Advanced Techniques in Gradient-Domain Rendering

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Marco Manzi

von Triengen

Leiter der Arbeit:

Prof. Dr. M. Zwicker
Universität Bern

Prof. Dr. F. Durand
Massachusetts Institute of Technology

Advanced Techniques in Gradient-Domain Rendering

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Marco Manzi

von Triengen

Leiter der Arbeit:

Prof. Dr. M. Zwicker
Universität Bern

Prof. Dr. F. Durand
Massachusetts Institute of Technology

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Bern, 31.10.2016 **Der Dekan:**
Prof. Dr. G. Colangelo

Abstract

Rendering realistic images requires solving the notoriously hard physically-based light transport problem. Almost all of the state-of-the-art physically-based rendering methods use Monte-Carlo sampling of the light paths contributing to the image. These methods suffer from variance until convergence. Depending on the scene, an impracticable amount of time might be required to get a clean image. A recently developed method called gradient-domain Metropolis light transport mitigates this problem. It first samples the image-space finite differences of paths alongside the paths themselves and then reconstructs a clean image from the sampled data by applying a screened Poisson reconstruction. The method exploits two properties: first, the gradients of natural images are usually much sparser than the image itself, thus sampling efforts can be concentrated in fewer regions of the path space. Second, sampling finite differences allows using correlated sampling in rendering, which can strongly reduce noise in the finite differences. Both properties combined lead to dramatical speed-ups compared to classical (Markov-chain) Monte-Carlo rendering methods.

This dissertation builds up on the aforementioned gradient-domain Metropolis light transport and proposes a number of improvements and generalizations. We improve the sampling by replacing finite differences by arbitrary differences and by combining different sampling strategies in an unbiased way. We also generalize the method to non-MLT rendering methods like bidirectional path tracing. Further, we develop an algorithm that regularizes the screened Poisson reconstruction by using auxiliary scene information in order to increase image quality. This leads to the first method that combines gradient-domain rendering with classical image-space denoising. And finally, we incorporate temporal finite differences in gradient-domain rendering in order to create stable animations, thus making gradient-domain rendering an even more appealing option for production rendering.

Acknowledgements

First and foremost, I want to offer my sincere thanks to my PhD supervisor Matthias Zwicker for sparking my interest in computer graphics and being a source of inspiration and motivation over the last five years. His constant guidance and confidence in me pushed me beyond what I thought would be possible when I started my PhD. This thesis would not have been possible without him. I also want to thank the co-referee Frédo Durand for agreeing to read and review this thesis and for his collaboration on the gradient-domain rendering publications.

Further, I want to thank all people of the computer graphics group in Bern for making my working environment in the last five years so pleasant. I especially thank Fabrice Rousselle for sharing his knowledge and experience with me during the first year of my PhD and ultimately paving the road for my research. I also thank my office mate Daniel Donatsch for motivating me to do some sports and organizing many the social events. Further, I thank Claude Knaus for showing me how to work efficiently, Daljit Singh Dhillon for teaching me cricket, Siavash Bigdeli for providing entertainment to the lab, Tiziano Portenier for the fruitful discussions on rendering and Shihao Wu for showing me together with his wife Magenta what real Chinese fondue tastes like. Special thanks goes to Peter Bertholet whose deep mathematical expertise helped me more than once and with whom I visited some great concerts. Lastly, I want to thank Dragana Esser for all the cake and being the heart and soul of the CGG group.

I also want to thank the team at the Aalto University in Helsinki for their fruitful collaboration that made most of the work presented here possible. Especially, I thank Jaakko Lehtinen for all the discussions and collaborative guidance with Matthias Zwicker on the the gradient-domain rendering projects and Markus Kettunen for the countless hours of discussion and his contribution to the work that is presented here. Further, I also thank the team at the R&D rendering group at Weta Digital for making my internship in New Zealand so inspiring. Special thanks go to Luca Fascione, Mark Droske and Jirka Vorba.

I also thank the people with whom I studied computer science and whose influence shaped my interest in computer graphics: Patrik Rauber, Gian Calgeer and Gregor Budweiser. Further, I thank the students that I had the pleasure to supervise, namely Michèle Wyss, Niclas Scheuing, Urs Gerber, Heinrich Reich and Marcel Zingg. Special thanks go to Delio Vicini whose work led to one of the publications presented in this paper.

This work was financed by the Swiss National Science Foundation, projects 143886 and 163045. Most of the methods presented in this thesis were implemented on top of the open source renderer Mitsuba by Wenzel Jakob.

My deepest gratitude goes to my wife Anita for her love, motivation and support. Without her, the last years wouldn't have been half as good and this thesis not half as satisfying. Lastly, I thank my parents Monique and Girolamo for their love and support through all these years. Without their encouragement this thesis would not have been possible. To them I dedicate this thesis.

Contents

1 Introduction **5**

1.1 Motivation . . . 5

1.2 Problem Statement . . . 6

1.3 Gradient-Domain Rendering . . . 8

1.4 Overview . . . 9

1.5 Contributions . . . 9

2 Light Transport **11**

2.1 Radiometry . . . 12

2.1.1 Spectral Power Distribution . . . 12

2.1.2 Radiant flux . . . 13

2.1.3 Irradiance . . . 13

2.1.4 Radiance . . . 13

2.2 Light Scattering . . . 14

2.2.1 Surface Scattering . . . 14

2.2.2 BSDF Types . . . 15

2.3 Light Propagation . . . 17

2.3.1 The Reflection Equation . . . 17

2.3.2 The Rendering Equation . . . 17

2.3.3 Transport Direction . . . 18

2.3.4 The Measurement Equation . . . 18

2.3.5 Surface Form of the Rendering Equation . . . 18

2.3.6 Path Space Formulation of Light Transport . . . 20

2.4 Path Expressions . . . 23

3 Monte-Carlo Integration **25**

3.1 Mathematical Foundations . . . 25

3.1.1 Random Variables . . . 25

3.1.2 Probability Calculus . . . 26

3.2 Introduction to Monte-Carlo Integration . . . 29

3.3 Variance Reduction Techniques . . . 30

3.3.1 Improved Coverage of the Integration Domain . . . 31

3.3.2 Adaptive Sampling . . . 32
3.3.3 Importance Sampling . . . 33
3.3.4 Multiple Importance Sampling . . . 33
3.3.5 Control Variates . . . 34
3.3.6 Correlated Sampling . . . 35
3.4 Metropolis Sampling . . . 37
3.4.1 Motivation . . . 37
3.4.2 Markov-Chains . . . 37
3.4.3 Metropolis-Hastings Algorithm . . . 37
3.4.4 Mutation Strategies . . . 38

4 Rendering Algorithms 41
4.1 Formulation of the Problem . . . 41
4.2 Path Tracing . . . 42
4.2.1 Reconstruction Filter . . . 43
4.2.2 Sampling Techniques . . . 44
4.2.3 Russian Roulette and Path Splitting . . . 45
4.2.4 Limitations . . . 46
4.3 Bidirectional Path Tracing . . . 46
4.3.1 MIS over Connection Strategies . . . 47
4.3.2 Direct Light Paths . . . 48
4.3.3 Limitations . . . 49
4.4 Metropolis Light Transport . . . 49
4.4.1 Start-up Bias . . . 51
4.4.2 Mutation Strategies . . . 52
4.4.3 Limitations . . . 59
4.5 Adaptive Sampling and Reconstruction . . . 60
4.5.1 A Priori Methods . . . 61
4.5.2 A Posteriori Methods . . . 61

5 Gradient-Domain Rendering 65
5.1 Motivation . . . 65
5.2 Basics . . . 66
5.2.1 Sampling Gradients . . . 67
5.2.2 Shift Mappings . . . 68
5.2.3 Reconstruction . . . 73
5.3 Theoretical Analysis . . . 75
5.3.1 Error analysis of Gradient Estimation . . . 75
5.3.2 Analysis of Screened Poisson Reconstruction . . . 79
5.4 Gradient-Domain Path Tracing . . . 81
5.4.1 MIS for Gradients . . . 82

5.4.2 Half-Vector Shift 83
5.4.3 Results 85
5.5 Gradient-Domain Metropolis Light Transport 87
5.5.1 Gradient Sampling with MLT 87
5.5.2 Manifold Exploration Shift 88
5.5.3 Results 89

6 Improved Sampling for G-MLT 91

7 Gradient-Domain Bidirectional Path Tracing 105

8 Regularized Reconstruction for Gradient-Domain Rendering 117

9 Temporal Gradient-Domain Path Tracing 129

10 Conclusions 139
10.1 Future Work 140

Chapter 1

Introduction

In the last couple of decades *computer graphics* (CG) has become more and more important in our day to day life up to a point that it is omnipresent. Nowadays, we encounter CG extensively in movies, video games, commercials, printed media, simulators and data visualizations, just to name a few. The uses of CG are sometimes very obvious, e.g. in form of fantastic special effects in science fiction movies and sometimes not so obvious like in photo-realistic furniture arrangements in commercial catalogues.

Before CG, special effects in media were more difficult to realize. For instance in a movie, every asset had to be physically present in some form, be it as a model, a drawing, a cardboard, a costume or an animatronic robot. Hence what was depictable was limited by what could be created physically with the available resources. With CG, the physical restrictions for creating imagery weakened severely. Suddenly, it was enough to have a virtual description of a scene in a computer, algorithms to convert them into images, and the computational power to transform these descriptions into images. While in theory every imaginable image became possible to create, practical limitations of the computation cost have always put limits to what was feasible with CG.

1.1 Motivation

In this thesis we will focus on the problems related to computing an image from a scene description. More precisely, given a scene description in some format we want to compute how an image would look like if captured from a virtual camera placed inside of this virtual scene. This process is called *rendering*. The specific goal on which we will focus in this thesis is to render images that are indistinguishable from real photography. We will concentrate on methods that simulate the physical processes that lead to photography in the real world.

Creating photo realistic images is difficult because in our tangible world they are the result of a very complex process: a camera creates an image by measuring the amount of light arriving at sensors inside of it. The amount of light arriving at these sensors depends on the light that arrives to the camera from all directions. This requires knowledge of how much light is reflected and emitted towards the camera from every point in the scene. However, the light reflected from any point in the scene towards the camera depends on all light that arrives at this point. In other words we need to know how much light is transported from any point in the scene to any other point in the scene. Therefore we call this the *light transport* problem.

Algorithms for approximating the light transport with arbitrary precision have been around since the 1980ies [50]. However, they have not been used widely because of their computational costs. Therefore until recently, rendering was always based as much as possible on cheaper *phenomenological* methods. Examples of such methods are non-physically based models for the surface appearance of objects [65] and methods for creating plausible soft shadows based on heuristics [26]. In general, phenomenological methods in rendering are based on models that can describe what something looks like without necessarily describing why it looks like that. The descriptive power of such models is often limited to a small range of visual phenomena. This led to very bloated processes for generating images since different visual phenomena had to be computed separately with different methods. These pipelines were complicated to handle since they required a lot of manual tweaking for every single asset that had to be rendered. This became a large cost factor for companies since artists needed to spend a lot of time to get this tweaking right.

This is in contrast to the more general light transport algorithms that produce photo-realistic images per definition and that require significantly less tweaking. With the decreasing computational cost due to faster and cheaper hardware, light transport algorithms thus became more attractive. As a consequence, around 2010, a dramatic shift in the movie-industry [55] occurred away from agglomerates of specialized phenomenological methods to physically-based light transport algorithms.

With physically-based light transport algorithms existing since decades and computation now being cheap enough to make these algorithms affordable, it might seem futile to invest more research efforts in light transport algorithms. However, since people tend to always push the limits of what is feasible, more computation power will not necessarily mean shorter rendering times. It is estimated that each year the average computational effort to render a blockbuster movie roughly doubles [25]. The reason for this is that the rendering industry is very competitive and therefore companies try to surpass each other with the level of realism they can achieve. As a consequence the scene complexity and the required precision of the light transport simulation increased dramatically over the last few years and will continue to do so. Since the increase in computation power seems to slowly approach a physical limit, companies in the rendering industry cannot rely exclusively on upgrading their hardware anymore. Hence, to further increase realism, more efficient light transport algorithms are required.

1.2 Problem Statement

Current light transport algorithms compute images by measuring the light arriving at each pixel represented by a virtual light sensor floating in the virtual scene. As it turns out, computing the quantity of light arriving at each sensor requires solving an equilibrium that tells how much light arrives and how much light leaves at *every* point in a scene simultaneously in a consistent way. Solving this problem is particularly hard since the light arriving at a certain point depends on the light arriving through any possible *path* from any light source to this point (Figure 1.1). Many methods have been developed in the last decades, and the most effective ones were based on *Monte-*

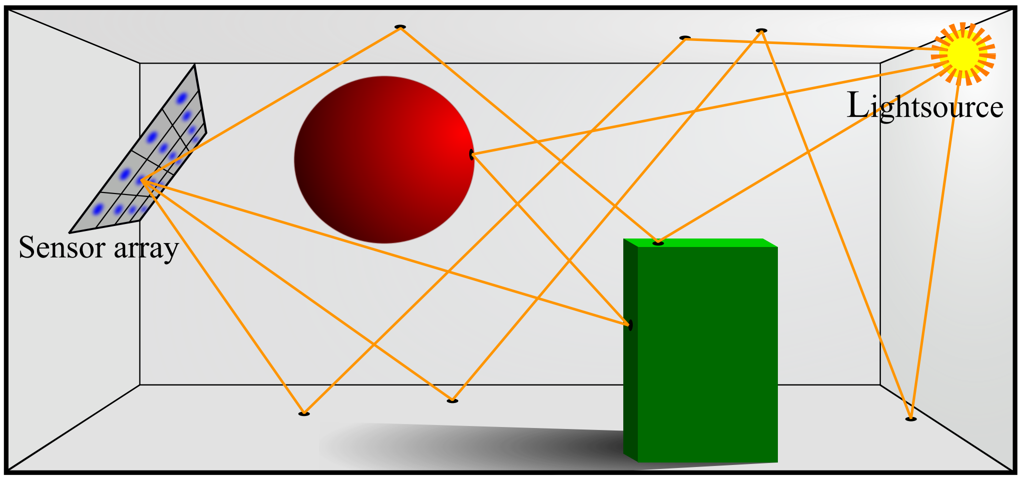


Figure 1.1: The image is represented as a virtual sensor array where each sensor roughly corresponds to a pixel. The color of each pixel depends on light travelling along *all* possible paths through the scene from the light source to the sensor. The number of possible paths is infinite since light can bounce off at arbitrary locations and surfaces an arbitrary number of times before reaching the sensor.

Carlo integration that sample these paths in a brute force manner. In a nutshell, these methods stochastically select paths that connect one of the light sources with the sensor, compute how much light they transport to the sensor, and compute the sensors response as a weighted sum of these path contributions. An image is then generated by computing the responses of many such sensors, each representing a pixel in the image. Intuitively one can interpret an image as the responses of many sensors arranged in a 2D array floating in the virtual scene. Due to the stochastic nature of selecting path trajectories the methods based on Monte-Carlo integration all suffer from variance in the result that becomes visible as noise at different frequencies. Depending on the scene setup, getting clean images can take an impractical amount of time. In general, Monte-Carlo methods for light transport can be classified into two classes: (1) *unbiased* rendering methods that produce results that are on average always correct solutions for the light transport problem, and (2) *biased* rendering methods that accept systematic errors in order to yield more pleasant results in shorter amount of time.

The most common Monte-Carlo rendering methods solve the light transport problem for every pixel separately [50, 62]. This is very wasteful since close-by pixels usually get illuminated similarly by the environment. More generally any two close-by points in space are illuminated similarly in most cases, thus information about the illumination in one point usually is also useful for computing illumination in close-by points. Put in other words, the light transport is usually smooth with respect to changing positions in the scene, meaning it usually does not change abruptly[1]. This can be exploited to speed up rendering by increasing the amount of information at arbitrary points in the scene by using information of close-by points. Naturally, many methods have been developed that exploit this particular property of light transport for rendering images: some interpolate the incoming light at any point, for instance photon mapping [49, 32], vertex merging [33, 28] or irradiance caching [112], some interpolate the sensor responses, for instance adaptive sampling and reconstruction methods [115],

[1]Of course this is not always true since occlusion due to objects in the scene will lead to abrupt changes in illumination. But these effects are very local and the vast majority of illumination is not affected by this.

and some interpolate the path trajectories themselves, for instance path space filtering [54]. However, these interpolation approaches generally introduce some kind of systematic errors in the computation due to simplifying assumptions about the light transport. These systematic errors decrease with increasing computation time and vanish in the limit. For finite computation times the resulting artefacts are often hard to predict and subject to additional parameters in the algorithms, making these algorithms harder to use. Methods without systematic error that also aim at exploiting the fact that similar paths transport similar amount of light are the Metropolis sampling methods [109, 53, 10] that are discussed in Section 3.4. However, these methods suffer from unpredictable low frequency noise and convergence behaviour, and they are hard to use in animations.

1.3 Gradient-Domain Rendering

Recently, a new class of unbiased Monte-Carlo rendering algorithms emerged that can exploit the smoothness of the light transport without introducing the aforementioned problems, the *gradient-domain rendering* algorithms. Gradient-domain rendering samples finite differences between pixels in the image alongside the pixel colors, and then reconstructs a high-quality image from the finite differences and pixel colors. Depending on what type of reconstruction is used, the resulting image can be unbiased or biased but with lower noise. A finite difference of the image can be sampled from *pairs* of paths, where each path in the pair contributes to the illumination of one of the two pixels that are involved in the finite difference computation. The difference of these two path contributions is then a sample of the finite difference.

The paths involved in the finite differences are sampled with techniques that make them as similar as possible, called *correlated sampling.* The finite difference sampling can be implemented on top of existing algorithms and we will show a number of rendering techniques that have been augmented for finite difference sampling. A property of these new sampling techniques is that in regions where the path space is smooth, both paths involved in the finite difference computation are extremely similar. A consequence of this similarity is that the finite differences created in this way have less noise than if they are created with traditional sampling techniques. Combined with the reconstruction, this effectively reduces noise contamination due to the stochastic Monte-Carlo sampling significantly in most regions of an image. Another useful property of gradient-domain rendering is that the image-space finite differences are a much sparser representation of the image than the color image itself. Concentrating computation efforts on the few regions where gradients are big, or in other words *where the action happens* in the image, leads to further benefits.

The first algorithm that used gradient-domain rendering is gradient-domain Metropolis light transport [68]. This method is based on Metropolis sampling and thus suffers from uneven convergence, is ill suited for animations and is challenging to implement. This made the algorithm unattractive for being used in industry. Hence, the goal of this thesis is to develop new algorithms that make gradient-domain rendering more useful in practice.

1.4 Overview

In Chapter 2 we discuss the mathematical foundations of the light transport problem. We describe the measurement quantities required to formulate the light transport problem, discuss the different components of light transport, and then show how the full light transport problem can be described as an equilibrium. In Chapter 3 we discuss Monte-Carlo integration that is the mathematical foundation of most modern methods to solve the light transport problem efficiently. There we also provide a short overview over probability calculus. We also put a large emphasis on advanced variance reduction techniques that help reducing noise in Monte-Carlo methods. In Chapter 4 we describe solution methods to the light transport problem that are based on Monte-Carlo integration. We discuss, in that order, path tracing, bidirectional path tracing and Metropolis light transport. We also give a quick overview of recent noise reduction methods called adaptive sampling and reconstruction that can be used in conjunction with Monte-Carlo rendering. Chapter 5 describes previous work on gradient-domain rendering on which our contributions build upon most directly. We first describe the basic concepts and theoretical properties of gradient domain rendering and show how they where applied on gradient-domain path tracing [57] and gradient-domain Metropolis light transport [68]. Chapters 6 to 9 consist of our unaltered publications on gradient-domain rendering. Finally, Chapter 10 concludes our findings and also discusses some potential avenues for future research in the field of gradient-domain rendering.

1.5 Contributions

In this thesis we present a set of contributions to gradient-domain rendering. Publications in which the author of this thesis was the primary contributor or one of the primary contributors are used in unaltered form in Chapters 6 to 9. These publications include

- **Improved Sampling for gradient-domain Metropolis Light Transport** [75]. A set of improvements upon the original Gradient-Domain Metropolis Light Transport algorithm [68] that reduce variance of the sampled gradients. This is achieved by proposing a new sampling technique resulting in even lower noise and by proposing a combination scheme for different sampling techniques for gradients. Additionally, the original method is generalized to use arbitrary difference constraints instead of finite differences in the $x-$ and $y-$direction, which yields further benefits.

- **Gradient-Domain Bidirectional Path Tracing** [73] An integration of gradient-domain rendering with bidirectional path tracing, including efficient sampling techniques for gradients.

- **Regularizing Image Reconstruction for Gradient-Domain Rendering with Feature Patches** [76]. A combination of gradient-domain rendering with image-space denosing. This is done by introducing a new regularized reconstruction which aims at constructing a smoother image by exploiting feature data.

- **Temporal Gradient-Domain Path Tracing** [74]. An extension to gradient-domain path tracing that also generates gradients across the time dimension. The solution is easy to add on top of existing gradient-domain path tracing implementations and greatly increases temporal coherence of animations.

Additionally, with the permission of all involved authors, parts of the following work that the author of this thesis co-authored are used in Chapter 5 (Section 5.3 and 5.4.3):

- **Gradient-Domain Path Tracing** [57]. A generalization of Gradient-domain metropolis light transport to unidirectional path tracing, including a theoretical framework that justifies why gradient-domain rendering is beneficial for algorithms that use uniform sampling distributions.

Chapter 2

Light Transport

Light transport algorithms compute how light interacts with an environment and are used in computer graphics to compute how a virtual environment would be seen by a virtual camera or observer. Such algorithms create an image by simulating the process that creates images in our physical world and are thus often referred to as *physically-based rendering* algorithms. These methods simulate how photons are emitted by light sources in a scene, how they travel through space and interact with matter and finally how sensors are stimulated by them. The culmination of many such stimuli for many sensors is then interpreted as an image.

The biggest appeal of physically-based rendering is that, given a simulation that is accurate enough, it is able to create images that are indistinguishable from real photographs. Surprisingly, pysically-based rendering is also getting used more and more to create non-photo-realistic images [55]. For instance, the shapes of objects and material properties can be changed to achieve a cartoonish look while the underlying physically-based algorithm will still ensure that the images look consistent. This means shadows will appear at the right place, reflections will be correct and even caustics will appear. This led to a unique look that has been used extensively in the last decade to create animation movies.

Common light transport algorithms in rendering use *ray optics* to model light. Ray optics is based on some simplifying assumptions about the nature of light: light is assumed to be consisting of infinitesimally small particles called *photons* travelling along straight lines with infinite speed. The color of such a photon is determined by the amount of energy that it transports. When these photons interact with a surface they are reflected, transmitted, absorbed or emitted. We further assume that photons do not interact in any way with each other. This implies that light is linear with respect to color and intensity. This means for instance that an optical system using the sum of two light sources as input will generate the same result as computing the system for each light source separately and then summing up the results. The linearity makes this model very appealing for light transport simulations since computing the full light transport in a scene can be reduced to compute all possible *paths* that a photon can travel through the scene separately.

However, the ray optics model is incomplete since it ignores the electromagnetic and quantum nature of light. It does therefore not describe the full spectrum of light phenomena that are observed in the real world. Specifically, ray optics starts to fail at describing light's behaviour on geometry that changes on scales approaching the wavelength of the emitted light. Hence effects caused by very small geometric features like diffraction due to microscopic scratches on a CD's surface or light slightly "bending" around sharp corners cannot be explained or simulated using ray optics. Similar problems occur with polarization effects or fluorescence. Luckily, in our day to day life these effects are usually not very prominent or too small in scale to be noticeable.

Although there are methods that use more complete models of light in rendering like waveoptics [102] or the finite speed of light [46], none of them are widely used in computer graphics because they are much more expensive to use. Therefore, in this thesis we will only focus on light transport algorithms based on ray optics.

In this chapter we will first give a short introduction into radiometry. Radiometry describes how light can be measured. These quantities are then used to measure and describe light transport in the remainder of this chapter. We will describe light transport in terms of its two basic operations. First, we will describe light scattering, this means how light is scattered when it interacts with a surface. Then we will describe how light is propagated from one surface to another. These two operations will then be combined to describe the full light transport in a scene as an equilibrium between the incoming and outgoing light at any point in a scene.

Note that our contributions have not been generalized to rendering volumes yet, therefore we will not include volumes in our discussion of light transport. We will therefore assume that light can only be scattered on surfaces of objects but not in mid air by interacting with particles as it happens for instance in clouds, smoke, water or fog. We refer to work by Veach [106], Cerezo et al. [8], Pharr and Humphreys [90] and Jarosz [47] for more details on this subject.

2.1 Radiometry

Radiometry describes measurement quantities for light. As such it describes the basic quantities that we want to measure when simulating light transport. Specifically we will discuss *radiant flux*, *irradiance* and *radiance*. These quantities are measured in terms of the *spectral power distribution* that describes the amount of light at every wavelength [90].

2.1.1 Spectral Power Distribution

In physics light is usually described as electromagnetic radiation quantified as packets called photons. Such photons are described by their wavelength that determines at which frequency the electromagnetic wave oscillates and by their polarization that determines how this oscillation is oriented. The higher the energy that a photon carries is, the faster it oscillates respectively the smaller its wavelength is. The spectrum of wavelengths interesting for creating images is the narrow band between 370 nm to 730 nm that includes all colors that can be perceived by the human eye. However, recall that we are interested in ray optics that ignores the wave nature of light. The notion of photons having a certain wavelength that determines their color is however still useful for describing light.

In radiometry light is not quantified as single photons but as a continuous distribution over the different wavelengths: the so-called spectral power distribution (SPD). Intuitively, one can imagine this quantity as packets of many photons where the SPD $S(\nu)$ describes their distribution in terms of wavelengths ν. Since we cannot store continuous distributions, SPDs are usually represented in terms of basis functions that allow approximating the SPD with only a few coefficients. Many such representations exist, the most commonly used are representations with three basis functions where each of them roughly corresponds to the three types of color receptors in the human eye, like the CIE1931 color space. Many sophisticated renderers nowadays do however support spectral rendering where more basis function are used or where the different wavelengths of the SPD are sampled in a stochastic way with Monte-Carlo sampling [113] (see Chapter 3).

2.1.2 Radiant flux

Radiant flux, also called radiant power, is denoted by the symbol Φ and measures how much radiant energy passes through a surface in space per unit time s. This radiant energy is proportional to the number of photons that passes through the surface. The radiant energy $Q(s)$ measures the electromagnetic radiation in joules J. Radiant flux is thus defined as radiant energy per unit time

$$\Phi(s) = \frac{\partial Q(s)}{\partial t}. \tag{2.1}$$

The radiant flux is measured in joules per second or watts, $W = J/s$. For instance, in order to measure the radiant energy emitted by a point light source we could measure the flux on spheres surrounding that point light. Measuring the radiant flux per wavelength ν leads to another useful quantity; the *spectral flux* $\Phi(\nu) = \partial\Phi/\partial\nu$.

2.1.3 Irradiance

Irradiance measures the radiant flux *arriving* on a surface, per surface area. Irradiance is a function of a point on the surface x:

$$E(x) = \frac{\partial\Phi(x)}{\partial A(x)}, \tag{2.2}$$

where Φ is the incoming radiant flux. We measure it with respect to the surface area measure around x, $A(x)$. $\partial A(x)$ is an infinitesimally small surface patch around a point x. If we measure the radiant flux emitted by the surface we talk of *radiant exitance* instead of irradiance. To better distinguish both we will use different subscripts for both terms, irradiance will be denoted by E_i and radiant exitance by E_o. Both irradiance and radiant exitance use as units watts per square meters, $W \cdot m^{-2}$. Analogous to spectral flux, the irradiance per wavelength ν, $E_i(x,\nu) = \partial E_i(x)/\partial\nu$, is called *spectral irradiance.*

2.1.4 Radiance

Radiance measures the radiant flux leaving or arriving on a point x from direction ω, per unit area surface perpendicular to ω and per unit *solid angle.* In order to define this we must first clarify the term solid angle:

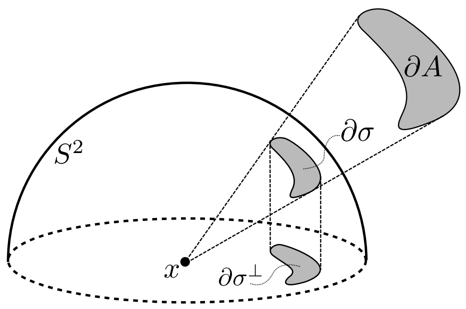


Figure 2.1: A geometric interpretation of the relationship of ∂A, $\partial\sigma$ and $\partial\sigma^\perp$. The surface patch A gets first projected onto the solid angle around x and then onto the *projected* solid angle around x.

Solid angles are used to measure how large objects appear from an observer. It is a measure for questions like "How much of the night sky is covered by the moon?". It is expressed in the dimensionless steradians (sr). Intuitively, given an observer at point x and an observed surface A, the solid angle measures the area of the intersection of a cone originating at x and spanning over A, and a unit hemisphere S^2 around x (Figure 2.1). Given this background the radiance is

$$L(x,\omega) = \frac{\partial^2\Phi(x,\omega)}{|cos\theta|\partial A(x)\partial\sigma_x(\omega)}, \tag{2.3}$$

where $cos\theta = |n_x \cdot \omega|$ and n_x is the surface normal around x. Intuitively this dot-product accounts for the fact that radiant energy coming from grazing angles is distributed over larger areas which leads to less energy per unit surface area. In most light transport literature the dot product is absorbed by the solid angle measurement leading to the *projected solid angle*

$$\sigma_x^\perp(\omega) = |n_x \cdot \omega|\sigma_x(\omega). \tag{2.4}$$

Radiance is measured as watts per steradian per square meter, $W \cdot sr^{-1} \cdot m^{-2}$. Radiance per wavelength ν, $L(x,\omega,\nu) = \partial L(x,\omega)/\partial\nu$, is called *spectral radiance.*

Note that irradiance is the radiance integrated over all incoming directions

$$E_i(x) = \int_{S^2} L(x,\omega)d\sigma_x^\perp(\omega). \tag{2.5}$$

2.2 Light Scattering

In ray optics photons interact with surfaces by being reflected, transmitted or absorbed (or any combination thereof). In order to describe these interactions we must be able to model how light scatters when interacting with a surface. We do this by using mathematical functions that tell us what fraction of light incoming from a specific direction is reflected into another specific direction.

2.2.1 Surface Scattering

Scattering on surfaces can be described by *bidirectional reflectance distribution functions* (BRDF) [86]. The BRDF describes the fraction of light incoming from a certain direction ω_i that is reflected

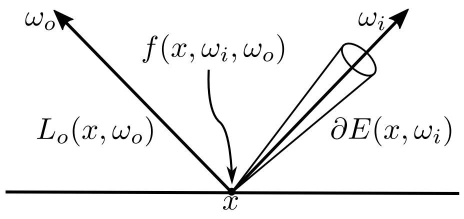


Figure 2.2: A depiction of a BRDF.

in another direction ω_o

Formally the BRDF is defined as the partial derivative of the radiance towards ω_o with respect to the differential irradiance from a infinitesimally small cone of directions around ω_i at a surface position x:

$$f(x, \omega_i, \omega_o) = \frac{\partial L_o(x, \omega_o)}{\partial E(x, \omega_i)} = \frac{\partial L_o(x, \omega_o)}{L_i(x, \omega_i) d\sigma_x^{\perp}(\omega_i)}. \tag{2.6}$$

The second step follows from Equation 2.5. Note that both ω_i and ω_o point away from the surface. The main property of the BRDF is that it is energy preserving, that is

$$\int_{S^2} f(x, \omega_i, \omega_o) d\sigma_x^{\perp}(\omega_i) \leq 1,$$

where S^2 is the hemisphere at x. The energy preservation property ensures that no new energy gets created without emission. This means that a non-emissive surface cannot reflect more light than it receives. Another important property is that the BRDF is symmetrical, that is

$$f(x, \omega_i, \omega_o) = f(x, \omega_o, \omega_i).$$

This leads to the *Helmholtz reciprocity* that states that reversing a path in a pure BRDF environment by swapping emitter and sensor does not change the measured flux on the sensor. This property is important for light transport algorithms since it allows paths to be constructed in reversed order. Note that neither energy preservation nor symmetry must hold for non-physically based BRDFs.

The BRDF only describes reflection, the analogous function that describes transmittance is the *bidirectional transmittance distribution function* (BTDF). While the pairs of directions ω_i and ω_o for the BRDF are on the same hemisphere of the surface x they are on opposite hemispheres for the BTDF. Note that the BTDF usually is not symmetric. Fortunately, there is a simple solution to this problem such that all BSDFs following physical laws can be described in a symmetric way [106].

Many materials are composed of both BRDFs and BTDFs as for instance water surfaces where a fraction of the light gets reflected and a fraction of the light gets refracted (see Figure 2.3) The BRDF and BTDF are usually unified by the *bidirectional scattering distribution function* (BSDF).

2.2.2 BSDF Types

In our tangible world a wide range of surfaces occur. The reflection properties of these surfaces can be arbitrarily hard to model precisely since they are often determined by microscopic structures and imperfections that can be of arbitrary complexity. The most direct way to construct BSDFs is to

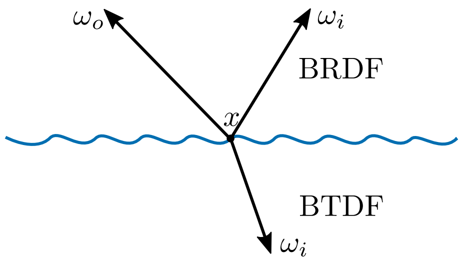


Figure 2.3: A watersurface has a BRDF and BTDF component.

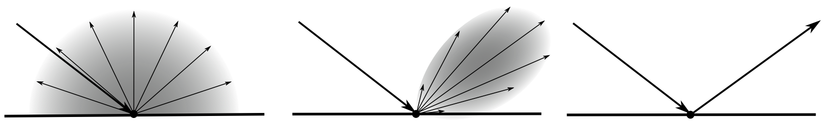


Figure 2.4: From left to right: a diffuse, a glossy and a specular BRDF.

directly measure the reflectance properties of a surface physically with measurement devices and then to use this data directly during rendering. For instance, the data can be tabulated for the measured ω_o and ω_i directions and then looked up during rendering. The disadvantage with this method is that measuring reflectance properties physically is usually an expensive and tedious process and that look-up and storage of this data is also expensive.

More commonly used BSDFs of real materials are approximated by simple mathematical models. Some of these models are phenomenological, meaning they do not really have a physical meaning. Such models are often designed with performance in mind and intuitive control of the parameters. The danger with such models is that, since they are not rooted in physical laws, they do not necessarily yield consistent results. For instance, energy preservation can be violated or they might behave in unexpected ways in certain situations.

Thus, photo-realistic rendering often relies on physically-based models. These models are usually more expensive to evaluate than phenomenological models but yield more realistic results. One example of such a model is the *micro-facet model* [11, 3]. We will not directly dive into the micro-facet theory in this thesis and instead explain it on a intuitive level: on a high level, micro-facet models allow to model surfaces with different levels of roughness. The roughness describes how strongly light gets scattered when interacting with the material. Micro-facet models achieve this by describing surfaces as consisting of many microscopically-sized randomly orientated surface patches (”*facets*”). The larger the variance of the facets orientation is, the rougher a material appears. On the ends of this spectrum are perfectly diffuse materials that scatter light equally in every direction and perfectly specular materials that scatter light only into a infinitesimally small cone of directions. Between those two extremes are the glossy materials. Figure 2.4 shows 2D slices of BSDF examples. In three dimensions a BSDF can also behave in an anisotropic fashion, meaning it can behave differently depending on the viewing direction. Brushed metal is a typical example of a anisotropic BSDF where the material appears rougher in the direction perpendicular to the brushing direction. Independent of the exact material roughness properties, we must further differentiate between conductor materials and dielectric materials. Conductors reflect a part of the light and turn another part of the light into heat by absorbing it. This means conductors do not refract light. Dielectric materials are different in the sense that they partially reflect light and partially refract light. The ratios of reflected light and absorbed light for conductors and the reflected light and refracted light for dielectrics are described by the Fresnel equations [39]. Examples for conductors are metals and for dielectrics water or glass. More complex materials are modelled as surfaces with several BSDF layers [36, 44]. In the simplest case this can be something like a wooden floor with a coating layer on top of it, but it can also be something very complex like skin that is composed of several translucent layers with dense participating media in between the layers.

2.3 Light Propagation

2.3.1 The Reflection Equation

By multiplying Equation 2.6 by the differential irradiance and integrating it, we get the *reflection equation*:

$$L_r(x,\omega_o) = \int_{S^2} f(x,\omega_o,\omega_i) L_i(x,\omega_i) d\sigma_x^{\perp}(\omega_i), \tag{2.7}$$

which states that the total reflected light in an outgoing direction is the integral of the incoming radiance times BSDF over all incoming directions. It is crucial for the understanding of light transport that this equation describes an equilibrium between the incident and reflected radiance.

2.3.2 The Rendering Equation

Radiance does not change along a ray, thus

$$L_i(x,\omega_i) = L_o(x',-\omega_i) \tag{2.8}$$

where

$$x' = \nu(x,\omega_i) \tag{2.9}$$

is the *ray casting function* that describes the first visible point viewed from x in direction ω_i. This shows that L_o and L_i are interchangeable and therefore we will only use L_o from now on and drop the subscript, i.e. $L = L_o$. By taking into account that the total outgoing radiance is the sum of the reflected light L_r and the emitted light L_e, i.e. $L = L_e + L_r$, we can write down the *rendering equation* [50, 42]:

$$L(x,\omega_o) = L_e(x,\omega_o) + \int_{S^2} f(x,\omega_i,\omega_o) L(x',-\omega_i) d\sigma_x^{\perp}\omega_i. \tag{2.10}$$

The solution to the rendering equation describes the equilibrium of radiance in the entire scene. This means that the solution of this equation describes a state of balance between the incident, reflected and emitted radiance everywhere in the scene simultaneously. In our tangible world - expect for phenomena of astronomical scale - this state is reached nearly instantly due to the speed of light. Hence, the light that we perceive as humans is always in the state of balance as described by the rendering equation.

2.3.3 Transport Direction

Recall from Section 2.2 that the Helmholz-reciprocity states that emitters and sensors can be exchanged and the flux on the sensor will not change. This implies that measuring light in the direction of radiance transport, i.e. from emitters to sensor, is equivalent to measuring a theoretical quantity that travels in the reversed direction from sensors to emitters. This theoretical quantity is commonly described as *importance* or *light measured along the importance direction*, since it measures how important paths are for the sensor. The equivalence of light measured along radiance and importance direction[1] gives us more freedom in how to construct light paths and allows light transport to be solved more efficiently: only a small subset of possible trajectories starting from emitters do actually reach a sensor, thus it is more efficient to develop algorithms that construct paths in the importance direction from the sensor towards the light than the other way round.

2.3.4 The Measurement Equation

The rendering equation gives us a tool to describe the radiance everywhere in a scene in a consistent way. However, in practice we are only interested in the light arriving at a specific location of the scene in order to render an image. That is, we want to compute the radiance arriving at our virtual camera. The virtual camera is modeled as an array of sensors. Each sensor is associated with a single pixel, thus the response measured at a sensor is the associated pixel's color value. The response of each sensor p is described by the *measurement equation*:

$$I_p = \int_{P_A} \int_{S^2} W_p(x, \omega) L(x, \omega) dA(x) d\sigma_x^{\perp}(\omega), \tag{2.11}$$

where P_A is the aperture of the sensor and S^2 is the sphere of all unit directions. The sensitivity of the sensor towards radiance with respect to position and direction is specified by the *importance function* $W_p(x, \omega)$ for sensor p. Typically each sensor has a non-zero sensitivity only for a very small spatial region representing the sensor aperture[2] and a very narrow cone of directions. The measurement equation combined with the rendering equation makes it possible to design algorithms that can generate images by mimicking the light transport in the real world.

2.3.5 Surface Form of the Rendering Equation

In this section we will reformulate the rendering equation such that it is a function of positions instead of a function of directions. This formulation is a useful step towards reformulating light transport into the path space formulation in Section 2.3.6.

Equation 2.10 defines the light transport problem as a recursive integral over solid angles. We want to reformulate it as an integration problem over surface areas. To do so, we first change the integration variable from projected solid angle to surface area. The projection of a surface patch around x' onto solid angle at x depends on the orientation of the surface with respect to x as well as

[1]With refractive materials equivalence is usually not achieved out of the box. However, Veach [106] describes simple workarounds to still achieve equivalence.

[2]If a pinhole camera is used, the aperture is a infinitesimally small point in space.

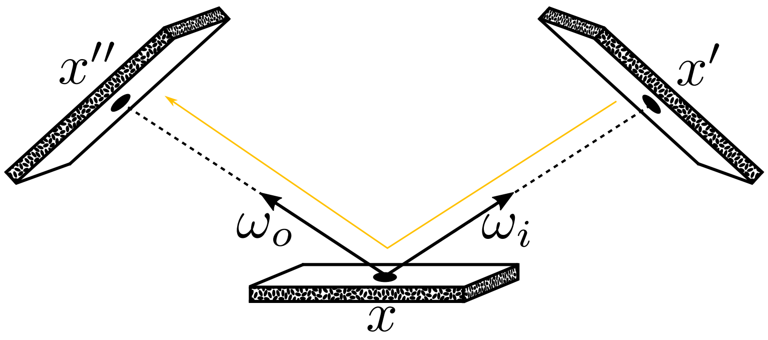


Figure 2.5: Parametrizing paths by surfaces. This figure shows the relationship between the incoming and outgoing directios ω_i and ω_o with the surface patches x,x' and x'' used in this subsection. The yellow arrow shows the direction of the light flow.

its distance to x (recall Figure 2.1):

$$d\sigma_x^{\perp}(\omega) = \frac{\cos\psi\cos\theta}{||x'-x||^2} dA(x) \tag{2.12}$$

where $\cos\psi = n_{x'} \cdot \omega_i$, $\cos\theta = n_x \cdot -\omega_i$ and $\omega_i = \frac{x'-x}{||x'-x||}$. With this in mind the rendering equation becomes

$$\begin{aligned} L(x,\omega_o) =& L_e(x,\omega_o) + \\ & \int_P f(x,\omega_i,\omega_o) L(x',-\omega_i) \underbrace{\frac{V(x',x)|\cos\theta|\cos\psi}{||x'-x||^2}}_{G(x',x)} dA(x'). \end{aligned} \tag{2.13}$$

We call $G(x',x)$ the *geometry term* between x' and x. $V(x',x)$ is a binary function called *visibility term* with

$$V(x',x) = \begin{cases} 1 \text{ if } x \text{ and } x' \text{ are mutualy visible and} \\ 0 \text{ otherwise.} \end{cases}$$

The directions can be described by the surface points they originate from and the surface they point to, e.g. $\omega_i = \frac{x'-x}{||x'-x||}$ (see figure 2.5). The transport of light between any three surfaces x, x' and x'' can thus be parametrized by the surfaces themselves. This leads the *three-point surface form of the rendering equation*:

$$\begin{aligned} L(x'',x) =& L_e(x'',x) + \\ & \int_P f(x',x,x'') L(x,x') G(x,x') dA(x'). \end{aligned} \tag{2.14}$$

In this case light is transported from x' to x to x'' as depicted in Figure 2.5.

Similarly the measurement equation can also be transformed in surface form:

$$I_p = \int_{P_A} \int_P W_p(x, x'') L(x'', x) G(x'', x) dA(x) dA(x'') \tag{2.15}$$

2.3.6 Path Space Formulation of Light Transport

The rendering equation (Eq. 2.10) completely describes the light transport problem and can be used to compute images with the measurement equation (Eq. 2.11). However, describing the light transport problem in this way has some limitations: it describes the light transport as one scattering event at a time and thus provides only a local view on the light transport problem. As a consequence, algorithms solving this equation construct paths incrementally, this means by evaluating one scattering event of a path at a time. This leads to conceptually simple solutions to light transport as for instance path tracing [50] that will be discussed in Section 4.2. However, since no global perspective on the problem is provided, algorithms that operate on full paths cannot be fully understood. Veach [108, 106] described a formulation of light transport that operates on full paths and showed how this can be used to formulate new algorithms that can create new paths by combining existing paths [107] (Section 4.3) or by perturbing existing paths [109] (Section 4.4).

In the following we will show how the measurement equation and rendering equation can be reformulated into one simple integral. More specifically, we will show how light transport can be formulated in the form $I_p = \int f_p^\star(\overline{\mathbf{x}}) d\overline{\mathbf{x}}$ where $\overline{\mathbf{x}}$ are paths and $f_p^\star(\overline{\mathbf{x}})$ is a function that evaluates the sensor response at pixel p due to a path $\overline{\mathbf{x}}$. To this end, we will first give a formal description of paths so that we can integrate over them, then we will discuss an operator formulation of light transport that reveals a new way of rewriting the rendering equation, and finally we will show how these ingredients can be combined to formulate light transport as an integral over the space of all possible paths.

Paths and Path Space A path $\overline{\mathbf{x}}$ is described by the ordered sequence of positions x_i, called *vertices*, that describe the positions of all scattering events along the path:

$$\overline{\mathbf{x}} = x_0 x_1 x_2 \ldots x_k. \tag{2.16}$$

The first and last path-vertex represent a point of emission or a measurement on a sensor, while the path-vertices in between mark any sort of scattering event. We define the length of a path, denoted by $\mathrm{len}(\overline{\mathbf{x}})$, as the number of path-vertices it consists of. This means for instance that direct illumination consists of paths of length 3 (one emission, one scattering and one measurement event). It does not really matter whether the vertices are ordered along the light flow or in the reversed direction, but as a convention in this thesis path vertices are always ordered in the importance direction (see Section 2.3.3). This means x_0 is always a vertex on a sensor and x_n a vertex on a emitter. All possible paths of length n are denoted by the space Ω^n. The union of all these spaces is called *path-space* $\Omega = \cup_{i=1}^{\infty} \Omega^i$. This space covers all paths of all lengths that can possibly occur in the scene. Next we define a new measure, the *area-product measure*, that allows us to integrate over paths of length k:

$$\mu_k(\overline{\mathbf{x}}) = \mu(x_0 x_1 ... x_{k-1}) = \prod_{i=0}^{i<k} dA(x_i) \tag{2.17}$$

Operator Formulation The area-product measure above allows us to formulate problems directly on full paths parametrized by their vertices. However, we need to know how the rendering equation can be formulated as a problem over paths directly instead of as a recursive operator. To do so we will discuss Arvos' operator formulation of light transport [2] that reveals how the rendering Equation can be reformulated as sum of integrals over different path lengths. It is noteworthy that the rendering equation is a Fredholm integral equation of the second kind [92] and that these can be solved by Liouville-Neumann series under certain conditions.

Arvo [2] showed that light transport can be expressed in terms of linear operators for scattering K and propagation G that can be solved directly without recursion. K is a linear operator that describes the outgoing radiance L_o in terms of the incoming radiance L_i after applying a single scattering operation:

$$KL_i(x, \omega_o) = \int f(x, \omega_i, \omega_o) L_i(x, \omega_i) d\omega_x^{\perp}(\omega_i) = L_o,$$

and G is a linear operator that describes the incoming radiance at a surface in terms of outgoing radiance from other surfaces:

$$GL_o(x, \omega_i) = \begin{cases} L_o(\nu(x, \omega_i) - \omega_i), \text{ if } \nu(x, \omega_i) \text{ exists} \\ 0, \text{ else} \end{cases} = L_i.$$

where $\nu(x, \omega_i)$ is defined in Equation 2.9. The two operators can be used to reformulate Equation 2.10 to

$$L = L_e + TL,$$

with $T = KG$. Note that we again omit the subscript in L_o and that L_e is the emitted light. By using $(I - T)L = L_e$ with I being the identity operator, the equation above can be rewritten to

$$L = (I - T)^{-1} L_e = SL_e.$$

Arvo further showed that S can be easily computed by series expansion if all materials in the scene are energy conservative:

$$S = (I - T)^{-1} = \sum_{i=0}^{\infty} T^i = T^0 + T^1 + T^2 + ...$$

The rendering Equation 2.10 can then be written in a non-recursive form as:

$$L = \sum_{i=0}^{\infty} T^i L_e = L_e + T L_e + T^2 L_e + ... \tag{2.18}$$

Note that each $T^k L_e$ is an integral over paths of length k. The crucial implication of this formulation is that light transport can be computed by solving integrals of paths of different length independently. This opens the door to a path space formulation of light transport.

The measurement equation (Eq. 2.11) can be formulated as the inner product of L and the importance function W_p from Section 2.3.4. From this we can directly see that

$$I_p = \langle W_p, L \rangle = \sum_{i=0}^{\infty} \langle W_p, T^k L_e \rangle. \tag{2.19}$$

The second step follows from the linearity of the inner product.

Path Space Integral Following Equation 2.18 we can decompose the rendering equation into paths of different length separately. Each term of the form $T^k L_e$ can be written out in area-product measure as:

$$(T^k L_e)(x_0, x_1) = \underbrace{\int_P ... \int_P}_{k \text{ times}} L_e(x_{k+1}, x_k)$$
$$\prod_{i=1}^{k} \left[f(x_{i-1}, x_i, x_{i+1}) G(x_i, x_{i+1}) \right] d\mu(x_2, ..., x_{k+1}).$$

The first two vertices are fixed by the direction and position on which we want to compute the radiance. Extending this term in a similar way as Equation 2.15 in order to compute the sensor response reveals that these represent paths of length $k+2$ from sensor to emitter.

$$I_{p,k+2} = \int_{P_A} \int_P W_p(x_0, x_1)(T^k L_e)(x_0, x_1) G(x_0, x_1) d\mu(x_0, x_1) \tag{2.20}$$

Interestingly, paths with a length below 2 are not expressed in this formulation. This makes sense since a path needs to have at least 2 vertices in order to transport any light to a sensor. Namely, it needs to have at least one vertex on the sensor and one vertex on an emitter. In the classic formulation, P_A is a point on the aperture, but we could easily generalize this to any point in the scene. W_p would simply be zero for all point that are not on the aperture. Doing so allows us to write the integral in area product measure (Eq. 2.17) for paths of length $k+2$,

$$I_{p,k+2} = \int_{\Omega^{k+2}} f_p^\star(\overline{\mathbf{x}}) d\mu_{k+2}(\overline{\mathbf{x}}), \tag{2.21}$$

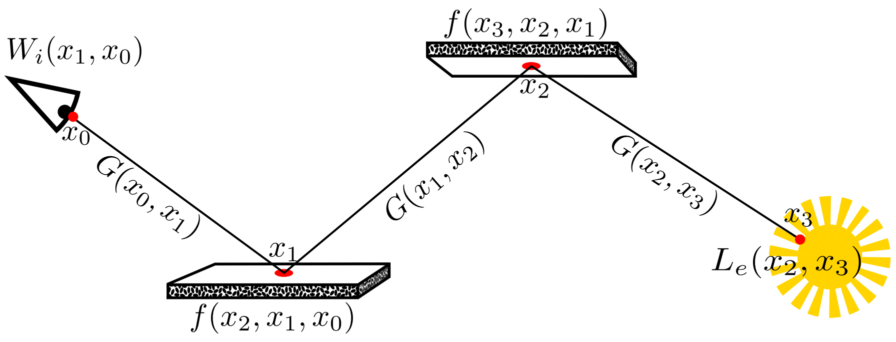


Figure 2.6: Depiction of the different components of the measurement contribution function for a path $\overline{\mathbf{x}}$ with $\mathrm{len}(\overline{\mathbf{x}}) = 4$.

where

$$\begin{aligned} f_p^\star(\overline{\mathbf{x}}) = \quad & W_p(x_1, x_0) \prod_{j=1}^{\mathrm{len}(\overline{\mathbf{x}})-2} f(x_{i+1}, x_i, x_{i-1}) G(x_i, x_{i+1}) \\ & L_e(x_{\mathrm{len}(\overline{\mathbf{x}})-2}, x_{\mathrm{len}(\overline{\mathbf{x}})-1}) G(x_{\mathrm{len}(\overline{\mathbf{x}})-2}, x_{\mathrm{len}(\overline{\mathbf{x}})-1}) \end{aligned} \tag{2.22}$$

is the *measurement contribution function* of path $\overline{\mathbf{x}}$. Figure 2.6 shows an example for $f_p^\star$. Using Equation 2.19 we can now formulate the full measurement equation

$$I_p = \sum_{k=2}^{\infty} \int_{\Omega^k} f_p^\star(\overline{\mathbf{x}}) d\mu_k(\overline{\mathbf{x}}). \tag{2.23}$$

Changing the domain of integration from path spaces of fixed length to the full path space finally allows us to integrate over all paths of all lengths at once. This finally yields the *path space integral*

$$I_p = \int_{\Omega} f_p^\star(\overline{\mathbf{x}}) d\mu(\overline{\mathbf{x}}). \tag{2.24}$$

2.4 Path Expressions

We will see later in Chapter 4 that different approaches to solve the light transport problem are well suited for different types of paths. Therefore, to simplify the discussion later, we review in this section a commonly used notation for describing the interaction along a path in a compact manner that allows us to classify paths.

To this end we use a simplified version of the notation by Heckbert [40], similar to the one used by Veach [106], that classifies each interaction along a path according to the BSDF of the surface that is either diffuse (D) or specular (S). Additionally vertices on the sensor are denoted with E (for eye) and vertices on a emitters with L (for light). A path can then be described as a regular expression consisting of these symbols. The order of symbols could be either in the radiance or importance direction, but to keep it consistent with the path notation we order the symbols in the importance direction.

A simple example is $EDSL$ that denotes a direct caustic path. A more complex path classifications example is $E(S|D)D^*S^+L$. As with regular expressions the symbol "*" means *zero or more occurrences*, the symbol "+" *one or more occurrences* and the symbol "|" *either the right-hand side or the left-hand side*. Therefore, the example above describes a path from sensor to emitter that consist of a specular or diffuse vertex followed by any number of diffuse vertices followed by at least one specular vertex before reaching the emitter.

Even though this notation is useful to explain theoretical concepts it is very limited in practice. For instance it ignores that material properties are rarely modelled as perfectly diffuse or perfectly specular surfaces, since neither of them exist in the real world. Most materials are something in between or even combinations of several layers interacting with each other. However, for the simplicity of discussion we assume that any material interaction can be classified as one of both categories based on its roughness. This means if a material has a roughness value over an certain threshold it is classified as diffuse and otherwise specular. The discussion about multi-layered materials [36] [44] on the other hand is more problematic. Depending on what a certain method aims at it defines the roughness as the average, minimum, maximum of all its layers or only considers the currently sampled layer if only a subset of the layers are evaluated. We will however assume that suitable classification methods are available and not discuss this further.

Chapter 3

Monte-Carlo Integration

In the previous chapter we showed how the problem of synthesizing realistic images can be solved by computing integrals over paths for every pixel. That is by solving the associated measurement equation (Eq. 2.11), or equivalently, its formulation in path space (Eq. 2.24). Despite the conceptionally simple formulations, solving these equations is difficult for two reasons:

First, for non-trivial scene configurations these integrals cannot be computed analytically. Which means that we need to approximate each integral's solution numerically. On a intuitive level this is done by evaluating the integrals at many different positions. Weighted sums of these evaluations are then used as approximations of the integrals.

Second, the integrals are high-dimensional since the paths have (infinitely) many degrees of freedom. This makes the integrals subject to the *curse of dimensionality* that will be discussed later in this chapter. Essentially it states that the cost of approximating an integral numerically with non-stochastic methods goes up exponentially with the number of dimensions of the integral.

The most established method for computing high-dimensional integrals is *Monte-Carlo integration.* Nearly all modern physically-based rendering techniques use Monte-Carlo integration. *Radiosity* algorithms are an alternative set of approaches that solve the light transport equilibrium as a system of linear equations [29, 42, 99]. They are however less practical than the Monte-Carlo integration based methods.

In this chapter we will first give an introduction into the required mathematical foundations for this thesis, then we will give a overview of Monte-Carlo integration and finally discuss some variance reduction techniques.

3.1 Mathematical Foundations

3.1.1 Random Variables

Random variables are created by some arbitrary random process. They are drawn from a certain domain that can be continuous or discrete. In this thesis random numbers will be denoted by capital letters, e.g X. Note that functions applied on random numbers again yield random numbers, e.g. $Y = F(X)$.

Density Functions Random processes are usually described by quantities that describe how likely it is to pick certain values. The *cumulative distribution function* (CDF) describes the probability of picking a value smaller or equal to a certain value

$$P(x) = Pr\left[X \leq x\right]. \tag{3.1}$$

By construction $P(x)$ is always inside the interval $[0,1]$ and a monotonically increasing function. The *probability density function* (PDF) on the other hand describes how likely it is to pick a specific value and is equal to the derivative of the CDF:

$$p(x) = \frac{\partial P(x)}{\partial x}. \tag{3.2}$$

As such the PDF is never negative and integrates to one over its domain.

Canonical Uniform Random Variables The simplest class of random variables that can be produced with pseudo-random number generators (PRNG) are *canonical random numbers* (CRN). That is, continuous random numbers ranging from zero to one that have the same probability for every value in the interval to be picked.

For most applications however, it is desirable to use random numbers that are generated with non-uniform probability densities. Such random numbers usually do not need specialized PRNG since they can be constructed directly from CRNs. In a first step a simple pseudo-random generator produces CRNs and in a second step these CRNs are warped onto functions with the desired distributions. In practice the second step can be done with the *inversion method*, where a CRN X is mapped onto the inverse of the CDF of the desired distribution, i.e. P^{-1}. The new random variable $Y = P^{-1}(X)$ then has per construction the desired distribution. Note that this requires P^{-1} to be computable, which is not always the case.

3.1.2 Probability Calculus

Random processes usually lead to non-deterministic results. Despite this we still want to be able make statements about what kind of results can be expected from a random process and how certain we are with our predictions.

Expected Value and Variance The *expected value* $E[X]$ describes the output of a random process X that can be expected on average. More precisely, it describes the weighted average of possible outcomes, where the weights are the PDF of this outcome:

$$E[X] = \int_D xp(x)dx \tag{3.3}$$

where D is the integration domain.

The *variance* $V[X]$ quantifies the uncertainty of the expected value. It is defined as the expected squared difference of the outcomes of X and the expected value of X:

$$V[X] = E\left[(X - E[X])^2\right]. \tag{3.4}$$

A direct consequence of this definition is that for any constant c

$$V[cX] = c^2 V[X]. \tag{3.5}$$

The square root of the variance $\sigma[X] = \sqrt{V[X]}$ is called *standard deviation*.

Covariance and Correlation Sometimes it is important to describe the relationship between random processes. A quantity to measure by how much two random variables X and Y change together is the *covariance*:

$$\text{Cov}[X, Y] = E\left[(X - E[X])(Y - E[Y])\right]. \tag{3.6}$$

Two random variables are *independent* if their covariance is zero. Note that covariance is a generalization of variance, since the covariance of a variable to itself is the variance, i.e. $\text{Cov}[X, X] = V[X]$. With this and Equation 3.5 we can compute the variance of weighted sums of random variables:

$$\begin{aligned} V\left[\sum_{i=1}^{N} c_i X_i\right] &= \sum_{i,j=1}^{N} \text{Cov}[c_i X_i, c_j X_j] \\ &= \sum_{i=1}^{N} c_i^2 V[X_i] + 2 \sum_{1 \le i < j \le N} c_i c_j \text{Cov}[X_i, X_j] \end{aligned} \tag{3.7}$$

Note that the second term disappears for independent random numbers. As a generalization of variance, covariance quantifies both the relationship of two random variables as well as their uncertainty. A quantity that only measures the relationship of random variables is *Pearson's correlation coefficient*, often simply called correlation:

$$\text{Cor}[X, Y] = \frac{\text{Cov}[X, Y]}{\sigma[X]\sigma[Y]}. \tag{3.8}$$

Estimators Many complex problems are to complex too be computed fully in an analytical way. The light transport problem is such a problem. Instead the solutions for these problems need to be computed numerically from observed sampled data. In this context the quantity that should be estimated is called *estimand*, the estimated result is called *estimate* and the rule that produces an estimate from a set of samples is called the *estimator*. Usually the quality of an estimator depends on the number of samples. As a notational convention when we denote an estimator we will always show the number of samples as a subscript, i.e. $\tilde{f}_N$.

Consistency and Bias Sometimes estimators do not really estimate the correct thing in all cases. This can for instance happen unintentionally due to systematic errors in the measurements or models with insufficient precision. But it can also happen intentionally by approximating complex problems by simpler ones for the sake of efficiency. The systematic error is called *bias* and measures by how much the expected value of the estimate is off from the estimand. Formally, given a estimand θ and its estimator $\tilde{f}_N$, the bias is defined as

$$\mathrm{Bias}[\tilde{f}_N] = E[\tilde{f}_N] - \theta. \tag{3.9}$$

A estimator is called *unbiased* when the bias is zero for any positive number N:

$$\forall N > 0 : \mathrm{Bias}[\tilde{f}_N] = 0, \tag{3.10}$$

and called *consistent* when it converges towards the estimand:

$$lim_{N \to \infty} \tilde{f}_N = \theta. \tag{3.11}$$

Note that both things are not equivalent. It is possible for an estimator to have only one of both properties. Unbiased but not consistent estimators are as well possible as biased but consistent estimators.

In computer graphics *biased* algorithms usually describe methods based on biased estimates regardless of whether they are consistent or not. In fact when speaking of biased algorithms in light transport most of the time biased but consistent estimators are meant.

Error The variance and the bias combined give us a tool to measure the accuracy of an estimator. That is how certain and how correct outcomes of the estimator are. The most commonly used measurement for error is the *mean squared error* (MSE), that measures the expected squared error that is equal to the sum of the squared bias and the variance:

$$\mathrm{MSE}[\tilde{f}_N] = E[(\tilde{f}_N - \theta)^2] = \mathrm{Bias}[\tilde{f}_N]^2 + V[\tilde{f}_N]. \tag{3.12}$$

As we will see later, when a perfect estimator is not possible, variance can often be reduced by increasing bias and vice versa. So designing good estimators often boils down to finding good trade-offs between variance and bias.

For computations of the expected error we are usually more interested in the *root mean square error* (RMSE), that is the square root of the MSE, $\mathrm{RMSE}[\tilde{f}_N] = \sqrt{\mathrm{MSE}[\tilde{f}_N]}$.

Another useful quantity is the *peak signal-to-noise ratio* (PSNR), which measures the maximal possible signal intensity with respect to the MSE of the signal. This quantity is usually measured in the log-domain and larger values are better than smaller values:

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{\max(I)}{\mathrm{MSE}[I]} \right).$$

3.2 Introduction to Monte-Carlo Integration

Motivation Evaluating integrals numerically is usually done by evaluating the integral at discrete positions, and then combining these evaluations to estimate the complete integrals value. Essentially this means we approximate the integrals with finite sums of selected weighted samples:

$$\int_D f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} \omega_i f(x_i) \tag{3.13}$$

These methods are called *quadrature rules* and differ from each other only by the rules that select the samples $x_i \in D$ and by how the weights ω are set. Note that these rules are deterministic, thus quadrature rules do not lead to any variance. The approximation error in Equation 3.13 depends on how large N is. If $\sum_{i=1}^{N} \omega_i/N = 1$ then the error is zero in the limit, which means that $\lim_{N\to\infty} \sum_{i=1}^{N} \omega_i f(x_i) = \int_D f(x)dx$. The rate at which the error goes down as a function of the number of samples is called *convergence rate*. For one dimensional integrals the convergence rate is $O(N^{-r})$ with $r \geq 1$. Note that the value of r depends on the specific quadrature rule that is used.

Unfortunately, as it turns out extending this formulation to multiple dimensions is problematic since

$$\begin{aligned}&\int_{D_1}\int_{D_2}...\int_{D_s} f(x_1, x_2, ..., x_s)dx_1dx_2...dx_s \\ &\approx \sum_{i_1=0}^{N}\sum_{i_2=0}^{N}...\sum_{i_s=0}^{N} \omega_1\omega_2...\omega_s f(x_1, x_2, ..., x_s).\end{aligned} \tag{3.14}$$

Intuitively, one can see that the number of weights and samples increases exponentially with the number of dimensions s. As a consequence, one can prove that quadrature rules cannot have a convergence rate better than $O(N^{-r/s})$ with $r \geq 1$. This means that getting the same approximation error for a high dimensional integral requires exponentially more samples than doing so for a lower dimensional integral. This is known as the *curse of dimensionality.*

Since the integrals used in light transport computations are high dimensional, using deterministic quadrature rules is infeasible due to the curse of dimensionality. However, as it turns out introducing randomness in our estimator provides an elegant way out of this dilemma.

The Monte-Carlo Estimator The idea of Monte-Carlo integration is to approximate an integral similarly to Equation 3.13 by a weighted sum of samples. The main difference to quadrature rules is that samples are selected *stochastically* instead of with deterministic rules. By additionally setting the weights ω_i to be the reciprocal of the PDF of picking the associated sample, we can show that the estimator will converge to the correct solution. Formally, the Monte-Carlo estimator F_N of a function f using N samples is defined as:

$$F_N = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)} \tag{3.15}$$

with $X_i \in D$. This is indeed an unbiased estimator of the integral that we want to approximate, since

$$\begin{aligned} E[F_N] &= E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] = \frac{1}{N}\sum_{i=1}^{N}\int_D \frac{f(x)}{p(x)}p(x)dx \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_D f(x)dx = \int_D f(x)dx. \end{aligned} \tag{3.16}$$

The most important property of the Monte-Carlo estimator is that its convergence rate is independent of the number of dimensions. To show this we need to understand how the error behaves as function of the number of samples. We know that the Monte-Carlo estimator is unbiased, so its RMSE is equal to its standard deviation, $\text{RMSE}[F_N] = \sigma[F_N]$. For algebraic reasons we first show how the variance behaves as a function of the number of samples. Assuming that the samples are independent, then

$$\begin{aligned} V[F_N] &= V\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N^2}\sum_{i=1}^{N}V\left[\frac{f(X_i)}{p(X_i)}\right] = \frac{1}{N}V\left[\frac{f(X_1)}{p(X_1)}\right]. \end{aligned} \tag{3.17}$$

The second step follows from Equation 3.7 and the fact that we use independent random variables. Since $\sigma[F_N] = \sqrt{V[F_N]}$ and $F_1 = f(X_1)/p(X_1)$ we can directly conclude that

$$\text{RMSE}[F_N] = \sigma[F_N] = N^{-1/2}\sigma\left[F_1\right]. \tag{3.18}$$

This means that the convergence rate is $O(N^{-1/2})$. Note that this statement does not put any bounds on the error of the estimate of a single realization of the estimator, it only puts bounds on the expected error. This means, on average the Monte-Carlo estimator is guaranteed to have an expected error of $O(N^{-1/2})$, but a single estimate can still have a arbitrarily large approximation error.

3.3 Variance Reduction Techniques

Equation 3.17 shows that the variance of the estimator depends on the number of samples, thus the most straightforward way to reduce the error of the Monte-Carlo estimator is to increase the number of samples. Unfortunately the cost of of the Monte-Carlo estimator increases linearly with the number of samples. Thus, following Equation 3.18, to halve the expected error of the estimator the costs of evaluating the estimator quadruple. Reducing variance to a level that is not perceivable any more can thus take an impractical amount of time. It is therefore important to examine alternative ways of reducing variance. *Variance reduction techniques* are techniques that reduce the variance of a Monte-Carlo estimator *without* increasing the number of samples.

The remainder of this chapter will give an overview of such techniques that are related to our contribution. Note that most techniques mentioned below are orthogonal and can be combined to

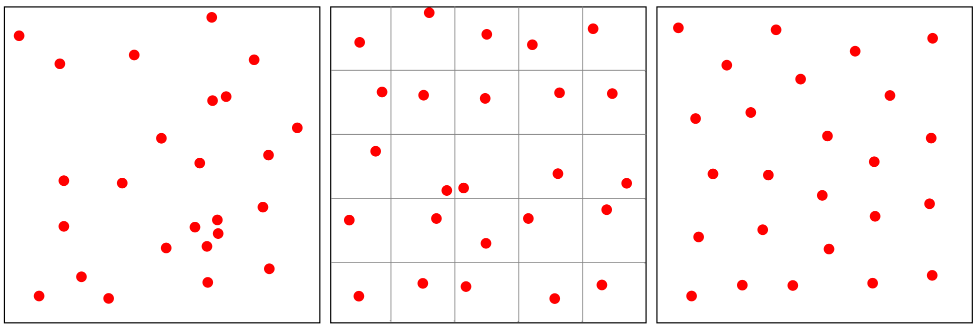

(a) Random Sampling (b) Stratified Sampling (c) Blue Noise Sampling

Figure 3.1: Different sampling strategies

further reduce variance.

3.3.1 Improved Coverage of the Integration Domain

One of the problems with purely random sampling is that the distribution of samples over the integration domain can be very uneven. For simplicity of discussion we will assume that the sampling domain with s degrees of freedom is a s dimensional hypercube. Samples from this hypercube can be mapped onto any s dimensional sampling domain so this does not constrain our discussion in any way. Pure random sampling can lead to clumping in certain regions and to missing entire regions of the sampling domain. As a consequence, important features of the integrand can be missed which increases variance. An example of such clumping in a 2D sampling space is depicted in Figure 3.1a. The simplest way to improve this behaviour is by using stratified sampling. Stratified sampling describes a sampling technique in which the sampling domain is subdivided into non-overlapping domains, i.e. the s-dimensional hypercube is divided into n^s cells, called *strata*, that are each sampled independently. The advantage of this method is that it guarantees that in every strata a certain amount of samples will be generated which generally improves coverage of the sampling domain. A fundamental problem with this kind of sampling is that it is unclear how to effectively generate strata for high-dimensional sampling domains, since again the curse of dimensionality leads to a exponential increase of strata in order to cover the sampling domain effectively. However, it is possible to apply stratified sampling only on a subset of the dimensions and to sample the other dimensions with random sampling. For instance a common way to use stratified sampling in rendering is to subdivide the 2D pixels of the image plane into sub pixels and to perform stratified sampling only on these dimensions [12, 79]. Figure 3.1b visualizes this. Note that inside of each strata clumping can still occur, thus these methods are not optimal. Shirley [98] showed that clumping can be effectively reduced with *half-jittered* sampling, that restricts samples in every strata to be located around to the center of the strata. A related alternative designed to sample higher dimensional domains more efficiently is *latin hypercube sampling* that was first introduced in rendering as *N-rooks sampling* [98]. With this method sampling patterns for the hypercube are generated by first generating samples only for the stratas in the diagonal of the hypercube. These samples are then shuffled along each dimension independently to create evenly distributed samples in the hypercube. Similar methods

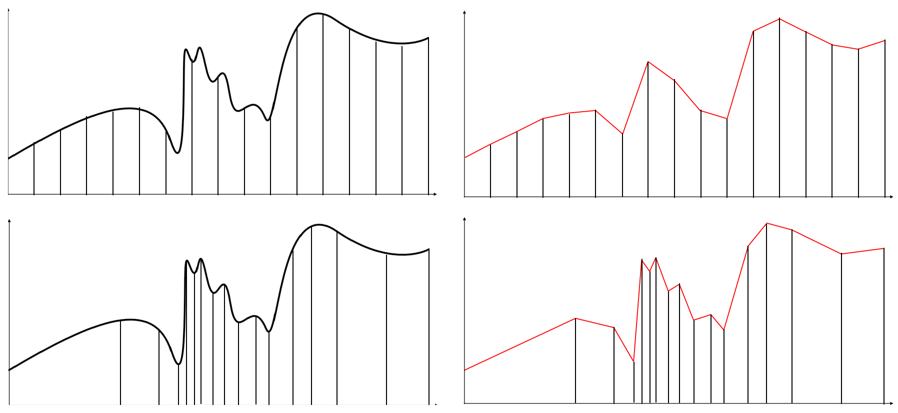


Figure 3.2: Adaptive sampling. On the left: continuous signal, on the right: reconstruction. Top: signal with 16 uniformly distributed samples. Bottom: same signal with 16 adaptively distributed samples. Note hoe the later one captures fine details of the signal much better. This figure is reproduced from Manzi [72].

with even better distributions are (s, t)-sequences and (t, m, s)-nets [87, 60]. These sequences are not random any more, thus methods using them are called *quasi-Monte-Carlo* (QMC) methods. It is interesting that for many integration problems QMC have asymptotically faster rates of convergence than standard MC methods [87]. Note however, that some of the variance reduction techniques applicable to standard MC methods can not be applied to QMC methods, like Russian roulette (Section 4.2.3). Mitchell [81] investigated how much better than random sampling stratified sampling strategies are. Intuitively, the smoother a function is the higher the benefit of using stratification is. For high-dimensional integrands however, the improvements of stratified sampling are minimal. Other methods put minimum distance constraints between samples to produce blue noise patterns (Figure 3.1c). A simple but expensive ways to approximate such a distribution is with rejection sampling or best-candidate sampling [80].

3.3.2 Adaptive Sampling

Ensuring uniformity of the sampling on a subset of the dimensions is often not desirable since it is blind to the integrand that is sampled. Figure 3.2 illustrates how distributing samples according to the signal that is integrated can be beneficial. Thus, better results with the same amount of samples can be achieved by concentrating the sampling efforts in regions where they pay off most. In order to do so the integration domain is subdivided into smaller regions, the sampling density of each subregion is then determined according to either analytical properties of the subregion or empirical information that is gathered during sampling [115]. In the later case the full sampling domain is initially sampled in a non-adaptive way. During this initial sampling, empirical information about each subregion is gathered and used to control how densely each subregion should be sampled after this initial phase [79]. For instance, one could compute the sample variance across each subregion to measure how complex the integrand locally is, and then distribute additional samples accordingly. In Section 4.5 we will discuss in more details how adaptive sampling techniques are applied in the context of Monte-Carlo rendering.

3.3.3 Importance Sampling

So far we have not discussed about the PDF of the samples. As it turn out, drawing samples from "good" distributions is a key in reducing variance of the Monte-Carlo Estimator. Examining Equation 3.15 reveals that the best possible PDF for sampling is proportional to the integrand. That is $p(X_i) = cf(X_i)$. In order to integrate to one over the integration domain the normalization factor c must be equal to $1/\int_D f(x)dx$. Given this sampling distribution we can show that the estimator has zero variance:

$$V[F_N] = \left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{cf(X_i)}\right] = V\left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{c}\right] = 0. \tag{3.19}$$

Of course this is of limited use in practice since we can only construct such a ideal sampling distribution if we know the integrand, which we do not. However, this example gives us an intuition of why it is good for a sampling distribution to be similar to the integrand. Luckily, simplifications of $f(x)$ can often still be obtained by factoring out some of the more complicated components. For instance, if the integrand can be written in the form $f(x) = g_1(x)g_2(x)$ and $g_1(x)$ is a function that we know and that can be sampled easily, then sampling according to $p(X) = g_1(X)/\int_D g_1(x)dx$ can in most cases reduce variance significantly. Importance sampling is similar to adaptive sampling in the sense that it optimizes sampling distributions with respect to the integrand. Note that importance sampling can lead to much higher variance than uniform sampling if used unwisely. For instance, if some regions of the integrand that have very large values also have a very low probability of being sampled according to $p(x)$, then samples in those regions will become huge because of simultaneous large numerators and small denominators. Such huge outlines can increase the variance dramatically, even if $p(x)$ matches the integrand in all other regions of the integrand perfectly.

3.3.4 Multiple Importance Sampling

One disadvantage of importance sampling is that we have to decide which sampling strategy we want to use, even though often good sampling strategies for several components of the integrand are available. For instance lets assume a integrand is of the form $f(x) = g_1(x)g_2(x)$ where good sampling strategies are known for both factors, but none can be constructed for the product of the factors. Assuming one half of the integration domain is nicely approximated by g_1 but badly by g_2, and the other half is nicely approximated by g_2 and badly by g_1. Regardless of which of both strategies we use we will always sample one half of the integration domain badly.

The naive way of combining different sampling strategies is to define a new estimator that combines estimators with T the different strategies, that is

$$\hat{F}_N = \sum_{k=1}^{T}\frac{\lambda_k}{N_k}\sum_{i=1}^{N_k}\frac{f(X_{i,k})}{p_k(X_{i,k})}, \tag{3.20}$$

with $\sum_{k=1}^{T} N_k = N$ and $\sum_{k=1}^{T}\lambda_k = 1$. Note that $X_{i,k}$ denotes the i-th sample that has been drawn with strategy k. Unfortunately the formulation above is ineffective at reducing variance since

$$Var\left[\hat{F}_N\right] = \sum_{k=1}^{T} \frac{\lambda_k^2}{N_k^2} Var\left[\sum_{i=1}^{N_k} \frac{f(X_{i,k})}{p_k(X_{i,k})}\right], \tag{3.21}$$

which states that the variance of the strategies is additive. A much better way of combining different sampling strategies is with the *multiple importance sampling* (MIS) estimator introduced by Veach [106], that is a generalization of Equation 3.20:

$$F_N^\star = \sum_{k=1}^{T} \frac{\lambda_k}{N_k} \sum_{i=1}^{N_k} \frac{w_k(X_{i,k}) f(X_{i,k})}{p_k(X_{i,k})}. \tag{3.22}$$

We simply added additional weights $w_k(X_{i,k})$ for every sample drawn with every strategy. The estimator $F_n^\star$ is unbiased if:

- The weights w_k of a specific sample X_i over all strategies sum up to one, i.e. $\sum_{k=1}^{T} w_k(X_{i,k}) = 1$,
- and if the weights w_k are never zero for parts integral domain that are sampled with probability bigger than zero, i.e. $p_k(X_{i,k}) > 0 \Rightarrow w_k(X_{i,k}) > 0$.

The most commonly used weights that fulfil these requirements and do a provable good job at reducing the variance have been first presented by Veach [108] and are of the form:

$$w_k(X_i, k) = \frac{p_k(X_{i,k})^\beta}{\sum_{j=1}^{T} p_j(X_{i,k})^\beta}. \tag{3.23}$$

with β being any non-negative exponent. If $\beta = 1$ we call this estimator the *balance heuristic* and if $\beta = 2$ we call it the *power heuristic*. On an intuitive level the reason why the power and balance heuristic work so well is that by design, the weight will always reflect how good a certain sampling strategy is at generating a specific sample compared to all other available sampling strategies. If the currently used strategy is inferior to at least one or several other strategies, then the denominator will be much bigger than the numerator and thus the weight will be low. On the other hand if one strategy is superior to all others, the denominator will be only marginally bigger than the numerator, thus the weight will be close to one. In summary, this weighting scheme effectively weights down bad sampling strategies and consequentially suppresses outliers generated by them.

3.3.5 Control Variates

Similar to importance sampling, *control variates* is a method that reduces variance of an estimator of an unknown quantity by exploiting information of similar known quantities. Given a function f that we want to integrate and a similar function g whose integral is known $\int_\Omega g(x)dx = \mu$, then we can reformulate the integral to

$$\int \tilde{f}(x)dx = \mu + \int_\Omega f(x) - g(x)dx. \tag{3.24}$$

The corresponding Monte-Carlo estimator is then

$$\tilde{F}_N = \mu + \frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i) - g(X_i)}{p(X_i)}. \tag{3.25}$$

The estimator $\tilde{F}_N$ has a lower variance than F_N when

$$V\left[\frac{f(X) - g(X)}{p(X)}\right] < V\left[\frac{f(X)}{p(X)}\right]$$

This tends to be true whenever g and f are strongly correlated. Examining the equation above we see that a function g that is already used for importance sampling, that is replacing $p(x)$ by $g(X)/\mu$, will not benefit at all from being used as control variate too since

$$V\left[\frac{f(X) - g(X)}{g(X)/\mu}\right] = V\left[\frac{f(X)}{g(X)/\mu}\right].$$

Thus one must decide whether to use a function g that is similar to f either for control variate or for importance sampling. It can be shown that, as a rule of thumb, importance sampling is preferable whenever f/g is nearly constant, and that control variate is preferable whenever $f - g$ is nearly constant [51]. A general advantage of control variates compared to importance sampling is the greater generality. For importance sampling we must consider how practical it is to draw samples from a distribution with the shape of g. Since usually the inversion method is used for this purpose, this boils down to requiring the inverse of the CDF of g to be computable.

Control variates have not been used widely in rendering since constructing a function whose expected value can be sufficiently closely approximated and that captures all relevant details of the integral are difficult to obtain. Nevertheless, there has been some research in this direction. First attempts have been performed by Lafortune and Williems [63] where they used a constant ambient term as control variates and later a piecewise constant function based on cached incident radiance in a 5D tree [64]. Later, Szirmay-Kalos et al. [104] proposed to use the solution of a radiosity pass as control variate of a second Monte-Carlo pass. Further, Scécsi et al. [103] proposed a weighting scheme to combine importance sampling with control variates, Fan et al. [24] introduced a generalization of multiple importance sampling that allows to combine of several control variates and Clarberg and Akenine-Möller [9] included visibility information in the used control variates.

3.3.6 Correlated Sampling

Comparing two unknown quantities that only differ slightly, for instance by simulating the same process with slightly changed parameters, can be very useful to get a deeper understanding of the simulated process. However if those processes are integrals that must be approximated numerically with Monte-Carlo, computing the differences between two similar instances can be quite inefficient. Given a difference of two integrals that should be computed

$$\int_{\Omega_i} f(x)dx - \int_{\Omega_j} g(y)dy,$$

then the classic Monte-Carlo estimator yields

$$F_N - G_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} - \frac{1}{N}\sum_{j=1}^{N}\frac{g(Y_j)}{p(Y_j)}. \tag{3.26}$$

the variance of this expression is

$$V[F_N - G_N] = V\left[\frac{f(X)}{p(X)}\right] + V\left[\frac{g(Y)}{p(Y)}\right] - 2\mathrm{Cov}\left[\frac{f(X)}{p(X)}, \frac{g(Y)}{p(Y)}\right]. \tag{3.27}$$

If g and f are computed in independent fashion, for instance with separate simulations using different random variables, then the correlation is zero. This means that the variance of the difference will be the additive variance of the processes. Since the differences often tend to be smaller than the subtrahends in magnitude, the signal to noise ratio of the difference will be much higher than the signal to noise ratio of the estimates of the individual subtrahends.

However, if the subtrahends are sampled in such a way that there is a positive covariance between them, then the variance will be lower since noise gets cancelled out due to the last term in Equation 3.27. One can exploit this by simply sharing the random variables of the Monte-Carlo simulation between the two integrals, that is

$$F_N - G_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i) - g(X_i)}{p(X_i)}. \tag{3.28}$$

Sampling two independent integrals in such a correlated fashion is called *correlated sampling* [1] or the usage of *common random numbers*. If the integrals are similar enough, this will lead to high correlation between the subtrahends and some variance will be cancelled out. Note that this formulation is very similar to the control variates estimator (Equation 3.25). Both methods eliminate variance by exploiting correlation between two functions. However there are two key differences:

- First, g is not a simplified version of f but its own quantity that just happens to be similar to f. Even the integration domain of f and g can be different.

- Second, the correction term μ is not used. This means that g itself can be a unknown quantity that must be sampled.

These differences show that the use-cases of both methods are quite different, control variates is used for noise reduction of an integral given a suitable a-priori known approximation function, while correlated sampling provides a way to sample differences of similar and potentially unknown integrals more efficiently. Correlated sampling is one of the foundations of the gradient-domain rendering methods presented in Chapter 5.

[1] Even though our terminology clearly makes a difference between correlated sampling and control variates, both terms are often used as synonyms in literature for describing methods that exploit correlation between two sampled quantities.

3.4 Metropolis Sampling

3.4.1 Motivation

Recall that with importance sampling the goal is to distribute samples according to a distribution that has a shape similar to the integrand. But such similar functions must be integrable and sampleable. Unfortunately, these requirements are not always easy to meet. For instance, importance sampling a high dimensional function like the path space formulation of light transport directly is nearly impossible. It is unclear how one could compute and sample PDFs that cover such a complex function adequately over its entire integration domain efficiently, even with the usage of MIS. In rendering a way to side-step this, is to importance sample only a small subset of all dimensions of the sampling space at once. However, this does not lead to optimal solutions.

Metropolis sampling [78, 37] is a Monte-Carlo integration technique that allows to importance sample arbitrary non-negative functions. It is neither required to integrate the function or to invert its CDF in order to sample accordingly to any specific distribution. The key idea is to develop an algorithm that generates in the limit a set of samples with a distribution that has the *exact* same shape as the integrand. Given enough samples, the unknown integral can be computed directly from the - appropriately normalized - histogram of the generated samples. In the next chapter a light transport method that is based on this sampling method will be presented that allows to importance sample the path space directly.

3.4.2 Markov-Chains

Metropolis sampling generates a sequence of samples $S = X_0, ...X_n$ from the integration domain D where each sample X_i depends on the previous sample X_{i-1} but not on any other preceding states. Such a sequence of samples is called a *Markov-chain.* The probability to transition from any state X_{i-1} to a state X_i is the *transition probability* $K(X_{i-1} \rightarrow X_i)$. Note that sequences generated this way will lead to correlation between the samples. Also stratification techniques to reduce variance cannot be applied any more [90]. If any state $Y \in D$ can be reached after a finite number of transitions from any other state $X \in D$ with a non-negative probability, then the chain is called *ergodic.* One can show that the probability distribution of samples generated by an ergodic Markov-Chain will converge towards a specific distribution in the limit, the *stationary distribution*, regardless of the initial sample X_0.

3.4.3 Metropolis-Hastings Algorithm

In physical systems the stationary distribution is usually determined by the transition probabilities of the system, and the system will evolve towards it over time. The Metropolis-Hastings algorithm [37] reverses these dependencies: given a desired stationary distribution, it constructs the transition probabilities of the system in such a way that the system converges towards the desired stationary distribution.

We construct such a system by introducing the concept of proposing a candidate for the next sample that can either be accepted or rejected. The *tentative transition probability* $T(X_{i-1} \rightarrow X_i')$

describes the probability of a sample X_i' to be proposed as successor of X_{i-1}. Whether or not this successor is accepted is determined by the *acceptance probability* $a(X_{i-1} \to X_i')$. Note that the transition probability $K(X_{i-1} \to X_i)$ from the last section is now decomposed in two components $T(X_{i-1} \to X_i)a(X_{i-1} \to X_i)$. The successor of sample X_{i-1} is defined as follows:

$$X_i = \begin{cases} X_i' & \text{with probability } a(X_{i-1} \to X_i'), \\ X_{i-1} & \text{otherwise.} \end{cases}$$

Asides from being ergodic, no constraints are put on $T(X_{i-1} \to X_i)$. The tentative transition function is therefore a free parameter of the method. Instead, the stationary distribution is controlled by the acceptance probability. The choice of the acceptance probability in the Metropolis-Hastings algorithm is justified in the following way: assume that the desired stationary distribution is already reached, that is $p \propto f$, then our distribution should not change any more. This is true when the following equation is fulfilled for any samples X and X':

$$f(X)T(X \to X')a(X \to X') = f(X')T(X' \to X)a(X' \to X).$$

One can show that this is fulfilled with the following choice of a:

$$a(X \to X') = \min\left(1, \frac{f(X')T(X' \to X)}{f(X)T(X \to X')}\right) \tag{3.29}$$

This choice also provides the fastest convergence rate towards the desired distribution $p \propto f$. Note that if the tentative transition probability is symmetric, that is $T(X' \to X) = T(X \to X')$, then the acceptance probability can be simplified to

$$a(X \to X') = \min\left(1, \frac{f(X')}{f(X)}\right). \tag{3.30}$$

This equation states that proposed samples with $f(X') \geq f(X)$ will always be accepted, while proposed samples with $f(X') < f(X)$ might be rejected with a probability proportional to $f(X')/f(X)$. These acceptance probabilities will lead to a sample distribution that is proportional to the function f.

3.4.4 Mutation Strategies

A way to realise the tentative transition function is by specifying a set of *mutation strategies*, with each of them designed to be good at one specific thing. Choosing a tentative sample then becomes a two stage approach. First, a mutation strategy is selected according to some probability and then the current sample is randomly mutated according to a set of rules specified by the selected mutation strategy.

How fast the stationary distribution is reached largely depends on the choice of the tentative transition function, and thus of the mutation strategies. A good set of mutation strategies should be efficient at fulfilling two objectives:

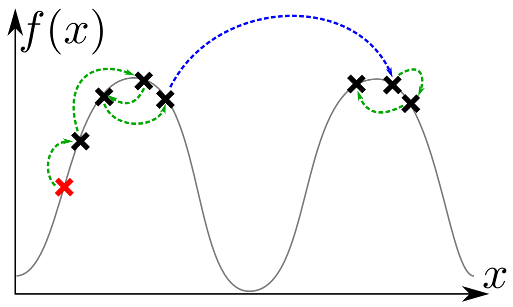


Figure 3.3: MLT mutation strategies. A sequence of samples taken from the function f using MLT starting at the red sample. Local mutation strategies are visualized as green arrows and global strategies as blue arrows. Note that the function f consists of two regions separated by a very low valued gap. The chance of crossing this gap with local strategies alone is minimal.

- they should explore the region around a local maxima thoroughly once one has been found,
- and they should allow to transition from one local maxima to another even if they are separated by very low valued regions.

The first objective ensures that many samples are drawn in a region of interest as soon as one has been found. Note that this property is crucial for having a high acceptance probability. For instance, if a sample X is close to a local maxima of f and thus has a large value $f(X)$, then it is likely that close-by samples X' will also have large values $f(X')$. The acceptance probability $a(X \to X')$ will thus not be too small when we have mutation strategies that are able to sample close-by points with high probability. On the other hand without mutation strategies that sample close-by points, it can become very unlikely to find new samples that also lead to large values. The Markov-Chain would be stuck at X for a long time and consequentially the integration domain would not be explored efficiently.

The second objective ensures that the Markov-chain is able to jump across local minima of f between different regions of interest. In order to fulfil both objectives, *local* mutation strategies and *global* mutation strategies are combined (see Figure 3.3). The local mutation strategies draw tentative samples that are close-by the current sample. It is easy to see that such strategies are good at fulfilling the first objective. The global mutation strategies on the other hand allow a tentative sample to be anywhere in the integration domain, independently of the current sample. The simplest such strategy is one that completely ignores the current sample and just randomly samples the integration domain. This allows the Markov-chain to jump into totally new areas of the integration domain and thus find new local maxima.

Chapter 4

Rendering Algorithms

In this chapter we discuss algorithms that solve the light transport problem using Monte-Carlo sampling to create an image. Often these methods are referred to as *physically-based rendering* algorithms. These algorithms find light paths that connect the sensor with the emitters by sampling the path space with Monte-Carlo methods. The path samples are then evaluated and used to create a final image.

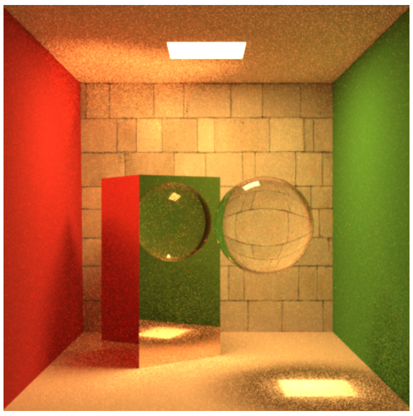
We will describe three well known path sampling algorithms that will be extended in the remainder of this thesis to gradient-domain rendering algorithms. We will start with describing *path tracing* that was the first full solution to the light transport problem [50] and still is the most widely used one due to its simplicity. This method was introduced as a solution to the measurement function (Equation 2.11) and samples paths in a recursive manner, starting from the sensor until an emitter is eventually hit. The two other algorithms use sampling strategies that are more efficient at sampling paths in complex illumination setups. The first of these two methods, *bidirectional path tracing* [62, 107], constructs paths from both directions, that is from the emitters to the sensor and from the sensor to the emitters, and combines different parts of these paths to create a large number of new paths at low cost. The last method discussed here is *Metropolis light transport* [109] that uses Metropolis sampling (Section 3.4) to directly importance sample the path-space and thus provides an efficient sampling solution for very difficult lighting situations where other methods fail. Figure 4.1 shows a motivational example of how these different path sampling techniques affect the obtained results at equal computation time.

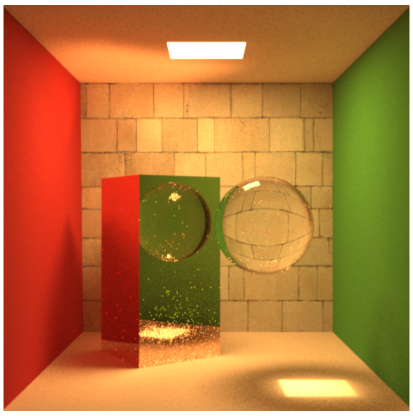
We conclude this chapter with a brief discussion of a set of recent techniques that reduce the variance of the resulting image through adaptive sampling and reconstruction.

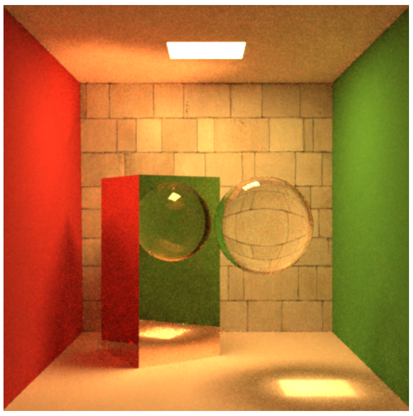
4.1 Formulation of the Problem

In order to render images in a physically-based way, we want to compute the Monte-Carlo estimator of Equation 2.24 in a way that is as efficient as possible. Formally, we want to compute

$$I_p \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f_p^\star(\overline{\mathbf{x}}_i)}{p(\overline{\mathbf{x}}_i)} \tag{4.1}$$


(a) Path tracing


(b) Bidirectional path tracing


(c) Metropolis light transport

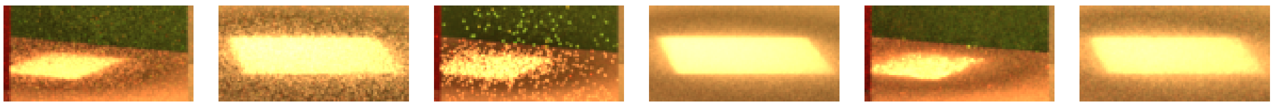


Figure 4.1: Comparison of different path sampling methods. The scene consists of a glass sphere and a reflective metal box. The sphere produces a caustic on the floor that is also mirrored on the metal box. Each image took 35 seconds to render. The images where rendered in Mitsuba [43].

for every pixel p of the image. Recall that the measurement contribution function $f_p^\star$ is computed with Equation 2.22 regardless of how $\overline{\mathbf{x}}$ was sampled. In the equation above it is often useful to separate pixel dependent quantities from pixel independent quantities. Since the only pixel dependent component is the importance function W_p, we can decompose Equation 4.1 into

$$I_p \approx \frac{1}{N}\sum_{i=1}^{N} W_p(\overline{\mathbf{x}}_i)\frac{f^\star(\overline{\mathbf{x}}_i)}{p(\overline{\mathbf{x}}_i)},$$

where we call $f^\star(\overline{\mathbf{x}})$ the *throughput* of $\overline{\mathbf{x}}$.

This allows a clear separation of the rendering algorithms into sampling and reconstruction. One step is sampling $f^\star$ and the other one is reconstructing pixel values I_p from this data by using a filter W_p. We will discuss the implications of this separation in Sections 4.2.1 and 4.5. Our interest for now lies in the sampling step. The main challenge there lies in finding sampling techniques for $\overline{\mathbf{x}}$ that lead to small variance of $f^\star/p$ and whose PDF can be computed efficiently. In the following we will discuss three ways of doing this.

4.2 Path Tracing

Kajiya introduced *random walks* into computer graphics [50]. He showed that light paths can be sampled effectively by incrementally constructing a path as a sequence of random decisions. In classical path tracing, paths are computed in the reversed direction of light transport, this means they start at the sensor and are then incrementally constructed backwards until a light source is eventually hit. For this reason path tracing is also often referred to as *backward tracing*. Specifically, for every pixel p paths are constructed with the following procedure:

1. Sample an initial position x_0 on the sensor.
2. Sample a new outgoing direction ω_i on the current vertex x_i by using any suitable sampling strategy.
3. Evaluate the ray casting function (Equation 2.9) at x_i and set the next vertex to be $x_{i+1} = \nu(x_i, \omega_i)$.
4. Repeat step 2 and 3 until a termination criterion is met.

The main degree of freedom in this procedure is how to sample the directions ω. We will discuss suitable sampling strategies in the next section. Note however, that we can easily control to which pixel or set of pixels a path contributes by sampling the initial direction ω_0 and position on the sensor x_0 such that the importance function (Section 2.3) of the currently processed pixel p will not be zero, i.e. $W_p(\overline{\mathbf{x}}) = W_p(x_0, \omega_0) > 0$. This is a form of stratification and will not lead to bias. By using this stratification, parallelism of path tracing is greatly simplified. Instead of computing the value of every pixel of an image sequentially, we can tile the image and compute each tile in a separate thread that samples only paths that might contribute to this tile.

In practice during generation of the path its throughput $f^\star(\overline{\mathbf{x}})$ and its PDF $p(\overline{\mathbf{x}})$ are also computed incrementally. In order to do so we define two variables that are incrementally updated during sampling. For the throughput this variable is defined recursively as:

$$\begin{aligned} \alpha_0 &= 1 \\ \alpha_i &= \alpha_{i-1} f(x_{i+1}, x_i, x_{i-1}), \end{aligned} \tag{4.2}$$

and for the PDF it is:

$$\begin{aligned} \beta_0 &= p_0^W(x_0) p_1^W(\omega) \\ \beta_i &= \beta_{i-1} p_{\sigma^\perp}(x_{i+1}, x_i, x_{i-1}). \end{aligned} \tag{4.3}$$

p_0^W is the probability of selecting a certain position on the sensor's aperture and p_1^W is the probability of selecting a certain initial direction. $p_{\sigma^\perp}$ is the PDF of a scattering event with respect to projected solid angle. Note that because Equation 2.11 is described in terms of area measure and because we implicitly perform a change of variables with a Jacobian determinant of $|\partial P_A / \partial p_{\sigma^\perp}| = 1/G(x_i, x_{i+1})$ at each vertex, the geometry terms are missing in α. If the last vertex x_n is on an emitter then the contribution of the path can be computed as

$$\frac{f^\star(\overline{\mathbf{x}})}{p(\overline{\mathbf{x}})} = \frac{\alpha_{n-1}}{\beta_{n-1}} L_e(x_n, x_{n-1}). \tag{4.4}$$

4.2.1 Reconstruction Filter

Let $\Omega_p = \{\overline{\mathbf{x}} \in \Omega : W_p(\overline{\mathbf{x}}) > 0\}$ denote the set of paths whose importance function for pixel p is non-zero. In general to avoid aliasing artefacts Ω_p and Ω_q may overlap if p and q are close-by in image space. Examples of such overlapping filters are Gaussians or Mitchell-Netravali filters [82]. We refer the reader for an in-depth discussion of the aliasing artefacts and how different reconstruction filters can mitigate them to the excellent book by Pharr and Humphreys [90].

Any path $\overline{\mathbf{x}}$ will thus contribute to several pixels. For efficiency reasons any sampled path $\overline{\mathbf{x}}$ should therefore be evaluated for all pixels it contributes to. This is usually done with a weighted average filter

$$I_p = \frac{1}{\sum_i W_p(\overline{\mathbf{x}}_i)} \sum_i W_p(\overline{\mathbf{x}}_i) f^\star(\overline{\mathbf{x}}_i)/p(\overline{\mathbf{x}}_i).$$

In practice, this sum can be computed efficiently as $I_p = C_p/W_p$ by incrementally updating a color buffer C and a weight buffer W where each time a sample contributes to p the buffers are updated by $C_p += W_p(\overline{\mathbf{x}}_i) f^\star(\overline{\mathbf{x}}_i)/p(\overline{\mathbf{x}}_i)$ and $W += W_p(\overline{\mathbf{x}}_i)$. Since we usually know in advance the importance function of our sensors[1] and since the importance functions usually have a small region of support in image space, we can efficiently evaluate the contribution of each sample $\overline{\mathbf{x}}$ for *all* pixels to which it might contribute.

4.2.2 Sampling Techniques

Randomly sampling each direction during construction of the path will generally lead to unacceptably high variance due to reasons that are explained in the next paragraphs. Importance sampling strategies are an efficient way to reduce variance. Different importance sampling strategies can be combined with a MIS estimator (Section 3.3.4) to further reduce variance.

BSDF sampling Surfaces often have glossy BSDFs. This means that these surfaces only reflect light incoming from a specific narrow cone of directions (recall Figure 2.4). If directions are sampled randomly only a small fraction of samples will lie in this cone which leads to high variance. Luckily, designing an importance strategy for this is relatively easy since the BSDFs are known in advance. Thus, at each scattering event we can importance sample the direction according to the BSDF at the current location.

Light Sampling Another reason for high variance is due to the fact that a path only carries light if it connects an emitter with the sensor. With traditional path sampling this requires a path to randomly hit such an emitter at some point during its incremental construction. Unfortunately this can become very unlikely if the emitters in the scene are very small or far away. A sampling strategy to ensure that lights get sampled, is to sample a point on the surface of any light source in the scene and to set this point as the next vertex of the path. Note that sampling a vertex on an emitter happens over area measure in contrast to projected solid angle measure that is used for the direction sampling in path tracing. This change of variables must be taken into account when computing the paths contribution, which means that the last geometry term $G(x_{n-1}, x_n)$ gets not cancelled out because of the Jacobian determinant $|\partial P_A / \partial p_{\sigma^\perp}|$ in Equation 4.4. Note that light sampling

[1]This is not true for some of the methods discussed in Section 4.5, since there the reconstruction filter is not known in advance. There however, this problem can be avoided by separating the reconstruction filter into two parts. First, an a priori known uniform image filter is used to create an image buffer. Then a second adaptive reconstruction filter is applied on this image buffer to create a final image. Only the first component of the filter is used during accumulation of the color values with the weighted average filter.

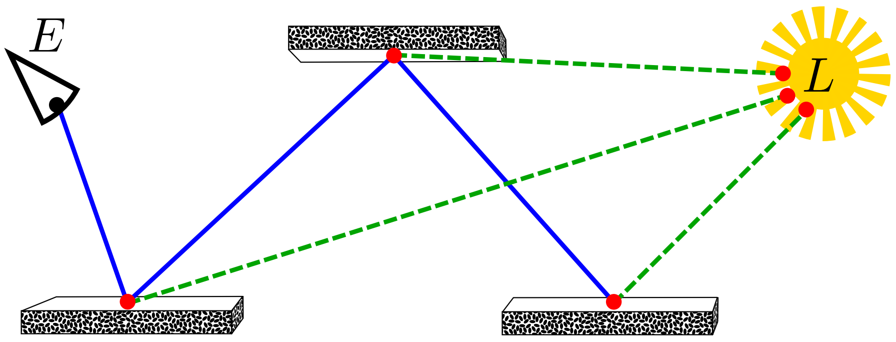


Figure 4.2: Next Event Estimation. Black is the main path. The red dots are the next events computed using area sampling and the green lines are the connections of partial paths to full paths with NEE.

cannot be applied on (perfectly) specular vertices, since the probability of sampling the reflection or refraction direction with area sampling is zero. Also note that the simple strategy described here ignores visibility, thus a sampled vertex on an emitter can still be occluded as seen from the previous vertex in the path. However, more sophisticated methods importance sample the locations of emitters with respect to the current vertex [97].

Path Tracing with Next Event Estimation This method describes an algorithm that generates a set of correlated paths out of each path sample. With naive path tracing, one single path $\overline{\mathbf{x}} = x_0, ...x_n$ is generated at a time. We will refer to this path in the following as the *main path*. *Next event estimation* (NEE) attempts to create one or more paths for each subpath of $\overline{\mathbf{x}}$ with $\text{sub}(\overline{\mathbf{x}})_m = x_0, ...x_m$ and $m < n$ that connect the sensor to one of the emitters. This is done by performing at each vertex x_k along the main path light sampling to create one or more extra paths of the form $x_0, ...x_k, y_e = \text{sub}(\overline{\mathbf{x}})_k y_e$ where y_e is on an emitter and $0 < k < m$ (see Figure 4.2). This process leads to pn paths $\text{sub}(\overline{\mathbf{x}})_k y_e$ for every main path $\overline{\mathbf{x}}$, where p is the number of light samples generated at each vertex of $\overline{\mathbf{x}}$. Note that any sampling strategy can be used to create the main path, for instance BSDF sampling. For all paths that are generated by this process, including the main path and any subpath of it, we then compute a MIS estimator that combines all used sampling strategies.

4.2.3 Russian Roulette and Path Splitting

One subtle problem that we ignored so far is that Equation 4.1 samples over paths of unbound length. This means infinitely long paths should also be considered and of course we cannot sample those. *Russian roulette* (RR) is a simple way to reduce the probability of sampling such infinitely long paths to infinitely small values while still producing unbiased results. In general, Russian roulette describes a method to *not* evaluate samples with a certain probability α. In order to still get an unbiased result, samples X that are not evaluated with a probability of α must be weighted by a correction factor $1/(1-\alpha)$ since $E[X] = (1-\alpha)E[X/(1-\alpha)]$. First introduced in the context of path sampling by Arvo and Kirk [1], Russian roulette can be used to terminate the incremental path construction with a probability of α. Specifically, given a path of length n with $\overline{\mathbf{x}} = x_0, ..., x_{n-1}$ a new sample of length $n+1$ with $\overline{\mathbf{y}} = x_0, ..., x_{n-1}, y_n$ will only be sampled and evaluated with a probability of $1-\alpha$. If the sample $\overline{\mathbf{y}}$ gets evaluated, $f^\star(\overline{\mathbf{y}})/p(\overline{\mathbf{y}})$ is weighted by $1/(1-\alpha)$. The termination criterion of α can vary for every sample and as long as $\alpha > 0$ for all paths with a length bigger than a certain finite number, then the probability of sampling infinitely long paths goes to zero. Note that Russian roulette generally increases noise from long paths since less samples are used to sample them.

A related method to Russian roulette also introduced by Arvo and Kirk [1] that helps increasing the sampling rates of important regions of path space is *path splitting.* With a certain probability β a subpath $\overline{\mathbf{x}} = x_0, ..., x_{n-1}$ is split into k new paths $\overline{\mathbf{y}}_1 = x_0, ..., x_{n-1}, y_{n,1}$ to $\overline{\mathbf{y}}_k = x_0, ..., x_{n-1}, y_n, k$. Distributed ray tracing [13] can be seen as an instance of path splitting. Russian roulette and path splitting together can form a quite efficient adaptive sampling scheme of the path space and thus many ways of defining heuristically based splitting or termination probabilities have been proposed. The weights can be based on variance analysis of nested estimators [7] or BSDF properties and user constraints [105]. Further, Veach [106] developed efficiency-optimized Russian roulette and Vorba et al. [111] an algorithm that aims at keeping the contribution of a path roughly constant along the path by combining Russian roulette and splitting with adjoint-driven importance sampling techniques [110].

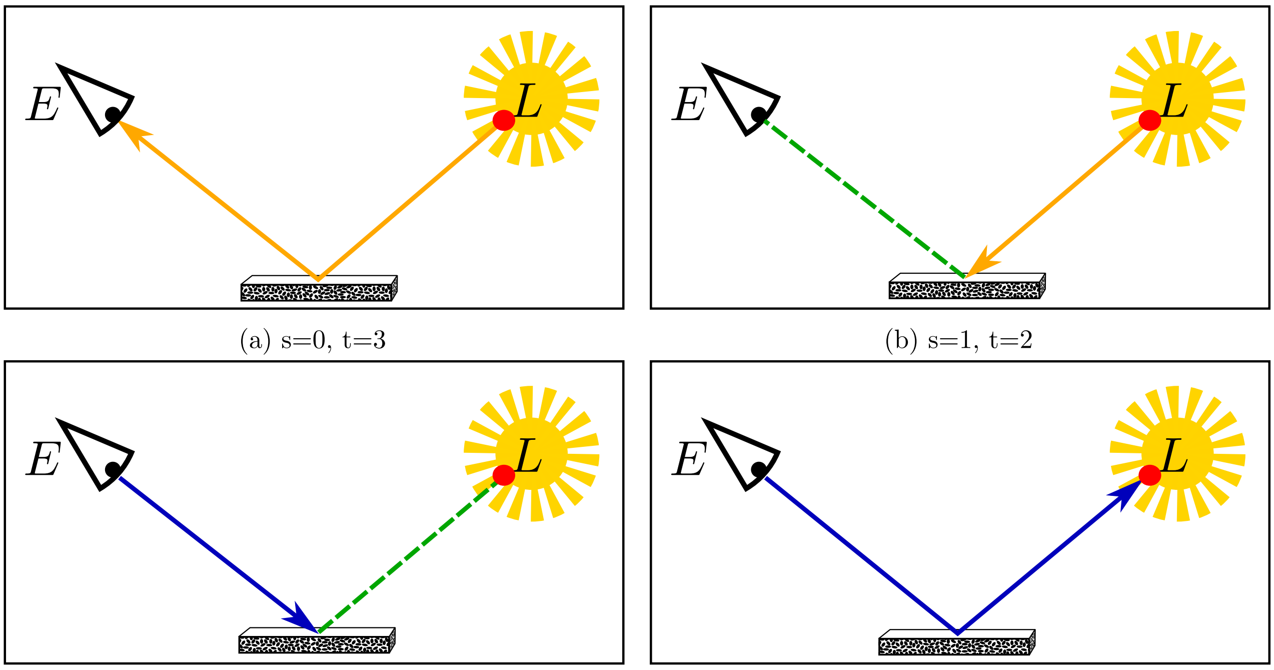
4.2.4 Limitations

The main limitation of path tracing (with or without NEE) is its inefficiency in sampling caustic paths, that is paths of type $E(D|S)^*S^+L$. Scenes in which such path configurations occur will suffer from extensive noise from these types of paths. An example is given in Figure 4.1a. Unfortunately in some scenes nearly all illumination comes from caustic paths, as for instance a room illuminated by the sun shining through a glass window or by light sources encapsulated in light bulbs. The reason for this is that NEE in its simplest form cannot be used to connect a vertex with a light source when a specular interface is in between (e.g. a glass window). Hence, in such cases path tracing with NEE degenerates to plain path tracing.

Recently, a method called *manifold next event estimation* (MNEE) extended NEE to support connections through simple transmissive specular interfaces [34]. It starts with a straight connection through the transmissive interface with zero throughput and performs a local optimization to find a close-by path in path space that has a non-zero throughput. However, this method is only effective if the nearby geometry on the specular interface is reasonably simple. Also in its current form MNEE cannot be used to perform NEE across reflective interfaces.

4.3 Bidirectional Path Tracing

A path can be constructed in more ways than it is done in path tracing. For instance instead of constructing a path from the sensor incrementally until a emitter is eventually hit, one could construct the path the reversed way round, that is from the emitter towards the sensor. This is what is done in *light tracing* [20]. *Bidirectional path tracing* (BDPT) exploits the fact that paths can be created by combining path tracing and light tracing and was developed independently by



(a) s=0, t=3 (b) s=1, t=2

(c) s=2, t=1 (d) s=3, t=0

Figure 4.3: Bidirectional Path Tracing. There are 4 ways of sampling a path of length 3 by combining path tracing with light tracing. Yellow subpaths are sampled in the radiance direction and blue subpaths in the importance direction. Green lines are connection segments. The s-t notation is according to Section 4.3.1.

Lafortune and Willems [62] and Veach and Guibas [107]. BDPT combines parts of the path traced subpath with parts of the of the light traced subpath by connecting them to new full paths. As it turns out, for a path of length k there are $k + 1$ ways of constructing it with such combined approaches (see Figure 4.3). Each of these strategies is good at computing certain types of paths and bad at computing other types of paths. All these strategies can be combined with a MIS estimator (Section 3.3.4) in a optimal way that weights down path samples that were sampled using an ill-suited sampling strategy [108].

4.3.1 MIS over Connection Strategies

BDPT combines a pair of paths consisting of one path starting from the sensor and one path starting from an emitter in all possible ways to generate a set of complete paths that connect the sensor to the emitter. That means, given a *sensor subpath* $\overline{\mathbf{y}} = y_0, ..., y_n$ where y_0 is on the sensor and an *emitter subpath* $\overline{\mathbf{z}} = z_m, ..., z_0$ where z_0 is on one of the emitters, then we compute all possible combinations from this two paths of the form

$$\overline{\mathbf{x}}_{s,t} = y_0, ...y_{s-1}, z_{t-1}, ..., z_0$$

where $0 \leq s \leq n$ and $0 \leq t \leq m$. Note that as a slight abuse of notation we set $\overline{\mathbf{x}}_{0,t} = \overline{\mathbf{z}}$ and $\overline{\mathbf{x}}_{s,0} = \overline{\mathbf{y}}$. We call all these different ways of creating the paths *s-t-connection strategies* and we call y_{s-1} and z_{t-1} the *connection vertices*. The resulting set of paths then consists of $n + m + 1$ different paths with each of them using a different connection strategy.

Every path in this set that was computed with a specific s,t-connection strategy could have been sampled from other sensor and emitter sub paths by using another s,t-connection strategy. It is easy to show that any path of the form $\overline{\mathbf{x}}_{s,t} = y_0, ...y_{s-1}, z_{t-1}, ..., z_0$ can be sampled by up to $s+t+1$ other connection strategy with $s' = k$ and $t' = (s+t) - k$ where $0 \leq k \leq s+t$ [2]. Each of these strategies will compute the same path with a different PDF. We can construct a MIS estimator with the power heuristic over the sampling strategies over all these samples:

$$\sum_{s=0}^{s\leq n}\sum_{t=0}^{t\leq m}\underbrace{\sum_{k=0}^{i\leq s+t}\frac{p_{s,t}(\overline{\mathbf{x}}_{s,t})^2}{p_{k,(s+t)-k}(\overline{\mathbf{x}}_{s,t})^2}}_{\omega_{s,t}(\overline{\mathbf{x}}_{s,t})}\frac{f^\star(\overline{\mathbf{x}}_{s,t})}{p_{s,t}(\overline{\mathbf{x}}_{s,t})}. \tag{4.5}$$

All components in this equation can be computed very efficiently by caching some additional data for every sampled vertex. We refer to the excellent thesis by Veach for the implementation details [106].

4.3.2 Direct Light Paths

Direct light paths are paths sampled by strategies that directly connect an emitter subpath to a point on a sensor or that are emitter subpaths that randomly hit the sensor. These are paths generated by connection strategies with $s \leq 1$. Note that connection strategies with $s = 0$ are only applicable when the sensor is not of infinitesimally small extent (i.e. it is not a pinhole camera). These types of paths are extremely important to sample caustic paths of the form $E(S|D)S^+L$ since the only other strategies that can produce them are the ones where the sensor subpath randomly hits the emitter (i.e. $t = 0$). However, we need to take special care of direct light paths since we cannot effectively control to which pixel(s) they might contribute. This means with $s \leq 1$ we do not know a priori to which pixels p the paths measurement contribution $f_p^\star(\overline{\mathbf{x}}_{s,t})$ will have a positive contribution. This has several consequences:

First, recall from Section 4.2 that path tracing allows for simple parallelization of the algorithm by evaluating image tiles in parallel processes. Since light paths cannot be generated such that we know a priori to which tile they contribute, a light path generated from any process can contribute to any location in the image. To avoid synchronization between the processes, each thread stores an own “light image”. This is an image that exclusively stores path samples with $s \leq 1$ generated by that thread. These light images are then combined after rendering is finished and added to the “eye image” that is used to store paths created with all other connection strategies.

However, we must also consider that the density of direct light paths in the image plane will be very non-uniform, which leads to the second consequence: in order to find the final color values we must incorporate this density into the final image. That means if more direct light paths land in a specific pixel than elsewhere, this pixel has to be brighter. We can ensure this by normalizing the light image by D/N where N is the total number of sampled paths and D the number of pixels. Note that this is different to what is done for the eye image where all paths that are contributing to a pixel are averaged together in some way (see Section 4.2.1).

[2]Connection strategies lead to paths with a throughput of zero when one of the connection vertices is specular. This is analogous to NEE.

4.3.3 Limitations

As with path tracing there are also paths that are notoriously hard to sample with BDPT. There are types of paths that can only be sampled by one single connection strategy, which essentially means that the MIS estimator of BDPT degenerates to an ordinary importance sampling estimator. This is called the *problem of insufficient techniques* [59]. Paths of the form $E(D|S)^*SDS(D|S)^*L$, short *SDS-paths*, are the most problematic paths for BDPT. The main problem is that the SDS part of the path can not be sampled by a deterministic connection. It must be sampled purely from one direction. If this SDS part happens to carry a lot of energy and requires a very specific configuration of the three involved vertices, then $f^\star(\overline{\mathbf{x}})/p(\overline{\mathbf{x}})$ becomes very big and we get spike noise in our image. A common example for these are caustics seen through a mirror like in the left close-up in Figure 4.1. There we can see that BDPT can even perform worse than unidirectional path tracing at equal computation time when the overhead of creating bidirectional samples does not pay off.

It turns out that also more common types of paths are problematic: for instance direct illumination seen through a mirror or glass, i.e. ES^+DL. These paths can only be sampled by connection strategies with $t \leq 1$ which essentially degenerates BDPT to path tracing with NEE. Even scenes that only consist of purely diffuse materials can be problematic when the light from the emitter needs to follow a very specific path to reach the sensor. A simple example for this is a scene setup with two rooms: the sensor is located in one of the rooms and the light source in the other one. If light can pass from one room to the other only through a thin crack in the wall, then the BDPT connection strategies will almost always fail due to occlusion. The only other way for a sensor- and emitter subpath to connect, is when the unlikely case happens that one of them passes through the crack. A famous example of such a scene is the DOOR scene used in the next chapter in Figures 5.11 and 5.13.

All these examples have in common that they have high energy regions in path space that are strongly localized and that cannot be sampled effectively by deterministically connecting parts of the paths, be it because of the material properties or because of the geometry setup.

Mitigating the problem of sampling SDS paths with BDPT has been achieved with *vertex merging* that combines BDPT with photon mapping [28, 33]. Solving the path space integral for certain paths more efficiently can also be achieved by reusing the light subpaths for multiple eye subpaths [91].

4.4 Metropolis Light Transport

In Section 3.3.3 we discussed that importance sampling is an effective strategy to reduce variance that essentially tries to find PDF $p(\overline{\mathbf{x}})$ that is as similar as possible to $f^\star$, and from which we can draw samples. Unfortunately, finding such a function turns out to be very challenging since the function $f^\star$ is high-dimensional, complex and unpredictable. Therefore, selecting good samples from the high dimensional path space directly is out of reach. Instead, methods like path tracing or BDPT accept suboptimal sampling strategies that rely on local information only. However, this is also the fundamental limitation of these approaches since there is no guarantee that the local strategies or any combination of them is powerful enough to handle all possible path configurations well.

A more principled solution was developed by Veach and Guidas [109], where they realized that

Metropolis sampling (Section 3.4) can be used to importance sample arbitrary path spaces directly. The main idea of the algorithm is to use Metropolis sampling to distribute samples proportionally to $f^\star$, i.e. to sample according to $p(\overline{\mathbf{x}}) = f^\star(\overline{\mathbf{x}})/b$ where $b = \int_\Omega f^\star(\overline{\mathbf{x}}) d\mu(\overline{\mathbf{x}})$ is a normalization constant. Plugging this PDF into Equation 4.1 yields

$$I_p \approx \frac{1}{N} \sum_i W_p(\overline{\mathbf{x}}_i) \frac{f^\star(\overline{\mathbf{x}}_i)}{f^\star(\overline{\mathbf{x}}_i)/b} = \frac{b}{N} \sum_i W_p(\overline{\mathbf{x}}_i). \tag{4.6}$$

This would mean perfect importance sampling of the path space. However, $f^\star$ usually returns a spectral value, i.e. one value per color-channel that is sampled (see Section 2.1.1), while Metropolis sampling only works with scalar functions. There is no real notion of how to sample according to a multivalued function. However, in Equation 4.6 Metropolis sampling is only used as a means to perform importance sampling of the throughput function $f^\star$. The target function of the metropolis sampler does not need to be the integrand $f^\star$. It is for instance perfectly valid to perform Metropolis sampling with a target function L and to compute $f^\star$ based on samples distributed with the density of L. We can thus generalize Equation 4.6 to:

$$I_p \approx \frac{1}{N} \sum_i W_p(\overline{\mathbf{x}}_i) \frac{f^\star(\overline{\mathbf{x}}_i)}{L(\overline{\mathbf{x}}_i)/b} \tag{4.7}$$

where now $b = \int L(\overline{\mathbf{x}}) d\mu(\overline{\mathbf{x}})$. To solve the problem with the spectral sampling we can distribute samples according to the luminance of the throughput. There is of course a strong correlation between the luminance of the throughput and the values of every channel. Thus in general this is still an excellent importance sampling strategy. Note that Hoberock et al. [41] suggest that using other target functions can be beneficial in some cases.

There are still two problems left: first, we need to somehow compute b and second, we need to avoid *start-up bias*. Start-up bias describes the problem with Metropolis sampling that the desired distribution, in our case L, will only be reached in the limit, that is after sampling an infinitely long Markov-Chain. Thus, without further refinements, MLT would only be consistent and not unbiased. Both problems can be solved by performing MLT in two passes: in a first pass an arbitrary light transport algorithm is used to generate a set of initial path samples and to compute b. Veach [109] suggested to generate the initial samples by using BDPT, but any other unbiased path sampling technique would work too. The value b does not depend on the pixel, it is the integral of the throughput of all paths, $\int_P I_p dp$, where P is the entire image plane. Thus, even when using only a few samples per pixel we will typically have several hundred thousands or millions of estimates for b. In a second pass these initial samples are used to start one or more Markov-chains as described in Section 3.4. By taking some precautions that will be discussed below in Section 4.4.1, start-up bias can be avoided and thus all samples will be generated with a PDF of L/b. We evaluate Equation 4.7 for every sample $\overline{\mathbf{x}}$ by computing its normalized sensor response $bf_p^\star(\overline{\mathbf{x}})/L(\overline{\mathbf{x}})$ and then adding the value at the corresponding pixel locations p in the image. Similar to how we store pixels in PT and BDPT, we compute all normalized sensor responses for a sampled path at once, since every path contributes only to a small number of pixels.

We can further increase efficiency by exploiting the expected value of the integrand. The idea of this method is very simple: if we have a process that repeatedly samples x with an acceptance probability of a and y with an acceptance probability of $(1-a)$, then the expected value of this process is $ax+(1-a)y$. Following this argument we do not need to waste rejected samples in the Markov-chain: rejected samples can be added to the estimate after being weighted by $(1-a)$, while accepted samples are weighted by a. Algorithm 4.1 describes the full Metropolis light transport algorithm with the subtleties described in this section. The main point that we still need to discuss is how start-up bias is removed and how to design the mutation strategies.

Algorithm 4.1 Metropolis Light Transport

```
 1: Generate $Q$ seed paths $\overline{\mathbf{z}}_i$ with BDPT
 2: Compute $b = Q^{-1}\sum_i^Q f^\star(\overline{\mathbf{z}}_i)/p(\overline{\mathbf{z}}_i)$
 3: Pick initial path $\overline{\mathbf{x}}_0$ from all $z_i$
 4: for $i = 1, ..., N$ do
 5:     Compute tentative sample $\overline{\mathbf{x}}'_i$ using $T(\overline{\mathbf{x}}_{i-1} \to \overline{\mathbf{x}}'_i)$
 6:     Compute $a = \min(1, \frac{L(\overline{\mathbf{x}}'_i)}{L(\overline{\mathbf{x}}_{i-1})}\frac{T(\overline{\mathbf{x}}'_i \to \overline{\mathbf{x}}_{i-1})}{T(\overline{\mathbf{x}}_{i-1} \to \overline{\mathbf{x}}'_i)})$
 7:     for all pixels $p$ with $W_p(\overline{\mathbf{x}}'_i) > 0$ do
 8:         $I_p += a\frac{b}{N}\frac{f_p^\star(\overline{\mathbf{x}}'_i)}{L(\overline{\mathbf{x}}'_i)}$
 9:     end for
10:     for all pixels $q$ with $W_q(\overline{\mathbf{x}}_{i-1}) > 0$ do
11:         $I_q += (1-a)\frac{b}{N}\frac{f_q^\star(\overline{\mathbf{x}}_{i-1})}{L(\overline{\mathbf{x}}_{i-1})}$
12:     end for
13:     Sample random number $r \in [0, 1]$
14:     if $r \leq a$ then
15:         $\overline{\mathbf{x}}_i = \overline{\mathbf{x}}'_i$
16:     else
17:         $\overline{\mathbf{x}}_i = \overline{\mathbf{x}}_{i-1}$
18:     end if
19: end for
```

4.4.1 Start-up Bias

As mentioned above, with Metropolis sampling, the desired distribution will only be reached in the limit, that is after sampling an infinitely long Markov-Chain. If the initial sample $\overline{\mathbf{x}}_0$ was drawn from a distribution different from the stationary distribution then the sample distribution of Metropolis sampling will be biased towards this initial sample after a finite number of samples. This is called the *start-up bias.* A pragmatic solution is to simply ignore the first k samples of the Markov-Chain, but this is wasteful and it is unclear how big k should be to reduce the bias to an acceptable level. A much better solution is to weight every sample of the Markov-Chain by a *MLT correction weight* $W_{\mathrm{MLT}} = L(\overline{\mathbf{x}}_0)/p(\overline{\mathbf{x}}_0)$ such that the estimator using these samples becomes unbiased again. In W_{MLT}, $p(\overline{\mathbf{x}})$ is the distribution that was used to draw $L(\overline{\mathbf{x}})$ in the initial step. A proof for the unbiasedness of this approach can be found in the appendix of Section 11 of Veach's thesis [106]. Intuitively, it adds the noise of the initial sample to the complete chain such that the expected value is still correct.

However, there is one problem with this approach: if the initial sample $\overline{\mathbf{x}}_0$ is zero, then *all* samples of the Markov-chain will be zero too. The solution is to sample n different paths with any sampling strategy (for instance BDPT) in the first step. Out of these n paths $k << n$ paths denoted by $\overline{\mathbf{x}}_{0,j}$ with $0 \leq j < k$ are selected that will be used to initialize k independent Markov-chains. The MLT correction weight for all these chains is then $W'_{\text{MLT}} = k^{-1} \sum_j L(\overline{\mathbf{x}}_{0,j})/p(\overline{\mathbf{x}}_{0,j})$. In practice this weight is not tracked across the chain but instead the k seed paths are sampled from the much larger set of initial paths with discrete probabilities proportional to $L(\overline{\mathbf{x}}_{0,j})/p(\overline{\mathbf{x}}_{0,j})$. The samples from all these chains are then accumulated in the end of the rendering process into one single buffer to produce a final image. As a nice side-effect this allows to naturally distribute the computation effort of MLT onto multiple cores.

4.4.2 Mutation Strategies

As discussed in Section 3.4, the main degree of freedom when designing a Metropolis-Hastings algorithm is in the choice of the mutation strategies. A mutation strategy in the context of light transport is a procedure that transforms a path $\overline{\mathbf{x}} \in \Omega$ into another path $\overline{\mathbf{y}} \in \Omega$ according to some stochastic process. Recall that having local as well as global mutation strategies is crucial for a Metropolis sampler to explore the integrand efficiently. From a practical point of view there are two additional properties that should be satisfied to have an efficient algorithm: first, the mutation strategies should be accepted as often as possible, and second, mutating a path should be as cheap as possible. While the global mutation simply amounts to occasionally re-sample a completely new path, the local mutation strategies are more complicated. Consequently, many different local mutation strategies have been proposed for MLT.

Primary Sample Space Mutations

Primary Sample Space (PSS) mutations as described by Kelemen et al. [53] operate directly on the input random values that are used to generate the path, and are used in *primary sample space Metropolis light transport* (PSSMLT). This method uses the fact, that path sampling can be formulated as an operation that maps random values onto paths. Formally, we define a function $Q : \mathbb{R}^n \to \Omega$ with

$$Q(\mathbf{p}) = \overline{\mathbf{x}}, \tag{4.8}$$

where $\mathbf{p} = p_0, ..., p_{n-1}$ is a sequence of n random values and $\overline{\mathbf{x}} \in \Omega$ a path in path space. $\mathbf{p}$ can be interpreted as the sequence of random numbers that was used by the sampling process to generate a path. In a simple path tracer the random values p_i would control the sampled outgoing direction ω_i at each intersection. Similarly, the p_i's can be used to sample other quantities during path sampling like the positions on emitters when NEE is used, which component of a multi-layered BSDF is sampled, where on a participating medium a scattering event occurs and so on. We call the domain on which $\mathbf{p}$ lives the *primary sample space* (PSS). Kelemen et al. [53] proposed to let mutations directly operate on this space instead on the path space, this means

$$\overline{\mathbf{x}}' = Q \circ M \circ Q^{-1}(\overline{\mathbf{x}})$$

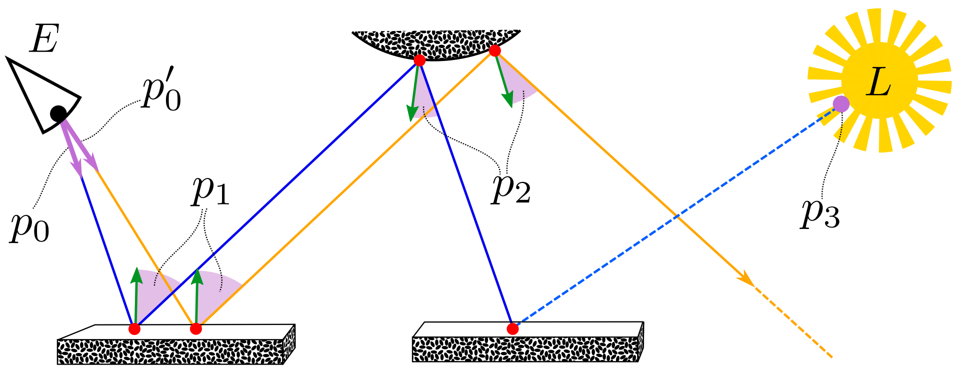


Figure 4.4: PSS Perturbation. This figure depicts a path of length 5 in blue that is created from PSS numbers $\mathbf{p} = p_0, p_1, p_2, p_3$. Factors that are directly controlled by PSS numbers are visualized in purple. The value p_0 controls the direction of the initial ray, p_1 and p_2 control the outgoing directions with respect to the local shading normal (green arrows), and p_3 determines the point on the light source for light sampling. Changing only the first PSS number (p_0 to p'_0) is a typical PSS perturbation strategy. The result of such a perturbation is shown in yellow. In this example the mutated path is very different from the original path since the curvature on the second surface leads to a strongly changed outgoing direction after the second bounce.

where M is the mutation. The reasoning behind this approach is that similar inputs $\mathbf{p}$ and $\mathbf{p}'$ usually lead to similar paths $\overline{\mathbf{x}}$ and $\overline{\mathbf{x}}'$. Local mutation strategies can thus be implemented by simply changing $\mathbf{p}$ a little bit. For instance, a mutation could be defined as a small random perturbation δ of one of the values of $\mathbf{p}$:

$$M(\mathbf{p}) = M(p_0, ..., p_{n-1}) = p_0, ..., p_{i-1}, p_i + \delta, p_{i+1}, ..., p_{n-1} = \mathbf{p}'$$

The path $\overline{\mathbf{x}}'$ should then be similar in path space to $\overline{\mathbf{x}}$. Implementing such mutation strategies is very simple. However, in practice these mutations have problems since parametrizing paths by their PSS values is suboptimal due to the potentially large partial derivatives of the throughput $\partial f^{\star}(Q(\mathbf{p}))/\partial p_i$. To illustrate this, consider a path that is constructed with a path tracer. Modifying the value p_0 that controls the initial outgoing direction ω_0 will lead to diverging paths if the surfaces are not planar or if we are not using perfectly diffuse materials. This means that the vertices of $\overline{\mathbf{x}}$ and $\overline{\mathbf{x}}'$ can be on very different surfaces after several bounces, which can lead to very different throughputs. Such an example is depicted in Figure 4.4. Since there is some considerable freedom in how to construct a path given a PSS mapping, each path $\overline{\mathbf{x}}$ could have been generated by different PSS values that have been used with different mappings $Q_i(\mathbf{p}_i) = \overline{\mathbf{x}}$. Hachisuka et al. [31] exploited this in the context of PSSMLT by using serial tempering that allows the Markov-chain not only to focus on the high contribution paths but also to focus on the best PSS mapping strategies for every region of path space. While being more powerful than the original PSSMLT algorithm, it still cannot rival some of the path space mutation strategies described in the next section for sampling SDS paths.

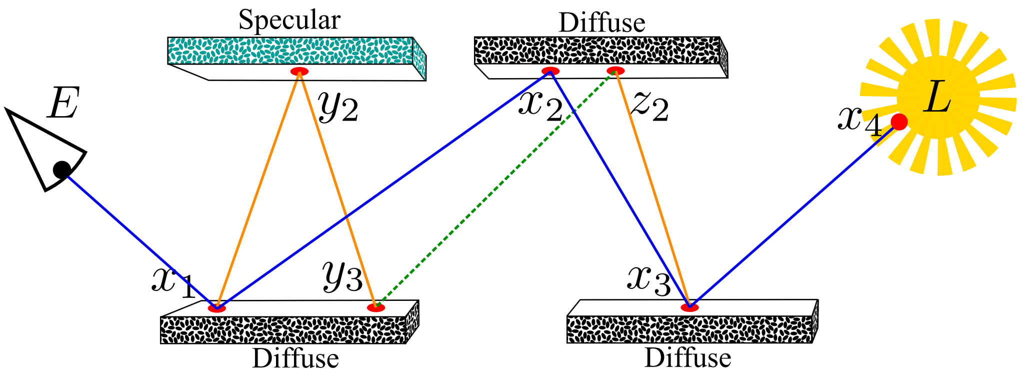


Figure 4.5: Bidirectional mutation. The base path is in blue and the perturbed path in yellow. In this example the subpath x_2 is deleted. The new path consists of an eye subpath $\overline{\mathbf{y}}_E = x_0, x_1, y_2, y_3$ and a sensor subpath $\overline{\mathbf{y}}_L = z_2, x_3, x_4$. Both subpaths are then connected at the segment $y_3 z_2$ that is depicted in green.

Path Space Mutations

Path space mutations describe a set of local and global mutation strategies that have been proposed by Veach and Guidas with their original MLT approach. Given a path $\overline{\mathbf{x}} = x_0, ..., x_n$ the perturbation will operate on the vertices of the path and create a new path $\overline{\mathbf{y}} = y_0, ..., y_m$ where n is not necessarily equal to m. Veach made a distinction between mutations and perturbations. Mutations denote modifications of paths that can drastically change the path and lead to large jumps in path space, while perturbations are "small" mutations that aim at exploring local neighbourhoods of paths. In order to simplify the following discussion we will refer to subpaths consisting of only one type of interaction as either *specular chains* or *diffuse chains*.

Bidirectional Mutation A randomly chosen subpath of the current path is replaced by a new subpath. To do so first a subpath of $\overline{\mathbf{x}} = x_0, .., x_n$ of the form $x_l, ..., x_k$ with $0 \leq l \leq k < n$ is deleted from the current path. This yields two subpaths $\overline{\mathbf{x}}_E = x_0, ..., x_{l-1}$ and $\overline{\mathbf{x}}_L = x_{k+1}, ...x_n$. The goal is now to create a new subpath that connects these two subpaths. We do this by first sampling which length t the new subpath should have and decomposing this length into two numbers p and q with $p + q = t$. From subpath $\overline{\mathbf{x}}_e$ we then sample p new vertices in the importance direction, and from $\overline{\mathbf{x}}_L$ q new vertices in the radiance direction. This yields two new subpaths $\overline{\mathbf{y}}_E = x_0, ..., x_{l-1}, y_l, ..., y_{l+p-1}$ and $\overline{\mathbf{y}}_L = z_{k-q+1}, ..., z_k, x_{k+1}, ..., x_n$. If y_{l+p-1} and z_{k-q+1} can be connected we have a new full path $\overline{\mathbf{y}} = \overline{\mathbf{y}}_E \overline{\mathbf{y}}_L$, if not we reject the path since it has zero contribution. While len($\overline{\mathbf{y}}$) can differ from len($\overline{\mathbf{x}}$), t should have a high probability of yielding paths of same length to ensure similarity of both paths. Furthermore, there should be a non-zero chance that the deleted subpath is the entire current path, i.e. $l = 0$ and $k = n$, which means that the complete path gets replaced. By allowing this we ensure ergodicity. Figure 4.5 shows an example of this mutation where $l = k = 2$, $p = 2$ and $q = 1$.

Lens Perturbation This mutation is designed to exploit the correlation of the pixel integrals such that parts of paths that contribute to one pixel can be reused for paths that contribute to other pixels in the neighbourhood. It does so by replacing the beginning of the unperturbed path of the

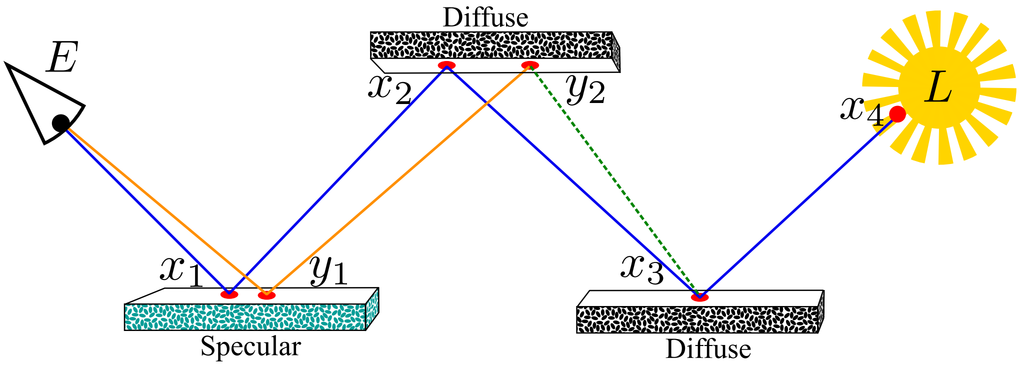


Figure 4.6: Lens perturbation. The paths have the same color coding as Figure 4.5.

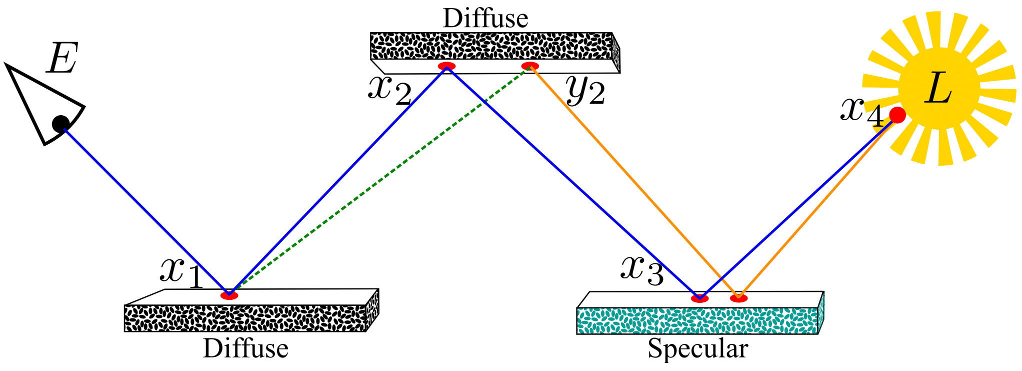


Figure 4.7: Caustic perturbation. The paths have the same color coding as Figure 4.5.

form $ES^*D(D|L)$ by slightly perturbing the primary outgoing direction $\overrightarrow{x_0x_1}$. This perturbation is propagated along the path until the first diffuse vertex y_k is hit. Note that the vertex positions of $y_1, ..., y_k$ will differ from the unperturbed path. The vertex y_k is then deterministically connected to the next vertex x_{k+1} of the unperturbed path. If x_{k+1} is specular then the mutated path will have zero throughput. Figure 4.6 shows an example where y_1 is on a specular surface, therefore the perturbation gets propagated to the next vertex which yields x_0, y_1, y_2. The vertex y_2 is diffuse and is then connected to x_3.

Caustic perturbation This mutation is designed to explore the path space around caustics. To do so the end of an unperturbed path of the form $(E|D)DS^*L$ is replaced by first perturbing the direction that connects the emitter to the last vertex $\overrightarrow{x_nx_{n-1}}$. The change of direction is then propagated along the radiance direction until a first diffuse vertex y_{n-k} is hit. Similar to the lens perturbation this means that if there is a specular chain between emitter and the diffuse vertex, the perturbation is propagated through it. The vertex y_{n-k} is then deterministically connected to x_{n-k-1}. Similar to the previous mutation, this perturbation also leads to zero throughput paths if x_{n-k-1} is specular. Figure 4.7 shows an example where $y_{n-1} = y_3$ is on a specular surface which means that the perturbation gets propagated further. The vertex y_2 is on a diffuse surface and is then connected to x_1.

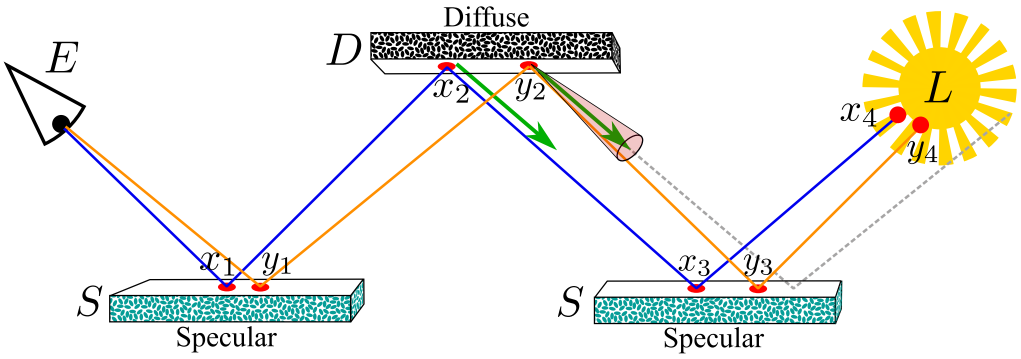


Figure 4.8: Multi-chain perturbation. The paths have the same color coding as Figure 4.5. Note that the main difference to the caustic perturbation is that the outgoing direction at y_2 is perturbed. The outgoing direction $\overrightarrow{y_2y_3}$ can thus be sampled from a cone of directions around $\overrightarrow{x_2x_3}$. If the outgoing direction at y_2 was the same as at x_2 (green arrow) then the perturbed path would not hit the light source any more in this set-up. This is depicted with the dotted gray line.

Multi-chain perturbation Neither lens nor caustic perturbation are able to perturb paths of the form ES^+DS^+L. An example for this is a caustic on the bottom of a swimming pool seen from above the water surface. The multi-chain perturbation is able in certain cases to perturb these types of paths. This perturbation is similar to the lens perturbation in that it perturbs the primary direction $\overrightarrow{x_0x_1}$ to replace the subpath of form ES^+D. We denote the last vertex that is changed by this propagation as y_{k1}. However, instead of deterministically connecting y_{k1} to x_{k1+1} as with the lens perturbation, it perturbs the corresponding outgoing direction of the original path $\overrightarrow{x_{k1}x_{k1+1}}$ and traces a new chain of vertices until a second diffuse vertex y_{k2} is hit. A connection is then attempted with the next unperturbed vertex x_{k2+1}. Alternatively the process is repeated when x_{k2+1} is specular. Figure 4.8 shows a $ESDSL$ path where the first specular chain x_0, y_1, y_2 is identical to the one computed in Figure 4.6 with the lens perturbation. A connection to x_3 is now not possible since x_3 is on a specular surface. Simply copying the outgoing direction of $\overrightarrow{x_2x_3}$ will likely lead to a path that misses the emitter. The multi-chain perturbation still can create a new path by slightly perturbing the outgoing direction $\overrightarrow{x_2x_3}$ and then tracing a new chain of vertices until a diffuse surface is hit. In this example the next diffuse vertex y_{k2} happens to be on the emitter, which makes an additional reconnection obsolete.

Manifold Perturbation Even with the multi-chain perturbation, MLT in its original form has difficulties in exploring paths around SDS paths. For instance, consider the path in Figure 4.11. SDS paths with small emitters make it very difficult for a multi-chain perturbation to still produce a path with non-zero throughput. The probability of the multi-chain perturbation to perturb the outgoing direction $\overrightarrow{x_{k1}x_{k1+1}}$ such that the light source is still hit can become arbitrarily low for small or far away light sources. Jakob and Marschner [45] developed a framework called manifold exploration (ME) that allows to move along specular manifolds in path space. Intuitively, specular manifolds are connected regions of specular paths in path space that have a non-zero throughput. Later in this section, we will provide a more formal definition of specular manifolds. Essentially, with ME we can find new specular paths that are located on the same specular manifold as an already

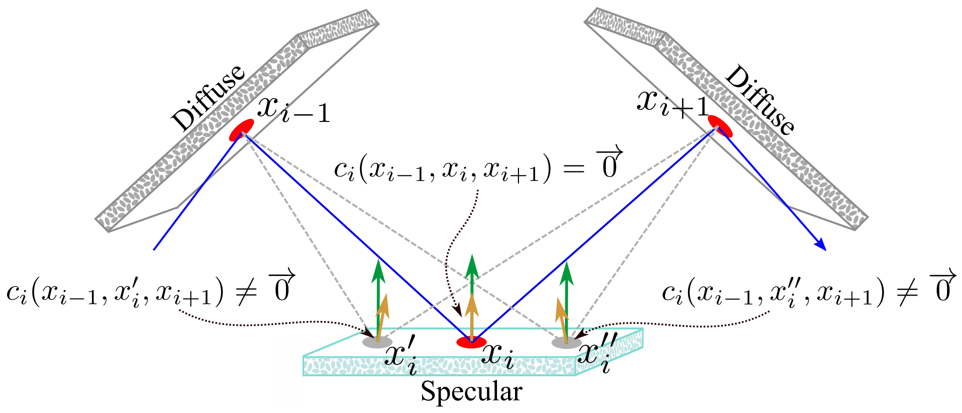

Diffuse
x_{i-1}
Diffuse
x_{i+1}
$c_i(x_{i-1}, x_i, x_{i+1}) = \overrightarrow{0}$
$c_i(x_{i-1}, x'_i, x_{i+1}) \neq \overrightarrow{0}$
$c_i(x_{i-1}, x''_i, x_{i+1}) \neq \overrightarrow{0}$
x'_i
x_i
x''_i
Specular

Figure 4.9: A motivating example for manifold exploration with fixed vertices x_{i-1} and x_{i+1}. In order for the blue path to have any throughput the normal direction (green arrow) and half vector direction (yellow arrow) at x_i must be the same which corresponds to the constraint function c_i to be zero. The position of x_2 is thus determined by the adjacent vertices in the path. In a local neighbourhood of x_i, no other vertex fulfils this constraint (grey points).

known specular path. Therefore, ME allows to explore the interesting regions of path space around specular paths in a much more robust way than the previously discussed perturbations. With the ME framework, Jakob and Marscher were able to modify the perturbation of the second vertex chain of the multi-chain perturbation in such a way that a new configuration of vertices is found that connects y_{k1} with x_{k2} (see Figure 4.11). MLT using this perturbation is referred to as *manifold exploration Metropolis light transport* (MEMLT).

The basic idea of the method is that the positions of any vertices on specular surfaces in a path are determined completely by the two adjacent vertices if we want the path to transport any light (Figure 4.9). Jakob and Marschner motivate this by the observation that any specular vertex x_i must satisfy $(\overrightarrow{x_i x_{i-1}} + \overrightarrow{x_i x_{i+1}}) || n_i$ in order to have any throughput. Note that n_i is the shading normal at x_i and"$||$" is the symbol to denote parallel vectors. This constraint can be reformulated in a form $c_i(x_{i-1}, x_i, x_{i+1}) = \overrightarrow{0}$ where

$$c_i(x_{i-1}, x_i, x_{i+1}) = T(x_i)^T h(x_i, \overrightarrow{x_i x_{i-1}}, \overrightarrow{x_i x_{i+1}}).$$

Here $T(x)$ is a matrix whose columns form a basis for the shading tangent plane and $h(x, v, w)$ is the generalized half vector. Note that $c \in \mathrm{R}^2$. These constraints state that the generalized half-vector at x_k projected onto the shading tangent plane of x_k should be zero. With this tool, specular paths with non-zero throughput in a neighbourhood of $\overline{\mathbf{x}}$, denoted by $\Omega_{N(\overline{\mathbf{x}})}$,[3] can be described by the *specular manifold*

$$S(\overline{\mathbf{x}}) = \{\overline{\mathbf{y}} \in \Omega_{N(\overline{\mathbf{x}})} : C(\overline{\mathbf{y}}) = \overrightarrow{0}\},$$

where $C(\overline{\mathbf{x}})$ is the stack of the constraints c_i for every specular vertex x_i in $\overline{\mathbf{x}}$. Jacob and Marschner noted that this manifold can be parametrized because of the implicit function theorem [101] with

[3]Neighbourhood essentially means that vertices stay on arbitrarily small surface patches around the vertices of $\overline{\mathbf{x}}$. It is not possible to have a global parametrization since for any two diffuse vertices separated by a specular chain there could be several very different specular chains in path space that connect them.

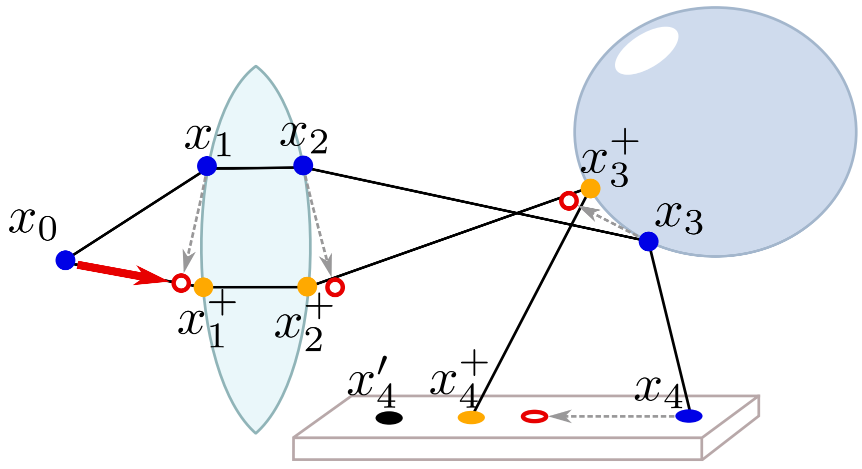


Figure 4.10: An example of one iteration the prediction-correction approach discussed below. A specular chain (blue dots) starting at x_0 and ending at x_4 should be modified such that its endpoint is shifted to x_4'. Prediction step: the endpoint x_4 is shifted towards the target endpoint x_4' yielding an intermediate endpoint $x_4 + \beta(x_4' - x_4)$. The vertex positions along the specular chain are modified according to a locally linear prediction model, potentially leading to vertices that are not located on surfaces any more (red circles). Correction step: instead of using of using these predicted positions directly, a new specular chain (yellow dots) with an initial direction of $\overrightarrow{x_0 x_1^+}$ (red arrow) is traced.

a function Q that determines the positions of all specular vertices in $\overline{\mathbf{x}}$ as a function of all diffuse vertices in $\overline{\mathbf{x}}$. The goal is to develop an algorithm that allows to move on this specular manifold. More specifically, given a subpath $\overline{\mathbf{x}} = x_{k1}, ..x_{k2}$ where x_{k1} and x_{k2} are diffuse vertices and the rest in between is specular, we want to have an algorithm that allows us to either move x_{k1} or x_{k2} while automatically modifying the positions of the specular vertices $x_{k1+1}, ..., x_{k2-1}$ such that the modified subpath still transports light. In order to do so the position of the specular vertices $x_{k1+1}, ..., x_{k2-1}$ must be expressed as a function of x_{k1} and x_{k2}. To do so, Jakob and Marschner compute a Jacobian of the constraint matrix C, denoted by ∇C. The matrix ∇C consists of 2×2 blocks $\partial c_i / \partial x_j$. We denote the columns of ∇C as B_j and set $A = [B_1, ..., B_{n-1}]$. The movement of all specular vertices $x_1, ..., x_{n-1}$ with respect to the endpoint x_n can then be computed as

$$\frac{\partial(x_1, ..., x_{n-1})}{\partial x_n} = -A^{-1} B_n.$$

With this it becomes possible to compute how the first specular vertex x_1 in the chain must move for the path to remain on a specular manifold if we move the diffuse endpoint x_n to x_n'. However, C is only defined for a infinitesimally small region around $\overline{\mathbf{x}}$. Thus non-infinitesimal movements of x_n will lead to prediction errors where the specular vertices may not even be on surfaces any more (Figure 4.10). To correct for this, Jakob and Marschner used a Newton-like prediction-correction approach, where the predicted location x_1 is only used to perturb the outgoing direction from x_0. Given a target endpoint x_n', $\beta = 1$ and a error-tolerance of ϵ the approach works as follows:

1. Compute $x_1^+, ..., x_n^+$ according to $-A^{-1} B_n$ of the current specular chain $x_0, ..., x_n$ when moving

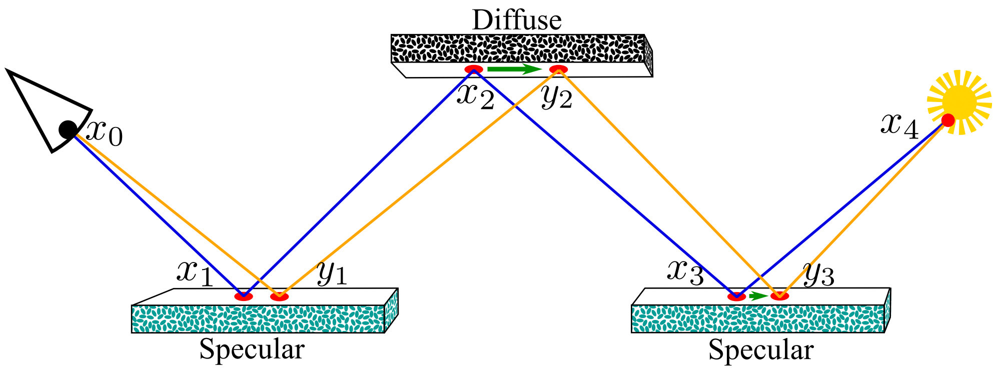


Figure 4.11: Manifold exploration perturbation of a SDS path. The first specular chain x_0, x_1, x_2 is propagated identically as in the lens perturbation yielding x_0, y_1, y_2. The second specular chain x_2, x_3, x_4 uses a manifold walk to be shifted onto y_2, y_3, x_4.

x_n to $x_n + \beta(x'_n - x_n)$ by tracing a new chain from x_0 with a modified direction $\overrightarrow{x_0 x_1^+}$.

2. If a new chain could be traced and $||x_n^+ - x'_n|| < ||x_n - x'_n||$ set $x_1, ..., x_n = x_1^+, ..., x_n^+$ and $\beta = \min(1, 2\beta)$, or else set $\beta = \beta/2$.

3. Repeat step 1 and 2 until $||x_n - x'_n|| < \epsilon$ or fail if we exceeded the maximum number of iterations.

Note that this algorithm is guaranteed to converge, however it can fail when the maximum number of iterations is reached before convergence. An analogous algorithm for moving the starting point of the chain can be used. There, ray tracing occurs in the reverse direction and we aim at finding a new incoming direction $\overrightarrow{x_n x_{n-1}^+}$ when moving x_0 to $x_0 + \beta(x'_0 - x_0)$ based on $-A^{-1}B_0$.

A perturbation strategy based on this approach called *manifold perturbation* is designed to perturb subpaths of the form $(E|D)S^+DS^+(D|L)$. It works as follows: first perturb $\overrightarrow{x_0 x_1}$ and propagate the perturbation until the first diffuse vertex is reached yielding $x_0, y_1, ..., y_{k1}$. Now perform a manifold exploration of chain $x_{k1}, ..., x_{k2}$ by moving the first vertex x_{k1} to y_{k1}. The manifold exploration then yields a new specular chain $y_{k1}, y_{k1+1}, ..., x_{k2}$. See Figure 4.11 for a simple example.

It is noteworthy that recently related perturbation strategies based on directly mutating the half-vectors in the specular chain and finding new valid paths on the specular manifold have been developed by Kaplanyan et al. [52] and later refined by Hanika et al. [35]. In another line of work Li et al. [70] discussed the difficulties of MEMLT to sample strongly anisotropic specular manifolds and proposed a method based on Hessian-Hamiltonian Monte-Carlo to perturb paths more efficiently.

4.4.3 Limitations

When rendering images with PT or BDPT it is very predictable how the image will be resolved over time. The noise will go down at roughly the same rate everywhere in the image, and in the limit a clear image will be rendered. For artists this is very useful since it allows previewing scenes with low sample counts. Since many artists improve their scene assets incrementally, speeding up the feedback loop of changing the scene and getting some results is paramount for quickly getting complex scene assets in production rendering.

Unfortunately MLT does not have such a predictable convergence behaviour. The reason is that MLT can explore a region around challenging paths very well once such a sample has been found, but MLT provides no mechanism other than bidirectional mutations to find such paths in the first place. As a consequence, small details in the image like small caustics may remain "undiscovered" for a very long time during the render process. In extreme cases an otherwise converged looking image can still be missing some features of the image that have simply not been discovered yet by the Markov-chains. This makes it hard to judge when an image is really converged and also makes non-converged images bad predictions for the converged image, since entire parts of the illumination might be missing. Also, up to date, no generalization of MLT over several frames for animation exists. Consequentially, MLT can lead to wild flickering in animations when the images are not very close to convergence. This problem is amplified since with MLT stratification techniques are not directly applicable. Because of these issues, to the best of our knowledge, MLT has not been used for production rendering anywhere.

A related method, *energy-redistribution path tracing* (ERPT) [10], tries to mitigate these issue by creating a very large number of very short Markov-chains each starting with paths that are stratified on the image plane and using local perturbation strategies only. This leads to a more uniform coverage of the sampling space and thus to a convergence behaviour that is more similar to PT and BDPT. Still, similar to MLT this algorithm also suffers from low frequency noise that makes it hard to use in animation rendering.

4.5 Adaptive Sampling and Reconstruction

The path sampling methods discussed above all suffer from noise artefacts when using finite numbers of samples since $f^\star$ is generally not band limited. Every pixel is computed mostly independently - even with MLT - and each sensor response will be approximated with slightly different error. The different errors from pixel to pixel will appear as high frequency noise to the viewer. The magnitude of this noise will, on average, depend on the number of samples and the used sampling strategy and will usually vary over the image plane because the complexity of the pixel integrals also varies greatly in a single image.

Adaptive sampling and reconstruction (ASR) methods try to recover the desired signal by removing this non-uniform noise from the image. They do this by attacking the problem from two sides. First, they aim at constructing an image that is as good as possible by adapting the reconstruction filter W_p locally for every pixel p. Second, they adapt the sampling densities such that the error of the image reconstructed with W_p is minimized. Adaptive sampling and adaptive reconstruction can be applied independently but they are usually combined in order to maximize efficiency. ASR methods are typically used on top of path tracing since path tracing allows for simple control of where in the image how many samples should be computed. However, they can also be used to some extent for BDPT[4] and for any other rendering method that allows for control of the sample density in image space. The reconstruction part alone could however be used for any rendering method as a post-process (i.e. also for MLT). ASR methods are in general not unbiased for two reasons, first

[4]Although the density of direct light paths in image-space cannot be controlled that easily.

reconstructing a smooth signal from noisy data will typically lead to some bias and second, as pointed out by Kirk and Avro [58] adaptive sampling can lead to bias. However, while not being unbiased most of the discussed methods are at least consistent.

The remainder of this section will give a brief overview of methods related to ASR. We will follow the classification of ASR methods by Zwicker et al. [115] into *a priori* methods that use analytical models of the light transport problem to compute good local reconstruction filters and sampling densities, and *a posteriori* methods that use empirical information obtained during the sampling process.

4.5.1 A Priori Methods

These methods increase efficiency of rendering by analysing the local properties of the light transport problem. These types of algorithms have been sparked by the frequency analysis of light transport developed by Durand et al. [19], where they performed a frequency analysis of local light field around single path samples. This work showed that one single sample could be enough to determine appropriate sampling densities and reconstruction filters in a local neighbourhood. However, due to complexity of this analysis most methods following this line of work only support subsets of the effects encountered in light transport. For instance Durands original paper was limited to direct illumination with glossy surfaces. A number of similar approaches were developed to support other effects like motion blur [23], soft shadows [22], ambient occlusion [21], depth of field [100], participating media [5] or to some extend even indirect illumination [6, 77].

Somewhat related are methods that perform an analysis based on the structure of a light field without relying on frequency analysis. The most notable methods reproject samples to multiply the amount of samples. The initial idea presented by Lehtinen et al. [66] was to exploit the light field structure to improve sampling of distributed effects like motion blur, depth of field or direct soft shadows. The method was later extended to support diffuse indirect illumination [67].

Another set of methods analyses local derivatives. Intuitively, by analysing partial derivatives of light transport one can recognize regions where the light transport is smooth or even locally constant. In context of sampling and reconstruction, detecting such regions is important since large reconstruction filters can be used on them without biasing the result too much and thus lower sampling rates are required. This idea was initially used in *irradiance caching* (IC) [112]. IC computes the irradiance gradients in image space and uses this information to determine local sampling densities for the irradiance as well as reconstruction kernels for reconstructing the irradiance independently of the direct light. The original approach was however restricted to diffuse surfaces. Later, the method was extended to support glossy surfaces [61] and participating media [48].

4.5.2 A Posteriori Methods

A posteriori ASR methods use data obtained during the rendering process to guide sampling and reconstruction. Often this is implemented as a feedback loop that periodically updates the sampling densities and the reconstruction filters as more data is acquired. Compared to a priori methods these methods tend to be more general since they are based on statistics that may not depend on any

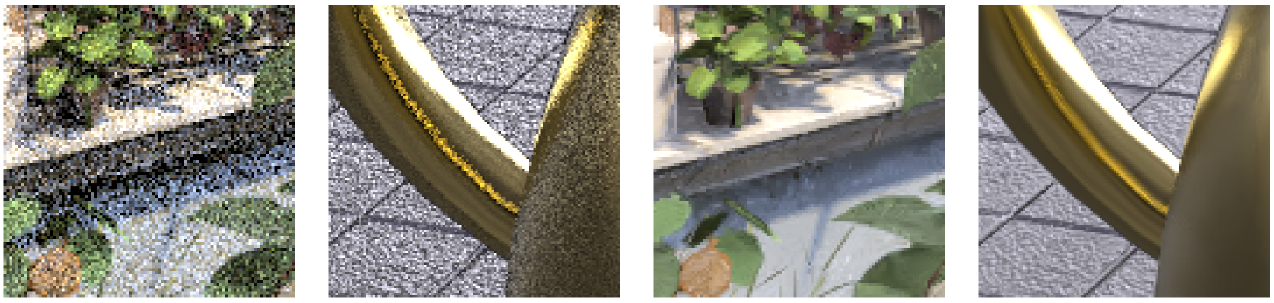

(a) unfiltered images (b) reconstructed images

Figure 4.12: Example of an a posteriori ASR method as discussed in Section 4.5.2. The method used in this example is RDFC. This figure was recreated from Rousselle et al. [95].

specific light transport effect. On the other hand, sample distributions and reconstruction filters only become efficient over time when enough data about the light transport was gathered, thus for very low sampling rates a posteriori ASR methods might perform poorly.

One of the earliest a posteriori ASR methods was developed by Mitchell [79]. This method only operates in the image space dimension. It subdivides the image space dimension into cells and decides whether the cells should be further subdivided based on a simple contrast metric of the samples generated so far. The adaptive sampling of this method is restricted on the image space dimension only. No adaptive sampling of any other dimension of the path space integral is performed (see Figure 4.1). Thus methods of this type are called *image-space* ASR method.

More recently, Hachisuka et al. [30] developed an algorithm that adaptively samples paths in multi-dimensional spaces. To do so, the path space is subdivided by a kd-tree and in each node statistics are stored of samples belonging to the corresponding subspace. Based on this, additional samples are distributed into each subregion. Additional dimensions like the position of the sensor for depth of field, the position on light sources for soft shadows or the point in time for motion blur allow to sample these effects much more efficiently than could be done with image space adaptive sampling. In addition the method also proposes to use different anisotropic reconstruction filters based on the sampling density. The methods main disadvantage is that its efficiency decreases with increasing number of dimensions that we want to sample adaptively, thus adaptively sampling the entire path space remains challenging. The idea of sampling in the multi-dimensional space directly was used by many of the a priori methods discussed earlier.

An approach operating on image space only that could use much more powerful filters than Mitchells approach was introduced by Overbeck et al. [88]. This method aims at sampling and reconstructing the image in a wavelet basis. The image is reconstructed by performing wavelet shrinkage and additional samples are distributed according to some prior function that estimates the quality of the reconstruction locally. A similar method targeting interactive ray tracing has been proposed shortly after it by Dammertz et al. [15]. The crucial part of Overbecks' algorithm is the feedback loop between the estimation of the error of the reconstructed image and distribution of additional samples according to the reconstruction error. A body of work by Rousselle et al. adapted this method to directly reconstruct an image from a filter bank, where at each pixel the reconstruction filter with the best trade-off between variance and bias is chosen. The initial method was restricted to use a filter bank of isotropic Gaussians [93], but it was later extended to use non-local means filters[94], joint bilateral filters [71] and mixtures of joint bilateral and non-local means filters [95]. The latest of these algorithms called *robust denoising using feature and color information* (RDFC) is shown as an example of an a posteriori ASR methods in Figure 4.12. Recently, a similar method was developed that applies adaptive sampling and reconstruction separately on different illumination effects that are then composed to a final image, at the same time motion vectors of each effect are tracked over different frames to provide a high quality reconstruction for animations [114].

While the aforementioned methods distribute samples all based on variance estimates of the samples or the reconstruction, other methods were proposed that use more sophisticated metrics to guide the sample distribution. For instance, Delbracio et al. [16] use information about the histogram of samples to detect similar pixel integrals and Sen et al. [96] use the correlation of input noise and output noise to be able to distinguish Monte-Carlo noise from high frequency geometry or textures that might produce similar patterns.

Other research is based on the realization that performing a reconstruction by filtering can be interpreted as performing a regression analysis over image patches to find simple function that approximate these patches. Filtering is then the result of averaging many such overlapping windows. One such filter yielding similar results as a bilateral filter is the guided image filter [38] that was later used in an ASR algorithm [4]. A similar approach with a more powerful regression based on auxiliary feature data was proposed by Moon et al. [83]. This approach has later been extended to use regressions of higher order [85] and also been adapted to allow interactive rendering at the price of some quality [84].

Chapter 5

Gradient-Domain Rendering

In this chapter we will discuss previous work on gradient-domain rendering upon which all our contributions are built. Gradient-domain rendering was first introduced in the context of Metropolis Light Transport by Lehtinen et al. [68] and then applied to path tracing by Kettunen et al. [57]. We will first discuss gradient-domain rendering in a general way independently of specific light transport algorithms and provide an end-to-end analysis that explains why gradient-domain rendering is beneficial. In the later parts of this chapter we will dive into the the specifics of gradient-domain rendering applied to path tracing and Metropolis light transport.

5.1 Motivation

For rendering images one separate integral must be solved for every pixel in the image. These integrals are usually strongly correlated for pixels that are close-by in image-space. This means there is inherently a large amount of redundant computation occurring when all pixel color values are estimated independently from each other. It is therefore natural that several methods in rendering try to exploit this. The most successful ones are reconstruction techniques in which samples belonging to one pixel are reused to estimate the color values of several pixels. In Section 4.5 we discussed how this is done by applying image filtering or sample re-projection. However, these techniques are biased. Further, many of these methods only work on a subset of the lighting phenomena or have a performance that strongly depends on the amount of additional information that can be obtained during the rendering process. Another type of techniques exploiting pixel correlation are MCMC rendering techniques (Section 4.4) that mutate paths such that subpaths can be reused for several pixels. These methods are unbiased but introduce correlation between the path samples that lead to an increase in low-frequency artefacts. This is especially disturbing in animations.

We will explore an alternative way of exploiting the similarities of the pixel integrals. Recall that correlated sampling[1] (Section 3.3.6) is a variance reduction technique tailored specifically for sampling the difference of similar integrals in an efficient way without the need of analytically knowing both integrals a priori. Despite rendering consisting of such similar integrals, correlated sampling has not found much interest in the rendering community until recently. However, if we could formulate

[1]In the sense of common random numbers, not control variates.

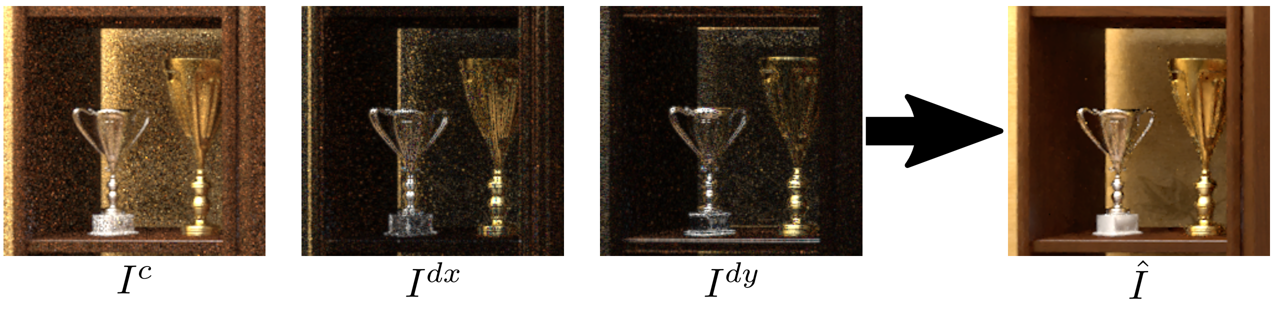


Figure 5.1: Gradient-domain rendering methods sample the finite differences of the throughput I^{dx} and I^{dy} alongside a much noisier color image I^c. A smooth image $\hat{I}$ is then reconstructed from this data with a screened Poisson reconstruction. These images were rendered with G-PT (Section 5.4).

the sampling problem in rendering in terms of color differences instead of pixel colors, correlated sampling could be applied on rendering.

It is well known that images can be modified by editing their gradients [89]. The gradients can be approximated by the finite differences in the x and y direction of the image. The desired modified image is then formulated as the image that minimizes the error in a set of linear equations that encode constraints on the finite differences and colors of the image. This amounts to solving a *screened Poisson equation*. The same machinery can be used in rendering to construct an image out of sampled finite differences and a potentially much noisier color image.

Thus, the basic idea of gradient-domain rendering is to compute the finite differences of pixels with correlated sampling alongside the pixel colors and then to reconstruct a much smoother image based on these sampled quantities by solving a screened Poisson equation (Figure 5.1).

5.2 Basics

In gradient-domain rendering the main quantities that are sampled are, as the name implies, gradients. More specifically, we want to compute the image-space finite differences of an image I:

$$\begin{aligned}\Delta_x(I_{i,j}) &= \Delta_{1,0}(I_{i,j}) = I_{i+1,j} - I_{i,j}\\ \Delta_y(I_{i,j}) &= \Delta_{0,1}(I_{i,j}) = I_{i,j+1} - I_{i,j}\end{aligned}$$

where $I_{i,j}$ denotes the color value of the pixel with image-plane coordinates (i, j). We can generalize this to differences between arbitrary pixels with

$$\Delta_\delta(I_p) = I_q - I_p \tag{5.1}$$

where p and q are any 2D image-plane coordinates $p = (i, j)$ and $q = (i', j')$ and $\delta = q - p$ is the offset.

5.2.1 Sampling Gradients

Recall that computing the finite differences of two pixels corresponds to computing the difference of two integrals (see Equation 2.24):

$$\Delta_\delta(I_p) = \underbrace{\int_\Omega W_q(\overline{\mathbf{y}}) f^\star(\overline{\mathbf{y}}) d\mu(\overline{\mathbf{y}})}_{I_q} - \underbrace{\int_\Omega W_p(\overline{\mathbf{x}}) f^\star(\overline{\mathbf{x}}) d\mu(\overline{\mathbf{x}})}_{I_p}. \tag{5.2}$$

In Section 3.3.6 we describe how the variance of the difference of two similar integrals can be reduced by using correlated sampling. In order to be able to perform correlated sampling we will reformulate Equation 5.2 to be an integral of a difference instead of a difference of integrals. The remainder of this section describes how this can be achieved.

To this end, we define a bijective function $T_\delta : \Omega_p \to \Omega_q$ where $\Omega_p = \{\overline{\mathbf{x}} \in \Omega : W_p(\overline{\mathbf{x}}) > 0\}$ and $\delta = q - p$. This function transforms a path with image-space coordinates p into a path with image-space coordinates q. We call such a function a *shift mapping*. In this context we call $\overline{\mathbf{x}}$ the *base path* and $T_\delta(\overline{\mathbf{x}})$ the *offset path*. Formally, T_δ must fulfil

$$\begin{aligned} &(1)\ \ W_p(\overline{\mathbf{x}}) = W_q(T_\delta(\overline{\mathbf{x}})) \\ &(2)\ \ T_\delta^{-1}(T_\delta(\overline{\mathbf{x}})) = \overline{\mathbf{x}}. \end{aligned} \tag{5.3}$$

Note that $T_\delta^{-1} = T_{-\delta}$. To avoid cluttering we will omit the subscript δ in T whenever it is clear from the context what the subscript should be. Given such a shift mapping we can follow the algebraic steps described in Lehtinen et al. [68] to reformulate I_q from Equation 5.2:

$$\begin{aligned} I_q &= \int_\Omega W_q(\overline{\mathbf{x}}) f^\star(\overline{\mathbf{x}}) d\mu(\overline{\mathbf{x}}) \\ &= \int_{T^{-1}(\Omega)} W_q(T(\overline{\mathbf{x}})) f^\star(T(\overline{\mathbf{x}})) d\mu(T(\overline{\mathbf{x}})) \\ &= \int_\Omega W_q(T(\overline{\mathbf{x}})) f^\star(T(\overline{\mathbf{x}})) \left| \frac{\partial\mu(T(\overline{\mathbf{x}}))}{\partial\mu(\overline{\mathbf{x}})} \right| d\mu(\overline{\mathbf{x}}) \\ &= \int_\Omega W_p(\overline{\mathbf{x}}) f^\star(T(\overline{\mathbf{x}})) \left| \frac{\partial\mu(T(\overline{\mathbf{x}}))}{\partial\mu(\overline{\mathbf{x}})} \right| d\mu(\overline{\mathbf{x}}). \end{aligned} \tag{5.4}$$

The step in the second line is a no-operation where we shift the paths by T but integrate over a path space that is shifted in the inverse direction $T^{-1}(\Omega)$. The next step performs a change of integration variables from $\mu(T(\overline{\mathbf{x}}))$ to $\mu(\overline{\mathbf{x}})$. This introduces the Jacobian determinant $|T| = |\partial\mu(T(\overline{\mathbf{x}}))/\partial\mu(\overline{\mathbf{x}})|$. The final step uses Equation 5.3. Given Equation 5.4 we can rewrite Equation 5.2 as

$$\Delta_\delta(I_p) = \int_\Omega W_p(\overline{\mathbf{x}}) \underbrace{[f^\star(T(\overline{\mathbf{x}}))|T| - f^\star(\overline{\mathbf{x}})]}_{g_\delta(\overline{\mathbf{x}})} d\mu(\overline{\mathbf{x}}), \tag{5.5}$$

which can be estimated by using correlated sampling as described in Equation 3.28 by using

$$I_p^\delta = \frac{1}{N}\sum_{i=1}^{N} W_p(\overline{\mathbf{x}}_i)\frac{g_\delta(\overline{\mathbf{x}}_i)}{p(\overline{\mathbf{x}})}. \tag{5.6}$$

Note that the Jacobian determinant $|T|$ can be interpreted as a correction factor for the fact that we sample $f^\star(T(\overline{\mathbf{x}}_k))$ with the PDF $p(\overline{\mathbf{x}}_k)$ instead of $p(T(\overline{\mathbf{x}}_k))$. Therefore, the Jacobian determinant must be $|T| = p(\overline{\mathbf{x}})/p(T(\overline{\mathbf{x}}))$. Plugging this into the equation above reveals that I_p^δ is indeed an estimator of $\Delta_\delta(I_p)$:

$$\begin{aligned} I_p^\delta &= \frac{1}{N}\sum_{i=1}^{N} W_p(\overline{\mathbf{x}}_i)\frac{f^\star(T(\overline{\mathbf{x}}_i))\frac{p(\overline{\mathbf{x}}_i)}{p(T(\overline{\mathbf{x}}_i))} - f^\star(\overline{\mathbf{x}}_i)}{p(\overline{\mathbf{x}}_i)} \\ &= \frac{1}{N}\sum_{i=1}^{N} W_p(\overline{\mathbf{x}}_i)\left[\frac{f^\star(T(\overline{\mathbf{x}}_i))}{p(T(\overline{\mathbf{x}}_i))} - \frac{f^\star(\overline{\mathbf{x}}_i)}{p(\overline{\mathbf{x}}_i)}\right] \\ &= \underbrace{\frac{1}{N}\sum_{i=1}^{N} W_q(T(\overline{\mathbf{x}}_i))\frac{f^\star(T(\overline{\mathbf{x}}_i))}{p(T(\overline{\mathbf{x}}_i))}}_{I_q^c} - \underbrace{\frac{1}{N}\sum_{i=1}^{N} W_p(\overline{\mathbf{x}}_i)\frac{f^\star(\overline{\mathbf{x}}_i)}{p(\overline{\mathbf{x}}_i)}}_{I_p^c}, \end{aligned} \tag{5.7}$$

since $E[I_p^c] = I_p$ and $E[I_q^c] = I_q$ and thus $E[I_p^\delta] = \Delta_\delta(I_p)$. Note, that Equation 5.7 reveals that in the context of Monte-Carlo sampling the shift mapping T needs to always create offset paths that are sampleable by the underlying path sampling strategy. This means that we need to ensure that

$$\forall \overline{\mathbf{x}} \in \Omega_p : p(T(\overline{\mathbf{x}}_i)) > 0.$$

We know from Equation 3.27 that the variance of the estimators I_p^δ decreases with increased positive correlation between $f^\star(T(\overline{\mathbf{x}}))|T|$ and $f^\star(\overline{\mathbf{x}})$. Therefore the main challenge lies in finding a shift mapping T that leads to maximal correlation between the throughputs of the base and offset paths.

5.2.2 Shift Mappings

From the previous section we know that T must fulfil Equation 5.3, this means it must be a bijection from Ω_p to Ω_q and T must shift $\overline{\mathbf{x}}$ such that value of importance function of pixel q applied on the offset path is equal to the value of importance function of pixel p applied on the base path. Further, T must always create sampleable offset paths, meaning $p(T(\overline{\mathbf{x}})) > 0$ for all $\overline{\mathbf{x}}$. And finally, in order to maximize efficiency, T should be designed such that

- the correlation between $\overline{\mathbf{x}}$ and $T(\overline{\mathbf{x}})$ is as strong as possible, and
- the computation of $T(\overline{\mathbf{x}})$ is as cheap as possible.

In the following we will review how to design shift mappings.

Primary Sample Space Shift

A simple shift mapping that fulfils our requirements is a *primary sample space shift* similar to the PSS mutation described in Section 4.4.2. Such a new shift mapping $T^\star$ operates on the PSS variables directly. The full shift T^{PSS} of a path $\overline{\mathbf{x}}$ to $\overline{\mathbf{y}}$ can be formalized as a transformation from path space to PSS Q^{-1}, followed by a shift in PSS $T^\star$, followed by a transformation back to path space Q:

$$W_q(T^{PSS}(\overline{\mathbf{x}})) = W_q(Q \circ T^\star \circ \underbrace{Q^{-1}(\overline{\mathbf{x}})}_{\mathbf{p}}) = W_p(\overline{\mathbf{x}}).$$

We assume that $\mathbf{p} = p_0, p_1, p_2, ..., p_n$ where p_0 controls the x-position on the image plane, p_1 the y-position and $p_2, .., p_n$ all other degrees of freedom of the path $\overline{\mathbf{x}}$. The PSS shift can then be defined as

$$T^\star_\delta(\mathbf{p}) = p_0 + \delta_x, p_1 + \delta_y, p_2, ..., p_n.$$

Note that T^{PSS} is a bijection from Ω_p to Ω_q by construction and that the Jacobian determinant is the same as for the general mapping described in Section 5.2.1:

$$|T^{PSS}| = \underbrace{\left|\frac{\partial \overline{\mathbf{y}}}{\partial \mathbf{p}'}\right|}_{|Q|} \underbrace{\left|\frac{\partial \mathbf{p}'}{\partial \mathbf{p}}\right|}_{|T^\star|} \underbrace{\left|\frac{\partial \mathbf{p}}{\partial \overline{\mathbf{x}}}\right|}_{|Q^{-1}|} = \left|\frac{\partial \overline{\mathbf{y}}}{\partial \mathbf{p}'}\right| \left|\frac{\partial \mathbf{p}}{\partial \overline{\mathbf{x}}}\right| = \frac{p(\overline{\mathbf{x}})}{p(\overline{\mathbf{y}})}$$

The first step follows from the chain rule, step two from $\partial \mathbf{p}'/\partial \mathbf{p} = 1$ and step three from $\partial \overline{\mathbf{x}}/\partial \mathbf{p} = 1/p(\overline{\mathbf{x}})$.

As for PSSMLT one can implement this shift by simply storing the PSS values of the base path and then retrace an offset path in the neighbouring pixel with the same PSS values $p_2, ..., p_n$. This is very simple to implement but has several drawbacks:

First, as in PSSMLT, paths can diverge significantly over the bounces due to fast changing normals. This often leads to rather low correlation between base and offset path. This problem is amplified by the fact that the values of $\mathbf{p}$ could be used for different purposes in base and offset path: a path shifted in PSS might hit different surfaces with different materials than the base path. These materials might use different numbers of random values of the PSS sequence than the materials encountered by the base path. This means a shifted path might use random values for different purposes which further reduces correlation and can even lead to base and offset path pairs of different length.

Second, computing the finite difference with this shift requires us to compute two full paths. In terms of computational costs a finite difference is thus roughly twice as expensive to sample as a pixel color. We will see in the next paragraph how this overhead can be reduced significantly by defining T such that the offset path reuses subpaths of the base path.

Deterministic Path Space Shifts

Because the resulting correlation of base and offset path with the PSS shifts is very hard to control, we want to use shifts that deterministically transform a path based on its geometrical properties.

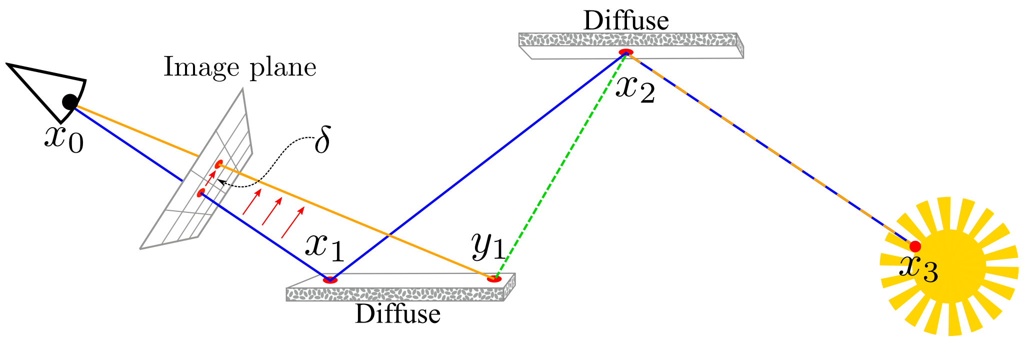


Figure 5.2: Simple path space shift mapping. The base path (blue) is shifted on the image plane by δ, creating a new primary path segment from x_0 to y_1 for the offset path (yellow). The new sub path is then reconnected with then next vertex of the base path (green). Note that the segment from x_2 to x_3 is shared between the base and offset path.

These strategies are similar to the lens mutation used in MLT but are deterministic. One of their main advantages over PSS is that they allow to reuse parts of the base path to construct the offset path, which means that less ray intersection operations need to be done for constructing the offset path.

A simple way to do so is depicted in Figure 5.2. For a given a base path $\overline{\mathbf{x}}$, a ray is shot through the image-plane position $s(x_1) + \delta$, where $s(x_1)$ is the primary vertex x_1 projected onto the image plane. This yields a new primary intersection point y_1. The remainder of the path is simply computed by trying to connect y_1 to x_2 and if successful directly reuse $x_3, ..., x_n$ for $\overline{\mathbf{y}}$.

However, the reconnection between y_1 and x_2 can lead to BSDF values at y_1 and x_2 that are very different from the corresponding BSDFs in the base path. For instance, if either y_1 or x_2 is on a near-specular or perfectly specular material, the BSDFs of the offset path will become very small or zero. This will reduce the correlation between the throughput of the base and offset path significantly. The key to handle such cases is to delay the re-connection to a later point along the path where the BSDFs are less sensitive to changes in direction. The challenges in designing such an approach lie in (1) determining when during the offset path reconstruction we can reconnect to the base path and (2) how to shift the vertices occurring before that reconnection such that the throughputs remains correlated. G-MLT and G-PT solve these challenges in slightly different ways that will be discussed in detail in Sections 5.4.2 and 5.5.2.

On top of those challenges a path space shift must be defined as a bijection from the sampleable paths in Ω_p to Ω_q, which in contrast to the PSS shift is not necessarily true and hard to achieve in practice. G-PT was developed after our paper presented in Chapter 6 and uses the formalism developed there. Specifically, symmetric gradients that allow to relax the bijection requirement of the shift mappings. Therefore we will - somewhat redundantly - already discuss this symmetric gradient formulation in the next subsection.

Symmetric Gradients

Since bijective shifts T_δ from the set of sampleable paths in Ω_p to the set of sampleable paths in Ω_q are hard to realize in practice, we will relax this constraint by sampling a gradient always from two

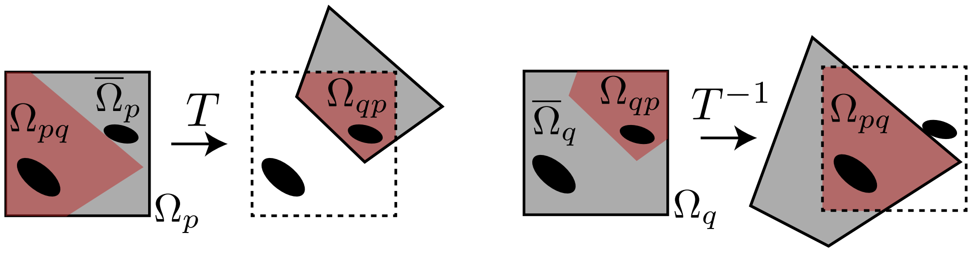


Figure 5.3: Symmetric gradients. Visualizes left the forward gradient and on the right the backward gradient. The quantities Ω_{pq}, Ω_{qp}, $\overline{\Omega}_p$, $\overline{\Omega}_q$, Ω_p and Ω_q are directly visualized. Note that the black regions are also part of $\overline{\Omega}_p$ respectively $\overline{\Omega}_q$ and visualize regions where $T^{-1}(T(\overline{\mathbf{x}})) \neq \overline{\mathbf{x}}$ respectively $T(T^{-1}(\overline{\mathbf{y}})) \neq \overline{\mathbf{y}}$. This figure is reproduced from Kettunen et al. [57].

directions: in a backwards and in a forwards computation. Specifically, we can split the computation of $\Delta_\delta(I_p)$ into computing two *directional gradients*, namely a forward and backward gradients with

$$\Delta_\delta(I_p) = \overrightarrow{\Delta}_\delta(I_p) + \overleftarrow{\Delta}_\delta(I_p). \tag{5.8}$$

The simplest possible definition of such directional gradients is

$$\begin{aligned}\overrightarrow{\Delta}_\delta(I_p) &= \frac{1}{2}\int_{\Omega_p} W_p(\overline{\mathbf{x}})g_\delta(\overline{\mathbf{x}})d\mu(\overline{\mathbf{x}}),\\ \overleftarrow{\Delta}_\delta(I_p) &= -\frac{1}{2}\int_{\Omega_q} W_q(\overline{\mathbf{x}})g_{-\delta}(\overline{\mathbf{x}})d\mu(\overline{\mathbf{x}}).\end{aligned}$$

Here the forward gradient $\overrightarrow{\Delta}_\delta(I_p)$ computes the finite difference by shifting a path in pixel p with T_δ to pixel $q = p + \delta$. The backward gradient $\overleftarrow{\Delta}_\delta(I_p)$ computes the same finite difference by sampling a path in pixel q, shifting it with the reverse shift $T_\delta^{-1} = T_{-\delta}$ to pixel p and finally flipping the sign of the gradient. Note that the quantities computed by the forward and backward gradient are identical if T is a bijection of sampleable paths and that these directional gradients will not fulfil Equation 5.8 otherwise.

In the remainder of this section we will construct directional gradients that fulfil Equation 5.8 even if the shift mapping $T : \Omega_p \to \Omega_q$ is not a bijection between sampleable paths. To that end, let us define a subset of Ω_p called the *reversible* part of Ω_p,

$$\Omega_{pq} = \{\overline{\mathbf{x}} \in \Omega_p : T^{-1}(T(\overline{\mathbf{x}})) = \overline{\mathbf{x}} \wedge p(T(\overline{\mathbf{x}})) > 0\}.$$

Intuitively, Ω_{pq} describes the subset of Ω_p where T is locally a bijection and shifts the base path to a sampleable offset path. The complementary region in Ω_p, that is $\overline{\Omega}_p = \Omega_p \setminus \Omega_{pq}$, is called the *irreversible* part of Ω_p. Analogous subsets Ω_{qp} and $\overline{\Omega}_q$ are defined for Ω_q with respect to T^{-1}. With these definitions we can modify T to operate only on the reversible parts of Ω_p and Ω_q, that is $T : \Omega_{pq} \to \Omega_{qp}$ and $T^{-1} : \Omega_{qp} \to \Omega_{pq}$. This is implemented by simply returning a dummy offset path $\overline{\mathbf{y}} = T(\overline{\mathbf{x}})$ with $f^\star(\overline{\mathbf{y}})|T| = 0$ whenever T is applied on a base path $\overline{\mathbf{x}}$ with $\overline{\mathbf{x}} \notin \Omega_{pq}$. Gradients sampled from the reversible part of the domain are called *reversible gradients*, all other gradients are called *irreversible gradients.* With these tools we redefine the directional gradients:

$$\begin{aligned}\overrightarrow{\Delta}_\delta(I_p) &= -\int_{\overline{\Omega}_p} W_p(\overline{\mathbf{x}})f^\star(\overline{\mathbf{x}})d\mu(\overline{\mathbf{x}}) + \int_{\Omega_{pq}} \omega_\delta(\overline{\mathbf{x}})W_p(\overline{\mathbf{x}})g_\delta(\overline{\mathbf{x}})d\mu(\overline{\mathbf{x}}) \\ \overleftarrow{\Delta}_\delta(I_p) &= \int_{\overline{\Omega}_q} W_q(\overline{\mathbf{x}})f^\star(\overline{\mathbf{x}})d\mu(\overline{\mathbf{x}}) - \int_{\Omega_{qp}} \omega_{-\delta}(\overline{\mathbf{x}})W_q(\overline{\mathbf{x}})g_{-\delta}(\overline{\mathbf{x}})d\mu(\overline{\mathbf{x}}).\end{aligned} \tag{5.9}$$

We will refer to $\omega_\delta(\overline{\mathbf{x}})$ as *gradient direction weights.* These weights must fulfil $\omega_\delta(\overline{\mathbf{x}}) + \omega_{-\delta}(\overline{\mathbf{y}}) = 1$ when $T_\delta(\overline{\mathbf{x}}) = \overline{\mathbf{y}}$. Note that the weights are only specified for reversible gradients. It is easy to verify that the sum of these directional gradients satisfies Equation 5.8 regardless of whether T is a bijection or not:

$$\begin{aligned}\overrightarrow{\Delta}_\delta(I_p) + \overleftarrow{\Delta}_\delta(I_p) &= \int_{\overline{\Omega}_p \cup \Omega_{pq}} W_p(\overline{\mathbf{x}})f^\star(\overline{\mathbf{x}})d\mu(\overline{\mathbf{x}}) \\ &\quad - \int_{\overline{\Omega}_q \cup \Omega_{qp}} W_q(\overline{\mathbf{x}})f^\star(\overline{\mathbf{x}})d\mu(\overline{\mathbf{x}}) \\ &= \Delta_\delta(I_p)\end{aligned} \tag{5.10}$$

With these tools it becomes much easier to design shift mappings since the mappings do not have to be bijective any more. Parts of the domain where the shift is not reversible are automatically handled by using naive gradient sampling, i.e. gradient sampling without correlation of the base and offset paths. When designing a shift mapping one should still aim at making the reversible parts of the sampling domain as large as possible, since non-reversible parts will tend to introduce more variance due to the uncorrelated base and offset path pairs.

For completeness' sake we show how the directional gradients can be sampled with Monte Carlo estimators $\overrightarrow{I}_p^\delta$ and $\overleftarrow{I}_p^\delta$:

$$\begin{aligned}\overrightarrow{\Delta}_\delta(I_p) \approx \overrightarrow{I}_p^\delta = N^{-1}\sum_i W_p(\overline{\mathbf{x}}_i)\overrightarrow{G}_\delta(\overline{\mathbf{x}}_i) \\ \overleftarrow{\Delta}_\delta(I_p) \approx \overleftarrow{I}_p^\delta = N^{-1}\sum_i W_q(\overline{\mathbf{x}}_i)\overleftarrow{G}_\delta(\overline{\mathbf{x}}_i)\end{aligned} \tag{5.11}$$

with

$$\begin{aligned}\overrightarrow{G}_\delta(\overline{\mathbf{x}}) &= \begin{cases}\omega_\delta(\overline{\mathbf{x}})g_\delta(\overline{\mathbf{x}})/p(\overline{\mathbf{x}}) & \text{if } T_\delta(\overline{\mathbf{x}}) \text{ is reversible} \\ -f^\star(\overline{\mathbf{x}})/p(\overline{\mathbf{x}}) & \text{else.}\end{cases}, \\ \overleftarrow{G}_\delta(\overline{\mathbf{x}}) &= \begin{cases}-\omega_{-\delta}(\overline{\mathbf{x}})g_{-\delta}(\overline{\mathbf{x}})/p(\overline{\mathbf{x}}) & \text{if } T_{-\delta}(\overline{\mathbf{x}}) \text{ is reversible} \\ f^\star(\overline{\mathbf{x}})/p(\overline{\mathbf{x}}) & \text{else.}\end{cases}\end{aligned} \tag{5.12}$$

5.2.3 Reconstruction

We define estimators I_p^{dx}, I_p^{dy} and I_p^c with

$$\begin{aligned} E[I_p^c] &= I_p \\ E[I_p^{dx}] &= \Delta_x(I_p) \\ E[I_p^{dy}] &= \Delta_y(I_p) \end{aligned}$$

Given these estimates for every pixel $p = (i, j)$, we can construct an image $\hat{I}$ that minimizes

$$\sum_p \left(\alpha||I_p^c - \hat{I}_p||^k + ||I_p^{dx} - \Delta_x(\hat{I}_p)||^k + ||I_p^{dy} - \Delta_y(\hat{I}_p)||^k \right). \tag{5.13}$$

We call the first term the *image fidelity* since it ensures that the colors of $\hat{I}$ are similar to the sampled color values, and the second and third term together the *gradient fidelity* since they ensure that the finite differences of $\hat{I}$ are similar to the sampled finite differences. The parameter α can be set by the user to balance the gradient- and color fidelity of the new image $\hat{I}$ with respect to the sampled data. The power k determines which type of distance we are trying to minimize.

L_2-Reconstruction

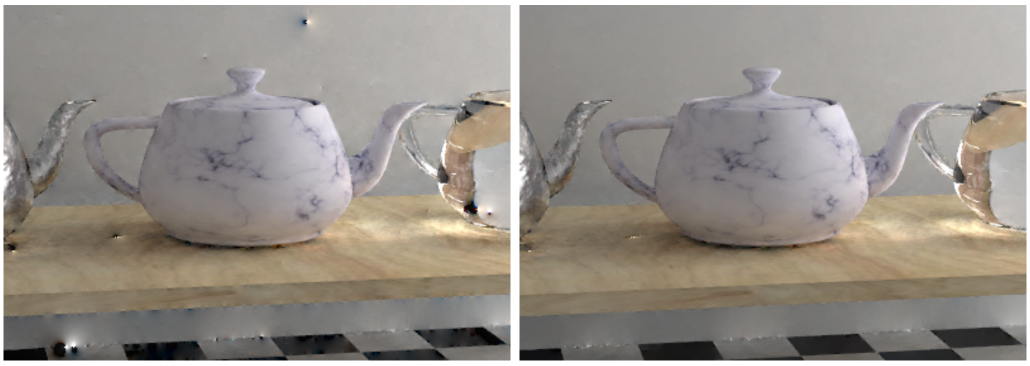
If we set $k = 2$ in Equation 5.13 then we minimize the squared Euclidean distance between the sampled data and reconstructed image. The equation system is then linear and corresponds to a screened Poisson equation. Since our system of equations is linear we can rewrite Equation 5.13 with $k = 2$ in matrix form as

$$\underset{\hat{I}}{\operatorname{argmin}} \left\| \underbrace{\begin{pmatrix} \alpha H^{Id} \\ H^{dx} \\ H^{dy} \end{pmatrix}}_{A} \hat{I} - \underbrace{\begin{pmatrix} \alpha I^c \\ I^{dx} \\ I^{dy} \end{pmatrix}}_{b} \right\|^2 \tag{5.14}$$

where I^c, I^{dx} and I^{dy} are column vectors in which the constraints for all pixels p are stacked, H^{Id} is a identity matrix and H^{dx} and H^{dy} are Laplace matrices that map an image onto its finite differences. Note that H^{Id}, H^{dx} and H^{dy} each have a dimensionality of $|p| \times |p|$ where $|p|$ is the number of pixels in the image. Thus, $A \in \mathbb{R}^{3|p|\times|p|}$, $\hat{I} \in \mathbb{R}^{|p|}$ and $b \in \mathbb{R}^{3|p|}$. Note that for an image with c color channels the $\hat{I}$ and b vectors are expanded to $\hat{I} \in \mathbb{R}^{|p|\times c}$ and $b \in \mathbb{R}^{3|p|\times c}$, which means that the equation system is solved for all color channels simultaneously.

From this follows that $\hat{I} = Sb$ with $S = (A^\top A)^{-1}A^\top$. If b is sampled in a unbiased way, i.e. $E(b) = (\alpha I^\top, \Delta_x(I)^\top, \Delta_y(I)^\top)^\top$, then it is trivial to show that $\hat{I}$ is a unbiased estimator of I because of the linearity of S:

$$E[\hat{I}] = E[Sb] = SE[b] = I.$$



(a) L_2 reconstruction (b) L_1 reconstruction

Figure 5.4: Result of the reconstruction with $k = 2$ and $k = 1$. These images were rendered with G-MLT (Section 5.5).

L_1-Reconstruction

Reconstructing images using the Poisson reconstruction with the L_2-norm is problematic since it cannot resolve contradicting information very well. For instance, if the information in I^c, I^{dx} and I^{dy} is contradicting, then the Poisson solver tries to re-conciliate this information by introducing ringing artefacts (Figure 5.4a). One way to avoid this is by using the L_1-norm in Equation 5.13, i.e. $k = 1$. The effect of the L_1-norm is that "bad" constraints that lead to large residuals are weighted down. Since large residuals occur in regions of the image where the information is contradicting, L_1 effectively prevents the solver from re-conciliating this information and thus helps avoiding the ringing and dipole artefacts (Figure 5.4b). However, in contrast to the L_2-reconstruction, the L_1-reconstruction is only consistent and not unbiased.

Implementation

The reconstruction under the L_2-norm is a linear least squares problem. This can efficiently be solved on GPUs by using a *conjugate gradient solver* (CGS) on the normal equation of the problem $A^\top A\hat{I} = A^\top b$.

The L_1 reconstruction requires minimizing $||A\hat{I} - b||$ under the L_1-norm. The solution to the L_1 optimization can be approximated with *iterative reweighed least squares* by iteratively solving a sequence of i L_2 optimizations of the form $||AW_i\hat{I}_i - W_ib||^2 = 0$ where W_i is a diagonal matrix that weights every constraint differently [27]. Each iteration i can be solved by applying CGS on $A^\top W_i^2 A\hat{I}_i = A^\top W_i^2 b$ yielding an intermediate solution $\hat{I}_i$. The algorithm is initialized with W_0 being the identity matrix, which means that every constraint is weighted equally. The result of every iteration $\hat{I}_i$ is used to update the weight matrix for the next iteration W_{i+1}. On a intuitive level W_{i+1} is set such that the residual of the previous iteration $r = A\hat{I}_i - b$ is minimized. This can be achieved by setting the jth diagonal element in W_{i+1} to $1/||r_j||$ where r_j is the jth line of r. Note that in practice some regularization must occur to prevent divisions by zero. This process is repeated up to a user defined number of maximal iterations k or until the current residual $||\hat{I}_i - b||$ is below a threshold. The last computed $\hat{I}_i$ with $i < k$ is then an approximation of the L_1 optimization.

Lehtinen et al. [68] reported that on modern GPUs the L_1 reconstruction of a mega-pixel image can be obtained in the orders of seconds with this method. Thus, compared to the rendering time for reasonably complex scenes which is usually in the orders of minutes or hours, the overhead due to the reconstruction is negligible.

5.3 Theoretical Analysis

This section describes an end-to-end analysis of the sampling and reconstruction process involved in gradient-domain rendering. We present a complete analysis that pinpoints why gradient sampling followed by Poisson reconstruction is beneficial. Under certain simplifications, the analysis predicts precisely by how much gradient-domain rendering reduces variance in each frequency compared to conventional sampling. We first study the error in gradient estimation compared to usual pixel estimation that is caused by Monte Carlo integration. Then we analyze screened Poisson reconstruction (Section 5.2.3) to understand the error distribution over frequencies of the final image[2].

5.3.1 Error analysis of Gradient Estimation

Gradient estimation involves computing and sampling path differences, Monte Carlo integration over path space, and pixel filtering. To make this problem amenable to Fourier analysis, we make the following simplifications: First, we work with 1D images to reduce clutter in the notation. Next, we assume paths are parameterized over a Cartesian hypercube, akin to the primary sample space by Kelemen et al.[53] described in Section 4.4.2, and the first path dimension is the image axis x. This means a path in this parametrization is described as $(x, \mathbf{p})$. The image contribution function is thus $f^\star(x, \mathbf{p})$. We restrict the analysis to uniform random sampling in this parametrization. We also assume that sampling is a wide-sense stationary stochastic processes. This implies that it extends over an infinite domain, which simplifies its frequency analysis because there are no boundary effects, but also means that we cannot model the restriction of the sampling grid to the unit hypercube. Assuming sampling over an infinite domain instead of restricting the samples around the non-zero support region of the integrand overestimates variance compared to practical algorithms. Finally, we use a simple shift mapping that only shifts the image coordinate by one pixel and leaves the other parameters untouched. This mapping has a unit Jacobian and determinant.

Given an image contribution function $f^\star$ defined over an image axis x and an arbitrarily long vector of path parameters $\mathbf{p}$ (Figure 5.5a), the problem is to integrate over $\mathbf{p}$ to obtain a sampled image and its gradients. We model gradient-domain rendering by first defining the shift mapping T and the corresponding path difference function g in path space (similar to the one defined in Equation 5.5). For this analysis we use a simple shift mapping $T(x, \mathbf{p}) = (x - 1, \mathbf{p})$. We omit the indices (i, j) because we employ this same shift everywhere over the image. Therefore, the path difference function is simply $g(x, \mathbf{p}) = f^\star(x, \mathbf{p}) - f^\star(x - 1, \mathbf{p})$, which we may write as a convolution $g(x, \mathbf{p}) = (d * f^\star)(x, \mathbf{p})$ with a difference operator $d(x, \mathbf{p}) = \delta(x) - \delta(x - 1)$. This is shown in

[2]The text and the figures used in this section were published earlier as part of the publication *Gradient-Domain Path Tracing* by Kettunen et al. [57] that was co-authored by the author of this thesis.

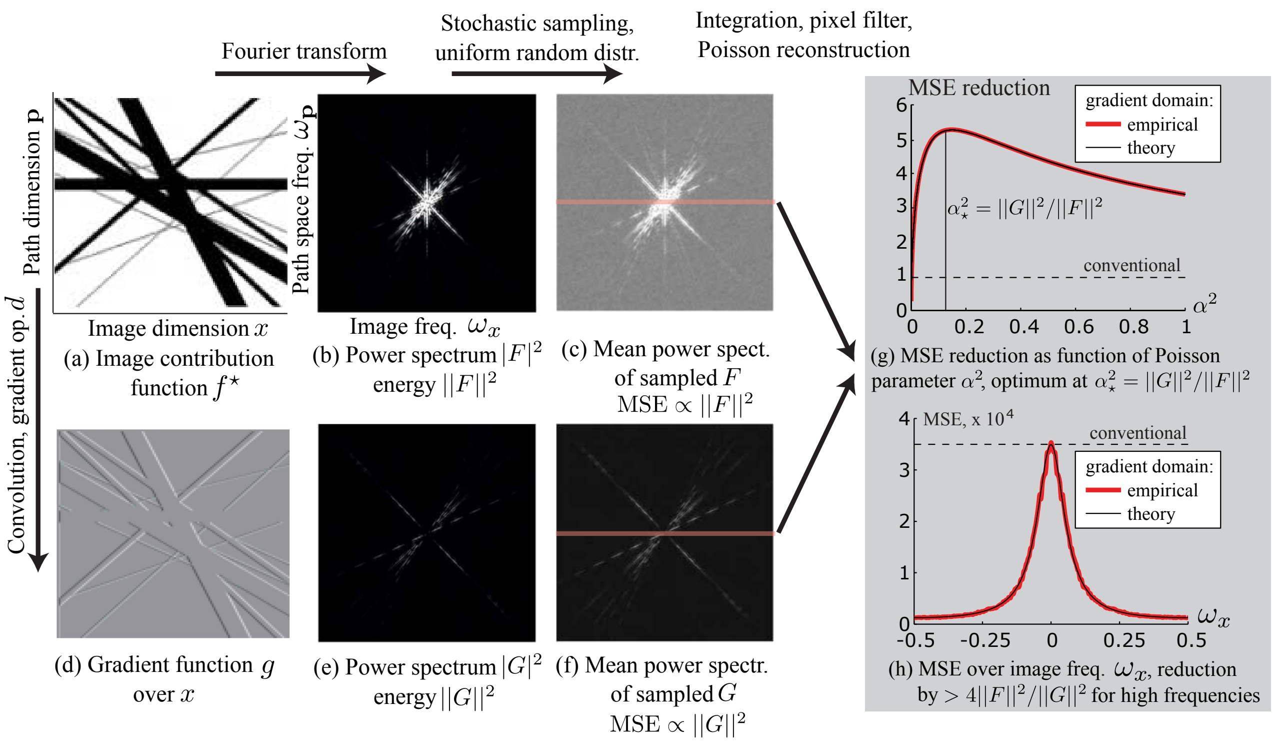


Figure 5.5: Overview of the frequency analysis (a-f) and its main results (g,h). Given an image contribution function $f^\star$ that we need to integrate over path dimension p to get an image (a), we express gradients g as a convolution of $f^\star$ with a gradient operator d in path space (d). We analyse stochastic sampling of $f^\star$ (b,c) and $g = d * f^\star$ (e,f) in the frequency domain. The error due to sampling turns out to be constant over all frequencies. It appears as a flat, gray background in (c,f), and it is typically smaller for gradients (f). Next, integration over path dimension p corresponds to slicing along the red lines in (c) and (f). Poisson reconstruction combines the two slices to form the final image using a parameter α. Our main results include the derivation of the optimal parameter α_* that leads to the highest error reduction over conventional rendering (g), and an error analysis of the final output that shows how high frequency error (close to the Nyquist frequency 1/2 of the image sampling grid) is strongly reduced (h). Empirical results of the 2D example shown here (red lines in g,h) closely match our theory.

Figure 5.5d, where we apply finite differencing horizontally along the image dimension x, but not vertically over path parameters $\mathbf{p}$.

In the Fourier domain (Figure 5.5b and 5.5e) $g = d * f^\star$ is a multiplication $G = DF$ where F and D are the Fourier transforms of $f^\star$ and d. The power spectrum of the path difference function $|G|^2$ is related to the power spectrum of the image contribution function $|F|^2$ by

$$|G(\omega_x, \omega_\mathbf{p})|^2 = (2 - 2\cos(2\pi\omega_x))|F(\omega_x, \omega_\mathbf{p})|^2 = |D(\omega_x, \omega_\mathbf{p})|^2 |F(\omega_x, \omega_\mathbf{p})|^2, \quad (5.15)$$

where ω_x and $\omega_\mathbf{p}$ are frequencies over the image and path space, respectively, and $|D|^2 = (2 - 2\cos(2\pi\omega_x))$ is the power spectrum of the finite-difference operator (as opposed to the continuous derivative, which would attenuate frequencies by $1/\omega_x^2$). We get the well known result that finite-differencing cancels out the DC, attenuates low frequencies, and boosts square magnitudes of high frequencies by a factor up to four for the Nyquist limit $\omega_x = 1/2$ of the image (unit pixel spacing).

We then analyze stochastic sampling of both the image contribution F and the path difference function G in the frequency domain (Figure 5.5c and 5.5f), and we derive the mean square error (MSE) introduced by sampling, which is equivalent to the variance. Sampling is a multiplication by a union of Diracs in the primal and a convolution in the Fourier domain, and we can interpret the error due to sampling as aliasing [17, 18]. We model uniform random sampling as a Poisson process, whose power spectrum is flat except for a Dirac at the DC [69]. As a key consequence, *the stochastic convolution results in constant expected errors $|\epsilon_F(\omega_x, \omega_{\mathbf{p}})|^2$ and $|\epsilon_G(\omega_x, \omega_{\mathbf{p}})|^2$ for all frequencies for both pixels and gradients*,

$$\begin{aligned} |\epsilon_F(\omega_x, \omega_{\mathbf{p}})|^2 &= \frac{1}{n}\|F\|^2, \\ |\epsilon_G(\omega_x, \omega_{\mathbf{p}})|^2 &= \frac{1}{n}\|G\|^2, \end{aligned} \tag{5.16}$$

which are inversely proportional to the sampling density n. In addition, the constants $\|F\|^2$ and $\|G\|^2$ are given by the total energy of the signals, that is, the integral over image and path dimensions of the power spectra in Equation 5.15,

$$\begin{aligned} \|F\|^2 &= \int |F(\omega_x, \omega_{\mathbf{p}})|^2 \mathrm{d}\omega_x \mathrm{d}\omega_{\mathbf{p}}, \\ \|G\|^2 &= \int (2 - 2\cos(2\pi\omega_x))|F(\omega_x, \omega_{\mathbf{p}})|^2 \mathrm{d}\omega_x \mathrm{d}\omega_{\mathbf{p}}. \end{aligned} \tag{5.17}$$

The MSE appears as a flat gray background in Figure 5.5c and 5.5f. The difference in brightness of the flat background indicates the difference in MSEs of the sampled signals.

Next, we model integration over path space to obtain the sampled image and its sampled gradients by slicing in the Fourier domain. After slicing, we convolve with an ideal pixel filter, which eliminates errors in frequencies above the Nyquist limit $\omega_x = 1/2$. We denote the resulting MSE of the integrated pixels $|\epsilon_F(\omega_x)|^2 = |\epsilon_F(\omega_x, 0)|^2$ and the MSE of the gradients $|\epsilon_G(\omega_x)|^2 = |\epsilon_G(\omega_x, 0)|^2$, and conclude

$$\begin{aligned} |\epsilon_F(\omega_x)|^2 &= \frac{1}{n}\|F\|^2, \quad \text{if } |\omega_x| < 1/2, \text{ otherwise } 0, \\ |\epsilon_G(\omega_x)|^2 &= \frac{1}{n}\|G\|^2, \quad \text{if } |\omega_x| < 1/2, \text{ otherwise } 0. \end{aligned} \tag{5.18}$$

Our MSEs come out as energies instead of variances here because of our sampling assumptions, which are crucial to be able to perform the derivation in simple terms. The means of our functions over the infinite sampling domain are zero, hence their energies represent their variances.

Discussion The integrals for $\|F\|^2$ and $\|G\|^2$ are the same except for the weight $|D(\omega_x)|^2 = (2 - 2\cos(2\pi\omega_x))$ introduced by finite differencing. *This reveals that the difference between the pixel error $|\epsilon_F|^2$ and gradient error $|\epsilon_G|^2$ depends on the relative amount of low and high frequencies in the image contribution function.* In the best case for gradient estimation, all the energy is in the low frequencies, and as $|D(\omega_x)|^2$ weights them down, the gradient energy can be arbitrarily smaller. At worst, all the energy is in the high frequencies and gradient estimates are four times as bad as pixel estimates. Interestingly, the spectra of typical image contribution functions appear to be favourable

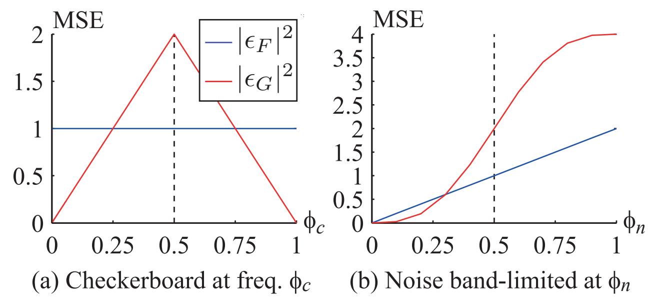


Figure 5.6: MSEs of sampled pixels $|\epsilon_F|^2$ and gradients $|\epsilon_G|^2$ for two image contribution functions, a (multidimensional) checkerboard, and band-limited noise, for different pattern frequencies ϕ_c and ϕ_n. For the checkerboard, the MSE of gradients oscillates between values zero and two, as the pattern switches between positive and negative correlation with the finite differencing stencil. The pixel MSE is independent of the checkerboard frequency, since we sample uniformly without stratification. The dotted line indicates the Nyquist frequency of the pixel grid. Even for pattern frequencies close to the Nyquist limit, gradients exhibit less MSE than pixels.

to gradient estimation: like natural images, they are dominated by sharp edges, and hence follow an inverse square power law.

In Figure 5.6 we illustrate the benefits and limitations of gradient compared to pixel sampling for two prototypical signals. The first defines the image contribution function as a (multidimensional) checkerboard pattern, and the second uses band-limited noise, both with amplitude one. Both cover an infinite domain of arbitrary dimensionality (arbitrary many path parameters $\mathbf{p}$). Dimensionality does not matter for our analysis since the difference operator $|D|^2$ does not change the frequency spectra over the path dimensions.

For the checkerboard pattern (Figure 5.6a), we compare the MSE of the pixels $|\epsilon_F|^2$ and the MSE of their gradients $|\epsilon_G|^2$ (given by the energies $\|F\|^2$, $\|G\|^2$, Equation 5.16) for different frequencies ϕ_c of the checkerboard tiles. Frequency $\phi_c = 0.5$ means the side-length of tiles is one pixel unit (two pixels for one cycle of the pattern). For the noise pattern (Figure 5.6b), we show the same comparison for different band-limits ϕ_n of the noise. Similarly, $\phi_n = 0.5$ means noise is cut off at the Nyquist frequency of the pixel grid.

In both cases, as the pattern frequency increases, their energies become less and less dominated by low frequencies. Hence the gradient MSE increases, and climbs above the pixel MSE for high frequency patterns close to the Nyquist limit. This is to be expected since $|D|^2$ amplifies the square magnitudes of these frequencies by a factor of up to four. But for frequencies not much below, gradient MSEs drop below pixel MSEs. For the square wave, the frequency where gradients have less MSE is at exactly $\phi_c = 0.25$, which corresponds to checkerboard tiles of only two pixels (four pixels per cycle). For the noise pattern, gradients have less MSE than pixels at even slightly higher noise cutoff frequencies $\phi_c \approx 0.3$. This indicates that gradient sampling is effective even for high-frequency patterns close to the Nyquist limit of the pixels.

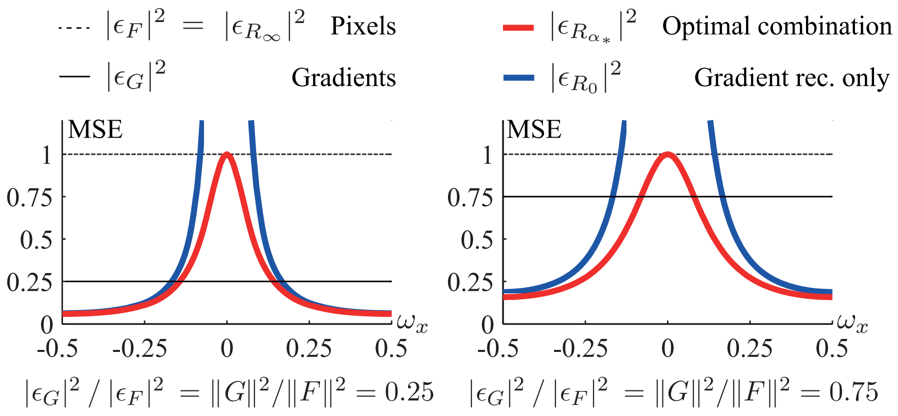


Figure 5.7: Poisson reconstruction combines noisy pixels with MSE $|\epsilon_F|^2$ and gradients with MSE $|\epsilon_G|^2$ using a parameter α, weighting the influence of these two inputs. Reconstruction $|\epsilon_{R_0}|^2$ with $\alpha = 0$ (blue lines) uses only gradients. It reduces high frequency error, but error explodes at low frequencies. At the optimal α_* (red lines), the reconstruction error is at most the pixel variance $|\epsilon_F|^2$ at low frequencies, but is smaller than $|\epsilon_F|^2$ by a factor $> 4\|F\|^2/\|G\|^2$ at high frequencies. The signal on the left corresponds to a checkerboard pattern with a tile size of eight pixels ($\|G\|^2/\|F\|^2 = 0.25$ corresponds to $\phi_c = 0.0625$, and $1/(2 \times 0.0625) = 8$). The signal on the right corresponds to a tile size of about 2.5 pixels.

5.3.2 Analysis of Screened Poisson Reconstruction

We conclude the end-to-end analysis by deriving the spectral MSE of the final image obtained through screened Poisson reconstruction. Screened Poisson reconstruction combines the image and its gradients using a parameter α that specifies the relative weights of the sampled image and the gradients. We explicitly include the MSEs of the input image pixels $|\epsilon_F|^2 = \|F\|^2/n$ and their gradients $|\epsilon_G|^2 = \|G\|^2/n$ (Equation 5.16) in our analysis and derive the per-image frequency reconstruction error

$$|\epsilon_{R_\alpha}(\omega_x)|^2 = \frac{1}{n}\frac{\alpha^4\|F\|^2 + |D(\omega_x)|^2\|G\|^2}{(\alpha^2 + |D(\omega_x)|^2)^2}. \tag{5.19}$$

Discussion To clearly understand the benefits of Poisson reconstruction, we plot the frequency-dependent reconstruction error $|\epsilon_{R_\alpha}(\omega_x)|^2$ for different values α (Figure 5.7). If we set $\alpha = \infty$, we consider only the pixel information and the reconstruction error becomes equivalent to the pixel error $|\epsilon_{R_\infty}|^2 = \|F\|^2/n$. More interesting is setting $\alpha = 0$, which means we consider only gradients. Then the reconstruction error amounts to

$$|\epsilon_{R_0}(\omega_x)|^2 = 1/n \cdot \|G\|^2/|D(\omega_x)|^2 = 1/n \cdot \|G\|^2/(2 - 2\cos(2\pi\omega_x)). \tag{5.20}$$

This confirms that gradient rendering is most beneficial for high frequencies and has a singularity for the DC. The image frequency where it may become beneficial depends on the relative energy of the path difference function and the image contribution function, $\|G\|^2/\|F\|^2$. In the worst case, we saw that $\|G\|^2$ is four times bigger than $\|F\|^2$, and the factor four gets canceled by the denominator

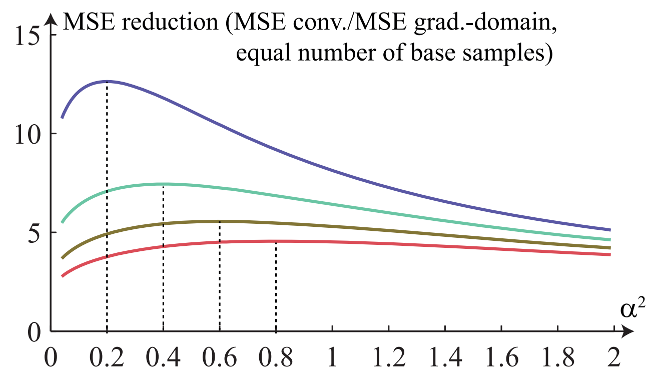


Figure 5.8: Error reduction of gradient sampling and reconstruction for 2D images. We plot the factor by which gradient sampling and reconstruction reduces the total MSE compared to conventional sampling and integration at equal number of base samples. We show different energy ratios $\|G\|^2/\|F\|^2 \in \{0.2, 0.4, 0.6, 0.8\}$, corresponding to the curves from top to bottom (purple to red). Note that each curve achieves its maximum at $\alpha^2 = \|G\|^2/\|F\|^2$.

in Equation 5.20 at the Nyquist limit. Compared to direct pixel rendering, the gradient reconstruction error therefore is the same for the highest frequency and worse for lower frequencies. As discussed above, typical image contribution functions tend to follow an inverse square power law. Hence finite differencing reduces their energies significantly and gradient reconstruction is beneficial for much of the spectrum except very low frequencies.

A key observation about Equation 5.20 is that the factor $2 - 2\cos(2\pi\omega_x)$ in the denominator is the inverse of the finite differencing operator we applied in the beginning to sample path differences. Hence, both finite differencing and Poisson reconstruction correspond to a per-frequency weighting of the power spectrum by a cosine function, which appears both in the numerator and the denominator in Equation 5.20. However, the factors do not cancel each other out, because the error of sampling finite differences is hidden in the integral of the gradient energy $\|G\|^2$ (Equation 5.17) due to the frequency scrambling or aliasing caused by stochastic sampling. It results in a lower energy and lower sampling error for gradients of functions with a frequency falloff. In contrast, the reconstruction affects the denominator and applies the inverse weight to the final image, which explains why gradient solutions reduce high frequency error.

Optimal Reconstruction It is easy to derive the value $\alpha_*(\omega_x)$ at each frequency that optimally combines pixels and gradients as $\alpha_*^2(\omega_x) = \|G\|^2/\|F\|^2$. It is interesting to plot the per frequency error at the optimal value α_*, which turns out to be

$$|\epsilon_{R_{\alpha_*}}(\omega_x)|^2 = \frac{1}{n}\frac{\|G\|^2\|F\|^2}{\|F\|^2|D(\omega_x)|^2 + \|G\|^2},$$

shown as red lines in Figure 5.7. At low frequencies, the reconstruction MSE approaches the pixel MSE $\|F\|^2/n$, but it falls off quickly. For high frequencies approaching $\omega_x = 0.5$ we take full advantage of the gradients and the MSE goes below $\|G\|^2/(4n)$.

It is important to understand that so far we compared conventional sampling with n samples with gradient-domain rendering with n conventional *and* n gradient samples. In this setup gradient-domain rendering at optimal α_* cannot be worse than conventional rendering. However, at equal number of individual path samples (counting each gradient as two path samples), gradient-domain rendering becomes ineffective if $\|F\|^2/\|G\|^2 < 1$. In addition, The algorithms discussed in Sections 5.4 and 5.5 obtains correlated gradient and conventional samples, and as a consequence, the optimal α_* is not applicable in practice.

Main Insights Our complete analysis leads to two main insights: first (Figure 5.5g), under our simplifications, the optimal reconstruction parameter is given by the ratio of the total energies of the gradient function and the image contribution function, $\alpha_*^2 = \|G\|^2/\|F\|^2$; second (Figure 5.5h), gradient-domain rendering reduces high frequency variance of the reconstructed output compared to conventional sampling by more than $4\|F\|^2/\|G\|^2$.

Finally, we investigate the reduction in total (as opposed to per-frequency) MSE of gradient-domain over conventional rendering at equal number of base path samples, for 2D images. That is, we compare the MSE of conventional rendering with n samples to gradient-domain rendering with n base paths, and in addition, one horizontal and vertical offset path. For this we numerically integrate a 2D version of the per-frequency reconstruction error in Equation 5.19 over all image frequencies. We plot the resulting ratio of the total error of conventional compared to gradient-domain rendering in Figure 5.8 for various ratios energy ratios $\|G\|^2/\|F\|^2 \in \{0.2, 0.4, 0.6, 0.8\}$ over values $\alpha \in (0, 1)$. For each ratio, the MSE reduction is best when that ratio is used as α, as described above.

5.4 Gradient-Domain Path Tracing

Gradient-domain rendering can be applied on path tracing by computing gradients alongside the color image, yielding the gradient-domain path tracing algorithm (G-PT) [57]. In short, for every sampled path $\bar{\mathbf{x}}$ for pixel (i, j) we compute four offset paths to the adjacent pixels by using a shift mapping T:

$$T_{0,1}(\bar{\mathbf{x}}),\ T_{0,-1}(\bar{\mathbf{x}}),\ T_{1,0}(\bar{\mathbf{x}}) \text{ and } T_{-1,0}(\bar{\mathbf{x}}). \tag{5.21}$$

We use these additional paths to sample the following four directional gradients:

$$\overrightarrow{\Delta}_{0,1}(I_{i,j}),\ \overleftarrow{\Delta}_{0,1}(I_{i,j-1}),\ \overrightarrow{\Delta}_{1,0}(I_{i,j}) \text{ and } \overleftarrow{\Delta}_{1,0}(I_{i-1,j}). \tag{5.22}$$

Figure 5.9 visualizes the relationship between these terms. The directional gradient samples will be correlated since they all use the same base path, further the base path is directly used to get a coarse estimate of the pixel color $I^c_{i,j}$. This introduces correlation between the coarse color and gradient images. However, the fact that the coarse image comes essentially for free amortizes the need of more samples due to this correlation [57]. The directional gradient samples can be accumulated in two buffers, one storing the x-gradient and one storing the y-gradient

$$\begin{aligned} \Delta_x(I_{i,j}) &= \overrightarrow{\Delta}_{0,1}(I_{i,j}) + \overleftarrow{\Delta}_{0,1}(I_{i,j}), \\ \Delta_y(I_{i,j}) &= \overrightarrow{\Delta}_{1,0}(I_{i,j}) + \overleftarrow{\Delta}_{1,0}(I_{i,j}). \end{aligned} \tag{5.23}$$

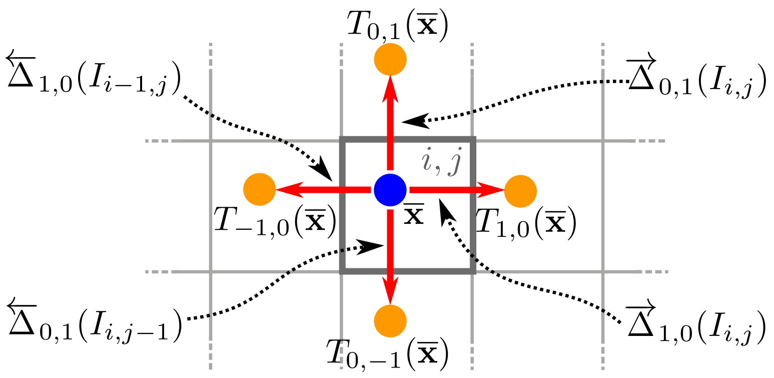

$T_{0,1}(\bar{\mathbf{x}})$
$\overleftarrow{\Delta}_{1,0}(I_{i-1,j})$
$\overrightarrow{\Delta}_{0,1}(I_{i,j})$
i,j
$T_{-1,0}(\bar{\mathbf{x}})$
$\bar{\mathbf{x}}$
$T_{1,0}(\bar{\mathbf{x}})$
$\overleftarrow{\Delta}_{0,1}(I_{i,j-1})$
$T_{0,-1}(\bar{\mathbf{x}})$
$\overrightarrow{\Delta}_{1,0}(I_{i,j})$

Figure 5.9: A visualization in image-space of the base an offset paths used in G-PT. Given a base path $\bar{\mathbf{x}}$ (blue) in pixel (i, j), $\bar{\mathbf{x}}$ it is shifted to the four adjacent pixels yielding four offset paths (yellow). The base and offset paths are used to compute four directional gradients (red).

The data can then be directly used to reconstruct an image according to Section 5.2.3. The authors recommend using an α value of 0.2 [57].

In the remainder of this section we will first discuss how an MIS estimator can be defined for gradients between their computation directions. Then we will describe in detail a simple shift mapping that yields strong correlation between base and offset paths that copies half-vectors until a reconnection to the base path is possible.

5.4.1 MIS for Gradients

When sampling a gradient as described in Equation 5.6 the PDF of the underlying sampler is used. In the case of G-PT this is the PDF of the base path, $p(\bar{\mathbf{x}})$, sampled with a path tracer. In path tracing, the density $p(\bar{\mathbf{x}})$ importance samples $f^\star(\bar{\mathbf{x}})$ (Section 4.2.2). However, this does not imply that $p(\bar{\mathbf{x}})$ is good at importance sampling the gradient $f^\star(T(\bar{\mathbf{x}}))|T| - f^\star(\bar{\mathbf{x}})$.

This can be seen by observing what happens when the path sampling density form the base path $\bar{\mathbf{x}}$ to the offset path $T(\bar{\mathbf{x}})$ increases: to correct for the fact that $T(\bar{\mathbf{x}})$ is sampled with $p(\bar{\mathbf{x}})$ the Jacobian determinant $|T|$ must be large. Unfortunately, when approaching concave edges or caustic manifolds this change of density and thus the Jacobian determinant can become arbitrary large. The value $f^\star(T(\bar{\mathbf{x}}))|T|$ is thus unbound which potentially leads to large variance in the gradient samples due to outliers. Note that this variance can become much bigger than the variance of ordinary pixel sampling. In the L_2 reconstruction these outliers can lead to dipole artefact (recall Figure 5.4a). Interestingly the reverse case when the density from base to offset path decreases is much less problematic. In the limit the offset paths contribution then becomes zero and we are left with a variance that will never be worse than the variance of naive gradient sampling.

Ways to avoid these worst case scenarios by combining different strategies for different parts of path space have been described in [75]. However, it turns out that for gradient-domain path tracing there is a much simpler way to weight down these problematic cases: we use MIS over the gradient sampling directions.

Recall, that Equation 5.9 and Equation 5.6 reveal that there are two ways to sample reversible gradient obtained from a shift mapping $T(\bar{\mathbf{x}}) = \bar{\mathbf{y}}$ with $x \in \Omega_{pq}$ and $\bar{\mathbf{y}} \in \Omega_{qp}$; we can use forward gradient sampling

$$W_p(\overline{\mathbf{x}})\frac{f^\star(T(\overline{\mathbf{x}}))|T| - f^\star(\overline{\mathbf{x}})}{p(\overline{\mathbf{x}})}$$

and backward gradient sampling

$$W_q(\overline{\mathbf{y}})\frac{f^\star(\overline{\mathbf{y}}) - f^\star(T^{-1}(\overline{\mathbf{y}}))|T^{-1}|}{p(\overline{\mathbf{y}})} = W_p(\overline{\mathbf{x}})\frac{f^\star(T(\overline{\mathbf{x}}))|T| - f^\star(\overline{\mathbf{x}})}{p(T(\overline{\mathbf{x}})|T|}.$$

It is easy to see that both strategies indeed sample the same gradient but with different PDFs. With this in mind we can easily define the gradient direction weights in Equation 5.9 to be MIS weights that use the balance heuristic (see Section 3.3.4):

$$\omega_\delta(\overline{\mathbf{x}}) = \frac{p(\overline{\mathbf{x}})}{p(\overline{\mathbf{x}}) + p(T(\overline{\mathbf{x}}))|T|}.$$

There are some subtleties to consider when implementing this scheme. Recall that for every base path we compute four offset path (Equation 5.22), and that each offset path is used to compute a different directional gradient (Equation 5.23). Thus during sampling we are always computing either the forward or backward directional gradient of a specific gradient but never both. This means we are using the one-sample model described in the Veach Thesis in Section 9.2.4 [106]. Consequently we must consider the probability of choosing one of the strategies. We denote the probability of using forward respectively backward direction for sampling gradient $\Delta_\delta(I_p)$ with $\overrightarrow{c}_p^\delta$ respectively $\overleftarrow{c}_p^\delta$. In a path tracer with uniform sample distribution over the pixels this probability is always $\overrightarrow{c}_p^\delta = \overleftarrow{c}_p^\delta = 0.5$. This comes from the fact that for an estimand $\Delta_\delta(I_p)$ as many samples are computed in the forward direction from pixel (p) as are computed in the backwards direction from pixel $q = p + \delta$. This factor must be multiplied with the used PDF which effectively means that we must multiply reversible gradients by a factor of 2 when using MIS. A sample of the forward gradients throughput (see Equation 5.11) for a reversible gradient using MIS is of the form

$$\overrightarrow{G}_\delta(\overline{\mathbf{x}}) = \frac{1}{\overrightarrow{c}_p^\delta}\frac{f^\star(T(\overline{\mathbf{x}}))|T| - f^\star(\overline{\mathbf{x}})}{p(\overline{\mathbf{x}}) + p(T(\overline{\mathbf{x}}))|T|} \tag{5.24}$$

and analogously for the backward gradient. Renderers with non-uniform sample densities over the image plane need to adapt the weights appropriately. In these cases the strategy selection probabilities become

$$\begin{aligned}\overrightarrow{c}_p^\delta &= |n|_p/(|n|_p + |n|_q)\\ \overleftarrow{c}_p^\delta &= |n|_q/(|n|_p + |n|_q)\end{aligned}$$

where $|n|_p$ respectively $|n|_q$ denotes the number of samples distributed in pixel p respectively q.

5.4.2 Half-Vector Shift

The key of the shift mapping used for G-PT is the observation that a path parametrized by its half-vectors yields smoother gradients of the throughput with respect to changing parameters than

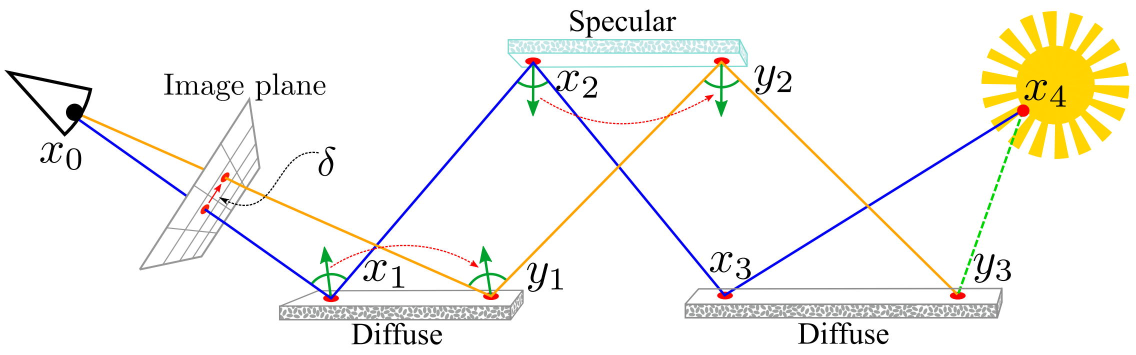


Figure 5.10: Half vector shift. The half-vectors (green arrows) of the base path (blue) are copied to trace the offset path (yellow) until two consecutive diffuse vertices are encountered. In this example the half-vectors of x_1 and x_2 are copied to y_1 and y_2 to trace new rays, since x_2 and y_2 are on a specular surface. The reconnection is thus delayed upon the emitter vertex x_4.

when surface or direction parametrizations are used [45] [52]. In a nutshell this means that paths with similar half-vectors usually have similar throughputs, more so than paths with similar positions or outgoing directions. This stems from the fact that the throughput is often dominated by the BSDF when we have glossy interactions.

A path space shift mapping should reconnect the offset path back to the base path after as few vertices as possible, since this reduces the computation costs of performing the shift and is an easy way to ensure parts of the base and offset path have strong correlation. However, it is crucial not to perform the connection on a segment that involves (near-)specular materials since this would change local BSDF evaluations strongly. G-PT therefore uses a scheme that performs the reconnection as soon as a segment is reached where no (near-)specular vertices are involved. Whether or not a segment is suitable for reconnection depends on a simple roughness threshold on the involved materials. As long as reconnection is not appropriate, the mapping samples a new direction on the current shifted vertex such that the local tangent-space projection of the half-vector of the base path is preserved.

In detail the mapping $T_\delta(\overline{\mathbf{x}}) = \overline{\mathbf{y}}$ works as follows:

1. Shift the primary ray by δ in image space. This yields a shifted primary vertex y_1.

2. If x_i or x_{i+1} are classified as specular we continue tracing the path. The outgoing direction from y_i is determined by choosing it such that the projected half-vector at y_i is identical to the projected half vector of the base path at x_i. Tracing the ray yields y_{i+1}. We always check for reversibility of the mapping when doing so.

3. If x_i,x_{i+1} and y_i are all not classified as specular we connect y_i with x_{i+1}. Note that if the segment y_i to x_{i+1} is occluded, then our shift would not be reversible and we do not need to do any further computation. If it is not occluded, we can copy the remainder of the path., i.e. $y_k = x_k$ for $i+1 < k \leq \text{len}(\overline{\mathbf{x}})$.

See Figure 5.10 for an example of the shift applied on a *EDSDL* path. Note that checks for reversibility should always be performed as soon as possible to provide early exists in the offset path computation. For instance in step 2, as soon as the path configuration of specular and non-specular vertices of the offset path differ from the base path, the computation of the offset path can be aborted immediately. Similarly, we also abort computation whenever refractions in the base path lead to total internal reflection in the offset path because of the change of incoming direction.

As a technical detail one must take special care of cases when the offset path is connected to an area light source, since then four strategies exist to sample this gradient: the base path could have been generated by NEE or BSDF sampling and for each of these possibilities the gradient can be computed by backwards or forwards gradient computation. In G-PT this case is handled by performing MIS between all these four strategies [57].

One advantage of this shift mapping is that at any time it only requires local information about the previous, current and next base vertex, therefore the offset-path can be constructed at the same time as the base-path is constructed. This allows for simple integration in most existing path tracers where full paths are never stored for efficiency. We imagine that this could be crucial for future implementations of G-PT on GPU path tracers. For rendering systems that store all the information of a path, alternative shift mappings like the ones used in G-MLT [68] or G-BDPT (Chapter 7) could be used.

5.4.3 Results

Gradient-domain path tracing shifts each base path to its four vertical and horizontal neighbours, resulting in standard finite difference gradient estimates. As discussed above this means that we compute four offset paths for each base path. Since the half-vector shift typically connects offset paths back to base paths after tracing only a few new offset path segments, the computational cost of offset paths is significantly lower than for base paths. In practice, we observe an overhead of about a factor 2.5 between G-PT with n base and $4n$ offset samples per pixel compared to standard path tracing with n samples per pixel.

Figure 5.11 compares G-PT to standard path tracing. Diffuse scenes like SPONZA favour G-PT the most, because the shift mapping always connects to the second base vertex (excluding the eye) and BSDF values stay constant under the shift. Scenes with many glossy surfaces like KITCHEN are more challenging because they lead to higher path differences and noisier gradients, but G-PT still achieves a significant improvements. The close-ups show results at equal number of base samples. Because of the 2.5 times overhead of G-PT, the images should be compared to the next PT image diagonally down to the right for an approximately equal time comparison (slightly skewed in favour of PT, since PT has $4/2.5 = 1.6$ times longer to compute these images). The close-ups reveal how G-PT effectively reduces high frequency noise without blurring texture detail or geometric edges, which is most apparent in SPONZA and BOOKSHELF. Even in scenes that are notoriously hard to render for PT, like the DOOR scene that is entirely illuminated by a light source in another room, G-PT provides a significant advantage. However, it cannot avoid artefacts and outliers due to the underlying unidirectional path sampling strategy.

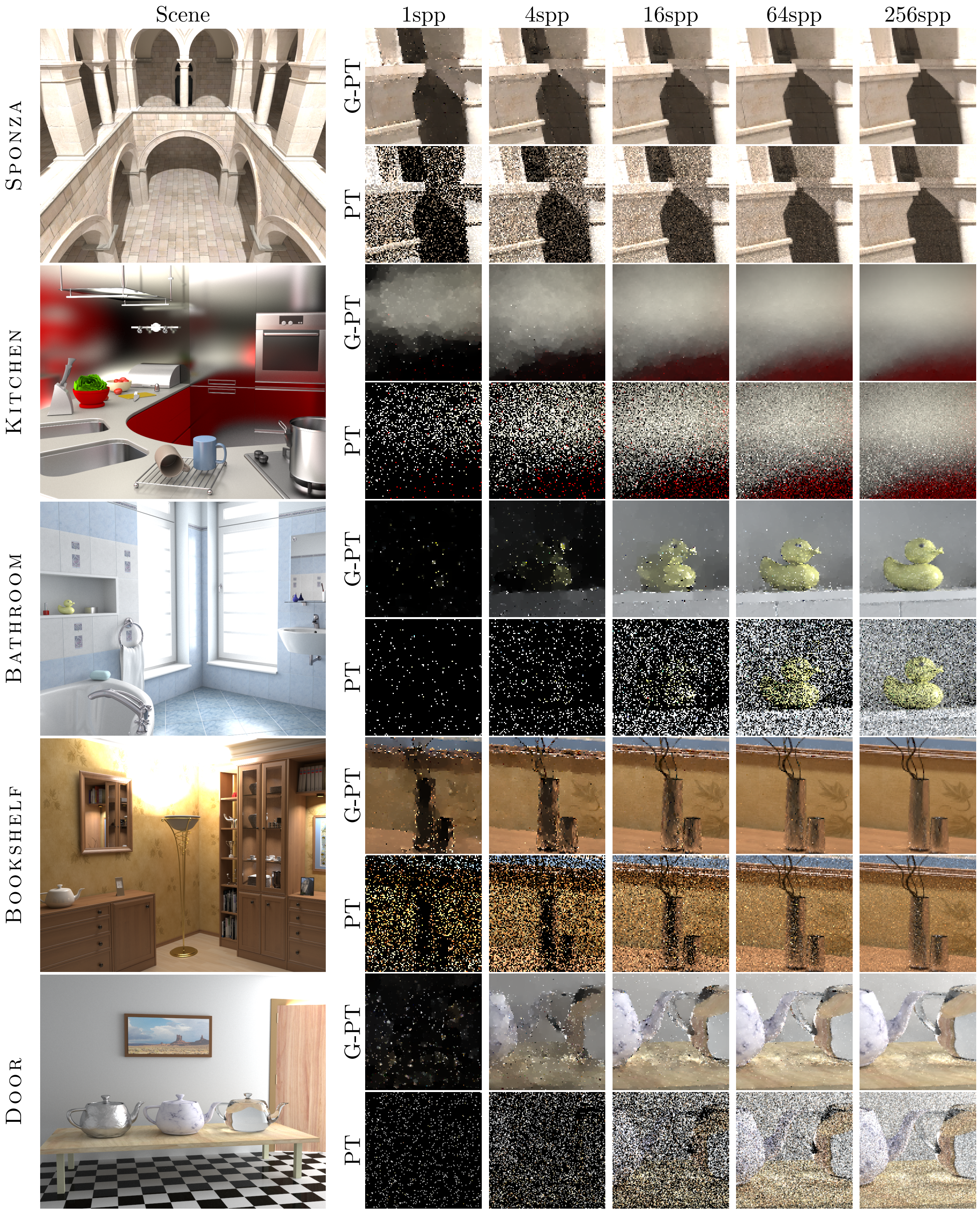


Figure 5.11: We compare standard path tracing (PT) to gradient-domain path tracing (G-PT) on five scenes. Counting only base path samples, G-PT has a performance overhead of a factor 2.5 to compute four offset paths. Hence the close-ups of G-PT (with L_1 reconstruction) should be compared to PT diagonally down to the right for an approximately equal time comparison, although PT takes $4/2.5 = 1.6$ times longer to render the corresponding images. G-PT effectively reduces high frequency noise without blurring details.

5.5 Gradient-Domain Metropolis Light Transport

G-PT can use an adaptive sampling scheme that concentrates sampling-efforts in regions where the variance of the gradients is big (Chapter 9). Doing so however suffers from the same problems as ordinary path tracing does with adaptive sampling. Namely, that adaptive sampling can introduce bias, that the sampling densities are based on heuristics and that adaptivity can only be achieved up to a certain granularity of the path space (Section 3.3.2). An alternative is MLT that can perform importance sampling of the throughput (Section 3.4). Analogously, one can combine MLT with gradient-domain rendering to allow for importance sampling the finite differences of the throughput.

Ironically, although being based on a much more complicated algorithm than G-PT, gradient-domain Metropolis light transport (G-MLT) was introduced before G-PT and was the first method to describe the principles of gradient-domain rendering [68]. Compared to G-PT, there are two major differences to consider: first, the sampling process of gradients is based on MLT and second, the shift-mapping is based on manifold walk exploration [45]. We will only discuss the basic G-MLT algorithm here, an improved algorithm is described in Chapter 6.

5.5.1 Gradient Sampling with MLT

G-MLT uses the same mutation strategies as standard MLT. Mutation strategies are performed always on the base path $\overline{\mathbf{x}}$. In contrast to G-PT, G-MLT does not sample all four adjacent gradients at once. Instead, during each mutation one of the four offset paths from Equation 5.22 is selected according to a random variable $r \in [0, 1]$. Therefore, we are always computing one of the directional gradients from Equation 5.23 at a time. To simplify notation in the remainder of this section, we generalize the notation of the G-terms in Equation 5.11 to incorporate this random choice:

$$G(\overline{\mathbf{x}}, r) = \begin{cases} \overrightarrow{G}_{1,0}(\overline{\mathbf{x}}) & \text{if } r < 0.25 \\ \overleftarrow{G}_{1,0}(\overline{\mathbf{x}}) & \text{if } 0.25 \leq r < 0.5 \\ \overrightarrow{G}_{0,1}(\overline{\mathbf{x}}) & \text{if } 0.5 \leq r < 0.75 \\ \overleftarrow{G}_{0,1}(\overline{\mathbf{x}}) & \text{if } 0.75 \leq r \end{cases}$$

Note that the gradient direction weights $\omega(\overline{\mathbf{x}})$ used in the definition of G in Equation 5.12 are always set to 0.5.

The key point of G-MLT is to use a target function for the Metropolis sampler that is proportional to the finite differences of the throughput instead of the throughput itself. In this case, the sampler will be driven towards sampling paths that create large finite differences. Instead of sampling according to the luminance of the throughput as in standard MLT, the authors suggested to sample according to a mix of the luminance of the throughput and the luminance of the finite differences

$$L(\overline{\mathbf{x}}, r) = \frac{\alpha}{4}\text{Lum}(f^{\star}(\overline{\mathbf{x}})) + \text{Lum}(G(\overline{\mathbf{x}}, r)), \tag{5.25}$$

where α is a factor that performs a trade-off between sampling according to the throughput and according to the gradients throughput. The authors suggest to use the same value as for the Poisson reconstruction, i.e. using $\alpha = 0.2$ [68]. The factor 1/4 comes from the fact that every base path can be sampled as part of 4 different gradients. Thus the base path is sampled 4 times more often than each of the finite differences that involve this base path. The fact that we always compute only one gradient direction instead of four like in G-PT might seem wasteful at first, but it enables smaller granularity in which gradients are actually sampled. This is useful in situations with axis aligned edges when the x- and y-gradients at a pixel have very different magnitudes. With a G-PT-like sampling scheme it would only be possible to either sample all gradients around a pixel or none. In constrast, with the G-MLT sampling scheme it is possible to really only focus on gradients of one axis and thus save computation time. This benefit gets amplified by the fact that G-MLT uses a shift mapping that is in general more expensive to compute than the half-vector shift from G-PT.

The gradients are then accumulated into buffers estimating the same four quantities as G-PT in Equation 5.22 by accumulating $W_k(\overline{\mathbf{x}})G(\overline{\mathbf{x}}, r)/L(\overline{\mathbf{x}}, r)$ with $k = p$ or $k = p + \delta$ depending on r into the associated directional gradient buffer. The buffer values are finally rescaled by b/N where N is the number of mutations per pixels and $b = \int L(\overline{\mathbf{x}}, r)$. Note that in order to compute b and in order to start the Markov-chains with samples taken from the distribution $L(\overline{\mathbf{x}}, r)$, the initial sampling pass needs to already compute finite differences. Besides computing finite differences during sampling, we will use the base paths to estimate I^c. Gradients are then stored in the corresponding gradient buffers similarly to how it is done in G-PT. To this end we accumulate them in an additional buffer

$$I_p^c = \frac{b}{4N} \sum_i^N \frac{W_p(\overline{\mathbf{x}}) f^\star(\overline{\mathbf{x}})}{L(\overline{\mathbf{x}}, r)}.$$

5.5.2 Manifold Exploration Shift

One of the crucial differences of MLT compared to PT from an implementation point of view is that PT never stores the full paths. Instead, it accumulates the throughputs and PDFs along the paths generation. This is the main reason why the shift mapping used in G-PT uses only local information of the path. For G-MLT we do not have these constraints since paths must always be stored as a whole in order to perform path space perturbations. Lehtinen et al. [68] thus suggested using a shift mapping that can use the global information of a path.

At the time when we want to compute $T(\overline{\mathbf{x}})$ we already know the full configuration of path $\overline{\mathbf{x}}$. Given such a path, Lehtinen et al. [68] designed a shift mapping that only modifies the prefix of the path of the form $ES^*DS^*(D|L)$. Note that the shift of G-PT modifies a prefix of the form $ES^*(DS^+)^*D(D|L)$ which can be much longer. The used shift mapping is essentially the manifold exploration perturbation (see Section 4.4.2) with a predefined deterministic perturbation that shifts the first diffuse vertex along the path such that the offset path contributes to the adjacent pixel.

To formalize the shift, we denote the sensor vertex with x_a, the first diffuse vertex along the path with x_b and the second diffuse vertex along the path with x_c. We call $x_a, ..., x_b$ the first specular chain and $x_b, ..., x_c$ the second specular chain, and $x_c, ..., x_n$ the suffix of path $\overline{\mathbf{x}}$. The shift works as follows:

1. Shift the primary ray by δ in image space. Propagate the shift through all vertices of the first specular chain, yielding a new specular chain $x_0, y_1, ..., y_b$.

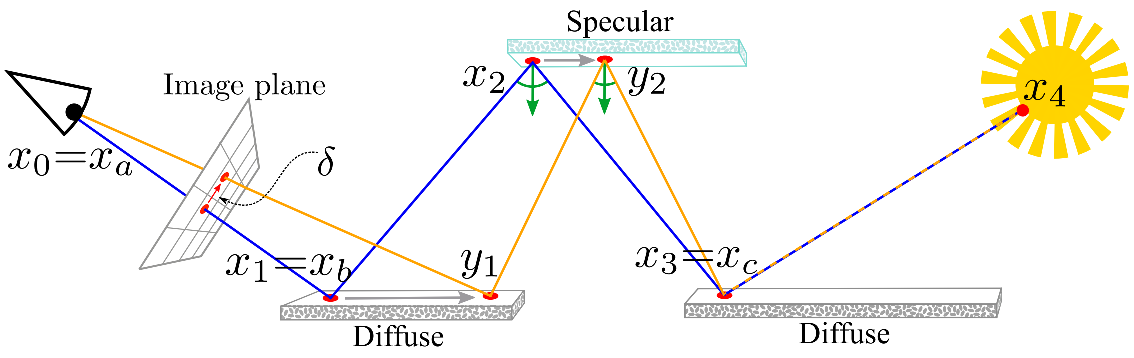


Figure 5.12: Manifold exploration shift. This figure uses the same color coding as Figure 5.10 and also the same base path configuration. The specular chain x_1, x_2, x_3 of the base path is replaced by a new specular chain y_1, y_2, x_3 for the offset path. y_2 is found by performing a manifold exploration on the invalid chain y_1, x_2, x_3. Note that compared to Figure 5.10 the manifold exploration shift allows the offset path to reconnect earlier to the base path (vertex x_3 instead of x_4).

2. If $x_{b+1} = x_c$ directly attempt to connect y_b with x_c. Otherwise perform a manifold walk on the second specular chain $x_b, ..., x_c$ where x_b is shifted to y_b which yields $y_b, ..., y_{c-1}, x_c$.

The full offset path is then $\bar{\mathbf{y}} = x_0, y_1, ..., y_{c-1}, x_c, ..., x_n$. See Figure 5.12 for an example.

5.5.3 Results

Figure 5.13 compares G-MLT to manifold exploration MLT [45]. G-MLT can concentrate efforts on regions of the image where the gradients are large and is able to resolve smooth parts of the image with less samples than MEMLT. Using the same rendering time, G-MLT thus leads to images that are much smoother than MEMLT. However, similar to G-PT, G-MLT inherits the sampling behaviour from its underlying sampler that is used to generate the base paths. In Figure 5.13 the base paths of G-MLT are sampled with MEMLT. On one hand this means that G-MLT is good at sampling difficult specular paths that work well with MEMLT. For instance the caustic of the Glass-Egg scene is resolved rather quickly with G-MLT and would take much longer to be resolved with G-PT. On the other hand G-MLT also suffers from problematic convergence behaviour of all MCMC based methods that make them impractical for animation rendering (Section 4.4.3). Additionally, G-MLT has problems in resolving regions around concave edges correctly. This effect is most prominent in the crop of the Sponza scene. This comes from the fact that shift mappings close to concave edges can lead to large Jacobian determinants $|T|$ and that in contrast to G-PT, G-MLT performs no MIS between the gradient sampling directions that could mitigate this problem (Section 5.4.1). In the next chapter we will introduce methods for G-MLT to mitigate some of these problems.

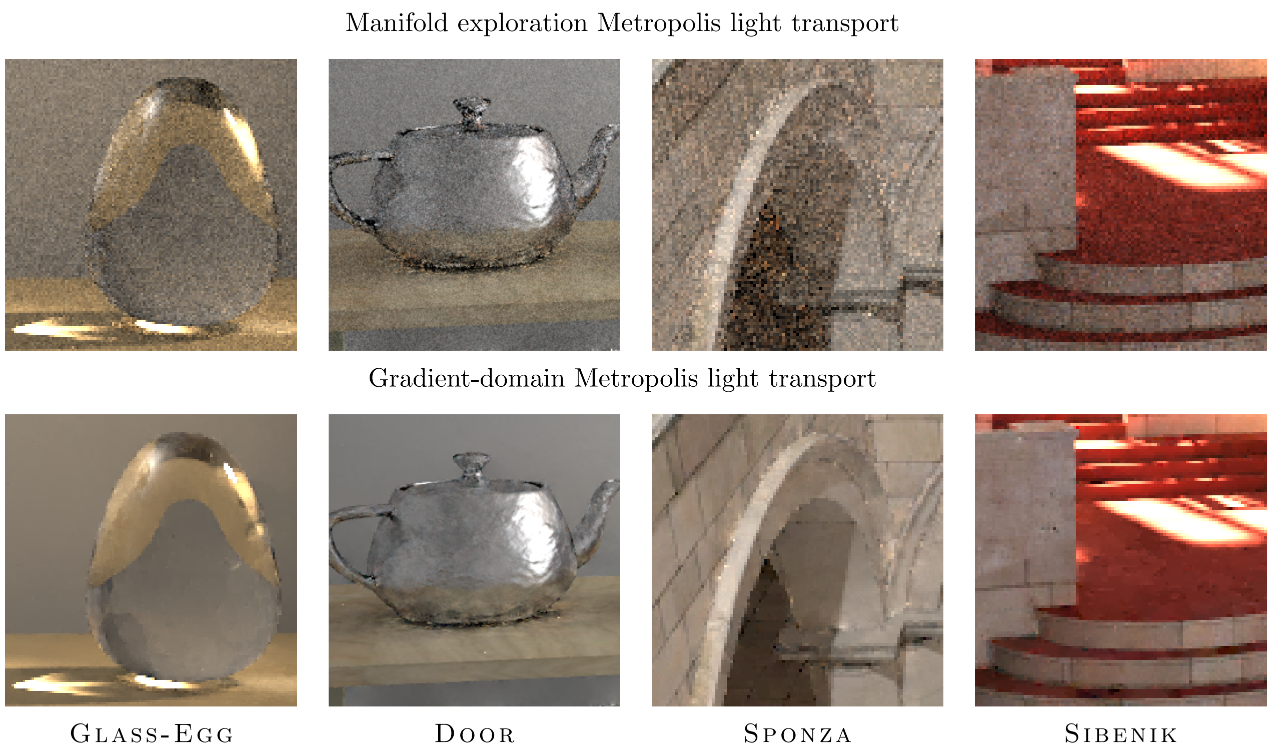

Manifold exploration Metropolis light transport
Gradient-domain Metropolis light transport
Glass-Egg
Door
Sponza
Sibenik

Figure 5.13: Comparison of MEMLT [45] to G-MLT [68] in different scenes. The top and bottom images used roughly equal time to render.

Chapter 6

Improved Sampling for Gradient-Domain Metropolis Light Transport

Published as:
Improved Sampling for Gradient-Domain Metropolis Light Transport
Marco Manzi, Fabrice Rouselle, Markus Kettunen, Jaakko Lehtinen and Matthias Zwicker.
The definitive version of record is published in ACM Transactions on Graphics, 33(6),
http://doi.acm.org/10.1145/2661229.2661291

Improved Sampling for Gradient-Domain Metropolis Light Transport

Marco Manzi, University of Bern
Fabrice Rousselle, University of Bern, Disney Research Zurich*
Markus Kettunen, Aalto University
Jaakko Lehtinen, Aalto University, NVIDIA Research
Matthias Zwicker, University of Bern

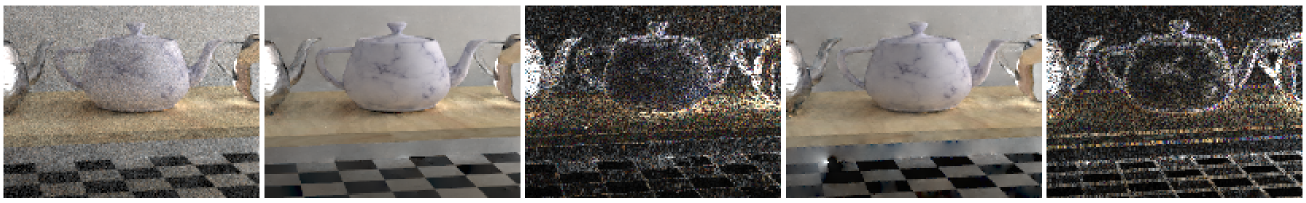


Figure 1: *Our generalized framework for gradient-domain Metropolis rendering includes three techniques to avoid sampling artifacts and reduce variance in sampled gradients. We (*OURS*) achieve visually and numerically improved results compared to previous work (*GDMLT*).*

Abstract

We present a generalized framework for gradient-domain Metropolis rendering, and introduce three techniques to reduce sampling artifacts and variance. The first one is a heuristic weighting strategy that combines several sampling techniques to avoid outliers. The second one is an improved mapping to generate offset paths required for computing gradients. Here we leverage the properties of manifold walks in path space to cancel out singularities. Finally, the third technique introduces generalized screen space gradient kernels. This approach aligns the gradient kernels with image structures such as texture edges and geometric discontinuities to obtain sparser gradients than with the conventional gradient kernel. We implement our framework on top of an existing Metropolis sampler, and we demonstrate significant improvements in visual and numerical quality of our results compared to previous work.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image generation—Display Algorithms;

Keywords: global illumination, metropolis light transport

Links: DL PDF WEB

1 Introduction

Monte Carlo sampling is firmly established as the most practical realistic image synthesis approach because of its flexibility and generality, but variance, which appears as visually distracting noise in the results, is a persistent challenge. In this paper, we build on the gradient-domain Metropolis light transport (GDMLT) algorithm of Lehtinen et al. [2013]. They realized that image space gradients between neighboring pixels, that is, pixel differences, can be sampled with little noise by sampling *pairs of paths* through the corresponding pixels such that they are *close to each other in path space*. Such paths tend to make similar contributions to the image, and hence they contribute small values to the gradient. By combining the estimated gradients with a noisy image using an L_2 Poisson reconstruction step, they obtained unbiased rendering results with significantly lower noise and lower error compared to sampling the image pixels only. Their approach, however, suffers from frequent sampling artifacts and singularities in the gradients, partially canceling their potential benefit. As a consequence, they proposed a reconstruction step using an L_1 error metric to increase robustness towards outliers, at the cost of introducing bias. Our goal is to avoid these issues by developing more robust gradient sampling schemes.

Here, we introduce a generalized framework for GDMLT that follows the same basic procedure as the original algorithm: first, we compute a coarse image and finite-difference gradients in image space using Metropolis sampling, and then reconstruct a higher quality image using a Poisson solver. Our contributions include three novel techniques that reduce sampling artifacts and variance in the sampled gradients, leading to significantly improved performance, both in L_2 (unbiased, see Figure 1) and L_1 reconstruction.

Our first technique provides a way to combine multiple gradient sampling strategies and weight them, akin to multiple importance sampling. In practice, we use a heuristic binary weighting function that weights down gradient samples around singularities and falls back to standard finite differencing there, effectively handling problematic paths more robustly. The two remaining techniques exploit the considerable freedom to determine neighboring paths to obtain a gradient integrand with significantly lower variance. In the second technique, given two pixels defining an image space gradient, we introduce a strategy to determine better *path space neighbors* that lead to more similar path contributions than in Lehtinen et al.'s [2013] original approach. In particular, our method automatically cancels out certain singularities that previously led to visual artifacts. In the third technique, we generalize the notion of image space gradients to include differences between arbitrary pairs of pixels. We show that by selecting pairs of similar pixels, we obtain gradients with smaller magnitudes and therefore less noise. In addition, we show that our technique more effectively preserves image structures during Poisson reconstruction than previous approaches.

*The research for this project was done while Fabrice Rousselle was a student at the University of Bern.

We implement our framework on top of an existing Metropolis sampler using manifold exploration [Jakob and Marschner 2012], and we demonstrate significant improvements in visual and numerical quality of our results compared to previous work. In summary, we make the following contributions:

- We introduce a generalized framework based on gradient-domain Metropolis light transport. It provides more flexibility for sampling gradients, leading to fewer artifacts.
- We describe how to evaluate gradients by computing multiple weighted integrals. We use a heuristic strategy to determine the weighting functions to avoid gradient singularities.
- We introduce a new method to determine path space neighbors when computing gradients between given pixel pairs. We leverage path space manifold walks to cancel out singularities.
- We generalize the concept of gradients to allow for more flexible image space gradient kernels.

2 Related Work

Physically-based Light Transport Physically based light transport algorithms render images by integrating over all possible light paths from a source to an image sensor. In its general form, the rendering equation [Veach and Guibas 1997]

$$I_j = \int_\Omega h_j(x) f^*(x)\,\mathrm{d}\mu(x) = \int_\Omega f_j(x)\,\mathrm{d}\mu(x) \qquad (1)$$

describes the radiance value I_j for each pixel j. The integral is over the space Ω of all light paths of finite length (*path space*), h_j is the pixel filter of the jth pixel, and $f^*(x)$ is the *spectral image contribution function* representing the amount of light reaching the sensor through a given path x in a given wavelength. We will also use the path contribution function $f_j(x) = h_j(x)f^*(x)$, which is the contribution of a path to a specific pixel j. A path x of length k consists of a sequence of vertices $\mathbf{x}_0, \ldots, \mathbf{x}_k$, and $\mathrm{d}\mu(x)$ is the *area product measure* $\prod_{i=0}^{k} \mathrm{d}A(\mathbf{x}_i)$.

Unbiased Monte Carlo rendering algorithms evaluate the pixel integrals with probabilistic methods. Basic Monte Carlo integration with importance sampling exploits that the integral I_j equals the expected value of $f_j(\mathbf{X})/p(\mathbf{X})$, with $\mathbf{X}$ a random variable distributed according to $p(\mathbf{X})$. Path tracers repeatedly draw random paths, evaluate the path contribution, and accumulate the weighted sample f/p, employing multiple importance sampling (MIS) when bidirectional samplers are used [Veach and Guibas 1995].

Adaptive Sampling and Reconstruction A large body of light transport algorithms adaptively sample the image (or the transport integrand), followed by an image reconstruction step. They attempt to direct computation so as to maximize attained image quality per unit of effort expended. Several techniques sparsely sample radiance and its (semi-analytic) gradients [Ward and Heckbert 1992; Dayal et al. 2005; Ramamoorthi et al. 2007], whereas we focus on the finite differences between pixels. Adaptive sampling (and reconstruction) techniques distribute more samples in image locations estimated to need them, and employ various sophisticated filters for reconstructing the final image [Rousselle et al. 2011; Bolin and Meyer 1995; Hachisuka et al. 2008; Overbeck et al. 2009; Egan et al. 2009; Egan et al. 2011]. They produce excellent results, but with no guarantee of unbiasedness.

Smart Filtering Filtering radiance estimates using auxiliary information gleaned from properties of the primary hits, such as normals, world space positions, and materials, is a powerful approach for reducing noise in Monte Carlo renderings [Ward et al. 1988; McCool 1999; Kontkanen et al. 2004; Sen and Darabi 2012; Rousselle et al. 2013]. These techniques make the natural but largely heuristic argument that the illumination solution correlates strongly with local scene features; thus, noisy estimates from "similar" regions can be blended to reduce variance. We build on the same observation, but as a crucial difference to earlier work, we present an unbiased algorithm that merely takes suggestions from such similarities.

Gradient-Domain Image Processing Finite difference gradients form the basis for an immense range of powerful image editing algorithms [Pérez et al. 2003]. We employ similar machinery to determine the image from computed gradients. We generalize standard finite differences, however, to include arbitrary pairs of pixels. This leads to generalized, data dependent Laplacians, similar to the Laplacians used in image segmentation [Shi and Malik 2000] and matting [Levin et al. 2008; Chen et al. 2013]. Recent work by Krishnan et al. [2013] shows how to solve the resulting Poisson problems efficiently. Like previous work, we also use a coarsely sampled primal image to aid reconstruction [Bhat et al. 2010; Lehtinen et al. 2013].

Metropolis Sampling Markov Chain Monte Carlo (MCMC) techniques draw random samples distributed according to functions that are difficult or impossible to sample from directly. In particular, given a target equilibrium distribution $f(x)$ and a *tentative transition function* $\tau(x \to y)$, the Metropolis-Hastings algorithm [Metropolis et al. 1953; Hastings 1970] constructs a Markov chain of samples distributed according to f. Starting with an initial state $\mathbf{x}_0$, it applies, at each step, a carefully chosen random change to the current state $\mathbf{x}_t$ to obtain the next state $\mathbf{x}_{t+1}$. In the limit, the samples will be distributed proportional to the desired target.

Metropolis Light Transport Assuming a converged chain, the samples produced by the Metropolis process can be used for integrating arbitrary (potentially vector-valued) functions. Metropolis Light Transport [Veach and Guibas 1997], short MLT, directly applies the above machinery to Equation 1 by generating a Markov chain of paths distributed according to the scalar luminosity $f(x)$ of their image contribution $f^*(x)$, and evaluating

$$I_j \approx \frac{C}{N} \sum_i \frac{h_j(x_i) f^*(x_i)}{f(x_i)}. \qquad (2)$$

The paths are distributed according to their luminance contribution to the image, and C is the integral of f^* estimated using other means, usually standard Monte-Carlo integration. Veach and Guibas propose several mutation schemes that act on the path itself. To alleviate the difficulty of implementation, Kelemen et al. [2002] introduced *primary sample space* mutations that remove the need to compute transition probabilities due to symmetry; however, some power is lost compared to path space mutations. Jakob and Marschner [2012] introduced a new mutation strategy, manifold exploration, that substantially improves the treatment of specular and highly glossy paths. Our algorithm builds on this approach.

Gradient-domain Metropolis Light Transport [Lehtinen et al. 2013] directly evaluates the horizontal and vertical finite differences $I_{j+1} - I_j$ between neighboring pixels without computing the actual values first. It does so by directly integrating in an extended path space that contains nearby pairs of paths, one through each pixel in question. Feeding the difference estimates and a low-fidelity version of the actual image to a screened Poisson solver [Bhat et al. 2010] then produces the final result. We defer further discussion to Section 3.1, as our novel derivation subsumes theirs.

3 Gradient Domain Rendering Framework

Here we introduce our gradient domain rendering framework, and three techniques to reduce variance in the gradient estimation. We first review the basic idea of computing gradient integrals in Section 3.1. In Section 3.2, we formulate a symmetric expression for these integrals, which is necessary to compute gradients in practice. In Section 3.3, we extend this approach to *multiple weighted* gradient integrals, which allows us to avoid sampling artifacts similarly to multiple importance sampling. In Section 3.4 we introduce an improved piecewise mapping function that leads to higher quality gradients than in previous work. Finally, in Section 3.5, we introduce generalized image space gradient kernels, which further improve the gradient quality. We simplify exposition using scalar radiance, and extension to the usual tristimulus (or spectral) rendering is easy.

3.1 Background

We start from Equation 1, which determines pixel intensities. The core idea in gradient domain rendering is to directly sample gradients, defined as differences between pairs of pixels, in addition to the pixel values themselves. This is beneficial because it is possible to sample the gradients with less variance than pixel intensities. Having sampled gradients and pixel values, we reconstruct the final image by solving a (screened) Poisson equation, where we use the pixel values as an additional constraint. This leads to results with less noise compared to the pixel values themselves.

Let us define a gradient $\Delta_{i,j}$ as the difference between two pixels i and j, where the pixel values I_i and I_j are given by their path space integrals. Hence,

$$\Delta_{i,j} = I_i - I_j = \int_\Omega f_i(x)\mathrm{d}\mu(x) - \int_\Omega f_j(x)\mathrm{d}\mu(x).$$

Instead of evaluating these two integrals separately, in gradient domain rendering we evaluate a gradient by sampling a single integral,

$$\Delta_{i,j} = \int_\Omega \left(f_i(x) - f_j(T_{ij}(x)) \left| \frac{\mathrm{d}T_{ij}}{\mathrm{d}x} \right| \right) \mathrm{d}\mu(x). \qquad (3)$$

Here T_{ij} is a *shift mapping* that deterministically maps a *base path* x to an offset path $\tilde{x} = T_{ij}(x)$. Below we drop the subscript from T_{ij} to reduce clutter. The indices will be clear from the context. The factor $|\mathrm{d}T/\mathrm{d}x|$ denotes the determinant of the Jacobian of $T(x)$ accounting for the change of integration variables for f_j.

A core idea is that we can design T such that $f_i(x) - f_j(\tilde{x})|\mathrm{d}\tilde{x}/\mathrm{d}x|$ generally has less variance than $f_i(x)$. Note that $T(x)$ usually only modifies a few vertices on path x while leaving the rest unchanged. Lehtinen et al. [2013] provide details on how to construct a suitable shift mapping, and we will build on and improve their method.

3.2 Symmetric Gradient Computation

For efficiency reasons, it is useful to sample the integrals for the pixel values I_i and the gradients $\Delta_{i,j}$ using the same probability density. This allows us to reuse a sample of the path contribution $f_i(x)$ for both I_i and $\Delta_{i,j}$. A probability density designed to sample the pixel integral correctly, however, may not sample the gradient correctly. In Metropolis sampling, for example, paths x with zero image contribution $f^*(x) = 0$ are never sampled. Yet, the corresponding offset paths may have a non-zero throughput, that is, $f(T(x)) > 0$, meaning that the sampler may not correctly sample Equation 3. Lehtinen et al. [2013] circumvented the issue by specifically checking for reversibility of the shift mapping, and weighting samples accordingly when local bijectivity was violated.

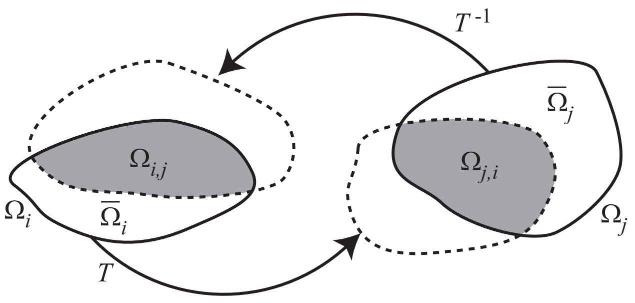


Figure 2: *Notation for the symmetric gradient computation: Ω_i is the region of path space contributing to pixel i. The region $\bar{\Omega}_i$ contributes to pixel i, but cannot be sampled from Ω_j via the inverse mapping. In contrast, Ω_{ij} can be sampled from Ω_j.*

We address this issue more generally by formulating an expression for the gradient Δ_{ij} that is symmetric in i and j. Specifically, we will integrate not only over differences $f_i(x) - f_j(T(x))|\mathrm{d}T/\mathrm{d}x|$ using the forward mapping T, but also $f_i(T^{-1}(x))|\mathrm{d}T^{-1}/\mathrm{d}x| - f_j(x)$ using the inverse mapping T^{-1}.

Let Ω_i be the region of path space that contributes to pixel i, that is $\Omega_i = \{x|h_i(x)f(x) > 0\}$, and similarly Ω_j, illustrated in Figure 2. We will apply the forward mapping only to paths in Ω_i, and the backward mapping to paths in Ω_j. In addition, we define $\bar{\Omega}_j = \Omega_j \setminus T(\Omega_i)$, that is, the paths that contribute to pixel j but that we do not sample using the forward mapping (because the corresponding base path has zero image contribution, $f^*(x) = 0$, or the corresponding base path has $h_i(x) = 0$ and does not contribute to pixel i). Similarly $\bar{\Omega}_i = \Omega_i \setminus T^{-1}(\Omega_j)$ are the paths that contribute to pixel i but that we do not sample using the backward mapping. Finally, $\Omega_{ji} = \Omega_j \setminus \bar{\Omega}_j$ are the paths that contribute to pixel j and that we do sample using the forward mapping, similarly Ω_{ij}, and $T(\Omega_{ij}) = \Omega_{ji}$. This means that we sample the differences between base-offset path pairs in Ω_{ij} and Ω_{ji} twice (using the forward and backward mapping), whereas for paths in $\bar{\Omega}_i$ and $\bar{\Omega}_j$ we sample the differences only once.

Hence, when sampling Ω_i using the forward mapping we compute its contribution Δ^i_{ij} to the gradient Δ_{ij} as

$$\Delta^i_{ij} = \int_{\bar{\Omega}_i} f_i(x)\mathrm{d}\mu(x) + \underbrace{\frac{1}{2}\int_{\Omega_{ij}} f_i(x) - f_j(T(x)) \left| \frac{\mathrm{d}T(x)}{\mathrm{d}x} \right| \mathrm{d}\mu(x)}_{\int_{\Omega_{ij}} f_i(x)\mathrm{d}\mu(x) - \int_{\Omega_{ji}} f_j(x)\mathrm{d}\mu(x)}, \qquad (4)$$

where we exploited $x \in \bar{\Omega}_i \Rightarrow f_j(T(x)) = 0$, therefore we do not need to evaluate this in $\bar{\Omega}_i$, and the factor $1/2$ compensates for duplicate sampling. In practice, we evaluate Δ^i_{ij} using one set of path samples x. We distinguish whether $x \in \bar{\Omega}_i$ or $x \in \Omega_{ij}$ and add a sample of the corresponding term to Δ^i_{ij}. We proceed analogously when sampling Ω_j and compute

$$\Delta^j_{ij} = -\int_{\bar{\Omega}_j} f_j(x)\mathrm{d}\mu(x) + \underbrace{\frac{1}{2}\int_{\Omega_{ji}} f_i(T^{-1}(x)) \left| \frac{T(x)^{-1}}{\mathrm{d}x} \right| - f_j(x)\mathrm{d}\mu(x)}_{\int_{\Omega_{ij}} f_i(x)\mathrm{d}\mu(x) - \int_{\Omega_{ji}} f_j(x)\mathrm{d}\mu(x)}. \qquad (5)$$

The desired gradient is then simply the sum of these two auxiliary values,

$$\begin{aligned}\Delta_{i,j} &= \int_{\bar{\Omega}_i \cup \Omega_{ij}} f_i(x)\mathrm{d}\mu(x) - \int_{\bar{\Omega}_j \cup \Omega_{ji}} f_j(x)\mathrm{d}\mu(x) \\ &= \Delta_{ij}^i + \Delta_{ij}^j.\end{aligned}$$

This strategy also works with partial mappings that may fail and not produce an output path at all for certain inputs, that is, $T(x)$ may be undefined for some x. This may happen for example due to numerical problems. For each x in Ω_i, if $T(x)$ fails, or $T^{-1}(T(x))$ fails, we can simply treat x as not belonging to Ω_{ij}. In addition, if we use an identity mapping for T (i.e., the offset and base paths coincide) the above symmetrized computation reduces to usual finite differencing between pixel intensities, since the Jacobian becomes unity and the identity mapping leads to $\bar{\Omega}_i = \Omega_i$, $\bar{\Omega}_j = \Omega_j$ and $\Omega_{ij} = \Omega_{ji} = \emptyset$.

3.3 Multiple Weighted Gradient Integrals

In the standard area product form, the path contribution function $f_i(x)$ consists of the product of the pixel filter, values of the BSDFs at the scattering event vertices, geometry terms, and light emission at the vertex on the source [Veach 1997]. Due to the geometry terms, $f_i(x)$ contains singularities, that is, it takes on infinite values for paths x where several vertices coincide. Besides paths with singularities, however, also paths close to them (paths with very short segments) are ill-behaved because their geometry terms contain divisions by small numbers. This leads to "exploding" path contributions. These issues can be observed in methods using virtual point lights where weak singularities occur in proximity of the cache entries [Kollig and Keller 2006; Walter et al. 2012]. In basic path tracing algorithms, however, this is not a problem — except numerically — because the geometry terms appear in both $f_i(x)$ and the sampling density $p(x)$, and thus cancel out in the sample weight.

Computation of $\Delta_{i,j}$ is not as forgiving. Because we sample both $f_i(x)$ and $f_i(x) - |T(x)/\mathrm{d}x| f_j(T(x))$ with the same density over x (see also Section 3.2), cancellation between $|T(x)/\mathrm{d}x| f_j(T(x))$ and $p(x)$ does not occur like it does with $f_i(x)$ and $p(x)$. Hence, a shift mapping $T(x)$ that moves offset paths closer to (or away from) singularities may lead to large finite differences between the two paths. Note that this is not incorrect as such: it merely means that the integrand that defines $\Delta_{i,j}$ has high variance, and is hence difficult to sample properly.

Our goal in this section is to design a method that detects cases close to singularities, and automatically falls back to a better-behaving sampling strategy when necessary (another approach is to design better shift mappings, cf. Section 3.4). In practice, we switch between two sampling strategies in a binary fashion. We show the validity of this approach by introducing a more general formalism that extends the integral in Equation 3 to multiple weighted integrals using a partition of unity defined by weight functions $w_k(x)$, with $\sum_k w_k(x) = 1, \forall x$. In addition, we apply a different mapping T_k to each weighted integral and obtain

$$\Delta_{i,j} = \sum_k \int_\Omega w_k(x) f_i(x) - w_k(T_k(x)) f_j(T_k(x)) \left| \frac{dT_k}{\mathrm{d}x} \right| \mathrm{d}\mu(x).$$

This formulation is similar to multiple importance sampling, except that we determine the weights using different heuristics. The key idea here is that by adjusting the weights $w_k(x)$ locally in path space according to the properties of the mappings $T_k(x)$, we can avoid sampling artifacts by automatically weighting down the sampling scheme close to a singularity. In practice we evaluate the symmetric formulation of the gradients with each mapping, but we omit the explicit formulation, which would be tedious (we simply need to include the weight functions in Equations 4 and 5). Also, we sample all integrals simultaneously with a single set of samples. Given a path sample x, we apply all mappings to it, and then compute a weighted sum of the gradients from all mappings.

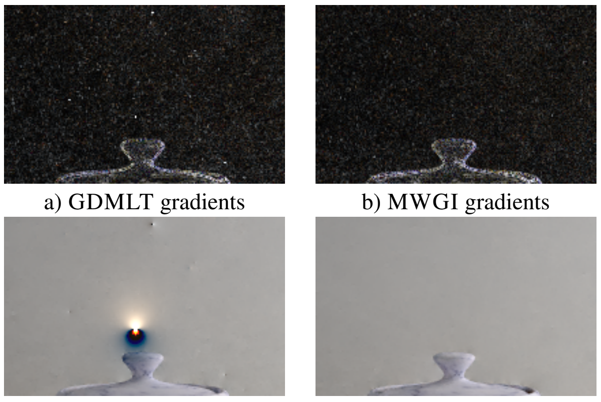


Figure 3: *Avoiding singularities with multiple weighted gradient integrals (*MWGI*): (a) basic gradients from gradient-domain MLT; (b) using our multiple weighted gradient integrals scheme; (c) and (d) corresponding L2 reconstructions. Note how artifacts in (c) correspond to outliers visible as bright peaks in the gradients.*

We propose a simple approach with two mappings: T_1 is the mapping described by Lehtinen et al. [2013], with important extensions designed to minimize the magnitude of gradients (Section 3.4), and T_0 is the identity mapping. As noted above, the identity mapping is equivalent to computing pixel differences at the end of the Metropolis sampling process. Our goal is to fall back on it when using the shift T_1 is numerically unstable due to singularities. While this approach is guaranteed not to add singularities to the gradient that are not present in the base path, it generally produces gradients with more variance (this is the rationale in using the shift mapping in the first place).

For this purpose we define binary weights

$$w_0(x) = \begin{cases} 1 & \text{if } \max\left(\frac{\|f_i(x)\|}{\|f_j(T_1(x))\|}, \frac{\|(f_j(T_1(x))\|}{\|f_i(x)\|}\right) > t \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where t is a user specified threshold, and $w_1(x) = 1 - w_0(x)$. This strategy falls back to using the identity mapping T_0 to compute the gradient if the offset path contribution $f_j(T_1(x))$ relative to the current path contribution $f_i(x)$ is above a threshold t. In Figure 3, we show that this simple strategy effectively reduces artifacts, although it comes with the disadvantage of requiring a user parameter. We leave the development of more sophisticated weighting strategies for future work.

3.4 Improved Mapping to Reduce Variance

Recall that in some geometric configurations, the offset path contribution function contains singularities due to divisions by zero in the geometry terms. In this section, we describe a novel shift mapping that reduces the occurrence of these singularities and variance caused by them compared to previous work [Lehtinen et al. 2013]. The result is a better-behaved integrand for $\Delta_{i,j}$ that is amenable to sampling with less noise.

We start by introducing the necessary notation to describe our approach. Let us denote a path parameterized by its vertices as $x = \{\mathbf{x}_0, \dots \mathbf{x}_n\}$, where $\mathbf{x}_0$ is the eye vertex, $\mathbf{x}_1$ is the primary hit vertex, and $\mathbf{x}_n$ is a vertex on a light source. The screen position of $\mathbf{x}_1$ is $\mathbf{s}_1$. The offset path produced by the mapping T is $\tilde{x} = T(x) = (\tilde{\mathbf{x}}_0, \dots \tilde{\mathbf{x}}_n)$, and $\tilde{\mathbf{s}}_1$ is the screen position of $\tilde{\mathbf{x}}_1$. In addition, let us assume each vertex is classified deterministically either as diffuse or specular based on its BSDF properties. Let $a < b < c$ be the indices of the first three diffuse vertices along a path, where the eye vertex is classified as diffuse by definition, so $a = 0$. Finally, $G(\mathbf{x}_i \leftrightarrow \mathbf{x}_j)$ is the (generalized) geometry term between vertex i and j [Jakob and Marschner 2012].

We build on the mapping proposed previously [Lehtinen et al. 2013] to obtain a gradient between pixels i and j. We review the previous approach first before introducing our improvement. The mapping consists of a concatenation of two steps: the first step updates vertices $\mathbf{x}_a, \dots, \mathbf{x}_b$, and the second step updates the rest, that is $\mathbf{x}_i$ with $i > b$. The Jacobian determinant of this concatenation is simply the product of the Jacobian determinants of both steps. In the first step, we calculate $\tilde{\mathbf{x}}_1$ such that $\tilde{\mathbf{s}}_1 - \mathbf{s}_1$ corresponds to the screen space offset between the centers of pixels i and j. Then if $b > 1$, the vertices $\mathbf{x}_i$ with $1 < i \leq b$ are updated by ray tracing to maintain a specular chain between $\tilde{x}$ and $\tilde{\mathbf{x}}_b$. The second step makes a case distinction based on whether $\mathbf{x}_{b+1}$ is diffuse or specular:

- **Diffuse:** The mapping T leaves all other vertices $\mathbf{x}_i$ with $i > b$ unchanged, that is, the second step is identity and its Jacobian determinant is one. Below we focus on analyzing and improving this case. Also note that in this case $c = b + 1$.
- **Specular:** We update the path segment $\mathbf{x}_b, \dots, \mathbf{x}_c$ such as to maintain the specular chain between $\mathbf{x}_b$ and $\mathbf{x}_c$. We achieve this using a manifold walk [Jakob and Marschner 2012].

Analysis We now analyze the offset path contribution $|\mathrm{d}\tilde{x}/\mathrm{d}x| f_j(\tilde{x})$ for the diffuse case. We discuss a common cause for singularities and variance, and then propose an approach to reduce them. Remember that the path contribution function is a product of BSDFs at the scattering vertices, (generalized) geometry terms between vertices, and light emission and importance functions. We observe that after the first step of the mapping, $\tilde{\mathbf{x}}_b$ and $\tilde{\mathbf{x}}_c$ may get arbitrarily close, which may lead to a singularity in the geometry term $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_c)$. We illustrate this schematically in Figure 4(a). Conversely, $\mathbf{x}_b$ and $\mathbf{x}_c$ may be very close in the base path, and move away from each other in the offset path, leading to a much smaller geometry term. Both cases cause variance in the gradients. In addition, the Jacobian of the first step of the mapping does not involve $\tilde{\mathbf{x}}_c$, and the Jacobian of the second step is identity, hence it is impossible that the Jacobian determinants would somehow cancel the problematic geometry term.

We illustrate the effect of singularities on the gradients in Figure 5, where we visualize the gradients next to the geometry terms $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$. In concave regions, very large gradients occur, and outliers in the geometry terms and gradients correlate closely.

Canceling Geometry Terms Our key observation and improvement is that the problems in concave regions can be addressed by modifying the second step of the mapping. Instead of doing nothing in the second step, we treat vertex $b + 1$ as specular, update the index c accordingly, and then run a manifold walk on the chain $\{\mathbf{x}_b, \dots, \mathbf{x}_c\}$ while shifting $\mathbf{x}_b$ to $\tilde{\mathbf{x}}_b$, similarly to the specular case (Figure 4(b)). Interestingly, one can show that the problematic geometry term $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$ cancels out with the Jacobian induced by the manifold walk. Intuitively, this is because the manifold walk changes the path densities as shown in Figure 4(c) and (d).

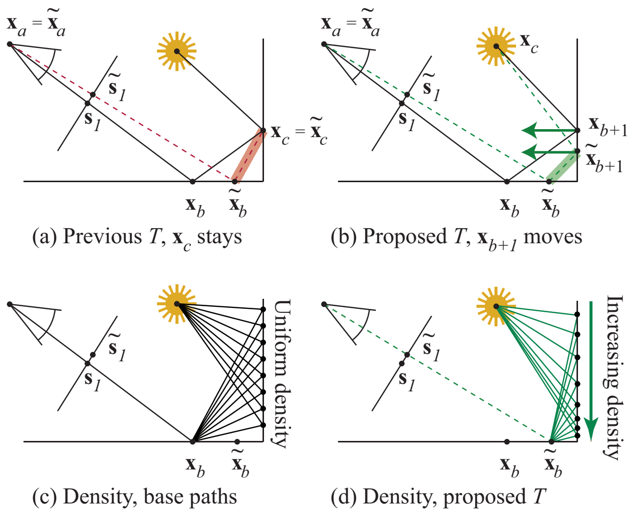


Figure 4: *(a) With Lehtinen et al.'s mapping T, singularities occur if $\tilde{\mathbf{x}}_b$ and $\tilde{\mathbf{x}}_c$ get arbitrarily close (red bar), which happens often in concave regions. (b) Declaring $\mathbf{x}_{b+1}$ specular and mapping it to $\tilde{\mathbf{x}}_{b+1}$ using a manifold walk on the segment $\mathbf{x}_b, \mathbf{x}_{b+1}, \mathbf{x}_c$ to preserve the half vector of $\mathbf{x}_{b+1}$ (green arrows) avoids this problem. Figures (c) and (d) explain this: (c) shows a family of base paths with uniform vertex density on the vertical surface. Because the proposed mapping T preserves the half vectors, it transforms these paths such that the vertex sampling density on the vertical wall increases as the geometry term $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$ approaches a potential singularity in the corner. Mathematically, the change of vertex densities is reflected in the Jacobian of the proposed mapping T, which cancels out the problematic geometry terms.*

For a mathematical explanation of this effect we need some more notation. First, a path segment $x^{bc} := \{\mathbf{x}_b, \dots, \mathbf{x}_c\}$ may be reparameterized in the *projected half vector domain* [Kaplanyan et al. 2014] as $h^{bc} = M(x^{bc}) = \{\mathbf{x}_b, \mathbf{h}_{b+1}^{\perp}, \dots, \mathbf{h}_{c-1}^{\perp}, \mathbf{x}_c\}$, where the $\mathbf{h}_i^{\perp}$ are the half vectors at the vertices projected onto the tangent planes. Now we can express our approach to map the path segment x^{bc} as

$$\tilde{x}^{bc} = M^{-1}(S(M(x^{bc}))),$$

that is, a reparameterization M into the half vector space, followed by a shift mapping S, and finally mapping back to area parameterization. The shift mapping $\tilde{h}^{bc} = S(h^{bc})$ simply moves vertex $\mathbf{x}_b$ to $\tilde{\mathbf{x}}_b$, where $\tilde{\mathbf{x}}_b$ is obtained in the first step described above. The key is that S operates in the half vector parameterization, and it does only shift the starting vertex $\mathbf{x}_b$, but it keeps the half vectors constant. Hence our procedure can be implemented using a manifold walk, which is designed to preserve half vectors while moving a single vertex position in a path.

Let $f(x^{bc})$ be the factors of the image contribution function of $f(x)$ that include only the vertices i with $b \leq i \leq c$ (we use the image contribution function f, since the pixel filter is not involved). We can now write the corresponding contribution $f(\tilde{x}^{bc})$ after our mapping,

$$f(M^{-1}(\underbrace{S(M(x^{bc}))}_{\tilde{h}^{bc}})) \left|\frac{\mathrm{d}M(x^{bc})}{\mathrm{d}x^{bc}}\right| \left|\frac{\mathrm{d}S(h^{bc})}{\mathrm{d}h^{bc}}\right| \left|\frac{\mathrm{d}M^{-1}(\tilde{h}^{bc})}{\mathrm{d}\tilde{h}^{bc}}\right|$$

$$= g(\tilde{h}^{bc}) \left|\frac{\mathrm{d}M(x^{bc})}{\mathrm{d}x^{bc}}\right| \left|\frac{\mathrm{d}S(h^{bc})}{\mathrm{d}h^{bc}}\right|,$$

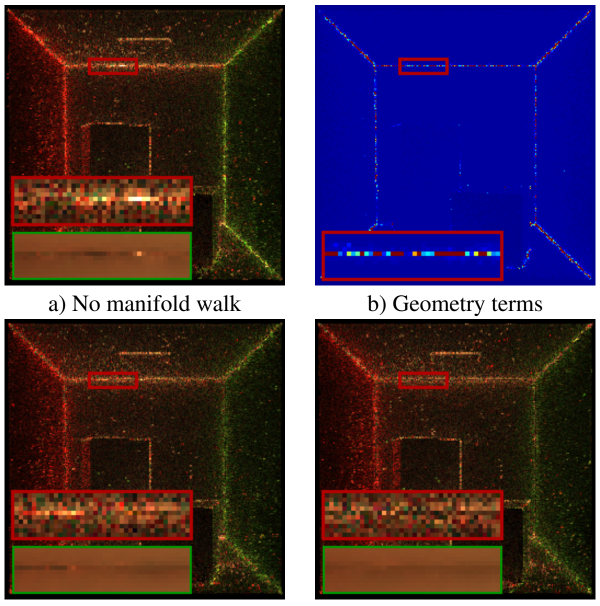


Figure 5: *(a) Gradients without our technique; (b) max-ratio of geometry terms $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$ over $G(\mathbf{x}_b \leftrightarrow \mathbf{x}_{b+1})$ (or vice-versa; blue is 1, red is high); (c) gradients where only vertex $b+1$ may be classified as specular, that is, one step of our incremental approach (one step* INCMW*); (d) gradients with the incremental approach (*INCMW*). We visualize gradient magnitudes and scale them for better visibility. The green close-ups show the result of the L1 reconstruction in the close-up region. All images used structure-adaptive gradients (Section 3.5).*

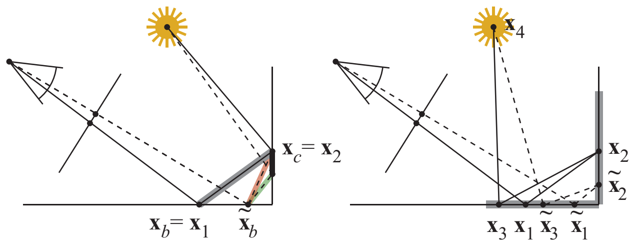


Figure 6: *(a) Assume we determine whether $\mathbf{x}_2$ should be declared specular using a threshold on the distance to $\mathbf{x}_1$. The threshold distance is indicated by the thick gray line. For a path on the decision boundary, the two possible offset paths (dotted lines) on either side of the boundary (red bar: identity mapping, green bar: manifold walk) leave a gap in path space, indicated by the black bar below $\mathbf{x}_2$. (b) The gray bars indicate regions where vertices are declared specular because of high ambient occlusion. The incremental manifold walk first generates $\tilde{\mathbf{x}}_2$ by applying a manifold walk on $\mathbf{x}_1, \mathbf{h}_2, \mathbf{x}_3$ while shifting $\mathbf{x}_1$ to $\tilde{\mathbf{x}}_1$. Then it generates $\tilde{\mathbf{x}}_3$ by applying a manifold walk on $\mathbf{x}_2, \mathbf{h}_3, \mathbf{x}_4$ while shifting $\mathbf{x}_2$ to $\tilde{\mathbf{x}}_2$.*

where the Jacobians account for the change in integration variables. In addition, $g(\tilde{h}^{bc})$ can be interpreted as the image contribution function in the half vector parameterization. Kaplanyan et al. [2014] showed that in $g(\tilde{h}^{bc})$ all geometry terms are canceled by the Jacobian of M^{-1}, except for the geometry term $G(\tilde{\mathbf{x}}_{c-1} \leftrightarrow \tilde{\mathbf{x}}_c)$, where $\tilde{x}^{bc} = M^{-1}(\tilde{h}^{bc})$. In addition, since S is a shift by a constant its Jacobian is identity. Also the Jacobian of M is given by the base path but independent of the offset path. Intuitively, the cancellation of the geometry terms by the Jacobian corresponds to the change of densities in Figure 4(d). Hence our approach avoids the potential singularity produced by $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$, and the benefits of this approach can be seen in Figure 5(c).

Incremental Approach An intuitive idea to avoid the remaining singularities from $G(\tilde{\mathbf{x}}_{c-1} \leftrightarrow \tilde{\mathbf{x}}_c)$ would be to declare further vertices as specular and increase the index c, until $\mathbf{x}_c$ is sufficiently far from $\mathbf{x}_{c-1}$. Unfortunately, this leads to "gaps" in the output of the mapping from x^{bc} to $\tilde{x}^{bc}$, as illustrated in Figure 6(a), causing bias in the resulting images.

We avoid this problem using a different heuristic, shown in Figure 6(b), where the key is that we decide for each vertex $\mathbf{x}_i$, one-by-one, in ascending order, whether it should be declared specular, in such a way that the decision does not depend on $\mathbf{x}_i$ itself. We observe that singularities tend to occur in paths where subsequent vertices are close to each other. Intuitively, such path configurations are common in regions of high ambient occlusion. If $\mathbf{x}_{i-1}$ lies in a region with high ambient occlusion, chances are high that $\mathbf{x}_i$ is close-by. In this case, we are likely close to a singularity. In practice, we test the ambient occlusion factor at $\mathbf{x}_{i-1}$, and if it is above a threshold, then $\mathbf{x}_i$ is considered specular. If $\mathbf{x}_i$ is declared specular, we map the vertex position $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_i$ using a manifold walk on $\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}$, where the shift of $\mathbf{x}_{i-1}$ is given by the difference to $\tilde{\mathbf{x}}_{i-1}$, which was obtained in the previous step. Since this approach amounts to a concatenation of mappings via manifold walks, the overall Jacobian determinant is simply the product of the Jacobian determinants of each step. In addition, since each step is a valid bijective mapping, this also applies to the concatenation of the mappings. We illustrate the benefits of this approach in Figure 5(d).

3.5 Structure-Adaptive Gradient Kernels

A basic intuition why gradient domain rendering reduces noise is that high-dimensional path contribution functions of neighboring pixels are often very similar, and one can exploit this similarity using suitable mapping functions in path space. Such mappings compute gradients as differences between similar base and offset paths, which are in general much smaller than the variance of the path contributions of either pixel. Hence the sampled gradients have less variance than the sampled path contributions. Our key observation is that we may compute gradients between arbitrary pairs of pixels, and select pairs such that their *path contribution functions are as similar as possible*. We find pixel pairs that are usually more similar than pairs among the 4-connected neighborhood of usual finite difference gradients. Therefore, the differences between base and offset paths become even smaller, and we further reduce variance.

Motivated by this, we define *structure-adaptive gradients* at each pixel as the pairwise differences to the n most similar pixels in a small window, as opposed to standard gradients consisting of pixel differences between horizontally and vertically adjacent pixels. We describe an affinity function used to determine similarity below. Given the similarity relations between pixels, we then define a generalized Poisson reconstruction problem.

More precisely, assume that the gradient Δ_{ij} occurs in the structure-adaptive kernel. Then we add a constraint $\tilde{I}_i - \tilde{I}_j = \Delta_{ij}$ to our equation system, where $\tilde{I}_i$ and $\tilde{I}_j$ are the unknown pixel values we would like to reconstruct. Like previous work, we also use the noisy "primal" image obtained as a by-product of gradient sampling as an additional constraint, and obtain an overconstrained equation system even if the structure-adaptive gradient matrix is

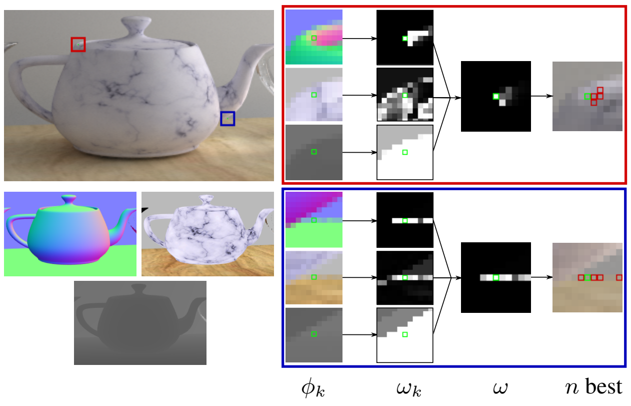


Figure 7: *Structure adaptive gradients are defined using the most similar neighbors. First, we compute bilateral weights ω_k for each feature ϕ_k in a box shaped neighborhood (brighter pixels mean bigger weights). For every pixel in the neighborhood we denote the weight of the most restrictive feature as ω. Finally, we select the n neighbors with the n biggest ω weights (here $n = 4$).*

arbitrarily rank deficient. This corresponds to the reconstruction used by Lehtinen et al. [2013], except with differences computed between adaptively-determined neighbors.

Pixel affinity We estimate similarities of path contribution functions by leveraging auxiliary per-pixel features (Figure 7). Features such as normals, depth, texture, and so on, serve this purpose well, since path contribution functions across feature discontinuities tend to be quite different. Indeed, using features for cross-bilateral filtering is highly successful for denoising Monte Carlo renderings [Ward et al. 1988; Dammertz et al. 2010]. Hence, we estimate the similarity between pixels i and j as $\omega(i,j) = \min_k[\omega_k]$ with

$$\omega_k = \exp\left(-\frac{(\max(0, |\phi_k(i) - \phi_k(j)|^2 - \tau_k)}{\sigma_k^2}\right) \tag{7}$$

where $\phi_k(i)$ is the kth feature at pixel i, τ_k a user-defined threshold and σ_k a bandwidth parameter for feature k. In other words, we define $\omega(i,j)$ similarly as a cross-bilateral filter weight using the feature that is least similar between i and j. We compute $\omega(i,j)$ for all pixels around i in a box shaped neighborhood. We define our n structure-adaptive gradients at each pixel as the differences to the n pixels with the largest weights in the neighborhood.

Since pixel differences are signed quantities, we should select one of the two possible orderings for each pixel pair defining a gradient. We choose to avoid this issue as follows to simplify implementation: Let ij be the ordered pair of pixels i and j. In addition, let P be the set of all pairs generated by our data adaptive procedure above, where i is always the center pixel and j a neighbor. Note that $ij \in P$ does not imply $ji \in P$. Now we distinguish two cases:

- **Non-Symmetric Neighbors:** If only $ij \in P$ but $ji \notin P$, we include the constraint $\tilde{I}_i - \tilde{I}_j = \Delta_{ij}$ in the equation system. We sum Δ^i_{ij} and Δ^j_{ij} to Δ_{ij} as described in Section 3.2.
- **Symmetric Neighbors:** If both $ij \in P$ and $ji \in P$, however, we add two separate constraints $\tilde{I}_i - \tilde{I}_j = 2\Delta^i_{ij}$, and $\tilde{I}_j - \tilde{I}_i = 2\Delta^j_{ji}$. This works because the second constraint is equivalent to $\tilde{I}_i - \tilde{I}_j = 2\Delta^j_{ij}$ (note $\Delta^j_{ij} = -\Delta^j_{ji}$), hence the average of these constraints represents the desired gradient.

4 Bilateral-Domain Metropolis Light Transport

In this section we present an MLT algorithm that implements the ideas presented in the previous chapter. We call the algorithm *Bilateral-Domain Metropolis Light Transport*. As per common practice, we apply the algorithm only to indirect illumination, and handle direct illumination separately in a prior pass using a simple recursive ray tracer. Our approach is currently limited to box functions as pixel filters.

Structure-Adaptive Gradients To compute the neighbors for structure-adaptive gradients, we gather pixel features in the direct illumination pass. We store features such as normals, depths and textures in auxiliary images, assuming that the features contain only insignificant residual noise. We achieve this by using sufficiently many samples in the direct illumination pass, or by denoising the features images [Rousselle et al. 2013].

Ambient Occlusion for Incremental Manifold Walk We guide our incremental manifold walk (Section 3.4) using screen space ambient occlusion coefficients, which we compute together with the feature images in the direct illumination pass. We define a threshold on the length of secondary ray segments, and for each pixel compute its ambient occlusion coefficient as the percentage of secondary rays whose lengths are below this threshold. Since our gradients are defined in screen space, we also define the threshold length in terms of its projection to screen space. As a consequence, the world space threshold increases for primary hit points further away from the camera. Following the iterative scheme from Section 3.4, we decide whether a vertex $\mathbf{x}_i$ should be perturbed using a manifold walk based on the ambient occlusion value at vertex $\mathbf{x}_{i-1}$. Since we use a screen space ambient occlusion map, we project the position of $\mathbf{x}_{i-1}$ to screen space and read the corresponding value. We perform a depth test to check whether $\mathbf{x}_{i-1}$ is occluded from the camera, and only use the ambient occlusion value if it is not occluded. While this approach is limited to path vertices that are visible to the camera, it is still effective at removing the most prominent artifacts due to near singular geometry terms in concave regions visible in the image. In addition, we suppress occasional remaining singularities with our weighting scheme from Section 3.3.

Gradient Sampling Here we describe in more detail how we sample the symmetric gradient integrals from Section 3.2 using a Metropolis sampler. For this we define an extended path space Ω'_i for each pixel,

$$\Omega'_i = \Omega_i \times N_i,$$

where N_i is a set of neighbor indices, such that for all neighbors $k \in N_i$, either k is one of the closest neighbors of i, that is, $ik \in P$, or vice versa i is one of the closest neighbors of k, that is, $ki \in P$. Hence, $|N_i|$ is, in general, different for each pixel i. The Metropolis algorithm samples the union Ω' of all per-pixel extended path spaces, $\Omega' = \bigcup \Omega'_i$. Denote a path in Ω' by z, consisting of a usual path x and a neighbor index j. Since we use box filters, x is uniquely assigned to a pixel i, and together with the neighbor index this specifies a gradient contribution Δ^i_{ij}.

In the Metropolis sampler we use conventional mutators to propose new paths x. Assuming x belongs to pixel i, we then complete the extended path z by proposing one neighbor $j \in N_i$ randomly. Finally we evaluate the image contribution $f^*(x)$ and the gradient contribution Δ^i_{ij}. Since this means we are counting the image contribution $|N_i|$ times more often than each gradient, we multiply the gradients by $|N_i|$. This procedure also assures that we sample any

Algorithm 1: SAMPLE CONSTRAINT

```
Input: Extended path $z$, consisting of base path $x$ in pixel $i$ and a
       neighbor index $j$.
begin
    /* Sum over mappings $T_k$ and weight functions $w_k$. */
    for $k \in 0, 1$ do
        /* Sample gradient contribution $\Delta^i_{ij}$, Equation 4. */
        /* Sample called $s$. */
        if $x \in \Omega_{ij}$ then
            $\hat{x} = T_k(x)$
            $s = 1/2(w_k(x)f_i(x) - w_k(\hat{x})f_j(\hat{x})|d\hat{x}/dx|$
        else
            $s = w_k(x)f_i(x)$
        /* Multiply with $|N_i|$, since we sample only one
           gradient for each base path. */
        $s = |N_i|s$
        /* Check neighbor symmetry, Section 3.5. */
        if $ij \notin P$ then
            /* Non-symmetric, constraint $\tilde{I}_j - \tilde{I}_i = \Delta_{ji}$ */
            $b(r(j,i)) = b(r(j,i)) - s$
        else if $ji \notin P$ then
            /* Non-symmetric, constraint $\tilde{I}_i - \tilde{I}_j = \Delta_{ij}$ */
            $b(r(i,j)) = b(r(i,j)) + s$
        else
            /* Symmetric, constraint $\tilde{I}_i - \tilde{I}_j = 2\Delta^i_{ij}$ */
            $b(r(i,j)) = 2s$
```

gradient Δ_{ij} symmetrically, because both $j \in N_i$ and $i \in N_j$. Finally, we use a target function $f(z)$ similar to Lehtinen et al. [2013],

$$f(z) = |N_i| \|\Delta^i_{ij}(x)\| + \alpha \|f^*(x)\|,$$

where the extended path z implies the indices i and j. Again, we include the factor $|N_i|$ because gradients are sampled $|N_i|$-times less often than the image contribution, hence their weight should be emphasized by the same factor in the target function.

Assembling the Gradient Constraints Algorithm 1 summarizes how we construct the gradient constraints from the sampled gradient contributions. We represent the gradient constraints as $H\tilde{I} = b$, where $\tilde{I}$ is the unknown image we will reconstruct, H has rows that are zero except for two entries with values -1 and 1 that encode the pixel pair involved in a gradient. The vector b stores the corresponding gradient values. The function $r(i, j)$ returns the row index where constraint $\tilde{I}_i - \tilde{I}_j$ is stored in the equation system. The pseudo-code takes as input an extended path z obtained from the Metropolis sampler, and it stores the sampled gradient contribution Δ^i_{ij} in the gradient constraints as described in Section 3.5. It computes a weighted sum over all mappings as described in Section 3.3, although in practice we implement this more efficiently by exploiting that one of our two mappings is the identity.

Poisson Reconstruction We reconstruct an image that best fits the gradient constraints and the coarse image that we sampled during rendering by solving a Poisson problem,

$$\min_{\tilde{I}} ||H\tilde{I} - b||^2 + ||\alpha(\tilde{I} - I_g)||_2^2, \tag{8}$$

where $\tilde{I}$ is the reconstructed image and I_g is the image obtained by sampling the path contributions. This approach leads to an unbiased estimate $\tilde{I}$ of the true image if we use the L_2-norm as above. For a proof we refer to Lehtinen et al.'s work [2013]. We can also obtain visually improved results by minimizing the L_1-norm with an iteratively reweighted least-squares solver, although this introduces bias.

5 Results

In this section we report on results and comparisons obtained with our approach, which we implemented on top of the Mitsuba renderer [Jakob and Marschner 2012].

Parameter Settings We set the threshold t for the binary weight ω_0 in Equation 6 (Section 3.3) to 20 for all scenes, which conservatively suppresses outliers. This is important since a more aggressive, lower threshold will fall back to the identity mapping too often and reintroduce noise in the gradients. We compute screen space ambient occlusion coefficients (Section 3.4) using 0.75% of the image size as the threshold for the length of secondary path segments projected to image space. The threshold on the ambient occlusion coefficient that triggers the incremental manifold walk is 0.95, so that weakly concavely curved surfaces do not always lead to an extended manifold walk.

For the structure-adaptive gradients (Section 3.5) we use $\sigma_k = 0.01$ for all features, since the selection is very insensitive towards this parameter. The parameter τ_k is critical to make sure our adaptive gradients are not overly sensitive to the features. In particular, we need to avoid having them degenerate to 1D gradients along 1D contours in the features. Empirically we found the values $\tau_{normal} = 0.02$, $\tau_{texture} = 0.001$ and $\tau_{depth} = 0.01$ to work well for features normalized to range $[0, 1]$ in all tested scenes. In addition we use a bilateral window of 5×5 pixels to compute affinities $\omega(i, j)$ and $n = 4$ as the number of structure-adaptive gradients in all our experiments.

Benefits of Each Technique Figure 8 shows how each of our suggested strategies improves quality of the L_2 reconstruction in certain regions of the DOOR scene. We compare the rMSE (relative mean squared error) and the per-pixel gradient energy of the close-up regions. The per-pixel gradient energy is defined as the sum of squared values of all gradient constraints on each pixel, that is, $\|H^T b\|_2^2$. Using structure-adaptive gradients (ADAP, Section 3.5) instead of ordinary gradients strongly reduces ringing artifacts along edges as can be seen in the second row of the red and green close-ups. Since our adaptive gradients are based on features from the direct illumination pass only, they do not necessarily minimize gradient energy in regions where indirect illumination effects like caustics or indirect shadows occur. Therefore, the quality does not improve in such regions compared to using ordinary gradients, as can be seen in the second row of the blue close-up. Adding our multiple weighted gradient integrals (MWGI, Section 3.3) to avoid singularities helps removing bullet hole artifacts. This improves the quality of regions where structure-adaptive gradients alone are useless as can be seen in the third row of the blue close-up. Since avoiding singularities means using the noisier but less singularity-prone identity mapping T_0, regions where a lot of singularities occur still tend to become noisier. This mostly happens around concave geometry edges, as can be seen in row three of the green close-up. Adding the ambient occlusion guided incremental manifold walk (INCMW, Section 3.4) greatly reduces the need to fall back to T_0 in those regions, which effectively reduces noise there (bottom row of the green close-up).

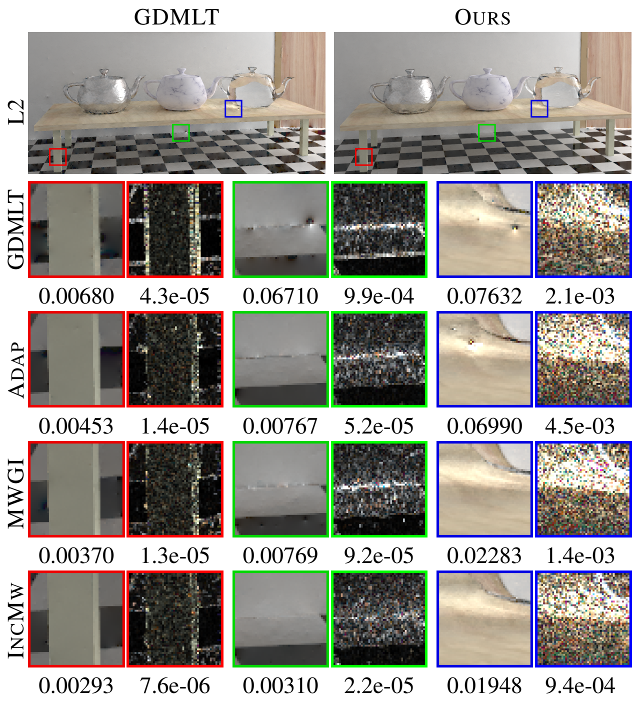

GDMLT
OURS
L2
GDMLT
0.00680 4.3e-05 0.06710 9.9e-04 0.07632 2.1e-03
ADAP
0.00453 1.4e-05 0.00767 5.2e-05 0.06990 4.5e-03
MWGI
0.00370 1.3e-05 0.00769 9.2e-05 0.02283 1.4e-03
INCMW
0.00293 7.6e-06 0.00310 2.2e-05 0.01948 9.4e-04

Figure 8: *We incrementally add our techniques on top of* GDMLT *in top to bottom order: unmodified* GDMLT, *structure adaptive gradients (*ADAP, *Section 3.5), multiple weighted gradient integrals (*MWGI, *Section 3.3) and incremental manifold walk (*INCMW, *Section 3.4). For every crop we show the L_2 reconstruction with the relative MSE and the energy of the gradients.*

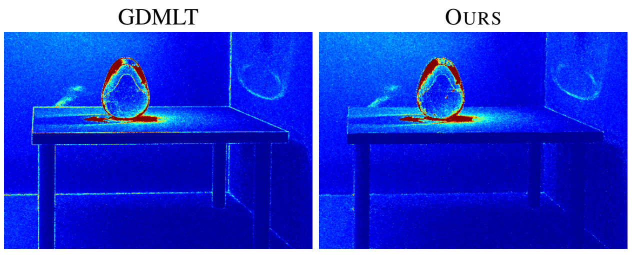


Figure 9: *A comparison of the sampling densities of our method and GDMLT. Note how our method distributes less samples along edges that are avoided by our structure-adaptive gradients.*

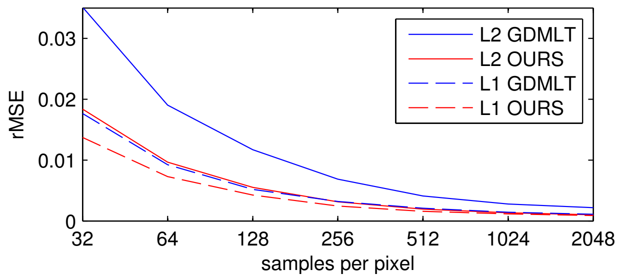


Figure 10: *The rMSE for* DOOR *using our method compared to* GDMLT *over increasing numbers of mutations per pixel.*

Comparison to Previous Work In Figure 13 we demonstrate the effectiveness of our approach by comparing it to gradient-domain Metropolis (GDMLT) [Lehtinen et al. 2013] and manifold exploration MLT (MEMLT) [Jakob and Marschner 2012]. We computed reference images of the DOOR, SIBENIK and BOX scenes with MEMLT with 32000 mutations per sample, and for SPONZA and BIDIR we used bidirectional path tracing with 32000 samples per pixel. For MEMLT we used the default parameters of Mitsuba, and for GDMLT we used the parameters suggested by Lehtinen et al. [2013]. We adjusted the set of path mutators, the maximum path length, and the length of the Markov chains for every scene separately. For each scene we used the same parameters for all compared methods. We applied GDMLT and our method on the indirect illumination only. We computed direct illumination separately and then simply added it to the result of the reconstruction. Our approach has less than 5% computational overhead compared to GDMLT.

We compare the relative mean square error (rMSE) and observe an improvement of our method over GDMLT of 20%-60% using L_2 reconstruction, and of 5%-40% using L_1. We also measured how many mutations per pixel were needed for MEMLT to achieve similar rMSE as the L_1 reconstruction of our method. Results suggest that the required number of mutation per pixel is highly scene dependent. While for SIBENIK only two and a half times more mutations achieve similar rMSE, for BIDIR we required 20 times more mutations.

In general we observed that the L_2 reconstruction of GDMLT often performs poorly because the sampling tends to get stuck due to gradient outliers. Huge gradient values may appear even in smooth image regions due to geometrical singularities in the offset paths, which are caused by specular objects in the scene. The Poisson reconstruction then attempts to reconcile these gradient outliers with the coarse sampled image I^g. This leads to visually and numerically prominent bullet hole artifacts. The L_1 reconstruction of GDMLT suffers less from these problems, since entries in the linear equation system leading to big errors are weighted down. Yet some artifacts remain, especially when they occur near edges (see L_1 reconstruction of GDMLT in green close-up of DOOR scene). In all scenes our algorithm suffers less from singularity artifacts than GDMLT, as can be seen when comparing L_2 reconstructions of our approach and GDMLT.

In Figure 9 we compare the sampling densities of our approach to GDMLT in the BIDIR scene using approximately 256 mutations per pixel. One interesting observation is that our method concentrates samples exclusively in those regions where gradients occur that are not detectable in the features we used to generate the structure-adaptive gradients. GDMLT on the other hand distributes samples along all gradients of the image. This can be seen at the edges belonging to the table. This is a desirable property of our approach since it means that most samples are concentrated around regions that require more effort to be rendered correctly (e.g., caustics, indirect shadows and so on).

Figure 10 plots the rMSE of GDMLT and our method for L_2 and L_1 reconstructions against increasing numbers of mutations per pixel using the DOOR scene. The plotted error values are the averages over 25 runs. L_2 OURS is consistently better than L_2 GDMLT. The difference between L_1 GDMLT and L_1 OURS is smaller but still significant. Notably L_2 OURS has similar rMSE values as L_1 GDMLT, meaning our unbiased L_2 reconstruction is of similar quality as the biased L_1 reconstruction of GDMLT.

Limitations Our binary weighting scheme cannot avoid artifacts due to singularities in the base path. Considering Equation 6, we observe that a singularity in $T_1(x)$ but not in x will likely lead to a big max-ratio and therefore we fall back to sampling strategy T_0. This means the singularity in $T_1(x)$ will not affect the final result. If x instead of $T_1(x)$ is close to a singularity, however, our approach is ineffective. Hence, like in other MLT methods our Markov chain may sometimes get stuck. Artifacts arising from this are isolated bright pixels, but they do not cause bullet holes as do outliers in the gradients.

The global parameters τ_k and t of our method can lead to locally suboptimal results in complex scenes: τ_k leads to a global trade-off between sensitivity towards a feature and avoidance of degenerate one-dimensional constraints, whereas t leads to a trade-off between bullet holes and more noise due to the fall-back to strategy T_0.

Like all MLT methods our method suffers from potentially incomplete coverage of path space, as can be seen in the missing highlight on the glass-egg in the BIDIR scene (Figure 13).

Analysis of Adaptive Gradients The Poisson reconstruction problem in Equation 8 is equivalent to solving

$$(H^T H + \alpha \mathrm{Id})\tilde{I} = \alpha I^g + H^T b, \tag{9}$$

where Id is the identity matrix, and the matrix $H^T H$ can be interpreted as a Laplacian matrix given by the gradient kernels. In Figure 11 we compare the eigenvectors of $H^T H$ of GDMLT and our approach. We visualize only a subset of all eigenvectors, since there are as many eigenvectors as pixels in the image. The Laplacian of the GDMLT kernel is the usual discrete Laplacian, and its eigenvectors are 2D sinusoidal functions corresponding to the discrete Fourier transform. The eigenvectors of our kernels, however, reflect and preserve the structures and edges of the feature images that we used to construct the kernels, even for eigenvectors corresponding to low eigenvalues (that is, low frequencies). Intuitively, the Poisson solver reconciles contradicting information in the coarse image and the gradients by suppressing high frequencies (see also the analysis by Lehtinen et al. [2013]). While suppressing high frequencies in the Fourier transform is prone to ringing artifacts, suppressing high frequency eigenvectors in our approach does not suffer from this problem, since our low frequencies still contain image edges.

In Figure 12 we analyze the contribution of the coarse image I^g and the gradients $H^T b$ to the solution of Equation 9 by simply setting the other part of the right hand side ($H^t b$ and I^g, respectively) to zero. A comparison of our approach to the conventional gradient kernel reveals how the image structure is built into our structure-adaptive kernels. Even if we force gradients to be zero and give little weight to errors with respect to the coarse image (low α values, top row) we still preserve sharp edges. In comparison, conventional kernels behave like low-pass filters in this setting.

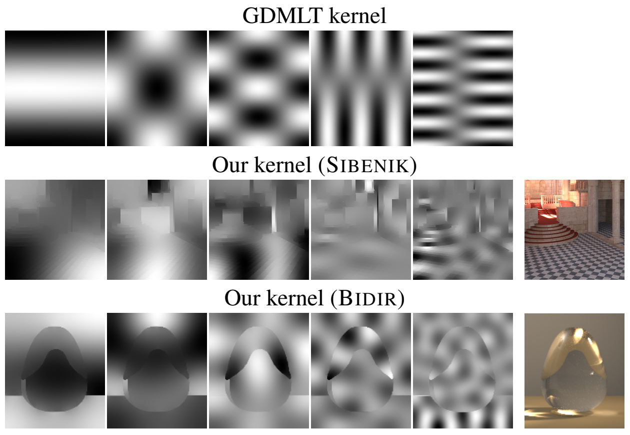


Figure 11: *Comparison of the eigenvectors corresponding to the 5th, 10th, 20th, 40th and 80th smallest eigenvalues (in that order) using the GDMLT kernels and our data-adaptive kernels. Note that the eigenvectors using our kernel adapt to the scene, while the eigenvectors using the GDMLT kernel do not. All visualized eigenvectors are normalized to* $[0, 1]$.

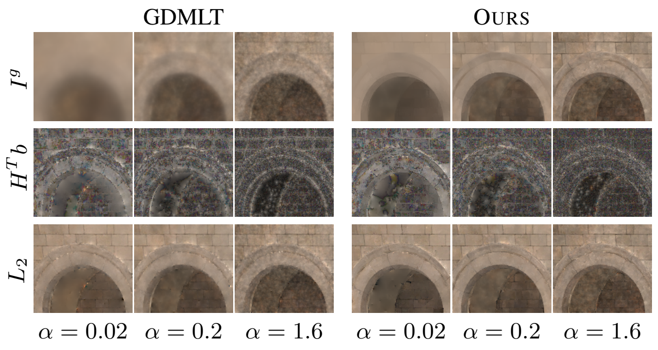


Figure 12: *Comparison of the effect of α on the Poisson solver on a close-up region of SPONZA with only 32 mutations per pixel. The top row shows the contribution of I^g to the solution, the middle row shows the contribution of the gradients $H^T b$ to the solution and the bottom row shows the L_2 reconstruction, which is equal to the sum of both images above. The tone-mapping of the gradient contribution has been adjusted to be more visible.*

6 Conclusions

In this paper we introduced a generalized framework for gradient domain Metropolis rendering, which we exploited to develop three techniques to avoid singularities and reduce noise in sampled gradients. A common insight that we explore in all techniques is that there is considerable freedom in how to determine gradients by computing offset paths in path space.

The first technique introduces the idea of applying several different shift mappings simultaneously to generate the offset paths. We show that we can weigh each mapping, akin to multiple importance sampling, to suppress the contribution of mappings that yield outliers and high variance. The second technique is an improved mapping function that avoids certain singularities that lead to artifacts in previous methods. This technique leverages the half vector parameterization of light paths, which by construction avoids most singularities of the usual area parameterization. Finally, our third technique builds on the observation that gradient domain rendering is not restricted to conventional image gradients. We exploit this and introduce structure-adaptive kernels that encode edges and details in auxiliary feature images. The structure-adaptive kernels avoid sampling gradients between pixels with highly different path contribution functions because of geometry or other discontinuities in the scene. Avoiding such difficult gradients further reduces noise. We also show that our kernels have interesting properties with respect to the Poisson reconstruction step. An eigenanalysis of the Laplacian matrix induced by our kernels shows that image structures are preserved even in eigenvectors with low eigenvalues, which means that the Poisson reconstruction process is less prone to ringing than conventional kernels.

We obtain results that significantly reduce sampling artifacts of previous approaches. Our unbiased L_2 reconstructions generally

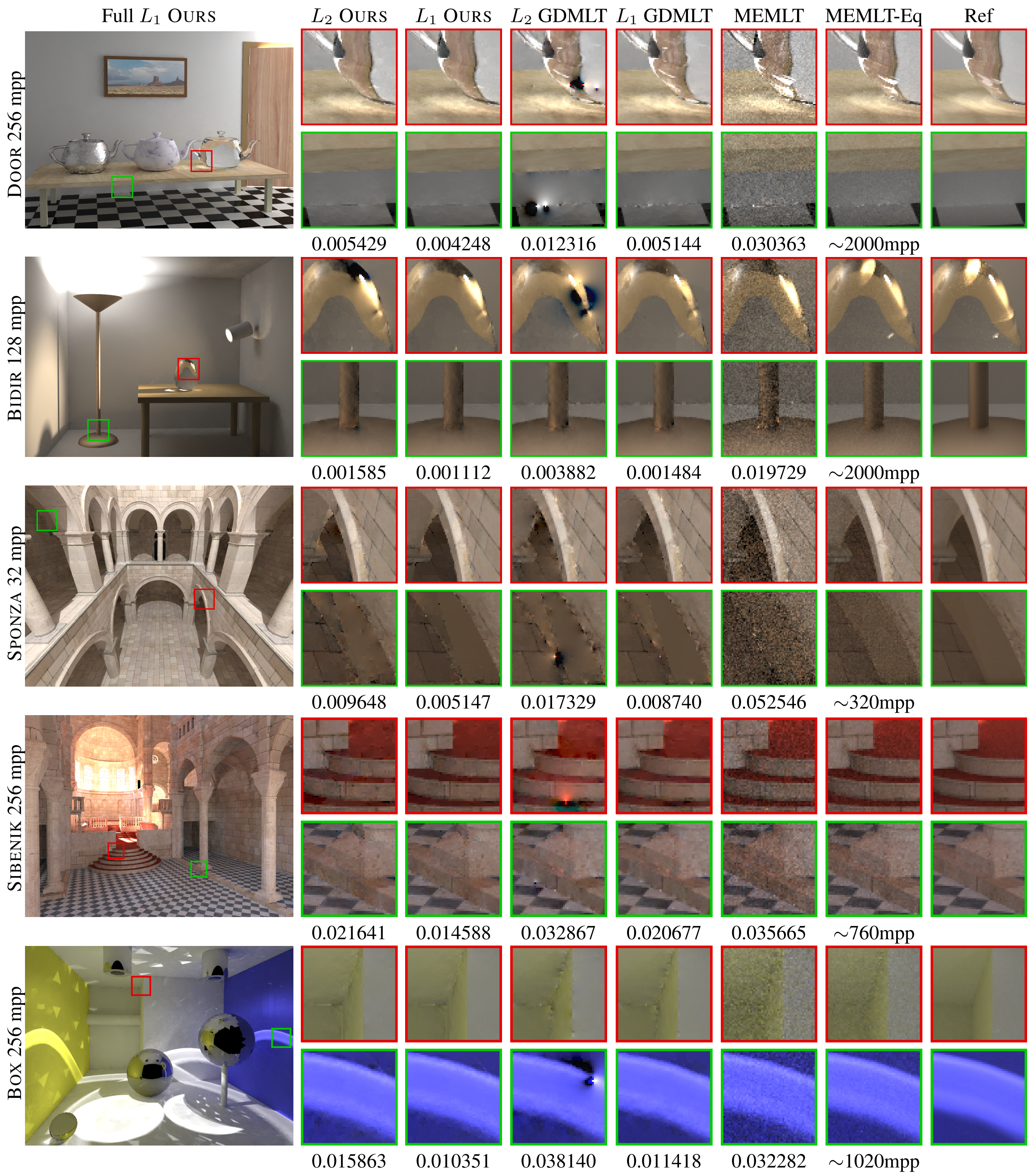

Full L_1 OURS
L_2 OURS
L_1 OURS
L_2 GDMLT
L_1 GDMLT
MEMLT
MEMLT-Eq
Ref
DOOR 256 mpp
0.005429 0.004248 0.012316 0.005144 0.030363 ∼2000mpp
BIDIR 128 mpp
0.001585 0.001112 0.003882 0.001484 0.019729 ∼2000mpp
SPONZA 32 mpp
0.009648 0.005147 0.017329 0.008740 0.052546 ∼320mpp
SIBENIK 256 mpp
0.021641 0.014588 0.032867 0.020677 0.035665 ∼760mpp
BOX 256 mpp
0.015863 0.010351 0.038140 0.011418 0.032282 ∼1020mpp

Figure 13: *Our proposed method using L_2 and L_1 reconstruction compared to gradient-domain MLT and manifold exploration MLT using approximatively the same number of mutations per pixel (mpp). MEMLT-Eq shows MEMLT with more samples in order to achieve approximately the same quality in terms of rMSE as L_1 OURS. The numbers below the close-ups with exception of MEMLT-Eq show the rMSE of the full images. All references where generated using MLT with 32k mpp.*

match previous biased L_1 results, and our L_1 results further improve on this. We believe our work shows that it is possible to achieve high quality gradient sampling, which may pave the way to robust, general purpose gradient rendering algorithms. In the future, we would like to further investigate robust techniques to sample gradients, for example by improving our weighting scheme.

Acknowledgements

This work was supported by the Swiss National Science Foundation under project nr. 143886. The box scene in Figure 13 was modeled by Toshiya Hachisuka [Hachisuka et al. 2008] .

References

BHAT, P., ZITNICK, L., COHEN, M., AND CURLESS, B. 2010. GradientShop: A gradient-domain optimization framework for image and video filtering. *ACM Trans. Graph. 29*, 2, 10:1–10:14.

BOLIN, M. R., AND MEYER, G. W. 1995. A frequency based ray tracer. In *Proc. ACM SIGGRAPH 95*, 409–418.

CHEN, Q., LI, D., AND TANG, C.-K. 2013. Knn matting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 35*, 9 (Sept), 2175–2188.

DAMMERTZ, H., SEWTZ, D., HANIKA, J., AND LENSCH, H. P. A. 2010. Edge-avoiding À-trous wavelet transform for fast global illumination filtering. In *Proc. High Performance Graphics 2010*, 67–75.

DAYAL, A., WOOLLEY, C., WATSON, B., AND LUEBKE, D. 2005. Adaptive frameless rendering. In *Proc. Eurographics Symposium on Rendering 2005*.

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

EGAN, K., HECHT, F., DURAND, F., AND RAMAMOORTHI, R. 2011. Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Trans. Graph. 30*, 2, 9:1–9:13.

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

HASTINGS, W. 1970. Monte Carlo samping methods using Markov chains and their applications. *Biometrika 57*, 1, 97–109.

JAKOB, W., AND MARSCHNER, S. 2012. Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph. 31*, 4 (July), 58:1–58:13.

KAPLANYAN, A. S., HANIKA, J., AND DACHSBACHER, C. 2014. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 33*, 4.

KELEMEN, C., SZIRMAY-KALOS, L., ANTAL, G., AND CSONKA, F. 2002. A simple and robust mutation strategy for the Metropolis light transport algorithm. *Comput. Graph. Forum 21*, 3, 531–540.

KOLLIG, T., AND KELLER, A. 2006. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, H. Niederreiter and D. Talay, Eds. Springer Berlin Heidelberg, 245–257.

KONTKANEN, J., RÄSÄNEN, J., AND KELLER, A. 2004. Irradiance filtering for monte carlo ray tracing. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, Springer, 259–272.

KRISHNAN, D., FATTAL, R., AND SZELISKI, R. 2013. Efficient preconditioning of laplacian matrices for computer graphics. *ACM Trans. Graph. 32*, 4 (July), 142:1–142:15.

LEHTINEN, J., KARRAS, T., LAINE, S., AITTALA, M., DURAND, F., AND AILA, T. 2013. Gradient-domain metropolis light transport. *ACM Trans. Graph. 32*, 4 (July), 95:1–95:12.

LEVIN, A., RAV-ACHA, A., AND LISCHINSKI, D. 2008. Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 10, 1699–1712.

MCCOOL, M. D. 1999. Anisotropic diffusion for Monte Carlo noise reduction. *ACM Trans. Graph. 18*, 2, 171–194.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., AND TELLER, E. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics 21*, 1087–1092.

OVERBECK, R., DONNER, C., AND RAMAMOORTHI, R. 2009. Adaptive wavelet rendering. *ACM Trans. Graph. 28*, 5, 140:1–140:12.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22*, 3, 313–318.

RAMAMOORTHI, R., MAHAJAN, D., AND BELHUMEUR, P. 2007. A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph. 26*, 1, 2:1–2:21.

ROUSSELLE, F., KNAUS, C., AND ZWICKER, M. 2011. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph. 30*, 6, 159:1–159:12.

ROUSSELLE, F., MANZI, M., AND ZWICKER, M. 2013. Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7, 121–130.

SEN, P., AND DARABI, S. 2012. On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph. 31*, 3 (June), 18:1–18:15.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 8 (Aug.), 888–905.

VEACH, E., AND GUIBAS, L. J. 1995. Optimally combining sampling techniques for Monte Carlo rendering. In *Proc. ACM SIGGRAPH 95*, 419–428.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *Proc. ACM SIGGRAPH 97*, 65–76.

VEACH, E. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University.

WALTER, B., KHUNGURN, P., AND BALA, K. 2012. Bidirectional lightcuts. *ACM Trans. Graph. 31*, 4 (July), 59:1–59:11.

WARD, G. J., AND HECKBERT, P. 1992. Irradiance gradients. In *Proc. Eurographics Workshop on Rendering '92*.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. In *Computer Graphics (Proc. ACM SIGGRAPH '88)*, 85–92.

Chapter 7

Gradient-Domain Bidirectional Path Tracing

Published as:
Gradient-Domain Bidirectional Path Tracing
Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand and Matthias Zwicker.
The definitive version of record is published by The Eurographics Association (2015), http://dx.doi.org/10.2312/sre.20151168

Gradient-Domain Bidirectional Path Tracing

Marco Manzi[1] Markus Kettunen[2] Miika Aittala[2] Jaakko Lehtinen[2,3] Frédo Durand[4] Matthias Zwicker[1]

[1]University of Bern [2]Aalto University [3]NVIDIA [4] MIT CSAIL

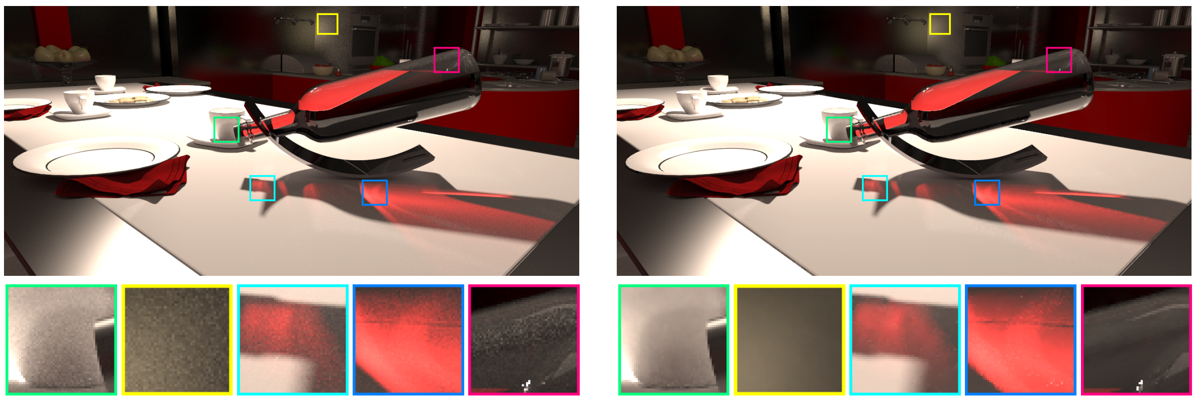

Bidirectional path tracing (BDPT)
Gradient-domain bidirectional path tracing (G-BDPT)

Figure 1: *We compare results of bidirectional path tracing (BDPT, left) versus gradient-domain bidirectional path tracing (G-BDPT, right) after thirty minutes of render time. While BDPT still exhibits visible residual noise, G-BDPT is free of artifacts nearly everywhere with the exception of some difficult regions around caustics.*

Abstract
Gradient-domain path tracing has recently been introduced as an efficient realistic image synthesis algorithm. This paper introduces a bidirectional gradient-domain sampler that outperforms traditional bidirectional path tracing often by a factor of two to five in terms of squared error at equal render time. It also improves over unidirectional gradient-domain path tracing in challenging visibility conditions, similarly to how conventional bidirectional path tracing improves over its unidirectional counterpart. Our algorithm leverages a novel multiple importance sampling technique and an efficient implementation of a high-quality shift mapping suitable for bidirectional path tracing. We demonstrate the versatility of our approach in several challenging light transport scenarios.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Gradient-domain methods have recently been introduced as efficient, general techniques for physically-based rendering [LKL*13, MRK*14, KMA*15]. Instead of directly estimating the radiance responses for each image pixel, they produce unbiased estimates of the *finite-difference gradients* between neighboring pixels by deterministically shifting paths between pixels. In a post-process step, the gradient estimates are integrated with a conventionally sampled, noisy "guide image" by solving the discrete screened Poisson equation. Together, these steps yield images with lower variance, and gradient-domain methods reduce the required render time to achieve the same quality compared to traditional samplers.

While gradient-domain rendering was originally proposed in the Markov Chain Monte Carlo (Metropolis) context, an upcoming paper [KMA*15] shows, both by theory and example, that similar benefits can also be claimed by a gradient-domain extension of standard path tracing with next event estimation. It is well known, however, that unidirectional path tracing is ineffective in scenes where light sources cannot be reached easily by tracing paths incrementally from the eye. Bidirectional path tracing deals with these situations much more robustly by constructing subpaths both starting at the eye and at light sources, and forming complete paths by making all possible connections.

In this paper, our objective is to combine the advantages of bidirectional path tracing and gradient-domain rendering. We describe a bidirectional gradient-domain light transport sampler (G-BDPT) that builds on bidirectional path tracing (BDPT). G-BDPT is useful in similar situations as conventional BDPT. This is the case in scenes with realistic light sources enclosed by light fixtures, or when the directly lit area is small, i.e., when sources contribute mainly indirect illumination. In both scenarios, connecting to light sources via shadow rays often fails, leading to excessive noise in unidirectional path tracers. In addition, we develop a novel multiple importance sampling technique, and describe an efficient implementation of a high-quality shift mapping to reduce sampling artifacts. Our results show that G-BDPT performs consistently better than its non-gradient counterpart, and that it yields significant improvement over standard (gradient) path tracing in scenarios that benefit from bidirectional sampling.

In summary, we make the following contributions:

- A bidirectional gradient-domain rendering algorithm (G-BDPT) based on a bidirectional light transport sampler;
- A multiple importance sampling (MIS) technique that combines MIS on gradients with conventional MIS for BDPT;
- An efficient implementation of a high-quality shift mapping using a modification of conventional BDPT path sampling.

2. Related Work

We base our work on the path space formulation of light transport due to Veach [Vea98]. That is, the intensity I_j for each pixel j in the image is obtained by integrating the radiance carried by all light paths with pixel filters:

$$I_j = \left(h(x) * \int_\Omega f(x,\bar{p})\,\mathrm{d}\mu(\bar{p})\right)(x_j). \tag{1}$$

Here x is a pixel position, the $\bar{p}$ range over the set of all additional path parameters Ω, $f(x,\bar{p})$ is the image contribution function, and $h(\cdot)$ is the (shift-invariant) pixel filter. We obtain the value I_j of pixel j by evaluating the convolution at its position x_j.

Several Monte Carlo methods have been proposed for evaluating Equation 1. In particular, constructing light paths using successive independent sampling of scattering events results in path tracing [Kaj86]; combining the results of successive independent sampling from the camera and from the light yields bidirectional path tracing [LW93, VG94]. While not always superior to standard path tracing, bidirectional sampling is particularly effective in reducing noise in scenes with small, difficult to reach light sources. Our gradient-domain bidirectional sampler retains this significant advantage. In another vein, Markov Chain Monte Carlo methods perform random walks on light paths instead of drawing independent samples [VG97, KSKAC02].

2.1. Gradient-Domain Rendering

We cursorily describe the necessary theoretical background on gradient-domain rendering, and refer the reader elsewhere for complete details [KMA*15]. Gradient-domain rendering techniques [LKL*13, MRK*14, KMA*15] build on strictly the same basis as previous Monte Carlo methods — that is, they aim to evaluate Equation 1 using Monte Carlo sampling. In contrast to regular (Markov Chain) Monte Carlo methods, they do this indirectly by sampling image gradients (differences in brightness between neighboring pixels) in addition to the pixel intensities, using pairs of correlated path samples. The final intensities for all pixels are found using the sampled gradients and pixel values by solving a screened Poisson equation. Recent work has demonstrated that this, perhaps surprisingly, yields a significant reduction in total integration error [KMA*15], and when used in the Markov Chain context, diverts computational effort to regions of path space that contribute to significant changes in the image [LKL*13]. Our work follows the same line of thought.

2.2. Gradient-Domain Path Tracing

In gradient-domain rendering, differences between pixel intensities are computed by directly evaluating the difference in light throughput between two paths separated by one pixel and integrating this over all paths. More precisely, we denote the difference between the intensities of two pixels i and j by $\Delta_{i,j}$. As recently shown [KMA*15], this can be written as the integral of a path difference function $g_{ij}(x,\bar{p})$ instead of the usual image contribution function $f(x,\bar{p})$ as

$$\begin{aligned}\Delta_{i,j} &= \left(h(x) * \int_\Omega f(x,\bar{p}) - f(T_{ij}(x,\bar{p}))\left|T'_{ij}\right|\mathrm{d}\mu(\bar{p})\right)(x_i)\\ &= \left(h(x) * \int_\Omega g_{ij}(x,\bar{p})\mathrm{d}\mu(\bar{p})\right)(x_i),\end{aligned} \tag{2}$$

where x is the image coordinate, $(x,\bar{p})$ is a light path with additional parameters $\bar{p}$ connecting a point on a light and a point on the sensor, f is the image contribution function, and T_{ij} is the *shift mapping* that deterministically maps a *base path* $(x,\bar{p})$ to a close-by *offset path* $T_{ij}(x,\bar{p})$. We only allow shifts that make sure that the offset path $T_{ij}(x,\bar{p})$ has the same pixel filter value as the base path, so we can express it as a single convolution. The factor $\left|T'\right| = |\partial T/\partial \bar{x}|$ denotes the determinant of the Jacobian of $T(\bar{x})$ accounting for the change of integration variables [LKL*13].

Symmetric Gradients The previous formulation assumes the shift mapping is a bijection on path space. But this is not the case in practice because the shift may fail due to numerical reasons (see Section 3.1). In addition, it assumes that we can sample all paths $(x, \bar{p})$ that lead to a non-zero offset path $f(T_{ij}(x, \bar{p}))$. Monte Carlo path tracers like BDPT only guarantee to sample all base paths with non-zero contribution $f(x, \bar{p})$, however, and we may miss some non-zero offset paths, leading to biased gradients. As shown by Kettunen et al. [KMA*15], we can ensure that all relevant paths are sampled in both pixels by using the symmetric formulation

$$\Delta_{i,j} = \left(h(x) * \int_\Omega w_{ij}(x, \bar{p}) g_{ij}(x, \bar{p}) \mathrm{d}\mu(\bar{p}) \right)(x_i) + \left(h(x) * \int_\Omega w_{ji}(x, \bar{p}) g_{ji}(x, \bar{p}) \mathrm{d}\mu(\bar{p}) \right)(x_j). \quad (3)$$

The two integrals sample the same difference, once by shifting from pixel i to j and vice versa. The multiple importance sampling weights w_{ij} and w_{ji} serve two purposes: they are normalized to add up to one, such that the two integrals correctly add up to the desired gradients, and they reduce variance by tempering the effect of the local squeezing of path space caused by the shift. Finally, we need to take into account that the shift may not be invertible for some parts of path space, which means the symmetric formulation cannot be evaluated in these cases. We deal with this by simply sampling the contributions of paths to the two pixels i and j separately, without applying any shift mapping, and add them to (respectively subtract them from) $\Delta_{i,j}$.

Gradient MIS To set the weights in Equation 3, the forward and inverse mappings are interpreted as two sampling techniques to obtain the same base path $\bar{x} = (x, \bar{p})$. This makes it possible to derive multiple importance sampling weights

$$w_{ij}(\bar{x}) = \frac{p(\bar{x})}{p(\bar{x}) + p(T_{ij}(\bar{x}))|T'_{ij}(\bar{x})|}. \quad (4)$$

Gradients with a large Jacobian determinant $|T'_{ij}(\bar{x})|$ obtain a MIS weight of approximately $1/|T'_{ij}(\bar{x})|$, which cancels their large contribution.

G-PT Algorithm The gradient-domain path tracing algorithm (G-PT) simply draws a number of base paths from each pixel, shifts them to the four horizontal and vertical neighbor pixels, evaluates the differences between throughputs, weighted as shown above, and accumulates the results in a throughput image and four additional gradient images. The process yields the inputs required by the screened Poisson solver. When the shift mapping is designed so that throughput differences between base and offset are small (including the effect of the Jacobian), the resulting gradient estimates have low variance, which translates to higher quality in the final reconstructed image [KMA*15].

3. Bidirectional Gradient Sampling

We now describe a gradient-domain version of bidirectional path tracing (G-BDPT). We follow the general outline of Kettunen et al. [KMA*15], and view the problem as formulating a bidirectional Monte Carlo sampler for Equation 3.

A direct translation of BDPT with multiple importance sampling to the gradient domain would, however, lead to a prohibitively expensive algorithm, because the number of individual paths sampled is large (all connections are made between the eye and light subpaths). The naive algorithm that applies the shift mapping to each one turns out to be too expensive. We alleviate the issue by selectively removing some bidirectional connection strategies. This reduction in work allows us to use a more sophisticated shift mapping compared to [KMA*15], which we demonstrate to yield a net performance win.

3.1. Shift Mapping

For G-BDPT we use the shift mapping proposed for gradient-domain Metropolis rendering by Lehtinen et al. [LKL*13], which builds on the manifold perturbation technique by Jakob and Marschner [JM12]. They express their shift in the path parameterization by surface position, which is the natural parameterization for BDPT. Hence, we can directly reuse their mathematical formulation (and implementation). To apply the shift to a given path we need to classify its vertices as diffuse or specular, and we follow their approach using a threshold on the material roughness.

The intuition behind the G-BDPT shift is to preserve half-vectors at vertices classified as specular, while trying to connect to the base path as soon as possible. The main advantage over the shift proposed for G-PT is that it always connects to the base path at the second diffuse vertex (starting from and excluding the eye), independent of any specular vertices before that. In contrast, the G-PT shift requires two *consecutive* diffuse vertices to reconnect to the base path. This means that the G-BDPT shift generally produces more similar base-offset pairs than the G-PT shift.

We briefly review the definition of the G-BDPT shift as introduced by Lehtinen et al. [LKL*13]. Let us describe a path $\bar{x}$ as a sequence of vertex positions $\mathbf{x}_i$, $\bar{x} = \langle \mathbf{x}_0, \ldots, \mathbf{x}_n \rangle$. We formulate the shift mapping as the concatenation of a path reparameterization, a *simple shift* that only modifies the image plane intersection of the reparameterized path without changing the other parameters, and a reparameterization back. We classify vertices into diffuse and specular vertices based on material roughness as mentioned before. The key idea now is to design the reparameterization such that the *simple shift* described above preserves half-vectors at specular vertices, while connecting to the base path as soon as possible.

Let a, b, and c be the indices of the first three vertices classified as diffuse along the path starting at the eye (including

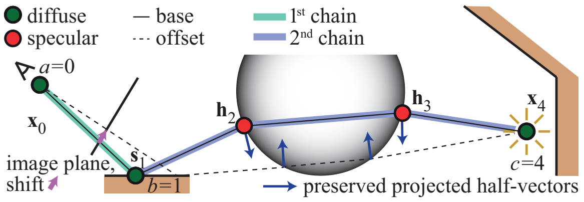


Figure 2: *Visualization of the reparameterization and the shift mapping in G-BDPT. Starting at the eye, the shift preserves two consecutive* half-vector chains *between the first three diffuse vertices a, b, and c along the path. It always connects to the base path at the third diffuse vertex (including the eye). In the figure, the first half-vector chain is* empty. *In contrast the G-PT shift [KMA*15] cannot connect to the vertex c on the light source.*

the eye vertex itself, which is classified as diffuse; hence we always have $a = 0$). In the reparameterization, we represent the specular vertices between the eye and b, and the specular vertices between b and c using projected half-vectors $\mathbf{h}_i$ (half-vectors projected onto local tangent planes) instead of vertex positions $\mathbf{x}_i$. We call vertices i with $a < i < b$ and $b < i < c$ the first and second *half-vector chain*, respectively. The first chain is *empty* if $b = 1$, and the second is empty if $c = b+1$. We write our reparameterization as $\hat{x} = \langle \mathbf{x}_0, \mathbf{s}_1, \mathbf{h}_1, \dots, \mathbf{h}_b - 1, \mathbf{h}_{b+1}, \dots \mathbf{h}_{c-1}, \mathbf{x}_c, \dots, \mathbf{x}_n \rangle$, where $\mathbf{s}_1$ is the image plane intersection of the path, and the position of vertex b is determined implicitly by the parameters of previous vertices. In this parameterization, the simple shift only moves $\mathbf{s}_1$ to a neighboring pixel. We illustrate the reparameterization and the shift in Figure 2.

We implement the shift by moving $\mathbf{s}_1$ to a neighbor pixel, and re-tracing the first specular chain from the eye, which yields vertex b on the offset path. We reconnect offset vertex b to the base path via the second half-vector chain by applying the manifold perturbation by Jakob and Marschner [JM12]. To formulate the Jacobian of the shift, let us denote the shifted offset path in the original parameterization by surface position as $\bar{y}$, and in the reparameterization using half-vectors as $\hat{y}$. The Jacobian determinant of the shift is then

$$|T'(\bar{x})| = \left|\frac{\partial \bar{y}}{\partial \bar{x}}\right| = \left|\frac{\partial \mathbf{x}_0, \dots, \mathbf{x}_n}{\partial \mathbf{y}_0, \dots, \mathbf{y}_n}\right| = \left|\frac{\partial \bar{y}}{\partial \hat{y}}\right| \left|\frac{\partial \hat{x}}{\partial \bar{x}}\right|. \quad (5)$$

We compute the Jacobian determinants of the reparameterizations $|\partial\bar{y}/\partial\hat{y}|$ and $|\partial\hat{x}/\partial\bar{x}|$ as described by Lehtinen et al. [LKL*13].

3.2. Efficient Gradient Sampling

The computational cost of the G-BDPT shift is not negligible, and we need to employ it carefully to avoid large overheads. A naive implementation of the shift mapping would apply it separately to each path sampled by BDPT. For each eye and light subpath, however, BDPT samples all paths that can be obtained by connecting these subpaths. Shifting all connected paths and computing the Jacobians separately from scratch is prohibitively expensive. We reduce this cost by slightly modifying the usual BDPT sampling strategy.

Our key modification of usual BDPT is to omit sampling techniques that include a specular vertex (according to our classification) as a connecting vertex between eye and light subpaths. Omitting these sampling techniques has little impact on the effectiveness of BDPT, since connections involving non-diffuse vertices typically contribute very little. On the other hand, it allows us to reduce the cost to compute the shift mappings and their Jacobians.

For the Jacobian of the shift mapping described above only vertices $\leq c$ are relevant, since the shift is independent of the others. With our restriction on BDPT sampling techniques, vertices $\leq c$ may have been sampled in only three different ways illustrated in Figure 3: (i) all vertices (both half-vector chains) are sampled on the eye path, (ii) vertices up to and including b (only the first half-vector chain) are sampled on the eye path, and vertices $> b$ on the light path, (iii) only vertex a is sampled on the eye path (the second half-vector chain is sampled on the light path). We call such paths *light tracing paths*. Case (ii) implies $c = b+1$ (the second half-vector chain is empty), since we do not make connections with non-diffuse vertices. Also, in this case the Jacobian is given by vertices $\leq b$, since the shift is independent of vertex $c = b+1$. Similarly, case (iii) implies $b = a+1$ (the first half-vector chain is empty)[†].

We take advantage of these observations as follows: Given an eye subpath $\bar{x}^E$, let us again denote the indices of its first three diffuse vertices a, b, c. We then apply the shift to vertices $\mathbf{x}_a^E, \dots, \mathbf{x}_c^E$, yielding an offset path $\bar{y}^E$ for the eye subpath. Under the previous considerations, this is sufficient to construct *all* connected offset paths for cases (i), where the connection is with a vertex $\geq c$ on the eye subpath, and (ii), where the connection is with vertex b . Hence we need only two different Jacobians for all these paths, in case (i) for the shift of both half-vector chains $a, \dots, b, \dots, c$,

$$|T^{(i)}(\bar{x}^E)| = \left|\frac{\partial \mathbf{x}_a^E, \dots, \mathbf{x}_c^E}{\partial \mathbf{y}_a^E, \dots, \mathbf{y}_c^E}\right|, \quad (6)$$

and in case (ii) for only the first chain $a, \dots, b$,

$$|T^{(ii)}(\bar{x}^E)| = \left|\frac{\partial \mathbf{x}_a^E, \dots, \mathbf{x}_b^E}{\partial \mathbf{y}_a^E, \dots, \mathbf{y}_b^E}\right|. \quad (7)$$

Dealing with light tracing paths in case (iii) is more expensive. Given a ligth subpath $\bar{x}^L$, each of its diffuse vertices needs to be connected to the eye to form a complete light tracing path. For each connected light tracing path we need

[†] Since we assume a pinhole camera, we omit a fourth case where both chains and the eye vertex are sampled from the light.

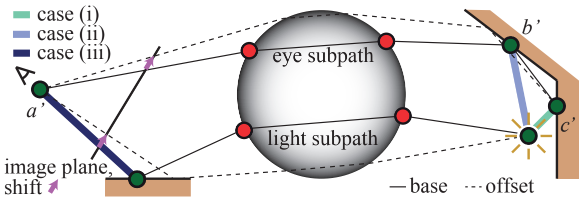


Figure 3: *Visualization of modified sampling strategy in G-BDPT. We do not sample subpath connections that include a specular vertex. We color code connections according to three cases: (i) both half-vector chains are sampled from the eye (the second chain is empty here, but this is not always the case); (ii) the first half-vector chain is sampled from the eye (the second one is always empty, since we omit connections to non-diffuse vertices); (iii) the second half-vector chain is sampled from the light (implying the first half-vector chain is always empty). To reduce clutter, the figure does not show connections of type (i) and (ii) to the last vertex on the light subpath, and it does not show the corresponding subpath connections on the offset paths.*

to recompute a shift and its Jacobian. Again, we compute all Jacobians as described by Lehtinen et al. [LKL*13].

3.3. Multiple Importance Sampling

Let us use the common notation (s,t) to represent the different sampling strategies in BDPT, where s is the number of vertices on the light subpath, and t the number of vertices on the eye subpath. In BDPT any given path with n vertices can be sampled using all techniques (s,t) where $s+t=n$. Multiple importance sampling (MIS) introduces a weight for each sampling technique to reduce variance in a provably good manner [VG95]. Here we extend usual MIS for G-BDPT by combining it with the gradient-MIS technique outlined in Section 2.

For each gradient sample we not only consider all the potential sampling techniques that could be used to sample the base path, but we also take into account that the gradient could be sampled using either the forward or the inverse mapping, as described in Section 2. The combined MIS weight for a gradient sample using the balance heuristics is then

$$w_{ij;st}(\bar{x}) = \frac{p_{s,t}(\bar{x})}{\sum_{k=0}^{k\leq s+t} p_{k,s+t-k}(\bar{x}) + p_{k,s+t-k}(T_{ij}(\bar{x}))|T'_{ij}|}, \quad (8)$$

where $p_{s,t}(\bar{x})$ is the probability density function (PDF) for the (s,t) sampling technique evaluated for the base path $\bar{x}$. Note that this implies that the sum of the weights over the two gradient directions is normalized, that is, $w_{ij;st}(\bar{x}) + w_{ji;st}(T_{ij}(\bar{x})) = 1$.

Figure 4 illustrates the effectiveness of our combined MIS

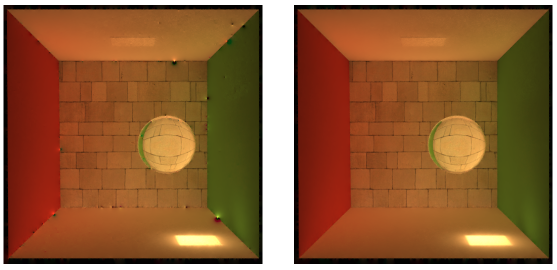


Figure 4: *Comparison of MIS techniques in G-BDPT using L_2 reconstruction. Left: G-BDPT with modified MIS weights given by Equation 9, that is, the balance heuristics for BDPT sampling of base paths with a discrete case distinction to avoid double counting of gradients. Right: G-BDPT with our combined MIS from Equation 8. Combined MIS effectively reduces gradient sampling artifacts in concave regions.*

approach. We compare our combined MIS weights from Equation 8 with a modified approach that only performs conventional BDPT MIS, but does not consider the two techniques to sample gradients. More precisely, the modified approach uses weights based on the balance heuristics

$$w_{ij;st}(\bar{x}) = \frac{r_{i,j}(\bar{x})p_{s,t}(\bar{x})}{\sum_{k=0}^{k\leq s+t} p_{k,s+t-k}(\bar{x})}, \quad (9)$$

where the factor $r_{i,j}(\bar{x})$ is necessary to make sure we do not double count gradient contributions in the symmetric formulation (Equation 3). We set $r_{i,j}(\bar{x}) = 1/2$ if there is a technique k that samples the gradient from the opposite direction, that is, there is a k with $p_{k,s+t-k}(T_{ij}(\bar{x})) > 0$. Otherwise, the gradient can be sampled only from one direction, and we set $r_{i,j}(\bar{x}) = 1$. The figure shows that combined MIS effectively reduces sampling artifacts in concave regions, which otherwise can only be avoided with sophisticated shift mappings [MRK*14].

4. Implementation

We implemented G-BDPT on top of the standard BDPT implementation in the freely available Mitsuba renderer [Jak12]. The basic structure of G-BDPT is very similar to BDPT, as shown in the pseudocode in Algorithm 1. For every sample, we draw an eye subpath $\bar{x}^E$ and a light subpath $\bar{x}^L$ (line 1). Then (lines 3-5), for each of the four horizontal and vertical neighbor pixels, we apply the shift mapping to the eye subpath to construct four shifted eye subpaths $\bar{y}^{E,j} = T_{ij}(\bar{x}^E)$. For each we obtain the corresponding Jacobians $|T_{ij}^{(i)}|$ and $|T_{ij}^{(ii)}|$ for cases (i) and (ii) as described in Equation 6 and Equation 7.

We then construct all complete base paths $\bar{x}$ by connecting $\bar{x}^E$ and $\bar{x}^L$ with all valid connection strategies (line 6), skipping connections between vertices classified as specular

```
      Input: Scene and camera specification, total number of
             bidirectional samples $N$.
      Output: Image $I$, gradient images $\Delta_{\cdot,\cdot}$.
      for all pixels and samples do
[1]       $\bar{x}^E, \bar{x}^L$ = sample eye and light subpath
[2]       $i$ = screen-space position of $\bar{x}_E$
          $a, b, c$ = first three diffuse vertices on $\bar{x}^E$
          for all neighbours $j$ of $i$ do
[3]           $\bar{y}^{E,j} = T_{ij}(\bar{x}^E)$ // shift eye subpath
[4]           $|T_{ij}^{(i)}| = \left|\frac{\partial[\mathbf{y}_a^{E,j},\ldots,\mathbf{y}_c^{E,j}]}{\partial[\mathbf{x}_a^E,\ldots,\mathbf{x}_c^E]}\right|$ // case (i) Jacobian
[5]           $|T_{ij}^{(ii)}| = \left|\frac{\partial[\mathbf{y}_a^{E,j},\ldots,\mathbf{y}_b^{E,j}]}{\partial[\mathbf{x}_a^E,\ldots,\mathbf{x}_b^E]}\right|$ // case (ii) Jacobian
          end
          for all connection strategies (s,t) do
[6]           $\bar{x} = connect(s, t, \bar{x}^E, \bar{x}^L)$ // base path
[7]           $i$ = screen-space position of $\bar{x}$
              $a, b, c$ = first three diffuse vertices on $\bar{x}$
[8]           if case (iii) // light tracing path, $t = 1$
              then
                  for all neighbours $j$ of $i$ do
[9]                   $\bar{y} = T_{ij}(\bar{x})$ // recompute shift
[10]                  $|T_{ij}| = \left|\frac{\partial[\mathbf{y}_a,\ldots,\mathbf{y}_c]}{\partial[\mathbf{x}_a,\ldots,\mathbf{x}_c]}\right|$ // Jacobian
[11]                  $\Delta_{ij} = \Delta_{ij} +$
                      $w_{ij;st}(\bar{x})\left[f(\bar{y})|T_{ij}| - f(\bar{x})\right]/p_{st}(\bar{x})$
                  end
              else
                  for all neighbours $j$ of $i$ do
[12]                  $\bar{y} = connect(s, t, \bar{y}^{E,j}, \bar{x}^L)$
                      if case (ii) then
[13]                      $\Delta_{i,j} = \Delta_{i,j} +$
                          $w_{ij;st}(\bar{x})\left[f(\bar{y})|T_{ij}^{(ii)}| - f(\bar{x})\right]/p_{st}(\bar{x})$
                      else
[14]                      $\Delta_{i,j} = \Delta_{i,j} +$
                          $w_{ij;st}(\bar{x})\left[f(\bar{y})|T_{ij}^{(i)}| - f(\bar{x})\right]/p_{st}(\bar{x})$
                      end
                  end
              end
[15]          $I_i = I_i + w_{i;st}(\bar{x}) f(\bar{x})/p_{st}(\bar{x})$
          end
      end
[16]  $I = I/N; \Delta_{\cdot,\cdot} = \Delta_{\cdot,\cdot}/N$
[17]  Reconstruct($I$, $\Delta_{\cdot,\cdot}$, $\alpha$)
```

Algorithm 1: *Pseudocode for gradient-domain bidirectional path tracing (G-BDPT).*

according to our roughness criterion. We also determine the pixel index i of the base path (line 7, this may be different from the pixel index corresponding to the eye subpath). Next (line 8) we check if the current base path is a light tracing path, that is, whether it connects a vertex on the light subpath directly to the eye (case (iii), meaning $t = 1$). Because in these cases the eye subpath has only one vertex (the eye), we cannot make use of the shifted eye subpaths $\bar{y}^{E,j}$ for shifting these paths. Instead, we must apply the shift mapping (and compute its Jacobian) to each light-traced path separately (line 9 and 10). We then compute the gradient sample contribution and weight it by the MIS weight defined in Equation 8 (line 11). For connection strategies that do not directly connect with the eye vertex, we form the offset path by connecting the shifted eye subpath $\bar{y}^{E,j}$ with the light subpath $\bar{x}^L$ (line 12). We account for the two cases (i) and (ii) by choosing the correct Jacobian determinants (lines 13 and 14).

We also store the value of the base sample in the primal image (line 15). The weight $\omega_{i;st}$ is the usual power or balance heuristic, not the one from Equation 8.

Finally, we normalize both the primal image and the gradient images by the total number of bidirectional samples (line 16), which also accounts for the light paths that are distributed non-uniformly over the image. Then we perform screened Poisson reconstruction on the output of the renderer (line 17), as in previous work [KMA*15]. Our L_1 solver is based on iteratively reweighed least squares (IRLS) implemented through the conjugate gradient method in CUDA. Its performance is less than a second for a 720p image.

As an important detail, we treat some situations in a slightly different manner than implied by the pseudocode. This is when in lines 9 and 12 the offset paths $\bar{y}$ cannot be constructed because the shift failed for numerical reasons, or when these offset paths are blocked, and when the base path $\bar{x}$ (line 6) is blocked. In these cases we fall back to naive gradient sampling, that is, we simply set the contribution of the offset path to zero, and we use conventional MIS weights for the base path, instead of combined MIS (Equation 8). In the case of failing shifts, this is our only option. In the case of blocked paths, it allows us to take a number of early exits in our implementation that lead to some performance gains.

5. Results and Discussion

We evaluate G-BDPT by comparing to standard bidirectional path tracing (BDPT), standard path tracing (PT), and gradient-domain path tracing (G-PT) [KMA*15]. All methods were implemented in the Mitsuba renderer [Jak12]. We generated reference images using BDPT with 32000 samples per pixel. Except where expressly stated otherwise, all evaluations use L_1 reconstruction. All results are computed with 24 rendering threads on a workstation with dual Intel Xeon E5645 processors with a total of twelve cores at 2.5GHz.

For comparisons, we use relative mean squared error (relMSE), which we compute as relMSE = average$[(X - R)^2/(R^2 + 0.001)]$, where R is a reference pixel and X our estimate. With all four compared methods, two of our test scenes (GLASS EGG and BOTTLE) suffer from massive spike noise due to difficult caustics. Because this corrupts the metric, we ignored the 0.01% of the highest pixel errors in the relMSE computation in these scenes[‡].

5.1. Evaluation of G-BDPT

In Figure 6, we visually compare the four methods at equal render time. Full resolution images with error scores are also provided as supplemental material. In Figure 5, we plot the numerical convergence of all methods. We now briefly discuss each scene.

GLASS EGG is a standard benchmark scene for BDPT. The light from the lamp on the left illuminates most of the scene indirectly, making it hard for unidirectional PT to connect paths with the light source. Additionally, the glass egg on the table is lit directly by a second light source which creates a strong caustic that is notoriously hard to sample with PT. Unsurprisingly, BDPT performs much better in this scene than PT. The behaviour of G-PT and G-BDPT is more interesting: G-PT reduces the error compared to PT by a very large margin, but for low sampling rates, it improves sublinearly with time. The reason for this is that direct caustics lead to strong spike noise with unidirectional sampling, and the L_1 reconstruction suppresses the effect of such noise as outliers. This means the L_1 reconstruction reduces the error significantly at the price of removing most of the caustic. With higher sampling rates, the convergence rate becomes more linear again since less of the caustic is removed in reconstruction. Since a bidirectional sampler can resolve caustics much better, G-BDPT does not suffer from this. Compared to its non-gradient counterpart, G-BDPT leads to an improvement of a factor of five, in terms of render time to same quality.

DOOR is a benchmark scene for testing rendering algorithms under challenging illumination conditions. All visible light has to pass from another room through a thin crack of the door into the visible part of scene. For an unidirectional sampler it is very unlikely to find a path that connects the eye to the light, since it has to randomly pass through the thin crack. For bidirectional path sampler this is slightly easier since it is enough if either the eye subpath or the light subpath randomly passes through the crack. However, the somewhat higher chance of finding valid paths is nullified by the higher overhead of BDPT. Therefore, the performance of both non-gradient algorithms is approximatively equal. Still, G-BDPT outperforms G-PT by a factor of approximately two. This is most likely due to the superior shift-mapping (see Section 5.3).

BOTTLE is a complex scene with many glossy and specular surfaces, and a prominent direct caustic due to a small area light source. Similar to GLASS EGG, PT and G-PT fail to capture the caustic in a satisfactory way, while BDPT and G-BDPT succeed. The non-linear convergence for low sampling rates is stronger here for G-PT than in GLASS EGG because the caustic covers a bigger fraction of the image. Again, G-BDPT does not suffer from this, and provides a benefit over BDPT by a factor of two.

Since the performance of the gradient-domain methods is highly dependent on the performance of the underlying sampler, we also analyzed scenes where bidirectional sampling strategies are not beneficial.

BATHROOM is a complex scene that is illuminated by a large light source from the outside through a glass window. As the light source is large, even the unidirectional sampler has a good chance to randomly hit the source, even in presence of the glass in between, making bidirectional sampling of not much use. Because of this, the unidirectional methods (PT and G-PT) both have, at higher sample counts, an error about 30% lower than their bidirectional counterparts. Nonetheless, G-BDPT still improves over BDPT by a factor of six.

Finally, SPONZA is a simple scene consisting of diffuse surfaces only that are illuminated by a large area light. Since there are no caustics and no challenging illumination conditions, the overhead of bidirectional sampling is not amortized in equal time comparisons. Thus in comparison both bidirectional samplers are beaten by the unidirectional ones roughly by a factor of two. Again, G-BDPT improves on its non-gradient counterpart, here by almost an order of magnitude.

Scene	G-PT	G-BDPT w/o LTP	G-BDPT w/ LTP	G-BDPT HV
Glass Egg	x2.87	x3.83	x5.71	x5.56
Door	x3.18	x3.34	x3.49	x3.48
Bottle	x2.16	x3.42	x3.48	x3.72
Bathroom	x2.39	x3.42	x3.83	x4.12
Sponza	x2.47	x3.88	x4.03	x4.14

Table 1: *The overhead of the different gradient-domain methods compared to their non-gradient counterparts at equal number of base samples. An overhead of 5 means that gradient and conventional samples are equally expensive. We compare G-BDPT in three set-ups: without light tracing paths and with the Manifold perturbation shift mapping (3rd column), with light tracing paths and the Manifold perturbation shift mapping (4th column), and with light tracing paths and the half-vector preserving mapping (5th column).*

5.2. Computational Overhead

Intuitively, there are two reasons why gradient-domain rendering improves over conventional approaches at equal render time: first, sampled gradient samples have less

[‡] All images, including the references, are available in the supplemental material for inspection.

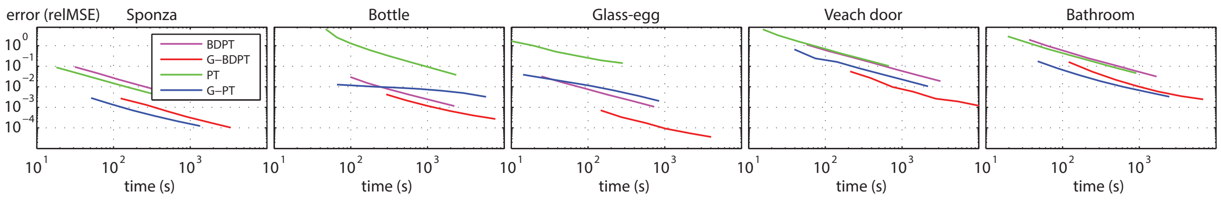


Figure 5: *Error plots of the scenes used in this paper, comparing bidirectional path tracing (BDPT), gradient-domain bidirectional path tracing (G-BDPT), path tracing (PT), and gradient-domain path tracing (G-PT) at equal render time. The error is measured as relative mean squared error (relMSE).*

variance than sampled pixels in general, as shown recently [KMA*15]; and second, the overhead for computing a gradient sample is typically cheaper than obtaining a conventional sample, because it does not require tracing a full path.

To show this empirically, we measured the overhead of gradient-sampling by comparing the rendering time of gradient and non-gradient methods with the same number of base samples per pixel. That is, we compare gradient rendering with n base samples and $4n$ offset paths to conventional rendering with n samples. Hence at equal costs per sample, gradient-domain rendering would have an overhead factor of 5 in this comparison. We summarize our empirical results in Table 1. For G-PT (first column) we measured a scene-dependent overhead factor of 2.2 to 3.2, which agrees with Kettunen et al. [KMA*15]. For G-BDPT we report on two configurations (second and third column), first without the expensive sampling strategies for light tracing paths, and then with it. To make the comparison meaningful we configured BDPT in the same way. The results show that without light tracing paths the overhead is roughly around 3.5 for all scenes. Including light tracing paths increases the overhead in general, but the increase is highly scene dependent. The overhead can even become larger than 5, meaning that gradient samples become more expensive than conventional samples. This is the case in GLASS EGG where many, potentially long light tracing paths need to be shifted for each base path. The different overheads of G-PT and G-BDPT without light tracing paths can probably be attributed to different levels of code optimization.

5.3. Evaluation of the shift mapping

To justify our decision to use the manifold perturbation shift mapping from Lehtinen et al. [LKL*13] we compared it to the simpler "half-vector shift mapping" from Kettunen et al. [KMA*15]. In a nutshell, this shift preserves the half-vectors of the base path along the offset path starting at the eye, and reconnects the offset path to the base path as soon as it encounters two consecutive diffuse vertices. For the comparison we implemented both shift mappings in our G-BDPT framework. In all tested scenes, G-BDPT with the manifold perturbation shift yielded more pleasing results than with the half-vector shift at equal time. Figure 7 shows an example. Surprisingly, despite the gradient descent optimization required in manifold perturbation, the overhead compared to the half-vector shift is very small in practice. We report empirical measurements in Table 1, rightmost column, which shows that the overheads are consistently very similar. There are two reasons. First, the gradient descent is only applied for a small fraction of all shifts; second, the manifold perturbation shift can often connect earlier to the base path, and thus must shift fewer vertices.

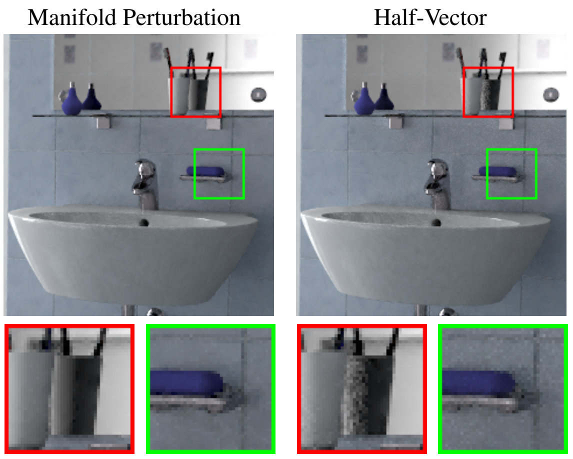


Figure 7: *Comparison of the Manifold perturbation shift mapping [LKL*13] and the half-vector preserving shift mapping [KMA*15] on the BATHROOM scene at 1024spp.*

6. Conclusions

We presented gradient-domain bidirectional path tracing, a gradient-domain rendering algorithm that significantly and consistently improves performance in comparison to standard bidirectional path tracing. Compared to previous unidirectional gradient-domain path tracing, this is most useful in scenarios where the additional cost of bidirectional sampling is justified, in particular for scenes with caustics or light sources that are not easily reachable for unidirectional path tracers. Our method retains the attractive properties of gradient-domain path tracing in that it is an unbiased estimator when using L_2 reconstruction, and can be used in conjunction with the more outlier-friendly L_1 reconstruction. In addition, we have shown that a shift mapping based on Manifold perturbation is advantageous compared to the half-vector preserving shift proposed previously for gradient-domain path tracing, providing improved image quality at almost no additional cost.

While this paper shows the viability and benefits of gradient-domain bidirectional path tracing, there are many attractive avenues for future research to further reduce variance and sampling artifacts. We will investigate more powerful reconstruction techniques, combining different shift mappings, and more advanced gradient sampling techniques.

Acknowledgments

The "Door" and "Glass-Egg" scenes were modelled after scenes by Eric Veach by Miika Aittala, Samuli Laine, and Jaakko Lehtinen. The "Sponza" scene is courtesy of Marko Dabrovic. The "Bottle" and "Bathroom" scenes have been ported to Mitsuba by Tiziano Portenier. This work was partially supported by the Academy of Finland grant no. 277833, NSF grant no. IIS-1420122, Swiss National Science Foundation grant no. 143886, and by the Helsinki Doctoral Education Network in Information and Communications Technology (HICT).

References

[Jak12] JAKOB W.: Mitsuba v0.4. http://mitsuba-renderer.org, 2012. 5, 7

[JM12] JAKOB W., MARSCHNER S.: Manifold exploration: A Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph. 31*, 4 (2012), 58:1–58:13. 3, 4

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proc. ACM SIGGRAPH 86* (1986), pp. 143–150. 2

[KMA*15] KETTUNEN M., MANZI M., AITTALA M., LEHTINEN J., DURAND F., ZWICKER M.: Gradient-domain path tracing. *ACM Trans. Graph. (to appear) 35*, 4 (2015). 1, 2, 3, 4, 6, 7, 8

[KSKAC02] KELEMEN C., SZIRMAY-KALOS L., ANTAL G., CSONKA F.: A simple and robust mutation strategy for the Metropolis light transport algorithm. *Comput. Graph. Forum 21*, 3 (2002), 531–540. 2

[LW93] LAFORTUNE E. P., AND WILLEMS Y. D.: Bi-Directional Path Tracing. *Proceedings of Computergraphics* (1993), p. 145–153. 2

[LKL*13] LEHTINEN J., KARRAS T., LAINE S., AITTALA M., DURAND F., AILA T.: Gradient-Domain Metropolis Light Transport. *ACM Trans. Graph. 32*, 4 (2013). 1, 2, 3, 4, 5, 8

[MRK*14] MANZI M., ROUSSELLE F., KETTUNEN M., LEHTINEN J., ZWICKER M.: Improved sampling for gradient-domain metropolis light transport. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 178:1–178:12. 1, 2, 5

[Vea98] VEACH E.: *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, 1998. 2

[VG94] VEACH E., AND GUIBAS L. J.: Bidirectional Estimators for Light Transport. *Eurographics Rendering Workshop Proceedings* (1994), pp. 147–162. 2

[VG95] VEACH E., GUIBAS L. J.: Optimally combining sampling techniques for Monte Carlo rendering. In *Proc. ACM SIGGRAPH 95* (1995), pp. 419–428. 5

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *Proc. ACM SIGGRAPH 97* (1997), pp. 65–76. 2

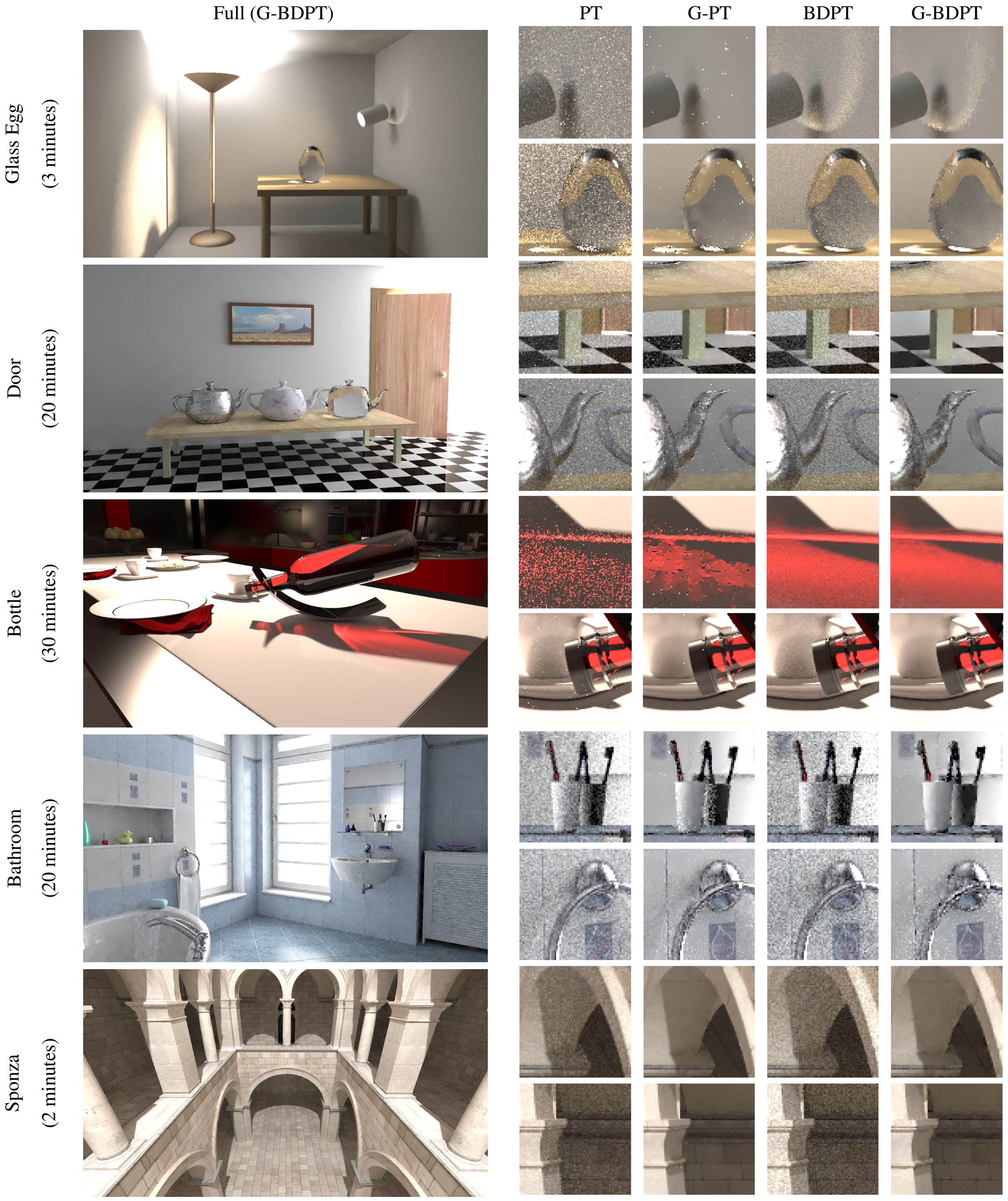


Figure 6: *Visual equal-time comparison of path tracing (PT), gradient-domain path tracing (G-PT), bidirectional path tracing (BDPT) and gradient-domain bidirectional path tracing (G-BDPT). We provide the full resolution images with numerical error measurements in the supplemental material.*

Chapter 8

Regularizing Image Reconstruction for Gradient-Domain Rendering with Feature Patches

Published as:
Regularizing Image Reconstruction for Gradient-Domain Rendering with Feature Patches
Marco Manzi, Delio Vicini and Matthias Zwicker.
The definitive version of record is published in Computer Graphics Forum 35(2),
http://dx.doi.org/10.1111/cgf.12829

Regularizing Image Reconstruction for Gradient-Domain Rendering with Feature Patches

M. Manzi, D. Vicini, and M. Zwicker

University of Bern, Switzerland

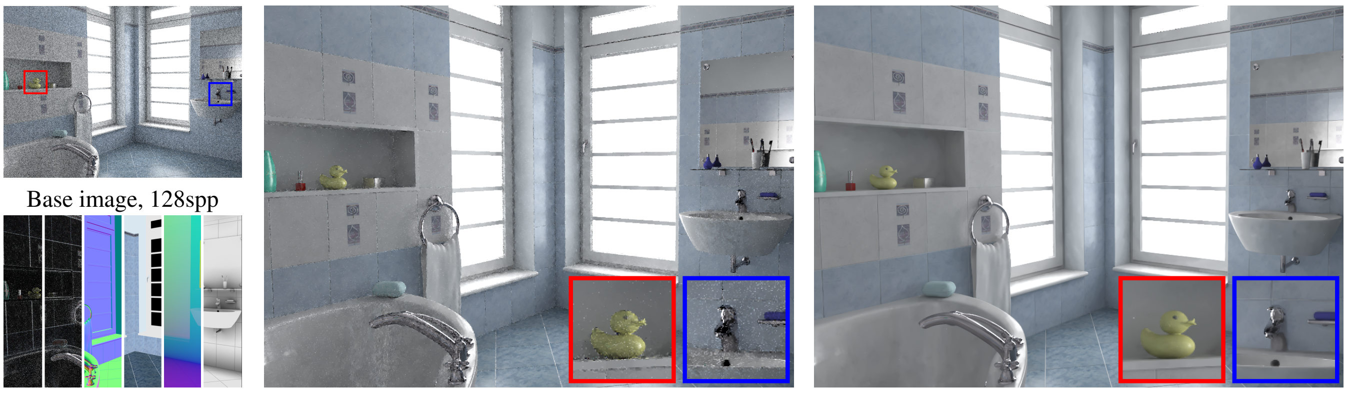

Base image, 128spp
Gradients and features
L_1 reconstruction, relative MSE 0.00662
Regularized reconstruction, relative MSE 0.00126

Figure 1: *Our regularized reconstruction for gradient-domain rendering obtains a high-quality image from a noisy base image, the sampled gradients, and auxiliary features (left). We (right) achieve significantly better images than standard L_1 reconstruction (middle). The depicted features are, from left to right, the vertical and horizontal gradients, normals, texture values, positions and ambient occlusion values.*

Abstract

We present a novel algorithm to reconstruct high-quality images from sampled pixels and gradients in gradient-domain rendering. Our approach extends screened Poisson reconstruction by adding additional regularization constraints. Our key idea is to exploit local patches in feature images, which contain per-pixels normals, textures, position, etc., to formulate these constraints. We describe a GPU implementation of our approach that runs on the order of seconds on megapixel images. We demonstrate a significant improvement in image quality over screened Poisson reconstruction under the L_1 norm. Because we adapt the regularization constraints to the noise level in the input, our algorithm is consistent and converges to the ground truth.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms

1. Introduction

With the ongoing path tracing revolution in the movie industry [KFF*15], there has been a renewed interest in noise reduction for Monte Carlo rendering. Monte Carlo algorithms are attractive because they are conceptually simple, general, and based on a physical model of light transport. Noise artifacts, however, have remained a challenge for real world applications. Since the level of noise is inversely proportional to the square root of the number of samples per pixel, it is often impractical to eliminate visual artifacts in a brute force manner.

In this paper we build on gradient-domain rendering techniques, which sample finite difference image gradients, in addition to the usual pixel values, and then reconstruct final images by solving a screened Poisson problem using the sampled gradients and pixels. Gradient-domain rendering was originally proposed for Metropolis light transport [LKL*13], but recently Kettunen et al. [KMA*15] and Manzi et al. [MKA*15] showed that it also significantly reduces the error compared to standard (bidirectional) path tracing at equal computation time. Visual artifacts, however, often remain even at hundreds of samples per pixel as shown in Figure 1. Even if we minimize the L_1 error of the screened Poisson reconstruction, which leads to bias but noticeably better visual quality than unbiased L_2 reconstruction, outlier pixels frequently persist.

Here we propose a technique to reduce these artifacts and further increase image quality by leveraging auxiliary image space information such as per pixel normals, albedo, or world space position. Our main contribution is to exploit these auxiliary image features to extend and regularize screened Poisson reconstruction. The basic idea of screened Poisson reconstruction is to find an output image whose pixel values and finite difference gradients are similar to the corresponding noisy, sampled data that we acquired during gradient-domain Monte Carlo rendering. Intuitively, outlier pixels in the reconstructed output appear if both the sampled pixel value and its surrounding gradients contain outliers that happen to be in rough agreement. Our regularization avoids these outliers by adding constraints based on the features. These constraints encourage each small patch in the reconstructed output to be similar to a weighted average of the corresponding feature patches. As shown in Figure 1, our approach leads to much cleaner results with significantly lower numerical error.

We present an error analysis in terms of bias and variance that allows us to study the influence of the parameters of our approach on the output error, and choose robust parameters in practice. We also describe a GPU implementation of our extended reconstruction technique that reduces the overhead of our method to a few seconds on megapixel images. Our results show a significant visual and numerical improvement over standard screened Poisson reconstruction, and we show that our technique is consistent and converges to the ground truth solution with increased sample counts. In summary, we make the following contributions:

- An extended screened Poisson reconstruction approach for gradient-domain rendering that leverages feature patches to regularize the solution.
- An error analysis that reveals the influence of the main parameters of our method on the bias and variance in the output.
- An efficient GPU implementation that runs in a matter of seconds on megapixel images.

2. Related Work

We discuss previous work in gradient-domain rendering and image space denoising for Monte Carlo rendering.

Gradient-Domain Rendering. The core idea in gradient-domain rendering is to sample finite difference image gradients in addition to the usual pixel values. A gradient sample is simply the difference between the contribution of two light paths that go through the neighboring pixels. By generating the two paths in a correlated fashion such that they are as similar as possible, the magnitude of their differences tend to become much smaller than their individual contributions. This leads to less noise in the sampled gradients compared to the conventionally sampled pixels, and screened Poisson reconstruction yields output images with a higher quality at equal computation time compared to conventional rendering. Gradient-domain rendering was proposed in pioneering work by Lehtinen et al. [LKL*13] in the context of Metropolis light transport [VG97]. Manzi et al. [MRK*14] improved the original approach by proposing more general gradient sampling techniques. They exploit feature buffers to construct adaptive gradient kernels, while we use them to regularize screened Poisson reconstruction. These Metropolis methods adapt the target distribution that is sampled by the Markov chain to include gradients, and to focus more samples in regions with high gradients. Kettunen et al. [KMA*15] demonstrated that, maybe counterintuitively, adapting the sampling distribution is not necessary to benefit from gradient sampling. They describe a gradient-domain path tracer that simply obtains four additional gradient samples for each conventional path constructed by a standard path tracer. Their approach consistently outperforms conventional path tracing in a variety of scenarios at equal render time. They also present a Fourier analysis that explains the benefits of gradient sampling and reconstruction under some simplifying assumptions. Manzi et al. [MKA*15] subsequently describe a bidirectional gradient-domain path tracer with similar benefits over the conventional approach. While screened Poisson reconstruction under the L_2 norm leads to unbiased results with all these techniques [LKL*13], they typically advocate the use of the L_1 norm, which introduces bias but improves the visual quality. Yet even L_1 reconstruction suffers from artifacts as shown in Figure 1.

Image Space Denoising for Monte Carlo Rendering. Image space filtering for Monte Carlo has a long history, but only recently these techniques attracted renewed interest in the research community and found application in movie production [Ren]. Here we focus on the most relevant and recent work in image space denoising, and we refer to the work by Zwicker et al. [ZJL*15] for a more comprehensive survey. A key idea common to the most effective techniques to date is to use auxiliary per-pixel features, such as normals, albedo, world space position, etc., to construct the denoising filter. These features are effective because they are highly correlated with the output image, but they are usually much less noisy. Bauszat et al. [BEM11] were some of the first to exploit this idea for real-time rendering, building on the guided image filter [HST10]. Dammertz et al. [DSHL10] instead perform a fast wavelet transform that considers the feature information. Shirley et al. denoise motion and defocus blur [SAC*11] by leveraging the depth buffer.

A number of approaches targeting off-line rendering exploit the features by using them to define a cross-bilateral filter [ED04]. Sen and Darabi [SD12] propose an information theoretic approach to deal with noisy features. Li et al. [LWC12] introduced a per-pixel error estimate based on *Stein's unbiased risk estimator* (SURE) [Ste81] to select the best filter from a filter bank on a per-pixel basis. Moon et al. [MJL*13] apply a non-local means filter (NL-means) [BCM05] guided by a virtual flash image. Rousselle et al. [RMZ13] combine NL-means filter weights [RKZ12] for the noisy color image with a cross-bilateral filter on the features, and they also use SURE over several candidate filters to minimize the output error. Kalantari et al. [KBS15] employ machine learning to predict the parameters of a cross-bilateral filter at each pixel based on image features. Our approach has similarities to Moon et al.'s techniques based on local weighted regression [MCY14] and linear prediction [MIGYM15]. While Moon et al. compute local linear approximations separately, we use local linear models as regularization terms in a global energy minimization setup.

Sparse Reconstruction. Our approach is also inspired by image denoising using sparse representations [AEB06], where the idea is to express the desired output as a weighted sum of prototype signal-atoms selected from an overcomplete dictionary. Signal reconstruction is performed by finding its sparsest representation, that is, a linear combination of signal-atoms consisting of the fewest possible elements. Sparsity has been shown to effectively regularize many problems including image denoising, image superresolution, deconvolution, and many more. In our context, local patches taken from the feature images around each pixel seem to be a natural choice as the signal-atoms, because they contain most of the image structure. Then it is straightforward to extend the screened Poisson equation by expressing the image as a weighted sum of the atoms, and solve for the sparsest representation of the image that satisfies the constraints on the output pixel values and the gradients. In our experience, however, this approach did not lead to satisfactory results. Because our images are often contaminated by sparse outliers, imposing sparsity on the reconstruction from the dictionary is not effective at eliminating the rare outliers. In addition, sparse reconstruction of mega-pixel images tends to be expensive even with fast solvers [vdBF08]. Instead, we require that the reconstructed image, locally over each image patch, should be similar to a weighted sum of a small number of predetermined basis functions. These local constraints are overdetermined, since there are more pixels in each local patch than predetermined basis functions. We assemble the constraints in a global system that leads to an energy minimization problem that can be solved with the same simple methods as the original screened Poisson equation.

3. Background

The key idea in gradient-domain rendering is to sample finite difference gradients between horizontal and vertical neighbor pixels, in addition to pixel values. The final image is then reconstructed by solving a screened Poisson equation. Gradient-domain rendering is beneficial because sampled gradients typically contain less noise than sampled pixels. The noise level of high frequencies in the reconstructed image is then determined by the noise in the gradients, not the noise in the pixels [LKL*13, KMA*15]. While our results in this paper build on the gradient-domain extensions of (bidirectional) path tracing [KMA*15, MKA*15], our approach is generic and applicable to all existing gradient-domain rendering techniques [LKL*13, MRK*14].

Let us denote the conventional image sampled by a gradient-domain rendering algorithm, also called the base image, by I^g and the horizontal and vertical finite difference images by I^{dx} and I^{dy}. Screened Poisson reconstruction solves for an image I that is most consistent with both the sampled image I^g and the sampled gradients I^{dx} and I^{dy},

$$I = \arg\min_{\bar{I}} \left\|\alpha(\bar{I} - I^g)\right\| + \left\| \begin{pmatrix} H^{dx}\bar{I} \\ H^{dy}\bar{I} \end{pmatrix} - \begin{pmatrix} I^{dx} \\ I^{dy} \end{pmatrix} \right\|, \quad (1)$$

where α weights the relative influence of the pixel and gradient constraints, and H^{dx} and H^{dy} denote the horizontal and vertical finite difference operators. Solving this equation under the L_2 norm leads to unbiased reconstructions, but is susceptible to visual artifacts. In practice, the L_1 norm leads to more pleasing results, at the cost of introducing bias. Even with the L_1 norm, however, artifacts occur frequently if the input pixels and gradients are too noisy.

4. Regularized Reconstruction

The goal of our technique is to regularize screened Poisson reconstruction from Equation (1) to better suppress artifacts even at high noise levels in the input. The key idea is to leverage feature images that contain per pixel normals, albedo, position, etc. Intuitively, we add regularization constraints that push each local patch in the reconstruction to be similar to a weighted sum of corresponding patches in the feature images. Since the feature patches are largely free of noise and outliers, this leads to much cleaner outputs. We illustrate our feature patch constraints in Figure 2.

At each pixel p, let us denote its patch neighborhood by $\mathcal{N}_p$. The neighborhood consists of $s = (2r+1)^2$ pixels, where we call r the patch radius. Given the feature images, we first construct orthogonal bases for the local feature patches using truncated SVD, as described below. Assuming we have m feature bases, we unroll them and compile them into a matrix $B_p \in \mathbb{R}^{s\times m}$. In addition, the matrix $\Gamma_p \in \mathbb{R}^{s\times n}$ selects the s pixels in the local neighborhood $\mathcal{N}_p$ from the desired output image $\bar{I}$, which has n pixels. Our patch constraint says that the patch of the desired output image, $\Gamma_p\bar{I}$, should be as similar as possible to the patch itself projected onto the orthogonal basis of the feature patch (multiplication with B_p^T), followed by back-projection (multiplication with B_p). That is,

$$\left\|D_p \cdot (B_p B_p^T - \mathrm{Id}_s)\Gamma_p \bar{I}\right\| = \|P_p \bar{I}\|, \quad (2)$$

should be as small as possible. Here, Id_s is the $s\times s$ identity matrix, $D_p \in \mathrm{R}^{s\times s}$ is an additional weighting matrix, explained in detail below. We summarize the constraint using an $s\times n$ matrix P_p.

We can stack all constraint matrices $P_p, p = 1,\ldots,n$ into a matrix $M \in \mathbb{R}^{sn\times n}$, and add these constraints to the original screened Poisson problem to obtain

$$I = \arg\min_{\bar{I}} \left\|\alpha(\bar{I} - I^g)\right\| + \left\| \begin{pmatrix} H^{dx}\bar{I} \\ H^{dy}\bar{I} \end{pmatrix} - \begin{pmatrix} I^{dx} \\ I^{dy} \end{pmatrix} \right\| + \|\beta M \bar{I}\|, \quad (3)$$

where we introduced the scalar weight β to control the influence of the patch constraints. Under the L_2 norm, we obtain the minimum energy by solving the system of linear equations

$$\left(\beta^2 M^T M + H^T H + \alpha^2 \mathrm{Id}_n\right) I = \alpha I^g + H^{dx^T} I^{dx} + H^{dy^T} I^{dy} \quad (4)$$

for the desired output image I. For simplicity, we introduced the notation $H^T H = H^{dx^T} H^{dx} + H^{dy^T} H^{dy}$, which represents the discrete Laplacian opertor.

Feature Bases via Truncated SVD. Assume that we have l input feature channels, where each element of vector valued features (such as per pixel normals) is considered its own channel. At each pixel p, we can unroll the channels into a $s\times l$ matrix C_p. We use a singular value decomposition (SVD) of C_p to obtain our matrix B_p that contains the orthogonal feature basis vectors. The SVD factors the matrix $C_p \in \mathbb{R}^{s\times l}$ into a product of the form USV^T, where $U \in \mathbb{R}^{s\times s}$ and $V \in \mathbb{R}^{l\times l}$ are orthogonal matrices and $S \in \mathbb{R}^{s\times l}$ is

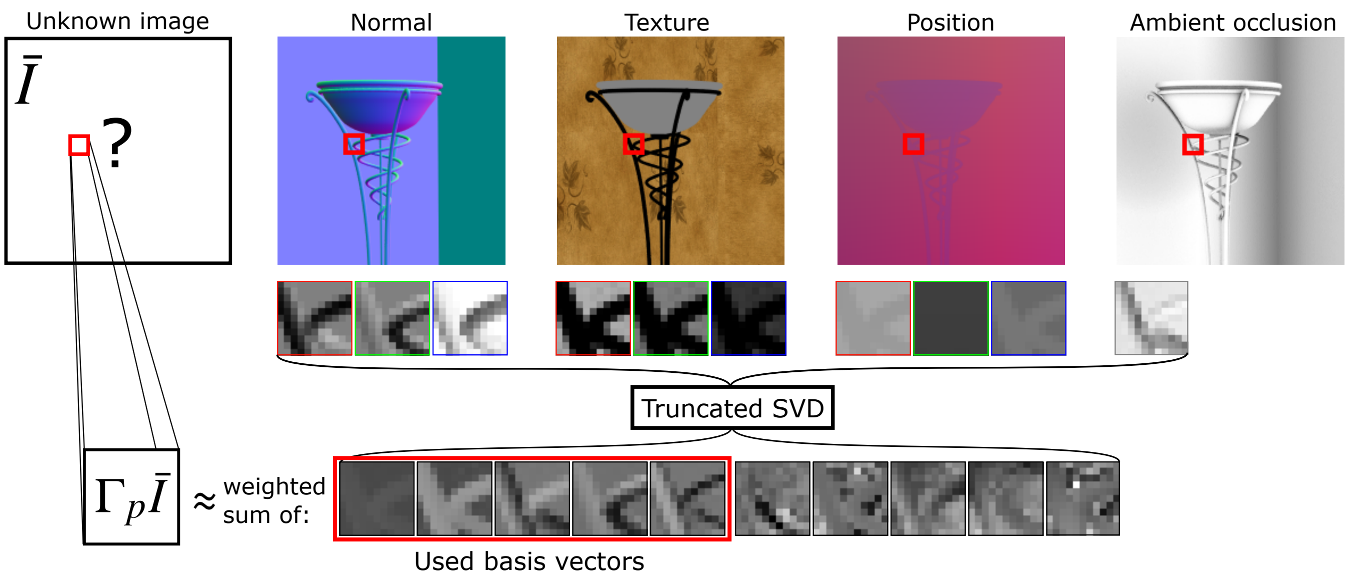


Figure 2: *Construction of the feature patch constraints. We try to reconstruct an image $\bar{I}$ where each patch $\Gamma_p \bar{I}$ is a weighted sum of the features in that patch. We orthogonalize the features by using SVD and truncate basis vectors of low singular values to avoid fitting to residual noise in the features. For better visualization we omit the weighting matrix D_p and show a larger patch size than used in practice.*

a rectangular diagonal matrix containing the singular values. The first s columns of U then form a basis of the range of C_p.

Although the features are less noisy than the sampled image and gradients, residual noise in the features will negatively impact the effectiveness of our regularization technique. Hence, we use a *truncated SVD* to remove noise from the feature subspace similar to Moon et al. [MCY14]. The vectors in U are ordered by the magnitude of the corresponding singular value. The smaller the singular value, the more noise is captured in the corresponding singular vector. This is demonstrated on an example in Figure 2. By discarding singular vectors with small singular values, one can remove noise from the subspace. We follow the approach by Moon et al. [MCY14] and discard basis vectors with a singular value that is below a pixel-wise threshold τ_p. The threshold is defined as $\tau_p = c\|E_p\|_2$, where $\|E_p\|_2$ is the spectral norm of E_p and c is a constant that we set to 0.1 in all our experiments. The entries in E_p are the square-roots of the pixel-wise variances of the features in the patch $\mathcal{N}_p$. We approximate these variances using a two-buffer approach, as described below, and approximate the spectral norm of E_p with the computationally cheaper Frobenius norm.

Per-Pixel Weights. The patch constraints can lead to inaccurate results if some pixels cannot be predicted well by the patch basis vectors. For example, a patch may contain pixels belonging to different geometric objects that may be lit differently, such that a single linear fit using the patch basis vectors cannot fit all pixels well. Each row of $(B_p B_p^T - I_s)\Gamma_p \bar{I}$ in Equation (2) measures the difference between a patch and its projection onto the feature subspace in one pixel for each color channel. The purpose of the diagonal weighting matrix D_p is to downweight the error of pixels that we consider too different to fit our model. We calculate the weights similarly to Rousselle et al. [RMZ13] by considering differences in features and color to the central pixel in the patch, and taking the minimum over all color and feature channels as our final weight.

We compute the color weights similar as in NL-means filtering [BCM05, RKZ12], that is, by averaging color differences over small neighborhoods of pixels. We derive the weights from the solution of the original Poisson problem (using the L_1 norm), because this is less noisy than the sampled pixels in I^g. We also normalize the color differences using their estimated variance, and we obtain the variance of the L_1 reconstruction using a two-buffer approach. During rendering we accumulate each half of the samples in two separate buffers, both for the base image and the gradients. We then solve the L_1 reconstruction twice, and we estimate the variance of each pixel based on the difference between the two reconstructed buffers. We blur the resulting variance with a Gaussian filter with standard deviation of two pixels, and take the maximum of the blurred two-buffer variance and the raw two-buffer variance. This reduces noise in the estimate, and it prevents the estimate from being too small when dealing with strong outliers. Concretely, the normalized color difference between two pixels p and q is

$$\Delta_i^2(p,q) = \frac{(u_i(p) - u_i(q))^2 - (\mathrm{Var}_i[p] + \mathrm{Var}_i[q])}{\varepsilon + k_c^2(\mathrm{Var}_i[p] + \mathrm{Var}_i[q])} ,$$

where p denotes the center pixel of the patch and q a neighboring pixel, $u_i, i \in \{1,2,3\}$ indexes a color channel, and $\varepsilon = 1e-10$ is a constant and k_c a user parameter. We next compute NL-means distances by averaging the color differences $\Delta_i^2(p,q)$ over small neighborhoods around p and q,

$$d_c^2(P(p),P(q)) = \frac{1}{3(2t_c+1)^2} \sum_{i=1}^{3} \sum_{n \in P(0)} \Delta_i^2(p+n, q+n) ,$$

where we sum over the color channels i of all pixels in the neighborhood represented by a set $P(0)$ of pixel offsets. The size of the neighborhood is given by the radius t_c. Finally, the color weights are derived from the NL-means distances as

$$w_c(p,q) = \exp(-\max(d_c^2(P(p),P(q)),0)). \quad (5)$$

Similar as for the color differences, we measure differences between pixels p and q of a feature channel f_j as

$$\Phi_j^2(p,q) = \frac{(f_j(p)-f_j(q))^2 - (\mathrm{Var}_j[p]+\mathrm{Var}_j[q])}{\varepsilon + k_f^2(V_j[p]+V_j[q]))},$$

where we use the same two-buffer estimate for the feature variance Var_j. In the denominator, we threshold the feature variance using $V_j[p] = \max(10^{-3}, \mathrm{Var}_j[p], \|\nabla_j[p]\|^2)$, where ∇_j denotes the finite difference gradient, to avoid being too strict with feature differences in non-smooth regions. Again, we set $\varepsilon = 1e-10$, and k_f is a user parameter. We normalize all features to have values in $[0,1]$. For vector-valued features, such as the normals, we view each vector element as one individual feature. We compute NL-means distances $d_{f_j}^2(p,q)$ equivalently to $d_c^2(p,q)$, using a radius t_f instead of t_c. Finally, the feature weights are

$$w_f(p,q) = \min_{j\in\{1,\dots,l\}} \exp(-\max(d_{f_j}^2(p,q),0)), \quad (6)$$

where l is the number of feature channels. From the color and feature weights we obtain our final weights

$$w(p,q) = \min(w_c(p,q), w_f(p,q)). \quad (7)$$

Finally, the elements of our diagonal weighting matrix are $D_p(q,q) = w(p,q)$.

It is important to realize that weighting the patch constraints does not change our patch basis vectors. Even if we downweight the patch constraint error for a certain pixel to near zero, that pixel is still included in our patch basis, and it will influence the projection of all other pixels in the patch onto the basis. We address this issue by completely removing pixels with very small weights below a threshold of $1e-10$ from the patch, that is, by removing these pixels from all feature vectors. This leads to patch constraints corresponding to arbitrarily shaped patches. We orthogonalize the modified feature vectors using SVD as described above.

5. Error Analysis

We perform an error analysis to better understand the behavior of our extended Poisson reconstruction approach. We express mean squared error (MSE) as the sum of squared bias and variance, and investigate the source of bias and variance separately. Let us write our noisy sampled image as $I^g = I_{\mathrm{ref}} + \Sigma^g$, where I_{ref} is the ground truth image, and Σ^g is a vector of i.i.d, zero-mean normally distributed random variables with variance σ^2. Similarly, we write the sampled gradients as $I^{dx} = H^{dx}I_{\mathrm{ref}} + \gamma\Sigma^{dx}$ and $I^{dy} = H^{dy}I_{\mathrm{ref}} + \gamma\Sigma^{dy}$. We assume both horizontal and vertical gradients have the noise variance $\gamma^2\sigma^2$, which is related to the variance of the pixels by a factor of γ^2. We also assume that the constraint matrix M is not influenced by noise in the input, and that we minimize the error under the L_2 norm. We then express our reconstruction I as the sum of the ground truth image I_{ref} and a per-pixel reconstruction error ε, $I = I_{\mathrm{ref}} + \varepsilon$. Substituting this into Equation (4) yields

$$\begin{aligned}&\left(\beta^2 M^T M + H^T H + \alpha^2 \mathrm{Id}_n\right)(I_{\mathrm{ref}}+\varepsilon)\\ &= \alpha(I_{\mathrm{ref}}+\Sigma^g) + H^{dx^T}(H^{dx}I_{\mathrm{ref}}+\gamma\Sigma^{dx}) + H^{dy^T}(H^{dy}I_{\mathrm{ref}}+\gamma\Sigma^{dy}).\end{aligned}$$

Taking into account that the ground truth image I_{ref} satisfies the constraints of the conventional screened Poisson equations, that is,

$$\left(H^T H + \alpha^2 \mathrm{Id}_n\right) I_{\mathrm{ref}} = \alpha I_{\mathrm{ref}} + H^{dx^T} H^{dx} I_{\mathrm{ref}} + H^{dy^T} H^{dy} I_{\mathrm{ref}}, \quad (8)$$

and after some algebraic reformulation, we obtain an expression for the error

$$\varepsilon = A^{-1}\left(\alpha\Sigma^g + H^{dx^T}\gamma\Sigma^{dx} + H^{dy^T}\gamma\Sigma^{dy} - \beta^2 M^T M I_{\mathrm{ref}}\right), \quad (9)$$

where, for simplicity, we introduced the shorthand notation

$$A = \left(\beta^2 M^T M + H^T H + \alpha^2 \mathrm{Id}_n\right). \quad (10)$$

Bias. We now obtain the expected error, that is the bias, of our reconstruction as

$$E[\varepsilon] = -A^{-1}\beta^2 M^T M I_{\mathrm{ref}}, \quad (11)$$

where we exploited that the noise Σ^g, Σ^{dx}, and Σ^{dy} is zero-mean. Clearly, bias vanishes if $\beta = 0$, or the ground truth image fully satisfies the patch constraints, that is $M^T M I_{\mathrm{ref}} = 0$. Note that $E[\varepsilon]$ expresses the per-pixel bias, and we can obtain the squared bias of an image (or image region) as $E[\varepsilon]^T E[\varepsilon]$.

Variance. To compute variance we start by formulating $\varepsilon - E[\varepsilon]$ by combining Equations (9) and (11),

$$\varepsilon - E[\varepsilon] = A^{-1}\left(\alpha\Sigma^g + H^{dx^T}\gamma\Sigma^{dx} + H^{dy^T}\gamma\Sigma^{dy}\right). \quad (12)$$

The variance over an image (or image region) is $E[(\varepsilon - E[\varepsilon])^T(\varepsilon - E[\varepsilon])]$, and we find that

$$\begin{aligned}&E[(\varepsilon-E[\varepsilon])^T(\varepsilon-E[\varepsilon])] = \sigma^2\left(\alpha^2\,\mathrm{tr}(A^{-1^T}A^{-1})\right.\\ &\left.+\gamma^2\left(\mathrm{tr}(H^{dx^T}A^{-1^T}A^{-1}H^{dx}) + \mathrm{tr}(H^{dy^T}A^{-1^T}A^{-1}H^{dy})\right)\right), \quad (13)\end{aligned}$$

where tr is the matrix trace. We observe that variance goes to zero for large values of β, because A^{-1} tends to be dominated by a factor $1/\beta^2$ (see Equation (10)). Using a spectral analysis, one can also show that for $\beta = 0$ variance is minimized at $\alpha^2 = \gamma^2$, which corresponds to the result from previous work [KMA*15].

Discussion. In Figure 3, we visualize squared bias, variance, and their sum (that is, MSE) for a small image region over a range of the α, β, and k_c parameters of our method. On the left we show the image region (top), and the bias over this region depending on our parameters. Parameter α varies along the vertical, and β over the horizontal axis. In addition, for each α we plot a range of k_c values along the vertical axis. We show variance and MSE (middle and right) for two noise levels, assuming Gaussian noise for pixels and gradients, where the variance of the gradients is $\gamma^2 = 0.2$ times the variance of the pixels. In the top row, the MSE is dominated by the bias, and in the bottom row by the variance. Our main observation is that the lowest variance for any β, and the lowest MSE, are both

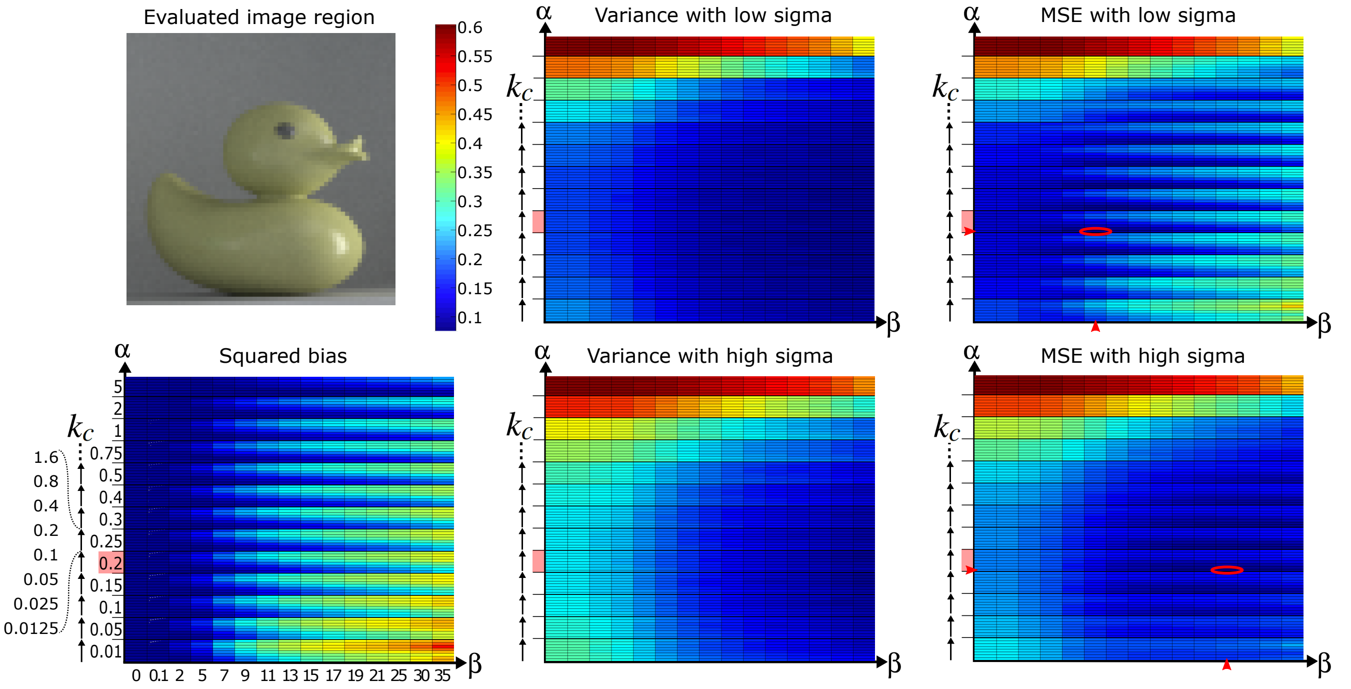


Figure 3: *We plot the logarithm of the bias, variance and mean squared error of a* 64×64 *pixels region of Bathroom according to Equation 11 and 13 for different* α, β *and* k_c. *We show* β *on the horizontal axis , and the unrolled parameters* α *and* k_c *on the vertical axis. We plot variance and MSE for two variance levels* $\sigma = 0.01$ *(low) and* $\sigma = 0.1$ *(high). We encircled the parameters with the minimal MSE in red. The optimal* α *and* k_c *are not affected by the noise level, whereas the optimal* β *gets shifted to the right with increasing noise.*

achieved at approximately $\alpha^2 \approx \gamma^2$. Parameter β provides a bias-variance tradeoff, where the value of β to minimize MSE depends on the noise level and on k_c.

6. Conjugate Gradient Solver

In practice, including the L_1 norm to solve Equation (3) provides superior results over pure L_2 minimization. We use an iteratively reweighted least squares (IRLS) approach, with the important extension to provide an estimate of the residual variance of the solution. This enables further post-processing based on the residual variance to improve the visual quality of the output, as described below, and to perform error estimation and reconstruction scale selection (Section 7). Solving Equation (3) is equivalent to obtaining

$$I = \arg\min_{\bar{I}} \left\| \underbrace{\begin{pmatrix} \alpha I_n \\ H^{dx} \\ H^{dy} \\ \beta M \end{pmatrix}}_{A} \bar{I} - \underbrace{\begin{pmatrix} \alpha I^b \\ I^{dx} \\ I^{dy} \\ 0 \end{pmatrix}}_{b} \right\|. \tag{14}$$

We estimate the variance of our solution by expressing the right hand side b as the sum of two image buffers $\hat{b}$ and $\check{b}$, where each contains one half of the samples that we rendered. We solve separately for both right hand sides, obtaining two solutions $\hat{I}$ and $\check{I}$. Because our solutions are linear in the two buffers $\hat{b}$ and $\check{b}$, we have $I = (\hat{I} + \check{I})/2$, and we can estimate the residual variance of I as $(\hat{I} - \check{I})^2/4$.

The matrix $A^T A$ is symmetric and positive-definite, hence we apply the conjugate gradient (CG) method. We summarize our modified CG algorithm in Figure 4. The main idea is to solve for $\hat{b}$ and $\check{b}$ simultaneously, while computing the IRLS weights based on the desired final solution for $b = (\hat{b} + \check{b})/2$. We use a fixed number of five reweighting and 500 CG steps. The IRLS weights are stored in the diagonal W matrix. We use a standard reweighting scheme (Line 14). We normalize the weights such that they average to one by multiplying W with the scalar $A.rows/\mathrm{tr}(W)$, where $A.rows$ is the number of rows in A and $\mathrm{tr}(W)$ the trace of W (Line 16). It is important, however, to note that we take the L_2 norm for the term $\|\alpha(\bar{I} - I^g)\|^2$ from Equation (3), that is, we do not reweight the corresponding elements of our system in Line 15. We found that using the L_1 norm for this term was not necessary thanks to our patch constraints, and the advantage of the L_2 norm is that we avoid the loss of image brightness due to outlier removal under the L_1 norm. Although we use a combination of L_1 and L_2 norm in practice, as opposed to the L_2 norm in our analysis in Section 5, we empirically observe similar behavior of bias and variance as summarized in Figure 3. In particular, we retain the ability to suppress a lot of noise by introducing only a little bias as shown in Figure 3.

GPU Implementation. We implemented our solver in CUDA. The regular structures of our sparse matrices allow us to calculate indices of non-zero values directly, instead of reading them from index arrays. As the algorithm is limited by memory bandwidth, this directly translates into performance gains. We also use shared

```
POISSONREGULARIZED($A, \hat{b}, \check{b}, x_0$)
1   for $k = 1$ to 5
2       $\hat{r}_0 = A^T W^2 \hat{b} - A^T W^2 A x_0$
3       $\check{r}_0 = A^T W^2 \check{b} - A^T W^2 A x_0$
4       $\hat{p}_0 = \hat{r}_0$; $\check{p}_0 = \check{r}_0$
5       for $i = 1$ to 500
6           $\hat{q} = A^T W^2 A \hat{p}_i$; $\check{q} = A^T W^2 A \check{p}_i$
7           $\hat{\alpha} = (\hat{r}_i^T \hat{r}_i)/(\hat{p}_i^T \hat{q})$; $\check{\alpha} = (\check{r}_i^T \check{r}_i)/(\check{p}_i^T \check{q})$
8           $\hat{x}_{i+1} = \hat{x}_i + \hat{\alpha}\hat{p}_i$; $\check{x}_{i+1} = \check{x}_i + \check{\alpha}\check{p}_i$
9           $\hat{r}_{i+1} = \hat{r}_i - \hat{\alpha}\hat{q}$; $\check{r}_{i+1} = \check{r}_i - \check{\alpha}\check{q}$
10          $\hat{\beta} = (\hat{r}_{i+1}^T \hat{r}_{i+1})/(\hat{r}_i^T \hat{r}_i)$; $\check{\beta} = (\check{r}_{i+1}^T \check{r}_{i+1})/(\check{r}_i^T \check{r}_i)$
11          $\hat{p}_{i+1} = \hat{r}_{i+1} + \hat{\beta}\hat{p}_i$; $\check{p}_{i+1} = \check{r}_{i+1} + \check{\beta}\check{p}_i$
12      $e = A(\hat{x}_{i+1} + \check{x}_{i+1}) - (\hat{b} + \check{b})/2$
13      for $j = 1$ to $A.rows$
14          $W_{jj} = 1/(||e_j||_2 + 0.05 \times 0.5^{k-1})$
15      $W = \text{setRowsToOne}(W, 1, x_0.rows)$
16      $W = W \cdot A.rows/\text{tr}(W)$
17  return $[\hat{I} = \hat{x}_{i+1}, \check{I} = \check{x}_{i+1}]$
```

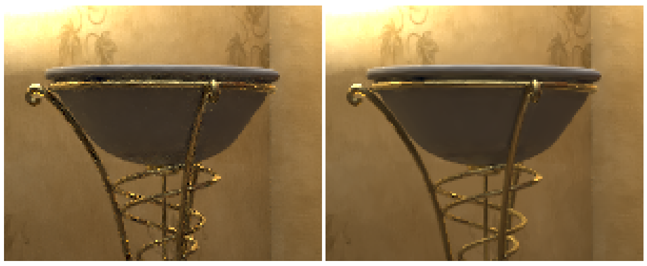
Figure 4: *Pseudocode of our modified IRLS CG solver. We compute the solution for two image buffers containing half the samples separately, which allows us to estimate the variance of the solution. Crucially, we compute the IRLS weights based on the residual of the average (Line 12), such that the average of our solution is equivalent to the solution of the average of the two buffers.*

memory and coalesced memory access patterns to further improve performance.

The most expensive step in the algorithm is the computation of the sparse matrix-vector product in line 6, where performance is limited by memory access to load non-zero matrix elements. We found that for our constraint matrix A it is beneficial to precompute the matrix product $A^T W^2 A$. For a feature patch radius of $r = 2$, the product has about seven times fewer non-zero elements than the matrix A itself. Precomputing $A^T W^2 A$ reduces the time required to read matrix elements for the computation of the matrix-vector product, and the corresponding performance increase for the matrix-vector product outweighs the cost of the precomputation.

We use a fixed number of conjugate gradient iterations, but it would also be possible to return earlier from the algorithm if it already has converged sufficiently. The convergence speed of the conjugate gradient algorithm depends on the root of the condition number of the matrix $A^T W^2 A$. Using our patch constraints, the condition number can get significantly larger than in the original Poisson problem and the algorithm can suffer from slow convergence. We thus use a default of 500 conjugate gradient iterations, while the original Poisson solver only uses 50 iterations. We could resort to the preconditioned conjugate gradient method, but finding a good preconditioner can be challenging and is left for furture work.

Post-Processing. The CG solver returns two images $\hat{I}$ and $\check{I}$, and we estimate the variance in the reconstructed image $I = (\hat{I} + \check{I})/2$ as $(\hat{I} - \check{I})^2/4$. We leverage this estimate in a post-processing step that is targeted at further reducing variance and artifacts. We achieve this by filtering I with a cross NL-means filter using weights as



Before post-processing (relative MSE: 0.0176)

After post-processing (relative MSE: 0.0083)

Figure 5: *This figure shows the effect post-processing step on Bookshelf using G-BDPT with 32 samples per pixel and the third reconstruction scale (Section 7).*

described in Section 4, Equation (7). We provide more details in Section 7. Figure 5 illustrates the post-processing step.

7. Multiscale Reconstruction and Scale Selection

We have shown (Figure 3) that by adjusting the parameters of our approach we can minimize the reconstruction error depending on the relative amount of noise in the rendered pixels and gradients (α parameter), and depending on the levels of bias and variance (β, k_c parameters). Since these properties vary within each image, we expect to be able to reduce error by adjusting parameters locally. We employ a multiscale reconstruction and scale selection approach similar as in previous work to achieve this: we obtain three scales by running our reconstruction with different parameters, estimate the error of each scale locally, and combine them into a final image by selecting for each pixel the scale that minimizes the error.

Reconstruction Parameters. Our goal was to determine the α, β, and k_c parameters for three reconstruction scales by finding parameters that minimize the error over a set of training images at different sample counts. We found that the other reconstruction parameters are insensitive to local bias and noise levels, and we set them to constants reported in Section 8. Let us denote the set of reconstruction scales in our search by P, where scale i has parameters $P_i = [\alpha_i, \beta_i, k_{c,i}]$. We determined the combination of three scales i_1, i_2 and i_3 that yielded the smallest error of all combinations.

For simplicity we ran a brute force search where we evaluated scales with $\alpha \in \{0.05, 0.1, \ldots 0.25\}$, $\beta \in \{3, 5, \ldots, 19\}$, and $k_c \in \{0.025, 0.05, 0.1, \ldots, 0.35\}$ for a total of $|P| = 360$ scales, and $360^3 \approx 47$m combinations. For each combination we computed the reconstruction error over the training images. Since we want to select scales locally, we computed the reconstruction error for small image patches. We chose 10×10 pixels in practice. Let relMSE(i, p) be the reconstruction error of scale i measured in terms of relative MSE over a patch around pixel p with respect to a reference image. The error of a combination of three scales for the patch around p is then relMSE$(i_1, i_2, i_3, p) = \min(\text{relMSE}(i_1, p), \text{relMSE}(i_2, p), \text{relMSE}(i_3, p))$, and the total error of each combination i_1, i_2, i_3 is the sum over all local patches in all training images.

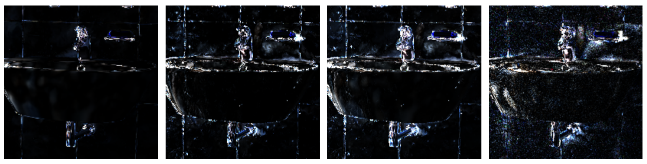


Figure 6: *We compare our error estimate, with variance and bias, to the ground truth on Bathroom (1024 samples per pixel).*

Error Estimation. Given the three scales, which we determined using the procedure described above, we estimate the MSE of each scale to enable local scale selection. We express MSE as the sum of squared bias and variance and estimate each separately.

The bias of a reconstructed scale I_i is defined as the difference of the expected value of the reconstructed image I_i and the ground truth image I_{ref}, that is, $Bias(I_i) = E[I_i] - I_{\text{ref}}$. Since the expected value of the base image I^g is the ground truth we can reformulate this to $Bias(I_i) = E[I_i - I^g]$. While $I_i - I^g$ is trivial to compute, I^g is typically very noisy, which will lead to a high variance in the bias estimate. Even worse, squaring it to get the squared bias will systematically overestimate the true bias. Hence, we reduce the noise of both I^g and I_i by filtering both images with an identical edge preserving filter. We use the color based NL-means distances $w_c(p,q)$ from Equation (5) with the parameters $k_c = 0.6$ and $f = 25$ as the filtering weights. Intuitively, this aggregates data of similar regions to get a more reliable bias estimate. For the variance estimation we use the squared difference between our two reconstructed buffers as explained in Section 6. Finally, after adding our variance and squared bias estimates, we filter this MSE estimate with a small 3×3 box-filter to further remove noise.

Scale Selection. We obtain our final image I by interpolating the values of the three scales in each pixel with weights that are inversely proportional to the estimated errors,

$$I(p) = \sum_{i=1}^{3} \frac{1/(\text{estMSE}(I_i(p)) + \varepsilon)}{\sum_j 1/(\text{estMSE}(I_j(p)) + \varepsilon)} I_i(p), \tag{15}$$

where $\text{estMSE}(I_i(p))$ is our error estimate of scale i at pixel p, $I_i(p)$ is the reconstructed value of scale i at pixel p, and $\varepsilon = 10e-10$ prevents divisions by zero. Figure 7 compares a selection based on our error estimate and a selection based on the true error. Our error estimation and scale selection effectively reduces the error of the final image below the error of any of the individual scales.

Final Algorithm. Figure 8 shows the complete reconstruction algorithm. The inputs are the rendering outputs distributed into two separate buffers, including the base image $\hat{I}^g, \check{I}^g$, the gradients $\hat{I}^{dx}, \check{I}^{dx}, \hat{I}^{dy}, \check{I}^{dy}$, and features $\hat{F}, \check{F}$, as well as the reconstruction scale parameters α_i, β_i and $k_{c,i}$, $i \in \{1,2,3\}$. The features are interpreted as images with one channel per feature. Many reconstruction steps must be applied on both buffers ($\hat{\cdot}$ and $\check{\cdot}$) separately. We represent steps that are applied on both buffers separately with the star subscript ($\cdot_\star$). We denote the average of both buffers as $\text{avg}(\cdot)$, and $\text{var}(\cdot)$ computes the two-buffer variance as described in Section 6.

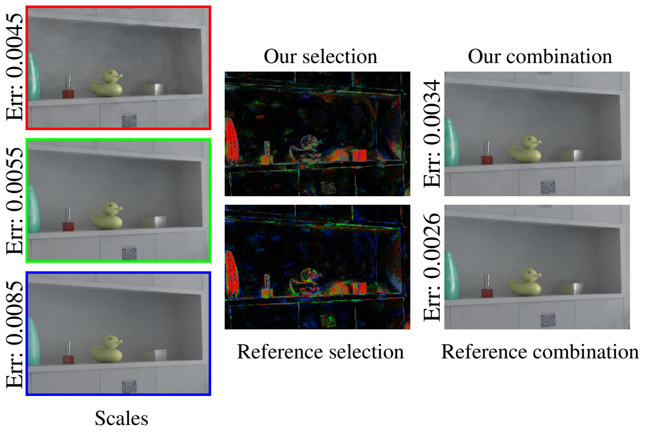


Figure 7: *We compare our scale selection to a reference selection based on ground truth data. The color of the selection maps in the middle are set according to the border colors of the scales on the left. To highlight regions where correct scale selection is crucial we modulated the brightness of the selection map per pixel by the highest error of any scale in that pixel. We successfully select the correct scales in regions where the scales differ dramatically (e.g. the highlights) and obtain an overall numerical benefit.*

Line 1 unrolls the input into a 1-d vector, used as the right hand side in Equation 14. Line 2 removes excess noise from the features with a NL-means filter, processing each channel of F separately. This is similar to the feature pre-processing described in Rousselle et al. [RMZ13], and we found it to be beneficial in addition to SVD truncation. We compute filter weights using Equations (5), (6), and (7). The function $\text{nlmFilter}(\cdot,\cdot,\cdot,\cdot,\cdot)$ takes as arguments, in this order, the image to be filtered, a guide image and its variance to compute color weights (Equation (5)), a multi-channel feature image (one channel per feature) and their variances to compute feature weights (Equation (6)). We set the parameters to $k_f = 1$, the NL-means radius to $t_f = 3$, and we ignore the color guide image. We use a filter window size of 11×11 pixels. Line 3 applies standard L_1 Poisson reconstruction with $\alpha = 0.2$ on both inputs, and the resulting images $G_\star$ are used in Line 5 to compute per-pixel weights for the patch constraints (Section 4).

We apply the reconstruction steps from Line 4 to 10 for each scale separately. Line 5 and 6 builds the constraint-matrix A of our equation system. Line 7 is the core of our algorithm where we perform the regularized reconstruction as described in Section 6. The found optimal parameters for the three scales are $\alpha_{1,2,3} = [0.25, 0.25, 0.1]$, $\beta_{1,2,3} = [5, 17, 19]$ and $k_{c,1,2,3} = [0.1, 0.35, 0.15]$. For the regularized Poisson step we further set $r = 2$, $t_c = 1$ and $t_f = 0$ and $k_f = \infty$ for all scales. Line 8 applies the post-process filter on the reconstruction to get rid of residual noise (see also Section 6). The parameters for the NL-means filter are $k_c = 0.45$, $k_f = 1$, $t_c = 1$, $t_f = 0$, and we filter over windows of 21×21 pixels. Finally, we get the reconstruction scale I_i by averaging the two reconstructed buffers (Line 9). Line 10 estimates the reconstruction error I_i as described earlier in this section. We obtain our final output by combining the three scales using Equation 15 (Line 11).

```
RECONSTRUCT($\hat{I}^g, \check{I}^g, \hat{I}^{dx}, \check{I}^{dx}, \hat{I}^{dy}, \check{I}^{dy}, \hat{F}, \check{F}, \alpha, \beta, k_c$)
1   $b_\star = \text{unroll}(I^g_\star, I^{dx}_\star, I^{dy}_\star)$
2   $F_\star = \text{nlmFilter}(F_\star, \text{null}, \text{null}, \text{avg}(F), \text{var}(F))$
3   $G_\star = \text{poisson}_{\text{L1}}(I^g_\star, I^{dx}_\star, I^{dy}_\star, 0.2)$
4   for $i = 1$ to 3
5       $M = \text{buildM}(\text{avg}(G), \text{var}(G), \text{avg}(F), \text{var}(F), k_{c,i})$
6       $A = \text{buildA}(M, \alpha_i, \beta_i)$
7       $[\hat{x}, \check{x}] = \text{poissonRegularized}(A, \hat{b}, \check{b}, \text{avg}(I^g))$
8       $x_\star = \text{nlmFilter}(x_\star, avg(x), var(x), avg(F), var(F))$
9       $I_i = \text{avg}(x)$
10      $\text{estMSE}_i = \text{estimateError}(I_i, \text{avg}(I^g), \hat{x}, \check{x})$
11  $I = \text{combineScales}(I_1, I_2, I_3, \text{estMSE}_1, \text{estMSE}_2, \text{estMSE}_3)$
12  return $I$
```

Figure 8: *Pseudocode of our complete reconstruction pipeline.*

8. Results and Discussion

Implementation and Performance. We implemented our approach on top of the gradient-domain path tracing (G-PT) [KMA*15] and gradient-domain bidirectional path tracing (G-BDPT) [MKA*15] implementations in Mitsuba [Jak10] by the original authors. We modified G-PT and G-BDPT to render the sampling data into two separate buffers and to store feature information (position, normal and texture). We also use an ambient occlusion feature that helps to preserve some shading effects. We use existing Mitsuba functionality to efficiently generate the ambient occlusion maps. On an NVidia Titan X our approach takes about one minute to reconstruct a one mega-pixel image. The pre-filtering of all our features (Line 2) takes 0.5 seconds, solving the L_1 reconstruction for both input buffers (Line 3) 0.3 seconds, building the patch constraint matrix M (Line 5) 5 seconds per scale, solving the regularized reconstruction (Line 7) 14 seconds per scale, and post processing both outputs (Line 8) takes 1.1 second per scale. The time taken for error estimation and final scale selection is negligible. Memory consumption of our implementation is rather high since we store $A^T W^2 A$ on the GPU. For a one mega-pixel image we require approximately 7.5GB of GPU memory. This could be reduced by performing the reconstruction on overlapping tiles of the image separately, at the cost of some performance.

Comparison to Previous Work. We compare the denoising performance of our algorithm in Figure 9 and 10 to gradient-domain rendering using conventional L_1 reconstruction (L1) and to *Robust Denoising with Feature and Color Information* (RDFC) [RMZ13]. We measure all errors as relative mean squared error (rel. MSE) $E = \text{mean}((I - I_{ref})^2/(I^2_{ref} + 10^{-3}))$. Figure 10 shows that our approach outperforms L_1 reconstruction by a large margin. While L_1 reconstruction suffers from isolated spikes and low frequency noise, our reconstruction yields a clean image even at low sampling counts. In our tested scenes we report a significant improvement in relative MSE of a factor of 1.5 to 5.

RDFC is representative for recent image space denoising techniques for Monte Carlo rendering, achieving state of the art performance for moderate and higher sampling rates (64 samples per pixel and higher) [KBS15]. It leverages features similar to our approach, and for a fair comparison we used the exact same features for both algorithms, including RGB texture, normal, world space position, and a scalar ambient occlusion term. We use RDFC with uniform sampling, since adaptive sampling for gradient-domain rendering has not been described in the literature yet. We compare our reconstruction on top of G-BDPT to RDFC on top of BDPT, and our reconstruction on top of G-PT to RDFC on top of PT. The authors of G-PT and G-BDPT report an overhead of their methods of roughly 2.5x for G-PT and 4x for G-BDPT compared to the base algorithms. Hence, for equal time comparisons RDFC can use 2.5x more base samples when applied on top of PT and 4x more base samples when applied on top of BDPT. Despite this, we still achieve slightly better results with our method in our tested scenes. With PT as base algorithm we show in Figure 10 that our method improves upon RDFC in Bathroom and Bookshelf by 12% and in Sponza by 50%. With BDPT as base algorithm we improve upon RDFC in Bottle by 16%. Finally, Figure 9 demonstrates the convergence of our method for the Bathroom scene using a log-log plot. We observe that our method converges to the correct solution, with errors by a factor of three to five lower than L_1 reconstruction. Independent of rendering time, our method also consistently outperforms RDFC time in this scene.

Discussion. The denoising results of our method and RDFC are quite similar despite the rather different reconstruction approaches. In regions that are well captured by features, both algorithms manage to remove variance nearly completely without introducing artifacts. Remaining error concentrates mostly at edges or around scene details that are not well captured by the features. Our method seems to benefit from the gradient information in these areas, which often manages to capture such details with less noise. One such example is the shading on the rubber duck in the Bathroom scene (Figure 10). Gradients do not guarantee to reduce noise, however, and there are also image regions where our approach has higher error than RDFC. An interesting observation is that RDFC uses very large filtering kernels of size 21×21 or even 41×41 pixels, while our method achieves comparable results using patch constraints of size 5×5 ($r = 2$). This is because our patch constraints are linked in a global system, and as we showed, the parameter β allows us to reduce variance independently of patch size.

To ensure a fair comparison to RDFC, we also tried to enhance RDFC with the gradient information directly. First, we tried to post-process the conventional screened Poisson reconstruction with RDFC, and second, we tried to prefilter both gradients and pixels with RDFC, followed by conventional screened Poisson reconstruction. In equal time comparisons, both these variants perform significantly worse than RDFC without gradients and our proposed approach.

Our approach is also orthogonal to specifics of the underlying gradient-domain rendering algorithm, as long as the renderer returns auxiliary feature data. Since variance computation relies on a two-buffer approach [RKZ12], our technique can also be used with Metropolis gradient-domain rendering [LKL*13, MRK*14], for example. This would simply require to run two Markov chains with different seeds. In summary, our approach closes the gap between gradient-domain rendering and image space denoising. A key advantage over existing image space denoising techniques is that it

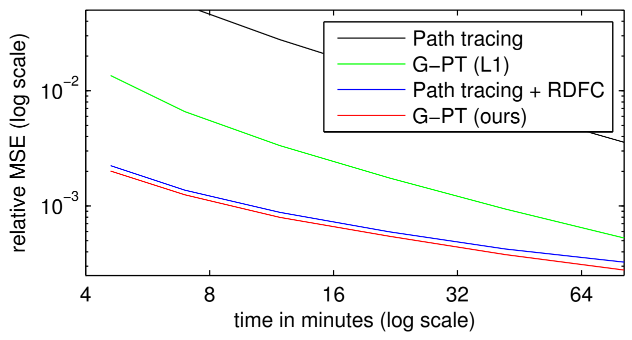


Figure 9: *Convergence for the Bathroom scene in a log-log plot. We show path-tracing without de-noising (black), path-tracing with RDFC (blue), gradient-domain path-tracing using L_1 reconstruction (green), and our reconstruction (red).*

will benefit from future improvements in gradient-domain rendering, such as improvements in gradient sampling using better shift mappings, higher order finite differences, or adaptive sampling.

9. Conclusions

We presented an improved reconstruction technique for gradient-domain rendering that leverages auxiliary image features. A GPU implementation processes one mega-pixel images in about one minute. Our approach outperforms previous solutions of the screened Poisson equation under the L_1 norm by a large margin. We also compare our approach to a state of the art image space denoising technique for Monte Carlo rendering and demonstrate, for the first time, that gradient-domain rendering can outperform conventional denoising. The margin, however, is modest and our observations raise the challenge to further improve gradient sampling. Since our approach is orthogonal to the underlying sampling distribution, there are various avenues for future work. For example, adaptive sampling based on the estimated error of the reconstructed image should be straightforward to add on top of our improved reconstruction technique. Finally, enforcing temporal consistency would also be an interesting direction for future work.

References

[AEB06] Aharon M., Elad M., Bruckstein A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Process. 54*, 11 (Nov 2006), 4311–4322. 3

[BCM05] Buades A., Coll B., Morel J.-M.: A non-local algorithm for image denoising. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2005.* (June 2005), vol. 2, pp. 60–65 vol. 2. 2, 4

[BEM11] Bauszat P., Eisemann M., Magnor M.: Guided image filtering for interactive high-quality global illumination. *Computer Graphics Forum 30*, 4 (June 2011), 1361–1368. 2

[DSHL10] Dammertz H., Sewtz D., Hanika J., Lensch H. P. A.: Edge-avoiding à-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (2010), Eurographics Association, pp. 67–75. 2

[ED04] Eisemann E., Durand F.: Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 673–678. 2

[HST10] He K., Sun J., Tang X.: Guided image filtering. In *Proceedings of the 11th European conference on Computer vision: Part I* (Berlin, Heidelberg, 2010), ECCV'10, Springer-Verlag, pp. 1–14. 2

[Jak10] Jakob W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 9

[KBS15] Kalantari N. K., Bako S., Sen P.: A machine learning approach for filtering monte carlo noise. *ACM Trans. Graph. 34*, 4 (July 2015), 122:1–122:12. 2, 9

[KFF*15] Keller A., Fascione L., Fajardo M., Georgiev I., Christensen P., Hanika J., Eisenacher C., Nichols G.: The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses* (2015), pp. 24:1–24:7. 1

[KMA*15] Kettunen M., Manzi M., Aittala M., Lehtinen J., Durand F., Zwicker M.: Gradient-domain path tracing. *ACM Trans. Graph. 34*, 4 (Aug. 2015). 1, 2, 3, 5, 9

[LKL*13] Lehtinen J., Karras T., Laine S., Aittala M., Durand F., Aila T.: Gradient-Domain Metropolis Light Transport. *ACM Trans. Graph. 32*, 4 (2013). 1, 2, 3, 9

[LWC12] Li T.-M., Wu Y.-T., Chuang Y.-Y.: Sure-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 194:1–194:9. 2

[MCY14] Moon B., Carr N., Yoon S.-E.: Adaptive rendering based on weighted local regression. *ACM Trans. Graph. 33*, 5 (Sept. 2014), 170:1–170:14. 2, 4

[MIGYM15] Moon B., Iglesias-Guitian J. A., Yoon S.-E., Mitchell K.: Adaptive rendering with linear predictions. *ACM Trans. Graph. 34*, 4 (July 2015), 121:1–121:11. 2

[MJL*13] Moon B., Jun J. Y., Lee J., Kim K., Hachisuka T., Yoon S.-E.: Robust image denoising using a virtual flash image for monte carlo ray tracing. *Computer Graphics Forum 32*, 1 (2013), 139–151. 2

[MKA*15] Manzi M., Kettunen M., Aittala M., Lehtinen J., Durand F., Zwicker M.: Gradient-domain bidirectional path tracing. In *Proc. Eurographics Symposium on Rendering* (2015). 1, 2, 3, 9

[MRK*14] Manzi M., Rousselle F., Kettunen M., Lehtinen J., Zwicker M.: Improved sampling for gradient-domain metropolis light transport. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 178:1–178:12. 2, 3, 9

[Ren] Renderman denoiser. `http://renderman.pixar.com/view/denoiser`. Accessed: 2015-08-30. 2

[RKZ12] Rousselle F., Knaus C., Zwicker M.: Adaptive rendering with non-local means filtering. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 195:1–195:11. 2, 4, 9

[RMZ13] Rousselle F., Manzi M., Zwicker M.: Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7 (2013), 121–130. 2, 4, 8, 9

[SAC*11] Shirley P., Aila T., Cohen J., Enderton E., Laine S., Luebke D., McGuire M.: A local image reconstruction algorithm for stochastic rendering. In *Prof. ACM Symposium on Interactive 3D Graphics and Games* (2011), pp. 9–14. 2

[SD12] Sen P., Darabi S.: On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph. 31*, 3 (June 2012), 18:1–18:15. 2

[Ste81] Stein C. M.: Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics 9*, 6 (1981), pp. 1135–1151. 2

[vdBF08] van den Berg E., Friedlander M. P.: Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing 31*, 2 (2008), 890–912. 3

[VG97] Veach E., Guibas L. J.: Metropolis light transport. In *Proc. ACM SIGGRAPH 97* (1997), pp. 65–76. 2

[ZJL*15] Zwicker M., Jarosz W., Lehtinen J., Moon B., Ramamoorthi R., Rousselle F., Sen P., Soler C., Yoon S.-E.: Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *Computer Graphics Forum 34*, 2 (2015), 667–681. 2

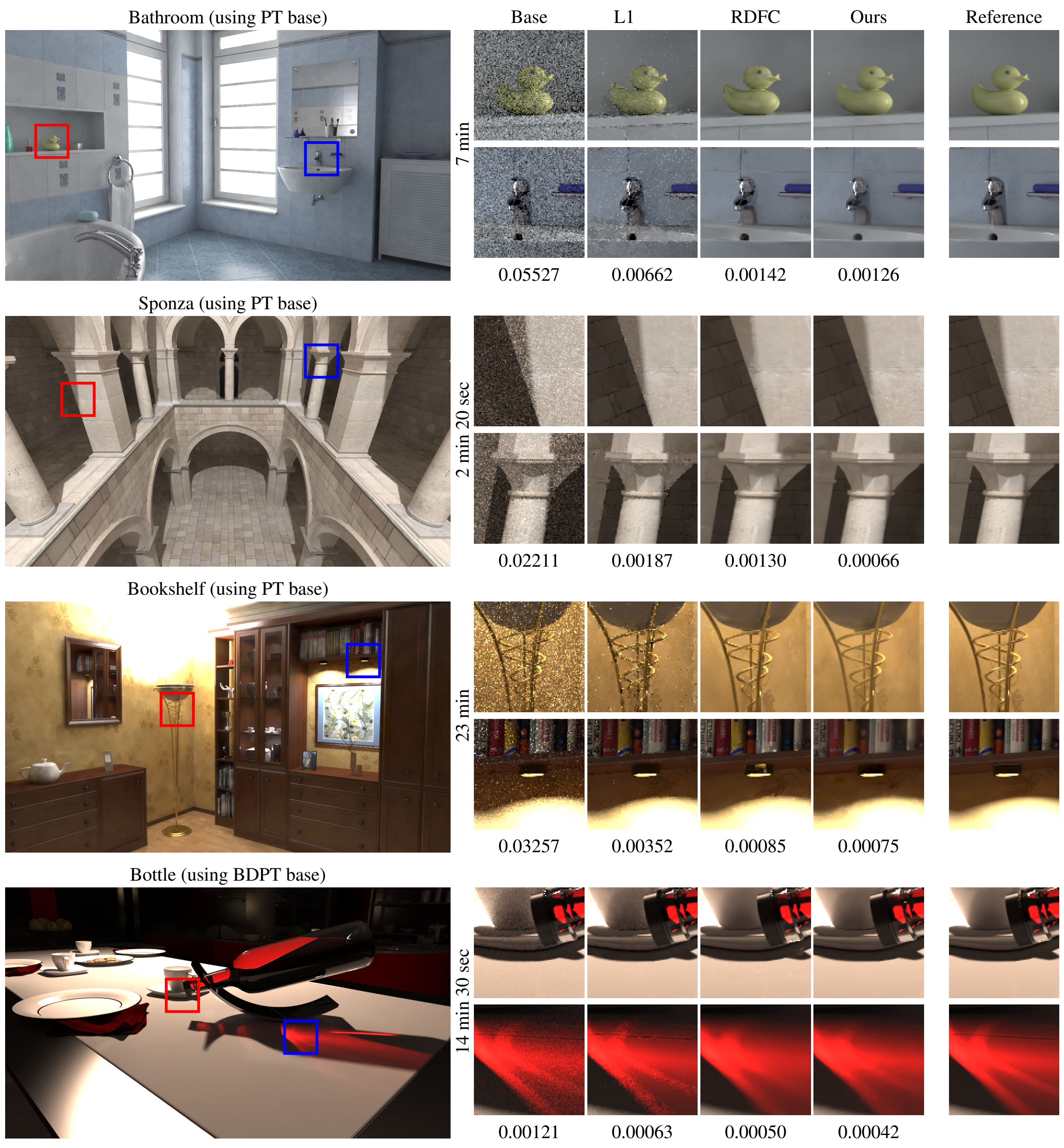


Figure 10: *We compare our new algorithm (Ours) on several scenes to gradient-domain rendering using the L_1 reconstruction (L1), to ordinary rendering using Robust Denoising (RDFC) and to the used base algorithm (Base). The base algorithm for Bathroom, Sponza and Bookshelf is unidirectional path tracing, while for Bottle it is bidirectional path tracing. The rendering times for all methods are approximatively the same (except for the reference image) and are shown on the left of the insets. Below the insets we show the relative mean squared error of each method for the entire image. The full resolution images on the left-most column show the result using our method.*

Chapter 9

Temporal Gradient-Domain Path Tracing

Published as:
Temporal Gradient-Domain Path Tracing
Marco Manzi*, Markus Kettunen*, Frédo Durand, Matthias Zwicker and Jaako Lehtinen.
* Both authors contributed equally to this publication.
The definitive version of record will be published in ACM Transactions on Graphics 35(6), http://dx.doi.org/10.1145/2980179.2980256

Temporal Gradient-Domain Path Tracing

Marco Manzi[1,⋆] Markus Kettunen[2,⋆] Frédo Durand[4] Matthias Zwicker[1] Jaakko Lehtinen[2,3]
[1]University of Bern [2]Aalto University [3]NVIDIA [4] MIT CSAIL
⋆ *Both authors contributed equally to this work.*

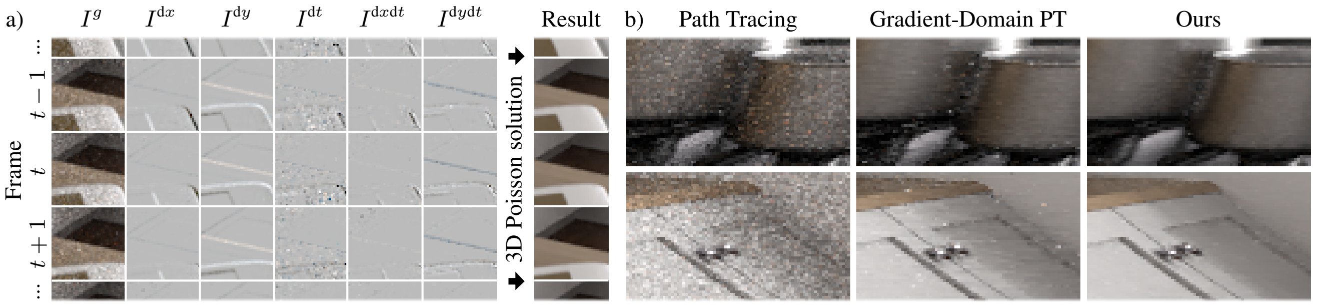


Figure 1: a) *In addition to standard path sampling, our method also estimates spatial, temporal and mixed finite differences for the frames of an animation. We then solve a 3D screened Poisson problem to reconstruct the animation whose frames best match the sampled data.* b) *Equal-time rolling shutter crops of the animation* KITCHEN 2. *The rows are extracted from sequential animation frames. Our method often reduces both spatial variance, seen as horizontal noise, and flickering, seen as vertical noise.*

Abstract

We present a novel approach to improve temporal coherence in Monte Carlo renderings of animation sequences. Unlike other approaches that exploit temporal coherence in a post-process, our technique does so already during sampling. Building on previous gradient-domain rendering techniques that sample finite differences over the image plane, we introduce temporal finite differences and formulate a corresponding 3D spatio-temporal screened Poisson reconstruction problem that is solved over windowed batches of several frames simultaneously. We further extend our approach to include second order, mixed spatio-temporal differences, an improved technique to compute temporal differences exploiting motion vectors, and adaptive sampling. Our algorithm can be built on a gradient-domain path tracer without large modifications. In particular, we do not require the ability to evaluate animation paths over multiple frames. We demonstrate that our approach effectively reduces temporal flickering in animation sequences, significantly improving the visual quality compared to both path tracing and gradient-domain rendering of individual frames.

Keywords: Monte Carlo rendering, gradient-domain rendering

Concepts: •**Computing methodologies** → *Ray tracing;*

1 Introduction

Monte Carlo path tracing is establishing itself as the algorithm of choice for movie production because of its physical realism and predictable behavior [Keller et al. 2015]. Rendering animations with hundreds of thousands of frames is still challenging, however, due to the significant computational effort it takes to reduce variance (noise) to acceptable levels. Current production pipelines employ conventional Monte Carlo rendering techniques that produce each frame separately, and apply post-process noise reduction filters. This seems wasteful, as the similarity between temporally nearby images is not exploited during the rendering process.

In this paper, we introduce a Monte Carlo technique that exploits temporal coherence during both rendering and reconstruction, as opposed to just a post-process, and yields consistent results without heuristic blending of samples across frames or other model-based reasoning. These properties contrast previous algorithms, including sample reprojection, spatio-temporal filtering, and the construction of smooth spatio-temporal function bases for the frames.

Our key idea is to sample the *differences* between corresponding pixels in temporally adjacent frames using correlated pairs of paths, and then integrate the Monte Carlo difference estimates across time. This effectively suppresses flickering, thanks to the surprising property, shown previously in gradient-domain path tracing [Kettunen et al. 2015], that correlated difference estimates followed by integration yields significant variance reduction for smooth signals.

Technically, we build on gradient-domain path tracing (GPT) [Kettunen et al. 2015], which samples spatial finite difference image gradients and reconstructs output images by solving a screened Poisson problem. We extend both the sampling and reconstruction steps into the three-dimensional spatio-temporal domain. We estimate the temporal differences using pairs of paths that share the same primary sample space coordinates in adjacent frames, further leveraging motion vectors provided by most standard path tracers to try to ensure that both paths share a similar primary hit point of the camera ray. This reduces variance in temporal differences, and hence increases quality, while remaining consistent. Finally, we introduce an adaptive sampling technique that exploits the sparsity of high variance regions in the gradient domain. Our method is lightweight and largely non-intrusive in the sense that it can be implemented relatively easily on top of an existing path tracer.

2 Related Work

Exploiting Temporal Coherence. While there is a broad literature on taking advantage of spatial coherence through filtering [Rushmeier and Ward 1994] or interpolation [Ward et al. 1988] to accelerate Monte Carlo rendering of individual images, there has been much less work on temporal coherence. Early approaches, as the one by Chen [1990] or Nimmeroff et al. [1996], relied on radiosity [Goral et al. 1984], and they focused on incrementally updating or interpolating radiosity solutions over time. Temporal coherence methods are widely studied in real-time rendering [Scherzer et al. 2012], but interactivity is outside the scope of this paper.

In the context of Monte Carlo techniques, Havran et al. [2003] first proposed to exploit temporal coherence in animations by updating and reprojecting samples to different points in time. Similarly, in final gathering in photon mapping [Tawara et al. 2004] and in irradiance caching [Smyk et al. 2005], final gather or irradiance samples can be sparsely updated and interpolated over time. An alternative strategy is to remove noise from image sequences produced by Monte Carlo rendering after the fact, which can be achieved using various image space filtering techniques. McCool [1999] proposed to use anisotropic diffusion, for example. He may have been the first to mention an extension to 3D spatio-temporal denoising for Monte Carlo rendering, although this was not demonstrated in his work. Meyer and Anderson [2006] use PCA analysis to construct a smooth basis for a sequence of images, then they project noisy images onto this basis for denoising. Recently, various image space filters have also been demonstrated in the spatio-temporal setting [Sen and Darabi 2012; Li et al. 2012; Rousselle et al. 2013; Moon et al. 2014]. Zimmer et al. [2015] further propose a path space decomposition approach and sophisticated motion estimation techniques to maximize temporal coherence.

A key difference to our work is that we take advantage of temporal coherence during both Monte Carlo sampling as well as reconstruction, instead of just filtering in a post-processing step. Furthermore, we sample temporal differences in an unbiased fashion, and our reconstruction can produce unbiased output if desired. This is not possible with any of the previous methods that use temporal coherence. In practice, though, an outlier-suppressing L_1 reconstruction is preferred over the unbiased method, like with previous gradient-domain rendering algorithms.

Gradient-domain Rendering. Gradient-domain rendering relies on sampling finite difference image gradients in addition to pixel intensities, where a gradient sample is the difference between a *base* and an *offset* light path through neighboring pixels. Offset paths are generated in a correlated fashion by shifting a base path sampled by a standard path tracer to a neighboring pixel, such that the two paths remain as similar as possible. Consequently, the magnitude of the difference in their throughputs is generally smaller than their individual contributions. Due to this reduced variance, output images after screened Poisson reconstruction are of higher quality compared to conventional rendering. Finally, gradient-domain rendering is unbiased if reconstruction employs the L_2 norm.

Lehtinen et al. [2013] originally introduced gradient-domain rendering for Metropolis light transport [Veach and Guibas 1997], and Manzi et al. [2014] proposed more general gradient sampling techniques to improve the original approach. Kettunen et al. [2015] showed that gradient sampling is also beneficial in conventional path tracing, and back their empirical results up with an analysis that studies the gradient sampling and reconstruction process through Fourier theory. Manzi et al. [2015] extend gradient-domain path tracing to bidirectional path tracing, combining the advantages of gradient and bidirectional sampling.

At its core, gradient-domain sampling estimates differences $\Delta_{i,j}$ between the intensities of neighboring pixels i and j as

$$\begin{aligned}\Delta_{i,j} &= \left(h(x) * \int_\Omega f(x,\bar{p}) - f(T_{ij}(x,\bar{p}))\left|T'_{ij}\right| \mathrm{d}\mu(\bar{p})\right)(x_i) \\ &= \left(h(x) * \int_\Omega g_{ij}(x,\bar{p})\mathrm{d}\mu(\bar{p})\right)(x_i), \qquad (1)\end{aligned}$$

where x is the image coordinate, $h(x)$ a pixel filter, Ω the space of light paths, $(x,\bar{p})$ a path with additional parameters $\bar{p}$, and f the image contribution function. We call T_{ij} a *shift mapping* that maps a base to an offset path, and $|T'| = |\partial T/\partial\bar{x}|$ is the determinant of the Jacobian of $T(\bar{x})$. Gradient-domain rendering techniques may use various methods to sample base paths $(x,\bar{p})$ in Equation 1, such as unidirectional [Kettunen et al. 2015] or bidirectional path tracing [Manzi et al. 2015]. While estimating the horizontal and vertical finite differences, they also sample the image itself in a conventional manner. Finally, finite differences are sampled in both directions, that is, for each $\Delta_{i,j}$ also $\Delta_{j,i}$ is sampled, and the two are combined using multiple importance sampling (MIS).

Depending whether i and j are horizontal or vertical neighbors, let us classify the differences $\Delta_{i,j}$ into horizontal and vertical, and unroll them into two vectors I^{dx} and I^{dy}. In addition, let I^g denote the conventional *primary image* unrolled into a vector. Screened Poisson reconstruction solves for an image I that satisfies

$$I = \underset{\bar{I}}{\operatorname{argmin}} \left\|\alpha(\bar{I} - I^g)\right\|^p + \left\|\begin{pmatrix} H^{\mathrm{dx}}\bar{I} \\ H^{\mathrm{dy}}\bar{I}\end{pmatrix} - \begin{pmatrix} I^{\mathrm{dx}} \\ I^{\mathrm{dy}}\end{pmatrix}\right\|^p, \qquad (2)$$

where α weights the influence of the pixel and gradient constraints, and H^{dx} and H^{dy} denote the horizontal and vertical finite difference operators on the 2D pixel grid. The solution image I simultaneously minimizes the pixelwise difference to the primary image, as well as the difference of its gradient to the sampled gradients. The solution of Equation 2 under the L_2 norm ($p = 2$) is unbiased, but often suffers from visual artifacts. The L_1 norm ($p = 1$) yields more pleasing results, at the cost of introducing bias.

3 Temporal GPT

Our temporal gradient-domain path tracing algorithm introduces three main conceptual components: finite differences over both space and time, a temporal shift mapping to obtain these gradients, and spatio-temporal screened Poisson reconstruction.

3.1 Space-Time Finite Differences

We sample three kinds of finite differences over space and time as illustrated in Figure 2:

1. regular image gradients over pairs of pixels in individual frames;
2. the temporal differences for a pixel between neighboring frames; and
3. second-order mixed space-time differences, that is, temporal differences between spatial gradients.

Spatial image gradients are computed precisely as in gradient-domain path tracing [Kettunen et al. 2015], and we will not review its *spatial shift mapping* here. In particular, we also evaluate spatial shifts across pixels in both forward and backward directions, and weight them using MIS. For clarity, our figures only feature the positive spatial shift. We call the shift mapping between frames the *temporal shift*, and the temporal differences consist of base-offset path pairs in temporally adjacent frames. We compute the mixed differences after-the-fact by subtracting corresponding spatial gradients between adjacent frames.

3.2 Primary Sample Space Temporal Shift Mapping

For temporal and mixed differences, we propose a temporal shift mapping that copies the primary sample space [Kelemen et al. 2002] coordinates of the base path to the next frame, just with the time coordinate incremented by one full frame. This simply means

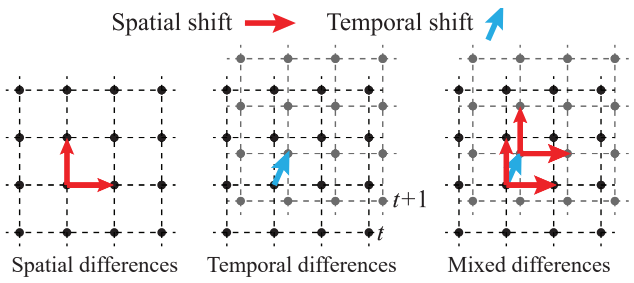


Figure 2: *Spatial, temporal, and mixed spatio-temporal finite differences over adjacent pixels in both space and time. Temporal finite differences are taken between paths in adjacent frames at times* t *and* $t+1$*. The mixed differences are second order, that is, temporal differences between spatial gradients in adjacent frames.*

the temporal offset path uses, with the exception of time, the same values for the uniform random variables that specify the base path.

The main advantage of this shift is its simplicity. The shift has unit Jacobian by construction, and as long as the renderer is deterministic, implementation is trivial. We can render each frame twice, as a *base frame* and a *temporal offset frame*. A base frame is rendered with conventional GPT as usual. The temporal offset frame is also rendered with GPT, by re-using the same random variables for each pixel as the previous base frame. Hence, subtracting a base frame from its succeeding temporal offset frame yields the temporal and mixed differences.

We illustrate this process in Figure 3. The red samples form the base frames, and the blue samples the temporal offset frames. Both base and temporal offset frames include a primary image (consisting of the samples indicated with filled circles) and spatial gradient images (consisting of the differences between the spatial offset samples and their corresponding primary samples).

The above scheme reduces the overhead of our approach by re-using computation similar to the original GPT algorithm. Specifically, for each base path (filled red circles) that contributes to the primal image in the base frame, we re-use it as the base path for our spatial gradients as in GPT. In addition, we re-use it as the base path for a temporal difference. Finally, we re-use the temporal offset paths, which we computed for the temporal differences, to again compute spatial gradients which we also use for the mixed gradients. Kettunen et al. [2015] report an overhead of $2.5\times$ of GPT over conventional path tracing. Our approach amounts to running GPT with half the samples for both the base and offset image in each frame, and hence we inherit the overhead of $2.5\times$.

A potential disadvantage of the primary sample space shift is that the path throughput function is often less smooth over primary sample space coordinates than other parameterizations, potentially leading to higher variance than the more sophisticated spatial shift that reuses path vertex positions instead. We address this by leveraging motion vectors (Section 4.2) to regain smoothness.

3.3 Spatio-temporal Reconstruction

Spatio-temporal screened Poisson reconstruction is a direct extension of the 2D case (Equation 2). In addition to spatial gradients, we now also take into account the temporal and mixed spatio-temporal differences, and instead of reconstructing a single frame $\bar{I}$, reconstruct a sequence of frames $\bar{\mathbf{I}}$ simultaneously. Denoting the sequence of sampled conventional images by $\mathbf{I}^g$, and the sequences of sampled spatial, temporal, and spatio-temporal differences by

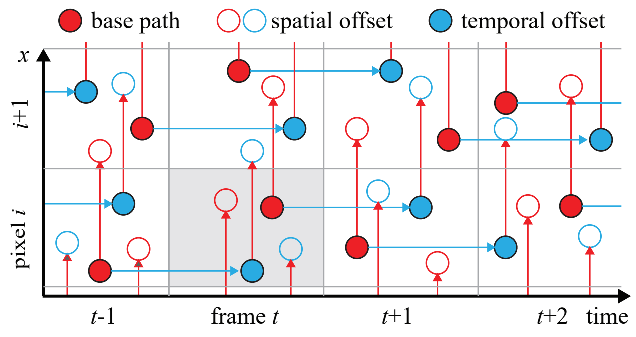


Figure 3: *We illustrate sampling of spatial, temporal, and mixed-spatio-temporal differences, using spatial (red) and temporal shifts (blue arrows). Our approach amounts to rendering each frame twice using GPT, as a base frame (red samples) and as a temporal offset frame (blue samples). Each offset frame has identical primary sample space values as its preceding base frame, which implements our temporal shift.*

$\mathbf{I}^{\mathrm{d}x}, \mathbf{I}^{\mathrm{d}y}, \mathbf{I}^{\mathrm{d}t}, \mathbf{I}^{\mathrm{d}x\mathrm{d}t}$, and $\mathbf{I}^{\mathrm{d}y\mathrm{d}t}$, we formulate spatio-temporal reconstruction as

$$\mathbf{I} = \operatorname*{argmin}_{\bar{\mathbf{I}}} \left\|\alpha(\bar{\mathbf{I}} - \mathbf{I}^g)\right\|^p + \left\| \begin{pmatrix} H^{\mathrm{d}x}\bar{\mathbf{I}} \\ H^{\mathrm{d}y}\bar{\mathbf{I}} \\ H^{\mathrm{d}t}\bar{\mathbf{I}} \\ H^{\mathrm{d}x\mathrm{d}t}\bar{\mathbf{I}} \\ H^{\mathrm{d}y\mathrm{d}t}\bar{\mathbf{I}} \end{pmatrix} - \begin{pmatrix} \mathbf{I}^{\mathrm{d}x} \\ \mathbf{I}^{\mathrm{d}y} \\ \mathbf{I}^{\mathrm{d}t} \\ \mathbf{I}^{\mathrm{d}x\mathrm{d}t} \\ \mathbf{I}^{\mathrm{d}y\mathrm{d}t} \end{pmatrix} \right\|^p, \qquad (3)$$

where $H^{\mathrm{d}x}$ and $H^{\mathrm{d}y}$ are the spatial finite difference operators on the pixel grid, $H^{\mathrm{d}t}$ is the temporal finite difference operator between adjacent frames, and $H^{\mathrm{d}x\mathrm{d}t}$ and $H^{\mathrm{d}y\mathrm{d}t}$ are the mixed spatio-temporal finite difference operators. These are simply the concatenations of the temporal and spatial operators, $H^{\mathrm{d}x\mathrm{d}t} = H^{\mathrm{d}t}H^{\mathrm{d}x}$ and $H^{\mathrm{d}y\mathrm{d}t} = H^{\mathrm{d}t}H^{\mathrm{d}y}$.

To process arbitrarily long input sequences we split them into overlapping subsequences, reconstruct each subsequence separately, and smoothly blend between overlapping temporal regions after reconstruction. Our approach is similar to the temporal extensions of screened Poisson reconstruction used by Bonneel et al. [2014; 2015]. However, they perform reconstruction in a causal manner, processing subsequent frames one-by-one over time.

We solve Equation 3 via its normal equations $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$ and a conjugate gradient solver in CUDA. The vector $\mathbf{x}$ consists of all pixels in all frames, and matrix $\mathbf{A}$ is sparse and its rows represent all the constraints in Equation 3. For efficiency our solver computes $\mathbf{A}^T\mathbf{A}$ on the fly without storing $\mathbf{A}$ explicitly.

3.4 Frequency Analysis

In analogy to the purely spatial case studied in previous work, the expected main benefit of adding temporal finite differences to the sampling and reconstruction process is the reduction of high frequency temporal variance (distracting flickering). To understand this better, we extend the frequency analysis of gradient-domain path tracing presented by Kettunen et al. [2015].

For simplicity, we assume that the mean squared errors (MSEs) of the sampled spatial, temporal, and spatio-temporal gradients are all equal, and denote them by $|\epsilon_G|^2$. Note that these directly correspond to their variances. We denote the MSE of the sampled conventional image by $|\epsilon_F|^2$. Since Monte Carlo sampling produces

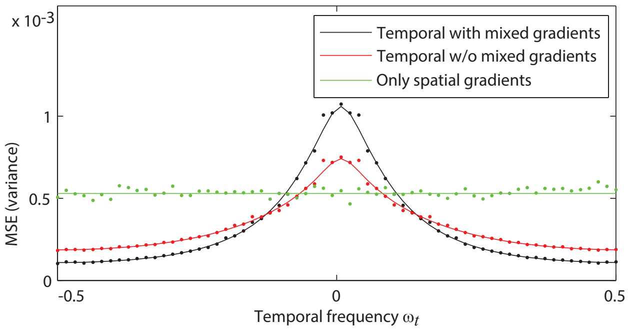


Figure 4: *The reconstruction error over temporal frequencies ω_t for different methods. The plot is based on an assumed standard error of the sampled conventional image $|\epsilon_F| = 0.1$, standard error of sampled gradients $|\epsilon_G| = 0.02$, and $\alpha = 0.2$, which is a typical scenario in practice. The dotted lines are from an empirical experiment to confirm the theory, where we added Gaussian noise with known variance to a sequence of images and their spatial and temporal differences, and then performed reconstruction numerically. The plots show an equal time comparison based on implementation details as described in Section 5.*

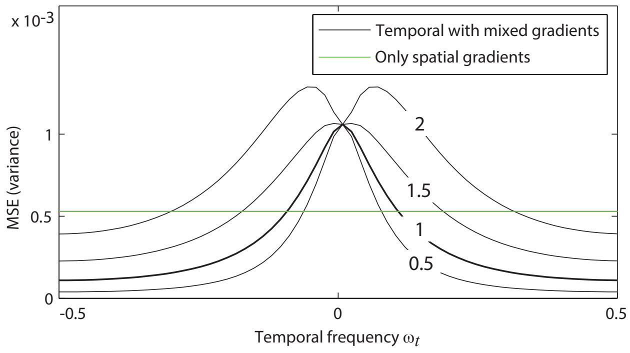


Figure 5: *We show the effect of different ratios of variances in temporal and mixed versus spatial gradients on the error in temporal frequencies. We plot ratios $\{0.5, 1, 1.5, 2\}$. Even if temporal and mixed gradients have more variance than spatial ones, which can be caused by fast camera or object motion, we still obtain high frequency noise reduction.*

white noise, the variances are identical across frequencies, simplifying the analysis. Kettunen et al. derive the relation between $|\epsilon_G|^2$ and $|\epsilon_F|^2$ under suitable simplifying assumptions, and show that typically sampled gradients have lower variance than sampled pixels. We take this as given.

Let us denote the Frequency domain MSE of the final screened L_2 Poisson reconstruction of the spatio-temporal animation sequence by $|\epsilon_{R_\alpha}(\omega_x, \omega_y, \omega_t)|^2$. Following the analysis of Kettunen et al., it is easy to see that in contrast to the errors $|\epsilon_F|^2$ and $|\epsilon_G|^2$ in the sampled base image and its gradients, the final reconstruction error *does* vary with spatio-temporal frequency ω_x, ω_y and ω_t. Introducing our temporal and mixed differences into their final reconstruction error, we get

$$|\epsilon_{R_\alpha}(\omega_x, \omega_y, \omega_t)|^2 = \frac{\alpha^4|\epsilon_F|^2 + |\epsilon_G|^2(|D^x|^2 + |D^y|^2 + |D^t|^2 + |D^{tx}|^2 + |D^{ty}|^2)}{(\alpha^2 + |D^x|^2 + |D^y|^2 + |D^t|^2 + |D^{tx}|^2 + |D^{ty}|^2)^2}, \tag{4}$$

where $|D^x(\omega_x, \omega_y, \omega_t)|^2 = 2 - 2\cos(2\pi\omega_x)$ is the power spectrum of the finite difference operator along horizontal spatial frequencies ω_x, and we omitted the arguments in the equation above for brevity. The spectra of the other finite difference operators along vertical spatial $|D^y|^2$ and temporal dimensions $|D^t|$ are defined analogously. The power spectrum of the mixed finite difference operator $|D^{tx}|^2$ is the product $|D^{tx}|^2 = (2 - 2\cos(2\pi\omega_x))(2 - 2\cos(2\pi\omega_t))$, and similar for $|D^{ty}|^2$. The error of screened Poisson reconstruction without temporal and mixed gradients, as in Kettunen et al. [2015], corresponds to Equation 4 without the $|D^t|^2$, $|D^{tx}|^2$, and $|D^{ty}|^2$ terms.

To gain insight, Figure 4 plots the reconstruction error from Equation 4 over temporal frequencies $|\epsilon_{R_\alpha}(\omega_t)|^2$, with spatial dimensions averaged over, assuming equal time taken for sampling. We compare standard GPT without temporal gradients (green) [Kettunen et al. 2015], temporal GPT with only first-order temporal differences $\mathbf{I}^{\mathrm{dt}}$ (red), and the full temporal GPT including also mixed differences $\mathbf{I}^{\mathrm{dxdt}}, \mathbf{I}^{\mathrm{dydt}}$ (black). Note that all algorithms include standard spatial gradients. The comparisons are produced by omitting the respective terms in Equation 4.

We see that without temporal differences, the error is white noise over time (green curve). This is expected, as the frames are sampled independently. The main benefit of temporal differences (red and black curves) is that high frequency temporal noise (distracting flicker) is reduced, in analogy to purely spatial gradient-domain rendering. In addition, we observe that mixed gradients (black) are more effective at high frequency noise suppression than using only temporal gradients (red). At equal computation costs, the temporal low frequency errors of both are higher than with only spatial gradients because of the additional overhead to compute the temporal and mixed gradients. This is not as perceptible as fast flickering, however, and the end result is more pleasing. In the figure, the MSE of conventional rendering without gradients is a constant $|\epsilon_F|^2 = 10 \times 10^{-3}$, which is long outside the range of the plots in the figure.

In Figure 5, we investigate what happens when the sampled spatial and temporal gradients have different variances. We slightly extend Equation 4 by allowing for separate variances for spatial gradients $|\epsilon_G|^2$, and temporal and mixed gradients $c|\epsilon_G|^2$, and denote their ratio by c. We plot curves for $c \in \{0.5, 1, 1.5, 2\}$, a typical range we observe in practice. The main observation is that even if temporal and mixed gradients have higher variance than the spatial ones, which may be due to fast object or camera motion, we still obtain some reduction in high frequency temporal error.

4 Extensions

4.1 Adaptive Sampling

Many previous authors have shown that distributing sampling efforts non-uniformly over the image plane to minimize the error of the output image may be beneficial. See Zwicker et al. [2015] for a recent survey. In addition, the energy of the gradients, and hence their variance, is usually sparsely distributed over the image [Olshausen and Field 1996]. This suggests that adaptive sampling may be even more effective in gradient-domain rendering, and that the sampling distributions for gradient-domain rendering will have stronger non-uniformity than ordinary adaptive sampling.

Ideally, to optimize output quality we should distribute samples according to the error of our reconstruction. As this is hard to estimate directly, we build on the following observation. We have found empirically that for reasonably high sampling rates, the variance of the L_1-reconstruction of individual frames correlates strongly with

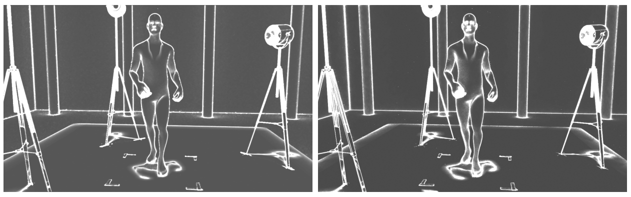


Figure 6: *Sampling distribution.* Left: *Average sampling map based on the variance of the spatial gradients over* 100 *runs at* 128 *samples per pixel in a frame of the* RUNNING MAN *scene.* Right: *Sampling map based on the variance of the L_1 reconstruction. We obtained the variance of the L_1 reconstructions by running the reconstruction* 100 *times on differently seeded input data at* 128 *samples per pixel each. The sampling maps look very similar, hence we use the estimated variance of the spatial gradients in practice.*

the sample variance of the spatial gradients. This is illustrated in Figure 6. While we cannot offer a theoretical justification for this, we further observe, and our results show, that distributing samples in each frame in proportion to the estimated variance of the spatial gradients yields significant improvements.

To get reliable variance estimates we distribute our samples in several batches: We distribute a first batch uniformly over the image, and use it to get an initial estimate of the sampling variance of the spatial gradients. We distribute all following batches such that they minimize the relative variance (variance divided by luminance of the pixel) of the gradients, and we also use them to update the variance estimates. Although this progressive adaptive sampling approach introduces bias, we consider it negligible in combination with our biased L_1 reconstruction. Usually the variance estimates of the gradients remain very noisy even for high sampling counts, in particular in scenes with specular paths. Hence we apply a 5×5 median filter on the variance estimates before using them to obtain sampling density maps. We further smooth the sampling density maps with a separable kernel $[0.1, 0.2, 0.4, 0.2, 0.1]$ and clamp the maximum values to eight times the average pixel sample count per sampling iteration to avoid sudden changes of sampling density.

Figure 7 compares the sampling distributions and final images achieved by the described algorithm to a standard path tracer with similar adaptive sampling based on the relative variance of its (regular) path samples. Adaptive sampling according to variance is clearly more effective in the gradient domain. This is due to the gradient sampler's ability to divert effort away from pixels where the underlying path space is smooth over the image coordinates. The regular sampler cannot exploit this, and is forced to a more uniform distribution by the variance along all path dimensions.

The primary sample space shift over time (Section 3.2) implies that the time shifted paths must be generated with the same random seeds as the paths in the base frame. In addition, the temporal offset frame must use the same sampling distribution as the base frame. To ensure this, we compute the sampling distribution map once for the base frame and store it on disk. To render the temporal offset frame, we read the sampling distribution map back from disk and distribute samples according to it.

4.2 Motion Vectors

Generating correlated pairs of path samples in two consecutive frames can be challenging. In presence of object or camera movement, the same primary sample space path shifted from one frame to the next, as described in Section 3.2, can hit very different geometry in these two frames. This leads to a high variance of the temporal differences. We mitigate this issue by taking into account scene and camera motion through per-pixel motion vectors that represent the motion of primary hit points from one frame to the next: instead of tracing the temporal offset path through the same pixel as the base path, we trace it through the corresponding pixel given by its motion vector. The intuition is that if we construct a temporal difference such that it follows the motion of the underlying object point, this often reduces the magnitudes of the temporal differences, and hence their variance. Of course, this procedure must be accounted for by the reconstruction algorithm. Note that this does not break consistency; good tracking decreases variance, but bad tracking does not break the algorithm. Many renderers, including Mitsuba that we build on, provide motion vectors for generating motion blur effects in post-processing.

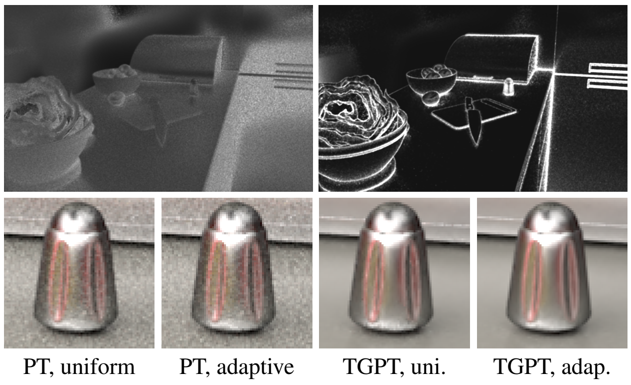


Figure 7: Top: *Sampling distribution based on the relative primal image variance (left) compared to sampling distribution based on the relative gradient variance (right) on the* KITCHEN 1 *scene. We computed the densities with four sampling batches with an average of 256 samples per pixel each.* Bottom: *path tracing (PT) vs. our approach (TGPT) at equal render time, both with uniform and adaptive sampling using the respective sampling maps from the top row. Path tracing does not benefit from adaptive sampling here, while adaptive TGPT benefits from the sparsity in the sampling map and produces a visible improvement over uniform TGPT.*

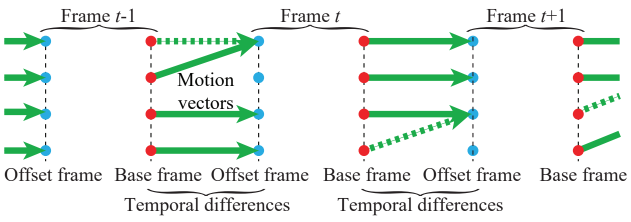


Figure 8: *Obtaining temporal differences with motion vectors. We render each frame twice using GPT, as a base (red) and an offset frame (blue circles as pixels). We obtain temporal differences as the difference between the offset frame at time $t+1$ and the base frame at time t, where pixel correspondences are given by the motion vectors (green arrows). Corresponding pixels use the same random seed to implement the primary sample space temporal shift (Section 3.2). Dotted green arrows indicate temporal differences that we ignore as described in Section 4.2.2.*

4.2.1 Motion-aware Temporal Differences

In our pipeline, we obtain the temporal differences by rendering each frame twice, as a base and a temporal offset frame (Section 3.2). We render both with GPT, so each consists of a primary image and its spatial gradients. Then, temporal and mixed differences are obtained by simply subtracting the base from its succeeding temporal offset frame. Here we discuss how to include motion vectors in this process, as illustrated in Figure 8.

When rendering the base frame, we also compute motion vectors, and store them to disk. A motion vector represents the movement of a surface location between two frames, projected to the image plane and measured as a 2D pixel offset rounded to the closest integers. All motion vectors of a frame form a motion vector image. To render the temporal offset frame, we now read the motion vector image of the *previous* frame, compute the pixel correspondences between the previous and the current frame, and then render the offset frame using the primary sample space shift (Section 3.2), i.e., using the primary sample space coordinates of the corresponding samples in the previous frame. If adaptive sampling is used, we also look up the sampling density in the previous frame.

After rendering, we average the base and offset frames to obtain the primary images $\mathbf{I}^g$ and the spatial gradients $\mathbf{I}^{dx}, \mathbf{I}^{dy}$. Next we warp each temporal offset frame onto the previous frame according to the motion vectors, that is, each pixel in a temporal offset frame is warped backward along its incident motion vector (green arrows in Figure 8). Temporal and mixed gradients $\mathbf{I}^{dt}$, $\mathbf{I}^{dxdt}$ and $\mathbf{I}^{dydt}$ are then obtained by subtracting the base frame (including its primary and spatial gradient images) from the warped, temporal offset frame. As shown in Figure 9, motion-aware temporal differencing generally significantly decreases the differences' magnitudes.

4.2.2 Reconstruction with Motion Vectors

Our conjugate gradient 3D Poisson solver computes $\mathbf{A}^T\mathbf{A}$ on the fly, which requires efficient access to the non-zero elements in rows of $\mathbf{A}$ and $\mathbf{A}^T$. Because we use motion vectors, however, the structure of $\mathbf{A}$ is modified, and not shift invariant any more. For simplicity, we explain how we take into account the motion vectors for the temporal constraints only. Computing the spatio-temporal constraints follows the same principles.

The temporal constraints consist of pairs of pixels (i, t) and $(\phi(i, t), t + 1)$, where i is a pixel index, t a frame, and $\phi(i, t)$ is the index of the corresponding pixel in the frame $t + 1$ given by the motion vector at pixel i and frame t. The row of $\mathbf{A}$ that represents the temporal gradient for pixel i at frame t has only two non-zero elements, given by the pixels i in frame t and $\phi(i, t)$ in frame $t+1$. Each row in the transpose $\mathbf{A}^T$ corresponds to a pixel i at frame t, and its non-zero elements correspond to all the constraints that (i, t) is involved in. This includes the temporal difference of some pixel j from frame $t - 1$, which maps to i via the motion vectors, that is, $\phi(j, t-1) = i$. To find j we need to construct the inverse mapping ϕ^{-1}. For this mapping to be well-defined the motion map ϕ must be one-to-one, which unfortunately is usually not the case a priori.

We achieve invertibility of ϕ by also querying for a reverse motion map from frame $t+1$ to t. We require that whenever ϕ maps a pixel to the next frame, the reverse motion map maps the result back to the original pixel. We allow a tolerance of one pixel since the motion vectors are rounded to the nearest pixel. This tests that a pixel and its correspondence in the next frame represent the same object, as pixels often represent objects that get occluded in the next frame. Including temporal difference constraints between different objects would lead to higher variance. We then check for any remaining cases of multiple pixels mapping onto one, and keep the ones that are mapped closest to the camera.

This means that not all pixels have a motion vector, hence some pixels do not have a temporal difference constraint. This is allowed since each pixel is still constrained by the primary image and the spatial gradients. With both the final ϕ and its inverse ϕ^{-1} accessible for the CUDA kernels, we quickly look up the non-zero elements in the rows of both $\mathbf{A}$ and $\mathbf{A}^T$ using $\phi(i, t)$ and $\phi^{-1}(i, t)$.

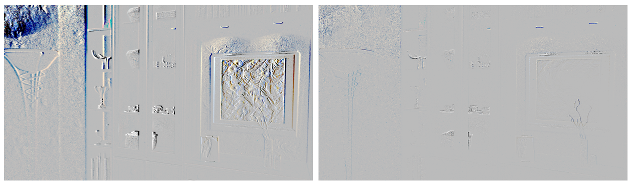


Figure 9: *Temporal differences with motion vectors.* Left: *Motion unaware temporal differences in a frame of the* BOOKSHELF *sequence at 512 samples per pixel.* Right: *Motion aware temporal differences for the same setup. The motion-aware temporal differences on the right are significantly smaller (gray represents zero).*

5 Implementation

We implemented our approach on top of the public GPT code by Kettunen et al. [2015] and made our code available on our web page. We built a spatio-temporal screened Poisson solver on the GPU, and process arbitrarily long animations by reconstructing overlapping sub-sequences. We use windows with 10 frames and overlap with the next frame by 5 frames. This limits temporal data re-use to 10 frames, but we did not observe any benefits of larger temporal windows in practice. We produce the final output by smoothly blending the overlapping regions of the sub-sequences. Reconstructing 10 frames at a resolution of 1280×720 pixels takes around 15 seconds on an Nvidia GTX Titan and consumes roughly 2.6 GB of VRAM on the GPU. The only user parameter of the reconstruction is α (Equation 3), which we set to $\alpha = 0.2$ for all our results.

6 Results and Discussion

In the following figures and the accompanying video we compare path tracing, gradient-domain path tracing, and our approach. All results are rendered with motion blur, which our approach handles without requiring any modifications, with an exposure time of half a frame. We further include comparisons of our approach with and without the proposed extensions (adaptive sampling and motion vectors).

Figure 10 visualizes the effect of temporal gradient sampling. At the top, we plot the luminance of a pixel over time for the different methods. At the bottom, we show a visual comparison using a space-time image of a short vertical 1D image segment over time. The plots and visualizations clearly show how our technique improves the temporal smoothness of the result. We also see the benefits of our extensions including adaptive sampling and motion vectors, which further reduce residual noise. The video and Figure 11 feature visual comparisons of our approach with conventional path tracing (PT), gradient-domain path tracing (GPT), and our approach (TGPT) at equal computation time. In the figure, we show one frame of the animation sequences for five test scenes (BOOKSHELF, SPONZA, RUNNING MAN, KITCHEN 1, KITCHEN 2). In addition to reducing temporal artifacts, TGPT clearly also suppresses spatial artifacts that are still visible in GPT.

To disentangle the effect of adaptive sampling and temporal differencing, the video includes a four-way comparison between spatial-only GPT, spatial-only GPT with adaptive sampling, TGPT with-

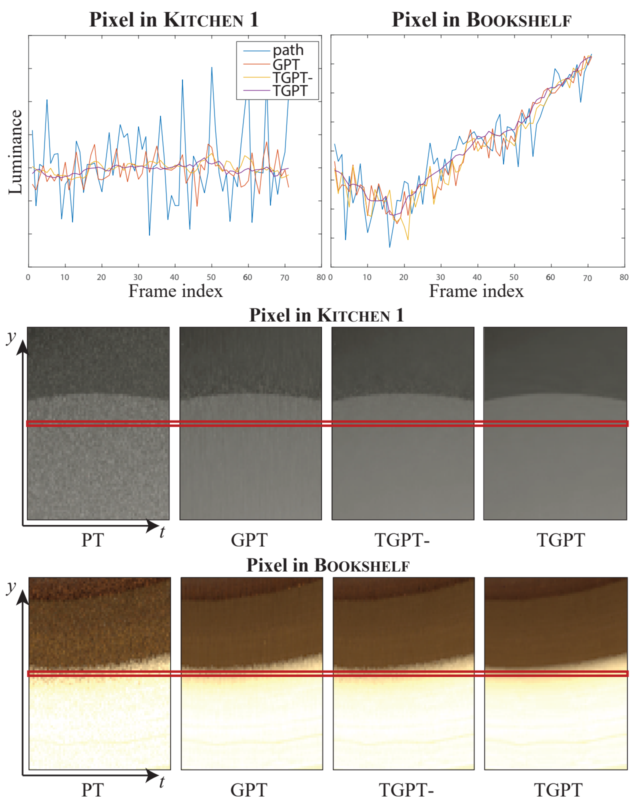


Figure 10: Top: *Temporal plots for one pixel in* KITCHEN 1 *and* BOOKSHELF. Bottom: *A* 95 × 1 *crop centered around the plotted pixels over time. We track the crops over time using the motion vectors of the central pixel, such that they are stationary relative to the surface hit point at the central pixel. Because rounding errors of the motion vectors accumulate over time, the image crops are still shifting a bit with respect to the central pixel. "TGPT-" denotes TGPT without any of the extensions discussed in Section 4.*

out adaptive sampling, and TGPT with adaptive sampling using the RUNNING MAN scene. Comparing the two spatial gradient-domain algorithms, we observe that the clear improvement due to adaptive sampling at corners and other high-variance areas comes at the price of *increased* temporal flicker in the smooth areas. This is to be expected, as the sample budget is limited and adaptivity merely shuffles samples around. Temporal differencing reduces the flicker drastically, but without adaptivity, corners and glossy areas still suffer from variance. The combination of adaptation and temporal differencing produces clearly the best results.

We further study the effect of the speed of motion on TGPT in the video by comparing the RUNNING MAN animation at $1\times$, $2\times$ and $4\times$ the original animation speed. We can observe that the quality of the reconstruction degrades with increasing motion speed. The reason for this is three-fold: First, motion blur becomes more prominent, and regions where one single motion vector is insufficient to describe the motion of the underlying pixels become larger. Second, larger changes in scene geometry from frame to frame lead to less correlation between the base and offset paths, and hence to larger temporal differences. And third, larger regions with motion blur tend to make the adaptive sampling distribution more uniform, thus making adaptive sampling less effective.

Finally, the video includes an experiment where only the temporal differences are used in TGPT, that is, we omit spatial and spatio-temporal gradients. This method is attractive because it merely augments a simple path tracer with the temporal difference machinery, which operates on primary sample space and is very simple to implement. Without spatial differences, however, diffusion of the noise in the reconstruction can only occur along the time dimension. Hence, much longer temporal reconstruction windows are required to achieve significant noise reduction, which makes GPU memory management more challenging. The experiment in the video uses a temporal reconstruction window of 10 frames and we observe some noise reduction, but the noise appears *glued* to the surfaces and the result is visually unappealing.

6.1 Discussion and Limitations

Standard (spatial-only) gradient-domain path tracing only reduces noise in regions where the path throughput function is smooth under the shift mapping, i.e., where correlated spatially neighboring path pairs can be generated. Similarly, TGPT only reduces flickering in temporally smooth regions. This motivates our use of motion vectors to find similar pixel pairs in adjacent frames. Fast-moving or high-frequency moving geometry, however, increases variance as the temporal signal becomes less smooth. In addition, like all gradient-domain methods so far, TGPT cannot overcome the weaknesses of the underlying sampling method. For instance, it is poor at resolving caustics, since the underlying path tracer cannot sample them efficiently. Conversely, when the underlying sampler performs well, TGPT is very effective.

Several alternatives could be explored to reduce the variance of temporal differences, like adapting the half-vector preserving [Kettunen et al. 2015] or the manifold-walk based shift [Lehtinen et al. 2013] to the temporal domain. This would require information about the motion of each path vertex from one frame to the next, however, which is not provided in most existing renderers. We also experimented with using different weights for the spatial, temporal, and mixed gradients in the screened Poisson reconstruction, but did not obtain any significant improvements. We observed, however, that the local distribution of variances is rather different between the different types of constraints. This indicates that locally adapting the weights could be more effective than setting them globally.

7 Conclusions

We presented a temporal extension of gradient-domain rendering that significantly reduces temporal flickering artifacts compared to previous work. Our approach is unique in that it exploits temporal coherence already during Monte Carlo sampling, instead of imposing it solely in a post-process. We propose a simple temporal shift mapping to obtain temporal gradients using a primary sample space shift, which can be implemented with only small changes to an existing gradient-domain renderer. Our approach also uses mixed-spatio temporal gradients, and a frequency-analysis shows that they can further reduce temporal high frequency error. In addition, we proposed extensions to include adaptive sampling and motion vectors, which effectively boost the quality of our results. In the future we would like to investigate more sophisticated spatial and temporal shift mappings to further improve the quality of gradient-domain rendering. We will also investigate other gradient stencils that may provide additional benefits over our current spatial, temporal, and mixed stencils. Finally, it would be interesting to analyze the benefits and limits of adaptive sampling from a theoretical perspective.

Acknowledgments

We thank Sampo Rask for RUNNING MAN, and Benedikt Bitterli and blendswap.com user Jay-Artist for KITCHEN 2. This work was supported by the Swiss National Science Foundation, projects 143886 and 163045, NSF project 1451830, and the Academy of Finland, grant 277833.

References

BONNEEL, N., SUNKAVALLI, K., TOMPKIN, J., SUN, D., PARIS, S., AND PFISTER, H. 2014. Interactive intrinsic video editing. *ACM Trans. Graph. 33*, 6 (Nov.), 197:1–197:10.

BONNEEL, N., TOMPKIN, J., SUNKAVALLI, K., SUN, D., PARIS, S., AND PFISTER, H. 2015. Blind video temporal consistency. *ACM Trans. Graph. 34*, 6 (Oct.), 196:1–196:9.

CHEN, S. E. 1990. Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system. *SIGGRAPH Comput. Graph. 24*, 4 (Sept.), 135–144.

GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., AND BATTAILE, B. 1984. Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph. 18*, 3 (Jan.), 213–222.

HAVRAN, V., DAMEZ, C., MYSZKOWSKI, K., AND SEIDEL, H.-P. 2003. An efficient spatio-temporal architecture for animation rendering. In *Proceedings of the 14th Eurographics Workshop on Rendering*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, EGRW '03, 106–117.

KELEMEN, C., SZIRMAY-KALOS, L., ANTAL, G., AND CSONKA, F. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. *Computer Graphics Forum 21*, 3, 531–540.

KELLER, A., FASCIONE, L., FAJARDO, M., GEORGIEV, I., CHRISTENSEN, P., HANIKA, J., EISENACHER, C., AND NICHOLS, G. 2015. The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses*, ACM, New York, NY, USA, SIGGRAPH '15, 24:1–24:7.

KETTUNEN, M., MANZI, M., AITTALA, M., LEHTINEN, J., DURAND, F., AND ZWICKER, M. 2015. Gradient-domain path tracing. *ACM Trans. Graph. 34*, 4 (July), 123:1–123:13.

LEHTINEN, J., KARRAS, T., LAINE, S., AITTALA, M., DURAND, F., AND AILA, T. 2013. Gradient-domain metropolis light transport. *ACM Trans. Graph. 32*, 4 (July), 95:1–95:12.

LI, T.-M., WU, Y.-T., AND CHUANG, Y.-Y. 2012. Sure-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph. 31*, 6 (Nov.), 194:1–194:9.

MANZI, M., ROUSSELLE, F., KETTUNEN, M., LEHTINEN, J., AND ZWICKER, M. 2014. Improved sampling for gradient-domain metropolis light transport. *ACM Trans. Graph. 33*, 6 (Nov.), 178:1–178:12.

MANZI, M., KETTUNEN, M., AITTALA, M., LEHTINEN, J., DURAND, F., AND ZWICKER, M. 2015. Gradient-domain bidirectional path tracing. In *Proc. Eurographics Symposium on Rendering*.

MCCOOL, M. D. 1999. Anisotropic diffusion for monte carlo noise reduction. *ACM Trans. Graph. 18*, 2 (Apr.), 171–194.

MEYER, M., AND ANDERSON, J. 2006. Statistical acceleration for animated global illumination. *ACM Trans. Graph. 25*, 3 (July), 1075–1080.

MOON, B., CARR, N., AND YOON, S.-E. 2014. Adaptive rendering based on weighted local regression. *ACM Trans. Graph. 33*, 5 (Sept.), 170:1–170:14.

NIMEROFF, J., DORSEY, J., AND RUSHMEIER, H. 1996. Implementation and analysis of an image-based global illumination framework for animated environments. *IEEE Transactions on Visualization and Computer Graphics 2*, 4 (Dec.), 283–298.

OLSHAUSEN, B., AND FIELD, D. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature 381*, 607–609.

ROUSSELLE, F., MANZI, M., AND ZWICKER, M. 2013. Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7, 121–130.

RUSHMEIER, H. E., AND WARD, G. J. 1994. Energy preserving non-linear filters. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '94, 131–138.

SCHERZER, D., YANG, L., MATTAUSCH, O., NEHAB, D., SANDER, P. V., WIMMER, M., AND EISEMANN, E. 2012. Temporal coherence methods in real-time rendering. *Computer Graphics Forum 31*, 8, 2378–2408.

SEN, P., AND DARABI, S. 2012. On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph. 31*, 3 (June), 18:1–18:15.

SMYK, M., KINUWAKI, S.-I., DURIKOVIC, R., AND MYSZKOWSKI, K. 2005. Temporally coherent irradiance caching for high quality animation rendering. *Computer Graphics Forum 24*, 3, 401–412.

TAWARA, T., MYSZKOWSKI, K., AND SEIDEL, H.-P. 2004. Exploiting temporal coherence in final gathering for dynamic scenes. In *Proc. Computer Graphics International, 2004*, 110–119.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *Proc. of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '97, 65–76.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph. 22*, 4 (June), 85–92.

ZIMMER, H., ROUSSELLE, F., JAKOB, W., WANG, O., ADLER, D., JAROSZ, W., SORKINE-HORNUNG, O., AND SORKINE-HORNUNG, A. 2015. Path-space motion estimation and decomposition for robust animation filtering. *Computer Graphics Forum 34*, 4, 131–142.

ZWICKER, M., JAROSZ, W., LEHTINEN, J., MOON, B., RAMAMOORTHI, R., ROUSSELLE, F., SEN, P., SOLER, C., AND YOON, S.-E. 2015. Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *Computer Graphics Forum 34*, 2, 667–681.

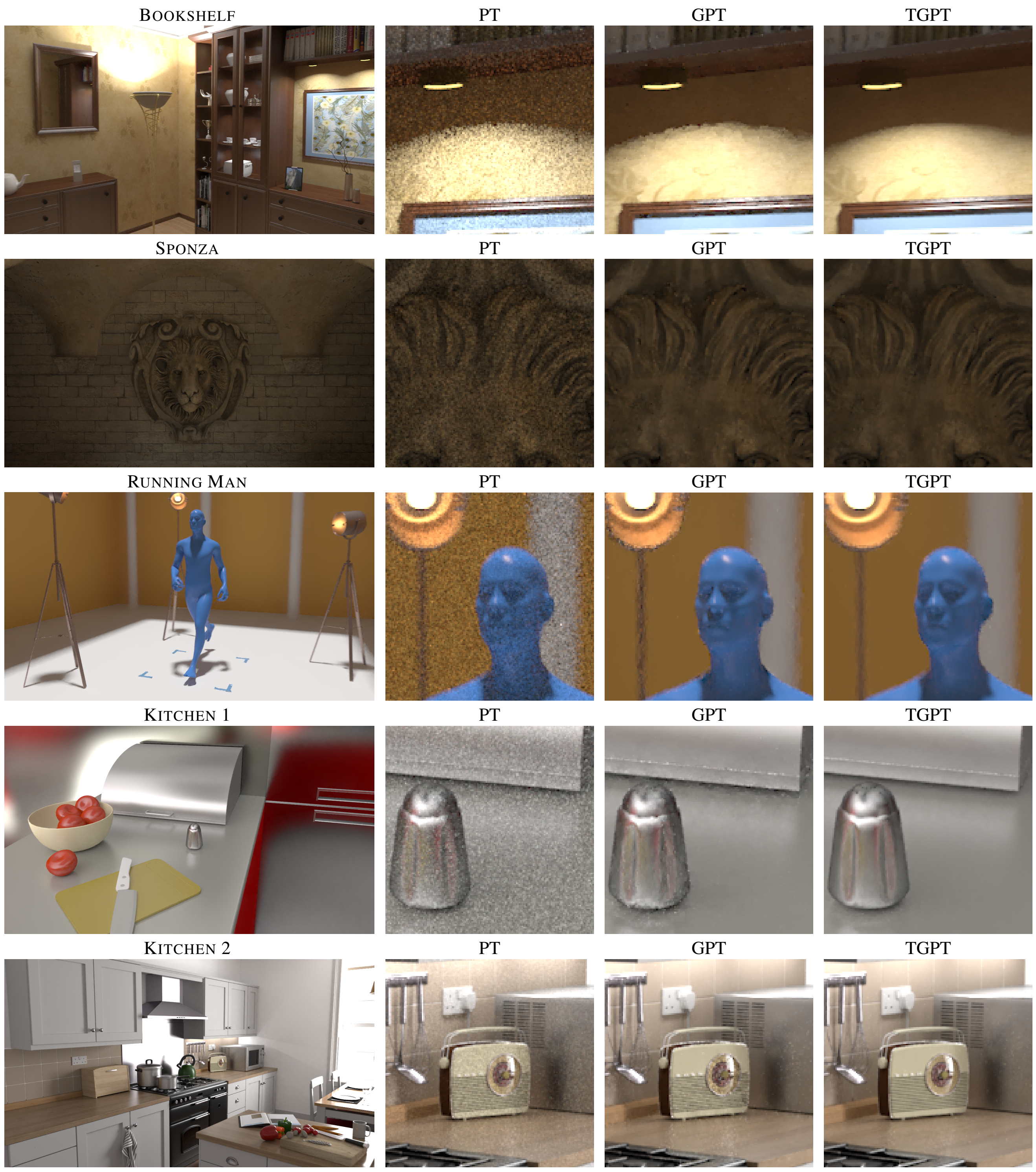


Figure 11: *Comparison of our approach with conventional path tracing (PT), gradient-domain path tracing (GPT), and our proposed temporal gradient-domain algorithm (TGPT) at equal computation time. We show one frame of the animation sequences for our five test scenes, and a close-up each. In addition to reducing temporal artifacts, TGPT clearly also suppresses residual spatial artifacts that are still visible in GPT. We rendered the first three scenes, from top to bottom, with 32 samples per pixel (spp) using GPT and TGPT, and 80 spp using PT, which amounts to equal time given the* $2.5\times$ *overhead of GPT over PT. The last two scenes have 256 spp for GPT and TGPT, and 640 spp for PT. TGPT splits the samples per frame into two halves to render each frame twice, as temporal base and offset frame (Figure 8).*

Chapter 10

Conclusions

Photo-realistic computer-generated images that grow in complexity and realism have been a driving force behind the development of more efficient rendering algorithms. Gradient-domain rendering is one of these methods. It reduces noise of rendered images by sampling the data in the gradient-domain. Then is uses this data to create a final image with a screened Poisson reconstruction. In contrast to adaptive sampling and reconstruction methods that also aim at reducing the noise of the rendered image (Section 3.3.2), gradient-domain rendering methods are more general. They do not rely on auxiliary information that is obtained during rendering[1] and can be unbiased if the reconstruction uses the L_2-norm. Gradient-domain rendering is based on the idea to sample the image in a sparser domain than traditional rendering methods. By doing so, we can concentrate on the regions of an image or path space where something interesting happens. Additionally, it turns out that the the finite differences of the path throughput that are sampled with gradient-domain rendering can benefit greatly from correlated sampling techniques.

The first algorithm that used gradient-domain rendering is gradient-domain Metropolis light transport [68]. However, this algorithm suffers from uneven convergence, is ill suited for animations and is challenging to implement. This made the algorithm unattractive for being used in industry. Thus, the goal of this thesis was to develop new algorithms that make gradient-domain rendering more useful in practice. In this thesis different directions were explored to achieve this goal, including:

- generalizing the finite difference constraints,
- combining different shift mappings,
- adapting gradient-domain rendering to different base path sampler,
- regularizing the reconstruction and
- augmenting the dimensionality of the problem to the temporal domain.

While most of these contributions have been discussed, analysed and implemented separately, combining them is expected to yield even greater benefits.

[1]With the exception of some of the extensions described in Chapters 6 and 8.

Our first contribution (Chapter 6) generalizes the original method to use arbitrary difference constraints instead of finite differences. This leads to an algorithm called bilateral-domain Metropolis light transport. Intuitively, this generalization tries to take the basic idea of gradient-domain rendering further than the original method: instead of sampling in the gradient-domain this method uses precomputed information of the scene to sample in an even sparser domain. Important and easy to detect features like texture edges or geometrical edges are baked into the Laplace matrices of the reconstruction such that there is no need to sample those features explicitly. This leads to an even better usage of the samples. Additionally, we described how several specialized shift mappings could be combined in an unbiased way such that specialized shift mappings could be used in different regions of the path space.

Our second contribution (Chapter 7) applies the principles of gradient-domain rendering on bidirectional path tracing. This leads to gradient-domain bidirectional path tracing (G-BDPT). We show that applying gradient-domain rendering to BDPT causes a combinatorial explosion of paths that need to be shifted. We introduce a smarter gradient generation scheme that significantly reduces the number of required shift operations.

Our third contribution (Chapter 8) modifies the reconstruction step of gradient-domain rendering to produce smooth results even at very low sample counts. Similar to recent adaptive sampling and reconstruction methods it uses auxiliary feature information to regularize the image. However, in contrast to adaptive sampling and reconstruction methods our approach uses a *global* regularization. This is achieved by adding new constraints to the reconstruction step. These new constraints enforce smoothness by regularizing image patches based on auxiliary features. While providing usually better results than the original screened Poisson reconstruction, this regularized reconstruction generates consistent but biased results. This method is general enough to be applicable on any gradient-domain rendering method.

Our final contribution (Chapter 9) is the extension of gradient-domain rendering on the temporal domain. We discuss how the 2D problem of reconstructing an image can be extended to a 3D problem of reconstructing a *sequence* of images. We do so by introducing temporal constraints between pixels in different frames and reconstructing batches of frames at once. We show how this method greatly suppresses high frequency noise along the temporal dimension and thus reduces temporal flickering. This extension makes gradient-domain rendering even more useful for animation rendering. We further discuss how adaptive sampling according to the variance of the gradients reduces noise even more.

10.1 Future Work

We believe that gradient-domain rendering provides several avenues for future research. One obvious improvement would be to develop new shift mappings that lead to even higher correlation than the existing manifold exploration shift and half-vector shift (Sections 5.4.2 and 5.5.2). Shifts based on more recent mutation techniques like the natural constraint representation [52, 35] or anisotropic Gaussian mutations [70] are thinkable. Also, it would be interesting to explore if different shift mappings can be combined in a way similar to MIS (Chapter 3.3.4) without introducing bias. While

Chapter 6 describes a method to combine different shift mappings with arbitrary weights in theory, it only used binary weights to do so. Robust combination schemes would open the door to designing very specialized shift mappings for specific subspaces of the path space.

An additional generalization of gradient-domain rendering would be to develop shift mappings that support participating media, since it usually is a strong source of variance with traditional (MC-)MC methods. Also, since participating media often changes rather slowly over path space, correlation of the base and offset paths would not necessary decrease by a lot due to participating media. We thus believe that gradient-domain rendering would be very well suited for reducing this type of noise.

Further, temporal G-PT (Chapter 9) showed that augmenting gradient-domain rendering by an additional time dimension helps at reconstructing animations. We believe that similar extensions over additional dimensions could be beneficial in cases when higher dimensional signals are generated, e.g. when light-fields or even animated light-fields are rendered as a whole.

Gradient-domain rendering that was originally developed on top of MLT, was later applied on unidirectional path tracing and BDPT. This shows that gradient-domain rendering is a very general technique that can be applied on a multitude of path samplers. Naturally, it would thus be interesting to adapt the algorithm to be applicable to other state-of-the-art MC rendering methods like vertex merging [28, 33] and many-light methods [56, 14].

Finally, bilateral-domain MLT (Chapter 6) showed that methods based on the same idea as gradient-domain rendering are not confined to use finite difference kernels to compute the smoothness constraints of the screened Poisson reconstruction. While there is an analysis for the finite difference kernels used in G-PT (Section 5.3), no such analysis exists for the structurally-adaptive kernels used in bilateral-domain MLT so far. We believe that a solid theoretical understanding of the benefits of those kernels could yield new insights with potential for better structurally-adaptive kernels. In general it is unclear what kind of kernels would be optimal. For instance, it would be interesting to explore if higher order finite differences could be beneficial either used alone or in conjunction with classic gradient-domain rendering. Another avenue could be to move away from sampling differences of paths altogether. For instance, Overbeck et al. [88] showed that sampling and reconstruction according to a wavelet basis can be beneficial in the context of adaptive sampling and reconstruction. The ideas used in gradient-domain rendering could probably be adapted to sample paths in the wavelet-domain directly. On a more fundamental level this raises the question of what the optimal domain is in which path space should be sampled in order to minimize the computational effort to render images. Research in this direction could give raise to new physically-based rendering algorithms that converge even faster.

Bibliography

[1] James Arvo and David Kirk. Particle transport and image synthesis. *SIGGRAPH Comput. Graph.*, 24(4):63–66, September 1990.

[2] James Richard Arvo. *Analytic Methods for Simulated Light Transport*. PhD thesis, New Haven, CT, USA, 1995. AAI9619140.

[3] Michael Ashikmin, Simon Premože, and Peter Shirley. A microfacet-based brdf generator. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 65–74, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[4] Pablo Bauszat, Martin Eisemann, and Marcus Magnor. Guided image filtering for interactive high-quality global illumination. In *Proceedings of the Twenty-second Eurographics Conference on Rendering*, EGSR '11, pages 1361–1368, Aire-la-Ville, Switzerland, Switzerland, 2011. Eurographics Association.

[5] Laurent Belcour, Kavita Bala, and Cyril Soler. A local frequency analysis of light scattering and absorption. *ACM Trans. Graph.*, 33(5):163:1–163:17, September 2014.

[6] Laurent Belcour, Cyril Soler, Kartic Subr, Nicolas Holzschuch, and Fredo Durand. 5d covariance tracing for efficient defocus and motion blur. *ACM Trans. Graph.*, 32(3):31:1–31:18, July 2013.

[7] Mark R. Bolin and Gary W. Meyer. An error metric for monte carlo ray tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, pages 57–68, London, UK, UK, 1997. Springer-Verlag.

[8] Eva Cerezo, Frederic Pérez, Xavier Pueyo, J. Francisco Seron, and X. François Sillion. A survey on participating media rendering techniques. *The Visual Computer*, 21(5):303–328, 2005.

[9] Petrik Clarberg and Tomas Akenine-Möller. Exploiting Visibility Correlation in Direct Illumination. *Computer Graphics Forum (Proceedings of EGSR)*, 27(4), 2008.

[10] David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tracing. *ACM Trans. Graph.*, 24(3):1186–1195, July 2005.

[11] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, January 1982.

[12] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, January 1986.

[13] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *SIGGRAPH Comput. Graph.*, 18(3):137–145, January 1984.

[14] Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. Scalable realistic rendering with many-light methods. *Comput. Graph. Forum*, 33(1):88–104, February 2014.

[15] Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik P. A. Lensch. Edge-avoiding À-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics*, HPG '10, pages 67–75, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[16] Mauricio Delbracio, Pablo Musé, Antoni Buades, Julien Chauvier, Nicholas Phelps, and Jean-Michel Morel. Boosting monte carlo rendering by ray histogram fusion. *ACM Trans. Graph.*, 33(1):8:1–8:15, February 2014.

[17] Mark A. Z. Dippé and Erling Henry Wold. Antialiasing through stochastic sampling. *SIGGRAPH Comput. Graph.*, 19(3):69–78, July 1985.

[18] Fredo Durand. A frequency analysis of Monte Carlo and other numerical integration schemes. Technical Report MIT-CSAIL-TR-2011-052, MIT CSAIL, 2011.

[19] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X. Sillion. A frequency analysis of light transport. *ACM Trans. Graph.*, 24(3):1115–1126, July 2005.

[20] Philip Dutré, Eric P. Lafortune, and Yves Willems. Monte Carlo Light Tracing with Direct Computation of Pixel Intensities. In *3rd International Conference on Computational Graphics and Visualisation Techniques*, pages 128–137, December 1993.

[21] Kevin Egan, Frédo Durand, and Ravi Ramamoorthi. Practical filtering for efficient ray-traced directional occlusion. *ACM Trans. Graph.*, 30(6):180:1–180:10, December 2011.

[22] Kevin Egan, Florian Hecht, Frédo Durand, and Ravi Ramamoorthi. Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Trans. Graph.*, 30(2):9:1–9:13, April 2011.

[23] Kevin Egan, Yu-Ting Tseng, Nicolas Holzschuch, Frédo Durand, and Ravi Ramamoorthi. Frequency analysis and sheared reconstruction for rendering motion blur. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 93:1–93:13, New York, NY, USA, 2009. ACM.

[24] Shaohua Fan, Stephen Chenney, Bo Hu, Kam-Wah Tsui, and Yu-chi Lai. Optimizing Control Variate Estimators for Rendering. *Computer Graphics Forum*, 2006.

[25] Luca Fascione. Rendering research at weta digital. Keynote speech at Eurographics Symposium on Rendering, 2015.

[26] Randima Fernando. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[27] James E. Gentle. *Matrix algebra.* Springer New York, 2007.

[28] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6):192:1–192:10, November 2012.

[29] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 213–222, New York, NY, USA, 1984. ACM.

[30] Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph.*, 27(3):33:1–33:10, August 2008.

[31] Toshiya Hachisuka, Anton S. Kaplanyan, and Carsten Dachsbacher. Multiplexed metropolis light transport. *ACM Trans. Graph.*, 33(4):100:1–100:10, July 2014.

[32] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. *ACM Trans. Graph.*, 27(5):130:1–130:8, December 2008.

[33] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. A path space extension for robust light transport simulation. *ACM Trans. Graph.*, 31(6):191:1–191:10, November 2012.

[34] Johannes Hanika, Marc Droske, and Luca Fascione. Manifold next event estimation. In *Proceedings of the 26th Eurographics Symposium on Rendering*, EGSR '15, pages 87–97, Aire-la-Ville, Switzerland, Switzerland, 2015. Eurographics Association.

[35] Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. Improved half vector space light transport. In *Proceedings of the 26th Eurographics Symposium on Rendering*, EGSR '15, pages 65–74, Aire-la-Ville, Switzerland, Switzerland, 2015. Eurographics Association.

[36] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 165–174, New York, NY, USA, 1993. ACM.

[37] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[38] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 1–14, Berlin, Heidelberg, 2010. Springer-Verlag.

[39] Eugene Hecht. *Optics.* Addison-Wesley, San Francisco, 2002. La couv. porte en plus : International edition.

[40] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *SIGGRAPH Comput. Graph.*, 24(4):145–154, September 1990.

[41] Jared Hoberock and John C. Hart. Arbitrary importance functions for metropolis light transport. *Comput. Graph. Forum*, 29(6):1993–2003, 2010.

[42] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 133–142, New York, NY, USA, 1986. ACM.

[43] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[44] Wenzel Jakob, Eugene d'Eon, Otto Jakob, and Steve Marschner. A comprehensive framework for rendering layered materials. *ACM Trans. Graph.*, 33(4):118:1–118:14, July 2014.

[45] Wenzel Jakob and Steve Marschner. Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph.*, 31(4):58:1–58:13, July 2012.

[46] Adrian Jarabo, Julio Marco, Adolfo Muñoz, Raul Buisan, Wojciech Jarosz, and Diego Gutierrez. A framework for transient rendering. *ACM Trans. Graph.*, 33(6):177:1–177:10, November 2014.

[47] Wojciech Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media.* PhD thesis, UC San Diego, September 2008.

[48] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. Irradiance gradients in the presence of participating media and occlusions. In *Proceedings of the Nineteenth Eurographics Conference on Rendering*, EGSR '08, pages 1087–1096, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[49] Henrik Wann Jensen. Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pages 21–30, London, UK, UK, 1996. Springer-Verlag.

[50] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, August 1986.

[51] M.H. Kalos and P.A. Whitlock. *Monte Carlo Methods.* Wiley, 2008.

[52] Anton S. Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Trans. Graph.*, 33(4):102:1–102:13, July 2014.

[53] Csaba Kelemen, Lszl Szirmay-Kalos, Gyrgy Antal, and Ferenc Csonka. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum*, pages 531–540, 2002.

[54] A. Keller, K. Dahm, and N. Binder. Path space filtering. In *ACM SIGGRAPH 2014 Talks*, SIGGRAPH '14, pages 68:1–68:1, 2014.

[55] A. Keller, L. Fascione, M. Fajardo, I. Georgiev, P. Christensen, J. Hanika, C. Eisenacher, and G. Nichols. The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, pages 24:1–24:7, New York, NY, USA, 2015. ACM.

[56] Alexander Keller. Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[57] Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. Gradient-domain path tracing. *ACM Trans. Graph.*, 34(4):123:1–123:13, July 2015.

[58] David Kirk and James Arvo. Unbiased sampling techniques for image synthesis. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 153–156, New York, NY, USA, 1991. ACM.

[59] T. Kollig and A. Keller. *Efficient Bidirectional Path Tracing by Randomized Quasi-Monte Carlo Integration*. Fachbereich Informatik. Fachbereich Informatik, Univ., 2001.

[60] Thomas Kollig and Alexander Keller. Efficient multidimensional sampling, 2002.

[61] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, pages 75:1–75:19, New York, NY, USA, 2008. ACM.

[62] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of 3rd International Conference on Computational Graphis and Visualizations Techniques, year = 1993, pages = 145–153, publisher =* .

[63] Eric P. Lafortune and Yves D. Willems. The ambient term as a variance reducing technique for monte carlo ray tracing. pages 163–171, June 1994.

[64] Eric P. Lafortune and Yves D. Willems. A 5d tree to reduce the variance of monte carlo ray tracing. In Pat Hanrahan and Werner Purgathofer, editors, *Rendering Techniques*, Eurographics, pages 11–20. Springer, 1995.

[65] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[66] Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. Temporal light field reconstruction for rendering distribution effects. *ACM Trans. Graph.*, 30(4):55:1–55:12, July 2011.

[67] Jaakko Lehtinen, Timo Aila, Samuli Laine, and Frédo Durand. Reconstructing the indirect light field for global illumination. *ACM Trans. Graph.*, 31(4):51:1–51:10, July 2012.

[68] 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.

[69] Oscar A.Z. Leneman. Random sampling of random processes: Impulse processes. *Information and Control*, 9(4):347 – 363, 1966.

[70] Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. Anisotropic gaussian mutations for metropolis light transport through hessian-hamiltonian dynamics. *ACM Trans. Graph.*, 34(6):209:1–209:13, October 2015.

[71] Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. Sure-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph.*, 31(6):194:1–194:9, November 2012.

[72] Marco Manzi. Adaptive sampling and reconstruction for interactive ray tracing. Master's thesis, Univerisity of Bern, 2012.

[73] Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. Gradient-domain bidirectional path tracing. In *Proc. Eurographics Symposium on Rendering*, 2015.

[74] Marco Manzi, Markus Kettunen, Fredo Durand, Matthias Zwicker, and Jaakko Lehtinen. Temporal gradient-domain path tracing. *ACM Trans. Graph. (to appear)*, 2016.

[75] Marco Manzi, Fabrice Rousselle, Markus Kettunen, Jaakko Lehtinen, and Matthias Zwicker. Improved sampling for gradient-domain metropolis light transport. *ACM Trans. Graph.*, 33(6):178:1–178:12, November 2014.

[76] Marco Manzi, Delio Vicini, and Matthias Zwicker. Regularizing Image Reconstruction for Gradient-Domain Rendering with Feature Patches. *Computer Graphics Forum*, 2016.

[77] Soham Uday Mehta, JiaXian Yao, Ravi Ramamoorthi, and Fredo Durand. Factored axis-aligned filtering for rendering multiple distribution effects. *ACM Trans. Graph.*, 33(4):57:1–57:12, July 2014.

[78] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[79] Don P. Mitchell. Generating antialiased images at low sampling densities. *SIGGRAPH Comput. Graph.*, 21(4):65–72, August 1987.

[80] Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 157–164, New York, NY, USA, 1991. ACM.

[81] Don P. Mitchell. Consequences of stratified sampling in graphics. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 277–280, New York, NY, USA, 1996. ACM.

[82] Don P. Mitchell and Arun N. Netravali. Reconstruction filters in computer-graphics. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, pages 221–228, New York, NY, USA, 1988. ACM.

[83] Bochang Moon, Nathan Carr, and Sung-Eui Yoon. Adaptive rendering based on weighted local regression. *ACM Trans. Graph.*, 33(5):170:1–170:14, September 2014.

[84] Bochang Moon, Jose A. Iglesias-Guitian, Sung-Eui Yoon, and Kenny Mitchell. Adaptive rendering with linear predictions. *ACM Trans. Graph.*, 34(4):121:1–121:11, July 2015.

[85] Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. Adaptive polynomial rendering. *ACM Trans. Graph.*, 35(4):40:1–40:10, July 2016.

[86] Fred E. Nicodemus. Directional Reflectance and Emissivity of an Opaque Surface. *Applied Optics*, 4(7):767, July 1965.

[87] Harald Niederreiter. *Random Number Generation and quasi-Monte Carlo Methods.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[88] Ryan S. Overbeck, Craig Donner, and Ravi Ramamoorthi. Adaptive wavelet rendering. *ACM Trans. Graph.*, 28(5):140:1–140:12, December 2009.

[89] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, July 2003.

[90] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[91] Stefan Popov, Ravi Ramamoorthi, Fredo Durand, and George Drettakis. Probabilistic connections for bidirectional path tracing. In *Proceedings of the 26th Eurographics Symposium on Rendering*, EGSR '15, pages 75–86, Aire-la-Ville, Switzerland, Switzerland, 2015. Eurographics Association.

[92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[93] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph.*, 30(6):159:1–159:12, December 2011.

[94] Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. Adaptive rendering with non-local means filtering. *ACM Trans. Graph.*, 31(6):195:1–195:11, November 2012.

[95] Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. Robust denoising using feature and color information. *Comput. Graph. Forum*, 32(7):121–130, 2013.

[96] Pradeep Sen and Soheil Darabi. On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph.*, 31(3):18:1–18:15, May 2012.

[97] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte carlo techniques for direct lighting calculations. *ACM Trans. Graph.*, 15(1):1–36, January 1996.

[98] Peter S. Shirley. *Physically Based Lighting Calculations for Computer Graphics.* PhD thesis, Champaign, IL, USA, 1991. UMI Order NO. GAX91-24487.

[99] F. Sillion and C. Puech. A general two-pass method integrating specular and diffuse reflection. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, pages 335–344, New York, NY, USA, 1989. ACM.

[100] Cyril Soler, Kartic Subr, Frédo Durand, Nicolas Holzschuch, and François Sillion. Fourier depth of field. *ACM Trans. Graph.*, 28(2):18:1–18:12, May 2009.

[101] Michael Spivak. *Calculus on manifolds : a modern approach to classical theorems of advanced calculus.* Mathematics monograph series. Reading, Mass. Addison-Wesley, 1965.

[102] Jos Stam. Diffraction shaders. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 101–110, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[103] Laszlo Szecsi, Mateu Sbert, and Laszlo Szirmay-Kalos. Combined Correlated and Importance Sampling in Direct Light Source Computation and Environment Mapping. *Computer Graphics Forum*, 23(3), 2004.

[104] László Szirmay-Kalos, György Antal, and Balázs Benedek. Global illumination animation with random radiance representation. In *Proceedings of the 14th Eurographics Workshop on Rendering*, EGRW '03, pages 64–73, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[105] Lszl Szirmay-Kalos, Gyrgy Antal, and Mateu Sbert. Go with the winners strategy in path tracing. *Journal of WSCG*, 13(2):49–56, 2005.

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

[107] Eric Veach and Leonidas Guibas. Bidirectional Estimators for Light Transport. 1994.

[108] 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.

[109] 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.

[110] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)*, 33(4), aug 2014.

[111] Jiří Vorba and Jaroslav Křivánek. Adjoint-driven russian roulette and splitting in light transport simulation. *ACM Trans. Graph.*, 35(4):1–11, July 2016. (accepted for publication).

[112] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph.*, 22(4):85–92, June 1988.

[113] A. Wilkie, S. Nawaz, M. Droske, A. Weidlich, and J. Hanika. Hero wavelength spectral sampling. *Comput. Graph. Forum*, 33(4):123–131, July 2014.

[114] Henning Zimmer, Fabrice Rousselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. Path-space motion estimation and decomposition for robust animation filtering. *Computer Graphics Forum (Proceedings of EGSR)*, 34(4), June 2015.

[115] Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, B Moon, Ravi Ramamoorthi, Fabrice Rousselle, P Sen, Cyril Soler, and S Yoon. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum*, 34(2):13, 2015.

Erklärung

gemäss Art. 28 Abs. 2 RSL 05

Name/Vorname:	Marco Manzi
Matrikelnummer:	05-113-022
Studiengang:	Computer Science

Bachelor ☐ Master ☐ [✓ Dissertation]

Titel der Arbeit:	Advanced Techniques in Gradient-Domain Rendering
LeiterIn der Arbeit:	Prof. Dr. Matthias Zwicker

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe r des Gesetzes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist. Ich gewähre hiermit Einsicht in diese Arbeit.

Ort/Datum

Unterschrift

Curriculum Vitae: Marco Manzi

Education

06/2012-10/2016	**PhD Candidate in Computer Science**, University of Bern, Switzerland, Computer Graphics Group. Thesis: *Advanced Techniques in Gradient-Domain Rendering*. Advisor: Matthias Zwicker
10/2010-05/2012	**Master in Computer Science**, Universities of Bern, Neuchâtel and Fribourg, Switzerland. Thesis: *Adaptive Sampling and Reconstruction for Interactive Ray Tracing*. Advisor: Matthias Zwicker.
10/2006-10/2010	**Bachelor in Computer Science,** University of Bern, Switzerland. Minor: Mathematics, Philosophy, Political Sciences Thesis: *Soft Shadows in Real-Time Applications*. Advisor: Matthias Zwicker

Professional Experience

03/2016-05/2016	**Rendering Intern,** Weta Digital Ltd., Wellington, New Zealand**.**
06/2012-02/2016 06/2016-10/2016	**Research Assistant,** University of Bern, Computer Graphics Group.
09/2011-05/2012	**Student Advisor,** University of Bern.
02/2009-05/2012	**Teaching Assistant,** University of Bern.