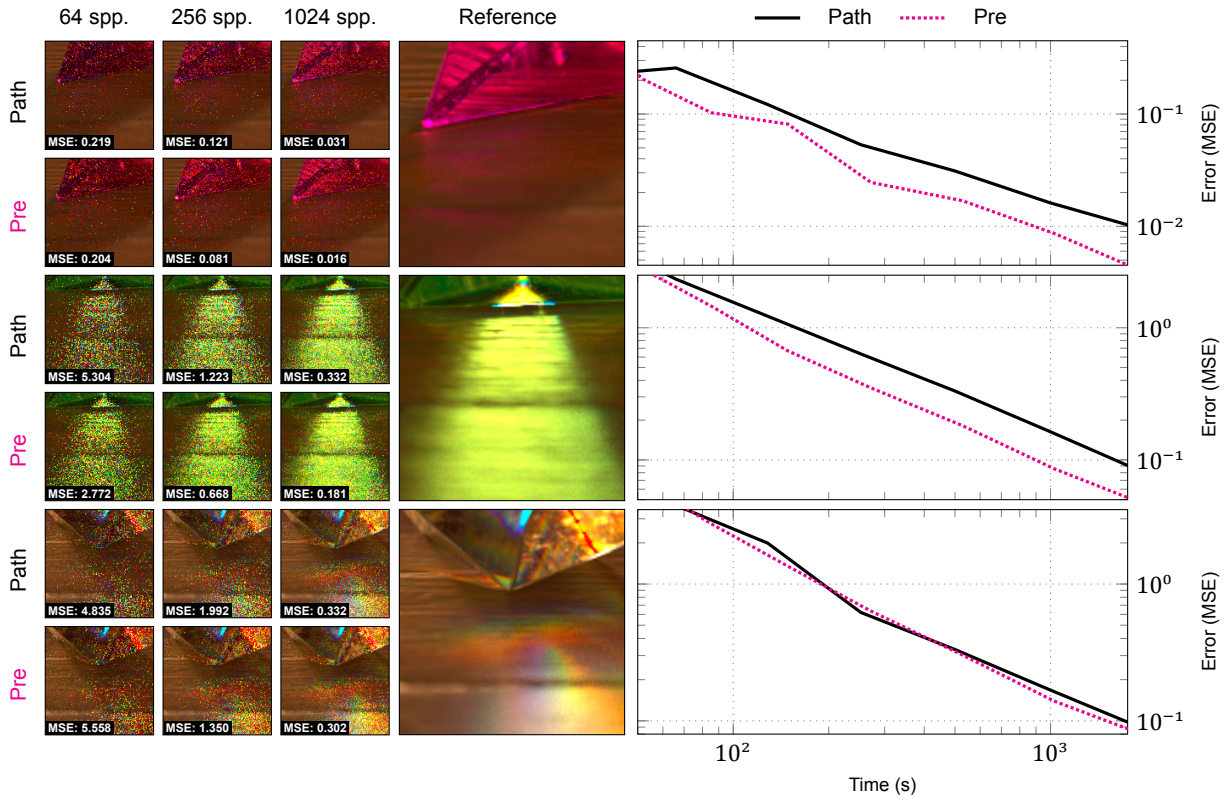
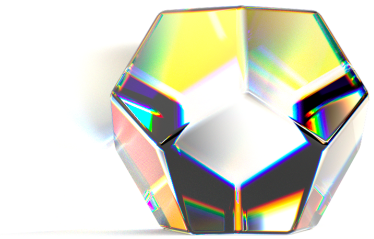


Figure 5.14: **Comparison against standard path tracing**. We compare the effectiveness of our method (*Pre*) against standard path tracing (*Path*) for the different insets we previously used in Section 5.1. All axes are logarithmic. Although our method shows significant improvements in *inset A* and *inset B*, it is (inconsistently) matched by standard path tracing in *inset C*, which is unsurprising given the difficult prismatic effects that occur there.



6 Discussion

We now discuss the results previously obtained in Chapter 5, and use these to evaluate to what extent our method handles the *wavelength sampling problem*, the basics of which we briefly repeat. Using the obtained results, we highlight several key advantages of our method in Section 6.1. Results also indicate, however, that our method has specific limitations, which we address in Section 6.2.

Recall how we noted that, if incoming radiance is non-uniform, we should preferably sample wavelengths from a distribution similar or highly proportional to said radiance. Finding a good distribution for this is, to aptly summarize Chapter 3, *hard*. Non-uniformness can occur due to multiple events: the presence of non-uniform emission, reflectance and transmittance spectra along a path as well as especially strongly wavelength-dependent phenomena which, unfortunately, make this issue hard to predict. We demonstrate different combinations of these events in Figure 5.14, where non-uniformly emitted light is transmitted through colored dielectrics and partially absorbed by a floor before it even reaches our camera. As shown, our method displays significant improvements over standard path tracing especially in insets A and B: for $N = 1024$, we obtain decreases in MSE of $0.031 \rightarrow 0.016$ and $0.332 \rightarrow 0.181$ respectively, effectively halving our error while incurring a minor time overhead of $252.50s \rightarrow 271.45s$. In inset C, which is notably less affected by non-uniform spectra but instead displays strong and highly local prismatic effects, our method provides only a minor improvement of $0.332 \rightarrow 0.302$. The largest gain for our method is obtained in the 2700K scene in Figure 5.10, where we vary only a single emission or reflectance spectrum. Here, our method visibly reduces image noise and, for $N = 1024$, provides a massive error decrease of $0.019 \rightarrow 0.005$. The scenarios demonstrated in Figure 5.10 are rather simple, which is why we discuss several more complicated scenarios in the following sections.

6.1. Advantages

Multiple Emitters In Figure 5.11, we illustrate a scene illuminated by three (explicitly) different non-uniform emission spectra, to reflect the problem we described in Figure 3.1. We compare our method against the emitter importance sampling technique first used by Evans et al. [8], where the latter fails entirely as importance sampling the wrong emitter evidently leads to an increase in variance. Fortunately, our method is more robust against this scenario, and manages to approximately halve the observed error. Of interest is the simple, single emitter scenario, where emitter importance sampling initially outperforms our method due to the added overhead, but both eventually converge at a similar rate from $N \approx 256$ onwards.

Reflectance Spectra To our knowledge, no technique currently exists that dynamically handles importance sampling of reflectance or transmission spectra present in a scene, so as to counteract the effects of non-uniform absorption. The effectiveness of our method for handling this is especially notable in the *Orange-yellow* scene in Figure 5.10, where light is partially absorbed by the colored floor, and where our method provides a considerable $0.011 \rightarrow 0.007$ error decrease for $N = 1024$.

Predictable Runtime Costs In Table 5.2, we provide timings for our method, which exhibits a significant time overhead due to preprocessing. This overhead is constant, and sampling the pre-estimate during rendering has a minor runtime impact comparable to performing a single extra texture sampling per pixel. This is evident from timings, as these runtime costs gradually become neglibile as N increases.

Unfortunately, as our method evidently exhibits a *startup period* for low sample rates, it is less effective than standard path tracing for $N < 128$ spp. Although we do not consider this to be a major issue as dispersive effects require thousands of samples to converge, we nonetheless address an optimization to reduce this ineffectiveness. Instead of using the established $\tilde{N} = 256$ sample rate for the pre-estimate, we can introduce a dependency on the actual sample rate N. In a simple example, we apply some multiplicative factor $p \leq 1$, and define a new *relative* sample rate as

$$\tilde{N} = \min(p \cdot N,\ \tilde{N}_{max}) \tag{6.1}$$

where $\tilde{N}_{max}$ is a limit beyond which we cut off this dependency, due to the diminishing returns from increased sample rates observed in Figure 5.7. We append this concept to our implementation, and provide comparative results for fixed and relative sample rates for a single scene in Figure 6.1.

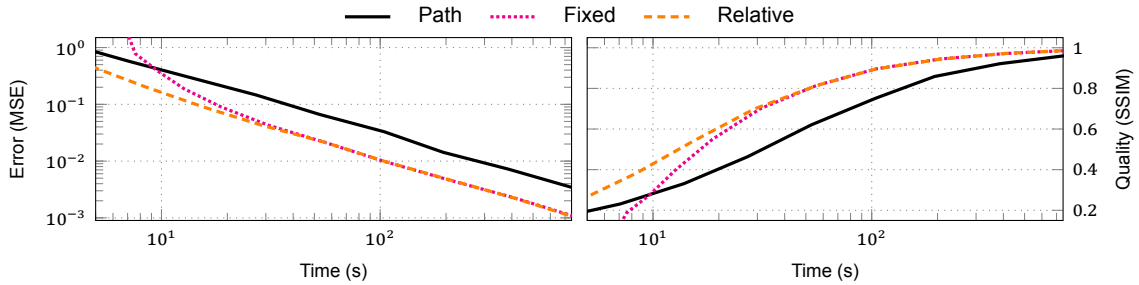


Figure 6.1: **Relative sample rate.** We demonstrate a relative pre-estimate sample rate ($p = 1$, $\tilde{N}_{max} = 256$). We compare MSE and SSIM metrics, in the `2700K` scene, using a standard path tracer (*Path*) as well as our method using fixed (*Fixed*) and relative (*Relative*) sample rates. MSE and Time axes are logarihmic. This small extension evidently allows our method to outperform path tracing even for low sample rates.

Specular Dielectrics As we demonstrate in Figure 5.12, our method demonstrates similar gains in both specular and non-specular (rough) scenarios, whereas hero wavelength sampling [38] only handles non-specular wavelength-dependent BSDFs. Hero sampling retains its effectiveness under the duress of non-uniform spectra, and as such outperforms our method for smaller sample rates ($N < 1024$), and performs equally well for higher sample rates. For clarification, we briefly revisit the timings provided in Table 5.2. Although hero wavelength provides a greater error decrease than our method for equal sample rates, its advantage diminishes gradually as our method incurs a comparatively constant overhead. If we apply the relative sample rate extension of Equation 6.1, this likely improves to our advantage.

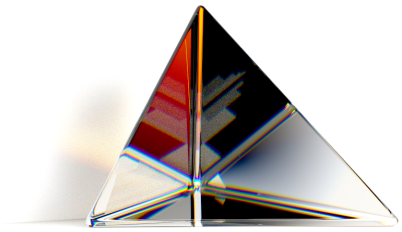
6.2. Limitations

Near-Uniform Spectra We compare our method to spectral gradient sampling [20] in Figure 5.13. Although this essentially matches and eventually outperforms spectral gradient sampling in a non-uniform scenario, the uniform scenario serves to highlight a minor flaw: pre-estimated spectral rendering is incapable of efficiently handling uniform spectral distributions. If observed radiance is uniform, the best distribution our method can provide is equal to random sampling. Unfortunately, as the pre-estimate itself contains some error and is unlikely to provide a perfectly uniform distribution, our method cannot be optimal in such a scenario. To counteract this limitation, we propose appending a minor optimization to our method. We define some threshold factor $t \geq 0$, and modify Equation 4.2 such that we apply this threshold to our pre-estimate as

$$p_j(\bar{x}, \lambda) = \begin{cases} r(\tilde{I})_j \cdot p(\bar{x} \mid \lambda) & \text{if } \sigma[r(\tilde{I})_j] > t, \\ p(\lambda) \cdot p(\bar{x} \mid \lambda) & \text{else.} \end{cases} \tag{6.2}$$

which effectively makes our method fall back to standard path tracing when a generated distribution is near-uniform. We did not implement this concept, and reserve it for future work.

Unpredictable Effectiveness There is a significant difference in performance gains between scenes, which is clearly visible even in Figure 5.10. In the `2700k` scene, our method converges significantly faster than in the `Orange-yellow` scene, even though both scenes are geometrically identical. We can attribute this difference to two factors. Either the pre-estimate is of insufficient quality in some scenarios, or the effectiveness of importance sampling is reduced depending on the non-uniformness of different spectral distributions. While the former is not likely in such a simple scene (it is almost entirely one distribution, after all), the latter is certainly a factor as the `Orange-yellow` reflectance spectrum is significantly less complicated than the `2700K` emitter spectrum.



7 Conclusion

We have developed *pre-estimated spectral rendering*, which is a simple and robust method that counteracts a multitude of wavelength sampling problems, which we identified as standing issues with traditional spectral light transport algorithms. While most spectral renderers sample wavelengths uniformly, we demonstrated that first generating and then using suitable spectral distributions can, given the presence of non-uniform spectra in a scene, improve convergence rates and decrease noise in a multitude of difficult scenarios. Although our method has limited application outside these scenarios, this happens to be an area in which current state-of-the-art methods, such as hero wavelength sampling [38] and spectral gradient sampling [20] provide little benefits, while our method is especially effective here.

In conclusion, our work provides a notable improvement to spectral rendering, that manages to largely negate a number of problems, and may in the future enable the use of realistic spectra for photorealistic renders. There are a number of directions we consider recommendable for further improvements, and we briefly cover these in Section 7.1.

7.1. Future work

Integration with orthogonal methods Although we developed our technique in the context of a standard spectral path tracer, the underlying concept of generating wavelength sampling distributions may be applicable to advanced path sampling techniques, such as bidirectional path tracing [14, 31] and metropolis light transport [33]. In addition, as our method is essentially orthogonal to both hero wavelength sampling [38] and spectral gradient sampling [20], exploring an integration of the pre-estimate with these methods can potentially improve convergence rates even further.

Improved reconstruction function Our method leverages a considerable number of samples in the pre-estimate, in order for proper distributions to be recoverable by the reconstruction function. We currently use a simple joint-bilateral filter [7, 21] for this task, but hope to see fast, alternative filtering algorithms for chromatic noise become available in the future. If an algorithm is developed that can recover a considerably higher quality image from undersampled wavelength-dependent phenomena, perhaps our method merits a revisit.

Robustness against near-uniform spectra As our method inefficiently handles near-uniform spectra, it does not provide a robust solution for every possible scenario. Although we mention falling back to standard path tracing for such scenarios in Chapter 6, falling back to alternative methods such as spectral gradient sampling [20] may provide additional benefits. We explore one potential option for this in Appendix A.

A

Decomposed Rendering

As we can gather from Chapter 5, spectral gradient sampling [20] provides significant improvements to convergence rates, albeit with a caveat: the *gradient* of encountered spectral distributions must be relatively minor. This is in direct contrast to our method, which garners improvements especially in the presence of highly non-uniform spectra. During development of this thesis, we briefly explored a flawed coupling between these methods to leverage both intermittently against their *preferred* scenarios.

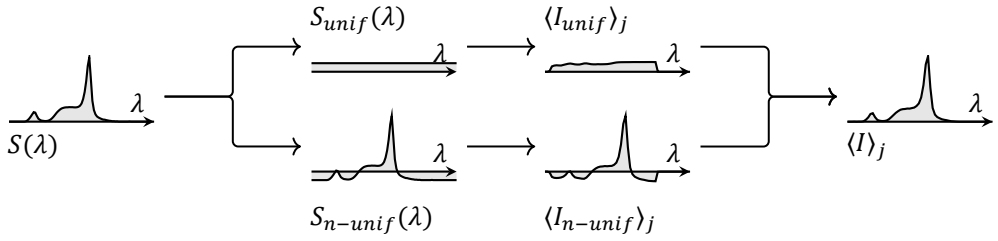
This coupling pits on the realization that we can decompose any non-uniform emitter SPDs in a scene into a combination of a uniform distribution and a non-uniform, *spiky* remainder. Formally, we simply decompose an SPD $S(\lambda)$ as:

$$S(\lambda) = S_{unif}(\lambda) + S_{n-unif}(\lambda), \tag{A.1}$$

while we then similarly form a viable estimator for I_j as:

$$\hat{I}_j = \hat{I}_{unif,j} + \hat{I}_{n-unif,j}, \tag{A.2}$$

where I_{unif} and I_{n-unif} are then estimated in scenes where all emitters *emit* using only their uniform or non-uniform decompositions, respectively. We illustrate the basic concept of what we call *decomposed rendering* in Figure A.1. We will first show that this does form a valid estimator for I_j in Appendix A.1, assuming for now that we can successfully perform the decomposition of Equation A.1 to do so. We then demonstrate a number of possible decompositions for emitter SPDs in Appendix A.2. Finally, in Appendix A.3, we analyse why this concept is flawed and essentially impractical, as it unfortunately fails to provide a solution to our problem.



(a) Decomposition of emitters.

(b) Re-composition of estimated radiances.

Figure A.1: **Decomposed rendering.** All representative SPDs $S(\lambda)$ of scene emitters are decomposed into uniform and non-uniform parts. Afterwards, two separate estimators are used that accomodate only their respective SPD decompositions, and their resulting observed radiances are recomposited to form a valid estimator for I_j.

A.1. Composition of Estimators

As we showed in Equation A.2, our estimator combines estimates of radiance values I_{unif} and I_{n-unif}. If we assume (or simplify) that, in our simplified model of light transport, the input of energy and effect

thereof are linearly related, we can indeed recompose radiance I_j for a pixel j as:

$$I_j = (I_{unif})_j + (I_{n-unif})_j. \tag{A.3}$$

We can then apply Equation 2.14 to show that the estimator of Equation A.2 indeed forms a valid estimator for I_j:

$$\begin{aligned} E[\hat{I}_{unif,j}] + E[\hat{I}_{n-unif,j}] &= (I_{unif})_j + (I_{n-unif})_j \\ &= I_j \\ &= E[\hat{I}_j]. \end{aligned} \tag{A.4}$$

Logically, we then leverage spectral gradient sampling for estimating I_{unif} and pre-estimated spectral patch tracing for estimating I_{n-unif}, as these are both better suited to rendering in these respective situations.

A.2. Decomposition of Emitters

This technique pivots on our capability of performing the decomposition in Equation A.1. Although we can attempt to decompose peaks from spectra, this hardly result in a perfectly uniform distribution being available for spectral gradient sampling. Instead, we propose to decompose a spectral distribution as:

$$S_{unif}(\lambda) = a,\ S_{n-unif}(\lambda) = S(\lambda) - S_{unif}(\lambda) \tag{A.5}$$

where $S_{unif}(\lambda)$ is clearly just a uniform distribution with some applied scaling factor a. This may result in a partially negative distribution $S_{n-unif}(\lambda)$, but should not have any impact on the correctness of our renderer as all algorithms given so far can be well-defined for negative values by treating them as positive values in all operations but multiplications. We consider a number of viable choices for determining a. The simplest of these choices would be to place a at either the mean or median power value of $S(\lambda)$, assuming $S(\lambda)$ is only defined over Λ:

$$a = \frac{1}{|\Lambda|} \int_{\Lambda} S(\lambda)\ d\lambda \tag{A.6}$$

or:

$$a = min\ S(\lambda) + \frac{max\ S(\lambda) - min\ S(\lambda)}{2} \tag{A.7}$$

However, neither of these choices for a are optimal in any way. Instead, we can choose to minimize the difference in emitted energy between $S_{unif}(\lambda)$ and $S_{n-unif}(\lambda)$, effectively equalizing the amount of emitted energy dealt with by both estimators:

$$\begin{aligned} a &= \min_x \left| \int_{\Lambda} x\ d\lambda - \int_{\Lambda} (S(\lambda) - x)\ d\lambda \right| \\ &= \min \left| \int_{\Lambda} S_{unif}(\lambda)\ d\lambda - \int_{\Lambda} S_{n-unif}(\lambda)\ d\lambda \right| \end{aligned} \tag{A.8}$$

We demonstrate what these various decompositions would look like for two viable SPDs in Figure A.2.

A.3. Potential Error Increases

Unfortunately, although the composition of Equation A.2 forms a valid estimator, we show that it is highly inefficient. Following our definition for cumulative variance in Chapter 2, we know that

$$V[\hat{I}_{comp,j}] = V[\hat{I}_{unif,j}] + V[\hat{I}_{n-unif,j}] + COV[\hat{I}_{unif,J}, \hat{I}_{n-unif,j}], \tag{A.9}$$

which implies that error from both estimators will accumulate, unless either both estimators are correlated such that negative covariance occurs, or both estimators significantly decrease the error in their respective scenarios. However, even this definition is incorrect and too optimal, as we have yet not accounted for the estimators' changed *sample rates*. Recall that the variance of a Monte Carlo estimator $\hat{I}$ is defined as

$$V[\hat{I}] = \frac{1}{N} V\left[\frac{f(x)}{p(x)}\right]. \tag{2.15}$$

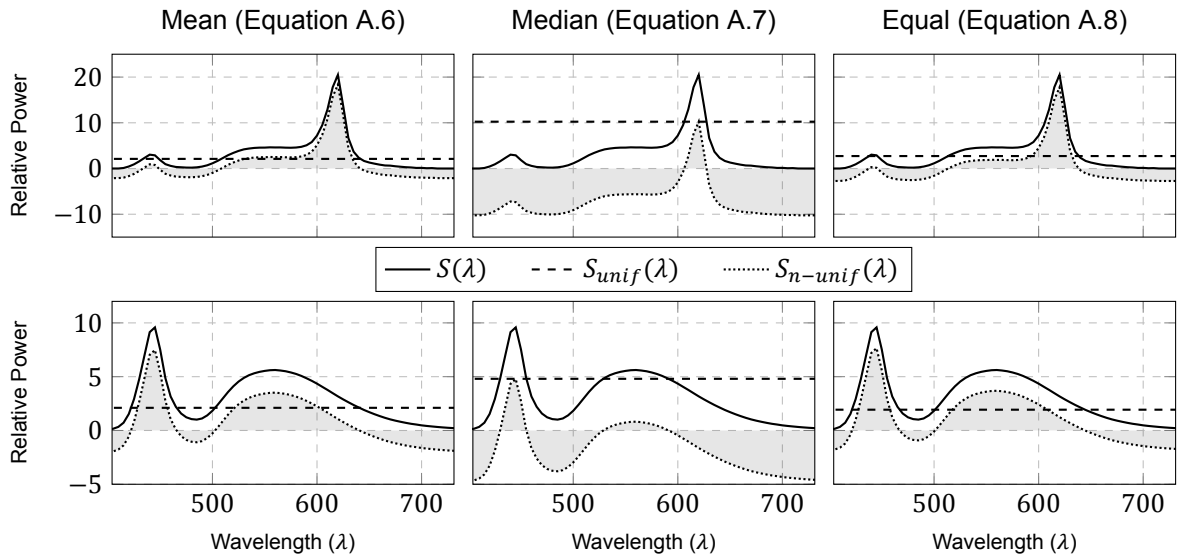


Figure A.2: Three decompositions of $S(\lambda)$ are shown for two different SPDs. The total area covered by the possibly negative $S_{n-unif}(\lambda)$ is marked in gray for clarity.

If we wish to compare the variance of our recomposed estimator to a standard Monte Carlo estimator, we must assume that the total number of samples available to the decomposed estimators is still N, and must be divided between them. If both estimators were to each share half of the available samples, i.e. $N' = 0.5N$, variance per decomposed estimator becomes

$$V[\hat{I}] = \frac{2}{N} V\left[\frac{f(x)}{p(x)}\right]. \tag{A.10}$$

A potential alternative that may successfully leverage a partial decomposition of emitters to introduce covariance, would be the application of *control variates* [19]. Unfortunately, exploring the concept fully was outside the scope of this thesis.

Bibliography

[1] LuxCoreRender - Open Source Physically Based Renderer. https://luxcorerender.org/, 2019. Last accessed: 2019-04-26.

[2] Autodesk. Arnold Renderer. https://www.arnoldrenderer.com/, 2016. Last accessed: 2019-07-30.

[3] Pravin Bhat, Brian Curless, Michael Cohen, and C. Lawrence Zitnick. Fourier analysis of the 2d screened poisson equation for gradient domain problems. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, pages 114–128, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-88688-4.

[4] Carlos F. Borges. Trichromatic approximation for computer graphics illumination models. *SIGGRAPH Comput. Graph.*, 25(4):101–104, July 1991. ISSN 0097-8930. doi: 10.1145/127719.122729. URL http://doi.acm.org/10.1145/127719.122729.

[5] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-Local Means Denoising. *Image Processing On Line*, 1:208–212, 2011. doi: 10.5201/ipol.2011.bcm_nlm.

[6] The BabelColor Company. BabelColor - The Colorchecker Pages. http://www.babelcolor.com/colorchecker.htm, 2019. Last accessed: 2019-08-29.

[7] Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph.*, 23(3):673–678, August 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015778. URL http://doi.acm.org/10.1145/1015706.1015778.

[8] Glenn F. Evans and Micheal D. McCool. Stratified wavelength clusters for efficient spectral monte carlo rendering. In *Proceedings of the 1999 Conference on Graphics Interface '99*, pages 42–49, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-632-7. URL http://dl.acm.org/citation.cfm?id=351631.351648.

[9] Brian Huebert. FreePBR - Free PBR Materials. https://freepbr.com/, 2019. Last accessed: 2019-07-31.

[10] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. *SIGGRAPH Comput. Graph.*, 20(4):133–142, August 1986. ISSN 0097-8930. doi: 10.1145/15886.15901. URL http://doi.acm.org/10.1145/15886.15901.

[11] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, August 1986. ISSN 0097-8930. doi: 10.1145/15886.15902. URL http://doi.acm.org/10.1145/15886.15902.

[12] Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. Gradient-domain path tracing. *ACM Trans. Graph.*, 34(4), 2015.

[13] Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Trans. Graph.*, 26(3), July 2007. ISSN 0730-0301. doi: 10.1145/1276377.1276497. URL http://doi.acm.org/10.1145/1276377.1276497.

[14] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal, December 1993.

[15] Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. Gradient-domain metropolis light transport. *ACM Trans. Graph.*, 32(4):95:1–95:12, July 2013. ISSN 0730-0301. doi: 10.1145/2461912.2461943. URL http://doi.acm.org/10.1145/2461912.2461943.

[16] M.W. Levine. *Fundamentals of Sensation and Perception*. Oxford University Press, 01 2000.

[17] Marco Manzi, Markus Kettunen, Frédo Durand, Matthias Zwicker, and Jaakko Lehtinen. Temporal gradient-domain path tracing. *ACM Trans. Graph.*, 35(6):246:1–246:9, November 2016. ISSN 0730-0301. doi: 10.1145/2980179.2980256. URL http://doi.acm.org/10.1145/2980179.2980256.

[18] NVIDIA. Mental ray. http://www.nvidia-arc.com/products/nvidia-mental-ray.html, 2017. Last accessed: 2019-04-26.

[19] Art Owen. Monte carlo theory, methods and examples. Last accessed 30-09-2019, 2013.

[20] Victor Petitjean, Pablo Bauszat, and Elmar Eisemann. Spectral gradient sampling for path tracing. In *Computer Graphics Forum (Proceedings of EGSR)*, 2018. URL http://graphics.tudelft.nl/Publications-new/2018/PBE18.

[21] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM Trans. Graph.*, 23(3):664–672, August 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015777. URL http://doi.acm.org/10.1145/1015706.1015777.

[22] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2016. ISBN 0128006455, 9780128006450.

[23] Pixar. Pixar's RenderMan. https://renderman.pixar.com/, 2016. Last accessed: 2019-07-30.

[24] Mikhail N. Polyanskiy. Refractive index database. https://refractiveindex.info, 2008. Last accessed: 2019-05-13.

[25] Michal Radziszewski, Krzysztof Boryczko, and Witold Alda. An improved technique for full spectral rendering. *Journal of WSCG*, 17, 01 2009.

[26] Johanne Roby and Martin Aubé. LSPDD | Light Spectral Power Distribution Database. http://galileo.graphycs.cegepsherbrooke.qc.ca/app/en/home, 2019. Last accessed: 2019-07-31.

[27] Sellmeier. Zur erklärung der abnormen farbenfolge im spectrum einiger substanzen. *Annalen der Physik*, 219(6):272–282, 1871. doi: 10.1002/andp.18712190612. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.18712190612.

[28] T Smith and J Guild. The c.i.e. colorimetric standards and their use. *Transactions of the Optical Society*, 33(3):73–134, jan 1931. doi: 10.1088/1475-4878/33/3/301. URL https://doi.org/10.1088%2F1475-4878%2F33%2F3%2F301.

[29] Walt Disney Animation Studios. Disney's Hyperion Renderer. https://www.disneyanimation.com/technology/innovations/hyperion, 2015. Last accessed: 2019-07-30.

[30] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162.

[31] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In Georgios Sakas, Stefan Muller, and Peter Shirley, editors, *Photorealistic Rendering Techniques*, pages 145–167, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-642-87825-1.

[32] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 419–428, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218498. URL http://doi.acm.org/10.1145/218380.218498.

[33] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 65–76, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. doi: 10.1145/258734.258775. URL https://doi.org/10.1145/258734.258775.

[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13:600–612, April 2004. doi: 10.1109/TIP.2003.819861.

[35] Jakob Wenzel. Mitsuba - Physically Based Renderer. https://www.mitsuba-renderer.org, 2010. Last accessed: 2019-04-26.

[36] Jakob Wenzel. Mitsuba - API Reference. https://www.mitsuba-renderer.org/api/, 2010. Last accessed: 2019-06-23.

[37] Jakob Wenzel. Mitsuba - Documentation. https://www.mitsuba-renderer.org/docs.html, 2010. Last accessed: 2019-06-23.

[38] A. Wilkie, S. Nawaz, M. Droske, A. Weidlich, and J. Hanika. Hero wavelength spectral sampling. In *Proceedings of the 25th Eurographics Symposium on Rendering*, EGSR '14, pages 123–131, Aire-la-Ville, Switzerland, Switzerland, 2014. Eurographics Association. doi: 10.1111/cgf.12419. URL http://dx.doi.org/10.1111/cgf.12419.

[39] Greg Zaal. HDRI Haven. https://hdrihaven.com/p/about-contact.php, 2019. Last accessed: 2019-07-31.