KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING INFORMATICA
Celestijnenlaan 200A – B-3001 Leuven – België

ON ROBUST MONTE CARLO ALGORITHMS FOR MULTI-PASS GLOBAL ILLUMINATION

Promotor:
Prof. Dr. ir. Yves D. Willems

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Frank SUYKENS - DE LAET

September 2002

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING INFORMATICA
Celestijnenlaan 200A – B-3001 Leuven – België

ON ROBUST MONTE CARLO ALGORITHMS FOR MULTI-PASS GLOBAL ILLUMINATION

Committee:
Prof. Dr. ir. Jean Berlamont, chairman
Prof. Dr. ir. Yves D. Willems, advisor
Prof. Dr. ir. Ronald Cools
Prof. Dr. ir. Philip Dutré
Prof. Dr. ir. Paul Suetens
Prof. Dr. ir. Marc Van Barel
Prof. Dr.-Ing. Philipp Slusallek (Universität des Saarlandes)
Dr. Per H. Christensen (Pixar Animation Studios)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Frank SUYKENS - DE LAET

U.D.C. 681.3*I3

September 2002

D/7515/2002/37

ISBN 90-5682-366-3

On robust Monte Carlo algorithms for multi-pass global illumination

Frank Suykens - De Laet

Department of Computer Science,
Katholieke Universiteit Leuven

ABSTRACT

Computer-generated realistic images are being used increasingly in applications such as architecture, lighting design and visual effects in the movie industry. Therefore, realistic image synthesis, the subject of this dissertation, has remained one of the most important research areas in computer graphics for many years.

Realistic image synthesis takes as input a description of a three-dimensional scene and a virtual camera looking upon that scene. By simulating the light transport, the intensity of each pixel in the image as seen by the camera can be computed. Light transport can be described by an integral equation, often called the *rendering equation*. Algorithms that compute all possible interreflections of light in the scene and thus provide a full solution to the rendering equation, are called *global illumination* algorithms. Such algorithms produce images that are virtually indistinguishable from a real photograph, but unfortunately they are slow.

In this dissertation we present several new techniques to construct more robust and more efficient global illumination algorithms. Our work focuses on *Monte Carlo algorithms* and *multi-pass methods*. Monte Carlo algorithms statistically estimate integrals and are very well suited for solving transport problems. Multi-pass methods combine several rendering algorithms into a single method. A good multi-pass configuration will preserve the strengths of each individual component, while avoiding its weaknesses.

First, new techniques for designing better multi-pass methods are presented, including weighted multi-pass methods that weight the contributions of different algorithms automatically in a provably good manner. Second, path differentials are introduced. They allow to augment light transport paths with neighborhood information which is useful in many global illumination algorithms. Finally a technique is presented to reduce the memory requirements and to control the error in photon mapping, an efficient and popular global illumination algorithm.

All these techniques result in more robust and efficient global illumination algorithms and contribute to the increasing use of these algorithms in computer graphics applications.

The research that led to this dissertation has been supported by a grant from the Fund for Scientific Research – Flanders (F.W.O.. grant #G.0263.97) and by the Flemish Institute for the Promotion of Scientific-Technological Research in Industry (I.W.T. ITA-II/980302).

Acknowledgements

Deciding to start a Ph.D. or *doctoraat* as we say in Dutch, is not as easy as it may seem, and neither is finishing it. Now that this work is almost completed, it strikes me that so many people have contributed in one way or another to this dissertation. Although a single page of acknowledgements can never do them justice, I will still try my best.

First of all, I would like to thank my advisor, prof. Willems, who, despite his honest and realistic view on getting a Ph.D., still convinced me to come back to Leuven. He made it possible for me to do some very interesting research in the graphics group, and supported me throughout the whole process.

Many thanks must go to prof. J. Berlamont, prof. R. Cools, prof. Ph. Dutré, prof. P. Suetens, prof. M. Van Barel, prof. Ph. Slusallek and Per Christensen for kindly accepting to be part of my evaluation committee and for their encouraging comments on the text.

I am particularly pleased that Per is in my committee, as he has been a tremendous source for ideas and discussions, and because he is probably one of the friendliest researchers around.

My current and former colleagues in the graphics group —Phil Dutré, Philippe Bekaert, Vincent Masselus, Pieter Peers, Fre Anrys, and Karl vom Berge— have always provided a very enjoyable atmosphere to work in. Phil taught me to go back to the rendering equation and start over whenever you get stuck. Philippe has been a tremendous help with getting my own research topics going, not in the least by his extraordinary programming skills that initiated RenderPark and XRML. Vincent turned out to be a very useful modeler around deadlines, and I owe him for several scenes in this dissertation. Also, I am most grateful to Pieter, who carefully read an early draft of the text, and who provided indispensable feedback including many corrections to the formulas.

I am indebted to many other people from the computer science department, for keeping our computer systems up and running, for handling all kinds of administration, for help, for ideas, or just for a nice chat.

I will always remember the many graphics conferences, especially the rendering workshop, for being interesting and fun. Many people that I met there, provided me with interesting ideas, comments, and beer. I especially would like to thank Henrik Wann Jensen, who has always supported and actively promoted my work. He made it possible for me to present my work at SIGGRAPH, for which I am most grateful. I also would like to thank Jan Přikryl for hosting me several times in Prague, and Piero Foscari for comments on the text.

I cannot thank enough my family (in particular, my parents, Luc and Paula, and my sister, Liesbet) and my friends (Andy, Kathleen F., Wim, Rob, BC. Zoersel, and many others), who always supported me, who always showed genuine interest when I tried to explain what exactly I was working on, and who gave me many other interesting things to do besides work.

Of course, I cannot finish without thanking Kathleen, whose unconditional support (and patience) helped me when I most needed it.

Thanks to all of you for reading this. I hope you will read on…

Frank Suykens,

Heverlee – August 2002.

To Kathleen

Contents

Notations and abbreviations

Abbreviations

BPT	bidirectional path tracing
cdf	cumulative distribution function
MIS	multiple importance sampling
MPC	multi-pass configuration
NN	nearest neighbor
pdf	probability density function
SPAR	stored partial radiance solution

Geometry

A	finite area
$\mathrm{d}A$	differential area
A_s	total scene area
A_{pix}	area of a pixel
A_{scr}	area of the screen (image plane)
ω	direction (unit vector)
ω_i	incoming or incident direction
ω_o	outgoing or exitant direction
Ω	finite solid angle
$\Omega_{2\pi}$	hemisphere (of directions)
$\Omega_{4\pi}$	unit sphere (all possible directions)
$\mathrm{d}\omega$	solid angle measure
$\mathrm{d}_\perp\omega = \cos\theta\,\mathrm{d}\omega$	projected solid angle measure
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	3D points in the scene, path vertices
$\overline{\mathbf{x}} = \mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_n$	a path with n vertices
$\overline{\Omega}$	path space
$\mathbf{y} = \mathrm{rc}(\mathbf{x} \to \omega_i)$	ray casting function
$V(\mathbf{x}, \mathbf{y})$	visibility term
$G(\mathbf{x}, \mathbf{y})$	geometry term ($V(\mathbf{x}, \mathbf{y}) \frac{\lvert\cos(\theta_i)\cos(\theta_o)\rvert}{\lVert\mathbf{x}-\mathbf{y}\rVert^2}$)

Probability and Monte Carlo

$E[\xi], E[f(\xi)]$	expected value of a random variable ξ or $f(\xi)$
$V[\xi]$	variance of ξ
σ	standard deviation
β	bias
`eff`	estimator efficiency
Prob$\{X\}$	probability of an event X
$\langle I \rangle_N$	N-sample estimator of I
$p(x)$	probability density function
$P(x)$	cumulative distribution function

Radiometry

Φ	flux
L	radiance (outgoing)
L_e	self-emitted radiance
L_i	incoming radiance
L_r	scattered or reflected radiance
E	irradiance
B	radiosity
B_e	self-emitted radiosity
$F_{i,j}$	form factor between two patches i and j
W	directional importance (or potential)
W_e	self-emitted directional importance
W_i	incoming directional importance
Γ	importance ($\sim$ irradiance)
Ψ	importance flux ($\sim$ flux)
f_r	bidirectional reflection distribution function (BRDF)
f_t	bidirectional transmission distribution function (BTDF)
f_s	bidirectional scattering distribution function (BSDF)
ρ	albedo
$\overline{f}(\bar{\mathbf{x}})$	path contribution function

Regular expression notation

E	eye or camera vertex
L	light source vertex
$\mathsf{D_r}, \mathsf{D_t}$	diffuse reflection/transmission
$\mathsf{G_r}, \mathsf{G_t}$	glossy reflection/transmission
$\mathsf{S_r}, \mathsf{S_t}$	specular reflection/transmission
$\mathsf{S_{r_\infty}}, \mathsf{S_{t_\infty}}$	perfectly specular reflection/transmission
$\vert$	'or' operator
$+$	addition
D	$\mathsf{D_r}\vert\mathsf{D_t}$
G	$\mathsf{G_r}\vert\mathsf{G_t}$
S	$\mathsf{S_r}\vert\mathsf{S_t}$
X	$\mathsf{D}\vert\mathsf{G}\vert\mathsf{S}$
X^*	zero or more occurrences of X
X^+	one or more occurrences of X
X^k	exactly k occurrences of X
$\mathsf{X}^{0..k}$	0 to k occurrences of X

Miscellaneous

I	A general measurement
$\langle f, g \rangle$	inner product of functions f and g
$\mathrm{J}\left(\frac{f(\mathbf{u})}{\mathbf{u}}\right)$	Jacobian matrix (where $\mathbf{u}$ is a multi-dimensional vector)
$\vert\mathbf{M}\vert$	determinant of a matrix $\mathbf{M}$
$\Phi(\mathbf{x}, \omega, \phi)$	a photon with position $\mathbf{x}$, incoming direction ω, and power ϕ
N_Φ	the number of photons or particles (which can be larger than the number of light paths traced (N))
N_{pix}	the number of pixels in an image
ε	error symbol

1 Introduction

This dissertation contributes to the field of computer graphics, and more specifically realistic image synthesis. New techniques and algorithms to compute the physically based light transport in a scene are presented; techniques that improve on previous work both in terms of efficiency and robustness of the algorithms.

In this introductory chapter, we will discuss the importance and difficulty of realistic image synthesis (§1.1) and physically based light transport (§1.2) in the broader context of computer graphics and its applications. Different approaches to simulating light transport are summarized in §1.3. The specific goals and contributions of this dissertation are stated in §1.4. Section 1.5 outlines the organization of the text and concludes this chapter.

1.1 Realistic image synthesis

The quest for visual realism has played an important role in computer graphics research over the last 30 years. Computer-generated images evolved from simple line drawings to photo-realistic images. Photo-realistic image synthesis aims at the generation of images by computer from a description of a three-dimensional scene, and this in such a way that the resulting rendered images appear to an observer as if they were real photographs.

Many applications benefit or even depend on the creation of realistic images. Special effects in movies, pilot training, the flourishing market of computer games, advertising, art and architecture all leverage the latest advancements in realistic rendering.

In many applications, the actual rendering algorithms used in production are based on *local illumination*: the appearance of objects is only determined by the illumination emitted directly by a light source. Indirect illumination received from the other objects is ignored, and this imposes a severe limitation on the realism of the generated images. Effects such as the soft indirect light caused by a designer lamp illuminating the ceiling, the color bleeding between differently colored walls, the hazy refraction through a frosted glass pane, or the caustics cast by glass objects are all not reproduced automatically. A local illumination model places the responsibility of creating a naturally looking illumination in a scene on the modeler, who often has to carefully place many additional lights to simulate the indirect illumination.

Global illumination algorithms do compute all the possible light interreflections in a scene and reproduce the aforementioned effects automatically. However, the global illumination problem is inherently much harder, and reliable algorithms are slow. This is why the simpler local illumination model has prevailed for such a long time. Nevertheless, times are changing. For example in August 2001, Kaveh Kardan,

R&D Manager at Square USA (where the realistic, computer generated motion picture 'Final Fantasy, The Spirits Within' was made), stated in an interview by Ars Technica: *"I believe that global illumination will become the norm. Direct illumination renderers like PRMan (Pixar's PhotoRealistic RenderMan) have reached the limit of the realism they can produce in terms of lighting.*" Although global illumination algorithms start to find their way into production renderers, there is still a great need for more efficient, more reliable and robust algorithms.

The work in this thesis is concerned with *physically based* image synthesis. Physically based image synthesis adds an additional requirement to just photo-realism or global illumination: the computations should be based on physical principles. This leads to quantitative results that are predictive: the computer generated images accurately predict the appearance of objects, without the need to actually build them first. The predictive nature of physically based image synthesis opens up a whole new set of applications, including lighting design, lighting optimization, architectural visualizations, and product design in general.

Physically based image synthesis leads to the ultimate realism by computing an accurate simulation of the light transport in a scene. Developing robust light transport algorithms that adapt well to different lighting conditions and complex scenes, is a difficult problem. It seems obvious, however, that the better and faster the algorithms become, the more they will be used. Given the improvement of both computational power and the added efficiency of new light transport techniques, it is to be expected that, in the near future, all photo-realistic rendering will be done using physically based light transport algorithms.

In the next few sections we will analyze the problem and solution of physically based light transport in more detail and indicate our contributions to this field.

1.2 Physically based light transport

The input to a physically based light transport simulation is a full description of the scene to be rendered. This description includes the 3D geometry, the material properties, the emission characteristics of the light sources and a virtual camera. The goal of the simulation is then to compute a realistic image of the scene, as seen by the virtual camera.

In physically based light transport, 'light' is represented by actual radiometric quantities. An important quantity is *radiance*, that can be defined intuitively as the intensity of an infinitely small beam of light. It is expressed as the energy per unit area, solid angle, wavelength and time (see chapter 2 for a more detailed description of radiometric quantities).

Computing an image requires the estimation of the average radiance that reaches each pixel in the image plane, which is done by simulating the transport of light from the light sources to the camera. The light transport itself is described by a mathematical model that defines how light is emitted and how it interacts with the surfaces in the scene. Such a model can be based on geometric optics, wave optics or even quantum

optics.

For graphics applications, a model based on geometric optics is usually sufficient. The rendering equation, proposed by Kajiya [54], is the prevalent model in computer graphics. Radiance is assumed to travel in straight lines and only interacts with the scene on the surfaces. With these assumptions, the radiance in the scene is expressed by an integral equation, a Fredholm equation of the second kind, that states the outgoing radiance on a surface as a function of the self-emitted radiance and the radiance incident on the surface. Of course, the incident radiance originates as outgoing radiance leaving from other surfaces, which reveals the recursive nature of this equation. All research into global illumination aims at efficiently solving this integral equation for the widest possible variety of scenes.

Some lighting effects, however, are not directly modeled by the rendering equation. This includes participating media, such as smoke or fire, and several effects that cannot be described by geometric optics alone, such as diffraction, polarization, fluorescence and interference. These effects can usually be included in a simulation by an extension of the mathematical model used.

Some global illumination algorithms make further simplifications to the model, for example by assuming all surfaces to be perfectly diffuse, by restricting the geometry to polygonal surfaces only, or by limiting non-diffuse scattering to perfectly specular reflection or refraction.

In this work we aim to solve the rendering equation in its full generality. No restrictions are placed on the surface characteristics or the geometry, and a full global illumination solution is required. (A detailed mathematical description of our light transport model is given in chapter 2.)

1.3 Algorithms for simulating light transport

Many algorithms for solving the global illumination problem have been proposed. In this section we overview some of the main characteristics of the different approaches.

1.3.1 Object-space versus image-space

Object-space algorithms An object-space approach computes a representation of the radiance within the scene itself, independent of any virtual camera.

For example, classical radiosity algorithms compute a constant diffuse radiance value for each (polygonal) element in the scene. Another example is a photon map, where photons or particles are emitted from the light sources and stored individually by recording their position and incident direction.

To compute an image, the average radiance through each pixel is estimated by reconstructing the radiance from the object-space solution on the visible surfaces.

The main advantage of object-space approaches is the view-independence of the solution: it can be used for many different views (although the accuracy is seldom high enough for special cases such as an

extreme close-up). The main difficulty, however, is the huge memory requirement. All algorithms have to make simplifying assumptions about the light transport model in order to reduce the memory requirements to an acceptable level. For example, the radiance reflected by highly specular objects is never included in an object-space solution.

Image-space algorithms An image-space algorithm directly computes pixel values, without first storing illumination information in the scene. Typical examples are ray tracing and path tracing, that trace paths from the camera through the pixels into the scene to gather the illumination.

Image-space algorithms only require a small amount of memory and adapt well to complex scenes. Since no illumination is stored, however, the same illumination may have to be computed over and over again. For example, the illumination of a surface that is directly visible as well as through a mirror reflection, is computed two times independently.

Multi-pass algorithms The most successful global illumination approaches combine object-space and image-space algorithms into a multi-pass method. The total light transport is separated in a number of disjunct parts, and each part is handled by a different algorithm. Typically diffuse-like illumination, which can be stored using a reasonable amount of memory, is computed in object-space. An image-space pass will then compute the other illumination but will also use the previously stored solution as much as possible.

The key to a good multi-pass algorithm is an intelligent separation of illumination and a clever combination of algorithms that minimizes memory consumption, computation time and error. A large part of this dissertation is focused towards deriving good multi-pass algorithms. We will present technical and theoretical tools that allow the construction of better multi-pass methods. Applying these techniques to current multi-pass methods, results in a significant improvement of these methods.

1.3.2 Monte Carlo algorithms

Global illumination algorithms can also be classified into deterministic and stochastic algorithms.

Deterministic algorithms use standard cubature rules to approximate the integrals in the rendering equation. Well known examples are deterministic radiosity algorithms: The rendering equation is discretized into a set of linear equations, and the form factors, 4D integrals that determine the energy transfer between a pair of elements, are computed with deterministic integration rules.

Stochastic or Monte Carlo algorithms estimate integrals by averaging a high number of statistical trials. They are very well suited for estimating the high-dimensional integrals that occur in global illumination. They adapt well to a complex geometry and are not limited to polygonal scenes.

Examples of Monte Carlo rendering algorithms range from image-space methods, such as path tracing or bidirectional path tracing, to object-space methods, such as Monte Carlo radiosity and several variations

of particle tracing.

Over the years it has become clear that Monte Carlo algorithms are faster, more robust, and more versatile than their deterministic counterparts. Therefore, we mainly use Monte Carlo algorithms in this work.

Unbiased versus consistent

An important property of Monte Carlo estimators is whether they are unbiased or not.

A Monte Carlo estimator is *unbiased* when its expected value is equal to the exact value of the estimated integral. The error in an unbiased estimator is only caused by statistical fluctuations around the exact solution. In computed images this error is perceived as noise. This noise slowly disappears with an increasing number of samples (i.e., as the square root of the number of samples). This slow convergence is the biggest drawback of Monte Carlo algorithms. Difficult illumination features can require a very large number of samples before the noise drops below an acceptable level.

The expected value of a *biased* Monte Carlo algorithm differs from the correct solution. Biased estimators are only useful when the systematic error, the bias, can be estimated or when it is known that the bias also disappears with an increasing number of samples. In the latter case, the estimator is said to be *consistent*. Consistent estimators can be much more efficient than unbiased estimators. Although the exact error is usually hard to estimate, the noise reduction that can be obtained with certain biased but consistent methods, makes their use worthwhile.

In this work we focus on Monte Carlo algorithms, because of their generality, robustness and ease of use. We will consider both unbiased and consistent estimators, and offer some techniques to trade noise for bias in order to get a better, visually pleasing solution.

1.4 Contributions of this work

In this section we will preview the main contributions of this dissertation to the global illumination research.

We have developed general techniques rather than proposing a single global illumination algorithm. Depending on the specific application, different global illumination methods may be preferable. Most of our contributions consist of versatile tools that can be plugged into many existing global illumination algorithms, so that their scope is not limited to one specific algorithm.

Our techniques are targeted towards both Monte Carlo and multi-pass algorithms, as they have proven to be the most efficient and robust global illumination algorithms.

More specifically our main contributions are the following:

- **Regular expression based path evaluation:** This technical tool helps the design of more advanced multi-pass configurations. Regular expressions are an easy way to specify a specific part of the

light transport. This technique derives the evaluation or contribution of individual light transport paths directly from the user-defined regular expressions. This provides a convenient way to specify complex multi-pass configurations with a high degree of separation.

- **Weighted multi-pass methods**: All existing multi-pass methods employ a perfect separation strategy: each part of the light transport is assigned to exactly one algorithm or combination of algorithms.

  Weighted multi-pass methods provide a new and general theoretical framework that allows overlapping transport in a multi-pass configuration. To ensure a correct solution, where all transport is only accounted for once, weighting is used instead of separation. Good weighting heuristics automatically preserve the strengths of the individual methods within the overlapping transport, without the need of an extreme separation.

  Underneath the weighted multi-pass methods lies a new Monte Carlo variance reduction technique that allows the combination of several sampling techniques of different dimensions, extending standard multiple importance sampling [116].

- **Path differentials:** Many global illumination algorithms trace paths through the scene. These paths depend on a number of generating variables that form a single point sample in path space. However, such a point sample does not provide any information about the neighborhood of a path: what is the region of influence of a vertex in a path or how far away are neighboring paths?

  This is exactly the information that is provided by path differentials. By computing partial derivatives of the directions and vertices in a path (with respect to the variables), and by estimating a small neighborhood around the sample point in path space, the region of influence or *footprint* of a path can be estimated in each path vertex.

  Such footprint information can be used in many global illumination applications. We have applied it to local texture filtering, hierarchical particle tracing radiosity and importance computations, but many more applications are possible. In all applications, the path differentials provide a good trade-off between bias and noise.

  Specific contributions are the computation of the partial derivatives for arbitrary Monte Carlo sampled paths, heuristics for the determination of a neighborhood that ensures coherence over the footprint, a convenient representation of the footprint and all the applications.

- **Density control for photon maps:** The final contribution is targeted towards a specific multi-pass algorithm named photon mapping. Photon mapping is an efficient and robust two-pass algorithm for computing global illumination. A first, object-space pass emits photons from the light sources and stores them all individually in a photon map. A second, image-space pass renders an image using the

information in the photon maps.

Storing all individual photons in the photon map requires a lot of memory. We present a density control framework that selectively stores photons until a certain target density criterion is met. By using a target density criterion that is based on the current viewing position, photons are concentrated in important parts of the scene.

Results of the density control show a significant decrease in the number of stored photons, and maybe even more important, the framework leads the way to fully error-controlled photon mapping.

1.5 Organization of the dissertation

The remainder of the text is organized as follows:

- Chapters 2, 3 and 4 are introductory.
  - Chapter 2 summarizes the specific mathematical model for light transport that is used in this dissertation. It also introduces the necessary geometrical and radiometrical concepts and notation.
  - Chapter 3 provides a brief introduction to Monte Carlo integration and Monte Carlo rendering. Basic concepts, variance reduction techniques, and some important Monte Carlo rendering algorithms that are frequently used throughout the text, are briefly reviewed.
  - Chapter 4 introduces multi-pass methods, a regular expression notation for light transport paths, and an overview of existing multi-pass configurations.

  Readers who are well familiar with Monte Carlo rendering and multi-pass methods, might want to just skim through the chapters to pick up the specific notation and conventions used in the text.
- Chapter 5 describes the regular expression based path evaluation. Its use is demonstrated by showing that a combination of bidirectional path tracing and a radiosity algorithm is easy with this tool.
- Chapter 6 presents the weighted multi-pass methods. The theory is explained in a general Monte Carlo setting and then applied to the combination of bidirectional path tracing and radiosity.
- Path differentials are developed in chapter 7. Besides the general theory and implementation details, two applications are presented: local texture filtering and hierarchical particle tracing radiosity. A third application is given as part of the density control framework for photon maps in chapter 9.
- In chapter 8, a general overview of photon mapping is given. An analysis of the strengths and difficulties is given, along with a number of important optimizations to make photon mapping efficient.

- The density control framework for photon mapping is described in chapter 9. This includes a redistribution technique to account for the power of unstored photons, an importance driven target density heuristic and several results.

- Conclusions are presented in chapter 10, along with some global directions for future research. More specific extensions of the techniques themselves are given in conclusions at the end of each chapter.

All the methods presented in this thesis were implemented in RENDERPARK [9], an extensive global illumination software package that is being developed together with Philippe Bekaert. All results and all images were generated with RENDERPARK. Its source code is freely available from "`www.renderpark.be`" and forms an interesting addition to this dissertation.

2 A mathematical model for light transport

2.1 Introduction

Physically based image synthesis requires the simulation of light transport in a scene. Computing such a simulation requires a well defined mathematical model for light transport. The model must specify the complete light transport problem, and should include the following aspects:

- A specification of the scene to be rendered: This includes a description of the geometry, the material properties and the light sources.
- The radiometric quantities used to describe the light transport.
- The transport model itself that describes the scattering and propagation of light.
- A specification of the measurements that need to be computed. For image synthesis, this requires a virtual camera model that relates pixel values to the light incident on the image plane.

The model used in this dissertation is based on the rendering equation. This model was first presented by Kajiya in 1986 [54]. It was a general mathematical model that encompassed all previous —more ad hoc— approaches to realistic image synthesis. Once the problem was cast into a clear mathematical model, many existing numerical techniques could be applied to solve it, and this is what happened over the years. Monte Carlo and finite element techniques were applied to solve the equation more efficiently, while others extended the model to include effects such as participating media, dispersion, and diffraction.

The rendering equation, however, was not really new. Light transfer is a transport problem governed by equations similar to those encountered in radiative heat transfer and neutron transport. Rendering research has borrowed and continues to borrow many ideas and techniques from these and other fields and adapted them to the typical needs of computer graphics: photo-realistic images.

Overview

This chapter summarizes the mathematical model used in this dissertation and establishes notation.

The scene geometry and some important geometrical units are defined in §2.2. Radiometric quantities are given in §2.3, and material models for the surfaces in the scene are defined in §2.4. Given these preliminaries, the actual transport model, the rendering equation, can be defined (§2.5). Finally, the measurement equation and a simple pinhole camera model define how pixel values can be estimated (§2.6).

The remaining sections provide some alternative formulations of the light transport model. Adjoint equations (§2.7) reformulate the light transport problem from a measurement viewpoint. Operators (§2.8) and the path integral formulation (§2.9) can provide convenient shorthand notations of the light transport model.

We have tried to restrict our description of the light transport model to a bare minimum. Only the things that will be needed further on, are described in some detail. There are many other works on global illumination that contain a much more elaborate discussion on mathematical models for light transport. For more information the reader is referred to the work by Arvo [4], Cohen and Wallace [22], Dutré [27], Lafortune [60] or the thorough treatment in the PhD dissertation of Eric Veach [114].

2.2 Geometry

This section defines some common geometrical entities used in the light transport formulations and algorithms.

2.2.1 Scene geometry

The *scene geometry* consists of a number of 2D surfaces, curved or planar, that can scatter, emit, or absorb light. Apart from its geometrical extent, each surface requires an intersection operation for ray tracing, normal vectors, and a parameterization (i.e., a mapping from the unit square to the surface) that is used for illumination storage and texture mapping.

A point on a surface is given by a position vector (notation: $\mathbf{x}, \mathbf{y}$). Associated with each point is a differential area $\mathrm{d}A$. The *total area* of the union of all surfaces in the scene will be denoted by A_s (with an 's' from 'scene').

2.2.2 Directions and solid angles

A *direction* is represented by a unit vector ω. The set of all possible directions is the unit sphere $\Omega_{4\pi}$, that has a solid angle of 4π steradian. A hemisphere $\Omega_{2\pi}$ covers 2π steradian. Associated with a direction is the differential solid angle, $\mathrm{d}\omega$, that is used for integration over finite solid angles.

Another interesting measure for integration over solid angles is the projected solid angle. The differential projected solid angle $\mathrm{d}_\perp\omega$ is defined as

$$\mathrm{d}_\perp\omega(\mathbf{x}) = \cos\theta(\mathbf{x})\,\mathrm{d}\omega(\mathbf{x}),$$

with $\cos\theta(\mathbf{x}) = \mathbf{N}_\mathbf{x} \cdot \omega$, and $\mathbf{N}_\mathbf{x}$ the normal in $\mathbf{x}$, a surface point in the scene. Further on, the argument $(\mathbf{x})$ will be dropped whenever it is clear which point is meant. The use of the projected solid angle simplifies many of the light transport equations.

2.2.3 Paths

In many global illumination algorithms paths are traced through the scene.

A *path vertex* $\mathbf{x}$ is a point on one of the surfaces in the scene (including the camera or any other sensor surface). The domain of a vertex covers the total scene area A_s.

A *path* $\bar{\mathbf{x}}$ is defined as a collection of vertices $\bar{\mathbf{x}} = \mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_n$. The *length* of the path is the number of vertices n. *Path space* $\overline{\Omega}$ is the domain containing all possible paths of arbitrary length. $\overline{\Omega}_n$ denotes the space of all possible paths of length n: It is the product space of the domain of each path vertex: $\overline{\Omega}_n = A_s \times \ldots \times A_s = A_s^n$.

2.3 Radiometry

This work deals with physically based rendering. Instead of ad hoc quantities such as 'light intensity' and colors that lie between 0 and 1, used for example in OpenGL, real radiometric quantities are used.

The following table gives an overview of the necessary quantities and units. The definitions of the quantities are expressed in terms of flux, that is taken as the base quantity.

Quantity	Symbol	Definition	Unit
Flux	Φ	Φ	Watt (W)
Radiance	L	$\frac{\mathrm{d}^2\Phi}{\mathrm{d}_\perp\omega\,\mathrm{d}A}$	W/m^2sr
Irradiance	E	$\frac{\mathrm{d}\Phi}{\mathrm{d}A} = \int_{\Omega_{2\pi}} L_\mathrm{i}\,\mathrm{d}_\perp\omega$	W/m^2
Radiosity	B	$\frac{\mathrm{d}\Phi}{\mathrm{d}A} = \int_{\Omega_{2\pi}} L\,\mathrm{d}_\perp\omega$	W/m^2

L_i refers to incoming or incident radiance, while L is outgoing or exitant radiance, i.e., radiance leaving the surface.

These radiometric quantities also depend on the wavelength of light λ, given in *nm*. We will not explicitly distinguish between the different wavelengths, but consider the quantities as vectors with a number of components dependent on the wavelength:

$$L = [L_{\lambda_1}\ldots L_{\lambda_k}]^\top.$$

Most of the time in graphics, also in our implementation, only three components (red, green, and blue) are used. The algorithms presented in this text apply without change to any number of wavelengths.

2.4 Material properties

The material properties define how a surface reflects, refracts, and emits light. The common abstraction for reflection and refraction used in graphics is the bidirectional scattering distribution function (BSDF),

defined and discussed in the next section. Light sources that emit radiance are modeled by an emission distribution function (EDF), defined in §2.4.2.

2.4.1 Reflection and refraction

Lets consider purely reflective surfaces first. The bidirectional reflection distribution function (BRDF, f_r) for a surface point $\mathbf{x}$ is defined as the ratio of the reflected radiance and the differential irradiance for an incident direction ω_i:

$$f_r(\mathbf{x}, \omega_i \rightarrow \omega_o) = \frac{L(\mathbf{x} \rightarrow \omega_o)}{L_i(\mathbf{x} \leftarrow \omega_i)\, \mathrm{d}_\perp \omega_i} .$$

The BRDF depends on both the incoming and outgoing direction. For a fixed point $\mathbf{x}$, the BRDF is a 4-dimensional function.

For physically based BRDFs the Helmholtz reciprocity principle holds, which allows interchanging the directions:

$$f_r(\mathbf{x}, \omega_i \rightarrow \omega_o) = f_r(\mathbf{x}, \omega_o \rightarrow \omega_i) .$$

A sampled representation of the 4D BRDF can be measured from real surfaces using, for example, a gonio-reflectometer. Usually such a sampled representation is converted into a simpler, empirical BRDF model such as the Ward [123], Lafortune [61], or (modified) Phong [69, 64] model. Other models exist that are based directly on the underlying surface physics. Their input consists of real physical parameters. The most comprehensive is the He-Torrance-Sillion-Greenberg model [36].

For transmissive surfaces a similar *bidirectional transmission distribution function* (BTDF, f_t)[1] can be defined. It is harder to measure and usually only perfectly specular refraction is considered, or else empirical models are tuned with handpicked parameters.

For general rendering algorithms that can handle both reflection and refraction, it is convenient to combine the BRDF and BTDF into the *bidirectional scattering distribution function* (BSDF, f_s). The BSDF is defined over the full hemisphere for both incoming and outgoing directions, so in fact it is a combination of 2 BRDF-BTDF pairs, one pair for each side of the surface.

Some interesting properties of BSDFs will be discussed in the next paragraphs.

2.4.1.1 Albedo

The *directional hemispherical reflectance* or *albedo* (ρ_r) of a BRDF is the fraction of the incoming radiance (for a single direction) that is reflected over the hemisphere:

$$\rho_r(\mathbf{x}, \omega_i) = \int_{\Omega_{2\pi}} f_r(\mathbf{x}, \omega_i \leftarrow \omega_o)\, \mathrm{d}_\perp \omega_o .$$

[1] Originally the BRDF f_r included subsurface scattering effects. The reflected radiance could emanate from a different position, and the distance to $\mathbf{x}$ was denoted by the r in f_r. In graphics, however, the BRDF does not include subsurface scattering, rendering r meaningless ($r \equiv 0$). Reassigning the meaning of r to be 'reflection', allows convenient notation for transmission f_t and general scattering f_s.

The *transmittance* ρ_t is defined similarly. The total albedo $\rho_s = \rho_r + \rho_t$ must be smaller than 1 to conserve energy. The *absorption* of a surface is given by $1 - \rho_s$.

Note that the albedo changes with the incoming direction ω_i. Many BRDF or BSDF models use a single, approximate albedo parameter to control the reflectance of a surface. For example, the modified Phong model uses the albedo for an incoming direction perpendicular to the surface.

2.4.1.2 BSDF components

BSDFs are often characterized by the directionality of the light scattering. We will distinguish between *diffuse*, *glossy* and *specular* reflection and transmission:

- **Diffuse reflection/transmission ($\mathsf{D}_r, \mathsf{D}_t$):** The BRDF or BTDF is constant. The scattered radiance is equal for all outgoing directions.
- **Specular reflection/transmission ($\mathsf{S}_r, \mathsf{S}_t$):** Very strong directional scattering: Most of the light is scattered over a very small solid angle. For perfect specular scattering, the limiting case, all light is scattered towards a single direction. In this case the BSDF is a delta function.
- **Glossy reflection/transmission ($\mathsf{G}_r, \mathsf{G}_t$):** Glossy reflection encompasses anything that is not diffuse nor specular. The glossy component exhibits moderate directional scattering: more light is reflected in a restricted part of the hemisphere, usually around the perfectly specular scattering direction.

2.4.2 Light sources

Some surfaces, the light sources, also emit light. A light source is specified by its geometry and an emission distribution function (EDF). The EDF defines the *self-emitted radiance*, $L_e(\mathbf{x} \to \omega)$, emitted from a point $\mathbf{x}$ on a light source into a direction ω.

The *emittance* or self-emitted radiosity is defined as the self-emitted radiance integrated over the hemisphere:

$$B_e(\mathbf{x}) = \int_{\Omega_{2\pi}} L_e(\mathbf{x} \to \omega)\, \mathrm{d}_\perp\omega\,.$$

The *self-emitted flux* for a light source l is given by

$$\Phi_e^{(l)} = \int_{A_l} B_e(\mathbf{x})\, \mathrm{d}A\,,$$

with A_l the surface area of the light source.

2.5 The rendering equation

This section presents the fundamental rendering equation or *radiance transport equation*. This equation describes the light transport in a scene. The radiance transport equation combines the geometry, the EDF,

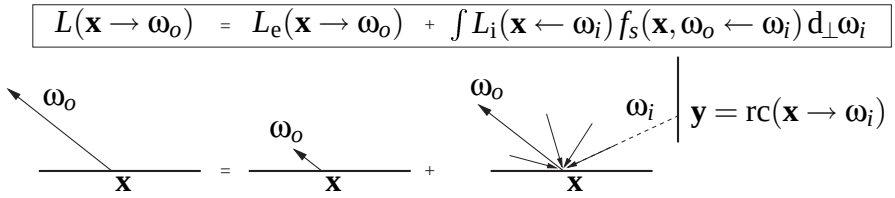


Figure 2.1: The rendering equation expresses the outgoing radiance as a function of the self-emitted radiance and an integral over all the incoming radiance.

and the BSDF definitions into an integral equation that expresses the radiance leaving a surface in a point $\mathbf{x}$ towards the direction ω_o:

$$L(\mathbf{x} \to \omega_o) = L_e(\mathbf{x} \to \omega_o) + \int_{\Omega_{4\pi}} L_i(\mathbf{x} \leftarrow \omega_i) f_s(\mathbf{x}, \omega_o \leftarrow \omega_i) \, d_\perp \omega_i \, . \tag{2.1}$$

The contributions to the outgoing radiance $L(\mathbf{x} \to \omega_o)$ are shown schematically in figure 2.1.

Note that the original rendering equation [54] was given in a slightly different form, but the form given here is more common and uses clearly defined radiometric quantities.

The unknown incoming radiance $L_i(\mathbf{x} \leftarrow \omega_i)$ in the equation originates from outgoing radiance on other surfaces, revealing the recursive nature of the integral equation.

In vacuum, without participating media in the scene, radiance leaving a surface will travel unchanged until it reaches another surface. The relation between the incoming radiance L_i in $\mathbf{x}$ and the outgoing radiance L is then given by the ray casting operation rc:

$$L_i(\mathbf{x} \leftarrow \omega_i) = L(\mathbf{y} \to \omega_i) \quad \text{with} \, \mathbf{y} = \mathrm{rc}(\mathbf{x} \to \omega_i) \, . \tag{2.2}$$

The ray casting operation $\mathbf{y} = \mathrm{rc}(\mathbf{x} \to \omega_i)$ finds the nearest surface intersection $\mathbf{y}$ for a ray shot from $\mathbf{x}$ in the direction ω_i (see also figure 2.1).

Equations (2.1) and (2.2) provide a mathematical model for light transport in a scene: the scattered radiance is computed by the integral in (2.1), in which the unknown incoming radiance is evaluated by casting a ray and evaluating the reflectance equation again.

While this model allows the generation of very realistic images, it is based on geometric optics and does not include all the physical light transport phenomena. Depending on the application it is important to realize that certain effects are not included. Several extensions have been proposed that model missing phenomena such as fluorescence [32], polarization [110], diffraction [104] and participating media [85].

2.6 Measurements

2.6.1 Definition

The rendering equation provides an expression for the radiance leaving a single point on a surface. A *measurement* is defined as the response of a certain sensor that (usually) combines a set of radiance values.

Examples of measurements in image synthesis are the flux through a pixel in an image, the irradiance for a certain surface in the scene, or the coefficients for the basis functions in a higher order radiosity algorithm.

A measurement is defined by a response function $W_e(\mathbf{x} \to \omega)$ that defines the sensitivity of a hypothetical (radiance) sensor placed in the scene. The total response of the sensor gives the measurement I:

$$I = \int_A \int_\Omega W_e(\mathbf{x} \to \omega) L_i(\mathbf{x} \leftarrow \omega) \, \mathrm{d}_\perp\omega \, \mathrm{d}A \, . \tag{2.3}$$

This is called the *measurement equation*.

Note that the response function is defined as an exitant quantity, and that it is combined with the incoming radiance function.

2.6.2 Inner product notation

Often, a measurement is written as an inner product of the response and the radiance function. Let the inner product of two functions $f(\mathbf{x},\omega)$ and $g(\mathbf{x},\omega)$ over a domain $A \times \Omega$ be defined by

$$\langle f, g \rangle = \int_A \int_\Omega f(\mathbf{x},\omega) g(\mathbf{x},\omega) \, \mathrm{d}_\perp\omega \, \mathrm{d}A \, .$$

Using this inner product definition the measurement equation can be written concisely as

$$I = \langle W_e, L_i \rangle \, .$$

2.6.3 Camera model

Computing an image corresponds to a set of specific measurements, one for each pixel. In this case, the response function is determined by the camera model used.

We use a simple pinhole camera model, which assumes that the aperture of the camera is a single point. For a pinhole camera this point is referred to as the 'eye', the 'viewpoint', or the 'camera position'.

The response function for our pinhole camera is given by

$$W_e(\mathbf{x} \to \mathbf{y}) = \begin{cases} 1 \cdot \delta(\mathbf{x} - \text{eye}) & \text{when } \mathbf{y} \in A_{\text{pix}} \\ 0 \quad \text{otherwise.} \end{cases} \tag{2.4}$$

The response function is given in terms of two points in the scene (but can be converted into a standard one in terms of a point and an outgoing direction). The response differs from 0 only when $\mathbf{x}$ is equal to the eye, and when $\mathbf{y}$ is located on the image plane in the pixel under consideration.

The Dirac impulse is needed for a pinhole camera in order to remove the integral over area in the measurement equation. For other camera models it could be replaced by $1/A_{\text{aperture}}$ or even more advanced response functions [59].

The response function (2.4) measures the flux through a pixel. Since the human eye responds to radiance rather than to flux, the pixel flux is usually converted to an average radiance value.

2.7 Adjoint equations

The measurement equation (2.3) suggests that measurements should be computed by recursively applying the radiance transport equation (2.1) to the unknown radiance $L_\mathrm{i}(\mathbf{x} \leftarrow \omega)$. This is the approach taken by path tracing for example (see §3.4.1).

Adjoint methods reverse this approach and apply the transport rules to the responsivity of the sensor. The responsivity of the sensor W_e is called the *emitted directional importance function*, analogous to the emitted radiance L_e. The *importance transport equation* describes the importance transport in the scene:

$$W(\mathbf{x} \rightarrow \omega_o) = W_\mathrm{e}(\mathbf{x} \rightarrow \omega_o) + \int_{\Omega_{4\pi}} W_\mathrm{i}(\mathbf{x} \leftarrow \omega_i)\, f_s(\mathbf{x}, \omega_o \rightarrow \omega_i)\, \mathrm{d}_\perp\omega_i \,. \tag{2.5}$$

This transport equation is the adjoint of the radiance transport equation. Note that in equation (2.5), the direction of the BSDF evaluation is reversed: $(\omega_o \rightarrow \omega_i)$ instead of $(\omega_o \leftarrow \omega_i)$. As such, the BSDF is always evaluated from the light source towards the sensor. For a symmetric BSDF, when $f_s(\mathbf{x}, \omega_o \rightarrow \omega_i) = f_s(\mathbf{x}, \omega_o \leftarrow \omega_i)$, this is not important. But when refraction or interpolated shading normals are involved, the BSDF is not symmetric and the direction of evaluation is important [114, chapter 5].

The measurement equation can now be reformulated in terms of the equilibrium directional importance function:

$$I = \int_A \int_\Omega W_\mathrm{i}(\mathbf{x} \leftarrow \omega) L_\mathrm{e}(\mathbf{x} \rightarrow \omega)\, \mathrm{d}A\, \mathrm{d}_\perp\omega = \langle W_\mathrm{i}, L_\mathrm{e} \rangle \,. \tag{2.6}$$

In the transport and measurement equations we have used incoming and outgoing quantities for radiance (L_i and L) and for directional importance (W_i and W). The self-emitted quantities L_e, W_e are defined as outgoing. Incoming and outgoing quantities can be converted into each other by ray casting or scattering. Measurements are always defined as a combination of an incoming and outgoing quantity

Importance quantities

Just as the directional importance corresponds to radiance, other importance quantities can be defined that correspond to flux or irradiance.

The following table gives an overview of the corresponding quantities used in this text:

Quantity	Symbol	Definition	Corresponding radiometric quantity
Directional importance (or 'potential')	W	W	Radiance
Importance	Γ	$\int_{\Omega_{2\pi}} W_\mathrm{i}\, \mathrm{d}_\perp\omega$	Irradiance
Importance flux	Ψ	$\int_A \Gamma\, \mathrm{d}A$	Flux

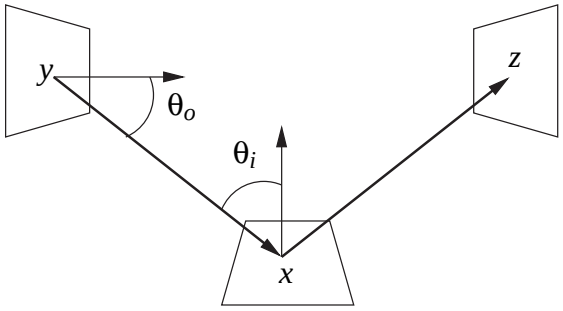


Figure 2.2: The three-point transport notation expresses transport in terms of vertices only.

2.8 Operator notation

Linear operators together with inner products are often used as a shorthand notation for integral equations. We will not give details about an operator notation here, because we do not use it explicitly in this text (except for the inner product notation, introduced earlier). However, some interesting properties of the light transport equations (e.g., convergence), can be derived elegantly from the operator notation. The reader is referred to [114, 19] for more information about light transport operators.

2.9 Path integral formulation

The radiance and importance transport equations express the transport rules in the form of an integral equation that describes *local* scattering on a surface. The global illumination is contained in the recursive nature of the integral equation. Several reformulations of the light transport problem were proposed that explicitly model the multiple scattering in global illumination.

Lafortune proposed the *global reflection distribution function* or GRDF [63]. The GRDF extends the BRDF to include multiple scattering effects from all surfaces. A specific sampling of the GRDF resulted in the bidirectional path tracing algorithm.

Veach used a path integral formulation [116], a common way to rewrite an integral equation as a single integral. As such, standard integration techniques can be applied to transport problems. For example, Veach proposed bidirectional path tracing using multiple importance sampling and the application of Metropolis sampling to light transport. We will also use the path integral formulation in the description of bidirectional path tracing and several multi-pass methods.

The path integral formulation requires two things: first, a variable transformation expresses the transport equations as a three-point transport, and second, paths and the path contribution functions are introduced to state the path integral.

2.9.1 Three-point transport

The radiance transport equation (2.1) is expressed in terms of incoming and outgoing directions with respect to a surface point $\mathbf{x}$. This can be rewritten using surface points only (see figure 2.2 for notation):

$$L(\mathbf{x} \to \mathbf{z}) = L_e(\mathbf{x} \to \mathbf{z}) + \int_{A_s} L(\mathbf{y} \to \mathbf{x})\, f_s(\mathbf{y} \to \mathbf{x} \to \mathbf{z}) G(\mathbf{x}, \mathbf{y})\, \mathrm{d}A_{\mathbf{y}}\,, \tag{2.7}$$

where G is the geometry term:

$$G(\mathbf{x}, \mathbf{y}) = V(\mathbf{x}, \mathbf{y}) \frac{\cos(\theta_i)\cos(\theta_o)}{\|\mathbf{x} - \mathbf{y}\|^2}\,,$$

with $V(\mathbf{x}, \mathbf{y})$ the visibility between $\mathbf{x}$ and $\mathbf{y}$ (1 if visible, 0 if not). The geometry factor appears, because the integration over projected solid angle is replaced by an integration over the surfaces in the scene.

The measurement equation (2.3) can be rewritten using surface points too, again by transforming the solid angle measure into an area measure:

$$I = \int_{A_s \times A_s} W_e(\mathbf{x} \to \mathbf{y}) L(\mathbf{x} \leftarrow \mathbf{y}) G(\mathbf{x}, \mathbf{y})\, \mathrm{d}A_{\mathbf{x}}\, \mathrm{d}A_{\mathbf{y}}\,.$$

2.9.2 The path integral

The path integral is obtained by repeatedly substituting the three point transport equation (2.7) into the transformed measurement equation. The resulting nested integrals can be reordered so that each term corresponds to an integral over paths of a specific length. The integrands of these reordered integrals are given by the *measurement contribution function*.

Given a path of length n, $\bar{\mathbf{x}} = \mathbf{x}_1 \ldots \mathbf{x}_n$, the measurement contribution function $\overline{f}(\bar{\mathbf{x}})$ is defined as the actual contribution of the path to the measurement I:

$$\begin{aligned}\overline{f}(\bar{\mathbf{x}}) = L_e(\mathbf{x}_1 \to \mathbf{x}_2) G(\mathbf{x}_1, \mathbf{x}_2)\, f_s(\mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_3) G(\mathbf{x}_2, \mathbf{x}_3) \ldots \\ \ldots f_s(\mathbf{x}_{n-2} \to \mathbf{x}_{n-1} \to \mathbf{x}_n) G(\mathbf{x}_{n-1}, \mathbf{x}_n) W_e(\mathbf{x}_{n-1} \to \mathbf{x}_n)\,.\end{aligned} \tag{2.8}$$

Paths of different lengths have different contribution functions, as the number of substitutions (and thus the number of nested integrals) differs. Also note that, since W_e is included in $\overline{f}$, each measurement has its own contribution function.

Integration of $\overline{f}(\bar{\mathbf{x}})$ over the domain $\overline{\Omega}$ uses the area product measure $\mathrm{d}\mu(\bar{\mathbf{x}}) = \mathrm{d}A_1 \times \ldots \times \mathrm{d}A_n$. Each vertex i has an associated area measure $\mathrm{d}A_i$. The path integral can now be written as

$$I = \int_{\overline{\Omega}} \overline{f}(\bar{\mathbf{x}})\, \mathrm{d}\mu(\bar{\mathbf{x}})\,. \tag{2.9}$$

Sometimes it may be instructive to write this as a sum of integrals, one for each path length:

$$I = \sum_{n=0}^{\infty} \int_{\overline{\Omega}_n} \overline{f}(\bar{\mathbf{x}})\, \mathrm{d}\mu(\bar{\mathbf{x}})\,. \tag{2.10}$$

More information on the path integral formulation can be found in [116].

2.10 Conclusion

This introductory chapter presented the underlying mathematical model for light transport that is used in this work. Although it does not model all the physical phenomena of light transport, it is sufficient for many applications that require global illumination computations. The model was formulated in different ways: using integral equations and as an integral over path space. Throughout the text these formulations will be interchanged depending on the notational convenience.

Now that the light transport problem is completely defined by the mathematical model above, it is time to look at algorithms to solve it. The next chapter discusses Monte Carlo integration, a very adequate technique for solving integral equations.

3 Monte Carlo integration

The mathematical model for light transport, developed in chapter 2, involves an integral equation that, in general, cannot be solved analytically. Monte Carlo integration is a mathematical technique that relies on statistical properties of random variables and random sampling to numerically estimate integrals. It is well suited for the high-dimensional integrals that arise from the light transport problem.

This chapter covers topics on Monte Carlo integration that are important for the remainder of the text. The explanation is brief, as numerous other books and articles —including many graphics dissertations— treat the method in great detail. Some good Monte Carlo introductions can be found in [55, 103, 34].

After some more introduction to the Monte Carlo method in §3.1, a bit of probability and basic Monte Carlo integration are explained in §3.2. Variance reduction techniques (§3.3) are key to the successful application of Monte Carlo to difficult integration problems. Section 3.4 discusses some important Monte Carlo rendering methods, that will be used in subsequent chapters. Conclusions are given in §3.5.

3.1 Introduction

This section reviews a little bit of history of the Monte Carlo method and discusses its main characteristics.

3.1.1 History

While some scientists have been known to use random sampling early on for estimating integrals [55, p. 4], the basis of the Monte Carlo method, along with its name, was formed during World War II in Los Alamos when the atom bomb was developed.

Monte Carlo methods estimate integrals —or other quantities that can be expressed as an expectation— by averaging the results of a high number of statistical trials. Computers are ideal for performing such trials, and the appearance of faster and faster computers has driven the wide spread application of Monte Carlo methods today.

This change towards stochastic simulation with computers was adequately described by Schreider [13]: *"It is interesting to note that computers have led to a novel revolution in mathematics. Whereas previously an investigation of a random process was regarded as being complete as soon as it was reduced to an analytic description, nowadays it is convenient in many cases to solve an analytic problem by reducing it to a corresponding random process and then simulating that process.*"

Monte Carlo methods are applied in fields as diverse as neutron transport problems, queuing theory, radiative heat transfer and, of course, computer graphics. In computer graphics, Cook et al. used randomly distributed rays to simulate glossy reflection, motion blur, and depth of field [24]. Although the method

was a Monte Carlo application, they did not identify it as such at first. Kajiya presented the rendering equation [54] and a Monte Carlo technique, path tracing, to solve it. This signaled the start of Monte Carlo rendering research. In the beginning a distinction was made between radiosity methods and (image-space) Monte Carlo methods, but later on further abstraction separated the storage method (finite elements) from the solution method (Monte Carlo), as it should be. Today, Monte Carlo methods compute pixel intensities but also illumination throughout the scene (e.g., Monte Carlo radiosity [5] and photon maps [47]), and they do this very successfully.

3.1.2 Characteristics

While Monte Carlo methods are applicable to a wide variety of problems, this text focuses specifically on solving integration problems. The core problem is the computation of an integral I:

$$I = \int_\Omega f(x)\,\mathrm{d}x\,.$$

Monte Carlo estimators will approximate I by taking a lot of random samples and averaging their contributions:

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N} g(\xi_i)\,,$$

where $g(\xi_i)$ is the contribution of the random sample $\xi_i \in \Omega$. As we will see further on, the contribution function g depends on the integrand f, the domain, and on the way the random samples are generated (§3.2.2).

There are several general characteristics of Monte Carlo estimators, that determine the strength and weaknesses of the method:

- The results are *statistical in nature*. This means that the estimate $\langle I \rangle$ can be wrong, even arbitrarily wrong. At first sight this seems to be a major drawback of the method, as one cannot be sure of the result. However, confidence intervals can be computed which indicate how far the estimate is likely to deviate from the correct result. These confidence intervals can be made as tight as necessary, for example by taking more samples. Therefore, the statistical nature of the results is largely compensated by other advantages of Monte Carlo integration.

- The evaluation of an estimator only requires the ability to *sample random points* ξ and evaluate $g(\xi)$ for these points. These are minimal requirements that make the method easily applicable to very difficult integration problems.

- The *convergence rate* —how quickly the error decreases with the number of samples— of basic Monte Carlo integration is proportional to $\sqrt{1/N}$. This means that to halve the error, four times as many samples are needed. This slow convergence is one of the main driving forces in Monte Carlo

research. Several variance reduction techniques are available that try to lower the variance of an estimator without using more samples, or that even try to improve the convergence rate. Variance reduction is often considered an art. Each application domain needs its own techniques and tuning to get the most out of the samples.

Compared to deterministic integration methods, it turns out that Monte Carlo methods are preferable for high-dimensional integrals. Deterministic integration rules of order r have a convergence rate of N^{-r} in one dimension. However, they will only converge as fast as $N^{-r/d}$ in d dimensions, because the N (regularly spaced) samples need to be distributed over all dimensions. Since the convergence of Monte Carlo methods is independent of dimension ($N^{-1/2}$), it is clear that for any order r a sufficiently high-dimensional integral will make Monte Carlo methods converge faster. Moreover, higher order integration rules (higher r) require very smooth integrands to be effective. This is not the case in global illumination, where many discontinuities may occur in the integrands, for example, due to visibility changes.

- The Monte Carlo method is *robust*. This is so because of the previous two reasons: only point sampling is required and convergence is assured (for unbiased estimators — see further).

3.2 Basic Monte Carlo integration

3.2.1 Probability

A *random variable* is a variable whose outcome is determined by a random process. The process (and the variable) can be discrete or continuous in nature. In the continuous case, each random variable ξ has an associated *cumulative distribution function* (cdf) $P(x)$ defined over a domain Ω. The cdf defines the probability of ξ being smaller than x for a single experiment:

$$P(x) = \text{Prob}\{\xi < x\}\,.$$

The *probability density function* (pdf) is defined as

$$p(x) = \frac{\mathrm{d}P(x)}{\mathrm{d}x}\,.$$

Note that any pdf must integrate to 1 over its domain.

A function $f(\xi)$ of a random variable ξ is itself a random variable. The *expected value* of a random variable $f(\xi)$ is given by

$$E\left[f(\xi)\right] = \int_\Omega f(x)p(x)\,\mathrm{d}x\,.$$

The *variance* of $f(\xi)$ is given by

$$V\left[f(\xi)\right] = \sigma^2\left[f(\xi)\right] = E\left[\left(f(\xi) - E\left[f(\xi)\right]\right)^2\right]\,.$$

The *standard deviation* $\sigma[f(\xi)]$ is given as the square root of the variance. It is a useful indication of how far the result of a single experiment can deviate from its expected value. The standard deviation is also called *standard error* or *RMS error*.

Multivariate distributions Given a pair of random variables (ξ,ζ), the *joint probability* $P(x,y)$ is defined as

$$P(x,y) = \text{Prob}\{\xi < x, \zeta < y\}\,.$$

The joint pdf is given by

$$p(x,y) = \frac{\partial^2 P(x,y)}{\partial x \partial y}\,.$$

The *marginal density* $p(x)$ eliminates the dependence on y:

$$p(x) = \int_{\Omega_x} p(x,y)\,\mathrm{d}y\,.$$

The *conditional probability density* $p(x|y)$ is given by

$$p(y|x) = \frac{p(x,y)}{p(x)}\,.$$

For independent random variables, the following identities hold:

$$p(x,y) = p(x)p(y)\,,$$

$$p(y|x) = p(y)\,,$$

$$V[x+y] = V[x] + V[y]\,.$$

The definitions in this paragraph were given for bivariate distributions, but are easily generalized to the multivariate case.

3.2.2 Monte Carlo estimators

Equipped with the necessary terms and definitions from probability, we can formulate Monte Carlo estimators and analyze their properties.

3.2.2.1 A basic Monte Carlo estimator

Suppose we want to compute the following integral:

$$I = \int_{\Omega_x} f(x)\,\mathrm{d}x\,,$$

with x a possibly multi-dimensional variable with domain Ω_x. Suppose we also have a pdf $p(x)$ ($p(x) > 0$ for $x \in \Omega_x$) according to which we can draw samples x_i. Using N independent samples, the integral I can now be estimated as

$$\langle I \rangle_N = \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}\,. \tag{3.1}$$

The contribution of a sample, $g(x_i)$ (see §3.1.2), is the integrand divided by the pdf, both evaluated in the sample point. This is the basic N-sample *Monte Carlo estimator* using importance sampling with a pdf $p(x)$. It is easy to show that the expected value $E[\langle I\rangle_N]$ equals the desired integral I.

The variance of the estimator (3.1) is given by

$$V[\langle I\rangle_N] = \frac{1}{N}\left(\int_{\Omega_x} \frac{f^2(x)}{p(x)}\,\mathrm{d}x - I^2\right).$$

The smaller the variance, the higher the probability that the estimate will lie close to its expected value.

3.2.2.2 Estimator properties

Many other estimators for integrals can be defined. This section defines some useful properties of general estimators. For an estimator $\langle I\rangle_N$ of a quantity I, the following properties can be defined:

Error

$$\mathrm{Error}(\langle I\rangle_N) = I - \langle I\rangle_N.$$

Unbiasedness An estimator $\langle I\rangle_N$ of a quantity I is called unbiased if

$$\forall N : E[\langle I\rangle_N] = I.$$

Bias An estimator is biased if its expected value differs from I. The bias of an estimator is

$$\beta[\langle I\rangle_N] = I - E[\langle I\rangle_N].$$

Consistency An estimator is consistent if its bias vanishes with an increasing number of samples:

$$\lim_{N\to\infty} \beta[\langle I\rangle_N] = 0.$$

Any unbiased estimator is of course also consistent. The advantage of consistent estimators is that the expected error can be made arbitrarily small, just by taking more samples. An additional advantage of unbiased estimators is that the error can be estimated easily, because the variance (and thus the standard deviation) can also be estimated from the samples, even for a moderate number of samples [55]. Estimating the error for biased (consistent) estimators is much harder and is usually only performed for the asymptotic case. However, this does not justify abandoning biased methods altogether. A small bias may outweigh the increased variance of an unbiased estimator.

In later chapters we will show that the introduction of some bias into the estimators can significantly improve the quality of the images.

Efficiency The efficiency of an estimator indicates the actual time it takes to compute an estimate to a certain precision. The efficiency of an estimator combines the variance of an estimator with the time it takes to compute the estimate. Let $T(\langle I \rangle)$ be the computation time, then the efficiency is given by

$$\texttt{eff}\langle I \rangle = \frac{1}{T(\langle I \rangle) \cdot V[\langle I \rangle]}.$$

3.2.3 Sampling random variables

To perform Monte Carlo integration on a computer, random variables must be sampled. Most mathematical libraries provide functions to sample uniform random variables over the unit interval[1]. Several techniques exist to sample variables from other distributions using only uniform random variables. These techniques are outlined below.

3.2.3.1 Transformation of variables

General transformation of random variables Suppose a certain random variable $x(x_1, \ldots, x_k)$ with pdf $p_x(x)$ is transformed into $z(z_1, \ldots, z_k)$ using a one-to-one function $z = g(x)$. What is the resulting distribution of z?

The relation between $p_x(x)$ and $p_z(z)$ is given by the Jacobian of the transformation g, similar to the change of variables in integration:

$$p_z(z) = \left| J\begin{pmatrix} x \\ z \end{pmatrix} \right| p_x(x),$$

with the Jacobian defined as

$$J\begin{pmatrix} x \\ z \end{pmatrix} = \begin{vmatrix} \frac{\partial x_1}{\partial z_1} & \cdots & \frac{\partial x_k}{\partial z_1} \\ \vdots & & \vdots \\ \frac{\partial x_1}{\partial z_k} & \cdots & \frac{\partial x_k}{\partial z_k} \end{vmatrix}.$$

For a one-dimensional variable, this is simply

$$p_z(z) = \frac{\mathrm{d}x}{\mathrm{d}z} p_x(x) = \frac{\mathrm{d}g^{-1}(z)}{\mathrm{d}z} p_x(g^{-1}(z)).$$

The Jacobian must be accounted for whenever a change of random variables occurs.

Sampling from a target distribution Transformation can also be used for sampling a target distribution. The problem is to find a transformation $z = g(x)$ with x uniformly distributed over $[0,1[$, so that z is distributed with a target pdf $p_z(z)$

Since x is uniform, $p_x(x) = 1$. The transformation of pdfs is:

$$p_z(z) = \left| J\begin{pmatrix} x \\ z \end{pmatrix} \right| \cdot 1 = \left| J\begin{pmatrix} g^{-1}(z) \\ z \end{pmatrix} \right|.$$

[1] In fact, these functions do not produce real random numbers, but deterministic, pseudo-random numbers that have properties similar to real random numbers. These numbers can be used as if it were real random numbers.

A transformation g must be found that makes this equation true, which involves solving a partial differential equation. For the one-dimensional case, it is simpler and boils down to the inverse cdf technique:

$$\begin{aligned} p_z(z) &= \left| J\begin{pmatrix} g^{-1}(z) \\ z \end{pmatrix} \right| = \left| \frac{\mathrm{d}g^{-1}(z)}{\mathrm{d}z} \right| \\ \Rightarrow x = g^{-1}(z) &= \int_{-\infty}^{z} p_z(z')\,\mathrm{d}z' = P(z) \\ \Rightarrow z &= P^{-1}(x)\,. \end{aligned}$$

Since it is usually easier to invert a one-dimensional cdf, one can also use marginal distributions to convert a multi-dimensional sampling into a sequence of 1D sampling procedures.

In Monte Carlo rendering, this is the most common technique for sampling from specific distributions, such as BRDF or EDF sampling.

3.2.3.2 Rejection sampling

If the transformation technique is too difficult, one can often use rejection sampling. Consider, for example, a one-dimensional pdf $p(z)$ over a finite domain. A sample z is chosen uniformly over the domain, but additionally a trial value t is sampled uniformly over the range of $p(z)$. If the point (z,t) lies under the function $p(z)$ the sample is accepted, otherwise a new pair is generated and tested.

The advantage of rejection sampling is that arbitrary distributions can be sampled. The efficiency of the sampler, however, can be quite low if many trials are rejected.

3.2.3.3 Metropolis sampling ($M(RT)^2$ algorithm)

The $M(RT)^2$ algorithm is an advanced sampling technique that can sample any density function in any number of dimensions. The algorithm generates samples by mutating a previous sample. An acceptance test determines if the new tentative sample is kept or if the previous sample is used again. Perfect sampling of the pdf is only reached asymptotically, and samples can be strongly correlated, depending on the mutation strategies used. Details can be found in [73, 55].

3.2.3.4 Custom methods

Several custom methods are available for sampling random variables with specific distributions. For example the maximum of k uniform random variables is a random variable z with cdf: z^k. See [55] for a detailed overview.

3.3 Variance reduction

A Monte Carlo estimator can be made arbitrarily accurate by simply increasing the number of samples. However more samples require more computation time, and a lot of research has gone into techniques to

increase the accuracy of estimators without using more samples. These variance reduction techniques will be outlined next, with a focus on those used further in the text.

3.3.1 Importance sampling

The basic Monte Carlo estimator (3.1) samples according to a certain pdf $p(x)$. Importance sampling refers to the effect that this pdf has on the variance of the estimator. It can be shown that the optimal pdf is given by

$$p(x) = \frac{f(x)}{\int_{\Omega_x} f(x)\,\mathrm{d}x}\,.$$

In other words, the best choice is a pdf that is proportional to the integrand. This pdf would result in a zero-variance estimator. However, the normalization factor, that ensures the pdf integrates to 1, is the integral of $f(x)$, which is exactly the (unknown) quantity to be computed.

In practice a pdf that closely matches the integrand (i.e., $f(x)/p(x)$ as constant as possible) will still significantly reduce the variance of the estimator. Owen [77] presents guidelines for choosing a pdf that ensures effective importance sampling.

In Monte Carlo rendering, importance sampling is one of the most frequently used variance reduction techniques. It can be especially effective when the integrand contains narrow peaks, for instance, when sampling directions in the presence of highly specular materials.

3.3.2 Multiple importance sampling

Importance sampling uses a single pdf to generate samples, but sometimes it may be difficult to construct a single pdf that works well in all circumstances. A typical example in rendering is the computation of the direct lighting of a surface. For a diffuse surface, the direct light is best computed by sampling points on a light source explicitly, while for a specular surface it is better to sample a reflected ray and see if it hits a light source. One could make this distinction based on surface properties, but multiple importance sampling provides a method for weighting the different sampling techniques in a way that preserves their strengths.

Multiple importance sampling was introduced by Veach. More information can be found in [116] and [114].

3.3.2.1 Multi-sample estimator

Recall the integral to be computed:

$$I = \int_{\Omega} f(x)\,\mathrm{d}x\,.$$

Suppose n pdfs $p_i(x)$ are used to estimate I, and that N_i samples are allocated to each pdf p_i. One could compute n estimates, one for each pdf, and weight the results, for instance based on the estimated variance of each individual estimate. However, it is possible to formulate combined estimators that assign an

appropriate weight to *each individual sample*, resulting in a better combination. Such an estimator, the multi-sample estimator, is given by

$$\langle I \rangle_{\text{MIS}} = \sum_{i=1}^{n} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})},$$

with $x_{i,j}$ sampled according to $p_i(x)$. Each sample is weighted with $w_i(x_{i,j})$ where $w_i(x)$ is a weighting function specific for pdf i.

This estimator is unbiased when the following constraints on $w_i(x)$ are fulfilled [114, p. 260]:

$$\begin{aligned} &(1) \quad \forall x \in \Omega : \sum_{i=1}^{n} w_i(x) = 1 \\ &(2) \quad \forall x \in \Omega, p_i(x) = 0 : w_i(x) = 0 . \end{aligned} \tag{3.2}$$

The first constraint takes into account that several pdfs can generate a certain sample x: the total weight of a sample over all pdfs is 1, so that the contribution of x is accounted for exactly once.

The second constraint is only important when some pdfs do not sample the complete domain Ω. When a sample cannot be generated by a certain pdf ($p_i(x) = 0$), the weighting function w_i will never be evaluated for this sample (as it is never generated with p_i) and thus should evaluate to zero. Otherwise, the total contribution of this sample would be to low.

Note that, contrary to 'single' importance sampling, it is not necessary that each pdf covers the complete domain Ω. It is sufficient that for each $x \in \Omega$, at least one pdf is able to generate it. This allows for interesting combinations of pdfs that can be tuned to certain parts of the domain [P2].

Any set of weighting functions that conforms to (3.2) makes a valid estimator. For example constant weights w_i over the whole domain Ω result in a weighted combination of separate estimators: $\langle I \rangle = \sum_i w_i \langle I \rangle_i$. Separation of the domain can be achieved by setting each w_i to one for a specific sub-domain of Ω. Of course, weights that minimize the variance of the combined estimator, are preferred. Veach proposes several weighting heuristics, which are discussed below.

The balance heuristic The balance heuristic is given by the following weighting functions:

$$w_i(x) = \frac{N_i p_i(x)}{\sum_{k=1}^{n} N_k p_k(x)} . \tag{3.3}$$

Both the number of samples N_i and the pdf evaluation have an effect on the weight of a sample:

- **Pdf:** When a sample $x_{i,j}$ is generated that has a very small pdf evaluation (an 'improbable' sample) but a larger integrand evaluation $f(x_{i,j})$, the ratio $f(x_{i,j})/p_i(x_{i,j})$ can become very large. In an unweighted estimator, such samples increase the variance of the estimate enormously, and the pdf itself is deemed bad.

In multiple importance sampling, however, such bad samples can be compensated by the weight: if there is another sampling technique $p_{i'}(x)$ that has a much larger pdf evaluation in $x_{i,j}$, the sum in the denominator of $w_i(x_{i,j})$ becomes large compared to the numerator, and the small weight of the sample compensates the large (unweighted) contribution $f(x_{i,j})/p_i(x_{i,j})$.

To put it another way, samples that are easily generated with a certain technique (large pdf evaluation), will also get a larger portion of the weight.

- **Number of samples:** The argument for the number of samples in the weighting functions is similar: techniques for which more samples are used (larger N_i), will get a larger weight for their samples, because overall the variance of their estimator is lower (due to the larger number of samples).

The balance heuristic is a good default choice when no extra knowledge about the integrand or estimators is known. Veach shows that the improvement of other weighting heuristics with respect to the balance heuristic is limited [114, p. 264]. Let $\langle I \rangle_{\text{BAL}}$ be the balance heuristic estimator and $\langle I \rangle_{\text{MIS}}$ any valid multiple importance sampling estimator, then it can be shown that

$$|V[\langle I \rangle_{\text{BAL}}] - V[\langle I \rangle_{\text{MIS}}]| \leq \left(\frac{1}{\min_i N_i} - \frac{1}{\sum_i N_i} \right) I^2 . \tag{3.4}$$

This means the variance decrease that can be obtained with the (unknown) optimal weighting heuristic instead of the balance heuristic, is limited by the rightmost term.

However, other heuristics can be useful in low variance cases where one of the sampling techniques is particularly good in a certain part of the domain. Most practical is the power heuristic discussed next. Other heuristics, such as the cutoff or the maximum heuristic, are only of interest theoretically. Details are given in [114].

Power heuristic The power heuristic uses the following weights:

$$w_i(x) = \frac{(N_i p_i(x))^\beta}{\sum_{k=1}^{n} (N_k p_k(x))^\beta} .$$

An exponent $\beta = 2$ works well in practice and is most common. The exponentiation increases the weight of a sample for those techniques that have a high probability of generating this sample, i.e., when $p_i(x)$ large.

3.3.2.2 One-sample estimator

The multi-sample estimator allocates a number of samples N_i to each of the sampling techniques. A one-sample estimator requires choosing a single sample from one of the sampling techniques. A probability γ_i is assigned to each of the techniques, and one of them is chosen randomly.

The balance heuristic for the one-sample estimator becomes:

$$w_i(x) = \frac{\gamma_i p_i(x)}{\sum_{k=1}^{n} \gamma_k p_k(x)}.$$

Veach shows that the balance heuristic is optimal for the one-sample case; no other heuristics are necessary.

3.3.2.3 Discussion

Multiple importance sampling provides a great tool for constructing robust Monte Carlo estimators. It is no longer necessary to construct the one perfect pdf that matches the integrand over the whole domain. Specific pdfs can be designed for difficult cases, and the weighting ensures a good combination of the individual samples.

Of course, multiple importance sampling does not come for free. Computing the weight $w_i(x_{i,j})$ for a single sample requires the evaluation of *all n* pdfs $p_k(x_{i,j})$ and not only the one that generated the sample. This makes the evaluation of a sample more expensive but this is compensated by the lower variance of the combined estimator.

The foremost important application of multiple importance sampling in rendering is bidirectional path tracing (§3.4.3), but it can also be applied to several problems in stochastic ray tracing (§3.4.1). In chapter 6 we will extend multiple importance sampling for use in multi-pass methods.

3.3.3 Sample placement

Much effort in Monte Carlo research was invested into careful sample placement to minimize the variance. Uniform random sampling can cause samples to clump together. The variance increases because the domain is not covered as well as it could be. Stratified sampling and Quasi-Monte Carlo (QMC) methods try to remedy this problem.

3.3.3.1 Stratified sampling

Stratified sampling divides the domain into a number of different, non-overlapping strata. Each stratum receives a predetermined fraction of the number of samples, for example one sample per stratum. This form of sampling ensures a better coverage of the sampling domain since all strata are sampled and clumping is reduced.

The variance and convergence properties of stratified sampling have been studied in detail in the literature. Some interesting points to note are the following:

- Stratified sampling can never be worse than uniform random sampling, which motivates its use whenever possible.

- It is better to increase the number of strata than the number of samples per stratum. In practice, this means using one sample per stratum.

- The variance can decrease as the square of the number of strata. Using N samples and also N strata, a convergence rate of $1/N^2$ is much faster than the normal $1/N$ rate. While this is true for smooth integrands, high variation within strata —discontinuities for instance— limit the benefits of stratified sampling.

A problem with stratified sampling is the division into strata. A d-dimensional domain divided into equal strata, gives a minimum of 2^d strata (one split in each dimension). For high-dimensional domains this results in many strata, liming the choice of the sample count. Several solutions exist that try to remedy this problem, for example, orthogonal array sampling [76] and also Quasi Monte Carlo methods.

3.3.3.2 Quasi-Monte Carlo methods

Quasi-Monte Carlo (QMC) methods go even further than stratified sampling and abandon the randomness of the sampling completely[2]. Special, deterministic number sequences are used to generate the samples. The sequences try to distribute the samples as evenly and well distributed as possible, but without introducing too much regular structure in them (regular structure introduces aliasing).

A large body of research discusses various sequences and their convergence properties on a number of integration problems. Depending on the smoothness of the integrand, a significantly faster convergence rate can be obtained. In computer graphics, however, discontinuities limit the convergence speedup. Still QMC sampling provides a convenient way to at least stratify the samples, without the need for explicit strata.

Examples of frequently used QMC sequences are the Halton sequence, the Niederreiter sequence, and (t,m,s)-nets. More information on the use of QMC integration in computer graphics can be found in [92, 57, 109, 5].

3.3.4 Russian roulette and splitting

Russian roulette and splitting are techniques to increase the efficiency of estimators and are closely related to transport problems.

Light transport can be computed by averaging a number of random walks. To increase efficiency, many random walks should be allocated to important regions in the scene or to important parts of the light transport. Russian roulette is used to terminate unimportant random walks, while splitting is used to increase the number of random walks in important regions. The concepts will be explained in a standard integral setting.

[2] As said before, random numbers generated with computer are only pseudo-random and thus also deterministic. The difference is that the pseudo-random numbers have similar properties as real random numbers, whereas QMC sequences do not.

Given the desired integral $I = \int f(x)\,\mathrm{d}x$ and some estimator $\langle I \rangle$, Russian roulette introduces an additional random process that determines whether the estimator is evaluated. Given an *acceptance* probability P_{rr} and a uniform random number $\xi \in [0,1[$, the estimate is given by

$$\langle I \rangle_{rr} = \begin{cases} \langle I \rangle / P_{rr} & \text{if } \xi < P_{rr} \\ 0 & \text{otherwise}\,. \end{cases}$$

The resulting estimator $\langle I \rangle_{rr}$ is unbiased. It has a higher variance than the original estimator $\langle I \rangle$, but the time to compute it is lower (under the assumption that P_{rr} is faster to compute than $\langle I \rangle$). This can be useful if the estimator is just a part of the total computation, for example when evaluating a random walk or a sum of terms:

$$I = I_1 + I_2 + I_3 + \ldots \tag{3.5}$$

If Russian roulette is applied to the individual terms with an acceptance probability that matches the (estimated) contribution of the term to I, the total efficiency of $\langle I \rangle$ can be increased.

Russian roulette is frequently, if not always, used in transport problems to terminate random walks. It ensures that more work is spent in shorter and more important walks (or paths).

Splitting is a similar technique. It uses more samples to estimate important terms in the sum (3.5) instead of using fewer samples for unimportant terms:

$$\langle I_j \rangle_{\text{split}} = \frac{1}{N} \sum_{i=1}^{N} \langle I_j \rangle_i\,,$$

where the $\langle I_j \rangle_i$ are independent estimates for I_j. In a random walk context, this corresponds to splitting an important particle j into N scattered particles and averaging all the contributions.

3.3.5 The use of expected values

A very straightforward but effective variance reduction technique is the use of expected values to reduce the dimensionality of the integral. Suppose we want to compute the following integral:

$$I = \int_{\Omega_y} \int_{\Omega_x} f_1(x,y)\,\mathrm{d}x\,\mathrm{d}y\,.$$

An estimator for I is given by

$$\langle I \rangle = \sum_{i=1}^{N} \frac{f_1(x_i,y_i)}{p(x_i,y_i)}\,. \tag{3.6}$$

Suppose also that the function $f_2(y)$ and the marginal density $m(y)$ are known:

$$f_2(y) = \int_{\Omega_x} f_1(x,y)\,\mathrm{d}x\,,$$
$$m(y) = \int_{\Omega_x} p(x,y)\,\mathrm{d}x\,.$$

By integration, the dependence on x is removed and the integration problem is reduced to an integral over Ω_y. An estimator for I using these functions is given by:

$$\langle I \rangle = \sum_{i=1}^{N} \frac{f_2(y_i)}{m(y_i)}\,. \tag{3.7}$$

This estimator has a lower variance than estimator (3.6). The dependence on x is substituted by the expected value and the dimensionality of the integration problem is reduced.

This reduction in variance is logical, as a part of the integral was computed analytically. However, when $m(y)$ is not equal to the marginal density, both integrals are computed with unrelated pdfs and nothing can be said about the respective estimator variance. In chapter 6, an example will demonstrate such a case where partial analytical integration does not reduce the variance. A solution to this problem is proposed using a combination of multiple importance sampling and expected values.

3.3.6 Other techniques

There are several other techniques for variance reduction, such as control variates, antithetic variables and regression methods. While they did not yet find many uses in computer graphics, they still hold good potential for variance reduction. More information can be found in previously cited references.

3.4 Monte Carlo rendering

Monte Carlo techniques excel in high-dimensional integration problems. Over the years they have become popular for solving integral equations. Typically these methods trace *random walks* by recursively sampling the transport equations.

Stochastic sampling in rendering surfaced with the work of Cook [24], Dippé and Wold [25], and Lee et al. [68]. The first full Monte Carlo solution of the rendering equation was proposed by Kajiya [54], spurring a large body of Monte Carlo rendering research.

In this section we will review several important Monte Carlo rendering methods: stochastic ray tracing (§3.4.1), particle tracing (§3.4.2) and bidirectional path tracing (§3.4.3). These methods form the basis of the global illumination methods presented in this dissertation.

The discussion of the algorithms is kept concise, because they are treated in great detail in many other graphics textbooks and dissertations.

3.4.1 Stochastic ray tracing

Stochastic ray tracing is an image-space algorithm that traces *eye-paths* from the eye or camera into the scene, as shown schematically in figure 3.1.

Stochastic ray tracing is, in fact, a straightforward Monte Carlo estimation of the measurement and radiance transport equation. First, the measurement equation (2.3) is estimated using N_0 samples per pixel:

$$\langle I \rangle = \frac{1}{N_0} \sum_{i=1}^{N_0} \frac{W_e(\mathbf{x}_0^{(i)} \rightarrow \omega_0^{(i)}) L_i(\mathbf{x}_0^{(i)} \leftarrow \omega_0^{(i)}) \cos\theta_0^{(i)}}{p(\mathbf{x}_0^{(i)}, \omega_0^{(i)})} .$$

The starting vertex $\mathbf{x}_0$ (the camera position) and a sampled direction ω_0 through the pixel determine the

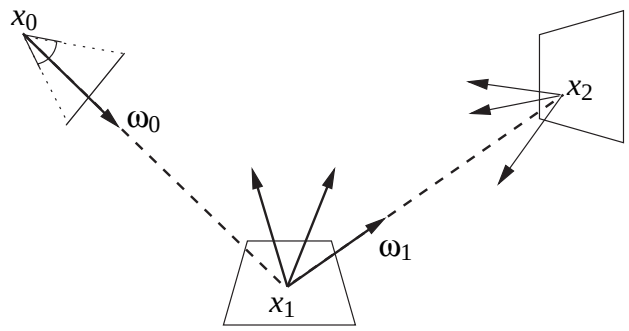


Figure 3.1: Stochastic ray tracing constructs paths starting from the eye. For each vertex scattered rays are traced to estimate incoming radiance.

starting ray of an eye path. The incoming radiance L_i in the estimator is unknown. It is equal to the outgoing radiance at the nearest intersection $\mathbf{x}_1$, which is determined by tracing a ray in the direction ω_o. The outgoing radiance, $L(\mathbf{x}_1 \to \omega_0)$, is then estimated by substitution with the radiance transport equation (2.1). The self-emitted radiance $L_\mathrm{e}(\mathbf{x}_1 \to \omega_0)$ in $\mathbf{x}_1$ can be evaluated directly, but the reflected and refracted radiance, L_r, requires another Monte Carlo estimation (using N_1 scattered samples):

$$\langle L_\mathrm{i}(\mathbf{x}_0 \leftarrow \omega_0)\rangle = L_\mathrm{e}(\mathbf{x}_1 \to \omega_0) + \langle L_\mathrm{r}(\mathbf{x}_1 \to \omega_0)\rangle\,,$$

with

$$\langle L_\mathrm{r}(\mathbf{x}_1 \to \omega_0)\rangle = \frac{1}{N_1}\sum_{j=1}^{N_1} \frac{L_\mathrm{i}(\mathbf{x}_1 \leftarrow \omega_1^{(j)})\, f_s(\mathbf{x}_1, \omega_0 \leftarrow \omega_1^{(j)})\cos\theta_1^{(j)}}{p(\omega_1^{(j)})}\,.$$

The unknown radiance $L_\mathrm{i}(\mathbf{x}_1 \leftarrow \omega_1)$ is estimated in the same way, by tracing a ray to $\mathbf{x}_2$ and sampling the incoming radiance at this hit point (see figure 3.1).

To end the recursive process a maximum path length can be chosen or Russian roulette can be used. A maximum path length ignores light from longer paths, and thus is biased. This bias, however, is small for a sufficiently large maximum. For Russian roulette, an absorption probability P_{abs} determines whether a path is prolonged and its contribution is adjusted accordingly. This is unbiased.

While the basic algorithm is extremely simple, several common variance reduction techniques can drastically reduce the rendering time:

- An important optimization is next event estimation, which corresponds to the direct sampling of the light sources. Instead of waiting for paths to hit the typically small light sources, path vertices are sampled on the light sources explicitly. Both sampling techniques, direction sampling and light source sampling, can be combined using multiple importance sampling (§3.3.2).

- The splitting factor N_k (or spawning factor) determines the number of samples that are used in each recursion level. N_0 determines the number of samples per pixel. $N_{1,2,\ldots}$ determine the number of scattered samples in the path vertices. If $N_{1,2,\ldots}$ are all set to 1, no splitting is applied, which results in

the *path tracing* algorithm. Path tracing with Russian roulette allocates more work to shorter, more important paths and it is a good choice lacking any extra information about the scene. Bolin and Meyer analyzed optimal splitting factors for stochastic ray tracing in [12].

- Directions ω_k are chosen according to a pdf $p(\omega_k)$. This pdf is usually taken proportional to the BSDF (times the cosine if possible). This is especially important for specular materials, that have a very peaked BSDF.

  If extra information about the incoming radiance is available —for example through an object-space radiance solution, computed on the fly or in a preprocessing step—, it can be taken into account into the importance sampling for further variance reduction [29, 65, 46].

Figure 3.2 (page 36) shows a reference image of an example scene that is used to compare the different Monte Carlo rendering algorithms. The scene contains two light sources. The one on the right is directed towards the specular glass sphere in the middle. The other light is directed towards the white diffuse panel on the left. The materials of the walls and floor all have diffuse and glossy components. This scene exhibits many different illumination features such as caustics, indirect diffuse illumination, glossy reflection, and specular refraction. This image is also shown in figure 3.6 (page 37) for direct comparison with the images rendered with the different Monte Carlo algorithms.

Figure 3.3 shows a path tracing solution of the example scene. The path tracing image was computed using 25 samples per pixel ($\pm$10 min.). This image reveals some interesting characteristics of stochastic ray tracing:

- Glossy and specular scattering from the eye are handled quite well. The light refracted through the sphere, as well as the glossy highlights from the panel and the right light source, are adequately sampled by eye paths.

- Important indirect diffuse illumination shows a high level of noise, most apparent in the region above the white panel and in the shadow of the sphere on the left.

- The light source on the right and the white panel cause caustic effects on the floor. Eye paths are not very well suited to compute such effects: when an eye ray hits the floor, only a very small part of the hemisphere contributes to the caustic: Reflected rays must be shot towards the glass sphere in such a way that the exiting refracted ray hits the light source or the panel. The smaller the light source, the worse this sampling problem becomes.

  The caustic effects cause a high variance in the estimator, which causes them to appear very noisy in the image.

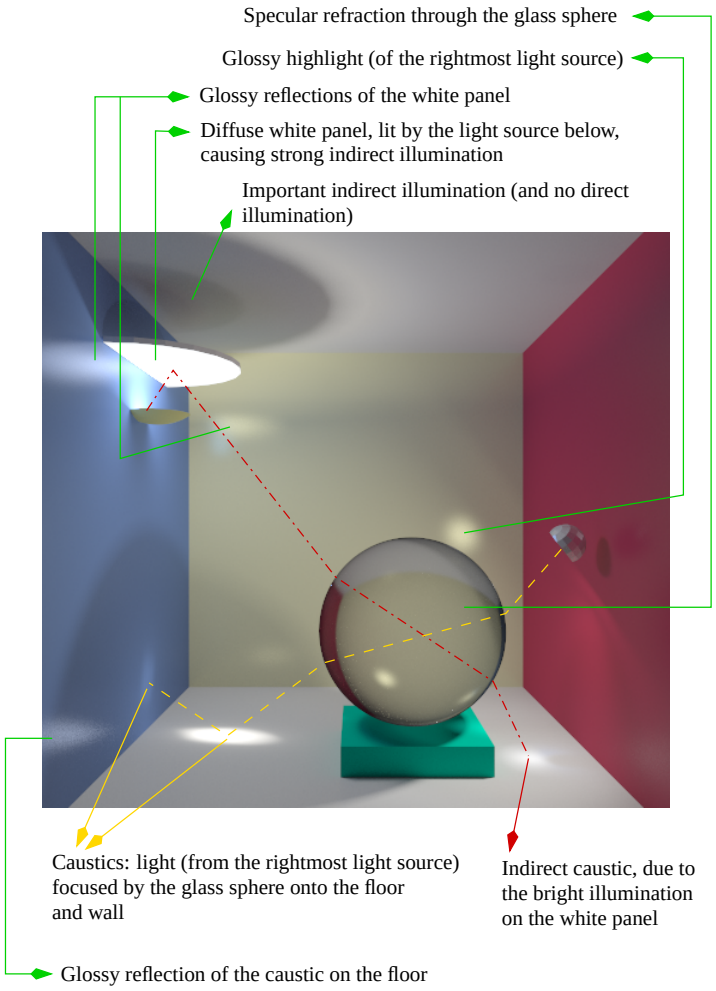


Figure 3.2: An example scene with many different illumination features is used to compare the different Monte Carlo rendering algorithms. This reference solution was rendered with bidirectional path tracing using 2048 bidirectional paths per pixel (about 20 hours of computation time).

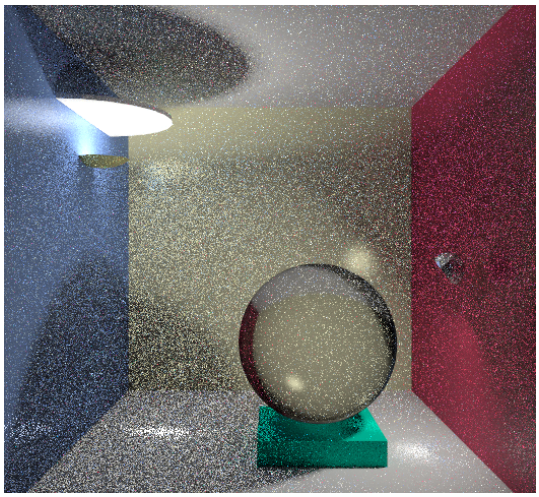


Figure 3.3: An example scene rendered with path tracing (25 samples/pixel). The scene exhibits many different illumination features such as caustics, indirect illumination, glossy reflection, and specular refraction

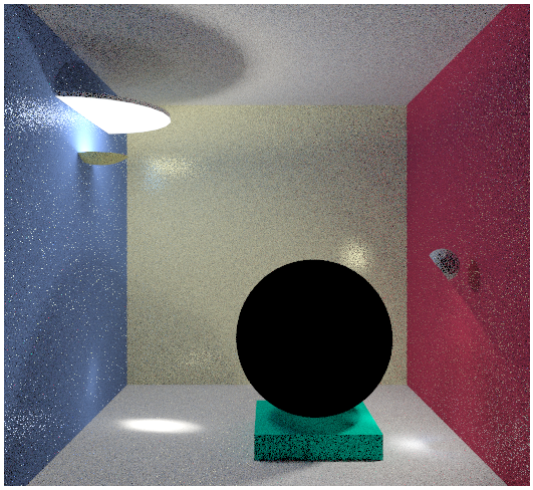


Figure 3.4: The same scene rendered with light ray tracing (25 samples/pixel).

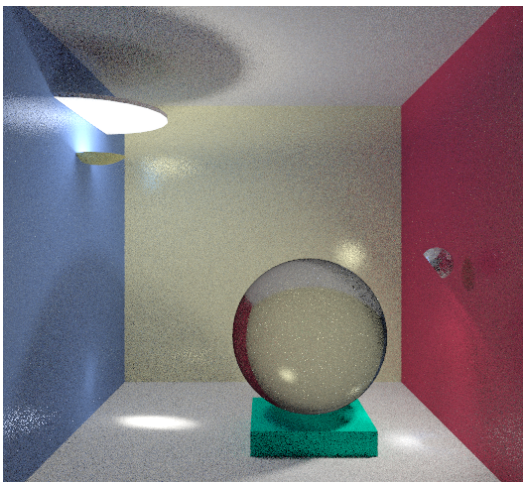


Figure 3.5: The scene rendered with bidirectional path tracing (16 samples/pixel, same computation time as above).

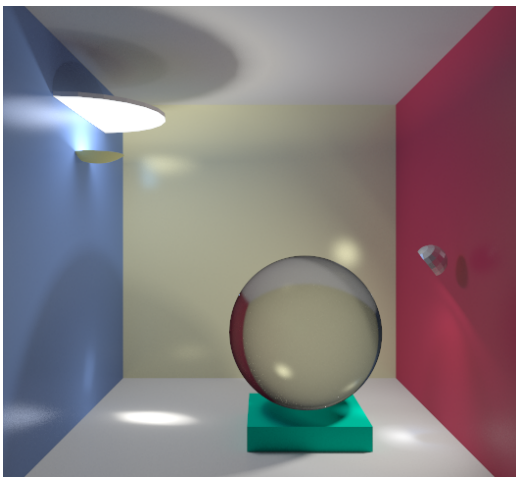


Figure 3.6: The reference solution, rendered with bidirectional path tracing (2048 samples/pixel).

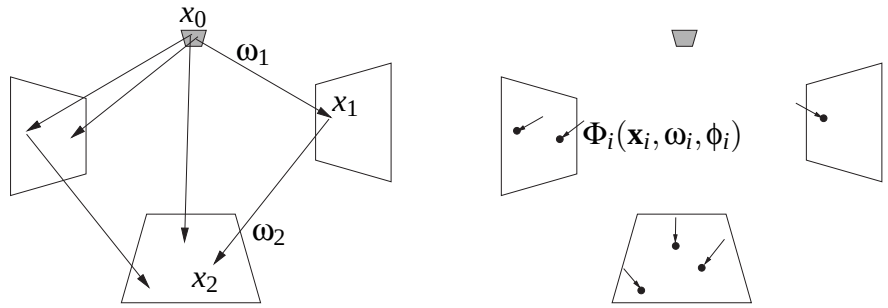


Figure 3.7: Particle tracing generates a number of light paths (left). All the vertices of the light paths form a set of particles that approximate the incoming radiance on surfaces in the scene (right).

3.4.2 Particle tracing

Particle tracing is similar to path tracing, but the paths are constructed starting from the light sources. Other names used for particle tracing are photon tracing[3], photon tracking and light ray tracing.

The tracing of the particles is independent of the final measurements computed with these particles. Therefore, it is convenient to explicitly split the transport and measurement (or reconstruction) phase. Conceptually this corresponds to tracing all the light paths first and collecting all the path vertices as a set of particles, and to computing the measurements using the set of recorded particles afterwards. This idea is shown schematically in figure 3.7.

3.4.2.1 Tracing the particles

A particle $\Phi(\mathbf{x}, \omega, \phi)$ consists of a position $\mathbf{x}$, a direction ω and an associated weight ϕ, the 'energy' or color of the particle. In our treatment of particle tracing we will consider incoming particles that arrive at surfaces and thus approximate the incoming radiance. In [114, p.121] an outgoing radiance approximation using particles leaving a surface was given, which we adapted for incoming particles.

Particles are constructed by tracing a number of light paths. Each vertex in the path forms a particle (this corresponds to a collision estimator random walk, see e.g., [103]). The following procedure is repeated for N light paths (see also figure 3.7):

- First, a point $\mathbf{x}_0$ on a light source and an outgoing direction ω_1 are sampled. Then, a ray is traced and the nearest intersection forms the position $\mathbf{x}_1$ of the first particle $\Phi_1(\mathbf{x}_1, \omega_1, \phi_1)$ with weight:

$$\phi_1 = \frac{1}{N} \frac{L_e(\mathbf{x}_0 \rightarrow \omega_1) \cos\theta_0}{p(\mathbf{x}_0) p(\omega_1 | \mathbf{x}_0)} .$$

- An absorption test determines whether the light path is extended. Russian roulette is applied using an acceptance probability P_{rr}.

[3] Although the name 'photon' is often used in the context of particle tracing, a real physical photon, with a single wavelength and a fixed energy, is quite different. However, within the simplified mathematical framework of the rendering equation, it is often intuitive to think about particles as photons.

- When the light path is extended, a new direction ω_2 is sampled and a ray is traced to $\mathbf{x}_2$. The resulting particle $(\mathbf{x}_2, \omega_2, \phi_2)$ has a weight

$$\phi_2 = \phi_1 \cdot \frac{1}{P_{rr}} \frac{f_s(\mathbf{x}_1, \omega_2 \to \omega_1) \cos\theta_1}{p(\omega_2|\Phi_1)},$$

  where $p(\omega_2|\Phi_1)$ is the conditional probability for sampling the new direction given the previous particle.

- The particle scattering is repeated until the particle is absorbed. Then a new light path is started.

The result is a set of particles $\Phi_i(\mathbf{x}_i, \omega_i, \phi_i)$ that approximates the equilibrium incoming radiance distribution L_i. Note that the number of particles, N_Φ, can be larger than the number of light paths, N, since more than one particle per path can be added to the set.

A single particle represents a differential flux $\phi_i \, \mathrm{d}_\perp\omega_i \, \mathrm{d}A$ incident to the surface in $\mathbf{x}_i$. Any measurement I, defined by an outgoing importance function $W_\mathrm{e}(\mathbf{x} \to \omega)$ can be estimated using the set of particles as follows:

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N_\Phi} \phi_i W_\mathrm{e}(\mathbf{x}_i \to \omega_i). \tag{3.8}$$

This is an unbiased estimator for I, which can be proven using the importance transport equation [114, 80].

For example, when W_e is set to 1 for a certain surface and the upper hemisphere above the surface, equation (3.8) reduces to the sum of the particle weights that hit the surface, which estimates the incident flux on that surface.

3.4.2.2 Analog simulation

In the Monte Carlo literature, an analog simulation is defined as a simulation that mimics the underlying transport rules [103]. For light transport, this corresponds to a simulation of the particle model of light [79]: A light source emits a number of particles (or 'photons') that all have the same power. When reaching a surface, a particle can be scattered or absorbed, but at any time a photon keeps the same power.

Such an analog simulation is useful, because a low variance in the particle powers (all powers are equal) will result in a lower variance of the measurement estimates (3.8).

An analog particle tracing algorithm must choose its pdfs in the following way:

- **Particle birth:** When the starting point $\mathbf{x}_0$ is chosen according to the emittance of the light sources, and the direction ω_1 is chosen with a pdf proportional to the EDF times cosine, the particle weight is:

$$\phi_1 = \frac{1}{N} \frac{L_\mathrm{e}(\mathbf{x}_0 \to \omega_1) \cos\theta_0}{\frac{B_\mathrm{e}(\mathbf{x}_0)}{\int_{A_\mathrm{s}} B_\mathrm{e}(\mathbf{x}) \, \mathrm{d}\mathbf{x}} \cdot \frac{L_\mathrm{e}(\mathbf{x}_0 \to \omega_1) \cos\theta_0}{\int_{\Omega_{2\pi}} L_\mathrm{e}(\mathbf{x} \to \omega) \, \mathrm{d}_\perp\omega}} = \frac{\Phi_e}{N}.$$

  Here we have used the definition of the total self-emitted flux $\Phi_e = \int_{A_\mathrm{s}} B_\mathrm{e}(\mathbf{x}) \, \mathrm{d}\mathbf{x}$ and the definition of the self-emitted radiosity B_e.

- **Particle scattering:** The acceptance probability for Russian roulette is set equal to the local albedo, and the pdf for direction sampling is taken proportional to the BSDF times cosine. This results in the following weight:

$$\begin{aligned}\phi_i &= \phi_{i-1} \cdot \frac{1}{P_{rr}} \frac{f_s(\mathbf{x}_{i-1}, \omega_i \to \omega_{i-1}) \cos\theta_{i-1}}{p(\omega_i | \Phi_{i-1})} \\ &= \phi_{i-1} \cdot \frac{1}{\rho(\mathbf{x}_{i-1}, \omega_{i-1})} \frac{f_s(\mathbf{x}_{i-1}, \omega_i \to \omega_{i-1}) \cos\theta_{i-1}}{\frac{f_s(\mathbf{x}_{i-1}, \omega_i \to \omega_{i-1}) \cos\theta_{i-1}}{\rho(\mathbf{x}_{i-1}, \omega_{i-1})}} \\ &= \phi_{i-1}\end{aligned}$$

Note that it is not always possible to perform a perfect analog simulation, because exact sampling according to the required pdfs may not be possible. But trying to keep the particle powers as homogeneous as possible, is still beneficial for the estimates.

3.4.2.3 Measurements and reconstruction

Many existing global illumination algorithms use particle tracing as their method for light transport calculations. The actual measurements can be computed from the particle representation directly:

$$\langle I \rangle = \sum_{i=1}^{N} \phi_i \cdot W_e(\mathbf{x}_i \to \omega_i) .$$

This formalism clearly separates the light transport from the measurement itself. Algorithms using particle tracing can now be distinguished by their response function W_e. Note that in their implementation, many algorithms do not need to store the individual particles.

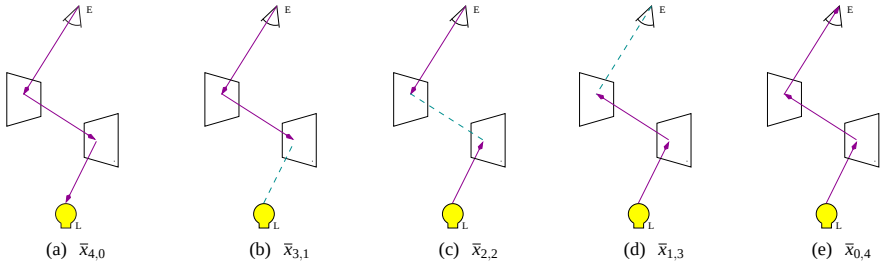
Examples of algorithms using particle tracing are several Monte Carlo radiosity algorithms [79, 111], density estimation [93, 122], photon maps [47, 50], caustic map computations [23], and light ray tracing with direct estimation of the pixel fluxes [28]. All but the last method are object-space methods, and will be explained in more detail further on.

Light ray tracing with direct estimation of the pixel fluxes could be called the adjoint method of path tracing. While path tracing is directly derived from the radiance transport equation, light ray tracing starts at the light sources and then the importance transport equation is applied recursively. Each path vertex is explicitly connected with the camera to estimate the pixel flux.

While both path tracing and light ray tracing are unbiased, they construct paths differently and thus define a different probability density on path space. This results in a different variance or noise level for the illumination features in a scene. Figure 3.4 shows a light ray tracing solution of the example scene. Compared to the path tracing solution (figure 3.3), the following differences can be noted:

- Glossy reflections are somewhat worse, and the specular glass sphere even turns up completely black. Specular sampling is best handled by sampling directions according to the highly peaked BSDF. In this case, however, light paths that are about to exit the sphere are explicitly connected to the eye[4].

[4] Note that a specular refracted direction will also be sampled, but the ray leaving the glass sphere will never hit our pinhole camera.



(a) $\bar{x}_{4,0}$ (b) $\bar{x}_{3,1}$ (c) $\bar{x}_{2,2}$ (d) $\bar{x}_{1,3}$ (e) $\bar{x}_{0,4}$

Figure 3.8: A bidirectional path $\bar{\mathbf{x}}$ can be constructed by connecting eye and light sub-paths of different lengths.

The resulting direction will almost never lead to a significant BSDF evaluation. This problem is similar to the caustics problem in stochastic ray tracing [5].

- Caustics are handled very well by light paths, because light paths hitting a specular object are extended by sampling a scattered ray according to a very sharply peaked BSDF.

- For this scene, indirect diffuse light is handled better compared to stochastic ray tracing. The bright white panel contributes significantly to the indirect illumination. Many light paths will hit this panel and will be scattered into the scene, contributing to the indirect illumination. In stochastic ray tracing, however, the white panel only contributes when an eye-path hits it. This does not happen very often, resulting in a higher noise level.

3.4.3 Bidirectional path tracing

As the name suggests bidirectional path tracing constructs paths in two directions: from the lights and from the camera. It was proposed by Lafortune [62, 63] and later independently by Veach [115]. Veach recognized that, when using the path integral formulation (§2.9), the different path sampling techniques in bidirectional path tracing (BPT) can be weighted using multiple importance sampling [116]. The resulting weights preserve the advantages of the sampling techniques, making BPT superior to path tracing or light ray tracing alone for many scenes.

3.4.3.1 Path construction

Bidirectional paths A bidirectional path is formed by connecting an eye and a light sub-path. Given an eye sub-path of length s and a light sub-path of length t, the bidirectional path will be written as $\bar{\mathbf{x}}_{s,t}$.

The contribution of a path is given by the measurement contribution function $\overline{f}(\bar{\mathbf{x}})$ (equation 2.8). The pdf evaluation for a path is the accumulated pdf of the eye sub-path multiplied by the pdf of the light sub-path. The connection itself is deterministic and does not influence the pdf. The path integral formulation requires that all pdfs are specified with respect to the area product measure (see §2.9). Consider for example

[5] In fact it is even worse, because direct hits on a area light source (as used in the example scene) do contribute to the caustics, but a direct hit on the aperture of our pinhole camera is not possible. Of course, if point light sources were used, none of the caustics would show up in stochastic ray tracing either.

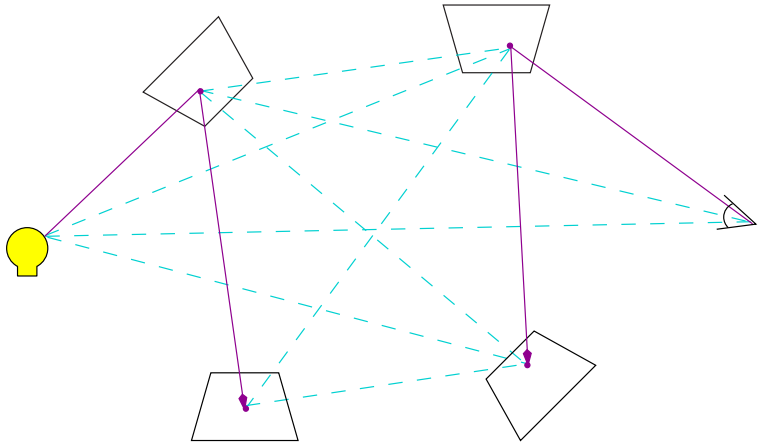


Figure 3.9: An efficient sampling of bidirectional paths combines all vertices of an eye sub-path with all the vertices of a light sub-path.

a path of length four, constructed by connecting an eye sub-path $\mathbf{e}_0\mathbf{e}_1$ with a light sub-path $\mathbf{l}_0\mathbf{l}_1$: $\bar{\mathbf{x}}_{2,2} = \mathbf{e}_0\mathbf{e}_1\mathbf{l}_1\mathbf{l}_0$ (Figure 3.8 (c)). The contribution and pdf of this path is given by

$$\overline{f}(\bar{\mathbf{x}}_{2,2}) = L_e(\mathbf{l}_0 \to \mathbf{l}_1) G(\mathbf{l}_0, \mathbf{l}_1) f_s(\mathbf{l}_0 \to \mathbf{l}_1 \to \mathbf{e}_1) G(\mathbf{l}_1, \mathbf{e}_1) \\ f_s(\mathbf{l}_1 \to \mathbf{e}_1 \to \mathbf{e}_0) G(\mathbf{e}_1, \mathbf{e}_0) W_e(\mathbf{e}_0 \to \mathbf{e}_1) \,,$$

$$p(\bar{\mathbf{x}}_{2,2}) = p(\mathbf{l}_0) p(\mathbf{l}_1|\mathbf{l}_0) p(\mathbf{e}_0) p(\mathbf{e}_1|\mathbf{e}_0) \,.$$

Note that for the connection, the evaluation of the geometry factor $G(\mathbf{l}_1, \mathbf{e}_1)$ requires a visibility test.

The same path could also be formed by connecting eye and light sub-paths of other lengths: $\bar{\mathbf{x}}_{0,4}$, $\bar{\mathbf{x}}_{1,3}$, $\bar{\mathbf{x}}_{3,1}$, and $\bar{\mathbf{x}}_{4,0}$. So a path of length 4 can be sampled in 5 different ways (Figure 3.8). In general there are $n+1$ different bidirectional sampling techniques to sample a path $\bar{\mathbf{x}}_n$ of length n.

The measurement contribution $\overline{f}$ for a path is always the same, no matter how the path is sampled. The pdf on the other hand, is different for each sampling technique $p_{s,t}$. As we have seen before when comparing path tracing and light ray tracing, each sampling technique will be able to handle certain illumination features well, while others will give rise to a high noise level. Combination of the different techniques with multiple importance sampling will preserve the strengths of the different sampling techniques (see §3.4.3.2).

Efficient sampling of bidirectional paths While independent eye and light sub-paths could be sampled for each bidirectional path, it is possible and much more efficient to construct several bidirectional paths from a single pair of eye and light sub-paths.

This efficient construction of paths is shown in figure 3.9. A light path and an eye path are traced. Each eye path vertex is connected with each light path vertex, resulting in several bidirectional paths.

The combination of an eye and a light sub-path to form a bidirectional path $\bar{\mathbf{x}}_{s,t}$ has to handle a few special cases:

- $s = 0$: The eye sub-path is empty. Such a path will only contribute if the light sub-path hits the camera directly. For a pinhole camera, such paths will never occur and the associated pdf evaluates to 0.
- $s = 1$: Such paths are particle tracing paths that are explicitly connected to the camera.
- $t = 1$: These are path tracing paths directly connected to a point sampled on a light source.
- $t = 0$: An eye path that hits a light source directly.

These special cases show that bidirectional path tracing includes all the paths that are generated with path tracing and particle tracing.

3.4.3.2 Multiple importance sampling

The previous section showed that a single path $\bar{\mathbf{x}}$ can be constructed using different sampling techniques $p_{s,t}$. Therefore, the contribution of a path must be weighted using multiple importance sampling. For N pairs of eye and light paths, the bidirectional estimator is given by

$$\langle I \rangle = \sum_{s \geq 0} \sum_{t \geq 0} \sum_{i=1}^{N_{s,t}} w_{s,t}(\bar{\mathbf{x}}_{i,s,t}) \frac{\overline{f}(\bar{\mathbf{x}}_{i,s,t})}{p_{s,t}(\bar{\mathbf{x}}_{i,s,t})}, \tag{3.9}$$

where $N_{s,t}$ is the number of samples used for the specific sampling technique s,t.

With the multi-sample model, the weights for a path $\bar{\mathbf{x}}$ using the power heuristic are given by:

$$w_{s,t}(\bar{\mathbf{x}}) = \frac{(N_{s,t} p_{s,t}(\bar{\mathbf{x}}))^{\beta}}{\sum_{s'=0}^{s+t} \sum_{t'=0}^{s+t-s'} (N_{s',t'} p_{s,t}(\bar{\mathbf{x}}))^{\beta}}.$$

To evaluate the weighting function for a single path, an evaluation of the pdf is required for each sampling technique s',t' that could generate the same path (when $s'+t' = s+t$). These pdfs have many common factors which allows an efficient evaluation scheme (cfr. [114]).

3.4.3.3 Optimizations

Several optimizations can be applied to make the BPT estimator even more efficient.

Multiple measurements Until now we considered only the estimation of a single pixel, but usually measurements I_j for every pixel in the image are needed. Some sampling techniques can be used to estimate all pixels I_j at once. This is the case for light paths that are connected directly to the camera ($p_{1,t}$, the pixel I_j is determined by the connecting edge in the path) or for light paths that would hit the camera aperture directly ($p_{0,t}$).

If N pairs of eye and light paths are traced *per pixel* for a total of N_{pix} pixels, $N_{\text{pix}} \times N$ light paths can contribute to any of the pixels. Thus, the number of samples used for these sampling techniques is $N_{0,t} = N_{1,t} = N_{\text{pix}} \times N$ while the number of samples for other techniques is $N_{s>1,t} = N$. The increased

number of samples for these two sampling techniques must also be used in the weighting functions, and will result in a larger contribution for such paths.

Direct light optimization This optimization concerns the sampling technique $p_{s,1}$, the direct lighting technique that connects an eye sub-path to a vertex on a light source. In the above description the eye sub-path would be connected to the starting vertex of the light sub-path, which is chosen uniformly over the light sources.

Several direct lighting techniques, however, optimize the choice of the light vertex based on the point to be lit (the last vertex in the eye sub-path)[94]. These techniques can be combined easily with BPT by adapting the sampling technique $p_{s,1}$ so that it chooses its own light vertex. This modifies the pdf $p_{s,1}$, which must be incorporated in the estimator and weighting functions.

Selective visibility tests Veach and Lafortune both proposed a technique to reduce the number of visibility tests required by all the connecting path segments. Both techniques are based on Russian roulette to determine whether the visibility test will be performed. More information can be found in [66] and [114].

3.4.3.4 Implementation

Our BPT implementation uses the power heuristic with $\beta = 2$. We use the multiple measurements optimization and the direct lighting optimization. Selective visibility tests are not implemented. While this could lower the number of (highly optimized) intersection tests, our experience is that the time spent in the BSDF, pdf, and weight evaluations, which always have to be evaluated, limits the possible gain in performance.

A maximum path length can be chosen independently for the eye paths, the light paths, and the combined bidirectional paths. This allows us to easily perform path tracing and light ray tracing by just changing these maxima. If not explicitly stated otherwise, all maxima are set to 7. While this excludes some light transport (i.e., longer paths) in the images, the difference to a full solution is small.

A minimum path length can also be chosen. Absorption tests using Russian roulette are only performed when a path has already reached the minimum path length. Since short paths carry the bulk of the transported light, the extra variance caused by the Russian roulette is undesirable. The minimum defaults to 2. For a typical scene, the average path length is around 4.

3.4.3.5 Results

Figure 3.5 shows the example scene rendered with bidirectional path tracing using 16 samples per pixel (same computation time as the other images). The comparison with path tracing (fig. 3.3) and light ray tracing (fig. 3.4) shows the following interesting differences:

- Caustics, as well as specular and glossy scattering, are sampled well. This is logical as BPT is a superset of path and light ray tracing, and thus is able to combine their strengths.

- The indirect light is even better than in the light ray tracing image, because additional estimators (with eye and light sub-paths longer than 1 vertex) also sample this illumination.

- One notable remaining source of noise is the highly glossy reflection of the caustic on the left. An eye path, that samples the glossy reflection well, has problems with the caustic. A light path, on the other hand, has trouble with the final glossy reflection towards the eye, but not with the caustic. There are several solutions to this problem, for example, explicitly storing caustics allows eye paths to use the stored caustic instead of having to generate it by sampling (see chapter 8).

3.4.4 Other methods

Many other Monte Carlo methods for rendering have been proposed. Most of them are variations or combinations of the three algorithms described above. We only want to mention the following:

- **Metropolis light transport:** Using the path integral formulation, Metropolis sampling can be applied to light transport[117]. An initial path is generated using another technique, for example BPT. The metropolis algorithm records the score for the current path and then applies a mutation to it. This mutation is accepted or rejected based on the principle of detailed balance. If accepted, the current path is replaced by the mutation.

  The advantage of Metropolis sampling is that paths are sampled proportionally to the measurement contribution function, the actual contribution of a path to the image. This results in perfect importance sampling, but the subsequent paths can be highly correlated depending on the mutation strategies. Metropolis light transport performs well for difficult lighting configurations, using carefully designed mutation strategies. Finding good mutation strategies is the most difficult part of metropolis light transport.

- **Monte Carlo radiosity** The methods described above are image-space algorithms. Of course, Monte Carlo techniques can also be applied to object-space algorithms, such as the radiosity method. Monte Carlo radiosity methods turn out to be very efficient and successful, as shown in [5].

3.5 Conclusion

Monte Carlo methods are widely used for tackling transport problems and their integral equations. They are frequently used and will continue to be used in physically based rendering algorithms such as described in this dissertation. This chapter summarized the Monte Carlo techniques used in the remainder of the

text. Some important Monte Carlo rendering algorithms —stochastic ray tracing, particle tracing, and bidirectional path tracing— were explained. These methods form the basis of the following chapters.

4 Multi-pass methods

This chapter introduces multi-pass methods. Several common object-space and image-space components of multi-pass methods are discussed, and a brief overview is given of multi-pass methods presented in the literature. A convenient regular expression notation is used to describe all the light transport that is covered by the separate components and the existing methods. The components and regular expression notation discussed in this chapter will be used frequently in subsequent chapters.

4.1 Introduction

Many (single-pass) global illumination algorithms have been proposed, ranging from finite element radiosity methods to stochastic ray tracing. All these methods have their specific strengths and weaknesses. A very successful approach to develop robust global illumination systems is to combine different algorithms into a two-pass or multi-pass method. A good combination of algorithms tries to preserve their respective strengths and diminish or even remove their weaknesses.

Technically, a multi-pass rendering method divides the computation of the light transport in a scene into two or more different passes. Each of these passes uses a different algorithm that tackles some part of the light transport. Typically one or more object-space passes compute an approximation to the radiance in the scene, that is used afterwards by an image-space pass.

Such a multi-pass approach introduces the concept of separation in the light transport calculations: each pass is assigned a specific part of the light transport. This separation of light transport over different algorithms must be done very carefully:

- No light transport may be accounted for more than once, otherwise the resulting solution will be incorrect (too bright).
- For a full global illumination solution, all light transport must be covered by at least one of the algorithms. No light transport may be missing in the final image.

In short, these requirements state that all possible light paths must be covered exactly once by the combination of algorithms, in order to obtain a correct, full global illumination solution.

In this chapter a simple framework for characterizing multi-pass methods is presented. It is based on standard regular expressions to describe the separated parts of the light transport (§4.2). Using the regular expression notation, we will analyze several common object-space and image-space components of current multi-pass methods (§4.3). The chapter ends with a brief overview of the literature on multi-pass methods (§4.4).

E	eye or camera vertex
L	light source vertex
$\mathsf{D_r}, \mathsf{D_t}$	Diffuse reflection/transmission
$\mathsf{G_r}, \mathsf{G_t}$	Glossy reflection/transmission
$\mathsf{S_r}, \mathsf{S_t}$	Specular reflection/transmission
$\mathsf{S_{r_\infty}}, \mathsf{S_{t_\infty}}$	Perfectly specular reflection/transmission
$\|$	'or' operator
$+$	addition[1]
D	$\mathsf{D_r}\|\mathsf{D_t}$
G	$\mathsf{G_r}\|\mathsf{G_t}$
S	$\mathsf{S_r}\|\mathsf{S_t}$
X	$\mathsf{D}\|\mathsf{G}\|\mathsf{S}$
X^*	zero or more occurrences of X
X^+	one or more occurrences of X
X^k	exactly *k* occurrences of X
$\mathsf{X}^{0\ldots k}$	0 to *k* occurrences of X

Table 4.1: Regular expression symbols for path classification

As we will see, several different object-space passes have been used in multi-pass configurations, but the (final) image-space passes are practically always simple variations of stochastic ray tracing. In the next chapter we will present a new technique for ray tracing based passes, that derives the path evaluation directly from regular expressions. This offers an improved flexibility in separating the light transport and tuning the multi-pass configuration (MPC). Most importantly, it will allow us to use bidirectional path tracing as a final image-space pass.

4.2 Regular expressions to describe light transport

This section introduces regular expressions to describe separate parts of the light transport.

In most multi-pass methods the separation of light transport is based on the type of materials that are supported by a certain algorithm. For example, a classical radiosity algorithm can only handle diffuse materials. Thus, if we consider all the possible light paths that are covered by such a radiosity algorithm, the *scattering components* that are used in the path vertices will only include the diffuse component.

Regular expressions are a convenient way to classify such transport paths. The symbols in the regular expression syntax identify the scattering components —diffuse, glossy or specular— used in the path vertices. Other symbols identify the light and eye vertices. An overview of all symbols is given in table 4.1. This notation extends the notation introduced by Heckbert [40].

All paths covered by a radiosity algorithm can now be described by LD^*. Caustics, for example, can be described by $\mathsf{LS}^+\mathsf{D}$ paths, meaning that one or more specular reflections or refractions occurred before a final diffuse scattering.

Note that the BSDF or scattering components refer to those components that are taken into account by

[1] In fact the '$+$' and '$|$' operators are the same, but for clarity $+$ is used to add larger regular expressions together

the algorithm and not the material type. The materials in the scene can have any combination of components in their BSDF. For example a classic radiosity algorithm will only cover the diffuse illumination (LD^*) whatever the material properties in the scene. The other components of the materials in the scene, will have to be handled by another algorithm.

All the possible transport paths from light to eye are written as

$$\mathsf{L}(\mathsf{D}|\mathsf{G}|\mathsf{S})^*\mathsf{E} = \mathsf{L}\mathsf{X}^*\mathsf{E}.$$

These paths must be covered by a rendering algorithm when a full global illumination solution is required. Different components in a multi-pass configuration will cover different paths, but their combination should cover all paths. This can be checked by combining the separate regular expressions of the different multi-pass components. This is explained in the next section.

4.3 A simple multi-pass framework

This section outlines a simple framework for multi-pass methods. Both object-space and image-space components of multi-pass configurations will be discussed and classified using regular expressions (§4.3.1 and §4.3.2).

4.3.1 Object-space algorithms: partial radiance solutions

Object-space algorithms compute an approximation of a partial radiance solution in the scene. The solution is *partial* because not all of the possible light transport will be covered by the object-space pass (e.g., only diffuse storage or only caustics). Since the storage and the transport simulation have a limited accuracy, these partial radiance solutions will be approximate.

Such a stored partial radiance approximation will be abbreviated as SPAR. A SPAR corresponds to a certain radiance approximation:

$$\text{SPAR } M : \hat{L}_M(x \rightarrow \omega).$$

It gives an estimate for the exact partial radiance $L_M(x \rightarrow \omega)$ for any point in the scene and in any direction. Each object-space algorithm is characterized by this radiance approximation.

Several important properties of object-space algorithms must be considered when applying them in a multi-pass configuration (MPC):

- **Covered transport:** The light transport that is actually computed in a SPAR M, can be expressed by a regular expression M. This expression is needed to identify redundant or missing transport in the MPC.

- **Efficiency of construction:** This indicates the time needed to compute the object-space solution. It is no use spending hours on an object-space solution, unless it is view independent and more than one view is required.

- **Efficiency of reconstruction:** A reconstruction evaluates $\hat{L}_M(x \rightarrow \omega)$ for a particular point and direction. The time needed for a reconstruction depends on how the data is stored. Since an image-space pass requires a large number of evaluations, the efficiency of reconstruction will have a big influence on the total rendering time.

- **Storage efficiency:** Storing the radiance approximation requires a certain amount of computer memory. A low memory usage is of course preferred, but this usually leads to a decrease in accuracy (see next item).

- **Accuracy:** The accuracy indicates the error made by approximating the exact radiance solution: $\varepsilon = \hat{L}_M - L_M$. A high accuracy requires more time and memory for the object-space step, but a low accuracy solution often requires a more expensive image-space pass to mask the errors (e.g., a final gathering, as discussed further on).

The most common object-space algorithms in MPCs are discrete radiosity methods (§4.3.1.1) and methods based on particle tracing (§4.3.1.2). We will analyze their characteristics next.

4.3.1.1 Discrete radiosity methods

Discrete radiosity methods are finite element methods that discretize the rendering equation into a system of linear equations. For each element in the finite element mesh, the radiance is approximated by a weighted set of basis functions.

Most versions only consider diffuse BSDF components, so that the illumination of an element is completely defined by its radiosity (hence the name of the method). Assuming a constant basis function (a constant radiosity over each element), the radiosity is given by the following system of linear equations:

$$B^{(j)} = B_e^{(j)} + \rho^{(j)} \sum_{i=1}^{N} F_{ij} B^{(j)} .$$

The form factor F_{ij} is the fraction of the radiosity of element j that reaches element i.

Many different techniques exist to solve this large system of linear equations. Also many extensions, such as hierarchical meshing [35, 101], discontinuity meshing [70], non-diffuse materials [45, 95, 19], higher order basis functions over the elements [130], and non planar elements [87] have been proposed to improve the accuracy and efficiency of radiosity methods. The reader is referred to the extensive radiosity literature for more information. A good overview can be found in [22] (pre-1993 radiosity) and [5] (Monte Carlo radiosity).

In general, radiosity methods have the following characteristics:

- **Covered transport:** Most radiosity algorithms only cover diffuse transport: $\mathsf{M}_{\text{rad}} = \mathsf{LD}^*$. Some extensions can also handle glossy materials. Some other methods take into account the specular component in the intermediate reflections, but the storage never includes the specular component, because that would require a very fine subdivision of the hemisphere in order to accurately represent the outgoing illumination.

- **Efficiency of construction:** Recent radiosity methods with hierarchical meshing and clustering can be quite efficient.

- **Efficiency of reconstruction:** Reconstruction is very fast: the element that contains the evaluation point must be located in the scene hierarchy, and then it is simply a matter of evaluating a few basis functions.

- **Storage efficiency:** Since illumination information is stored for each element in the scene, the storage depends on the complexity of the scene geometry. For very complex scenes, this can become a significant drawback for radiosity methods.

- **Accuracy:** The accuracy in radiosity methods is largely determined by the discretization error, the error made by mapping the radiosity function onto the finite element mesh. Typically the accuracy is good for slowly varying illumination, but not so good for high illumination gradients such as sharp shadows. Implementing good and robust meshing strategies is a very difficult part of radiosity algorithms.

4.3.1.2 Particle tracing methods

Besides radiosity methods, almost all other object-space preprocessing passes in an multi-pass configuration are based on particle tracing (See §3.4.2). The difference between the algorithms lies between the storage and reconstruction method, and in the covered light transport.

Covered light transport Particle tracing can accommodate any material type. Accurate storage of the specular component, however, requires too much memory, so that only diffuse and (sometimes) glossy components are stored. This leads to the following covered transport options:

$$\mathsf{M}_{\text{pt}} = \mathsf{L}(\mathsf{X})^*(\mathsf{D}|\mathsf{G}) \quad \text{or} \quad \mathsf{L}(\mathsf{X})^*(\mathsf{D})\,.$$

Particle tracing is very adequate for computing caustics, and several methods use it to compute the caustics separately:

$$\mathsf{M}_{\text{caustic}} = \mathsf{L}(\mathsf{S})^+(\mathsf{D}|\mathsf{G}) \quad \text{or} \quad \mathsf{L}(\mathsf{S})^+(\mathsf{D})\,.$$

Characteristics of particle tracing methods The characteristics of particle tracing methods depend on how the illumination is stored in the scene.

Many methods use a mesh or texture maps on the surfaces to accumulate contributions during particle tracing. When a mesh is used, the methods are called *continuous radiosity* methods or *particle tracing radiosity* methods [79, 5]. Texture maps containing illumination information are often called *light maps* [3, 40].

The characteristics of these finite element storage methods (texels in a texture map are also a form of mesh elements) are:

- **Covered transport:** Since a finite element mesh is used, only the diffuse component of the radiance is stored, but the scattering of particles can accommodate any BSDF component.
- **Efficiency of construction:** Tracing particles is fast. The number of particles needed to get a decent radiance estimate in each element, however, depends largely on the size of the elements. Each element needs a significant amount of hits for its radiance estimate to be accurate enough. For small elements, many particles may be needed.
- **Efficiency of reconstruction:** Reconstruction is as fast as in discrete radiosity methods.
- **Storage efficiency:** The storage requirements depend on the number of elements, just as in discrete radiosity methods.
- **Accuracy:** The accuracy is determined by the element size, and the variance of the radiance estimate in each element. Large elements decrease the variance (the noise) but introduce a larger bias (e.g., blurred shadow boundaries). Small elements, on the other hand, require many particles to reduce the variance in the elements to an acceptable level.

  Using hierarchical subdivision to adapt the size of elements, can help with both the accuracy and construction efficiency. This is discussed in more detail in §7.8.

Other algorithms store the particles themselves. This leads to the following characteristics:

- **Covered transport:** Some methods that store the particles also use them to reconstruct glossy illumination, while others only reconstruct diffuse illumination. (In the former case more particles are needed.)
- **Efficiency of construction:** The construction time depends on how many particles are needed, which in turn depends on the required accuracy.
- **Efficiency of reconstruction:** Reconstructing radiance from a number of particles requires density estimation techniques and is typically much slower than evaluating a few basis functions.

On the other hand, the reconstruction can be made more intelligent as information from each of the particles is available, which is not the case when each particle contributes directly to the radiance of an element.

- **Storage efficiency:** Some methods, such as photon mapping (chapter 8), do not link particles to elements directly. For highly tessellated scenes with many small elements, less particles than elements may be needed for a sufficiently accurate radiance approximation. Decoupling storage from the geometry of the scene is one of the important advantages that particle tracing methods may have.

  For accurate solutions, storing all particles requires a lot of memory, and it may be advantageous to try to adapt the storage so that more particles are stored in areas where the illumination is important (see chapter 9).

- **Accuracy:** The accuracy is determined by the number of particles that are stored, and by the method used to reconstruct the illumination from the particles (see also chapter 8).

To summarize, the most frequently used object-space components in a multi-pass configuration are discrete radiosity methods and particle tracing methods. In our work, we will use the former in chapters 5 and 6, and the latter in chapters 7, 8 and 9.

4.3.2 Image-space algorithms: readout strategies

Image-space algorithms compute a radiance value for each pixel in the image. In a multi-pass configuration (MPC), an image-space pass will use the stored partial radiance approximations (SPARs) that were computed in one or more object-space preprocessing passes, in order to compute the image more efficiently.

Almost all image-space passes in a MPC are based on path sampling algorithms. The most frequently used image-space algorithms are classical ray tracing and stochastic ray tracing, the difference being that classical ray tracing only handles perfectly specular scattering and thus will never amount to a full global illumination solution. A few MPCs use rasterization algorithms, for instance with a Z-buffer to determine the visibility.

4.3.2.1 Readout strategy

How a SPARis actually used by an image-space pass is defined by what we call the *readout strategy*. Each SPAR has its own readout strategy. For example, as we will see further on, a radiosity solution may be visualized indirectly, after a diffuse reflection, while a caustic map is best visualized directly.

Readout strategies can also be identified by regular expressions. Consider an eye path that ends on a surface where some stored illumination is used. The readout strategy defines which components of the BSDF are used in the intermediate vertices of the path.

Formally, given a SPAR M and an eye path of length k, a readout strategy is identified by the following regular expression:

$$\begin{array}{rcll} \mathsf{MQ}^{(k)} & = & \mathsf{MQ}_{k-1}\ldots\mathsf{Q}_1 E & \text{when the SPAR is read} \\ & = & \emptyset & \text{when the SPAR is not used for paths of length } k \end{array}$$

Each $\mathsf{Q}_{k'}$ determines the scattering components that were used in the vertex k'. The complete expression $\mathsf{Q}^{(k)}$ indicates which eye paths are covered, and combined with M, the covered transport of the SPAR, the contribution to the pixel radiance for paths of length k is obtained.

For example, a combination of radiosity and stochastic ray tracing uses a typical readout strategy as follows (given for a number of paths of different length):

$$\begin{array}{rccccc} 2: & & & & \mathsf{M}_{\text{rad}} & E \\ 3: & & & \mathsf{M}_{\text{rad}} & (G|S) & E \\ 4: & & \mathsf{M}_{\text{rad}} & (G|S) & (G|S) & E \\ 5: & \mathsf{M}_{\text{rad}} & (G|S) & (G|S) & (G|S) & E \\ & \ldots & & & & \end{array} \tag{4.1}$$

The radiosity solution is read at the end vertex of every eye path. The scattering components in the intermediate vertices (Q_k) are restricted to glossy and specular only.

If all scattering components were used, including the diffuse component, the diffuse illumination would be accounted for more than once. This can be shown, by considering the contribution of paths of length 2 and 3 when all components are used:

$$\begin{array}{rrcl} 2: & \mathsf{M}_{\text{rad}} E & = & \mathsf{LD}^*\mathsf{E} \\ 3: & \mathsf{M}_{\text{rad}}(D|G|S)E & = & \mathsf{LD}^*(\mathsf{D}|\mathsf{G}|\mathsf{S})\mathsf{E} = \mathsf{LD}^*(\mathsf{G}|\mathsf{S})\mathsf{E} + \mathsf{LD}^*\mathsf{E} \end{array} \tag{4.2}$$

4.3.2.2 Identification of redundant and missing transport

The regular expressions provide an easy way to check if a multi-pass configuration computes some light transport more than once, or whether some transport is missing.

For a single SPAR the formal definition of a readout strategy allows to formulate a constraint that indicates whether some light transport is computed more that once. In general, a SPAR M and its readout strategy do not compute any redundant light transport if there is no overlap between the regular expressions for different path lengths:

$$\forall k, k' \geq 0, k \neq k' : \mathsf{MQ}^{(k)} \cap \mathsf{MQ}^{(k')} = \emptyset . \tag{4.3}$$

The total contribution I to the image by a SPAR and its readout strategy is the union of the contributions of all different path lengths:

$$\mathsf{Q} = \bigcup_{k=0}^{\infty} \mathsf{Q}^{(k)},$$
$$\mathsf{I} = \mathsf{MQ}.$$

When different SPARs and readout strategies are combined, the regular expressions can be used to check whether the combined transport covers all illumination exactly once:

$$\begin{array}{ll} \text{No redundant transport:} & \forall i,j : \mathsf{I}_i \cap \mathsf{I}_j = \emptyset \\ \text{All transport covered:} & \bigcup_i \mathsf{I}_i = \mathsf{LX}^*\mathsf{E} \end{array}$$

Once the regular expressions of the different components are identified, these constraints can be checked easily.

4.3.2.3 Typical readout strategies

Only a few typical readout strategies are used in today's multi-pass configurations. Most common are direct visualization and final gathering, both combined with specular and, when necessary, glossy reflections:

- **Direct visualization:** At every vertex in an eye path the SPAR is evaluated directly. Scattering in the vertices does not include storage components to avoid redundant transport: $\mathsf{Q}_k = (\mathsf{G}|\mathsf{S})\,\text{or}\,(\mathsf{S})$. The readout strategy in equation (4.1) is an example of a direct visualization.

  Direct visualization requires a very accurate radiance approximation ($\hat{L}_M(x \to \omega)$), otherwise the errors in the reconstructed radiance will be visible in the image.

- **Final gathering:** With final gathering the SPAR is only used indirectly. Suppose, for example, that illumination is only stored for the diffuse component of the BSDF. Final gathering uses the stored illumination after exactly one diffuse scattering. For a radiosity SPAR, this would correspond to the following readout strategy (given for several eye path lengths):

$$\begin{array}{lccccccc}
2: & & & & & L & E & \\
3: & & & & L & (G|S) & E & + \\
3: & & & \mathsf{M}_{\text{rad}} & & D & E & \\
4: & & & L & (G|S) & (G|S) & E & + \\
4: & & & \mathsf{M}_{\text{rad}} & (G|S) & D & E & + \\
4: & & & \mathsf{M}_{\text{rad}} & D & (G|S) & E & \\
5: & & L & (G|S) & (G|S) & (G|S) & E & + \\
5: & & \mathsf{M}_{\text{rad}} & (G|S) & (G|S) & D & E & + \\
5: & & \mathsf{M}_{\text{rad}} & (G|S) & D & (G|S) & E & + \\
 & & \ldots & & & & &
\end{array}$$

  In practice, the extension of an eye path must choose between a diffuse or glossy/specular component. Once a diffuse scattering has occurred, further scattering only uses the glossy or specular component.

  The complete expression for this readout strategy is given by $((\mathsf{G}|\mathsf{S})^*\mathsf{D}(\mathsf{G}|\mathsf{S})^*\mathsf{E})$. The total covered transport is the same as for direct visualization, but the solution differs in three aspects:

  – Direct illumination (LXE) is always computed by the image-space pass; no precomputed storage is used.

  – Errors in the stored solution are masked by the indirect visualization. Therefore, a much less accurate and thus faster object-space algorithm can be paired with final gathering.

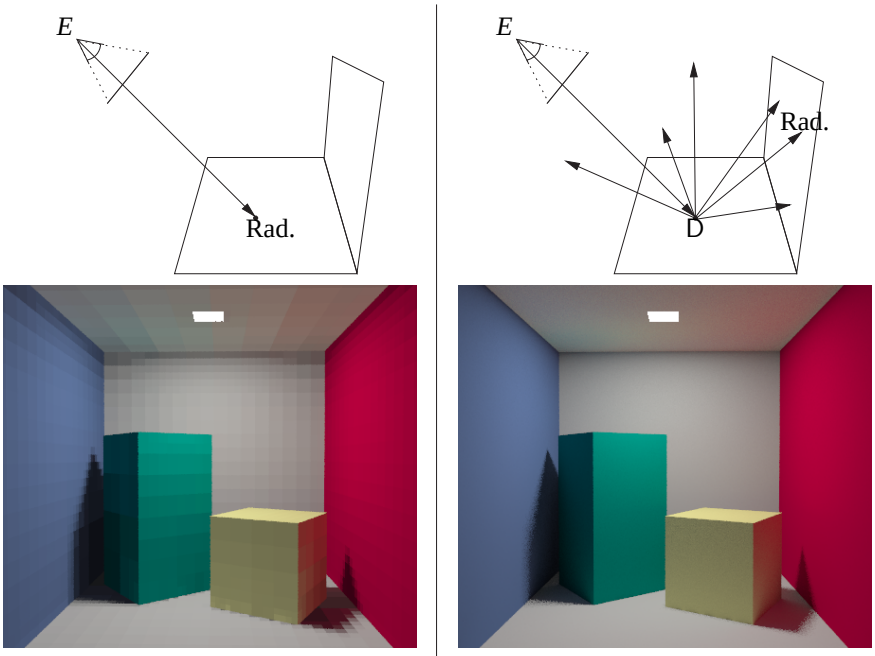


Figure 4.1: Comparison of direct visualization and final gathering for a radiosity solution. The final gather only uses the radiosity solution indirectly. This masks the discretization artefacts, but it takes a much longer time (25 min. versus a few seconds) because many diffusely scattered rays must be traced for an accurate solution.

– The final gathering itself requires much more computation time. Stochastic ray tracing, for example, computes good specular reflections with only a few rays, but many more rays are required for an accurate diffuse reflection.

In figure 4.1 a comparison is made between direct visualization and final gathering. In the left column, a coarse radiosity solution is visualized directly. The visualization took just a few seconds, but the underlying mesh is clearly visible. In the right column, a final gathering is shown. The indirect visualization completely masks all meshing artefacts, but the computation time for this image was around 25 minutes (16 samples/pixel and 64 samples for the diffuse reflection).

4.4 Overview of existing multi-pass configurations

This section gives an overview of the main multi-pass methods presented in the global illumination research literature. We will distinguish between methods that use finite element storage and methods that explicitly store particles. Coincidentally, this also corresponds more or less to a chronological order, since the storage of particles only has become popular in the last few years.

4.4.1 Finite element storage

The most common two-pass method, which we have used throughout the chapter as an example, is a radiosity preprocess (LD^*) followed by a ray tracing pass to compute specular and glossy reflections ($(G|S)^*E$

paths). The resulting image covers $\mathsf{LD}^*(\mathsf{G}|\mathsf{S})^*\mathsf{E}$ paths. Note that glossy/specular to diffuse transport will be missing from the image (e.g., caustics). Several variations have been proposed to remedy this missing transport.

In '87 Wallace et al. [120] presented one of the first two-pass algorithms. It uses a radiosity preprocess and a second, view-dependent pass based on the Z-buffer algorithm.

By rendering a mirrored version of the scene seen through a specular surface, both the radiosity and the Z-buffer pass were extended to include a single, perfectly specular reflection. This technique is limited to a few planar mirrors, because the scene must be rendered for each mirroring surface. The resulting extended radiosity algorithm covers $\mathsf{M} = \mathsf{L}(\mathsf{S}_{\mathsf{r}_\infty}^{0\ldots1}\mathsf{D}_\mathsf{r})^*$ paths: between each two diffuse reflections, a single, perfectly specular reflection ($\mathsf{S}_{\mathsf{r}_\infty}$) is possible.

Their image-space pass covers $\mathsf{Q} = \mathsf{S}_\mathsf{r}^{0\ldots1}\mathsf{E}$ paths and reads out the radiosity solution directly. The total covered transport is $\mathsf{MQ} = \mathsf{L}(\mathsf{S}_{\mathsf{r}_\infty}^{0\ldots1}\mathsf{D}_\mathsf{r})^*\mathsf{S}_\mathsf{r}^{0\ldots1}\mathsf{E}$. This method can generate good looking images, but several restrictions are imposed on the scene model (planar surfaces, perfect mirrors), and important transport paths are missing in the image. Extending their method to recursive reflections would be possible but costly.

Sillion and Puech [97] generalize the previous method by using extended form factors, that are computed by ray tracing. Extended form factors include recursive specular reflections and the ray tracing can handle non-planar specular objects. Classical ray tracing is used as a final pass. The covered transport is $\mathsf{MQ} = \mathsf{L}(\mathsf{S}_{\mathsf{r}_\infty}^*\mathsf{D}_\mathsf{r})^*\mathsf{S}_{\mathsf{r}_\infty}^*\mathsf{E}$, which is the same as Wallace's but with recursive specular reflections.

The methods above use a radiosity method to include indirect diffuse illumination in the images. At the same time, other methods were proposed that compute and store caustics. For example, Arvo [3] used texture light maps to store caustics that were computed by 'backward ray tracing' —tracing paths from the light sources.

Shirley [91] uses a three-pass method that includes both radiosity and caustics:

- In a first particle tracing pass, caustics are computed. They are stored in light maps: high resolution texture maps attached to each diffuse patch in the scene. No distinction is made between glossy and specular materials; the caustics include $\mathsf{M}_\mathsf{c} = \mathsf{L}(\mathsf{G}|\mathsf{S})^+\mathsf{D}$ paths.

- The second pass computes soft indirect illumination. Indirect diffuse illumination is computed using a progressive radiosity algorithm. Extended form factors are computed via ray tracing. Paths covered are $\mathsf{M}_{\text{ind}} = \mathsf{L}(\mathsf{G}|\mathsf{S})^*\mathsf{D}(\mathsf{G}|\mathsf{S}|\mathsf{D})^*\mathsf{D}$.

- A final stochastic ray tracing pass completes the method. Direct illumination is computed on the fly, while the indirect diffuse illumination and the caustics are visualized directly from the stored radiance solutions. This separation between direct and indirect light is seen in many other MPCs. Scattering in the readout strategies only includes the non-diffuse components: $\mathsf{Q} = (\mathsf{G}|\mathsf{S})^*\mathsf{E}$.

The total covered transport of the method is:

$$(M_c + M_{ind} + L + LD)(G|S)^*E) = L(G|S|D)^*E\,.$$

In theory this method covers all light transport. The main drawbacks are a high preprocessing time and the use of light texture maps. These maps are mesh dependent and non-hierarchical. Convergence can be slow if the caustics are not localized in small parts of the scene.

Heckbert [40] suggests a three-pass method: A 'size' pass from the eye records screen size information in the scene, that can be used to guide subdivision in a subsequent light pass. This light pass traces particles from the light sources and records radiosity in adaptive textures on diffuse surfaces. The light pass handles perfectly specular reflections storing $(LS^*_{r_\infty}D)$ paths, so that the radiosity textures include caustics. The textures are subdivided adaptively (see also §7.8.2). Particles can also be emitted from lit textures to simulate diffuse interreflections ($L(S^*_{r_\infty}D)^*$ paths). A final image-space pass uses classical ray tracing to read out the radiosity textures. This method covers all specular and diffuse interreflections. The implementation, unfortunately, did not include the size pass nor the diffuse interreflections, limiting the results to direct diffuse illumination, caustics and specular reflections from the eye.

Chen et al. [18] recognized that it is hard to remove all meshing artefacts in a radiosity solution. Therefore, they propose a progressive algorithm that uses radiosity for early feedback, but resorts to brute force stochastic ray tracing and particle tracing for the final high-quality solutions.

In the final image, caustics are computed by particle tracing (stored in light maps), and all other illumination is computed by stochastic ray tracing. The radiosity solution is used for paths that were already reflected diffusely, signaling the first use of a final gathering method.

The final image produced by this method was not much faster than plain stochastic ray tracing, but a decent image was available early on for fast feedback to the user. Rushmeier [84] improved on this progressive method by using simplified geometry to accelerate the radiosity algorithm.

It must be noted that, when these multi-pass methods were proposed, it was recognized that they were quite accurate but very slow. The separation of light transport that they proposed, however, is still used in many multi-pass algorithms today. Large improvements in radiosity and particle tracing methods and much faster computers, now have made such multi-pass algorithms most popular.

4.4.2 Particle storage

Several recent methods store illumination information in the form of particles instead of using light maps or meshes. Since the particles can be stored independently from the underlying geometry, meshing artefacts can be avoided.

In '95, Shirley et al. [93] presented density estimation. Particle tracing (using analog simulation) stores particles on diffuse surfaces only. Using density estimation, an optimized mesh containing the stored il-

lumination (LX^*D) is constructed. Stochastic ray tracing completes the method, tracing $((G|S)^*E)$ paths. Since the stored illumination is visualized directly, a very high number of particles is needed for an accurate solution. Several improved versions of density estimation have been presented [122, 121].

Jensen introduced a slightly different method using photon maps [47, 50]. Particles hitting diffuse but also glossy surfaces are stored in a kd-tree, which is independent of the underlying scene geometry. The stored transport covers ($M_g = LX^*(G|D)$) paths. A separate, high-resolution caustic map only stores particles contributing to caustic effects ($M_c = LS^+(G|D)$). Nearest neighbor density estimation is used to reconstruct illumination directly from the particle representation. Again, stochastic ray tracing is used in the final pass, but the stored illumination M_g is only used indirectly after a final gather, requiring less particles. The caustic map is visualized directly, because caustics cannot be reproduced well using eye paths. This promising and popular method is described in detail in chapter 8. Note that the separation is very similar to Chen's method.

Keller presented another multi-pass method that stores a number of particles, which are then used as virtual light sources in the second pass [56]. The original algorithm only supported diffuse illumination. In the image-space pass, graphics hardware can be used to render the scene lit with each of the particles separately. Adding all these images delivers the final image. Due to the restricted dynamic range of graphics hardware (only 8-bits per color channel at the time), the accuracy was limited.

Recently this method was used together with a fast, coherent ray tracer [119]. Using full floating point arithmetic, without the limitations of the graphical hardware, a decent, full global illumination solution can be obtained.

Another multi-pass approach splats the particles directly to the screen [107] to obtain glossy illumination. This method also uses hardware rendering as a final pass. While the method covers all illumination, the accuracy is quite limited due to the hardware rendering.

4.4.3 Lighting networks

Whereas all previous methods proposed one specific multi-pass algorithm, lighting networks provides a framework for developing multi-pass methods. Lighting networks were presented by Slusallek et al. [99, 100]. The framework allows a user to combine partial transport operators into arbitrary multi-pass algorithms. The transport operators can be restricted to only a part of the scene, and storage conversion operators are provided to facilitate the connection of different partial light transport operators. Regular expression operations allow to test if there is redundant or missing transport in the network. While this framework allows many different configurations, the task of configuring an efficient lighting network is difficult and requires an experienced user.

Our regular expression framework is a subset of the lighting networks framework. The improvements

presented in the next chapter, could also be implemented into the framework of lighting networks.

4.5 Conclusion

This chapter introduced multi-pass methods. A simple framework was presented that is based on regular expressions to classify different multi-pass methods. A discussion of common object-space and image-space components was given, together with an overview of the current multi-pass research. The key to a good multi-pass method is an intelligent separation of light transport, so that the strengths of the different components are preserved.

5 Regular expression based path evaluation

In the previous chapter, regular expressions were used to describe the light transport covered by specific multi-pass components. This was convenient, because overlapping or missing transport could be identified by these expressions. In this chapter, we turn this concept around: the covered transport of the components is *determined* directly from an arbitrary regular expression that is supplied by the user.

The technique presented in this chapter is simple but practical. It can be used with any algorithm that is based on path sampling. The user just has to supply a textual regular expression for the paths that need to be covered (for example in an image-space path tracing pass) and the evaluation of the paths will automatically adapt to the requested transport.

In §5.1, a simple example of an enhanced combination of radiosity and path tracing will illustrate that this requires a decoupling of the path sampling and the path evaluation, because in a single path vertex different combinations of BSDF components may be required in the path evaluation.

Section 5.2 offers details on how a regular expression based path evaluation can be implemented efficiently.

Section 5.3 explains the added flexibility that a regular expression based path evaluation brings to the design of more advanced multi-pass configurations. One important consequence is that bidirectional path tracing can now be easily added to a multi-pass configuration.

Results in §5.4 show a detailed combination of radiosity and bidirectional path tracing. The regular expressions are used to fine-tune the combination so that each part of the light transport is assigned to the most appropriate method. Section 5.5 concludes this chapter.

The technique presented in this chapter was previously presented in [P3]. It will also be used in the next chapter on weighted multi-pass methods.

5.1 An enhanced combination of radiosity and path tracing

A common multi-pass method is the combination of radiosity and path tracing (see also chapter 4). Radiosity computes all the diffuse interreflections and path tracing fills in the glossy and specular reflection (and refraction) from the eye. The scattering in path tracing, however, is restricted to the $(\mathsf{G}|\mathsf{S})$ components, because inclusion of the diffuse component results in the computation of redundant transport (see §4.3.2). The paths covered by the combination are $((LD^*(G|S)^*E)$, which misses some transport.

In this section we present a simple enhanced readout strategy that does result in a full global illumination

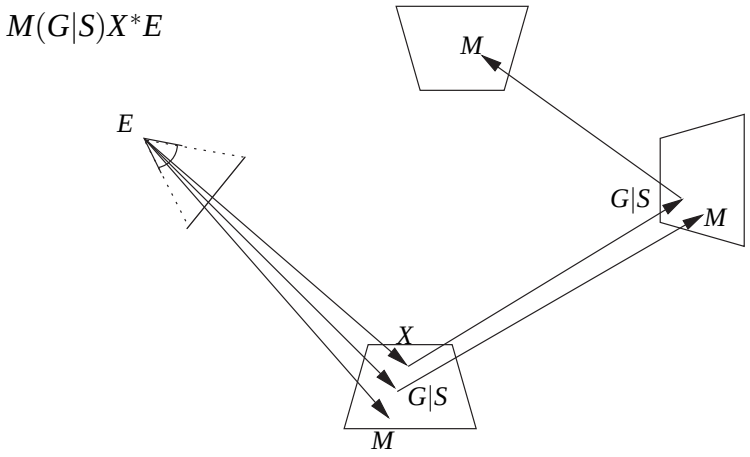


Figure 5.1: Path evaluations based on regular expressions can require different scattering components for the BSDF evaluations in a single vertex. This figure shows the evaluation of an eye path of length 4 and all its sub-paths according to the regular expression $\mathsf{M(G|S)X^*E}$.

solution. The paths covered by path tracing are defined by the regular expression $((G|S)X^*E+E)$.

Enumerating the contribution of paths with this regular expression gives:

$$\begin{array}{lccccc} 2: & & & & \mathsf{M}_{rad} & E \\ 3: & & & \mathsf{M}_{rad} & (G|S) & E \\ 4: & & \mathsf{M}_{rad} & (G|S) & (X) & E \\ 5: & \mathsf{M}_{rad} & (G|S) & (X) & (X) & E \\ & \ldots & & & & \end{array} \tag{5.1}$$

In this readout strategy all the BSDF components (X) are taken into account, except for vertex $(k-1)$ in a path of length k. With this enhanced readout strategy, the image contains all possible illumination $\mathsf{LX^*E}$. This can be shown as follows (with M_{rad} substituted by $\mathsf{LD^*}$):

- Combining the first rows in equation 5.1 gives:

$$2+3: \mathsf{LD^*E+LD^*(G|S)E = LE+LD^*DE+LD^*(G|S)E} \\ \mathsf{= LE+LD^*(D|G|S)E = LD^*X^{0\ldots1}E}.$$

- When adding the following rows, the range of the exponent of X increases, leading to the following total covered transport:

$$2+3+\ldots: \mathsf{LD^*X^{0\ldots\infty}E = LD^*X^*E = LX^*E}.$$

Thus a slight modification of the readout strategy, changes the covered transport so that all light transport is computed.

While the extension in this example is simple, the important thing to note is that different scattering components are used for the same vertex depending on the length of the path evaluated. This is illustrated in figure 5.1. An eye path of length 4 and its evaluated sub-paths are shown. The same vertex sometimes

includes all scattering components, sometimes only glossy and specular. Therefore, the BSDF evaluations must be done when the path is evaluated and cannot be determined and fixed when sampling the new directions (all components are taken into account in the pdf when sampling a direction). This decoupling is one the important aspects of the regular expression based path evaluation.

An example of the enhanced combination will be shown in §5.4.3.

5.2 Implementation

To efficiently evaluate paths according to a (textual) regular expression, some changes need to be made to the implementation of the path construction and evaluation.

5.2.1 Path construction

When evaluating paths with regular expressions, different BSDF components may need to be used in a single vertex in the path (as was illustrated in the example in the previous section).

This has two consequences for path construction:

- Since the exact BSDF components for scattering in a vertex are not known, the pdf for sampling a new direction should include all components of the BSDF $(\mathsf{D}|\mathsf{G}|\mathsf{S})$.
- The BSDF evaluation is stored separately for each component. This allows a fast evaluation of any combination of components later on. The overhead compared to a standard BSDF evaluation is small, since most BSDFs have separate code for different components anyway.

Thus, after a path is sampled, a list of BSDF evaluations for each separate component is available in each path vertex[1].

5.2.2 Path evaluation

To prevent expensive operations on the textual regular expression each time a path is evaluated, a simple data structure is precomputed.

For every possible path length, a list of 'simple' regular expressions is constructed. A simple regular expression does not contain any enumeration operators such as $+$ or $*$. Such a simple expression will have as many elements as there are vertices in the path. To construct the list, all possible expansions of the complex regular expression are enumerated. For example, the list for the expression $\mathsf{L}\mathsf{D}^*(\mathsf{G}|\mathsf{S})^*\mathsf{E}$ and a path of length 4 would look like:

$$\begin{array}{cccc} L & D & D & E \\ L & D & (G|S) & E \\ L & (G|S) & (G|S) & E \end{array}$$

[1]Note that if some of the components of the BSDF are not used in the evaluation, the pdf used for sampling the direction may be slightly sub-optimal, because these unused components were included in the sampling. Although, in some cases, unused components in a certain vertex may be deduced beforehand from the regular expressions, the benefits of this would be small compared to the additional implementation.

The evaluation of a path will iterate over the list, gather the necessary BSDF evaluations, and add the results. Note that other factors in the path evaluation that do not depend on the regular expressions, such as cosine factors and the self emitted radiance, can be multiplied once, separate from the BSDF evaluations.

Such a list of simple expressions is kept for each different path length. In our implementation, a maximum path length is specified that limits the number of interreflections and thus also the number of lists. Of course, higher order reflections are missing, but this is not a problem for a sufficiently high maximum path length.

The evaluation lists and the separate BSDF components limit the overhead of regular expression based path evaluation to about $5-10\%$ in our moderately optimized implementation.

5.3 Consequences of the path evaluation from regular expressions

As a consequence of the path evaluation technique, we can now allow arbitrary readout strategies by deriving them directly from the regular expressions that determine the covered transport.

We use it to tune the covered transport in image-space passes. For each stored radiance solution (SPAR), a (textual) regular expression can be supplied and parsed into a readout strategy. The paths constructed during the image-space pass are evaluated using this readout strategy and the correct scattering components are determined for each vertex.

This approach has several advantages:

- The path evaluation is decoupled from the path construction. The BSDF components to use, which are dictated by the regular expression, are fixed at the time of evaluation and independent of how the path was constructed.

  This decoupling enables the usage of other path sampling techniques, such as bidirectional path tracing (BPT). The use of BPT in multi-pass methods was in fact the main motivation for the regular expression based framework [P3].

- Additionally the support for arbitrary readout strategies allows for flexible fine tuning of multi-pass configurations:

  – Missing transport in a MPC can be identified by the regular expressions. Given several SPARs M_i and their readout strategies Q_i, any missing transport is given by:

  $$(\mathsf{LX}^*\mathsf{E}) \setminus (\bigcup_i \mathsf{M}_i\mathsf{Q}_i)\,.$$

  This expression is a regular expression in itself and the necessary contributions can be computed with path tracing or bidirectional path tracing by specifying the self-emitted radiance as a SPAR

and the above expression as the readout strategy. This makes it very easy to add the missing transport to existing MPCs.

  - Instead of adding missing transport, one can also remove some specific light transport. For example it could be transferred to another, better method or even completely removed from the image if it is an important source of spike noise. The latter results, of course, in a biased solution.

- Although not important for robust rendering algorithms per se, the framework provides a nice didactical tool for analyzing light transport. It is easy to separately render specific light transport features such as caustics, indirect light, or paths of a specific length.

The extensive example in the next section, a combination of bidirectional path tracing and Galerkin radiosity, will demonstrate some of these benefits.

5.4 Combining radiosity and bidirectional path tracing

The flexibility of the regular expression based path evaluation will be demonstrated by an extensive example: the combination of radiosity and bidirectional path tracing.

Starting from a plain radiosity solution (§5.4.1), the method will be gradually improved from a traditional two-pass method using path tracing (§5.4.2) and an enhanced two-pass method covering all light transport (§5.4.3) to the combination with bidirectional path tracing (§5.4.4) and an optimization for indirect caustics (§5.4.5). Each time a different allocation of light transport over different algorithms is used, showing the flexibility of the framework.

Note that in our implementation, all the separations in this example can be specified with just a few textual regular expressions.

5.4.1 Radiosity only

A higher order Galerkin radiosity method with hierarchical meshing and clustering is used. An accurate cubature rule for form factor computation and a low maximum-link-error threshold ensure a high-quality solution that can be visualized directly [5].

In figure 5.2 the radiosity solution is shown for the same scene that was used in chapter 3 (around 10 minutes to compute this accurate solution). Clearly this solution ((LD^*E) paths) is missing important light transport.

For comparison purposes, an image computed with bidirectional path tracing is shown in figure 5.3. While the indirect diffuse illumination is noisy compared to the radiosity solution, this image includes all

light transport. The most apparent differences are the caustic effects and the fact that we can see through the glass sphere.

5.4.2 Traditional two-pass method

Figure 5.4 shows an image computed with a traditional two-pass method. Path tracing with only glossy and specular scattering is combined with the radiosity solution. For this image 9 paths per pixel are used. It took a few minutes (on top of the radiosity computation) to compute this image. All subsequent image-space passes take about the same computation time.

The image covers $\mathsf{LD}^*(\mathsf{G}|\mathsf{S})^*\mathsf{E}$ paths. While refraction through the sphere is handled correctly, caustics are still missing.

5.4.3 Enhanced two-pass method

In §5.1 an enhanced readout strategy for the path tracing pass was given, which covers $((\mathsf{G}|\mathsf{S})(\mathsf{X})^*\mathsf{E}+\mathsf{E})$ paths. Combined with the radiosity solution it was shown that all light transport is covered.

Figure 5.5 shows an image computed using this enhanced readout strategy (9 samples/pixel). While all light transport is covered, the image exhibits a very high noise level, because the eye paths are not adequate for all illumination features. For example, the caustic effects are present, but they are very noisy, because, as explained in §3.4.1, eye paths do not sample these effects well.

Therefore, it would be interesting to combine bidirectional path tracing with the radiosity solution.

5.4.4 Bidirectional path tracing

Due to the regular expression based path evaluation, it is easy to incorporate bidirectional path tracing into the multi-pass configuration. We will keep using path tracing to read out the radiosity solution, but we will also use bidirectional path tracing (BPT), which reads out the self-emitted radiance. The eye paths, that are traced for the bidirectional paths, are used simultaneously for path tracing.

To combine BPT and the radiosity solution we will transfer some light transport from the radiosity-path tracing combination to BPT. From the covered transport of radiosity (LD^*) and path tracing ($((\mathsf{G}|\mathsf{S})(\mathsf{X})^*\mathsf{E}+\mathsf{E})$ it follows that ($\mathsf{L}(\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E}$) transport is computed by eye paths that directly hit a light source.

This transport can be transferred to BPT easily. The stored radiosity solution is modified so that it does not include the self-emitted radiance L_e, thus covering only (LD^+) paths. Radiosity with path tracing now covers ($\mathsf{LD}^+(\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E}$). The remaining transport, that includes the specular to diffuse transport, is handled by BPT (($\mathsf{L}(\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E}+\mathsf{E}$) paths).

Figure 5.6 shows the image computed with this new multi-pass configuration. Only 4 paths per pixel are traced, because the bidirectional paths are more expensive.

The left caustic comes out much better, but the one on the right is still noisy. This is because it is an indirect caustic. It is caused by the glass sphere, that focuses the bright diffuse illumination of the white panel onto the floor. This effect consists mainly of $\mathsf{LDS}^+\mathsf{DE}$ paths, still handled by eye paths that are reflected diffusely first, and then travel through the sphere until they hit the white panel. Clearly, eye paths do not sample the effect well.

5.4.5 Indirect caustics optimization

Since the indirect caustic is identified as $\mathsf{LDS}^+\mathsf{DE}$ transport, the configuration can again be adjusted so that this transport is handled by BPT. This requires a separate storage of direct diffuse (LD) and indirect diffuse (LDD^+) illumination in the radiosity solution.

The optimized configuration now has three different SPARs:

- L: Self-emitted radiance combined with BPT, covering $(\mathsf{L}(\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E})$ plus all $(\mathsf{LDS}^*\mathsf{DE})$ paths.
- $\mathsf{M}_{\mathsf{LD}} = \mathsf{LD}$: Direct diffuse illumination combined with path tracing, covering the following paths: $(\mathsf{M}_{\mathsf{LD}}((\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E}+\mathsf{E}))$ minus $(\mathsf{M}_{\mathsf{LD}}\mathsf{S}^+\mathsf{DE})$.
- $\mathsf{M}_{\mathsf{ind}} = \mathsf{LDD}^+$: Indirect diffuse illumination, which is also combined with path tracing, covering $(\mathsf{M}_{\mathsf{ind}}((\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E}+\mathsf{E}))$ paths.

In total, all light transport is covered. Figure 5.7 was computed using this configuration, and it is clear that the indirect caustic is handled much better. This example shows that a flexible separation enable the design of better MPCs.

However, in other circumstances, for example if the diffuse panel were larger, path tracing using the radiosity solution might become better. In the next chapter a technique is presented that weights such overlapping transport so that an extreme separation is not necessary.

5.5 Conclusion

The careful separation of light transport presented in this chapter shows that regular expression based path evaluation is a valuable tool in designing multi-pass configurations. Since most existing multi-pass methods resort to path tracing as a final image-space pass, the regular expression based approach would be an interesting addition to these methods. An additional benefit is that bidirectional path tracing can now be used to compute part of the light transport.

The careful separation in the radiosity–bidirectional path tracing example also raises an interesting point: it is not always obvious, for a specific part of the light transport, to which algorithm or combination of algorithms it must be assigned. In the next chapter a technique is presented that makes a weighted combination of methods, which preserves the strengths of the methods even within overlapping transport.

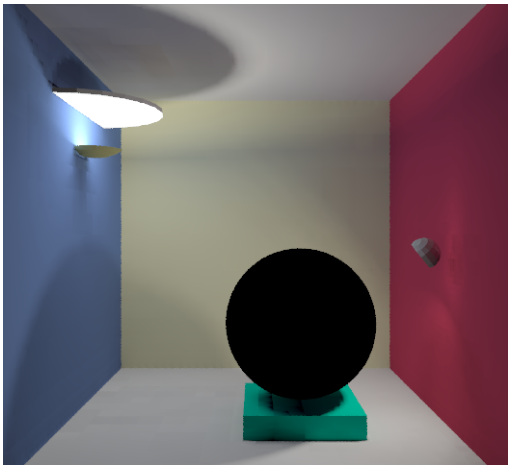


Figure 5.2: Radiosity solution only

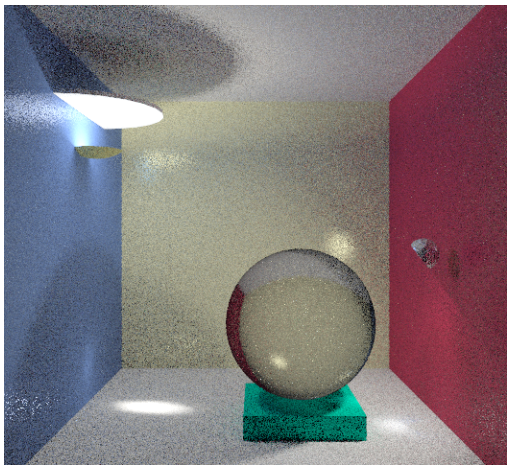


Figure 5.3: Plain bidirectional path tracing (BPT) (4 s/p)

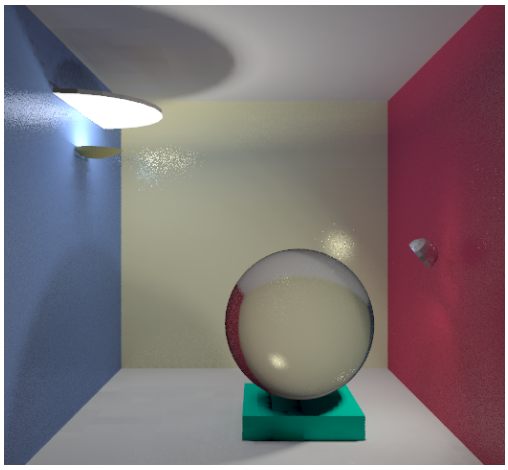


Figure 5.4: Classic two-pass method (9 s/p): Radiosity + Path tracing. Only $\mathsf{LD}^*(\mathsf{G}|\mathsf{S})^*\mathsf{E}$ paths are covered; other transport is missing

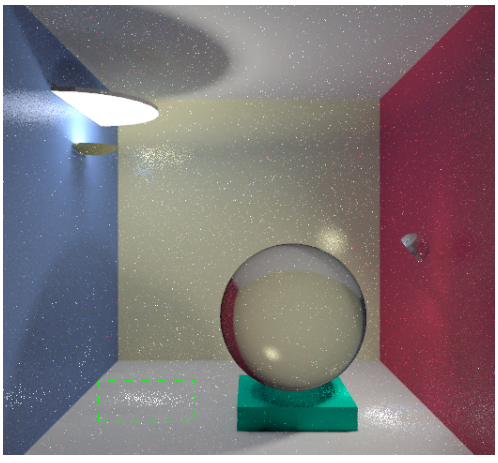


Figure 5.5: Enhanced two-pass method (9 s/p): path tracing covers $((\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E})+\mathsf{E}$ paths, so that all illumination is present in the (noisy) image.

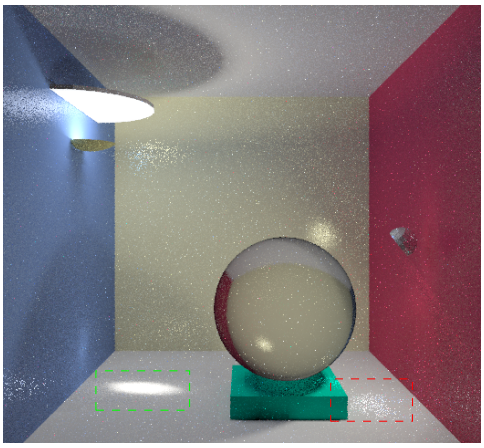


Figure 5.6: Radiosity and BPT (4 s/p): Caustics are computed with BPT and come out much better.

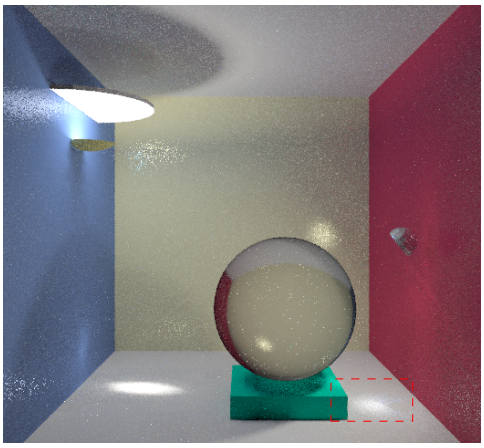


Figure 5.7: Radiosity and BPT with indirect caustic optimization (4 s/p): The right caustic originates from the white diffuse panel and is now also computed using BPT.

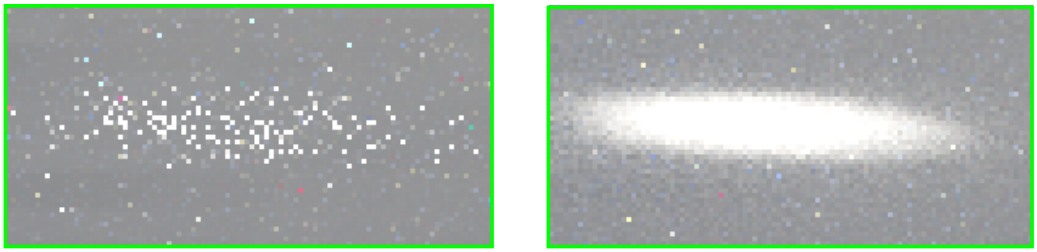


Figure 5.8: Magnification of the caustic from figure 5.5 (left) and figure 5.6 (right).

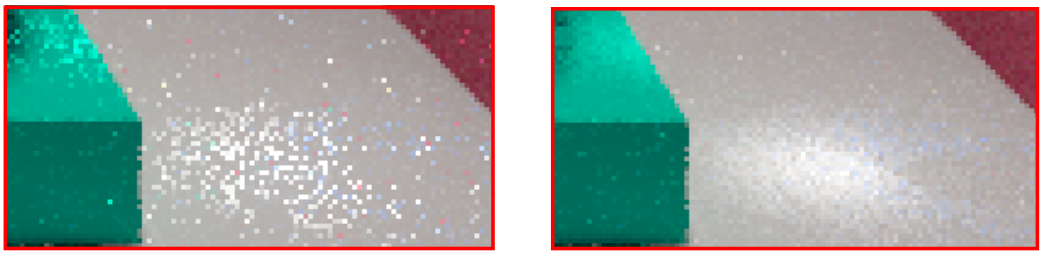


Figure 5.9: Magnification of the indirect caustic from figure 5.6 (left) and figure 5.7 (right).

6 Weighted Multi-Pass Methods

Previous multi-pass methods separate light transport in disjunct parts, and assign each part to a specific algorithm or combination of algorithms. In this chapter a new technique is presented that allows us to make a weighted combination of specific overlapping light transport.

6.1 Introduction

In the previous chapter, a detailed example showed a manually tuned separation of light transport for the combination of bidirectional path tracing (BPT) and radiosity. While the use of regular expressions allowed to transfer indirect caustics to BPT to improve the image, it would be interesting if the multi-pass method was able to *automatically* assign the transport to the most appropriate method.

One way to achieve such an automatic procedure is to use weighting instead of separation. The key point of weighting is to allow overlapping transport between different methods in a multi-pass configuration (MPC), but to assign a higher weight to the contribution of the most appropriate method.

The problem of preserving the strengths of different rendering methods changes from finding a good separation to finding good weighting heuristics. A single, constant weight for each method that computes overlapping transport, for example, will not be better (and even worse) than separation because such a weight cannot differentiate between the different illumination features within the overlapping transport.

Weighted multi-pass methods, introduced in this chapter, provide an interesting, theoretical framework for developing good weighting heuristics. The weighting technique has the following characteristics:

- The technique applies to MPCs that use Monte Carlo path sampling techniques (path tracing, BPT) for their final pass.
- Weights are based on individual paths generated in the image-space pass. Good weighting heuristics ensure that the final contribution will be small for ‘bad’ paths, that would cause a high variance in the image.
- The weighting heuristics should ensure an unbiased, correct solution. Therefore, constraints are developed that must be satisfied by the heuristics.

The benefit of weighting individual paths over separation is apparent:

- Separation must choose a single method to assign illumination features. Each separated feature will cover a whole set of paths.
- The weighting of paths can differentiate good and bad paths even within overlapping illumination.

- Different methods can contribute to the same illumination. For example, if a difficult feature is not handled well by any of the methods, the total weight will be evenly distributed between them so that they all contribute to the feature.

The theory of weighted multi-pass methods is developed in §6.2. A generalization of multiple importance sampling is presented that handles partially precomputed integrals. Constraints to ensure an unbiased solution and appropriate weighting heuristics are derived. This generalization of multiple importance sampling is not restricted to graphics, but is applicable whenever the method of expected values and multiple importance sampling are combined.

In §6.3, the application of the weighting theory to multi-pass methods is discussed in general. The weighted combination of radiosity and bidirectional path tracing in §6.4, shows a practical example of the weighting theory. This example shows that weighting allows the development of more robust and better multi-pass configurations, but it also shows that this is not an easy task and that the weights must be handled carefully. General conclusions are discussed in §6.5.

We presented the theory and application of weighted multi-pass methods previously in [P4], which is the first paper that used weighting instead of separation in a multi-pass method. This chapter provides a more in-depth discussion of the method and suggests further extensions and applications (see §6.5.2).

6.2 Multiple importance sampling and the method of expected values

Multiple importance sampling applies when different pdfs are used to estimate the same integral (§3.3.2). For example in bidirectional path tracing, the different methods for constructing paths correspond to different path sampling pdfs, and for each path an appropriate weight must be computed. The weight of a single path depends on all the techniques that can generate the exact same path.

The problem dealt with in this chapter is different, because some sampling techniques use precomputed, partially integrated results. These techniques actually use the method of expected values to reduce the dimensionality of the integration problem. This is shown schematically in figure 6.1 using integration over paths as an example. Both paths compute the same integral, but the shorter path uses precomputed information. Although the shorter path estimates a lower dimensional integral, its variance may be higher than that of the longer path. This is the case when its pdf does not fit the remaining integrand well.

In this section we will generalize multiple importance sampling to accommodate partially precomputed integrals. The theory will be derived using two pdfs and extended later on to an arbitrary number of pdfs and precomputed results.

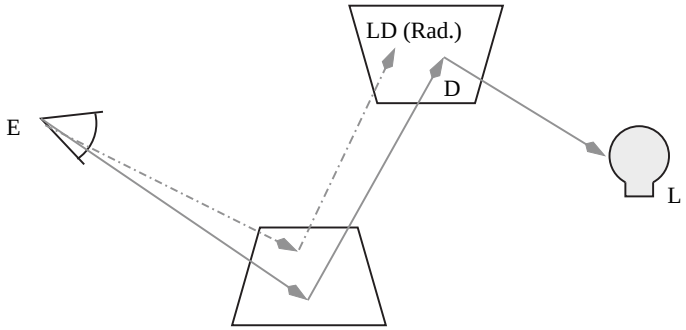


Figure 6.1: Two differently sampled paths that compute the same transport, although their lengths differ. The shorter, dashed path uses a precomputed radiosity solution. This difference must be taken into account when weighting the paths.

6.2.1 Problem statement

Suppose we want to compute the following integral :

$$I = \int_{\Omega_y} \int_{\Omega_x} f_1(x,y)\,\mathrm{d}x\,\mathrm{d}y\,. \tag{6.1}$$

Both variables x and y may be multi-dimensional vectors.

Define a function f_2 that is the result of partially integrating f_1 over the domain Ω_x:

$$f_2(y) = \int_{\Omega_x} f_1(x,y)\,\mathrm{d}y\,. \tag{6.2}$$

The integral I can also be computed using f_2:

$$I = \int_{\Omega_y} f_2(y)\,\mathrm{d}y\,. \tag{6.3}$$

The integrals (6.1) and (6.3) can be estimated with Monte Carlo integration, using importance sampling according to the pdfs $p_1(x,y)$ and $p_2(y)$ respectively. This leads to two estimators for I :

$$\langle I\rangle_1 = \frac{1}{N_1}\sum_{i=1}^{N_1}\frac{f_1(x_i,y_i)}{p_1(x_i,y_i)}\,, \tag{6.4}$$

$$\langle I\rangle_2 = \frac{1}{N_2}\sum_{j=1}^{N_2}\frac{f_2(y_j)}{p_2(y_j)}\,. \tag{6.5}$$

These estimators compute the same integral. At first sight one would say that (6.5) would be most efficient to compute I, since the precomputed part of the integral is exact and a lower dimensional integral is computed. However p_1 and p_2 have an important influence on the variance of the estimator. For some sub-domain of $\Omega_x \times \Omega_y$, p_1 can be much more efficient than p_2 in the corresponding sub-domain of Ω_y. Therefore, we would like to use weighted estimators.

6.2.2 Weighted estimators

To combine the two different estimators, each sample is multiplied by a weighting function, that depends on the sample itself:

$$\langle I\rangle_c = \frac{1}{N_1}\sum_{i=1}^{N_1} w_1(x_i,y_i)\,\frac{f_1(x_i,y_i)}{p_1(x_i,y_i)} + \frac{1}{N_2}\sum_{j=1}^{N_2} w_2(y_j)\,\frac{f_2(y_j)}{p_2(y_j)}\,. \tag{6.6}$$

This equation represents a whole class of estimators, instantiated by a particular choice of the weighting functions.

6.2.3 Weighting constraints

Not all weighting functions will deliver a correct result. Weighting constraints can be derived by requiring the combined estimator to be unbiased:

$$E\left[\langle I\rangle_c\right] = I = \int_{\Omega_x}\int_{\Omega_y} f_1(x,y)\,\mathrm{d}x\,\mathrm{d}y\,.$$

Computing the expected value gives

$$\begin{aligned}
E\left[\langle I\rangle_c\right] &= E\left[\frac{1}{N_1}\sum_{i=1}^{N_1} w_1(x_i,y_i)\frac{f_1(x_i,y_i)}{p_1(x_i,y_i)} + \frac{1}{N_2}\sum_{j=1}^{N_2} w_2(y_j)\frac{f_2(y_j)}{p_2(y_j)}\right]\\
&= \frac{1}{N_1}\sum_{i=1}^{N_1} E\left[w_1(x_i,y_i)\frac{f_1(x_i,y_i)}{p_1(x_i,y_i)}\right] + \frac{1}{N_2}\sum_{j=1}^{N_2} E\left[w_2(y_j)\frac{f_2(y_j)}{p_2(y_j)}\right]\\
&= \frac{1}{N_1}N_1\int_{\Omega_x}\int_{\Omega_y} w_1(x,y)\frac{f_1(x,y)}{p_1(x,y)}\,p_1(x,y)\,\mathrm{d}x\,\mathrm{d}y\\
&\quad + \frac{1}{N_2}N_2\int_{\Omega_y} w_2(y)\frac{f_2(y)}{p_2(y)}\,p_2(y)\,\mathrm{d}y\\
&= \int_{\Omega_y}\left(\left(\int_{\Omega_x} w_1(x,y)\,f_1(x,y)\,\mathrm{d}x\right) + w_2(y)\,f_2(y)\right)\mathrm{d}y\\
&\overset{(6.2)}{=} \int_{\Omega_y}\left(\int_{\Omega_x} w_1(x,y)\,f_1(x,y)\,\mathrm{d}x \;+\; w_2(y)\int_{\Omega_x} f_1(x,y)\,\mathrm{d}x\right)\mathrm{d}y\\
&= \int_{\Omega_x}\int_{\Omega_y}\left(w_1(x,y)+w_2(y)\right)f_1(x,y)\,\mathrm{d}x\,\mathrm{d}y\,.
\end{aligned}$$

To ensure an unbiased solution, the following constraint must hold:

$$\forall x\in\Omega_x, y\in\Omega_y : w_1(x,y)+w_2(y) = 1\,. \tag{6.7}$$

Note the similarity with the multiple importance sampling constraint given in equation (3.2) ($\sum_{k=1}^{n} w_k(x) = 1$).

This constraint ensures that the total contribution of a sample (x,y) is not too large or too small. However, the constraint must hold for any value of x and y. Since y determines $w_2(y)$, it implicitly determines

$w_1(x,y)$ through the constraint. Therefore, the weight $w_1(x,y)$ *should not depend on* x. From now on the notation $w_1(y)$ will be used.

This is an important constraint that will have a great influence on the weighting heuristics. It has some interesting implications:

- If Ω_x covers a large part of the total domain $\Omega_x \times \Omega_y$ (e.g., 8 dimensions out of a 10-dimensional domain), the weights, only depending on y, will be rather coarse. Therefore, best results are expected when Ω_x is small compared to Ω_y.

- For samples (x_i,y_i) which are drawn according to pdf p_1, the sampled value y_i may depend on the value of x_i (i.e., when x_i is determined first and the sampling of y_i depends on that result). To evaluate the weighting heuristic $w_1(y)$, *all* dependencies on x must be removed. This requires a careful analysis of the sampling procedure for $p_1(x,y)$.

  As we will see in the application (§6.4), one way to remove these dependencies is to compute a partial weight factor in the preprocessing pass where the integral over Ω_x is computed.

These implications will manifest themselves when applying the theory to multi-pass algorithms.

For standard multiple importance sampling, another constraint was set for the weighting heuristics (eq. 3.2 (2), p. 28): $p_k(x) = 0 \Rightarrow w_k(x) = 0$. This constraint implies that the weight of a sample must be zero if the sample cannot be generated with the corresponding pdf. This constraint is only important when some of the pdfs do not cover the whole domain of the integrand.

A corresponding constraint for our generalization can be derived and is given by

$$\forall y \in \Omega_y : \begin{cases} \int_{\Omega_x} p_1(x,y)\,\mathrm{d}x = 0 \;\Rightarrow\; w_1(y) = 0 \\ \int_{\Omega_x} p_1(x,y)\,\mathrm{d}x \neq 0 \;\Rightarrow\; \forall x \in \Omega_x : p_1(x,y) \neq 0 \\ p_2(y) = 0 \;\Rightarrow\; w_2(y) = 0 \end{cases} \tag{6.8}$$

Similar to standard multiple importance sampling, the weights must be zero if the pdf cannot generate the sample. For example, if $p_2(y) = 0$ for a certain point y, this point will never be generated by p_2 and its weight $w_2(y)$ must be zero.

The corresponding constraint for $p_1(x,y)$ and $w_1(y)$ is slightly different, because $w_1(y)$ cannot depend on x. Therefore, the complete domain Ω_x must be taken into account in the constraint. The first part states that, if a certain y cannot be generated by $p_1(x,y)$ (for any value of x, hence the integral), the weight $w_1(y)$ must be zero. The second part, on the other hand, states that if y *can* be generated, then, for that value of y, $p_1(x,y)$ must cover the whole domain Ω_x. This is because the weight $w_1(y)$ is not able to differentiate between the parts of Ω_x that are and are not covered by p_1.

This second constraint can be derived by computing the expected value of the combined estimator while taking into account that the pdfs do not cover the complete domain. In our application, the pdfs do cover the whole domain, so that this constraint is of less interest for the rest of this chapter.

6.2.4 Weighting heuristics

Now that constraints on the weighting functions are set, we can look for good weighting heuristics. The optimal weighting functions will minimize the variance of the combined estimator. First, an expression for the variance will be established (§6.2.4.1). Then we will (partially) minimize the variance, and derive weighting heuristics from the minimization (§6.2.4.2).

6.2.4.1 Variance

The variance of the combined estimator is given by

$$\begin{aligned}
V\left[\langle I\rangle_c\right] &= \frac{1}{N_1^2}\sum_{i=1}^{N_1} V\left[w_1(y_i)\frac{f_1(x_i,y_i)}{p_1(x_i,y_i)}\right] + \frac{1}{N_2^2}\sum_{j=1}^{N_2} V\left[w_2(y_j)\frac{f_2(y_j)}{p_2(y_j)}\right] \\
&= \frac{1}{N_1^2}N_1\left(\int_{\Omega_y}\int_{\Omega_x} w_1^2(y)\frac{f_1^2(x,y)}{p_1^2(x,y)}p_1(x,y)\,\mathrm{d}x\,\mathrm{d}y \right. \\
&\qquad \left. -\left(\int_{\Omega_y}\int_{\Omega_x} w_1(y)\,f_1(x,y)\,\mathrm{d}x\,\mathrm{d}y\right)^2\right) + \\
&\quad \frac{1}{N_2^2}N_2\left(\int_{\Omega_y} w_2^2(y)\frac{f_2^2(y)}{p_2^2(y)}p_2(y)\,\mathrm{d}y - \left(\int_{\Omega_y} w_1(y)\,f_2(y)\,\mathrm{d}y\right)^2\right) \qquad (6.9) \\
&= \int_{\Omega_y}\left(\frac{w_1(y)^2}{N_1}\int_{\Omega_x}\frac{f_1(x,y)^2}{p_1(x,y)}\,\mathrm{d}x + \frac{w_2(y)^2}{N_2}\frac{\left(\int_{\Omega_x} f_1(x,y)\,\mathrm{d}x\right)^2}{p_2(y)}\right)\mathrm{d}y \\
&\quad -\left(\frac{1}{N_1}\left(\int_{\Omega_y}\int_{\Omega_x} w_1(y)\,f_1(x,y)\,\mathrm{d}x\,\mathrm{d}y\right)^2\right. \\
&\qquad \left. +\frac{1}{N_2}\left(\int_{\Omega_y}\int_{\Omega_x} w_2(y)\,f_1(x,y)\,\mathrm{d}x\,\mathrm{d}y\right)^2\right).
\end{aligned}$$

First some definitions are given:

$$c_1(y) = \int_{\Omega_x}\frac{f_1^2(x,y)}{p_1(x,y)}\,\mathrm{d}x\,, \qquad (6.10)$$

$$c_2(y) = \frac{f_2^2(y)}{p_2(y)} \quad \left(= \int_{\Omega_x}\frac{f_2^2(x,y)}{p_2(y)}\,\mathrm{d}x\right), \qquad (6.11)$$

$$\begin{aligned}
\mu_1 &= \int_{\Omega_y} w_1(y)\int_{\Omega_x} f_1(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_{\Omega_y} w_1(y)\,f_2(y)\,\mathrm{d}y\,, \qquad (6.12)\\
\mu_2 &= \int_{\Omega_y} w_2(y)\int_{\Omega_x} f_1(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_{\Omega_y} w_2(y)\,f_2(y)\,\mathrm{d}y\,.
\end{aligned}$$

Now the variance (6.9) can be rewritten as

$$V\left[\langle I\rangle_c\right] = \int_{\Omega_y}\left(\frac{w_1(y)^2}{N_1}c_1(y) + \frac{w_2(y)^2}{N_2}c_2(y)\right)\mathrm{d}y - \left(\frac{\mu_1^2}{N_1} + \frac{\mu_2^2}{N_2}\right). \qquad (6.13)$$

6.2.4.2 Minimizing the variance

Minimizing the complete variance expression (6.13) is very difficult. The integrals μ_1 and μ_2 appear squared in the expression. This prevents a reformulation to an expression containing a single integral, which would allow a simpler minimization of the integrand only.

The first term of the variance is a single integral, and the integrand of this term will be minimized separately. This will deliver weighting heuristics. For the second term, containing the squared integrals, bounds will be derived that indicate how much better an optimal weighting function can be, compared to the derived heuristics.

Minimization of the first term Recall the term to be optimized,

$$\int_{\Omega_y} \left(\frac{w_1(y)^2}{N_1} c_1(y) + \frac{w_2(y)^2}{N_2} c_2(y) \right) \mathrm{d}y \,.$$

The integrand is always positive, so if it can be minimized for an arbitrary y, the integral will also be minimal. Further on, we will drop the y in the integrand to simplify the notation.

From the minimization the weighting heuristics will be derived. These heuristics must, of course, satisfy the constraints given in §6.2.3. The first constraint, $w_1 + w_2 = 1$, is introduced into the minimization using a Lagrange multiplier λ (The resulting heuristics will automatically satisfy the second constraint (6.8)).

Minimizing for w_1 and w_2, all partial derivatives (for w_1, w_2, λ) of the following equation must be zero:

$$m = \frac{w_1^2}{N_1} c_1 + \frac{w_2^2}{N_2} c_2 + \lambda(w_1 + w_2 - 1) \,.$$

The partial derivatives are given by

$$\begin{aligned} \frac{\partial m}{\partial w_1} &= 2\frac{w_1 c_1}{N_1} + 0 + \lambda \,, \\ \frac{\partial m}{\partial w_2} &= 0 + 2\frac{w_2 c_2}{N_2} + \lambda \,, \\ \frac{\partial m}{\partial \lambda} &= w_1 + w_2 - 1 \,. \end{aligned}$$

Solving this set of equations for w_1 and w_2 yields

$$\begin{aligned} w_1(y) &= \frac{N_1/c_1(y)}{(N_1/c_1(y)) + (N_2/c_2(y))} \,, \\ w_2(y) &= \frac{N_2/c_2(y)}{(N_1/c_1(y)) + (N_2/c_2(y))} \,. \end{aligned} \tag{6.14}$$

These weighting functions minimize the first term of the variance.

Bounding the second term The second part of the variance, $\frac{\mu_1^2}{N_1} + \frac{\mu_2^2}{N_2}$, is bounded by

$$\frac{1}{N_1 + N_2} I^2 \leq \frac{\mu_1^2}{N_1} + \frac{\mu_2^2}{N_2} \leq \frac{1}{min(N_1, N_2)} I^2 \,. \tag{6.15}$$

This can be proved as follows:

- The upper bound follows from:

$$\frac{\mu_1^2}{N_1} + \frac{\mu_2^2}{N_2} \leq \frac{1}{\min(N_1, N_2)} (\mu_1^2 + \mu_2^2) \leq \frac{1}{\min(N_1, N_2)} (\mu_1 + \mu_2)^2 = \frac{1}{\min(N_1, N_2)} I^2 \,.$$

- The proof for the lower bound is a bit more involved. The weighting heuristics determine the value of μ_1 and μ_2 (eq. 6.2.4.1). The term $\frac{\mu_1^2}{N_1} + \frac{\mu_2^2}{N_2}$ can be minimized over all possible weighting heuristics, given the constraint that $\mu_1 + \mu_2 = I$. This minimum determines the lower bound of the term. The minimization can again be performed with a Lagrange multiplier. All partial derivatives of

$$\frac{\mu_1^2}{N_1} + \frac{\mu_2^2}{N_2} - \lambda(\mu_1 + \mu_2 - I)$$

must be zero. The resulting set of equations delivers the minimum values:

$$\mu_1 = \frac{N_1}{N_1 + N_2} I \qquad \mu_2 = \frac{N_2}{N_1 + N_2} I \,.$$

Substituting these results into the variance term $\frac{\mu_1^2}{N_1} + \frac{\mu_2^2}{N_2}$ yields the lower bound.

Discussion The minimization and derivation of the bounds are very similar to multiple importance sampling. The weights derived from the minimization,

$$\begin{aligned} w_1(y) &= \frac{N_1/c_1(y)}{(N_1/c_1(y)) + (N_2/c_2(y))} \\ w_2(y) &= \frac{N_2/c_2(y)}{(N_1/c_1(y)) + (N_2/c_2(y))} \,, \end{aligned}$$

are generalizations of the balance heuristic (eq. 3.3). They reduce to the balance heuristic when Ω_x is the empty domain, as will be shown in §6.2.5.3.

This generalized balance heuristic minimizes the first term of the variance (6.13). Other weighting heuristics can only improve upon the balance heuristic within the specified bounds (6.15). This means that the generalized balance heuristic is always a good choice for computing the weights. Just as with multiple importance sampling, one could define generalizations of the power heuristic or other heuristics, which are expected to have similar properties. In practice, we have used the generalized balance heuristic in all our applications.

6.2.4.3 Relation to the method of expected values

The method of expected values is a special case of our weighted estimator, namely when p_2 is the marginal pdf of p_1:

$$p_2(y) = \int_{\Omega_x} p_1(x,y)\,\mathrm{d}x \,.$$

It is well known that, in this particular case, the single estimator using pdf p_2 is preferred (see §3.3.5).

It is interesting, from a theoretical viewpoint, to investigate what weights would be computed with the balance heuristic. Using the equality $p_1(x,y) = p_2(y)p(x|y)$, that relates marginal and conditional pdfs, the weight w_2 can be rewritten as:

$$w_2(y) = \frac{N_2}{(N_1 / \int_{\Omega_x} \frac{f_1^2(x,y)/f_2^2(y)}{p(x|y)}\,\mathrm{d}x) + N_2} \,. \tag{6.16}$$

The weight will be close to one if the integral in the denominator becomes large.

The variance of a single sample Monte Carlo estimator that estimates $f_2(y)$ for a given y by integrating over Ω_x is given by

$$V\left[\frac{f_1(\xi)}{p(\xi|y)}\right] = \int_{\Omega_x} \frac{f_1^2(x,y)}{p(x|y)}\,\mathrm{d}x - f_2^2(y)\,. \tag{6.17}$$

Dividing this equation by $f_2^2(y)$ gives

$$V\left[\frac{f_1(\xi)}{p(\xi|y)}\right] / f_2^2(y) = \int_{\Omega_x} \frac{f_1^2(x,y)/f_2^2(y)}{p(x|y)}\,\mathrm{d}x - 1\,. \tag{6.18}$$

When the variance increases, the integral on the right side also increases. This integral also appears in equation (6.16), and an increase causes $w_2(y)$ to be closer to one. In other words, if a difficult integral is precomputed over Ω_x, the weights for the samples using the precomputed result will also become larger. If a simple, low variance integral is precomputed, f_2 is estimated accurately by just a few x_i samples and there is no reason not to count the samples generated using $p_1(x,y)$.

This result shows that the balance heuristic gives a reasonable weight even in this limiting case. Of course, the number of samples used for each estimator also influences the weights. In practice, one would choose $N_1 = 0$, and allocate all samples to the second and better estimator.

6.2.5 Generalization for any number of sampling techniques

The results for two sampling techniques can be extended to any number of sampling techniques, each with a possibly different precomputed integral. The result is almost identical, but to concisely represent the general case we first need to introduce some notation.

6.2.5.1 Definitions and notation

Suppose that we have M techniques available to estimate a multi-dimensional integral $I = \int_\Omega f(x)\,\mathrm{d}x$. Each technique k, has an associated sub-domain Ω_k for which $\int_{\Omega_k} f(x)\,\mathrm{d}x_k$ is precomputed.

Just as in the case of two pdfs, the weights must not depend on any of the precomputed sub-domains Ω_k. We will define the *common sub-domain*, Ω_y, as the sub-domain that none of the techniques precompute[1]:

$$\Omega_y = \Omega \setminus (\bigcup_{k=1}^{M} \Omega_k)\,.$$

The *residue sub-domain*, $\Omega_{\neg k}$, indicates that part of the total domain that is not precomputed and that is not part of the common sub-domain:

$$\Omega_{\neg k} = \Omega \setminus (\Omega_k \cup \Omega_y)\,.$$

The integration variable x can also be decomposed in a precomputed, a common, and a residue part:

$$x = x_k\, x_{\neg k}\, y\,.$$

[1]The sub-domains used in the general formulation can cover different dimensions. The set operations on these domains are implicitly extended to handle such cases. For example, the union of two domains Ω_x and Ω_y that cover different dimensions in the complete domain Ω is given by the product-space of these domains: $\Omega_x \times \Omega_y$.

The remaining integral to be computed by a technique k is given by

$$I = \int_{\Omega_{\neg k}} \int_{\Omega_y} f_k(x_{\neg k}, y)\, \mathrm{d}y\, \mathrm{d}x_{\neg k}\ .$$

These integrals are estimated using a pdf p_k using N_k samples.

6.2.5.2 Weighting heuristic

The combined estimator uses all the different techniques to generate samples, and weights each sample with an appropriate weight w_k:

$$\langle I \rangle_c = \sum_{k=1}^{M} \frac{1}{N_k} \sum_{i=1}^{N_k} w_k(y_i) \frac{f_k(x_{\neg k,i}, y_i)}{p_k(x_{\neg k,i}, y_i)}\,. \tag{6.19}$$

The generalized balance heuristic always provides decent weights, and is given by

$$\begin{aligned} w_k(y) &= \frac{N_k / c_k(y)}{\sum_{k'=1}^{M} N_{k'} / c_{k'}(y)}\,, \\ \text{with} \quad c_k(y) &= \int_{\Omega_{\neg k}} \frac{f_k^2(x_{\neg k}, y)}{p_k(x_{\neg k}, y)}\, \mathrm{d}x_{\neg k}\ . \end{aligned} \tag{6.20}$$

The factors c_k remove any dependency of the weights on the residue domain. As a result the weights only depend on the common sub-domain. The derivation of this heuristic is analogous to the derivation of the heuristic for two pdfs.

6.2.5.3 Relation to multiple importance sampling

The balance heuristic as given in equation (6.20) generalizes the balance heuristic of standard multiple importance sampling and the one we derived for just two pdfs. It allows an arbitrary combination of sampling techniques with partial precomputation of the target integral.

Multiple importance sampling (MIS), for example, corresponds to the case were nothing is precomputed: $\Omega_k = \emptyset\ (= \Omega_{\neg k})$ and $y = x$. The $c_k(y)$ factors become

$$\text{MIS:} \quad c_k(y) = \frac{f_k^2(y)}{p_k(y)} = \frac{f^2(x)}{p_k(x)}\,.$$

This leads to the weights

$$\begin{aligned} \text{MIS:} \quad w_k(x) &= \frac{N_k / c_k(x)}{\sum_{k'=1}^{M} N_{k'} / c_{k'}(x)} \\ &= \frac{N_k p_k(x) / f^2(x)}{\sum_{k'=1}^{M} N_{k'} p_{k'}(x) / f^2(x)} \\ &= \frac{N_k p_k(x)}{\sum_{k'=1}^{M} N_{k'} p_{k'}(x)}\,, \end{aligned}$$

which are identical to the balance heuristic given in equation (3.3).

It is interesting to note that, while the weights in standard MIS only depend on the pdf, the integrand f_k is present in the c_k terms. This is logical: Some of the estimators use a partially precomputed integral. The weighting heuristics must take into account how difficult the precomputation is for those techniques that do

not use the precomputed information. The c_k terms include an indication of the variance of estimating the partial integral with the other pdfs.

6.3 Implications for weighted multi-pass methods

This section relates the general theory developed in the previous section to its application in Monte Carlo rendering and multi-pass methods in general.

The integrals to be computed in global illumination, are flux values for each pixel. Chapters 4 and 5 dealt with multi-pass methods, where part of the illumination is precomputed in one or more object-space passes. A final image-space pass uses the stored illumination to compute the pixel flux.

The combination of bidirectional path tracing (BPT), that used the self-emitted radiance, and path tracing, that used a precomputed radiosity solution, was an example where different path sampling techniques in the image-space pass could cover the same light transport. In the previous chapter, separation was used to prevent computing some of the light transport twice, but now weighting will be used instead.

In this section we will discuss the formulation of the combined estimator and the balance heuristic in a general multi-pass context. Section 6.4 will give a detailed example using the BPT-radiosity combination.

6.3.1 Problem statement

The pixel flux is the solution of an integral over all possible light transport paths (eq. 2.9):

$$I = \int_{\overline{\Omega}} \overline{f}(\overline{\mathbf{x}})\, \mathrm{d}\mu(\overline{\mathbf{x}}) \; .$$

Note that I may only represent a partial pixel flux if the light transport is separated in different parts. In any case, weighted estimators can only be applied to methods that compute exactly the same part of the light transport. Otherwise, contributions only accounted for by one of the methods would also be multiplied by the weights.

The domains and paths in the pixel flux integral relate to the domains and variables used in the weighting theory. The domain $\overline{\Omega}$, path-space, represents the full domain, which is split into several sub-domains. Each object-space pass (k) that precomputes illumination, covers a certain part of path-space, $\overline{\Omega}_k$.

If full paths, that connect a light source with the eye, are represented by $\overline{\mathbf{x}}$, then these paths can be split into several sub-paths

$$\overline{\mathbf{x}} = \overline{\mathbf{x}}_k \overline{\mathbf{x}}_{\neg k} \overline{\mathbf{y}} \, ,$$

where $\overline{\mathbf{x}}_k$ represents the precomputed part of the path, and $\overline{\mathbf{y}}$ is the common sub-path, not precomputed by any of the k preprocessing passes. The residue sub-path, $\overline{\mathbf{x}}_{\neg k}$, is not precomputed by method k but by at least one other preprocessing pass. This is shown schematically in figure 6.2 (page 81).

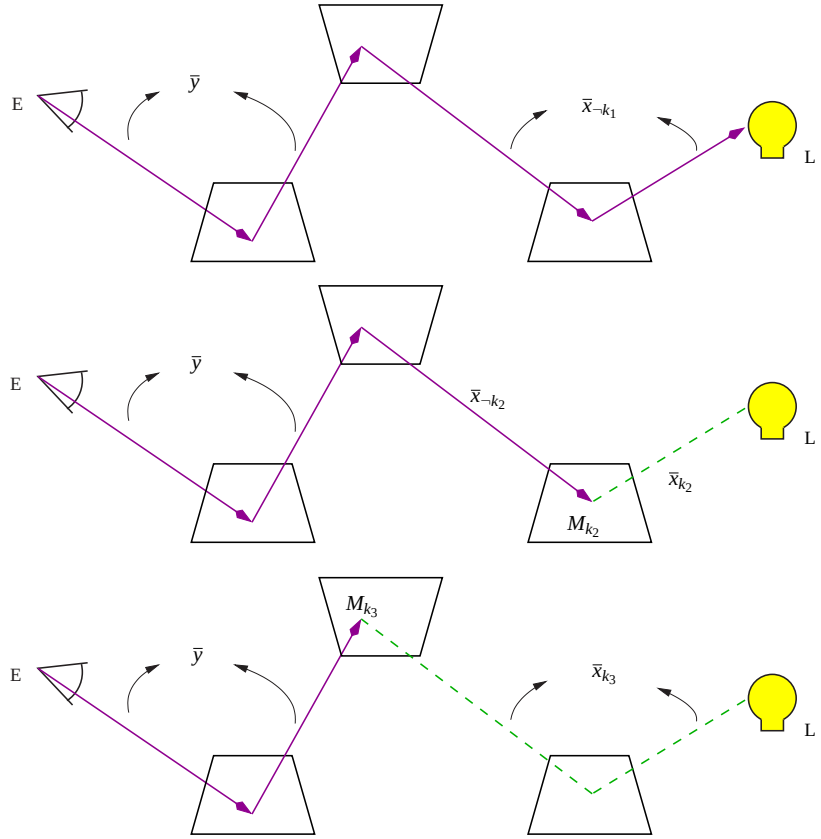


Figure 6.2: Splitting of a full path in a common sub-path $\bar{\mathbf{y}}$, a precomputed sub-path $\bar{\mathbf{x}}_k$ (for which the result is precomputed into a radiance solution M_k) and a residue sub-path $\bar{\mathbf{x}}_{\neg k}$ (precomputed by some method, but not by k itself). The split is shown for three different methods (k_1, k_2, k_3), where the first does not use any precomputation, and the other two do but over different sub-paths.

Consider, for example, a radiosity method, that precomputes LD^* paths. A general path can be split into $\bar{\mathbf{x}}_{rad}$, being the longest LD^* prefix, and $\bar{\mathbf{y}}$, the rest of the path. For bidirectional path tracing, that uses no precomputed illumination, a path is split in the self-emitted radiance, L, a residue path that contains the sequence of initial diffuse reflections, and a common sub-path $\bar{\mathbf{y}}$ that is the same as for radiosity.

6.3.2 Weighted estimators

The combined estimator (eq. 6.19) is evaluated in the final pass of a weighted multi-pass configuration. For each estimator k, N_k paths $\overline{\mathbf{x}_{\neg k}\mathbf{y}_i}$ are sampled and evaluated. The evaluation requires the computation of the unweighted contribution f_k/p_k and the weight w_k:

- The unweighted contribution, $f_k(\overline{\mathbf{x}_{\neg k}\mathbf{y}})/p_k(\overline{\mathbf{x}_{\neg k}\mathbf{y}})$, uses the stored information (or the self-emitted radiance) at the end of the path, and evaluates the contribution to the image.
- The evaluation of the weight first requires identification of the common sub-path $\bar{\mathbf{y}}$ by stripping the necessary vertices. Subsequently $w_k(\bar{\mathbf{y}})$ can be evaluated, for example, using the balance heuristic.

The allocation of samples over the different pdfs, i.e., the choice of N_k, is often dictated by how the paths are generated. For example, if path tracing reading a radiosity solution is combined with bidirectional path tracing as was done in chapter 5, then the eye sub-paths of the latter can be reused for path tracing. This fixes the number of samples for path tracing as soon as the number of samples for bidirectional path tracing is chosen.

6.3.3 Balance heuristic and evaluation

The evaluation of the balance heuristic requires the evaluation of the terms $c_k(\bar{\mathbf{y}})$ (equation 6.20) for each sampling technique. These terms will contain an integral over the residue domain:

$$c_k(\bar{\mathbf{y}}) = \int_{\Omega_{\neg k}} \frac{f_k^2(\bar{\mathbf{x}}_{\neg k}, \bar{\mathbf{y}})}{p_k(\bar{\mathbf{x}}_{\neg k}, \bar{\mathbf{y}})} \, \mathrm{d}\bar{\mathbf{x}}_{\neg k} \, .$$

It is highly inefficient to compute these integrals each time a weight is evaluated; the integrals are as complex as integrals computed in the preprocessing passes.

Two approaches can be taken to prevent this costly integration:

- Precompute $c_k(\bar{\mathbf{y}})$ during the preprocessing passes. This can be done by separating the integrand of $c_k(\bar{\mathbf{y}})$ into two parts: one part is only dependent on $\bar{\mathbf{y}}$ and can be evaluated on the fly, while the other part contains anything dependent on the integration variable $\bar{\mathbf{x}}_{\neg k}$. The latter part is integrated and stored as an object-space solution.

- Another approach is to approximate the integral in the c_k terms so that some or any dependency on $\bar{\mathbf{x}}_{\neg k}$ is removed.

The application in the next section will use both approaches to efficiently evaluate the weights.

6.4 Application: Radiosity and bidirectional path tracing

In this section the theory above will be applied to the combination of radiosity and bidirectional path tracing discussed in the previous chapter. First we will review the multi-pass configuration (MPC) and indicate where weighting can be useful (§6.4.1), followed by the actual computation of the weights (§6.4.2) in this configuration, and the results obtained (§6.4.3).

6.4.1 Weighted multi-pass configuration

Our MPC consists of two components:

1. The first component is standard bidirectional path tracing (BPT). This is an image-space pass, that actually encompasses several path sampling pdfs. No precomputed storage is used, only the self-emitted radiance of the light sources.

2. The second component is a precomputed radiosity solution that is read by a final path tracing pass (RADPT). The path tracing pass is performed simultaneously with the BPT pass, so that the eye-paths can be reused.

In the previous chapter, these components were organized so that all light transport was nicely separated between them. This required separating the direct diffuse light in order to assign indirect caustics to BPT. Here we will use a slightly different organization:

- $\mathsf{L_e}$ **part** [$\mathsf{L}\ ((\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E}+\mathsf{E}) \Rightarrow$ BPT]: Self-emitted radiance followed by a non-diffuse bounce is best handled by BPT. No precomputed illumination is used anyway, and BPT generates all the paths that path tracing would.

- LD **part** [$\mathsf{LD}\ ((\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E}+\mathsf{E}) \Rightarrow$ RADPT+ BPT]: The illumination due to direct diffuse illumination cannot be assigned completely to one of the components. In the previous chapter, we used further separation to solve this problem. In this chapter we will use weighting instead.

- LDI **part** [$\mathsf{LDD}^+\ ((\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E}+\mathsf{E}) \Rightarrow$ RADPT]: The indirect diffuse illumination is taken from the radiosity solution, and BPT is not used. The expected benefits from weighting this illumination are small, because the longer sub-paths (LDD^+), precomputed by the radiosity algorithm, are harder to compute by BPT.

All path evaluations are derived from the regular expressions as described in the previous chapter.

Figures 6.6 (b), (c) and (d) on page 90 show the three separated parts of the light transport for our example scene. For the LD part (c) the weighted solution, that is discussed next, is shown. Figure (a) shows the sum of the three components, and covers all light transport.

6.4.2 Weighting *LD* transport

Weighting the LD part requires identifying all the different sampling techniques and their pdfs, and computing weights for each pdf.

6.4.2.1 Different pdfs

Figure 6.3 shows all the different sampling techniques that contribute to the LD part for a path of length 4. Bidirectional path tracing consists of several sampling techniques $p_{s,t}$ that construct paths up to the light source. Only one sampling technique $p_{\text{RADPT}} = p_{s,0}$ (tracing eye paths) is used for the radiosity solution. These paths do not reach a light source, but end on a surface where the radiosity solution is read.

The contribution of a path is its normal evaluation, multiplied by the weight (that only depends on the common sub-path). Thus for BPT, the light vertex must be stripped before evaluating the weight. An

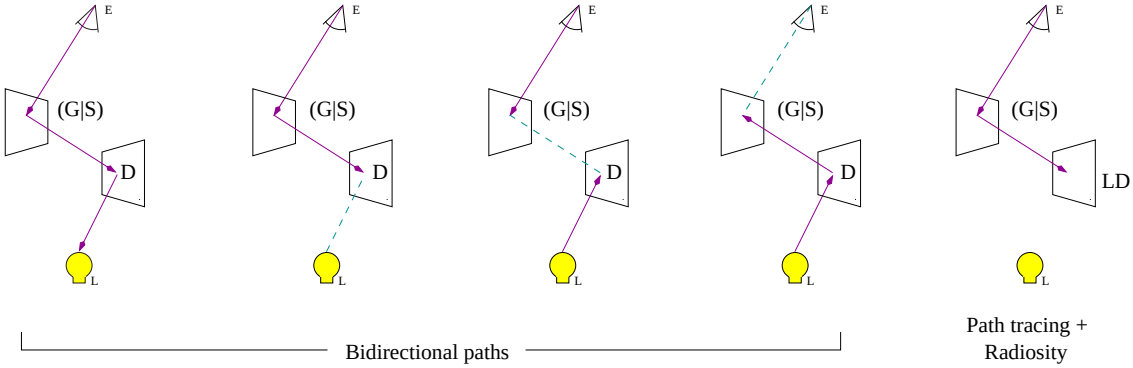


Figure 6.3: Several different path sampling techniques can sample overlapping transport ($\mathsf{LD(G|S)X^*E}$). The eye path on the right, that uses the radiosity solution, is shorter than the bidirectional paths on the left.

Algorithm 1 Weighted combination of radiosity and bidirectional path tracing

---

Compute pixel:

1. For $i = 1$ to N samples per pixel:
   (a) Sample eye path
   (b) Sample light path
   (c) For ($s <$ number of eye path vertices) and ($t <$ number of light path vertices)
      i. Form a path by connecting the vertices s and t and compute the following evaluations
      ii. $\mathsf{L_e}$: Evaluate ($\mathsf{L((G|S)X^*E+E)}$) with bidirectional path tracing (BPT)
      iii. LD : Evaluate ($\mathsf{LD((G|S)X^*E+E)}$) weighted with BPT
      iv. if ($t == 0$)
         LD : Evaluate ($\mathsf{LD((G|S)X^*E+E)}$) weighted using radiosity solution
         LDI : Evaluate unweighted contribution of ($\mathsf{LDD^+((G|S)X^*E+E)}$) using indirect radiosity solution

Evaluate ($\mathsf{LD((G|S)X^*E+E)}$) weighted with BPT (BiPath: $\mathbf{x\bar{y}}$, s ,t):

1. `uw` = unweighted contribution of $\mathbf{x\bar{y}}$
2. Strip light vertex **x** from path
3. `NC` = $N_{s,t}/c_{s,t}(\bar{\mathbf{y}})$ //weight term of this technique
4. `sumNC` = `NC`
5. For all other BPT techniques s', t' and the radiosity technique
   (a) `sumNC` $+=$ $N_{s',t'}/c_{s',t'}(\bar{\mathbf{y}})$ //weight term of other techniques
6. `weight` = `NC` / `sumNC`
7. `contribution` = `weight` * `uw`

Evaluate ($\mathsf{LD((G|S)X^*E+E)}$) weighted using radiosity solution (BiPath: $\bar{\mathbf{y}}$, s, 0):

- This is similar to the previous function, except no light vertex must be stripped.

---

overview of the complete algorithm, including other, unweighted contributions and the reuse of sub-paths, is given in algorithm 1.

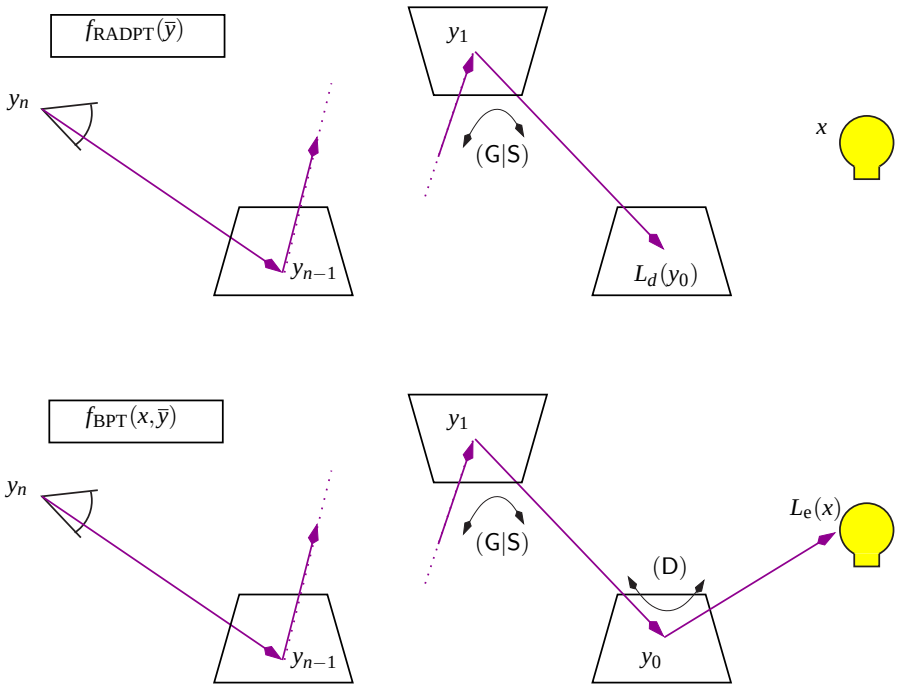


Figure 6.4: The path evaluation for the LD part of the light transport, for both radiosity-path tracing and bidirectional path tracing.

6.4.2.2 Computing the weights

Let $\mathbf{x}$ be the light vertex and $\bar{\mathbf{y}} = \mathbf{y}_0 \dots \mathbf{y}_n$ be the common sub-path, for which the last vertex $\mathbf{y}_n$ is the eye and $\mathbf{y}_0$ is the vertex where the radiosity solution is used. The path $\mathbf{x}\bar{\mathbf{y}}$ is a full bidirectional path.

The path evaluation for radiosity is given by (see equation 2.8):

$$f_{\text{RADPT}}(\bar{\mathbf{y}}) = L_d(\mathbf{y}_0) G(\mathbf{y}_0, \mathbf{y}_1) f_{(\mathsf{G}|\mathsf{S})}(\mathbf{y}_0 \to \mathbf{y}_1 \to \mathbf{y}_2) \dots W_e(\mathbf{y}_{n-1} \leftarrow \mathbf{y}_n),$$

with L_d the direct diffuse radiance taken from the radiosity solution ($B(\mathbf{y}_0)/\pi$). Because ($\mathsf{LD}((\mathsf{G}|\mathsf{S})\mathsf{X}^*\mathsf{E}+\mathsf{E})$) paths are considered, the evaluation of the first scattering in $\mathbf{y}_1$ uses non-diffuse components only (for sampling, any pdf can be used).

The path evaluation for a full (bidirectional) path is given by:

$$f_{\text{BPT}}(\mathbf{x}, \bar{\mathbf{y}}) = L_e(\mathbf{x}) G(\mathbf{x}, \mathbf{y}_0) \frac{\rho_d(\mathbf{y}_0)}{\pi} G(\mathbf{y}_0, \mathbf{y}_1) f_{(\mathsf{G}|\mathsf{S})}(\mathbf{y}_0 \to \mathbf{y}_1 \to \mathbf{y}_2) \dots W_e(\mathbf{y}_{n-1} \leftarrow \mathbf{y}_n).$$

Only diffuse components are used for the self-emitted radiance L_e and for the first scatter in $\mathbf{y}_0$, and a glossy or specular scatter in $\mathbf{y}_1$ in order to get the correct overlapping transport. The path evaluation is illustrated in figure 6.4. Note that we use the regular expression based path evaluation; we only have to supply the regular expressions and the evaluations above are derived automatically.

Each sampling technique has its own weighting function for which the general expression is given by equation (6.20). In practice the evaluation of a weighting function requires the evaluation of a term $N_k/c_k(\bar{\mathbf{y}})$ for each sampling technique. The different terms that appear in our combination are discussed below. There

is only one term for the radiosity-path tracing combination (RADPT: only 1 path sampling technique, eye paths), but there are several for bidirectional path tracing.

RADPT term With N the number of bidirectional paths sampled per pixel, there will also be N eye paths that use the radiosity solution:

$$N_{\mathrm{RADPT}} = N\,.$$

For a specific eye path $\bar{\mathbf{y}}$, $c_{\mathrm{RADPT}}(\bar{\mathbf{y}})$ is given by:

$$c_{\mathrm{RADPT}}(\bar{\mathbf{y}}) = \frac{f^2_{\mathrm{RADPT}}(\bar{\mathbf{y}})}{p_{s,0}(\bar{\mathbf{y}})}\,.$$

This is easily computed; it is the square of the path evaluation divided by the pdf. Both factors were already computed for the non-weighted contribution.

BPT terms The computation of the BPT terms requires more care. Several pdfs $p_{s,t}$ are used to generate the bidirectional paths, where s is the number of vertices in the eye sub-path and t the number of vertices in the light sub-path. All techniques use N samples to estimate a pixel, except $p_{1,t}$ (light tracing paths that are directly connected to the eye), that uses all paths (N times the number of pixels) for each pixel.

The general form of $c_{s,t}(\bar{\mathbf{y}})$ is given by

$$c_{s,t}(\bar{\mathbf{y}}) = \int_{\Omega_x} \frac{f^2_{\mathrm{BPT}}(\mathbf{x}\bar{\mathbf{y}})}{p_{s,t}(\mathbf{x}\bar{\mathbf{y}})}\,\mathrm{d}\mathbf{x}\,.$$

The integral removes the dependency on the light vertex $\mathbf{x}$; Ω_x covers all the light sources ($\Omega_x = A_l$). The dependence of $\bar{\mathbf{y}}$ on the light vertex depends on how the bidirectional path was constructed, and more specifically on the number of vertices t in the light sub-path. This dependency is shown schematically for several values of t in figure 6.5.

Note that in figure 6.5 (a), vertex $\mathbf{y}_1$ may depend on $\mathbf{x}$, because the direction from $\mathbf{y}_0$ towards $\mathbf{y}_1$ is usually sampled according to the BSDF. For non-diffuse BSDFs this outgoing direction depends on the incoming direction and hence on $\mathbf{x}$. Note also that, given a fixed $\mathbf{y}_0$ and $\mathbf{y}_1$, the subsequent vertex $\mathbf{y}_2$ will not depend on $\mathbf{x}$, because the direction towards $\mathbf{y}_2$, determined by BSDF sampling in $\mathbf{y}_1$, will only depend on the incident direction (from $\mathbf{y}_0$ to $\mathbf{y}_1$). Thus, figure (a) covers all cases where $t > 2$, and only in these cases $\mathbf{y}_1$ may depend on $\mathbf{x}$ (i.e., not when $t \leq 2$).

This dependency complicates the computation of the integral over the area of the light source, because three vertices have to be taken into account ($\mathbf{x}$, $\mathbf{y}_0$ and $\mathbf{y}_1$). Therefore, in this case, we *approximate* the pdf for sampling the direction in $\mathbf{y}_0$ towards $\mathbf{y}_1$ ($p(\mathbf{y}_1|\mathbf{y}_0,\mathbf{x})$) by assuming the direction sampling in $\mathbf{y}_0$ only used the diffuse component. For a diffuse scattering the sampled direction does not depend on the incident direction (all outgoing directions are equally important).

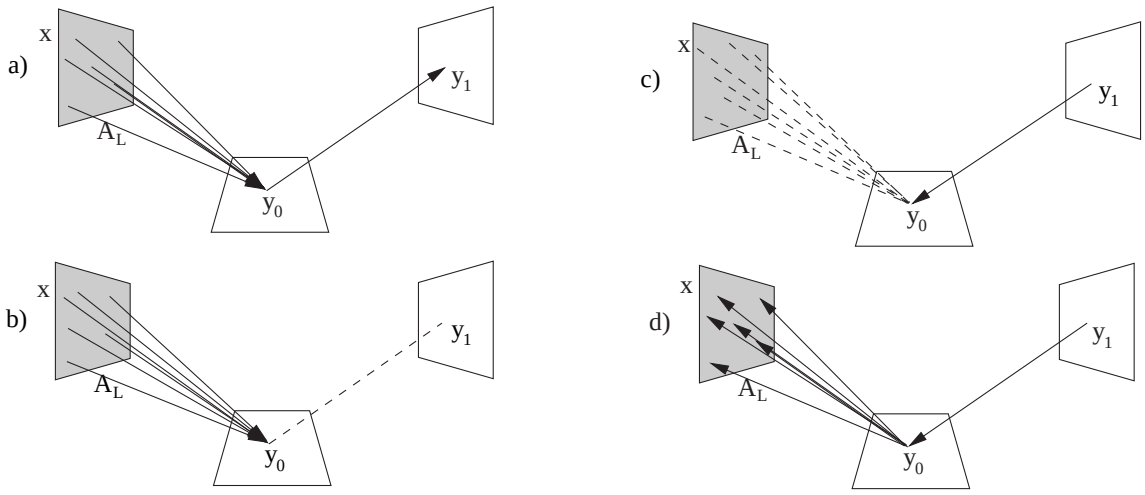


Figure 6.5: The weight of a bidirectional path must not depend on the light vertex $\mathbf{x}$. This dependency is removed by an integral over the light sources (indicated by the multiple connections to A_L). The specific integral varies with the length of the light sub-path t: (a) $t > 2$, (b) $t = 2$, (c) $t = 1$, (d) $t = 0$. The dotted line indicates the connection of the light and eye sub-paths.

Although the actual sampling may include other components, assuming the sampling of a diffuse scattering is a reasonable approximation, because the *evaluation* of the BSDF in $\mathbf{y}_0$ also considers the diffuse component only (because of the covered transport, see figure 6.4).

As a result of this approximation, all cases (i.e., $t \geq 0$) now only have $\mathbf{y}_0$ dependent on $\mathbf{x}$. Therefore, the integrand of $c_{s,t}(\overline{\mathbf{y}})$ can be split into a component only dependent on $\overline{\mathbf{y}}$, that can be moved out of the integral, and a second component $c'_{s,t}(\mathbf{y}_0)$ that is actually integrated over the area of the light sources.

For $c'_{s,t}(\mathbf{y}_0)$ only three different cases need to be considered depending on the light sub-path length t:

- $t > 1$ (Figure 6.5 (a) and (b)): For this case c' is given by

$$c'_{s,t>1}(\mathbf{y}_0) = \int_{A_l} \frac{(L_e(\mathbf{x})G(\mathbf{x},\mathbf{y}_0)\frac{\rho_d(\mathbf{y}_0)}{\pi})^2}{p(\mathbf{x})p(\mathbf{x}\rightarrow\mathbf{y}_0)}\,\mathrm{d}\mathbf{x}\,.$$

  When an analog simulation is used for generating the light paths (§3.4.2.2), the pdf $p(\mathbf{x})p(\mathbf{x}\rightarrow\mathbf{y}_0)$ is proportional to $L_e(\mathbf{x})G(\mathbf{x},\mathbf{y}_0)$. The proportionality factor is equal to the total self emitted flux Φ_e. By canceling out the factors, the integral simplifies to

$$\begin{aligned} c'_{s,t>1}(\mathbf{y}_0) &= \Phi_e\tfrac{\rho_d(\mathbf{y}_0)}{\pi} \times \textstyle\int_{A_l} L_e(\mathbf{x})G(\mathbf{x},\mathbf{y}_0)\tfrac{\rho_d(\mathbf{y}_0)}{\pi}\,\mathrm{d}\mathbf{x} \\ &= \Phi_e\tfrac{\rho_d(\mathbf{y}_0)}{\pi}L_d(\mathbf{y}_0)\,. \end{aligned}$$

  This factor can be evaluated easily because $L_d(\mathbf{y}_0)$ is given by the direct diffuse part of the radiosity solution.

- $t = 1$ (Figure 6.5 (c)):

$$c'_{s,1}(\mathbf{y}_0) = \int_{A_l} \frac{(L_e(\mathbf{x})G(\mathbf{x},\mathbf{y}_0)\frac{\rho_d(\mathbf{y}_0)}{\pi})^2}{p(\mathbf{x})}\,\mathrm{d}\mathbf{x}\,.$$

  Only the pdf for generating $\mathbf{x}$ is kept in the integral. Vertex $\mathbf{y}_0$ is not dependent on $\mathbf{x}$ but on $\mathbf{y}_1$, so that $p(\mathbf{y}_0)$ can be moved outside of the integral. When $\mathbf{x}$ is sampled proportionally to the emitted power

of the light sources, $p(\mathbf{x})$ is proportional to L_e. The proportionality factor is again Φ_e:

$$c'_{s,1}(\mathbf{y}_0) = \Phi_e \int_{A_l} L_e(\mathbf{x}) G^2(\mathbf{x},\mathbf{y}_0) (\frac{\rho_d(\mathbf{y}_0)}{\pi})^2 \, d\mathbf{x} \, .$$

This integral can be computed simultaneously with the radiosity computations, using a slightly different radiosity kernel (G^2/π^2 instead of G/π) and a different diffuse reflection coefficient (ρ^2 instead of ρ). Virtually no extra work is required: The visibility tests needed for computing standard form factors are reused to compute the 'special' form factors for these integrals. Only some extra storage is needed, because the partial weighting function $c'_{s,1}(\mathbf{y}_0)$ is stored on the finite element mesh, together with the radiosity solution.

- $t = 0$ (Figure 6.5 (d)): An empty light sub-path means that the eye sub-path hits a light source directly. This sampling technique is mainly useful for light sources visible through a specular reflection. Since a diffuse reflection is considered in $\mathbf{y}_0$, the expected benefits of this sampling technique are so low that we do not use it for the weighted overlapping transport (But it is used for the $\mathsf{L_e}$ part, see §6.4.1, page 82).

The factors $c'_{s,t}(\mathbf{y}_0)$ define a partial weighting function for any position $\mathbf{y}_0$ in the scene. The part of the weighting terms $c_{s,t}$ independent of $\mathbf{x}$ (outside the integral) can be computed similarly to the normal BPT weights and just need to be multiplied by $c'_{s,t}(\mathbf{y}_0)$.

Once all the weighting terms $N_{s,t}/c_{s,t}$ are computed, the total weight of a path is evaluated using equation (6.20).

Discussion The previous paragraphs showed the derivation and the computation of weights for one particular overlapping multi-pass configuration. Before showing the results of the weighting, two remarks can be made:

- The weighting functions are rather intricate to derive theoretically. One must take care of dependencies between the vertices and study the exact form of the weighting terms to find a tractable evaluation of the integral.

- Once the weights are derived, the practical evaluation of these weights in the implementation turns out to be very efficient. Nearly no computational overhead and a limited storage overhead are introduced. The evaluation of the weights is similar to evaluating the standard BPT weights.

6.4.3 Results

Recall that the total illumination in the image was split into three parts: $\mathsf{L_e}$, LD and LDI (see §6.4.1). Figure 6.6, page 90, shows the total weighted solution (a) and the three different parts (b, c and d). For

comparison, an unweighted image, with LD computed using radiosity and path tracing only, is shown in (e). Image (a) is superior due to the weighting of the LD part. This is most obvious for the indirect caustic, also shown in the magnifications.

The LD part was computed both with bidirectional path tracing and radiosity using the appropriate weights. Figure 6.7, page 91, shows the results of this weighting, using 4 samples per pixel for each image:

- The unweighted contributions are shown in (a) for bidirectional path tracing and in (b) for radiosity teamed up with path tracing. In (b) the direct diffuse illumination is noiseless, but the indirect caustic on the right is quite noisy. For BPT the dimmer direct illumination suffers from more noise, but, on the other hand, the indirect caustic is handled very well. The weights should retain the good qualities of both approaches. Note that these images are computed simultaneously, since eye paths are reused for the radiosity readout.

- In (c) and (d) the *weighted* contributions are shown for BPT and radiosity respectively. These images show that the weighting heuristics divide the light transport quite well between the two methods:

  – The indirect caustic, which is best handled by light paths, is entirely computed by bidirectional path tracing.

  – The direct diffuse illumination is taken from the radiosity solution, as desired. For the direct light on the white panel, however, BPT is assigned a larger weight. The factor $N_{1,1}/c_{1,1}$ is large because there is a high probability that light paths hit the panel —the light on the panel is an easy integral over A_l— and because there are N times the number of pixels light paths.

  – The glossy reflection of the panel in the left wall, is evenly distributed between the two methods. This is logical, as both techniques handle the effect equally well (or badly).

  These weighted images use exactly the same paths as the unweighted images, only the weighting factor was applied.

- Image (e) shows the sum of the weighted contributions. This result is superior to both (a) and (b), as the weighting preserves the strengths of both methods.

The final result (figure 6.6 (a)), is very similar to the separated result from the previous chapter (figure 5.7, page 69), which indicates the following:

- The separation in the previous chapter was chosen well and delivered quite good results for this example.

- The weighted combination, however, did not need the extra separation. The weighting heuristics automatically preserve the strengths of the different methods, and adapt to other scenes for which the separation might not perform as well.

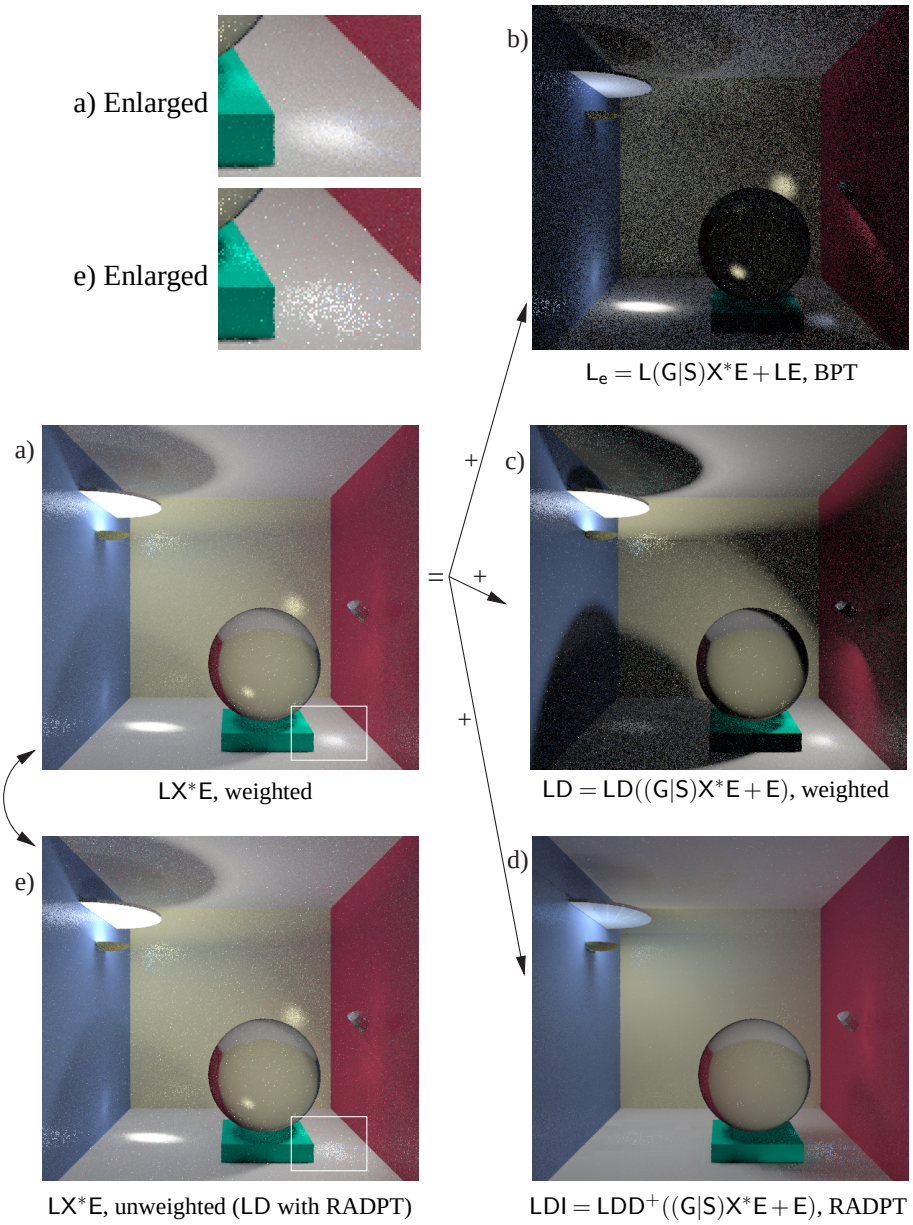


Figure 6.6: Weighted multi-pass results. (a) shows the final result, that consists out of three parts: (b) is computed with bidirectional path tracing only, (c) is a weighted combination of BPT and the radiosity solution (with path tracing), (d) only uses the radiosity solution (with path tracing). Figure (e) shows an unweighted combination, where the direct diffuse based illumination ($LD((G|S)X^*E + E)$) is computed using the radiosity solution only. Clearly (a) shows a much better indirect caustic.

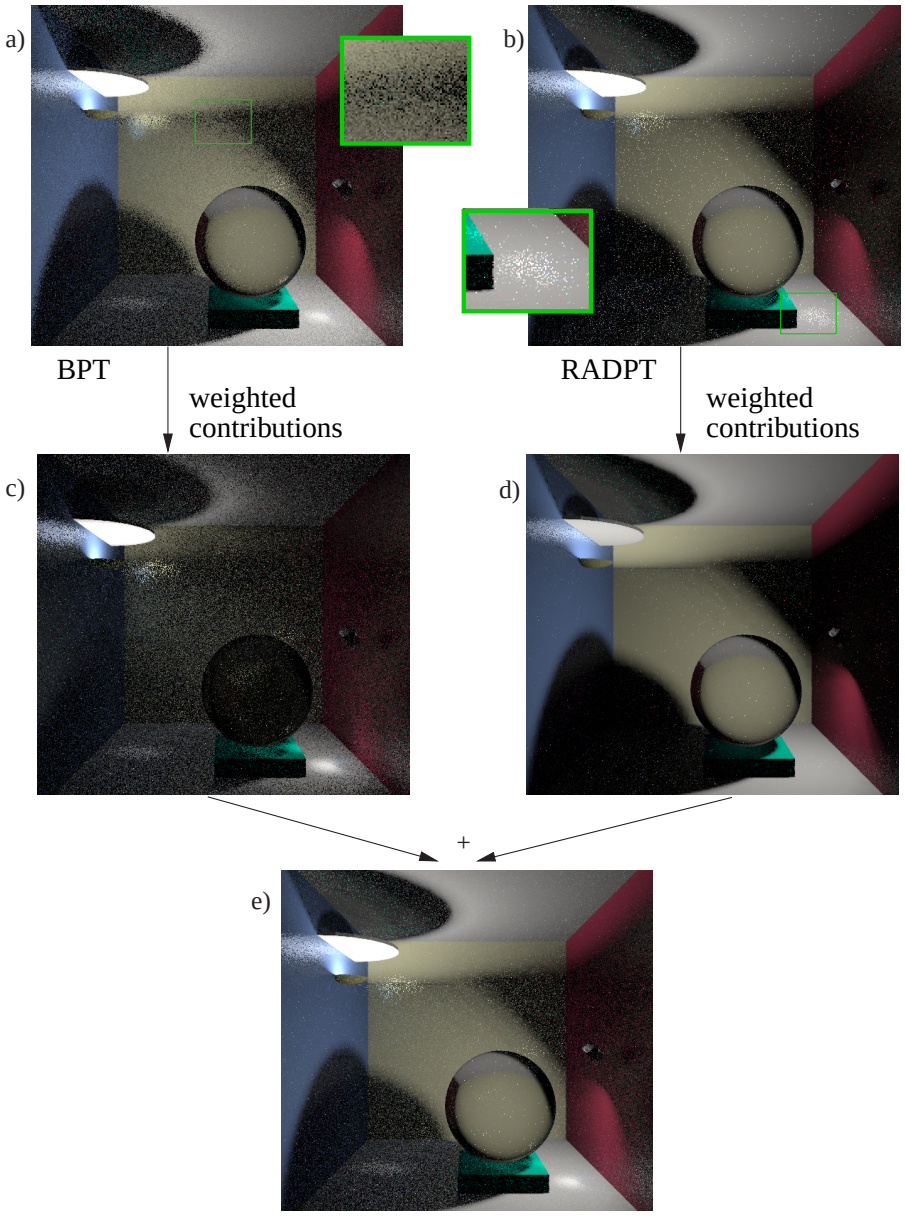


Figure 6.7: Weighting of the overlapping transport $(LD((G|S)X^*E+E))$. The weighting heuristics preserve the strengths of the respective methods: the indirect caustic is completely assigned to bidirectional path tracing, while the noiseless, direct diffuse illumination is taken from the radiosity solution (except for the diffuse white panel, which is hit by many light paths in bidirectional path tracing). For features that are difficult for both methods, such as the glossy reflection of the white panel seen in the blue wall on the left, the weight is evenly distributed so that both methods contribute to the weighted image.

Still some noise remains in both the images. In the blue wall, the glossy reflection of the caustic is particularly noisy. This is because none of the sampling techniques used in this multi-pass configuration is able to handle this effect well. In [P5] we presented a filtering technique that targets such remaining noise, by using density estimation techniques on the image plane.

6.5 Conclusion

6.5.1 Summary

In this chapter we presented a theory for weighted multi-pass methods. This theory generalizes multiple importance sampling for estimating integrals of different dimensions. A necessary and sufficient constraint on the weighting functions was developed and a provably good heuristic for the weights was derived. This theory can be applied as a variance reduction technique to multi-pass methods that employ a Monte Carlo image-space pass. Weighted multi-pass methods do not require extreme separation, but assign (partial) light transport automatically to the different algorithms.

The method was applied to a combination of bidirectional path tracing and radiosity, and showed that the weighting heuristics adequately weight overlapping transport. For the same computation time, better results were obtained with the weighted combination.

The automatic weighting is the main strength of weighted multi-pass methods: the weights can adapt to different scenes and illumination conditions, resulting in a more robust global illumination method. While the derivation of the weights for a specific multi-pass configuration is intricate and requires great care, the resulting weighting heuristics can be computed efficiently.

6.5.2 Directions for future research

The weighting technique has only been applied to one multi-pass configuration. Although developing the weights is not a trivial task, many other multi-pass configurations can benefit from the weighting technique and should be re-investigated in this context.

Antal et al. [2], for example, recently constructed a weighted multi-pass method that combines path tracing with ray-bundle tracing [108]. They use an approximate weighting heuristic that loosens the constraint, but is still unbiased.

Another promising direction for future research is incorporating *error control* in the weighting functions. Radiosity methods for instance can sometimes give an estimate for the accuracy of the radiosity solution [10], which can be integrated easily into the weights. A low accuracy (e.g., near sharp shadow boundaries) will lower the weight of the radiosity solution, favoring other methods for some specific part of the illumination. Such an approach was presented in [88] by Scheel and Stamminger for final gathering. This approach could also be incorporated into a weighted multi-pass method.

As can be seen in the images, the important contributions of bidirectional path tracing are often very localized. Adaptive importance sampling [29] or even Metropolis sampling, especially for the light subpaths, could help to spend more work on the important parts of the light transport. However, the combination of varying pdfs with the weighting heuristics remains an open problem.

7 Path differentials

This chapter describes a new tool for global illumination algorithms: path differentials. A path is augmented with derivatives of its vertices, directions and its evaluation. These derivatives give an estimate of the sensitivity of the path with respect to its generating variables and allow the estimation of the footprint, the region of influence of a path. This footprint can be useful in many global illumination algorithms.

7.1 Introduction

A path, whether traced from the eye or a light source, is a point sample in a multi-dimensional space. Different *sampling events* such as sampling a point on an area light source or choosing a reflected direction, introduce new variables, new degrees of freedom, in a path. Each path can be generated by a particular instantiation of these *path variables*.

But a path is only a point sample. The evaluation of a path only takes into account the information available at the exact location of its vertices. Consider, for example, several eye rays that are traced through neighboring pixels and hit a textured surface. The texture is only evaluated at the location of the vertex, resulting in aliasing artefacts when the texture frequency is higher than the density of the paths. Such undersampling problems occur in many situations in image synthesis. In Monte Carlo image synthesis, stochastic sampling trades aliasing artefacts for noise, but the undersampling problems remain.

Supersampling —just throwing in more samples— is a simple and effective solution to these problems, but it is also expensive. For instance, in the texture mapping example, it might be better to filter the texture locally around the vertex instead of tracing more eye rays, but the question is what region of the texture should be filtered. If more information were known about the region of influence of a path, about the distance to neighboring rays, then such a local filtering would be possible. We will call such a region of influence of a vertex in a path informally *the footprint* of a path in a certain vertex.

Several approaches have been tried to compute such a footprint (an overview is given in §7.2). All these techniques, however, only consider classical ray tracing, where a path is only determined by two path variables, namely the image plane coordinates. Perfectly specular scattering is deterministic and does not introduce extra variables in the path.

Path differentials provide a heuristic for computing the footprint of a path, and they are not restricted to perfectly specular scattering. They accommodate arbitrary BSDFs, area light sources, Phong interpolated normals, and curved surfaces. Any global illumination algorithm based on path sampling may benefit from the information provided by path differentials.

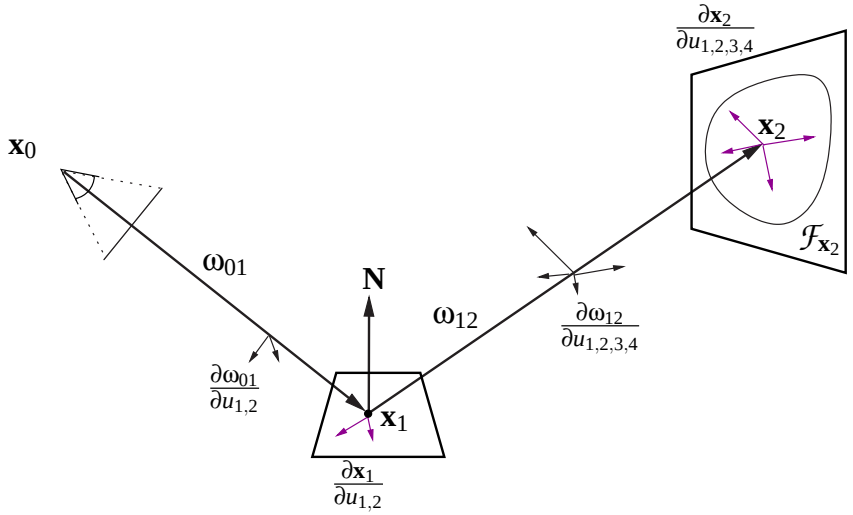


Figure 7.1: An example of footprint estimation with path differentials for a path of length three. For each vertex and direction, partial derivatives are computed with respect to all generating variables u_k. Multiplying the derivatives with a perturbation interval Δu_k gives differential vectors that are used to estimate the footprint $\mathcal{F}_{\mathbf{x}_2}$.

The idea behind path differentials is based on partial derivatives and a first order Taylor approximation of a path. With path differentials the footprint of a path is estimated as follows:

- A path is traced by sampling new directions and vertices. A new vertex or direction depends on the previously sampled vertices and directions in the path, but may also depend on newly introduced (random) variables. Partial derivatives of vertices and directions with respect to every variable in the path are computed. These derivatives indicate the sensitivity of the vertex in terms of each path variable.

- A small perturbation of the path variables moves a vertex over a small distance. This displacement can be approximated by a differential: the multiplication of the derivatives with the perturbation, forming a first order Taylor approximation of the vertex. By choosing appropriate *perturbation intervals*, the set of displaced vertices forms an area that we define as the footprint of the path in the vertex.

- This footprint can now be used for texture filtering or other applications.

Figure 7.1 illustrates the estimation of a footprint by path differentials for a path of length three.

The choice of perturbation intervals is very important for a good footprint heuristic. Large intervals result in large footprints, that do reduce noise or aliasing, but that also introduce bias or blurring in the solution. A very small footprint, however, will not sufficiently reduce the noise. We will explore several heuristics based on the number of paths traced, the rate of change of the path evaluation over the footprint, and the second order partial derivatives. These heuristics combined ensure an adequate coherence over the footprint and a good noise versus bias trade-off.

Path differentials can be used in many global illumination algorithms. We have used the footprint for filtering textures locally on surfaces. We also used it for a hierarchical refinement oracle in particle tracing

radiosity and applied it to importance calculations in photon maps (see chapter 8). Many other applications are still possible, and we will discuss a few at the end of this chapter.

The rest of this chapter is organized as follows: Related work that tries to extend paths with some kind of footprint information is discussed in §7.2. A general definition of a footprint is given in §7.3. How to approximate the footprint with partial derivatives is discussed in §7.4. The footprint approximation requires the computation of partial derivatives (§7.5) and perturbation interval heuristics to determine a small neighborhood around a point sample in the path domain (§7.6). Two applications are presented, texture filtering (§7.7) and hierarchical particle tracing radiosity (§7.8). A conclusion, along with some other possible applications and extensions, are given in §7.9. An appendix at the end of the chapter (§7.A) presents some additional technical details about computing path derivatives.

We have presented path differentials previously in [P8] and [P7]. This chapter presents a more in-depth discussion of path differentials, with a more rigorous definition of the path footprint. Also, the work on second order derivatives is new.

7.2 Related work

This section gives an overview of existing techniques that try to exploit coherence in the neighborhood of a path by tracking some kind of footprint. Three main categories can be identified: extension of a path to a finite size (§7.2.1), tracking connectivity between neighboring paths (§7.2.2) and differential techniques (§7.2.3). Our method belongs to the the last category.

7.2.1 Extension of a path to a finite size

A first approach, most popular in the early days of ray tracing, extends infinitely thin rays to a finite size. The differences between these techniques lie in the representation of the extended ray, the intersection computations, and the way reflected and refracted rays are traced.

Beam tracing In ’84, Heckbert proposed *beam tracing* [42]. A ray is extended to a beam with a polygonal footprint. The scene is rendered starting with a single pyramidal beam using the eye as the top and the screen as the rectangular base. Intersection is performed by clipping the beam against all the polygons in the scene. The beam is fragmented by the visible portions of the intersected polygons. A perfectly specular scattered beam is formed by the polygonal footprint and a new, mirrored center for the beam.

Lighting calculations can be performed coherently over the footprint of the beam (Heckbert used a rasterization algorithm over the footprint), which, for example, resolves aliasing due to undersampled textures.

Beam tracing is limited to polygonal scenes only. Only perfect specular reflection and refraction are supported, and even then the refracted beam is approximate due to the non-linearity of refraction. Inter-

section calculations require a robust, general clipping procedure and are difficult and time consuming for complex scenes.

Cone tracing A similar technique, *cone tracing*, was presented by Amanitides et al. [1], and extends a ray to a cone. The intersections of the cone with objects in the scene determine, or better approximate, the fraction of the cone that is intersected by the objects. This fraction is used for anti-aliasing over the footprint. The cone is fragmented by the intersected surfaces, and reflected cones are traced. By widening the cone angle, glossy reflections can be simulated. Soft shadows were also simulated by tracing a cone from the (point) light sources.

Pencil tracing *Pencil tracing*, proposed by Shinya et al. [90], extends a ray to a set of paraxial rays. Propagation and scattering are approximated by a linear system matrix, that provides a theoretical basis for these operations. However, scattering and even propagation of a ray using the system matrix are only approximate. To circumvent difficult intersection tests, the pencil is only used when its intersection is contained completely in a single surface. When a pencil crosses an object boundary and hits several objects, the method resorts to plain ray tracing. Clearly, this is prohibitive for complex scenes, containing many small primitives.

Pyramidal rays An approach similar to beam tracing uses *pyramidal rays* [31]. These pyramidal rays are beams with a rectangular footprint. Intersection splits the ray hierarchically to determine visibility up to a predefined threshold, similar to the Warnock hidden surface removal algorithm [127]. Intersections are thus simpler than with beam tracing, that uses a Weiler-Atherton clipping algorithm [128] to compute the visible parts of the polygons exactly. Therefore, pyramidal rays are able to handle complex scenes better than beam tracing, but require a finer fragmentation of the rays in order to resolve the visibility accurately.

Reflection is done similarly to beam tracing, with inclusion of blurred reflections by widening the pyramid and fragmentation of the pyramid on curved reflectors.

Wavefront tracing Elber approximates light wavefronts emitted by a point or spherical light source by a parametric surface that moves forward with time [30]. The wavefront is reflected by a (single) free-form surface. Irradiance in a point is given by the Gaussian curvature of the wavefront at that point. This method was only demonstrated for direct lighting from a single light source on a simple surface. For complex scenes the reflection (and fragmentation) of the wavefront becomes difficult.

Conclusion All these extended ray methods suffer from two important disadvantages:

- Intersection calculations are intricate and time consuming: the intersection of a ray extended to a

geometric primitive (beams, cones, ...) with an object in the scene is much harder to compute than an intersection of an infinitely thin ray. Therefore, these methods only handle relatively simple scenes. For such simple scenes, however, these methods work well, since visibility coherence is exploited well by the extended rays. For infinitely thin rays, on the other hand, highly optimized and simple algorithms exists for intersecting them with a great number of different types of objects, so that many rays can be traced for the same cost as a single extended ray.

- The methods only handle perfectly specular reflection and refraction. Even for these simple BSDF models, some methods require approximations. Glossy reflection is supported by cone tracing and pyramidal rays using a trick that widens the ray footprint.

  Extension to other, more general BSDFs is difficult. Reflection and refraction cannot be represented by a single extended ray, so several scattered beams, cones or pencils should be traced. One approach to this problem would be to decouple lighting and visibility calculations. For example, reflection could be computed by tracing a beam that encompasses the complete hemisphere and by fragmenting this ray based on object visibility. Since visibility is completely resolved with respect to the ray's origin, lighting computations can be performed coherently over the visible portions of the object surfaces. Such an approach is taken to the limit in global visibility methods, such as the visibility skeleton [26], where *all* visibility relations between objects in the scene are resolved analytically (without sampling). Again, these methods are limited to relatively simple scenes.

7.2.2 Tracking path connectivity

Another approach to compute information about the neighborhood of a path is to explicitly maintain connectivity information between the neighboring paths.

Collins uses such an approach to construct caustic texture maps [23]. With standard ray tracing he constructs light paths that are splatted into a caustic texture map. A (Gaussian) filter kernel is used to distribute the power of a light path vertex over several texels in the caustic texture. For all starting rays, emitted from a point light source, a pointer is kept to the four neighboring starting rays. This connectivity is tracked during propagation and scattering of the light path. The splat size is determined by the distance to the vertices of neighboring paths: When neighboring paths are far away, the density of paths near the vertex is low, and a larger splat size is used. The footprint of a ray is thus estimated by the distance to the neighboring paths.

This approach requires only a simple extension of classical ray tracing, namely tracking the neighbors, and leaves intersection tests unchanged. It works well when neighboring rays follow the same path and have the same ray tree (i.e., when the same objects are hit by corresponding vertices in the neighboring paths). The connectivity is lost, though, when the neighboring rays hit different objects, resulting in different ray

trees. Using arbitrary BSDFs and Monte Carlo path sampling is possible in theory, but the stochastic sampling often results in different scattered directions for neighboring rays. This leads to highly diverging ray trees, limiting the usefulness of the connectivity information.

7.2.3 Differential techniques

Differential techniques differ from the previous approaches in that they keep considering an infinitely thin path. Any information about a path's neighborhood or its footprint is derived from the path itself, avoiding difficult intersection tests or divergent ray trees.

Differential geometry A first approach tracks local properties of the light wavefront along a path using differential geometry techniques. Mitchell and Hanrahan use differential wavefront tracing to compute caustics from curved reflectors [74]. Caustic paths (or Fermat paths) are found by minimizing the optical distance traveled by a path from a point light source over a specular reflector to a target illumination point. Once such a path is found, a differential wavefront is computed along the path. The Gaussian curvature of the wavefront gives the caustic intensity at the target point.

Wavefront tracing was also used by Loos, Slusallek and Seidel for the design of progressive lenses [71]. Reflection and refraction were considered to be perfectly specular. Although differential geometry is well studied for specular reflection and refraction [106], it is unclear how to extend this work to sampled BRDFs and full global illumination.

Path derivatives A different approach, not relying on differential geometry and wavefront properties, uses partial derivatives of paths.

Chen and Arvo propose a method to efficiently compute perturbations of specular paths [17, 16, 15]. Specular paths are paths that only account for perfectly specular scattering in the intermediate vertices. The perturbations are computed using a second order Taylor approximation of a path. Starting from a sparse set of sampled paths, the remaining paths in between are computed by perturbation. They present two applications: an efficient computation of specular reflections for implicit surfaces and the direct computation of iso-lux contours for caustics. While good results are obtained, the method is currently limited to perfectly specular paths only and an automatic choice of an appropriate set of sparse samples is not obvious.

A related approach, proposed by Igehy [44], computes ray differentials. In a classical ray tracing setting, an eye path only depends on the image plane coordinates of the path. Any other vertex or direction is directly determined by these coordinates by ray transfer and perfectly specular reflection or refraction.

Igehy computes partial derivatives of the vertices and directions with respect to the position on the image plane. These derivatives give the sensitivity of a vertex or direction in terms of the image plane coordinates and are used to approximate the footprint of a pixel in a path vertex. Since the differentials are based on

simple calculus, derivative procedures for pixel sampling, perfectly specular reflection and refraction are easily obtained from the original sampling procedures.

The ray differentials are applied to texture filtering. Instead of using supersampling to reduce texture aliasing, the texture is filtered locally on the surface over the pixel footprint (without the need to trace more paths). Since the vertex derivatives take into account minification or magnification of textures (due to the lens-like effects of reflection and refraction), the use of the pixel footprint delivers a very adequate texture filtering.

The use of standard calculus for path derivatives forms the main strength of ray differentials (and specular path perturbations). It easily adapts to complex scenes even with non-physical properties such as interpolated surface normals.

Path differentials extend ray differentials to arbitrary scenes including sampled BSDFs and area light sources. Therefore, path differentials are applicable to any global illumination algorithm based on path sampling. The problem dealt with by path differentials is inherently more difficult because a path is dependent not just on the image plane coordinates, but on many other variables introduced while sampling vertices and directions in a path. Path differentials must take into account the full dimensionality of the global illumination problem. This requires an extension of the notion of a footprint —it's not merely the pixel footprint—, and new derivative procedures.

Conclusion Differential techniques have the advantage over path extension that a ray remains a point sample. The robust and efficient intersection tests developed for ray tracing need no change for the differential techniques. This explains their better handling of complex scenes. Point samples, on the other hand, ignore visibility changes over the estimated footprint. While this may result in blurring or some artefacts, the handling of complex scenes largely outweighs this limitation. This is why ray differentials have been adopted rapidly in commercial rendering systems.

7.3 Footprints for local variance reduction

This section gives a general definition of a footprint of a path in a full global illumination setting and gives an overview of the footprint approximation using path differentials.

7.3.1 Paths as a function of unit random variables

Tracing paths involves the sampling of new vertices and directions. A newly sampled vertex or direction depends on the previous vertices and directions in the path and possibly on some new variables. Figure 7.2 illustrates the sampling of a short light path. The first vertex in the path, $\mathbf{x}_1$, is a point chosen on the light source. For an area light source, the vertex is generated by a 2D sampling event using variables u_1, u_2.

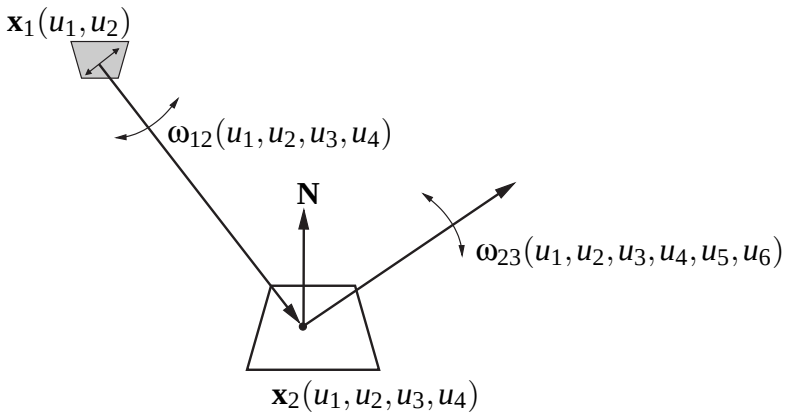


Figure 7.2: Each vertex or direction in a path is a function of a number of path variables. For this light path, two variables determine a vertex $\mathbf{x}_1$ on the light source. Two extra variables determine the direction ω_{12}, and so on.

Then a new direction ω_{12} is sampled according to some angular distribution function. This is again a 2D sampling event, introducing variables u_3, u_4, that, for example, determine the azimuth and elevation angles for the new direction. The vertex $\mathbf{x}_2$ is found by tracing a ray from $\mathbf{x}_1$ in the direction ω_{12}. This does not introduce new variables, as the new vertex is uniquely defined by $\mathbf{x}_2$ and ω_{12}. The new direction ω_{23} again introduces new variables u_5, u_6 and so the tracing of the path continues.

As such, each vertex or direction in a path can be seen as applying a function to the generating variables. We define 'g' as a function that generates a vertex in a path, and 'h' as the corresponding function that generates directions in a path:

$$\begin{aligned} \mathbf{x}_1 &= g(u_1,u_2)\,, \\ \omega_{12} &= h(\mathbf{x}_1,u_3,u_4) = h(u_1,u_2,u_3,u_4)\,, \\ \mathbf{x}_2 &= g(\mathbf{x}_1,\omega_{12}) = g(u_1,u_2,u_3,u_4)\,, \\ &\dots \end{aligned} \tag{7.1}$$

A flexible definition for the functions g and h will be adopted: they can take any number of variables as parameters, but also other vertices or directions. The exact definition will be clear from the notation used.

Without loss of generality, we choose the unit interval as the domain of all the variables u_k. In this case, g and h represent a mapping from $[0,1]^M$ to $\mathbb{R}^3$ (object-space) and $\Omega_{4\pi}$ (directions) respectively, with M the number of variables in the path. This choice maps nicely to Monte Carlo sampling: When paths are generated stochastically, importance sampling is used to transform uniform random numbers in the unit interval to a desired distribution. The variables u_k represent these random numbers that are used in importance sampling.

In general, the path generation functions map a single point in the M-dimensional unit hypercube to a

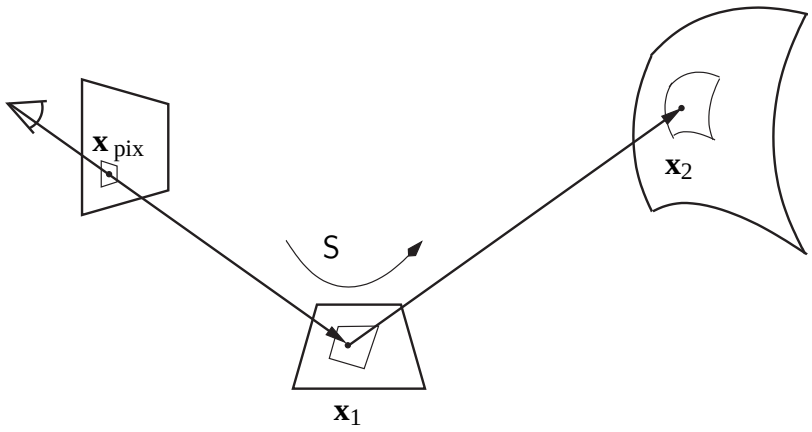


Figure 7.3: An example of exact footprints in classical ray tracing. A neighborhood around the starting ray is projected onto the first planar surface as a quadrilateral. All the reflected rays hit the curved surfaces, forming a curved footprint.

vertex or direction in the scene:

$$g : [0,1]^M \to \mathbb{R}^3 : \mathbf{x} = g(\mathbf{u}), \tag{7.2}$$

$$h : [0,1]^M \to \Omega_{4\pi} : \omega = h(\mathbf{u}), \tag{7.3}$$

$$\text{with } \mathbf{u} = [u_1 u_2 \dots u_M]^\top .$$

Since a vertex consists of three coordinates, g (and h) can be split into three components g_x, g_y and g_z: one function for each coordinate.

7.3.2 Footprint definition

Until now we have used the term *footprint* intuitively as a certain region of influence around a vertex. In this section we will give an exact definition of what we mean by a footprint.

Let $\mathbf{x} = g(\mathbf{u})$ be a vertex in a path. Consider for each variable u_k in $\mathbf{u}$ a certain *perturbation interval* Δu_k. For any perturbation $\varepsilon_k \in [-\Delta u_k/2, \Delta u_k/2]$, the vertex $\mathbf{x}$ moves over a small distance:

$$\mathbf{x} + \delta\mathbf{x}_k = g(u_1, \dots, u_k + \varepsilon_k, \dots, u_M).$$

Given a perturbation interval Δu_k for each variable u_k, the footprint $\mathcal{F}_\mathbf{x}$ in $\mathbf{x}$ is defined as *the set of all vertices that are reachable from* $\mathbf{x}$ *by a perturbation within the given perturbation intervals:*

$$\mathcal{F}_\mathbf{x} = \{\mathbf{y} = g(\mathbf{v}) | \forall k = 1 \dots M : |v_k - u_k| < \Delta u_k/2\}. \tag{7.4}$$

In other words, the perturbation intervals define a finite neighborhood around $\mathbf{u}$ in the domain of the path (an M-dimensional unit hypercube) This neighborhood is projected to a set of vertices in object space by the path generating function g. This set of vertices forms the footprint, that relates a neighborhood in the domain with a neighborhood in object-space.

In figure 7.3, a simple example is shown for classical ray tracing. Two variables determine a position on the image plane, for example the center of a pixel $\mathbf{x}_{\text{pix}}$. A ray is traced and hits a surface in $\mathbf{x}_1$. A neighborhood in the variable domain determines a rectangle on the image plane around the sampled position $\mathbf{x}_{\text{pix}}$ (e.g., the complete pixel). The projection of this neighborhood determines a quadrilateral on the surface in $\mathbf{x}_1$. The perfectly specular reflection does not introduce extra variables and the footprint is transferred along the reflected ray. The reflected ray hits a curved surface in $\mathbf{x}_2$, resulting in a curved footprint.

Equation (7.4) gives a very general definition of a footprint that uses g to determine all the points in the footprint:

- A footprint can consist of disjoint parts, for example when there is a change in visibility near the path that splits the footprint.
- The footprint can be non planar, for example for vertices that lie on a curved surface. The footprint will follow the curvature of the surface.

In general the exact footprint is very hard to compute, and approximations need to be made. The previous section, §7.2, summarized some previous approaches to compute the footprint, but these were limited to perfectly specular scattering.

7.3.3 Overview of footprint estimation using path differentials

Path differentials, developed in this chapter, provide a first order Taylor approximation of a footprint that is easily computed for complex scenes with arbitrary material properties.

For the Taylor approximation, partial derivatives of the path vertices (and directions) are computed for each variable in the path. The magnitude of the partial derivative determines the sensitivity of the vertex in terms of the variables. Multiplying the derivatives with the corresponding perturbation intervals results in the differential vectors, from which the footprint will be constructed.

A very important choice when computing footprints, is the size of the perturbation intervals Δu_k. The intervals should be small enough to ensure coherence of the illumination contributions over the footprint. On the other hand, the intervals should be large enough to enable an effective anti-aliasing or variance reduction, for example by filtering textures over the footprint.

Algorithm 2 (page 104) gives an overview of the footprint computation when tracing paths. In the following sections, we will detail each aspect of the footprint computation: The mathematics behind the first order Taylor approximation and a convenient representation of the footprint are given in §7.4. The details on the computation of partial derivatives are given in §7.5. Suitable heuristics for the perturbation intervals are proposed in §7.6.

Algorithm 2 Estimation of the footprint using path differentials

1. Trace a path $\bar{\mathbf{x}}$
2. For each vertex $\mathbf{x}_i$ in the path (depending on M variables u_k):
   (a) Compute all partial derivatives ($\frac{\partial \mathbf{x}_i}{\partial u_k}$) (§7.5)
   (b) Determine appropriate perturbation intervals Δu_k for this vertex (section §7.6)
   (c) Compute the differential vectors $= \frac{\partial \mathbf{x}_i}{\partial u_k} \Delta u_k$
   (d) Estimate the footprint using the differential vectors (§7.4)
3. Evaluate the contribution of the path, using the footprint for anti-aliasing or noise reduction (Applications, §7.7 and §7.8)

7.4 Footprint approximation using partial derivatives

This section develops a footprint approximation based on a first order Taylor expansion. The theory will be given for the footprint of a vertex in a path. For a direction a similar derivation can be given using h instead of g.

7.4.1 Differential vectors

The footprint was defined in terms of vertex perturbations (Eq. 7.4). These perturbations can be estimated using a Taylor expansion of g.

Given $\mathbf{x} = g(u_1, \ldots, u_M)$, a small perturbation ε_k applied to a variable u_k moves the vertex over a small distance:

$$\mathbf{x} + \delta\mathbf{x}_k = g(u_1, \ldots, u_k + \varepsilon_k, \ldots, u_M)\,.$$

This change can be approximated using a first order Taylor expansion:

$$\delta\mathbf{x}_k \approx \frac{\partial g(u_1, \ldots, u_k, \ldots, u_M)}{\partial u_k} \varepsilon_k\,.$$

The magnitude of the partial derivative determines the sensitivity of the vertex in terms of u_k. A large partial derivative means that a small change in the variable results in a large displacement of the vertex.

For a general perturbation $\boldsymbol{\varepsilon} = [\varepsilon_1, \ldots, \varepsilon_M]^\top$ that changes several or all the variables, the displacement of $\delta\mathbf{x}$ is given by:

$$\delta\mathbf{x} = \sum_{k=1}^{M} \delta\mathbf{x}_k \approx \sum_{k=1}^{M} \frac{\partial g(\mathbf{u})}{\partial u_k} \varepsilon_k\,.$$

This perturbation can be written concisely in terms of the Jacobian matrix of g:

$$\delta\mathbf{x} \approx \mathrm{J}\begin{pmatrix} g(\mathbf{u}) \\ \mathbf{u} \end{pmatrix} \boldsymbol{\varepsilon},$$

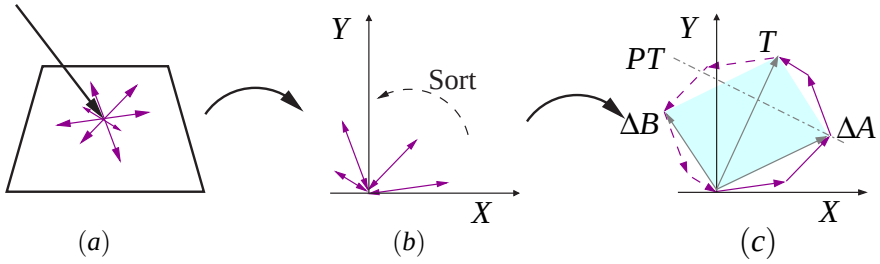


Figure 7.4: Each path vertex has several differential vectors lying in the same plane (a). These vectors can be used to approximate the footprint of a path. By transforming to the $X-Y$ plane (b), the polygonal footprint approximation can be constructed by combining the vectors in a particular order (c). A good approximation is given by vectors ΔA and ΔB.

where $\boldsymbol{\varepsilon}$ is a vector of length M and the $3 \times M$ Jacobian matrix is defined as

$$\mathrm{J}\begin{pmatrix} g(\mathbf{u}) \\ \mathbf{u} \end{pmatrix} = \begin{bmatrix} \frac{\partial g_x(\mathbf{u})}{\partial u_1} & \cdots & \frac{\partial g_x(\mathbf{u})}{\partial u_M} \\ \frac{\partial g_y(\mathbf{u})}{\partial u_1} & \cdots & \frac{\partial g_y(\mathbf{u})}{\partial u_M} \\ \frac{\partial g_z(\mathbf{u})}{\partial u_1} & \cdots & \frac{\partial g_z(\mathbf{u})}{\partial u_M} \end{bmatrix}.$$

The set of vertices $\mathbf{x}+\delta\mathbf{x}$ that can be reached by perturbations within the supplied perturbation intervals, forms the first order Taylor approximation of the footprint defined in equation (7.4).

If we consider only the perturbations of a single variable: $\varepsilon_k \in [-\Delta u_k/2, \Delta u_k/2]$, the set of perturbed vertices forms a line segment centered around the vertex $\mathbf{x}$. This line segment is given by the vector $\Delta\mathbf{x}_k$:

$$\Delta\mathbf{x}_k = \frac{\partial g(u_1,\ldots,u_k,\ldots,u_M)}{\partial u_k}\Delta u_k\,.$$

The vectors $\Delta\mathbf{x}_k$ are called the *differential vectors*. For each of the variables, such a differential vector can be constructed.

7.4.2 From differential vectors to footprint

In §7.5.2 it will be shown that the partial derivatives $\frac{\partial g(\mathbf{u})}{\partial u_k}$ are all tangent to the surface in $\mathbf{x}$ and, consequently, that all differential vectors lie in the same plane. These planar differential vectors can be used to construct a footprint approximation. This is shown in figure 7.4.

Any possible perturbation $\mathbf{x}'$ (within the perturbation intervals) of a vertex $\mathbf{x}$ can be written as a linear combination of the differential vectors:

$$\mathbf{x}' = \mathbf{x}+\delta\mathbf{x} = \mathbf{x}+\sum_{k=1}^{M}\delta\mathbf{x}_k \approx \mathbf{x}+\sum_{k=1}^{M}\gamma_k\Delta\mathbf{x}_k\,,$$

where $\gamma_k \in [-0.5, 0.5]$ since the footprint is centered around $\mathbf{x}$.

The area defined by a combination of a number of line segments is given by the Minkowski sum ($\oplus$) of the line segments. Thus, the footprint approximation is given by the Minkowski sum of the differential vectors:

$$\mathcal{F}_\mathbf{x} = \bigoplus_k \Delta\mathbf{x}_k = \{\mathbf{x}' = \sum_{k=1}^{M}\gamma_k\Delta\mathbf{x}_k | \; -0.5 < \gamma_k < 0.5\}\,.$$

When only two variables determine a path, as is the case in the classical ray tracing setting used by Igehy [44], there are only two differential vectors. The footprint is then given by the parallelogram formed by the two vectors.

In general the Minkowski sum is a polygon with $2M$ edges. Each differential vector appears twice as an edge. This polygon can be constructed as follows (see figure 7.4):

- The differential vectors all lie in the same plane perpendicular to the normal in $\mathbf{x}$ and are centered around $\mathbf{x}$ (Figure 7.4 (a)).
- In a first step, transform all vectors to the $X-Y$ plane. The vectors start in the origin and point towards the positive Y direction (Figure 7.4 (b)).
- Sort the vectors according to the angle made with the X-axis. (Figure 7.4 (b), dotted arrow).
- The first half of the polygon is constructed by adding the vectors one by one in the sorted order. The resulting poly-line reaches the top of the polygon (Figure 7.4 (c), solid lines, purple).
- The remainder of the polygon is formed by subtracting the vectors, again in the sorted order, starting from the top (Figure 7.4 (c), dashed lines, purple).

Since each sampling event, such as light source sampling and reflections, can introduce two new variables, the total number of variables M can become large for long paths. The footprint, which has $2M$ edges, becomes impractical for more that 2 differential vectors. To perform operations such as texture filtering over the footprint, a convenient representation is needed. Therefore, we compute two representative vectors ΔA and ΔB that give a good approximation of the covered area. The construction of the representative vectors is also shown in figure 7.4 (c):

- Let T be the top of the polygonal footprint. T is the sum of all differential vectors.
- Construct a vector PT perpendicular to T.
- $\Delta A =$ the sum of all differential vectors $\Delta\mathbf{x}_k$ that comply to $\Delta\mathbf{x}_k \cdot PT > 0$. This gives the rightmost vertex of the polygon (or leftmost, depending on the direction of PT).
- $\Delta B = T - \Delta A$

Constructing the representative vectors processes all differential vectors twice: Once for computing T and a second time for ΔA. No sorting is needed for computing the sums, making the construction a linear operation in the number of differential vectors (M vectors).

The parallelogram formed by ΔA and ΔB (Figure 7.4 (c), cyan) is inscribed in the footprint and provides a conservative and convenient estimate of the footprint of the path in $\mathbf{x}$. If coherence is ensured over the footprint, it will also be ensured over the (smaller) parallelogram.

Thus, if we can compute all the partial derivatives and find suitable perturbation intervals, we can estimate the footprint with the above procedure.

7.4.3 A footprint based on convolution

We defined a footprint as the projection of a small neighborhood in the path domain by the path generation function g. All points reachable around a path vertex by perturbations within the given perturbation intervals are part of the footprint.

Other footprint definitions are possible and an interesting alternative uses convolution:

- Assign a certain filter kernel to each perturbation interval Δu_k. A box filter, for example, would assign an equal weight to the center and the endpoints of the interval. A Gaussian filter assigns a larger weight to the center.
- The resulting filter for the full neighborhood $\Delta\mathbf{u}$ around a sample $\mathbf{u}$ in the path domain, is given by the convolution of the separate interval filters. While this filter could be evaluated by explicitly sampling and weighting several samples $\mathbf{u}$, the filter can also be approximated using the differential vectors:
  - For a single perturbation interval, a corresponding filter can be defined over the differential vectors. The box or Gaussian is stretched over the length of the differential vector.
  - The convolution footprint is approximated by the convolution of the filters over all the differential vectors. The convolution footprint defines a new filter locally around a path vertex

Gaussian filters might be interesting to use with a convolution footprint: The convolution of each pair of differential vectors forms an elliptic Gaussian around the vertex. Since the convolution of elliptic Gaussians is again an elliptic Gaussian [39, p.55],[6], an exact convolution could be computed.

Such a convolution footprint could be useful for the texture filtering application that we will present further on. For our other applications a convolution footprint is less useful.

7.5 Computing partial derivatives

In this section the actual computation of the partial derivatives of vertices and directions in a path will be discussed.

Tracing paths involves the sampling of new directions and vertices. For stochastic sampling, importance sampling is used to generate directions and vertices according to some probability density function (pdf). For each procedure, each sampling event, that computes a new vertex or direction, a partial derivative procedure must be developed.

Consider for example a scattering event that generates a new direction ω'. In general, this direction depends on a previous vertex and direction (if these exist) and on two new variables (u, v) introduced in the

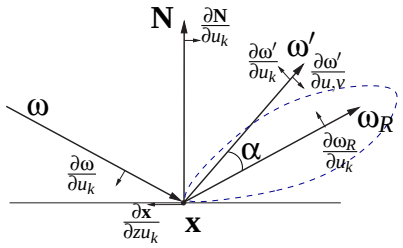


Figure 7.5: Partial derivatives for Phong lobe sampling. ω' has derivatives for previous sampling variables u_k but also for new variables u, v from BRDF sampling.

2D sampling event:

$$\omega' = h(\omega, \mathbf{x}, u, v)\,.$$

Partial derivatives of ω' can be computed by simply deriving h for all previously introduced variables u_k and the new variables u, v. Since ω and $\mathbf{x}$ depend on the previous variables u_k, the partial derivatives $\frac{\partial \omega}{\partial u_k}$ and $\frac{\partial \mathbf{x}}{\partial u_k}$ were computed before and can be used to compute the new partial derivatives $\frac{\partial \omega'}{\partial u_k}$ using the chain rule for derivatives of nested functions [86]. Figure 7.5 illustrates the sampling of the new direction, and the partial derivatives that must be computed for a Phong lobe (pdf $\sim f_r \sim \cos^S \alpha$).

Other relevant sampling events in path generation are pixel sampling, (ray) transfer and light sampling. We will briefly discuss each event and the corresponding derivatives below. Detailed information on the derivative computation for these events can be found in the appendix (§7.A). In this overview, $\mathbf{x}$ denotes the previous vertex, ω the previous direction, $\mathbf{x}'$ a newly sampled vertex, and ω' a new direction. Variables u_k denote variables introduced in previous sampling events, while u and v are new variables for the sampling event under consideration.

7.5.1 Pixel sampling

Pixel sampling occurs when tracing an eye path. Let the initial vertex $\mathbf{x}_0$ of the path be the eye[1]. Pixel sampling generates a ray direction ω' based on a (randomly) chosen point in the pixel. Computation of the two derivatives of ω' is straightforward and given in the appendix (§7.A).

7.5.2 Transfer

Transfer computes a new vertex in a path by tracing a ray from the previous vertex $\mathbf{x}$ in the direction ω. The new vertex is given by

$$\mathbf{x}' = \mathrm{rc}(\mathbf{x} \to \omega) = \mathbf{x} + t\omega\,,$$

with t the traveled distance to the new vertex. $\mathbf{x}'$ depends on $\mathbf{x}$ and ω, but no new variables are introduced.

The partial derivatives are given by

$$\frac{\partial \mathbf{x}'}{\partial u_k} = \frac{\partial \mathbf{x}}{\partial u_k} + t\frac{\partial \omega}{\partial u_k} + \frac{\partial t}{\partial u_k}\omega\,,$$

[1] We assume a pinhole camera model. Otherwise, the generation of the initial eye vertex is already a sampling event that generates a vertex on the aperture of the camera.

with

$$\frac{\partial t}{\partial u_k} = -\frac{(\frac{\partial \mathbf{x}}{\partial u_k} + t\frac{\partial \omega}{\partial u_k}) \cdot \mathbf{N}_g}{\omega \cdot \mathbf{N}_g},$$

where $\mathbf{N}_g$ is the geometric normal of the surface in $\mathbf{x}'$ (see §7.A for more information).

Some interesting properties about these derivative expressions are the following:

- The expressions show that the partial derivatives of $\mathbf{x}'$ are easily computed from the previously computed derivatives of $\mathbf{x}$ and ω.
- The resulting derivatives $\frac{\partial \mathbf{x}'}{\partial u_k}$ are perpendicular to the geometric normal in $\mathbf{x}'$:

$$\begin{aligned}\frac{\partial \mathbf{x}'}{\partial u_k} \cdot \mathbf{N}_g &= (\frac{\partial \mathbf{x}}{\partial u_k} + t\frac{\partial \omega}{\partial u_k}) \cdot \mathbf{N}_g + \frac{\partial t}{\partial u_k}(\omega \cdot \mathbf{N}_g) \\ &= (\frac{\partial \mathbf{x}}{\partial u_k} + t\frac{\partial \omega}{\partial u_k}) \cdot \mathbf{N}_g - (\frac{\partial \mathbf{x}}{\partial u_k} + t\frac{\partial \omega}{\partial u_k}) \cdot \mathbf{N}_g \\ &= 0.\end{aligned}$$

  This means that all partial derivatives of a vertex generated by tracing a ray are co-planar. This property was used when constructing the footprint from the differential vectors (§7.4.2).

7.5.3 Scattering

A scattering event generates a new direction ω', given a direction ω incident in a vertex $\mathbf{x}$. Usually the pdf for direction sampling is chosen proportional to the BSDF or, even better, the BSDF times the cosine. Partial derivatives of the resulting importance sampling procedure must be computed, both for previous variables u_k and new variables u and v.

An example for a glossy Phong BRDF was shown in figure 7.5. The new direction ω' is distributed according to $\cos^S \alpha$ around the perfectly reflected direction ω_R. Since ω' depends on ω_R, and ω_R depends on ω, $\frac{\partial \omega'}{\partial u_k}$ can be different from zero. The new derivatives $\frac{\partial \omega'}{\partial u,v}$ depend on the specific importance sampling procedure. For Phong lobe sampling, the derivatives are given in the appendix (§7.A).

As an illustration, we will show a simpler case where the new direction is sampled uniformly over the hemisphere. The importance sampling procedure $\omega' = h(u,v)$ for uniform direction sampling is given by:

$$\begin{aligned}&\varphi = 2\pi u, \quad \cos\theta = 1 - v \qquad \text{with } u,v \in [0,1], \\ &\omega' = h(u,v) = (\cos\varphi\sin\theta, \sin\varphi\sin\theta, \cos\theta).\end{aligned}$$

The derivatives are easily computed from this procedure:

$$\begin{aligned}&\frac{\partial \omega'}{\partial u} = \frac{\partial \omega'}{\partial \varphi}\frac{\partial \varphi}{\partial u} = (-2\pi\sin\varphi\sin\theta, 2\pi\cos\varphi\sin\theta, 0), \\ &\frac{\partial \omega'}{\partial v} = \frac{\partial \omega'}{\partial \cos(\theta)} \times -1 = (\cos\varphi\cos\theta/\sin\theta, \sin\varphi\cos\theta/\sin\theta, -1), \\ &\text{where we have used } \frac{\partial \sin\theta}{\partial \cos\theta} = \frac{\partial\sqrt{1-\cos^2\theta}}{\partial\cos\theta} = -\cos\theta/\sin\theta.\end{aligned}$$

The partial derivatives are perpendicular to the new direction ω', which can be shown by computing the dot product of ω' with the derivatives. This means that all derivatives of directions are co-planar. Similar to the computation of the footprint for a vertex, a footprint for a direction could be computed.

Note that for the special case of perfectly specular reflection or refraction, the outgoing direction is determined completely by the incoming direction ω and the vertex $\mathbf{x}$, so that no new variables (or derivatives) are introduced. This special case was already handled by Igehy in [44].

7.5.4 Light source sampling

When light paths are constructed, a starting point must be chosen on a light first. Then a light direction is sampled. Again, the derivatives are easily computed from the specific sampling procedure (Appendix §7.A).

7.5.5 Other sampling events

Any other sampling event that is not covered here (e.g., scattering in a participating medium) can usually be derived simply from the sampling procedure. This is the advantage of using simple calculus (for example, instead of differential geometry) to compute the footprint approximation.

Some sampling procedures, however, use rejection sampling (§3.2.3.2). In rejection sampling a higher dimensional sample is generated and tested for acceptance. It is not possible to simply differentiate such a procedure. While it might be possible to develop derivative procedures in another way, based on the pdf itself instead of the sampling procedure, we have not closely investigated this issue because all common sampling procedures in rendering can be performed without rejection sampling.

7.6 Choosing perturbation intervals

In the previous section it was explained how partial derivatives can be computed for different sampling events. These derivatives constitute one factor in the differential vectors that make up the footprint. The other factor is given by the perturbation intervals. This section proposes several heuristics for choosing the perturbation intervals.

Recall that the perturbation intervals Δu_k determine a small neighborhood around a point $\mathbf{u}$ in the M-dimensional domain of $g(\mathbf{u})$. The footprint is the projection of this neighborhood into object space through the path generation function $g(\mathbf{u})$.

The choice of the perturbation intervals directly influences the size of the footprint. Therefore, these intervals should be chosen in such a way that coherence is ensured over the footprint. For example, in a texture filtering application, a large footprint will blur the texture seen in the image, while a small footprint will not reduce the noise.

There are several factors that can be taken into account to determine appropriate perturbation intervals:

- **Number of samples :** The number of samples determines the density of the samples in the path domain. A higher density means that the distance between neighboring samples will be smaller. We

use the number of samples to estimate the expected distance to neighboring samples and use that distance as the perturbation interval (§7.6.1).

- **Path contribution :** In our applications we consider the contribution of a path to be constant over the footprint. This approximation breaks down when the contribution changes too much over the footprint. In §7.6.2 we will compute the *path gradient* and use it to limit the size of the perturbation intervals when the gradient is too large.

- **Second order derivatives :** The second order derivative indicates how well the first order derivative, the tangent vector, approximates the perturbation of the vertex. This can also be used to determine the intervals (§7.6.3).

- **Visibility :** Since the footprint is based on derivatives of a point sample, visibility changes near the sampled path are ignored in the footprint. This could violate the coherence assumption of the footprint. Currently we do not use visibility to determine perturbation intervals. How visibility could be incorporated into the footprint estimate is discussed in §7.9.

In practice we use a combination of the heuristic based on the number of samples, the path gradient and the second derivatives (§7.6.4).

7.6.1 Number of samples

The heuristic based on the number of samples chooses intervals Δu_k that correspond to the expected distance to a neighboring sample (i.e., the closest sample $(u_1, \ldots, u_M)$ differing only in u_k). Multiplying the intervals with the corresponding partial derivative results in differential vectors that approximate the expected distance to a neighboring path. For example, after sampling a diffuse reflection, differential vectors will usually be larger than after sampling a glossy reflection, because the partial derivatives are larger while the deltas are the same. The rays get 'spread out' more by the diffuse BRDF.

This heuristic is also used by Igehy in ray differentials and, in a slightly different setting, by Collins when the distance to neighboring rays is tracked explicitly. They used classical ray tracing, which only requires estimating the distance to neighboring samples in the image plane or neighboring samples for rays sent out by a point light source. For arbitrary path sampling all dimensions in the path domain must be considered to determine a neighboring sample.

We distinguish two cases, local intervals and global intervals. Recall that each variable u_k has a unit interval domain, which will facilitate the choice of intervals.

7.6.1.1 Local intervals

Local intervals choose Δu_k based on the number of samples that were used in the sampling event that introduced the variable u_k.

For example, if N samples per pixel are traced, these samples are distributed over the unit square, introducing two variables u_1 and u_2. Δu_1 and Δu_2 should be chosen as an approximate distance to a neighboring sample. For regularly spaced samples this distance is $1/\sqrt{N}$ for both intervals. We have found this distance to be useful for stochastic sampling also.

If extra information about the sampling process is known (e.g., nonuniform stratification), different values for Δu_1 and Δu_2 may be better.

Suppose that, after pixel sampling, the path is continued and a scattering event takes place. For scattering (another 2D sampling event) a splitting factor N' determines how many scattered samples are spawn. Again we choose the interval size to be $1/\sqrt{N'}$. If many samples are spawn, the corresponding differentials will be smaller.

Similar intervals are used for other scattering events.

7.6.1.2 Global intervals

The local intervals do not work well for path tracing (or light tracing) that uses a large number of samples per pixel, but only a single sample for scattering.

In this case, we consider the complete M-dimensional domain (a unit hypercube), and consider the samples to be spaced evenly over the domain. An estimate of the distance to a neighboring sample in one dimension is now given by $1/\sqrt[M]{N}$. All Δu_k are chosen equal. Longer paths will have larger intervals, as N samples have to be distributed over a higher dimensional domain.

It is also possible to incorporate *Russian roulette*, an unbiased way to limit the length of paths (§3.3.4), into this approach. The absorption probabilities P_{rr} used in each vertex $\mathbf{x}_i$ are accumulated along the path, and intervals are computed as $1.0/\sqrt[M]{N \times \prod_i P_{rr}(\mathbf{x}_i)}$. Thus the number of samples for 'this kind of path' is decreased by the absorption probabilities, and intervals grow larger. This accounts for the decrease of path density in an infinitesimal neighborhood around the sample point $\mathbf{u}$.

This approach works well for path tracing and particle tracing as will be demonstrated in the applications. However, we observed that longer paths tend to have large intervals, due to the 'curse of dimensionality': The number of samples per dimension, $\sqrt[M]{N}$, becomes very small for larger M. Therefore, we developed additional heuristics based on the path gradient and the second order derivatives.

A note on adaptive sampling Both the local and global interval heuristics assume that the total number of samples is known beforehand. Special care must be taken for adaptive sampling. Adaptive sampling starts with a small batch of samples and increases the number based on the observed variance of the sample contributions. Using the small, initial number of samples in the interval heuristic may cause a large footprint, leading to a small variance but a large bias. Due to the small variance, adaptive sampling would decide the estimate is accurate enough and would not increase the number of samples enough, leading to a

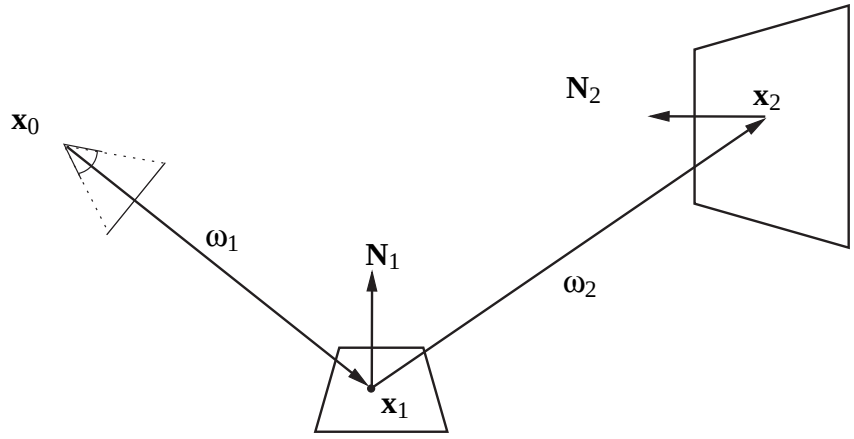


Figure 7.6: Symbols used in the derivation of the path gradient.

strong bias in the solution. Therefore, in adaptive sampling the interval heuristic should use a larger number of samples, for example the maximum number of samples that would ever be used by the adaptive sampling procedure.

7.6.2 Path gradient

In this section the path gradient will be developed. The path gradient is a vector of partial derivatives of the *path evaluation*, which is a function of a path that determines the contribution to the quantity (e.g., the pixel flux) that we want to compute. The derivatives give the rate of change of the path evaluation when perturbations are applied. The path gradient will be used to estimate perturbation intervals: a large gradient means that the evaluation changes rapidly and that small intervals should be used.

The technique is presented for eye paths, but it is equally applicable to light paths. First we will discuss the path evaluation function (§7.6.2.1), then the computation of partial derivatives (§7.6.2.2) and a heuristic for choosing the perturbation intervals (§7.6.2.3). Implementation issues are given in §7.6.2.4 and some additional discussion is provided in §7.6.2.5.

7.6.2.1 Path evaluation

The light flux reaching the eye $\mathbf{x}_0$ through a pixel is given by the following measurement equation (see §2.6):

$$I_{\text{pix}} = \int_{\Omega_{\text{pix}}} W_{\text{e}}(\mathbf{x}_0 \to \omega_1) L_{\text{i}}(\mathbf{x}_0 \leftarrow \omega_1)\, \mathrm{d}\omega_1 \,.$$

The unknown radiance L_{i} can be expanded by tracing a ray to $\mathbf{x}_1$ and substituting the radiance transport equation 2.1 (The symbols used are explained in figure 7.6):

$$I_{\text{pix}} = \int_{\Omega_{\text{pix}}} W_{\text{e}}(\mathbf{x}_0 \to \omega_1) \Big[L_{\text{e}}(\mathbf{x}_1 \to \omega_1) + \int_{\Omega_{4\pi}} f_s(\mathbf{x}_1, \omega_1 \leftarrow \omega_2))\, (\mathbf{N}_1 \cdot \omega_2) L_{\text{i}}(\mathbf{x}_1 \leftarrow \omega_2)\, \mathrm{d}\omega_2 \Big]\, \mathrm{d}\omega_1 \,.$$

Again the unknown radiance L_i can be substituted by the radiance transport equation. Each new scattering introduces a BSDF and a cosine factor in the integrand. To simplify notation, we introduce the *eye-path evaluation function* $f_n^{(w)}$ that contains the product of the self-emitted importance and all the BSDF and cosine factors for a path of length n. The full evaluation of a path $\bar{\mathbf{x}}$ with a length of at least n can now be written as:

$$f(\bar{\mathbf{x}}) = f_n^{(w)}(\mathbf{x}_n \to \omega_{n+1}) \cdot L_\mathrm{i}(\mathbf{x}_n \leftarrow \omega_{n+1})\,. \tag{7.5}$$

The function $f_n^{(w)}$ is defined recursively as:

$$f_0^{(w)}(\mathbf{x}_0 \to \omega_1) = W_\mathrm{e}(\mathbf{x}_0 \to \omega_1)\,,$$

$$f_n^{(w)}(\mathbf{x}_n \to \omega_{n+1}) = f_{n-1}^{(w)}(\mathbf{x}_{n-1} \to \omega_n)\, f_s(\mathbf{x}_n, \omega_n \leftarrow \omega_{n+1}))\, (\mathbf{N}_n \cdot \omega_{n+1})\,. \tag{7.6}$$

This path evaluation function fits into eye path tracing as follows:

- When an eye path is extended, the evaluation of the path accumulates an extra BSDF and cosine term. Thus, path extension corresponds to going from $f_n^{(w)}$ to $f_{n+1}^{(w)}$.
- Actual contributions are made to the image when a path hits a light source, or when light sources are sampled directly. The former case corresponds to replacing the incoming radiance in equation (7.5) by the self-emitted radiance of the light source that is hit: $L_\mathrm{i}(\mathbf{x}_n \leftarrow \omega_{n+1}) = L_\mathrm{e}(\mathbf{x}_{n+1} \to \omega_{n+1})$

  The latter case samples a point $\mathbf{x}_{n+1}$ on a light source directly, which determines the direction ω_{n+1}.

Note that all the vertices and directions present in the path evaluation depend on the variables u_k used to sample the path.

7.6.2.2 Path gradient computation

The path gradient of a path $\bar{\mathbf{x}}$ of length n (with M variables) is defined as the vector of partial derivatives of the path evaluation:

$$\text{Path gradient} = [\frac{\partial f(\bar{\mathbf{x}})}{\partial u_1} \ldots \frac{\partial f(\bar{\mathbf{x}})}{\partial u_M}]^\top\,.$$

A partial derivative $\frac{\partial f(\bar{\mathbf{x}})}{\partial u_k}$ of the path evaluation indicates how fast the evaluation changes when the variable u_k changes.

Together with derivatives of vertices and directions, we also compute the path gradient. The individual partial derivatives of equation (7.5) are given by (for brevity, arguments are dropped):

$$\frac{\partial f}{\partial u_k} = \frac{\partial f_n^{(w)}}{\partial u_k} L + f_n^{(w)} \frac{\partial L}{\partial u_k}\,.$$

We compute *relative partial derivatives* by dividing this expression by the path evaluation f itself:

$$\frac{\partial f}{\partial u_k} / f = \frac{\partial f_n^{(w)}}{\partial u_k} / f_n^{(w)} + \frac{\partial L}{\partial u_k} / L\,.$$

The expression gives the *relative* change of the path evaluation in terms of u_k.

Because the common factors cancel out, the relative partial derivatives also provide a convenient way to compute the derivatives when extending a path (equation 7.6):

$$\frac{\partial f_n^{(w)}}{\partial u_k} / f_n^{(w)} = \frac{\partial f_{n-1}^{(w)}}{\partial u_k} / f_{n-1}^{(w)} + \frac{\partial f_s}{\partial u_k} / f_s + \frac{\partial \mathbf{N}_n \cdot \boldsymbol{\omega}_{n+1}}{\partial u_k} / (\mathbf{N}_n \cdot \boldsymbol{\omega}_{n+1}) .$$

Each factor in the path evaluation contributes a separate term to the relative partial derivative. Extending a path just requires addition of two terms to the sum accumulated up to that point.

The start of a path requires the computation of $\frac{\partial f_0^{(w)}}{\partial u_k} / f_0^{(w)}$. In our camera model (see §2.6), W_e is constant over the whole image plane, which makes the initial partial derivative term, $\frac{\partial f_0^{(w)}}{\partial u_k}$, zero. The end of a path, when a light source is hit or sampled directly, adds a final term to the path evaluation derivative. This final term accounts for the change of emitted light in terms of the variables u_k.

In appendix §7.A, details are provided on the exact computation of the path gradient derivatives for several sampling events.

7.6.2.3 Perturbation intervals using the path gradient

Recall that the perturbation intervals Δu_k determine the size of the neighborhood around a point $\mathbf{u}$ in the path domain. The relative change of the path evaluation $f(\mathbf{u})$ when a perturbation Δu_k is applied to $\mathbf{u}$, can be approximated as follows:

$$\Delta f_k \approx \left(\frac{\partial f}{\partial u_k}(\mathbf{u}) / f(\mathbf{u}) \right) \Delta u_k . \tag{7.7}$$

For example, a gradient Δf_k of 300% means the evaluation of the perturbed path can be three times higher than the non perturbed evaluation $f(\mathbf{u})$.

The same perturbation interval Δu_k also defines a differential vector in a certain vertex $\mathbf{x}$ in the path. Thus, Δf_k also gives the change of the path evaluation along the differential vector. In our applications, the evaluation of a path ($f(\mathbf{u})$) is considered to be constant over the footprint. Of course, f does change over the footprint and the path gradient indicates how much.

When the gradient is large, the evaluation will change significantly over the footprint, and coherence is not ensured. By constraining the relative change Δf_k to be within a certain threshold Δf_{max}, perturbation intervals that do ensure coherence, can be computed using equation (7.7):

$$\Delta u_k = \frac{\Delta f_{max}}{\frac{\partial f}{\partial u_k} / f} . \tag{7.8}$$

This heuristic determines the perturbation intervals based on the path gradient.

A very small gradient, however, can lead to arbitrary large perturbation intervals and corresponding footprints. Combination with the heuristic based on the number of samples can solve this problem (see §7.6.4).

7.6.2.4 Implementation

The computation of the path gradient is straightforward. The BSDF and cosine terms, which appear in the path evaluation, are easily expressed in terms of vertices and directions in the path. Partial derivatives of these vertices and directions are already computed for the differential vectors (§7.5). The additional cost for computing the path gradient is thus very small.

The choice of Δf_{max} determines how much the path evaluation is allowed to change over the differential vectors. Its choice is important because it determines the size of the path footprint directly. In our current implementation it is a user-defined parameter. For our applications we typically use values around 100%, meaning that the path evaluation may double over a single differential vector.

7.6.2.5 Discussion

The path evaluation function that we have used in the path gradient is the integrand of the expanded measurement function with respect to a spherical angle measure. There are other alternatives that we could have used as a path evaluation function:

- An integrand with respect to area measure, as was used in the formulation of the path integral in §2.9, could also be used. Such a formulation transforms $\mathrm{d}\omega_i$ into $\frac{\mathbf{N}_i \cdot \omega_i}{\|\mathbf{x}_i - \mathbf{x}_{i-1}\|^2}\,\mathrm{d}A_i$. The factor introduced by this transformation results in an extra term in the relative partial derivatives. The squared distance results in a derivative term that has a third power of the distance in the denominator (see also §7.A.2).

  We have tried computing the path gradient using the integrand with respect to area measure, but found that it does not make a big difference except close to corners where the distance between vertices becomes small and where the third power in the denominator causes a very large gradient. In our hierarchical radiosity application, this resulted in some undesirable fine subdivision near object corners.

- Another interesting function of a path is the *score function*. The score function of a path $\bar{\mathbf{x}}$ in Monte Carlo integration is $f(\bar{\mathbf{x}})/p(\bar{\mathbf{x}})$, where p is the pdf used for generating the path. It is in fact the score function that is evaluated and averaged when computing pixel fluxes with Monte Carlo ray tracing. This could be an interesting choice for tracking derivatives.

  Usually pdfs for direction sampling are chosen proportionally to the BSDF or the cosine. These factors cancel out and the score function only contains factors not used in path sampling. We have chosen to compute derivatives of f itself, and not of the score function for two reasons:

  – As will be shown more formally in the texture filtering application, path differentials provide a tool to estimate an *average* evaluation over the path footprint. This average accounts for all possible perturbations of a path within a small neighborhood in the path domain. These

perturbations are not distributed according to the pdf with which a path is generated; the average is just an integral over the neighborhood, that is approximated with the path footprint.

– When the pdf is chosen proportional to the BSDF (or the cosine), the path gradient includes a BSDF derivative term that is proportional to the derivative of the pdf. The outgoing direction, on the other hand, is generated by an importance sampling procedure that is based on the pdf. In other words, the partial derivatives of the procedure for direction sampling are used to estimate the footprint, while the partial derivatives of the pdf determine the path gradient and are used to limit the perturbation intervals.

  The derivative of the pdf is related to the *second order derivative* of the importance sampling procedure (and thus provides additional information). This is easy to show for a one-dimensional sampling, where x is sampled according to a pdf $p(x)$:

$$\begin{aligned} & x = P^{-1}(u) \\ \Rightarrow & \frac{\partial x}{\partial u} = \frac{1}{p(x(u))} \\ \Rightarrow & \frac{\partial^2 x}{\partial u^2} = \frac{-1}{p^2(x(u))} \frac{\partial p(x(u))}{\partial u} . \end{aligned}$$

  For a multi-dimensional sampling, the relation is more involved because the derivative of the Jacobian determined by the transformation must be computed (see also §3.2.3). The determinant computation in the Jacobian looses some information about the sampling of the separate components in a direction. For example, sampling according to the cosine of the angle with the normal results in a completely determined direction, while the pdf (the cosine itself) only takes into account the angle with the normal and not the azimuthal angle.

  Recently we performed some tests that compute second order derivatives of the sampling procedure explicitly. Details are given in the next section.

Note that for perfectly specular scattering $f_{n+1}^{(w)} = f_n^{(w)} \cdot f_s$ with f_s the *constant* specular reflection or refraction coefficient, derivatives of f_s are zero. As a result the path gradient does not yield any extra information for the method presented by Igehy.

7.6.3 Second order derivatives

The second order derivatives of vertices and directions in a path can also be used to estimate appropriate perturbation intervals: The second order derivative indicates how well the first order derivative approximates the perturbation of a vertex, so it can be used for error bounding.

7.6.3.1 Second order Taylor approximation

Let $\mathbf{x}$ be a vertex generated by $g(\mathbf{u})$. If only one variable u_k is considered, $\mathbf{x} = g(u_k)$ defines a parametric curve from $[0,1]$ to $\mathbb{R}^3$.

A second order Taylor approximation of $\mathbf{x}$ in terms of a perturbation Δu_k is given by

$$\mathbf{x} + \delta\mathbf{x}_k^{(2)} = \mathbf{x} + \frac{\partial g(u_k)}{\partial u_k}\Delta u_k + \frac{1}{2!}\frac{\partial^2 g(u_k)}{\partial u_k^2}\Delta u_k^2 ,$$

where $\delta\mathbf{x}_k^{(2)}$ indicates the second order perturbation of $\mathbf{x}$.

While this could be used for the computation of more accurate path perturbations, we use it for bounding the error on a first order Taylor approximation.

7.6.3.2 Perturbation interval heuristic

The difference between a first order and a second order Taylor approximation is given by the second order derivative term. This term also indicates the accuracy of a first order Taylor approximation. Instead of using this term directly to bound the error on the approximation, we bound the *relative error* on $\delta\mathbf{x}_k$ to a maximum error $\varepsilon_{\max}$:

$$\frac{\delta\mathbf{x}_k^{(2)} - \delta\mathbf{x}_k^{(1)}}{\delta\mathbf{x}_k^{(1)}} = \frac{\frac{1}{2!}\frac{\partial^2 g(u_k)}{\partial u_k^2}\Delta u_k^2}{\frac{\partial g(u_k)}{\partial u_k}\Delta u_k} \leq \varepsilon_{\max} .$$

Given the maximum relative error, the perturbation interval Δu_K can now be estimated as

$$\Delta u_k = \varepsilon_{\max}\frac{\frac{\partial g(u_k)}{\partial u_k}}{\frac{1}{2!}\frac{\partial^2 g(u_k)}{\partial u_k^2}} .$$

Thus, computing the second order derivatives of vertices and directions in a path enables another perturbation interval heuristic.

7.6.3.3 Implementation

We have not fully integrated second order derivatives throughout the whole path generation pipeline. A vertex is a function of a previous vertex and direction, which in turn are functions of other previous vertices and directions and so on. A vertex can thus be written as a number of nested functions depending on the path variables. Derivatives are computed using the chain rule, but this becomes increasingly complex for higher order derivatives. Therefore, second order derivatives require a significant amount of extra implementational and computational effort with respect to a first order derivative implementation.

In our current implementation, we only compute second order derivatives for new variables at the moment they are introduced in a sampling event. For example, if a direction is sampled using new variables u and v, then the second order derivative of this direction is computed for u and v but not for previous variables u_k. This approach reduces the amount of implementation (and computation during execution) considerably. On the other hand, this approach only accounts for local effects of the second order derivative at the moment the new variables are introduced. In practice it turns out that a combination with the other heuristics delivers the best results (see §7.6.4).

Algorithm 3 Combined heuristic for perturbation intervals

Compute interval for variable u_k:

1. nrs = intervals based on number of samples (local or global intervals)
2. grad = MIN(1.0, $\frac{\Delta f_{max}}{\frac{\partial f}{\partial u_k}/f}$) // gradient heuristic interval
3. second = MIN(1.0, $\varepsilon_{\max} \frac{\frac{\partial g(u_k)}{\partial u_k}}{0.5\frac{\partial^2 g(u_k)}{\partial u_k^2}}$) // second order derivative interval
4. interval = nrs $\times$ MIN(grad,second)

Whether the full integration of second order derivatives is worth the additional computational effort is left as a direction for future research. In any case, the path gradient will remain useful because it includes factors in the path evaluation that are not taken into account by the sampling pdfs. Therefore, the path gradient is able to prevent excessive blurring when the pdf does not adequately sample the integrand. This is not possible using solely the second order derivative of the path generating function.

7.6.4 Combined heuristic

In practice, we use a combination of the three heuristics presented in the previous sections to determine the perturbation intervals. The combined heuristic should provide perturbation intervals that always give a good noise versus bias trade-off. It is based on the following observations:

- The combined heuristic should definitely depend on the number of samples. This ensures consistent estimators, because the footprint becomes smaller with an increasing number of samples.
- The heuristics based on the path gradient and the second order derivatives can give arbitrarily large perturbation intervals. This is the case when the gradient or second order derivative is very small. Such large footprints are undesirable as they will introduce a large bias.

We start with the intervals based on the number of samples and then check the other heuristics and scale down the intervals by the heuristic that would deliver the smallest interval. The combined heuristic is summarized in algorithm 3. We use the same maximum relative error for both the gradient and the second order derivative heuristic. In the applications described next, we will show the influence of the different heuristics and show that the combination usually gives good perturbation intervals. Still, we think that further research might deliver even better heuristics.

7.7 Application: Texture filtering

The first application that demonstrates the use of path differentials is texture filtering. In this application the path footprint will be used to filter textures locally over a surface. This reduces noise in the image due

to variations in the texture.

7.7.1 Problem statement

Textures are widely used in rendering to increase the visual complexity of surfaces without introducing additional geometrical detail. Textures modulate BSDF parameters, usually the diffuse reflectance, to vary the material properties over a surface.

The use of textures may introduce high frequency components in the radiance incident to the image plane. For example textures may be viewed under a high minification so that a large part of the texture falls within a single pixel. A typical example in graphics is a checkerboard texture on a large plane. The checkerboard tiles become smaller when the distance from the camera increases and cause a high variation of the incoming radiance.

Any form of ray tracing point-samples the incident radiance. For Monte Carlo integration the variance of the estimate is proportional to the variance of the integrand. Hence a lot more samples are needed to reduce the variance caused by the variation in the textures. Brute force supersampling works, but it is expensive to trace all the additional paths.

An interesting approach to this problem is to reduce the variance in the textures locally on the surface by filtering over an appropriate footprint. The main problem with local filtering is the area over which to filter: an area too large will blur the image, but an area too small will not reduce the variance adequately.

7.7.2 Local filtering

Local filtering in Monte Carlo ray tracing can be motivated mathematically by investigating the pixel flux estimator.

The flux in a pixel I is estimated by tracing a number of paths and averaging their contribution:

$$\langle I \rangle = \sum_{i=1}^{N} \frac{f(\bar{\mathbf{x}}(\mathbf{u}_i))}{p(\bar{\mathbf{x}}(\mathbf{u}_i))} \,. \tag{7.9}$$

Each $\mathbf{u}_i$ represents a sample in the path domain.

When few paths are traced, only a few samples are explored in the path domain. For such low sampling cases, it would be better to use an average path evaluation over a neighborhood $\Delta\mathbf{u}$ around a sample $\mathbf{u}$. This average evaluation is given by:

$$f_{\Delta\mathbf{u}}(\mathbf{u}) = \frac{\int_{\Delta\mathbf{u}} f(\mathbf{u})\,\mathrm{d}\mathbf{u}}{\|\Delta\mathbf{u}\|} \,.$$

The filtered estimator is formed by just replacing f with $f_{\Delta\mathbf{u}}$ in (7.9). The filtered estimator will have a lower variance because a part of the integral is precomputed in the average path evaluation. However, because of the integral in the average evaluation, its exact computation is as difficult as the original integration problem. Therefore, an approximate computation of $f_{\Delta\mathbf{u}}(\mathbf{u})$ is needed.

Let **x** be a vertex in the path that resides on a (diffuse) textured surface. The neighborhood $\Delta\mathbf{u}$ in the domain projects to the footprint around **x**. If we consider the path evaluation to be constant over the neighborhood, except for the variation in the texture in **x**, an approximate average evaluation can be computed by filtering the textures locally over the footprint. Such a local filtering can be performed efficiently in texture space (see §7.7.3).

Thus, in order to perform local filtering a good footprint estimate is needed. Several footprint estimates have been proposed specifically for texture filtering:

- Early approaches used the projection of the pixel onto the first visible surface as the footprint. This works very well for directly visible surfaces, but is difficult to extend to reflected and refracted rays. Commonly, the total length of a path is then used to determine the level of detail needed in the texture map. While this works reasonably for reflection off planar surfaces, curved surfaces can diverge or converge the local light wavefront and cause arbitrary minification or magnification of the texture.

- Other approaches have tried to trace extended rays through the scene. These methods were already discussed in §7.2.

- Igehy uses ray differentials to estimate the footprint in classical ray tracing [44]. The ray differentials provide the easiest and most robust method for local filtering when the scene is limited to perfectly specular materials.

- We use path differentials to estimate the footprint for arbitrarily Monte Carlo sampled paths. This application of path differentials directly extends the local filtering presented by Igehy to glossy reflection and refraction.

7.7.3 Texture filtering techniques

Before the results obtained with local filtering are discussed, we will overview a few techniques for efficient texture filtering.

The footprint in a vertex is represented by two vectors ΔA, ΔB. These vectors are transformed into texture space by the texture projection that defines the exact mapping of the texture on the surface. The transformed vectors define a parallelogram $\mathcal{F}_{\text{tex}}$ over which the texture is to be filtered.

If $\mathcal{F}_{\text{tex}}$ is very small, smaller than one texel (one pixel in the texture map), bilinear interpolation between the nearest texels is used to get the texture value. For such small footprints, no filtering is necessary.

Larger footprints covering many texels need an efficient and accurate texture filter. A *box filter* assigns an equal weight to any texel covered by the footprint. A *Gaussian filter* imposes an elliptical Gaussian weight function over the footprint. This can give better results because the center of the footprint —the location of the vertex **x**— will have a higher weight in the averaged evaluation.

A simple method to filter over $\mathcal{F}_{\text{tex}}$ is *direct convolution* [38]. All texels under the footprint are weighted and added to the average texture value. While this is a very accurate filtering technique, it is also very slow. Each texel in the footprint has to be processed individually.

To speed up the direct convolution, a hierarchical representation of a texture can be used [129]. Mip-mapping is a well known representation for square textures. The base texture is repeatedly downsampled by a factor of 2. Four texels are averaged to form a new, lower-level texel, until the coarsest representation only contains one or a few texels. Each level in the mip-map represents a filtered version of the base texture.

There are several variations possible for filtering a texture over a parallelogram using a mip-map:

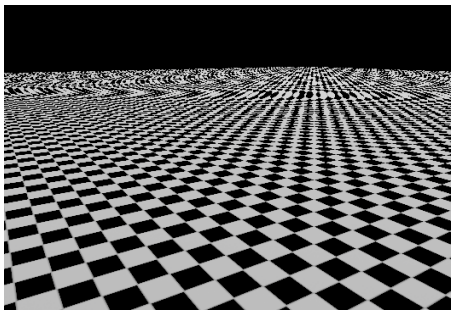
- A simple method uses the largest axis of the parallelogram to determine the required level of detail in the mip-map [37]. Using trilinear interpolation (interpolation between the 4 nearest texels in both a slightly coarser and finer level in the mip-map), a single filtered value is obtained efficiently. This method does not take into account the anisotropy of the parallelogram: a single square, axis aligned box filter is used. Because the largest axis of the parallelogram is used for the level of detail, aliasing disappears but the texture is blurred too much in the direction where the parallelogram is thin.

- A better, but also more expensive method performs anisotropic mip-map filtering. The smallest axis of the footprint is used to determine the level in the mip-map. Several samples on this level are averaged along the larger axis [89]. The number of samples is determined by the ratio of the larger and smaller axis (which can be large for elongated footprints). This method results in an approximate box filter over the footprint.

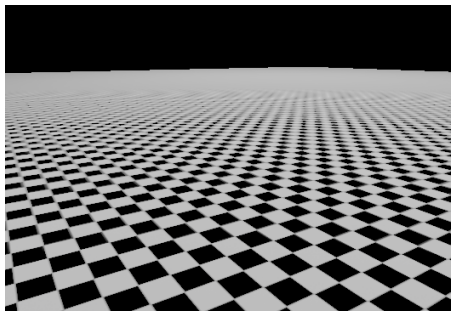
- A Gaussian filter can be approximated by a variation of the previous method, called *elliptic weighted average filtering with a mip-map* [33]. In this method several Gaussian weighted samples are also taken along the smaller axis. A finer level in the mip-map is determined by dividing the smaller axis by a certain number of samples for this axis. For each sample along the smaller axis, several Gaussian weighted samples are taken along the larger axis. This method is more expensive than the previous box filter, but for large footprints it is still much faster than direct convolution.

We have tried all these filtering techniques (direct convolution: Gaussian and filtered, simple mip-map, anisotropic mip-map, elliptic weighted average). A simple comparison is shown in figure 7.7, which shows the typical checkered floor that is filtered with a pixel footprint. The footprint was in fact computed with path differentials, but for this simple case it is almost equal to the exact projection of the pixel on the plane.

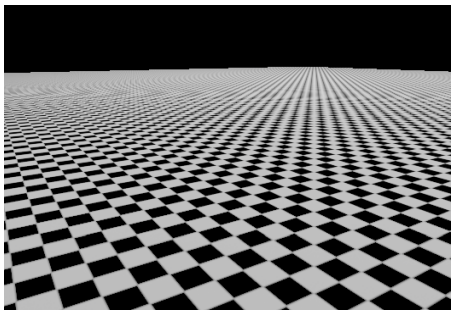
The Gaussian direct convolution filter gives the best result, but it is also the most expensive filter. The mip-mapped box and Gaussian filters are very similar in quality (we only used 3 Gaussian samples along



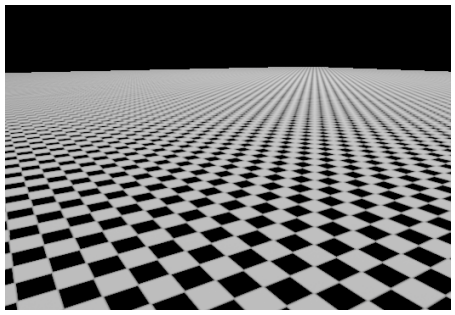
No filtering (9 sec.)



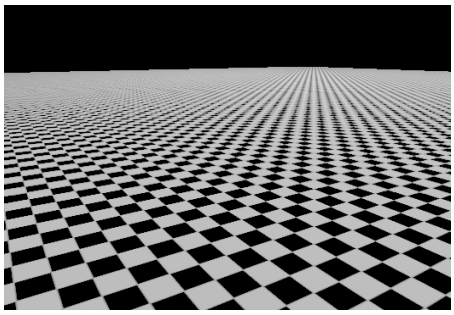
Isotropic mip−map (11.5 sec.)



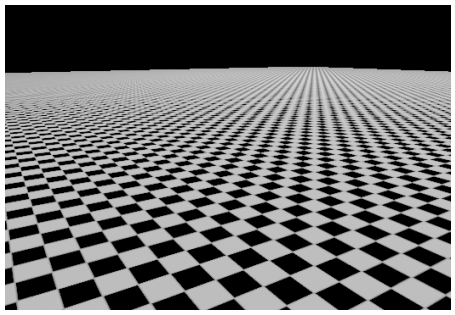
Direct convolution, box filter (35 sec.)



Direct convolution, gauss filter (36 sec.)



Anisotropic mip−map, box filter (12 sec.)



Anisotropic mip−map, gauss filter (16 sec.)

Figure 7.7: Texture filtering: These images show some different filtering techniques. Each image was generated with a single sample per pixel and used the (pixel) footprint to filter textures.

the smaller axis, compared to 1 for the box filter), but the Gaussian filter is slightly more expensive. Simple mip-mapping clearly overblurs near the horizon, but is still much better than the horrible unfiltered image.

For all the following examples we have used the mip-mapped box filter, because it delivers good quality for a reasonable amount of computation time.

7.7.4 Results

To test path differentials for texture filtering, a classical ray tracer was extended with glossy materials.

The example scene (see figure 7.8) consists of a room with a diffuse textured floor and several diffuse textured walls. Checkerboard textures were used to clearly demonstrate the effects of the filtering. The back wall is not textured and highly glossy (a modified Phong BRDF with an exponent equal to 1000), as is the left sphere (exponent 150). The right sphere has a glossy transparent BTDF (Phong-like refraction, exponent 1000). A single light source illuminates the scene from above.

This scene is ray traced using several samples per pixel. Glossy scattering uses a pdf proportional to the BSDF. No diffuse scattering is performed.

Figure 7.8 shows a comparison between an unfiltered, a filtered, and a reference image. The filtered image used path differentials to estimate the footprint and used the combined heuristic for the perturbation intervals (with global intervals for the number of samples). For both the filtered and unfiltered image, 4 samples per pixel were used. Exactly the same paths were used for both images. The reference image was rendered with 1024 samples per pixel.

As expected, the filtered image shows less noise where textures are visible. This can be seen on textured surfaces that are directly visible (e.g., the left wall), reflected once (back wall, part of the reflective sphere), but also on textured surfaces reached after several scatterings (glass sphere, glass sphere reflected in back wall). The two cut-outs zoom in on some of the interesting parts of the images:

- Left cut-out: The back wall reflects both the side wall and the opposite wall in the room. The side wall is viewed through the reflection under a slant angle, which causes the derivatives of the reflected direction to be projected to an elongated footprint. The opposite wall is further away but perpendicular and is blurred less by the glossy reflection.

- Right cut-out: The glossy glass sphere causes different levels in magnification of the textures. Mainly due to the path gradient, the footprint size adapts nicely to the magnification delivering an even noise versus bias trade-off over the whole sphere.

In figure 7.9 a comparison is made between the different heuristics for choosing perturbation intervals:

- Top image: Using only the number of samples to determine the perturbation intervals overblurs textures for longer paths. This is especially visible in the glass sphere. The reflections in the back

wall also show too much blurring, because those paths already depend on 4 variables (2 for pixel sampling, 2 for the reflection).

- Middle image: This image combines the path gradient with the number of samples. The path gradient successfully reduces the footprint for longer paths that magnify the textures. This is especially noticeable in the glass sphere and its reflection.

- Bottom image: This image combines the heuristic based on the (limited) second order derivatives with the one based on the number of samples. While it reduces blurring for glossy reflections (back wall), it is unable to adequately limit the blurring in the multiple refractions of the glass sphere. Since the second order derivative is only computed at the moment new variables are introduced, the global focusing effect of the refractions is not handled well. A full integration of the second order derivative might give better results that match the (cheaper) path gradient heuristic.

The combined heuristic (figure 7.8, top) preserves the advantages of all heuristics: consistency (convergence to the correct solution), noise reduction, but with a limited bias in reflections and refractions.

7.8 Application: Particle tracing

In the second application, path differentials are used for particle tracing. Particle tracing constructs paths starting from the light sources. This Monte Carlo simulation of light transport is used in many global illumination algorithms (see §3.4.2 and §4.3.1.2).

We present a hierarchical radiosity application to demonstrate the usefulness of path differentials for particle tracing, but it can be used as well for the other algorithms.

We will briefly review particle tracing radiosity in §7.8.1. Existing techniques for hierarchical refinement are given in §7.8.2, which are theoretically compared with our new refinement oracle in §7.8.3. Results are discussed in §7.8.4.

7.8.1 Particle tracing radiosity

Particle tracing radiosity is a form of Monte Carlo radiosity, that computes a radiosity solution on the patches in a discretized scene, but it prevents discretization of the light transport equations themselves. Continuous random walks —light paths— are constructed from the light sources. Contributions are accumulated on patches hit by the light paths, resulting in a discretized version of the radiosity function on the surfaces. Since continuous paths are traced, the integral equation that governs the light transport is solved correctly (at least asymptotically) and is not discretized into a set of linear equations. The advantage is that glossy and specular materials are easily incorporated into particle tracing radiosity by just using standard

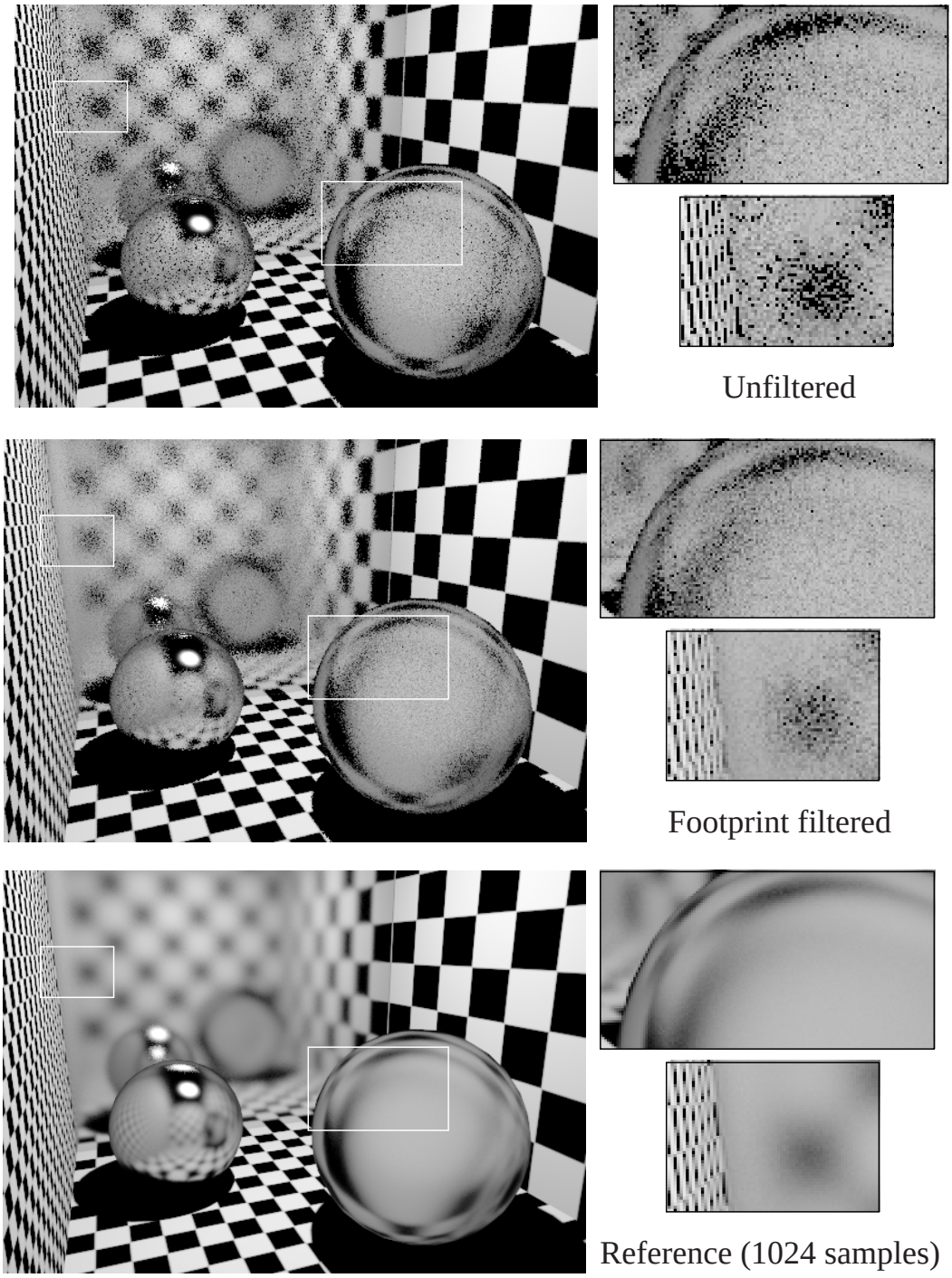


Figure 7.8: Texture filtering: in the top image (4 samples/pixel), no texture filtering was applied. In the middle image (4 samples/pixel), textures were filtered over the path footprint estimated using path differentials. Clearly, the noise is reduced by the filtering, while the different levels of magnification or minification of the textures are closely followed by the estimated footprint. A reference image is shown at the bottom.

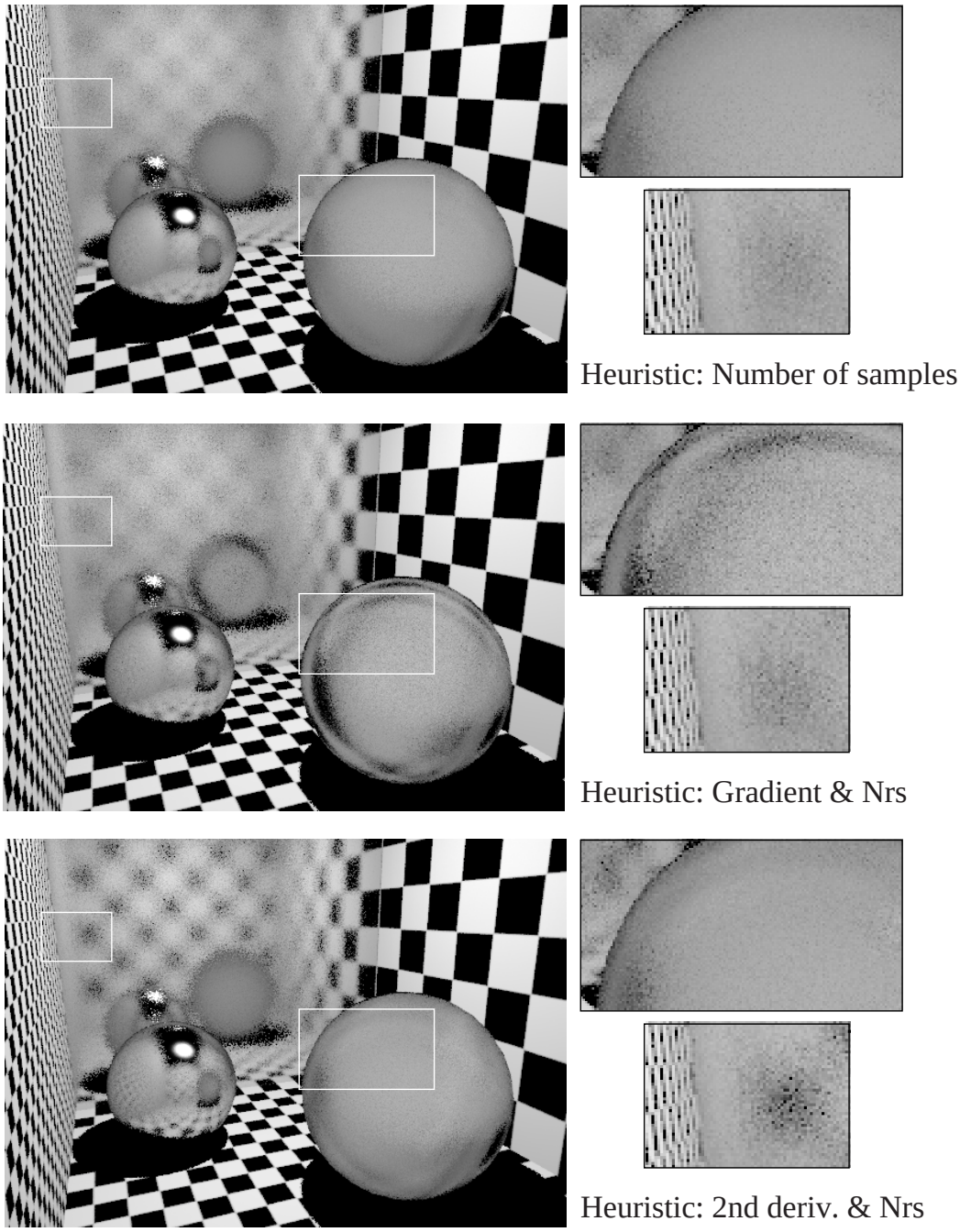


Figure 7.9: Comparison between different perturbation interval heuristics (4 samples/pixel). Top: Using only the number of samples overblurs the textures. Middle: Adding the gradient heuristic improves the footprint for paths scattered multiple times (the glass sphere). Bottom: Using the second order derivative mainly shows improvement for the (single) glossy reflections (on the back wall). The combined heuristic (Figure 7.8, top) preserves the benefits of all these different heuristics.

light tracing. This is much harder for discretized radiosity methods, since an angular representation of the radiance function is needed on non-diffuse surfaces (see [21]).

Extensive information about particle tracing radiosity can be found in the literature [79, 78, 41, 5]. We use:

- A collision estimator: each vertex in a light path contributes to the patch that was hit.
- Constant basis functions: the radiosity solution is assumed to be constant over the patch.

Formally, given a set of particles $\Phi_i(\mathbf{x}_i, \omega_i, \phi_i)$ from particle tracing (see §3.4.2), the radiosity on a patch j, for a constant basis function, is estimated as:

$$B_j = \frac{\int_{A_j} B(\mathbf{x})\,\mathrm{d}\mathbf{x}}{A_j} = \frac{\sum_{i=1}^{N_\Phi} v(\Phi_i, j)\,\phi_i}{A_j},$$

where j denotes a patch in the scene and $v(\Phi_i, j) = 1$ when particle Φ_i resides on patch j and zero otherwise.

In practice, the contributions are added to B_j on the fly during particle tracing. The patch j hit by a particle is directly known from the ray-patch intersections.

The assumption that the radiosity is constant over the patches in the scene, requires a pre-meshing of the scene. Since the assumption usually does not hold for large patches, the scene geometry must be subdivided in a large number of small elements in a preprocessing step. The smaller the elements, however, the larger the number of particles that are needed to reduce the variance of the radiosity solution.

7.8.2 Hierarchical subdivision

Hierarchical subdivision provides a solution to the meshing problem for radiosity methods. Instead of pre-meshing the scene, the patches are subdivided appropriately on the fly.

Several methods have been proposed for hierarchical subdivision in particle tracing radiosity:

- Heckbert [41] uses an adaptive radiosity texture for each surface in the scene. Initially the textures are subdivided into a user-defined number of elements. Then, light paths are traced and the hits are accumulated on the elements. Several iterations are performed, and, after each iteration, the elements that received a certain minimum number of hits are recursively split into 4 sub-elements. Since only the number of hits was recorded in the parent element, nothing is known about the difference in radiosity of the sub-elements. Therefore, these elements are cleared and the rays have to be re-shot in the next iteration, in order to find the correct distribution of hits over the sub-elements.
- Tobler et al. [111] propose an extension of Heckbert's adaptive radiosity textures. The particle hits are accumulated on two different levels in the hierarchy. The elements in the finest level, which is called the preview level, indicate whether the constant radiosity of the coarser level is adequate to represent the real radiosity function. If the difference between the preview elements exceeds a

certain threshold, the preview level becomes the main level and a new preview level is constructed by subdivision. Also in this scheme, hits are discarded when subdividing: Several hits may have been recorded in the coarse, abandoned level that were not recorded in the preview level. The hits cannot be distributed over the children because, again, only the number of hits is recorded.

Pope and Chalmers [82] improved on this approach by using a more sophisticated threshold to trigger subdivision. Still subdivision leads to the discarding of particles.

- Density estimation prevents discarding of particles by storing all particle hit points [93]. After tracing a large number of particles and storing the hit points in a file, density estimation is used to reconstruct the radiosity on a very dense mesh. The mesh is then decimated for display purposes. While no hits are discarded, the storage overhead for storing all particles is large. The method is not progressive, because the expensive reconstruction is performed on a fixed set of particles.

All these subdivision schemes are not really oracles: they do not predict the appropriate level of subdivision for a certain particle beforehand. The disadvantages of such a-posteriori criteria are that at some point particles have to be discarded, unless all particles are stored.

7.8.3 A subdivision oracle based on path differentials

Path differentials allow for a simple, yet effective, subdivision oracle for particle tracing radiosity:

- Particle tracing consists of tracing light paths through the scene. Each vertex in a light path forms a single particle.
- For each light path, path differentials are computed. The footprint estimate in each vertex indicates a region around the vertex where the contribution of the path can be considered more or less constant.
- The area of the footprint can thus be used as an indication for the size of the element to which the particle should contribute.

The subdivision oracle simply descends the element hierarchy until an element is found that best matches the area of the particle footprint. Elements are subdivided as necessary while descending the hierarchy.

To display the radiosity solution, the radiosity accumulated in higher levels of the hierarchy is pushed down towards the leaves. This is a standard operation in (discrete) hierarchical radiosity [22].

The path differential oracle has several interesting properties:

- The oracle easily accommodates clustering. Clustering groups patches together into a cluster hierarchy. When clusters are far apart, there is no need to compute transport between the individual small patches that form the clusters. Computing transport between the clusters themselves can highly speed up radiosity computations. It is frequently used in discretized radiosity algorithms [101, 96].

Our oracle can use clustering if the footprint area is larger than the individual patch that was hit by the particle. A contribution will be made to the cluster that contains the patch and that has a cross section area closest to the footprint area.

- While the radiosity solution in previous subdivision approaches for particle tracing may be hierarchical, the transport itself is not. Each particle contributes only to the finest subdivision level (or the two finest levels). For small elements, this causes a high variance in the solution.

  Path footprints, on the other hand, are computed for individual paths, so that each particle can contribute to a different level in the hierarchy.

- No storage is needed, except for the radiosity solution.

- No particles are discarded when subdividing, because the oracle predicts an appropriate level a-priori for each particle.

The path differential oracle is, to our knowledge, the first oracle for particle tracing that can determine an appropriate subdivision level for a single path.

7.8.4 Results

A hierarchical particle tracing radiosity implementation based on path differentials was added to RENDERPARK. The implementation uses constant basis functions and has support for clustering. The test scene (see figure 7.10) contains several diffuse objects and a glass sphere on the table. The glass was modeled using a Phong-like refraction model. Note that the glass sphere shows up black, because only the diffuse component of the radiance distribution in the scene is stored.

Figure 7.10 shows a comparison between a non-hierarchical pre-meshed method and the hierarchical method based on path differentials:

- The pre-meshing subdivides all patches into elements of a certain size. Both a coarse and a fine pre-meshed scene were used. While the coarse subdivision shows relatively little noise, the area around the caustic needs a finer subdivision. The fine subdivision is adequate in the caustic area, but leads to a noisy solution in dimmer, indirectly lit areas, such as the floor beneath the table.

- The hierarchical subdivision with path differentials shows virtually no noise, but still the caustic area is nicely subdivided. Even compared to the coarse fixed subdivision, the region under the table shows less noise.

The two top images in figure 7.11 show a comparison between two different interval heuristics. The top image only uses the number of samples, which, just as in the texture filtering application, leads to a bad subdivision for the caustic.

The bottom image in figure 7.11 was generated with a hierarchical version of stochastic Jacobi radiosity [8, 5]. This is a discrete radiosity method, that only computes diffuse transport. About the same number of rays was used for this method as for the path differential images. It is interesting to see that, apart from the caustic (that is not computed at all by the discrete method), the subdivision is very similar. This comparison shows that path differentials provide a refinement oracle for continuous radiosity methods that can match the oracles used in discrete Monte Carlo radiosity methods.

7.9 Conclusion

In this chapter we presented path differentials, a general tool for global illumination algorithms. Path differentials compute partial derivatives of vertices and directions in a path for all the path variables. Using path differentials, the region of influence or footprint of a vertex can be estimated by a first order Taylor approximation. Such a footprint estimate can be useful in many applications, as was demonstrated in two applications: local texture filtering for glossy BSDFs and a subdivision oracle based on single paths for hierarchical particle tracing radiosity.

Path differentials are easy to compute: the theory relies on standard calculus, requiring simple differentiation of existing sampling procedures. Although derivatives for all sampling procedures have to be implemented, it is not too difficult to add it to an existing rendering system. Path differentials scale well with scene complexity, because they are computed from a single, standard, infinitely thin path. Intersection calculations do not need to change.

While path differentials already resulted in a significant variance reduction in the applications, there is still a lot of room for refinements, extensions and other applications. Some interesting directions are outlined next.

Refinement of perturbation intervals The choice of the perturbation intervals, that determine a small neighborhood around a sampling point in the path domain, is vital for a good footprint estimate. These intervals have to ensure coherence of the path contribution over the footprint. Several interval heuristics were explored and successfully combined, but refining these heuristics is definitely an important area for future research:

- Second order derivatives may be very helpful for error bounding and interval estimation. While we have performed some experiments with the second order derivative, a full second order Taylor approximation of the footprint may lead to improved estimates and better results.

  However, adding second order derivatives throughout the whole rendering system and for all sampling events involves a significant amount of implementation. An interesting but open question is whether

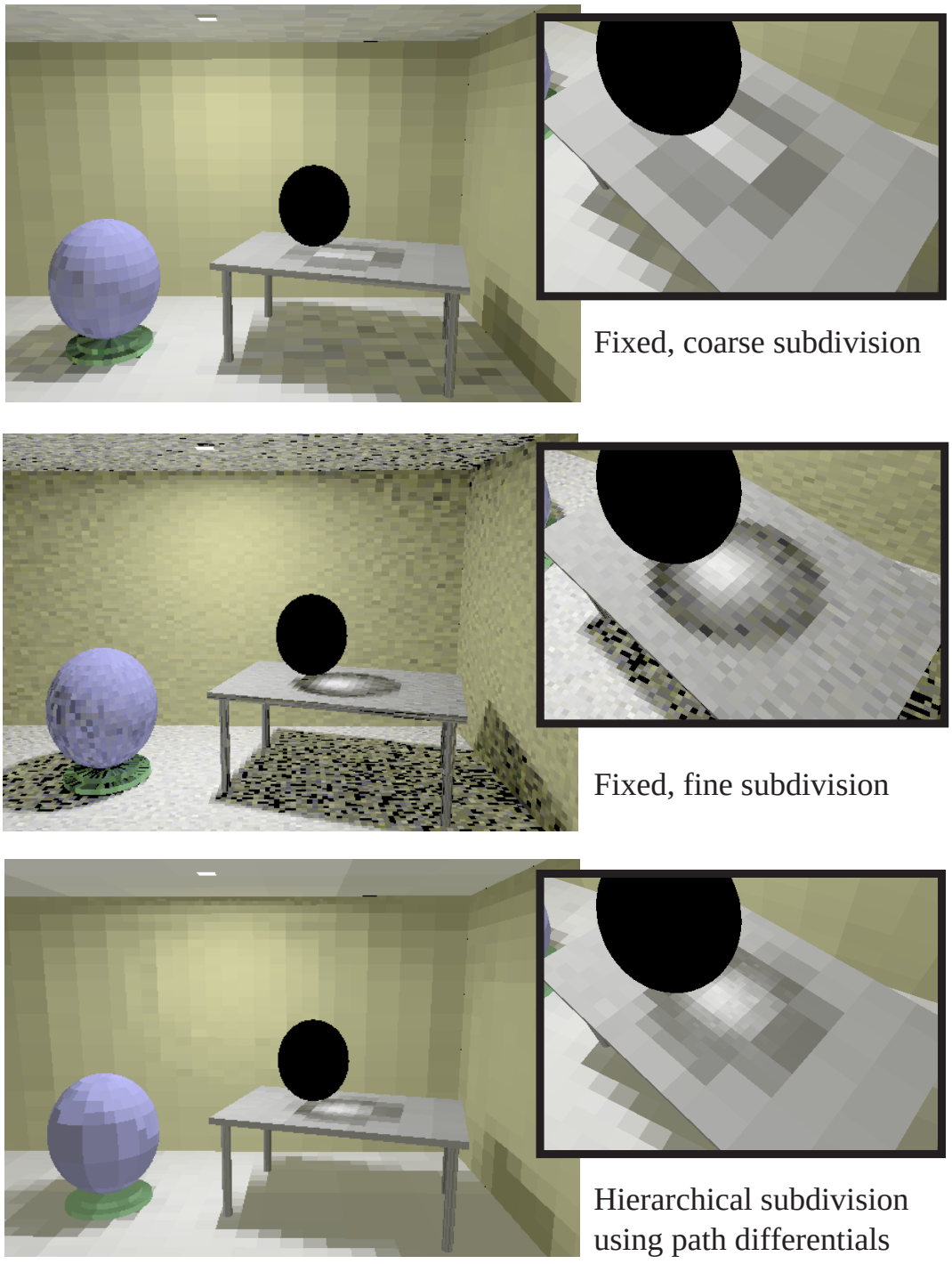


Figure 7.10: Particle tracing radiosity (400k paths): The two top images use a non-hierarchical method with a pre-meshed scene. A coarse subdivision reduces noise, but cannot reproduce the caustic accurately. A fine subdivision needs a lot more particles to reduce the noise. The bottom image uses the refinement oracle based on path differentials. The subdivision level, which is determined for individual paths, ensures a virtually noiseless solution, while still the caustic is reproduced quite accurately.

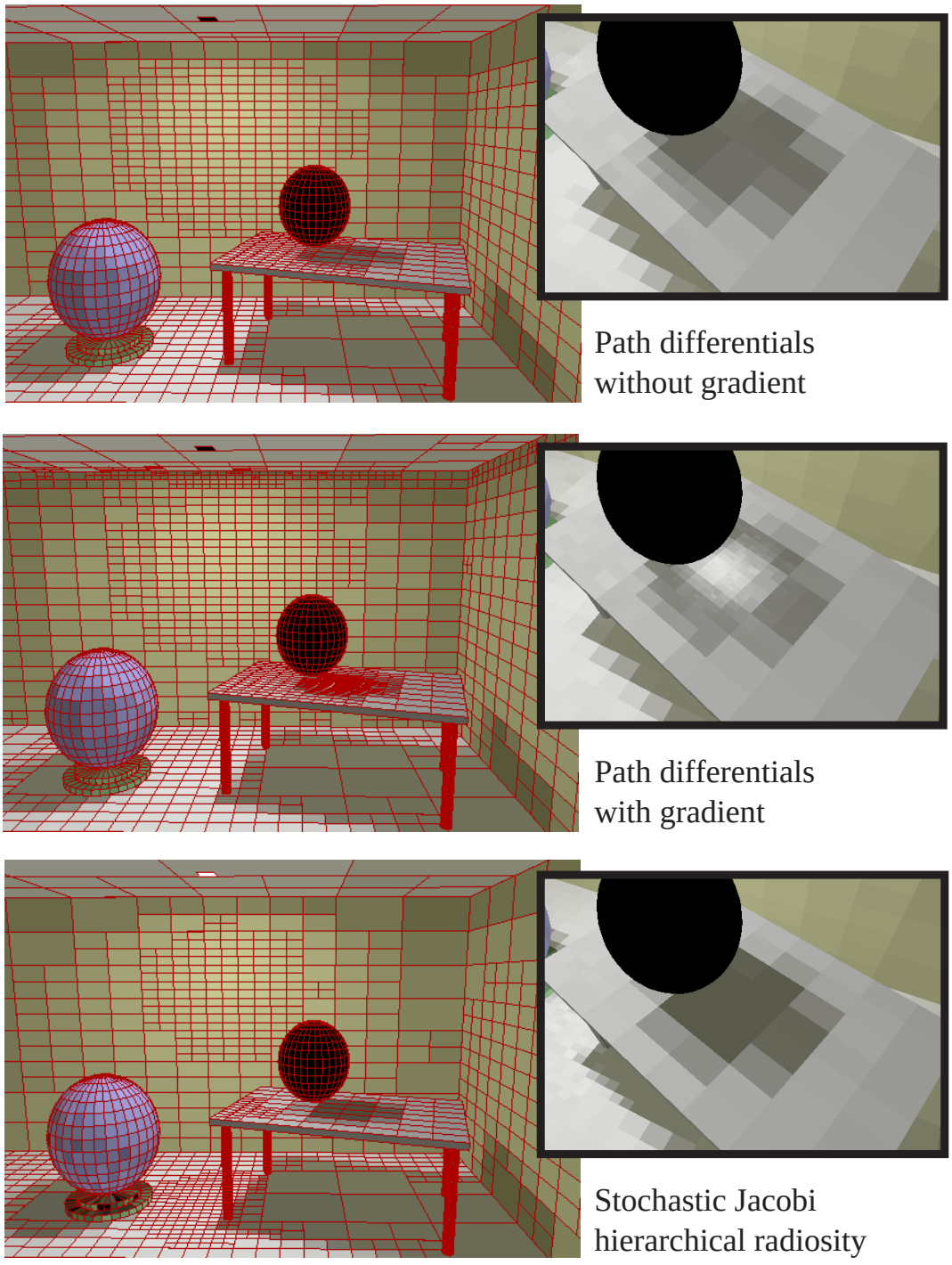


Figure 7.11: Hierarchical radiosity (400k paths): The two top images are computed with path differentials. The top image did not use the path gradient, which leads to a bad subdivision in the caustic area. The bottom image was computed with hierarchical stochastic Jacobi radiosity, which is a discrete Monte Carlo radiosity algorithm. The overall subdivision is very similar to the one of path differentials, with the exception of the caustic (the method only handles diffuse transport).

the computational overhead of the second order derivatives is justified by the improvement of the footprint estimate.

- Currently, the footprint estimate does not take visibility into account. While allows an easy computation of the footprint, it might be interesting to perform some selective visibility tests and reduce the footprint when a change in visibility is detected. For example, a small number of rays could be allocated to test visibility on the boundary of the footprint or the differential vectors.

Non-constant footprints Currently, our applications consider the path contribution to be constant over the footprint. The path gradient, that indicates the change of the path evaluation over the footprint, is only used to limit the footprint size. An interesting extension would be to use the path gradient and possibly higher order derivatives of the path evaluation to explicitly approximate the path evaluation over the footprint. Specialized texture filters could be designed that match the evaluation over the footprint. For the radiosity application, higher order basis functions could be fitted to the evaluation over the footprint resulting in a better radiosity solution.

Reducing the number of differential vectors The number of partial derivatives (and the number of corresponding differential vectors) increases with the number of variables in a path. For long paths, this number grows large and many derivatives must be tracked when tracing a ray or scattering a direction. This results in non-negligible increase in computation time (up to 70% for tracing a single path in a simple scene, using an unoptimized implementation).

Reducing the number of differential vectors before transfer or scattering could limit the total number of differential vectors. Suppose, for example, that an incoming direction depends on 4 variables. Combining the four differential vectors into two new vectors before scattering, limits the number of differential vectors of the outgoing direction again to 4 (2 combined and 2 new variables).

How to combine the differential vectors before scattering is an open problem. The derivatives of the new directions can depend on the derivatives of the previous directions in complex ways. It is not obvious how a reduction such as the one used to simplify the footprint to a parallelogram, would influence the further vector derivatives.

Other applications Many other global illumination algorithms may benefit from path differentials:

- Other algorithms based on particle tracing could use a method similar to the subdivision oracle. For example, in density estimation or photon mapping the footprint could be used to determine a splatting size or a maximum search radius for a particle.

- An interesting application would be to combine path differentials with bidirectional path tracing. While computing path differentials for the separate eye and light paths is straightforward, it is not obvious how to handle the connection between the sub-paths with respect to the path footprints.

- We have used path differentials for importance computations in photon mapping. This application will be discussed in detail in the next chapter.

- The partial derivatives can also be used to efficiently compute perturbations of a path. Instead of constructing a footprint, these perturbations could be used directly to construct new paths. While these path perturbations cannot be used directly in a standard Monte Carlo estimator —the probability distribution for sampled paths is different from that of the path perturbations—, it could be used to compute mutations of paths in Metropolis light transport. This application was already mentioned by Chen [15] but her framework currently only includes specular paths. Our framework allows arbitrary BSDFs and provides a step forward towards this application.

- Time could also be introduced as a variable in a path for animation applications. Partial derivatives with respect to time could enable the development of specific filters to efficiently compute motion blur.

These are just some examples of applications, and we are convinced that many other applications will benefit from path derivatives and path differentials.

Appendix 7.A Derivative computation details

This appendix contains a detailed description of the computation of partial derivatives for several sampling events that occur in the Monte Carlo sampling of paths. The partial derivatives for vertices, directions and the path evaluation will be given for pixel sampling, ray transfer, BSDF sampling and light source sampling. The information in this appendix is mainly targeted at people who want to implement path differentials themselves.

The derivative procedures for pixel sampling and ray transfer are similar to those presented by Igehy [44], because these events are the same for classical and Monte Carlo ray tracing. They are repeated here for completeness.

We want to stress again that the derivatives are straightforward to deduce from the sampling procedures. Any other importance sampling procedure is easily derived in the same manner as the examples given here.

In these derivative examples, we will use the following notation:

- $\mathbf{x}$, ω: the previous vertex and direction

- $\mathbf{x}'$, ω': a newly sampled vertex or direction

- u_k: A variable introduced in a previous sampling event
- u, v: Variables introduced for the sampling event under consideration (all events use 2D sampling)
- **N**: the shading normal
- $\mathbf{N}_g$: the geometric normal
- ω_R: the perfectly specular reflection vector (given an incoming direction ω in a vertex **x**).

7.A.1 Pixel sampling

Pixel sampling constructs a direction ω' by sampling a uniform point on a certain pixel. Together with the eye vertex $\mathbf{x}_{\text{eye}}$, the direction forms the starting ray of an eye path.

Direction derivative Given 2D image coordinates (α, β) and a camera coordinate system (**View**, **Right**, **Up**), a non-normalized viewing direction is given by:

$$\mathbf{d}(\alpha, \beta) = \mathbf{View} + \alpha\mathbf{Right} + \beta\mathbf{Up}\,.$$

Uniform sampling of a pixel using (random) unit variables u, v supplies the image coordinates:

$$\alpha = u\,\text{pix}_w + \text{pix}_l\,,$$
$$\beta = v\,\text{pix}_h + \text{pix}_t\,,$$

with pix_w, pix_h the width and height of the pixel, and pix_l, pix_t the left and top coordinate of the pixel under consideration.

The normalized sampled direction is:

$$\omega' = \frac{\mathbf{d}}{(\mathbf{d}\cdot\mathbf{d})^{1/2}}\,. \tag{7.10}$$

The derivative for u is:

$$\frac{\partial\omega'}{\partial u} = \frac{\partial\omega'}{\partial\alpha}\frac{\partial\alpha}{\partial u} = \frac{\partial\omega'}{\partial\alpha}\,\text{pix}_w\,,$$

where $\frac{\partial\omega'}{\partial\alpha}$ is found by differentiating (7.10) with respect to α:

$$\frac{\partial\omega'}{\partial\alpha} = \frac{(\mathbf{d}\cdot\mathbf{d})\mathbf{Right} - (\mathbf{d}\cdot\mathbf{Right})\mathbf{d}}{(\mathbf{d}\cdot\mathbf{d})^{3/2}}\,.$$

For $\frac{\partial\omega'}{\partial v}$ a similar equation can be derived.

Evaluation derivative Pixel sampling introduces a factor $W_e(\mathbf{x}_{\text{eye}} \to \omega')$ into the path evaluation. This factor depends on the camera model and the reconstruction filter of a pixel. In our implementation we use

a box filter and a simple pinhole camera model. For this case, $W_e(\mathbf{x}_{eye} \to \omega')$ is constant across the image plane. Therefore,

$$\frac{\partial W_e(\mathbf{x}_{eye} \to \omega')}{\partial u, v} = 0\,.$$

The start of an eye path initializes the sum to zero and the terms of the relative partial derivatives of the path evaluation are summed during the tracing of the paths (§7.6.2.2).

7.A.2 Ray transfer

Ray transfer computes a new path vertex $\mathbf{x}'$ by tracing a ray from a previous vertex $\mathbf{x}$ in a direction ω. Transfer does not involve sampling new variables: tracing a ray is completely deterministic. Therefore, only partial derivatives with respect to previous variables u_k must be computed.

Vertex derivative The new vertex is given by:

$$\mathbf{x}' = \mathrm{rc}(\mathbf{x} \to \omega) = \mathbf{x} + t\omega\,,$$

with t the traveled distance to the new vertex.

The partial derivatives are given by:

$$\frac{\partial \mathbf{x}'}{\partial u_k} = \frac{\partial \mathbf{x}}{\partial u_k} + t\frac{\partial \omega}{\partial u_k} + \frac{\partial t}{\partial u_k}\omega,$$

with

$$\frac{\partial t}{\partial u_k} = -\frac{\left(\frac{\partial \mathbf{x}}{\partial u_k} + t\frac{\partial \omega}{\partial u_k}\right)\mathbf{N}_g}{\omega \cdot \mathbf{N}_g},$$

where $\mathbf{N}_g$ is the geometric normal of the surface in $\mathbf{x}'$. This result can be proved for an arbitrary surface [43].

The three terms in this derivative can be explained intuitively (by considering the derivatives in the other two terms to be zero):

- $\frac{\partial \mathbf{x}}{\partial u_k}$: A small displacement of the previous vertex results in an equal displacement of the new vertex (when all other terms are zero).
- $t\frac{\partial \omega}{\partial u_k}$: A small change in the direction of the ray is amplified by the traveled distance.
- $\frac{\partial t}{\partial u_k}\omega$: The first two terms indicate the positional offset of the vertex $\mathbf{x}'$ with respect to the position and direction of the ray, but it does not take into account the orientation of the surface that is hit. This is exactly what the third term does: the positional offset is projected onto the tangent plane of the surface in $\mathbf{x}'$.

  For example, when a surface is hit at a grazing angle ($\omega \cdot \mathbf{N}_g$ small), the derivative of t can grow large because of the small value of the nominator. If the positional offset has a direction similar to the normal, this effect is the largest (numerator goes towards 1).

As was shown in §7.5.2, the vertex derivatives are co-planar and lie in the tangent plane of the surface in $\mathbf{x}'$.

Evaluation derivative Tracing a ray does not change the evaluation of a path as specified in §7.6.2.1.

Alternatively, the evaluation could be expressed in terms of area measure. In that case the tracing of a ray corresponds to a transformation from spherical angle measure to area measure, which adds a factor $\frac{\omega\cdot\mathbf{N}}{t^2}$ to the evaluation.

The relative partial derivatives of this factor are:

$$\frac{\partial\frac{\omega\cdot\mathbf{N}}{t^2}}{\partial u_k}\Big/\frac{\omega\cdot\mathbf{N}}{t^2} = \frac{\frac{\partial\omega}{\partial u_k}\cdot\mathbf{N}+\omega\cdot\frac{\partial\mathbf{N}}{\partial u_k}}{\omega\cdot\mathbf{N}} + \frac{-2\frac{\partial t}{\partial u_k}}{t^3}\,.$$

As said before, we do not use this term in the path gradient, because the t^3 factor in the denominator causes high gradients for short distances, and resulted in a fine subdivision near corners in the radiosity application.

7.A.3 Direction sampling

Direction sampling generates a new direction ω' given an incident direction ω in a vertex $\mathbf{x}$. The partial derivatives depend on the sampling procedure which can be different for each BSDF model. We will derive derivative formulas for a diffuse reflection BRDF that uses a cosine distribution sampling, and for Phong lobe scattering.

In our rendering framework, a transformation to a local sampling coordinate frame is applied first. A new direction ω'_s is generated in the sampling frame and transformed back into world coordinates giving the direction ω'. Section 7.A.3.1 shows how the transformation is handled when computing derivatives. Diffuse reflection and Phong lobe sampling in the local frame are detailed in §7.A.3.2 and §7.A.3.3.

7.A.3.1 Coordinate transformation

Given three vectors $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ that form an orthonormal basis in the current frame, the matrix $\mathbf{T}_s = [\mathbf{XYZ}]^\top$ represents the rotation to the new coordinate frame.

The sampled direction in the new coordinate frame, ω'_s, is given by

$$\omega'_s = \mathbf{T}_s\omega'\,.$$

The derivatives for a variable u_k are

$$\frac{\partial\omega'_s}{\partial u_k} = \frac{\partial\mathbf{T}_s}{\partial u_k}\omega' + \mathbf{T}_s\frac{\partial\omega'}{\partial u_k}\,,$$

where

$$\frac{\partial\mathbf{T}_s}{\partial u_k} = \left[\frac{\partial\mathbf{X}}{\partial u_k}\frac{\partial\mathbf{Y}}{\partial u_k}\frac{\partial\mathbf{Z}}{\partial u_k}\right]^\top.$$

Given the derivatives of the local vector ω'_s, the derivatives of ω' are computed as

$$\frac{\partial\omega'}{\partial u_k} = \mathbf{T}_s^\top\left(\frac{\partial\omega'_s}{\partial u_k} - \frac{\partial\mathbf{T}_s}{\partial u_k}\omega'\right).$$

The derivatives of the basis vectors $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ depend on how the local frame is chosen, which in turn depends on the BRDF model. This transformation is also used to transform derivatives of the incoming direction, $\frac{\partial\omega}{\partial u_k}$, to $\frac{\partial\omega_s}{\partial u_k}$.

7.A.3.2 Diffuse reflection

Direction derivative A diffuse BRDF scatters light equally in all directions. The BRDF is a constant, so sampling can be done proportionally to the cosine with the surface normal ($\cos\theta = \mathbf{N}\omega'$). After a transformation $\mathbf{T}_s$ which turns the normal into the Z-axis, the sampling of ω'_s is given by

$$\begin{aligned} \cos\theta &= \sqrt{1-u}, \\ \varphi &= 2\pi v, \\ \omega'_s &= [\cos\varphi\sin\theta, \sin\varphi\sin\theta, \cos\theta]^\top. \end{aligned} \tag{7.11}$$

The derivatives for previous variables u_k in the local frame are zero, because the outgoing direction does not depend on the incident direction, so that $\frac{\partial\omega'}{\partial u_k} = \mathbf{T}_s^\top(-\frac{\partial\mathbf{T}_s}{\partial u_k}\omega')$. Thus, these derivatives depend on the change of the normal (i.e., the sampling frame) in terms of u_k.

The derivatives for the new variables u and v are easily derived from (7.11) by standard calculus.

Evaluation derivative Since a diffuse BRDF is constant, all the evaluation derivatives are zero. The cosine factor in the evaluation ($\mathbf{N}\cdot\omega'$) adds a term $(\frac{\partial\mathbf{N}}{\partial u_k}\cdot\omega' + \mathbf{N}\cdot\frac{\partial\omega'}{\partial u_k})/(\mathbf{N}\cdot\omega')$ to the path gradient for any BSDF.

Second order derivative The second order derivative is only computed for new variables u, v in the sampling frame. As explained in §7.6.3, it is currently used to estimate the perturbation intervals. The second order derivatives in the sampling frame are straightforward to derive from the first order derivatives.

7.A.3.3 Phong scattering

A modified Phong BRDF samples a new direction ω' around the perfectly specular reflected direction ω_R with a pdf proportional to a $\cos^S\alpha$ distribution (where α is the angle between ω' and ω_R).

Direction derivatives Given the incoming direction ω, ω_R is given by the well known expression

$$\omega_R = \omega - 2(\omega\mathbf{N})\mathbf{N}.$$

A transformation $\mathbf{T}_s$ to a sampling frame is applied that turns ω_R into the Z-axis. The derivatives of ω_R are given by

$$\frac{\partial\omega_R}{\partial u_k} = \frac{\partial\omega}{\partial u_k} - 2\left[(\omega\cdot\mathbf{N})\frac{\partial\mathbf{N}}{\partial u_k} + (\frac{\partial\omega}{\partial u_k}\cdot\mathbf{N} + \omega\cdot\frac{\partial\mathbf{N}}{\partial u_k})\mathbf{N}\right].$$

The derivative of the shading normal ($\frac{\partial\mathbf{N}}{\partial u_k}$) depends on the underlying geometry. As shown by Igehy [44], the derivative for a plane is zero, and for a sphere it is given by $\frac{\partial\mathbf{x}}{\partial u_k}/R$ with R the radius of the sphere. In our implementation, we use normal-interpolated triangles and quadrilaterals. The derivatives are similar to the sampling of a vertex on a triangle or quad, which are given in §7.A.4.1. As Igehy points out, the use of

standard calculus easily accommodates non-physical phenomena such as interpolated normals. This would be harder with differential geometry techniques.

In the local sampling frame, the sampled direction is given by Phong lobe sampling, a well known importance sampling procedure [64]:

$$\begin{aligned}\cos\theta &= u^{1/(S+1)},\\ \sin\theta &= \sqrt{1-\cos^2\theta},\\ \varphi &= 2\pi v.\end{aligned}$$

The direction ω_s' is now given by

$$\omega_s' = [\cos\varphi\sin\theta, \sin\varphi\sin\theta, \cos\theta]^\top.$$

To compute ω_s', we just need to derive these equations for u and v (only non zero derivatives are shown):

$$\begin{aligned}\frac{\partial\cos\theta}{\partial u} &= \frac{1}{S+1}u^{-s/(s+1)} = \frac{1}{(S+1)\cos^S\theta},\\ \frac{\partial\sin\theta}{\partial u} &= -\cos\theta * \frac{\partial\cos\theta}{\partial u}/\sin\theta,\\ \frac{\partial\varphi}{\partial v} &= 2\pi.\end{aligned} \tag{7.12}$$

The derivatives for ω_s' are easily computed from these equations.

One remark:

- Near the pole of the lobe, the factor $\frac{\partial\sin\theta}{\partial u}$ becomes large as $\sin\theta$ goes to zero. Limiting $\sin\theta$ to a certain small ε is sufficient to prevent this problem. The large partial derivatives that still appear around the pole, are countered by a large second order derivative that limits the perturbation interval.

Evaluation derivatives The BRDF evaluation is $C(\omega_R\cdot\omega')^S$, with C a material constant. Derivatives are easily computed.

Second order derivatives The second order derivative is computed by differentiation of the first order derivative. While this is standard calculus, the expressions become increasingly complex and are not reproduced here.

7.A.4 Light source sampling

This section details derivative computations for light source sampling.

Sampling a path vertex on a light source involves two steps, choosing a light source (§7.A.4.2) and sampling a point on that light source (§7.A.4.1).

7.A.4.1 Choosing a point on a light source

We will demonstrate the computation of derivatives for a regular quadrilateral and a triangle.

Quadrilateral Given a regular quadrilateral light source (i.e., a parallelogram) with vertices **A**, **B**, **C**, **D** and given random numbers u, v, a uniformly sampled vertex $\mathbf{x}'$ and its derivatives are given by

$$\mathbf{x}' = \mathbf{A} + u(\mathbf{B}-\mathbf{A}) + v(\mathbf{D}-\mathbf{A})$$

$$\Rightarrow \quad \frac{\partial \mathbf{x}'}{\partial u} = \mathbf{B}-\mathbf{A}, \qquad \frac{\partial \mathbf{x}'}{\partial v} = \mathbf{D}-\mathbf{A}.$$

For irregular quadrilaterals, the sampling procedure and its derivatives are more complex. First a mapping from uniform (u,v) to bilinear coordinates must be performed. The point is then sampled using the standard bilinear mapping: $\mathbf{x}' = u(\mathbf{B}-\mathbf{A}) + v(\mathbf{D}-\mathbf{A}) - uv(\mathbf{B}-\mathbf{C}+\mathbf{D}-\mathbf{A})$. See [5, p. 247] for more information.

Triangle A simple procedure for sampling a uniform point in a triangle $\mathbf{A}, \mathbf{B}, \mathbf{C}$ is given by [112]:

$$\text{if } (u+v>1.) \text{ then } u = 1-u \;;\; v = 1-v$$
$$\mathbf{x}' = \mathbf{A} + u(\mathbf{B}-\mathbf{A}) + v(\mathbf{C}-\mathbf{A}).$$

In fact a point is sampled in a parallelogram, and one half ($u+v>1$) is mirrored back into the triangle. This doubles the density of points compared to the density in the parallelogram. Dividing the derivatives by a factor $\sqrt{2}$ corrects for this higher density. The derivatives are

$$\frac{\partial \mathbf{x}'}{\partial u} = (\mathbf{B}-\mathbf{A})/\sqrt{2},$$
$$\frac{\partial \mathbf{x}'}{\partial v} = (\mathbf{C}-\mathbf{A})/\sqrt{2}.$$

For example, suppose that N uniform samples are generated in the triangle. The expected density Υ of sampled points in a point $\mathbf{x}(u,v)$ in the triangle is[2]

$$\Upsilon(\mathbf{x}') = \frac{N}{\frac{\partial \mathbf{x}'}{\partial u} \times \frac{\partial \mathbf{x}'}{\partial v}} = \frac{N}{\mathbf{AB} \times \mathbf{AC}/2} = \frac{N}{\text{Area}(\mathbf{A},\mathbf{B},\mathbf{C})}.$$

7.A.4.2 Choosing the light source

Suppose there are a number of light sources that emit in total a flux Φ_{tot}. Usually light sources are chosen according to their emitted power, resulting in a probability P_l for choosing a light source l:

$$P_l = \frac{\Phi_l}{\Phi_{\text{tot}}}.$$

Note that this is a discrete sampling event, and no derivatives can be computed. We can, however, account for this probability by scaling the two partial derivatives of the sampled points with a factor $\sqrt{P_l}$. This factor accounts for the lower density of points on a single light source l (fewer samples are taken on this light source).

[2] Note that other procedures for sampling uniform points in a triangle are possible (see [112]). These result in other partial derivatives, but the density (i.e., the vector product of the derivatives) stays the same for all procedures.

Note that other discrete sampling events can also be handled by scaling the derivatives (e.g., Russian roulette).

8 Photon mapping

This chapter outlines a popular, two-pass global illumination algorithm, named photon mapping. In a first pass, particles are traced and stored in photon maps, which are used in a second, image-space stochastic ray tracing pass. In this chapter the standard photon mapping method is discussed along with several important optimizations needed to make photon mapping efficient. In the next chapter we present density control for photon maps, a technique to construct memory-efficient photon maps.

8.1 Introduction

Photon mapping is a two-pass method for global illumination that is able to handle all possible illumination features, such as indirect illumination, specular reflections and caustics, in a reasonably efficient manner.

The first pass in photon mapping consists of standard Monte Carlo particle tracing. The particles or 'photons' are shot from the light sources and are all stored individually in *photon maps*. A photon is represented by a position, an incoming direction, and the photon power. Typically two photon maps are constructed:

- A *global photon map* that stores all the particles and gives an approximate representation of the complete radiance function in the scene.
- A separate, high resolution *caustic photon map* is used for accurate caustic reconstruction. The caustic photon map will be visualized directly, while the global map will only be visualized indirectly.

To reconstruct radiance from the photon maps, techniques borrowed from density estimation are used: A number of photons nearest to the point of reconstruction are collected from the photon maps and are used to estimate the radiance. Photons are only stored on diffuse or glossy surfaces, because a reconstruction of the specular radiance component would require too many photons.

The second pass in photon mapping uses a stochastic ray tracing algorithm that is modified to take advantage of the different photon maps. The caustic photon map is visualized directly, because, as we mentioned before, caustics are much easier computed with light paths than with the eye paths of stochastic ray tracing. The global map is visualized indirectly after a diffuse or glossy bounce (a final gathering step).

Photon mapping was developed by Henrik Jensen. The standard method described above was published in '96 [47]. Earlier papers that used photon maps, focused on the caustic reconstruction [48], efficient shadow generation [51], and optimizing path tracing by photon map based direction sampling (but without explicitly using the radiance reconstruction from the photon map) [46]. Later, several extensions and optimizations were presented both by Jensen and other authors [53, 81, 58, P6, P9, P10].

Over the last few years photon mapping has become one of the most popular algorithms for computing global illumination. The main reasons for its popularity are the following:

- A photon map is independent of the underlying geometry. A photon has a position, a power, and an incoming direction, but it is not explicitly attached to a particular surface or patch. A photon map scales well with increasingly complex scenes. For highly tessellated scenes, it is possible to compute the illumination with less photons than there are geometrical primitives in the scene.

- Photon mapping handles all illumination phenomena in a reasonably efficient way. This is due to the specific multi-pass configuration: Caustics are represented separately in the caustic map, direct illumination is computed accurately by explicit sampling of the light sources, and indirect illumination is accurately computed through a final gather.

- The implementation of photon mapping only requires simple extensions to a standard Monte Carlo path tracer. The photons are stored in a kd-tree, and all transport is computed by Monte Carlo ray tracing.

Although the separate aspects of photon mapping, such as storing individual particles or final gathering, have been presented before, it is the specific combination of all these aspects that makes photon mapping such a robust and elegant algorithm:

- The storage of individual particles was also used in density estimation as presented by Shirley et al. [93] and improved by Wade [118] and Walter et al. [122, 121]. For each patch all particle hits are collected in a file. Each patch is then subdivided in a fine mesh, and for each mesh vertex the diffuse illumination is reconstructed from the particle hits using density estimation. The fine mesh is decimated for display purposes.

  While the concept of this method is similar to photon mapping, there are two important differences. The first difference is that a photon map is independent of the underlying geometry, while in density estimation the particles are collected per patch. Another difference is the use of a final gathering step. The indirect visualization of the global map allows a less accurate radiance reconstruction and, consequently, requires far fewer photons.

- Final gathering is frequently used for radiosity methods [18, 21, 7], because it allows a less accurate radiosity solution. The global photon map is also a coarse approximation of the radiance in the scene, but it includes non-diffuse, glossy illumination and is independent of geometric complexity. Glossy global illumination in complex scenes can become problematic for radiosity, leaving photon mapping as more robust alternative.

- Similar multi-pass configurations that separate direct illumination, indirect illumination, and caustics have been presented before [91, 18] (see also chapter 4). Again the geometrical independence of the photon map data structure provides an advantage with respect to these methods.

These advantages have led to a widespread adoption of photon mapping. It is currently available in many commercial and free rendering software [72, 113, 105, 9] and is being used increasingly in high quality animation production. For example, photon maps were used, although still sparingly, in the computer generated movie 'Final Fantasy, The Spirits Within' (Square Pictures).

The remainder of this chapter explains the different steps in the photon mapping method in more detail: Photon map construction (§8.2), the radiance reconstruction (§8.3), and the rendering pass (§8.4). We will indicate several problems, improvements and optimizations, but for a more extensive treatment we refer to Jensen's excellent book on photon mapping [50] and the SIGGRAPH 2000, 2001, and 2002 course notes [49, P9, P10] to which also other authors (P. Christensen, T. Kato and myself) contributed.

8.2 Photon map construction

The construction of a photon map is performed by standard particle tracing (§8.2.1). Since *all* the individual particles will be stored, some care has to be taken when choosing a data structure for the photon (§8.2.2).

8.2.1 Particle tracing

Particle tracing constructs paths starting at the light sources. The mathematics behind particle tracing were already discussed in chapter 3 (§3.4.2). Each vertex in a light path results in a particle. Therefore, N light paths can result in more than N particles. The number of particles is denoted by N_Φ.

Recall that a particle $\Phi(\mathbf{x}, \omega, \phi)$ consists of a position $\mathbf{x}$, a direction ω, and an associated weight ϕ, the power, energy, or color of the particle. A photon map stores incoming particles, that arrive at a surface in $\mathbf{x}$; the direction ω represents the incident direction of the particle. The global map stores all the particles and approximates the equilibrium incoming radiance distribution L_i.

Particles are only stored when the material of the intersected surface has a diffuse or glossy component. Reconstruction of the specular component of the outgoing radiance would require too many incoming photons in order to achieve a reasonable accuracy: A specular BSDF has a significant value for only a very small set of incoming directions (given a certain outgoing direction), so that only a tiny fraction of the photons (that do have a matching incoming direction) will contribute in the radiance reconstruction.

A high-resolution caustic map is constructed separately from the global map. As the name indicates, this map stores the caustic illumination which is formed by $(\mathsf{LS}^+(\mathsf{D}|\mathsf{G}))$ paths[1]. To construct the caustic

[1] If desired, indirect caustics can be added to a caustic map using the regular expression technique described in chapter 5

map, the scattering of light paths only takes the specular BSDF components into account. Again, particles are only stored on diffuse or glossy surfaces.

To accelerate the construction of the caustic map, Jensen suggests the use of projection maps [52]. Many particles emitted from the light sources may not hit a specular surface, and will not lead to caustic photons. A projection map encodes the solid angles through which specular surfaces are visible from a light source. Particles only need to be emitted into these directions, preventing emittance of unnecessary particles in other directions.

The photon tracing pass results in two sets of particles, one for the caustic and one for the global map. When participating media, such as smoke or clouds are present in the scene, a third map, a volume photon map, may be constructed [53].

Usually in the context of photon maps, particles are referred to as *photons*, which we will also do in the remainder of the chapter.

8.2.2 Photon data structure

Since all photons are stored, a memory efficient data structure for a photon will reduce storage requirement.

By clever encoding of the power and the direction of the photon, it is possible to reduce the memory needed by a single photon to 20 bytes [50, p.70].

8.3 Radiance reconstruction from the photon map

The radiance reconstruction tries to estimate the outgoing radiance on the surfaces in the scene using the set of photons in the photon map. This reconstruction problem can be formulated as a density estimation problem (§8.3.1). The photon map reconstruction uses *nearest neighbor* density estimation, where a number of nearest photons are located around the reconstruction point and used in the estimate (§8.3.2). This reconstruction is only approximate given the limited number of photons. In §8.3.3, we will analyze important characteristics of the reconstruction and discuss several extensions and improvements. Section 8.3.4 contains some remarks about how to efficiently query the nearest neighbors.

8.3.1 Density estimation

The field of density estimation addresses the problem of estimating a certain probability density function (pdf) from a number of observed samples. We will now introduce the necessary basics of density estimation and summarize the most important methods in density estimation.

More information on density estimation in general can be found in the excellent book by Silverman [98], which provides many practical insights in the method (compared to the focus on the asymptotic behavior of the estimators in many other density estimation papers).

8.3.1.1 Problem statement

Let $p(x)$ be an unknown probability density, with x a (possibly) multi-dimensional vector in a domain Ω_x. Given a number of observed samples x_i generated according to $p(x)$, density estimation tries to reconstruct the pdf $p(x)$ over the whole domain Ω_x. We are concerned with *non-parametric* density estimation, which means that we do not assume that $p(x)$ follows some known distribution (e.g., a normal distribution), for which only its parameters must be estimated (e.g., the mean and the standard deviation).

Since only a number of point samples are given, some degree of smoothing must be applied to the samples in order to reconstruct $p(x)$.

8.3.1.2 Three important methods

Three very important methods in density estimation are the histogram method, the kernel method, and the nearest neighbor method. For simplicity, formulas will be given for the estimation of a one-dimensional pdf. These are straightforward to extend to higher dimensions [98].

Histogram method The histogram method defines a number of bins in Ω_x and just counts the number of samples that fall into each bin. Let h be the width of the bins, and N the number of samples, then $p(x)$ is approximated as

$$\hat{p}(x) = \frac{1}{Nh} \times \text{ number of samples in same bin as } x\,.$$

The histogram method results in a piece-wise constant approximation; the estimate is constant over each bin. The size of the bin determines the amount of smoothing of the data. Larger bins reduce the variance in the estimate, but introduce more bias.

Heckbert was the first to recognize that radiance reconstruction from particles can be formulated as a density estimation problem [40]. In a particle tracing radiosity algorithm, he uses the histogram method, but adapts the bin sizes so that each bin exhibits about the same variance. This method was explained in some more detail in §7.8.2.

The histogram method is inferior both to the kernel and the nearest neighbor method, because of the discontinuities in the approximation. Also, the choice of origin of the bins can have an important influence on the estimates, which is undesirable: just shifting the bins can result in other estimates.

Kernel method The kernel method places a normalized kernel function around x and estimates $p(x)$ by weighting all samples that fall within the kernel support. Let $\mathcal{K}(x)$ be a kernel centered around the origin and normalized:

$$\int_{\Omega_x} \mathcal{K}(x)\,\mathrm{d}x = 1\,.$$

The kernel estimate is then given by

$$\hat{p}(x) = \frac{1}{Nh} \sum_{i=1}^{N} \mathcal{K}\left(\frac{x - x_i}{h}\right) . \tag{8.1}$$

where h, the *kernel width* or *bandwidth*, determines the amount of smoothing. The kernel estimate can be interpreted as a summation of copies of the kernel placed on each sample point. By increasing the number of sample points, a limit distribution is obtained:

$$\lim_{N\to\infty} \hat{p}(x) = \int_{\Omega_x} \mathcal{K}\left(\frac{x - x'}{h}\right) p(x') \, \mathrm{d}x' .$$

The limit distribution is the convolution of the kernel with the actual probability density. This shows that just increasing the number of samples will not lead to a correct result. A consistent estimator requires the bandwidth to decrease with the number of samples.

Adaptive bandwidth kernel methods, that choose a variable bandwidth for each x (or each sample point x_i), can better adapt to variations in the density of the samples and do provide consistent estimates.

In graphics, kernel density estimation has been used to reconstruct a finite element solution from particles [93, 118, 122]. Walter's dissertation [121] provides the most elaborate adaptive kernel method for this problem. In [P5] we presented a progressive algorithm that uses an adaptive kernel method to perform density estimation in the image plane.

Nearest neighbor method A third method, nearest neighbor density estimation, reconstructs $p(x)$ by locating the M nearest sample points x_i around x. Let $d_M(x)$ be the (positive) distance of the M-th nearest sample with respect to x, then the *generalized nearest neighbor estimate* is given by equation (8.1), but with the bandwidth h replaced by the distance $d_M(x)$:

$$\hat{p}(x) = \frac{1}{N d_M(x)} \sum_{i=1}^{M} \mathcal{K}\left(\frac{x - x_i}{d_M(x)}\right) .$$

The amount of smoothing is now determined by M, that, in turn, determines the bandwidth of the estimate. Thus, the nearest neighbor method provides an adaptive kernel method. The estimate is continuous in x, but its derivative is not, because $d_M(x)$ has a discontinuous derivative whenever the set of nearest samples changes. This fact complicates mathematical analysis of (asymptotic) error properties of the estimate.

On the other hand, nearest neighbor methods can be computed efficiently, which makes it quite a popular method. Fast algorithms are available for finding nearest neighbors, and the amount of work to evaluate the estimate is bounded by M, the fixed number of nearest neighbors. In the (adaptive) kernel method , all samples within a certain area have to be evaluated, and this number is not bounded.

The *standard nearest neighbor estimate* is a special case of the generalized estimate, where a constant kernel is chosen. The estimate reduces to

$$\hat{p}(x) = \frac{M - 1}{2 \, N \, d_M(x)} .$$

In graphics, the nearest neighbor method is used to reconstruct the radiance from photon maps. Additional properties of this method will be discussed further on.

8.3.2 Nearest neighbor radiance reconstruction

8.3.2.1 Radiance reconstruction as a density estimation problem

The problem of reconstructing radiance from a set of photons is slightly different from plain density estimation. In density estimation each sample has a unit weight, because it is an exact sample of the pdf to be estimated. We have to reconstruct a radiance function, which is not a pdf, using photons that can have an arbitrary weight, depending on how they were generated (see §3.4.2). This weight must be accounted for in the estimates.

Another difference is that we want to reconstruct the outgoing radiance; thus the photons, that represent incoming radiance, still have to be weighted by the BSDF.

Given these differences, a standard, two-dimensional, nearest neighbor (NN) estimator for the radiance leaving a planar surface in $\mathbf{x}$ is given by

$$\hat{L}(\mathbf{x} \to \omega_o) = \frac{\sum_{i=1}^{M} \phi_i \, f_s(\mathbf{x}, \omega_i \to \omega_o)}{\pi d_M^2(\mathbf{x})} \,. \tag{8.2}$$

Note that the number of light paths traced in the particle tracing pass (N) is hidden in the photon power ϕ_i (see §3.4.2). Note also that in the two-dimensional NN estimator, the sum of sample weights is divided by the area of the smallest circle centered in $\mathbf{x}$ that encloses the M nearest samples ($= \pi d_M^2(\mathbf{x})$).

The evaluation of this estimator requires finding the M nearest photons to a point $\mathbf{x}$. The BSDF is evaluated for each photon, but always in the point $\mathbf{x}$ itself. So for the BSDF evaluation, it is assumed that all photons arrive precisely in $\mathbf{x}$. The distance to the furthest photon is used to compute the area $\pi d_M^2(\mathbf{x})$, that determines the local density of photons.

The estimate (8.2) assumes that all nearest photons arrive on a planar surface around $\mathbf{x}$. Near the corners in a scene this may not be true. What happens in this case is discussed, together with other properties of the estimator, in the analysis of the reconstruction (see §8.3.3).

8.3.2.2 Radiance reconstruction as a Monte Carlo estimator

The radiance estimate (8.2) can also be derived as a standard Monte Carlo estimator. In §3.4.2.3 it was shown that a measurement could be estimated by the inner product of a self-emitted directional importance function with the radiance approximation defined by a set of particles.

The radiance reconstruction in $\mathbf{x}$ from a photon map corresponds to a particular choice of the self-emitted directional importance $W_e^{(\mathbf{x})}(\mathbf{y} \to \omega)$. This function is defined for any surface point $\mathbf{y}$: it defines the self-emitted importance in any point $\mathbf{y}$, given that we want to reconstruct radiance in $\mathbf{x}$.

The function $W_e^{(\mathbf{x})}$ is chosen as follows: The relation between the incoming radiance (w.r.t. projected solid angle) in $\mathbf{x}$ and the outgoing radiance (in a given direction ω_o) is given by the BSDF. By choosing the BSDF $f_s(\mathbf{x}, \omega \to \omega_o)$ as the self-emitted directional importance $W_e^{(\mathbf{x})}(\mathbf{y} \to \omega)$, the inner product with the incoming radiance gives the outgoing radiance in $\mathbf{x}$ for the given outgoing direction ω_o. However, no photons will arrive in $\mathbf{x}$ exactly, so an average over a certain circular area A_d is taken, with d the radius of a circle around $\mathbf{x}$. The directional importance is then given by

$$W_e^{(\mathbf{x})}(\mathbf{y} \to \omega) = \begin{cases} \frac{f_s(\mathbf{x}, \omega \to \omega_o)}{\pi d^2(\mathbf{x})} & \text{when } \|\mathbf{y} - \mathbf{x}\| \leq d \\ 0 \quad \text{when } \|\mathbf{y} - \mathbf{x}\| > d \end{cases}$$

This leads to the following estimator:

$$\hat{L}(\mathbf{x} \to \omega_o) = \langle W_e^{(\mathbf{x})}, \hat{L}_i \rangle = \sum_{j=1}^{N_\Phi} W_e^{(\mathbf{x})}(\mathbf{x}_j \to \omega_j) \cdot \phi_j = \frac{\sum_{i}^{\|\mathbf{x}_i - \mathbf{x}\| \leq d} \phi_i \, f_s(\mathbf{x}, \omega_i \to \omega_o)}{\pi d^2(\mathbf{x})} .$$

In this estimator, the first sum is taken over all photons j; $W_e^{(\mathbf{x})}$ will be zero for photons at a distance larger than d. The second sum only includes the photons i inside the circular area around $\mathbf{x}$.

The expected value of the estimator is given by

$$\begin{aligned} E\left[\hat{L}(\mathbf{x} \to \omega_o)\right] &= \int_{A_d} \int_{\Omega_{4\pi}} W_e(\mathbf{y} \to \omega) L_i(\mathbf{y} \leftarrow \omega) \, \mathrm{d}_\perp\omega \, \mathrm{d}\mathbf{y} \\ &= \frac{1}{\pi d^2} \int_{A_d} \int_{\Omega_{4\pi}} L_i(\mathbf{y} \leftarrow \omega) \, f_s(\mathbf{x}, \omega \to \omega_o) \, \mathrm{d}_\perp\omega \, \mathrm{d}\mathbf{y} . \end{aligned}$$

This is the convolution of a constant kernel defined over A_d with the reflected radiance (except that the BSDF is always evaluated in $\mathbf{x}$), which conforms to the expected values of standard density estimators.

The radius d can be chosen arbitrarily. If it is chosen as the distance to the Mth nearest photon to $\mathbf{x}$, then the resulting estimator for $\langle W_e^{(\mathbf{x})}, L_i \rangle$ is given by equation (8.2).

Jensen presents a slightly different derivation of the estimator, based on the differential flux of a particle [50]. While the resulting estimators are the same, alternative formulations, such as ours using a specific Monte Carlo estimator, can provide additional insight.

8.3.3 Reconstruction analysis

This section provides an analysis of the radiance estimate from the photon map. We will analyze several properties of the estimator, some of which are inherent to all nearest neighbor estimators, while others are introduced by the difference between the radiance estimator and standard density estimators. The following properties are discussed:

- The effect of searching for the nearest photons inside a sphere instead of a circle (§8.3.3.1).
- The impact of varying photon powers (§8.3.3.2).

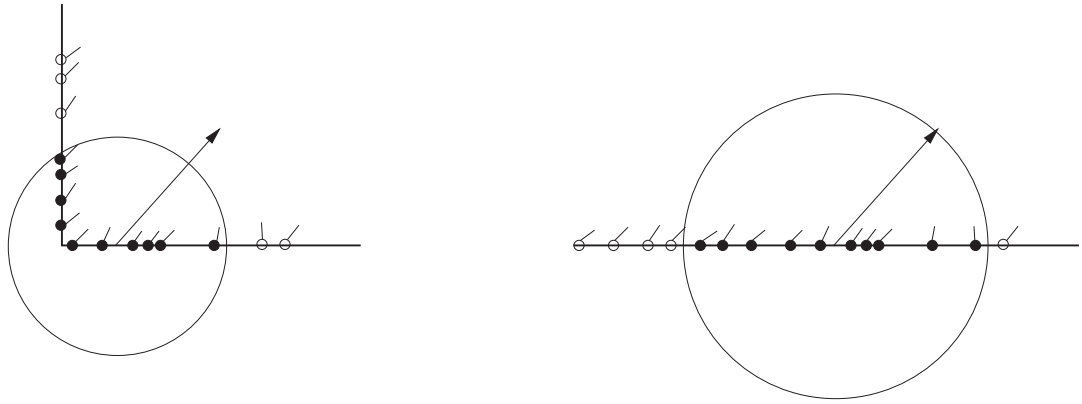


Figure 8.1: The use of a sphere to locate the nearest photons may include wrong photons, or photons at a distance different than when they would have hit the tangent plane in **x**. This can lead to an underestimation of the area (and a corresponding overestimation of the radiance).

- Boundary bias (§8.3.3.3).
- Radiance discontinuities and filtering (§8.3.3.4).
- Consistency (§8.3.3.5).

These properties are important for the next chapter on density control.

8.3.3.1 Nearest photons in a sphere

In the radiance estimate, it was assumed that all relevant photons arrive at a planar surface around **x**. On curved surfaces or near abutting surfaces, this assumption does not hold. This is why the nearest photons are located in a sphere around **x**. The circular cross-section of the sphere is used as the area estimate.

This introduces additional approximations in the radiance estimate: photons may lie closer by or further away compared to the case where they would have hit the tangent plane in **x**. This is shown in figure 8.1.

These effects can cause a slight overestimation of the radiance, which is sometimes visible in corners (e.g., the case shown in figure 8.1)

Jensen mentions that using a disk or an ellipsoid (by compressing the sphere in the direction of the normal) can lower such errors [50], because fewer 'bad' photons will be included. This would make the nearest neighbor queries slower, though, since testing if a photon lies within a disk or an ellipsoid is more expensive.

8.3.3.2 Photon powers

The radiance estimate accommodates varying photon powers. Although the varying powers do not change the expected value of the estimator, they have an important influence on the variance of the estimator. The variance of the estimator is directly proportional to the variance in the power of the different photons

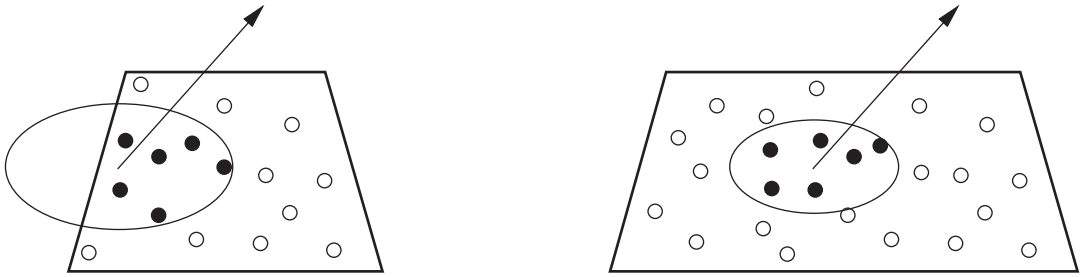


Figure 8.2: Nearest photon queries near object boundaries will only include photons at the inside of the surface. This leads to an overestimation of the area (left) compared to queries that stay fully inside a surface (right). This effect is called boundary bias.

(because their power appears in the estimator sum). Therefore, it is very important to keep this variance low.

Particle tracing should follow an analog simulation (§3.4.2.2) as close as possible. A perfect analog simulation results in an equal power for all photons, but requires direction sampling to be proportional to the BSDF times the cosine, which is not possible for all BSDF models.

Peter and Pietrek, for example, presented a method for importance driven construction of photon maps [81] that guides more photons to important parts in the scene. Their method results in highly varying photon powers, and is not suitable for direct radiance reconstruction; it was only used for importance sampling with the photon map.

Our density control framework in the next chapter will make sure that the photon powers remain homogeneous. The photon powers may differ a lot, but not locally within a small area.

8.3.3.3 Boundary bias

The term *boundary bias* indicates bias due to an underestimation of the density near the boundaries of the domain. Consider, for example, an isolated rectangular surface. At the boundaries of the rectangle, the sphere enclosing the nearest photons will have to grow larger because no photons are found outside of the rectangle (see figure 8.2). The larger sphere causes a larger area estimate, and thus a lower radiance reconstruction. The radiance darkens near isolated surface edges.

In kernel density estimation, there are a number of ways to remedy this problem:

- *Kernel mirroring* mirrors the part of the kernel outside the domain back into the domain and uses the samples found there. Shirley et al. used this method in their 1995 density estimation paper [93].
- *Kernel rescaling* finds the intersection of the kernel with the domain and scales the kernel accordingly (kernel support area divided by the intersection area). In rendering this has been used in [122].

These two methods depend on a geometrical description of the domain and run into problems with complex geometry.

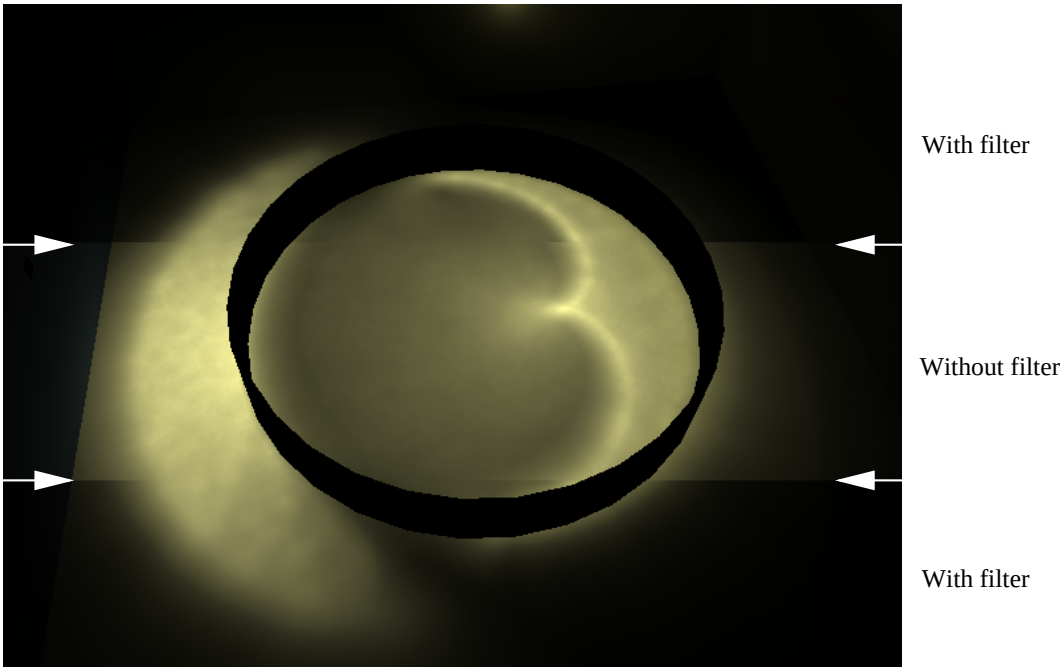


Figure 8.3: A caustic map, with and without filtering (Epanechnikov kernel). With filtering, blurring is reduced. (64.000 caustic photons, 200 nearest photons in the reconstruction)

Jensen [50] and Christensen [P10] suggest the use of kernel rescaling by finding the convex hull of the nearest photons to determine a better area estimate. This effectively reduces the boundary bias, but at the cost of a more expensive reconstruction.

8.3.3.4 Radiance discontinuities

Radiance discontinuities cause a sharp change in the density of photons. This is especially problematic when the radiance and thus the density drops to zero. Finding the nearest photons in the dark area will include photons at the boundary of the bright area.

In the case that the radiance drops to zero, the reconstructed radiance falls off as $1/r^2$ where r is the distance to the radiance discontinuity. This is illustrated in figure 8.3 (unfiltered parts). Radiance leaks under the ring into the shadowed area, due to the nearest neighbor search. In nearest neighbor (NN) density estimation, this is referred to as the *heavy tails* problem: the tails of the distribution are significantly overestimated.

In an area that receives no photons at all, the standard radiance estimator (8.2) will not even converge to the correct solution with an increasing number of photons, since always a number of nearest photons on the nearest discontinuity is used.

A solution to this problem is filtering the photons by certain filter kernel. This corresponds to going

from standard NN to a generalized NN estimator. Given a filter kernel $\mathcal{K}$, the radiance estimate becomes

$$\hat{L}(\mathbf{x} \to \omega_o) = \frac{\sum_{i=1}^{M} \phi_i f_s(\mathbf{x}, \omega_i \to \omega_o) \mathcal{K}(\frac{\|\mathbf{x}_i - \mathbf{x}\|}{d_M(\mathbf{x})})}{d_M^2(\mathbf{x})}, \tag{8.3}$$

where $\mathcal{K}$ is a normalized kernel on the unit circle. Note that the π factor disappears in this equation; this is because a box kernel, which was used in the standard estimator, evaluates to $1/\pi$ on the unit circle.

Any kernel that evaluates to zero on its boundary, will lead to a consistent estimator in a photon-less area: increasing the number of photons will cause the nearest photons (at the nearest radiance discontinuity) to lie closer to the boundary, reducing the estimated radiance.

The exact choice of the filter kernel is not overly important [98]. Jensen suggests the use of a cone filter, a Gaussian filter or a specialized filter based on differential checking [52].

We use an Epanechnikov kernel, commonly used in density estimation because of its good theoretical qualities [98]. The 2D Epanechnikov kernel is given by

$$\mathcal{K}_{\text{Ep}}(\frac{\|\mathbf{x}_i - \mathbf{x}\|}{d_M(\mathbf{x})}) = \frac{2}{\pi}(1 - \frac{\|\mathbf{x}_i - \mathbf{x}\|^2}{d_M^2(\mathbf{x})}).$$

This turns out to be very efficient to evaluate, because the nearest neighbor queries return a set of photons together with the squared distance to $\mathbf{x}$ (at least in our implementation)[2]. Figure 8.3 shows a comparison of a filtered and unfiltered reconstruction for a caustic map. In the filtered parts the blurring is reduced.

Some more advanced nearest neighbor reconstruction techniques were presented by Myszkowski [75]. These techniques work well when a high number of photons is used in the reconstruction, and were used for a direct visualization of the photon maps.

8.3.3.5 Consistency

The bias in the radiance estimate decreases with a smaller radius d_M. The variance decreases when more nearest photons are used in the estimate. For a consistent estimator, both the variance and the bias must vanish asymptotically. This can be accomplished by increasing the number of photons in the photon map, and (at a lower rate) the number of nearest photons in the estimate. Apart from the special case mentioned in §8.3.3.4, the radiance estimate will be consistent.

This is an interesting property of radiance reconstruction with photon maps: by just shooting more photons, the radiance approximation can be made as accurate as necessary.

8.3.4 Efficient nearest neighbor queries

In the rendering pass, many queries will be performed on the photon map. These queries consume a large part of the total computation time, and should be as efficient as possible.

[2]The cone filter requires a square root of the squared distances, while the Gaussian filter evaluates an exponential function per photon

A data structure for efficient nearest neighbor queries is the kd-tree [11]. The number of dimensions in the search space is given by 'k', so in the case of photon maps, a 3d-tree is constructed.

A kd-tree is a binary tree. Each node contains a single photon and splits the (remaining) search space in two half-spaces along a single dimension. The remaining photons are divided according to which half-space they belong to, and two subtrees are constructed recursively. The splitting dimension is indicated by a flag in the node. The split is best performed in the dimension that shows the biggest spread of the photon positions.

Most efficient is a balanced kd-tree, where the depth of the left and right sub-tree in a node differs by at most one level. A left balanced kd-tree fills nodes (breadth first) from left to right. Such a tree can be represented in a contiguous array; children of a node are obtained by simple index arithmetic, saving two child-pointers that are needed in an unbalanced tree. The memory gain is significant, because the photon data structure is already small in size.

In our implementation we have added support for both unbalanced and balanced kd-trees. An unbalanced kd-tree is useful for querying the photon map *during* its construction. This will be needed in our density control framework.

More details and an example implementation can be found in [50]. Our implementation is available through the RENDERPARK source code [9]. A few more optimizations can be found in [P9, P10].

8.4 Rendering with photon maps

The rendering pass in photon mapping is based on stochastic ray tracing and uses a fairly standard multi-pass configuration. Figure 8.4 shows a schematic overview of the different contributions in the rendering pass.

When tracing an eye path, the different radiance components are computed as follows:

- **Specular component:** The specular component is always computed by extending the eye path, sending out a specularly scattered ray.
- **Diffuse and glossy component:** The diffuse and glossy component of the reflected (and refracted) radiance can be computed in two ways: an accurate estimate or an approximate estimate using the global photon map:
  - Accurate estimate: The accurate estimate computes the direct illumination by explicitly sampling the light sources. Caustics are reconstructed from the high density caustic map, and indirect illumination is computed by final gathering. For the final gathering, many scattered rays are traced (using $(\mathsf{D}|\mathsf{G})$ components only) to sample the incoming radiance.

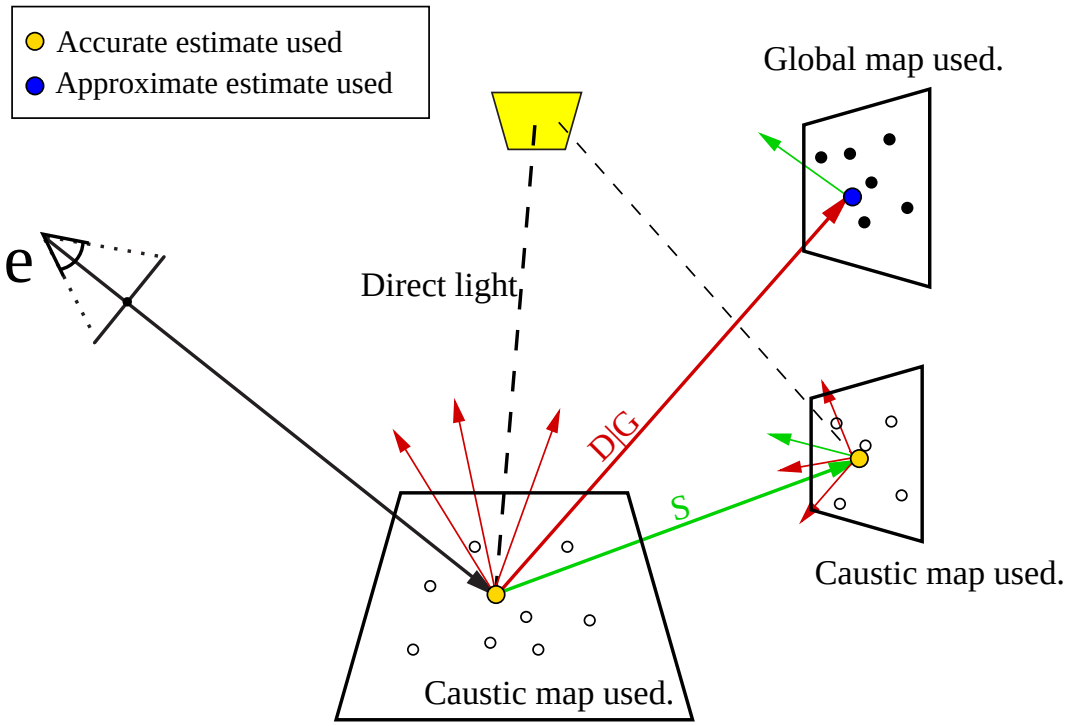


Figure 8.4: The different contributions when rendering with photon maps.

– Approximate estimate: The approximate estimate reconstructs diffuse and glossy radiance directly from the global photon map.

The accurate estimate is used whenever a path has not been scattered with a diffuse or glossy component, thus on directly visible surfaces and after one or more specular reflections or refractions. The approximate estimate is used for the final gather rays, thus as soon as a diffuse or glossy scatter has taken place.

Note that the self-emitted radiance, which is not present in the radiance reconstruction from the global map, does not have to be taken into account: if a final gather ray hits a light source directly, it would contribute to the direct light, which is computed separately in the accurate estimate. Alternatively, a final gather ray might hit a light source after some specular reflections. Also in this case the self-emitted radiance should not be included, because it would contribute in the form of a $(\mathsf{LS}^+(\mathsf{D}|\mathsf{G}))$ path, that is already handled by the caustic map.

The final gathering masks the errors in the radiance reconstruction from the global map. However, the error on surfaces very close to the point where final gathering is performed, may be visible in the accurate estimate. This case is shown in figure 8.5. The wall close-by (red rays) covers a large part of the hemisphere with respect to the final gathering point. The error in the reconstruction in that area will have an important influence on the error in the final gathering estimate. Therefore, the low-frequency noise of the radiance reconstruction along the contact line between the floor and the wall, may show up in the final gathering.

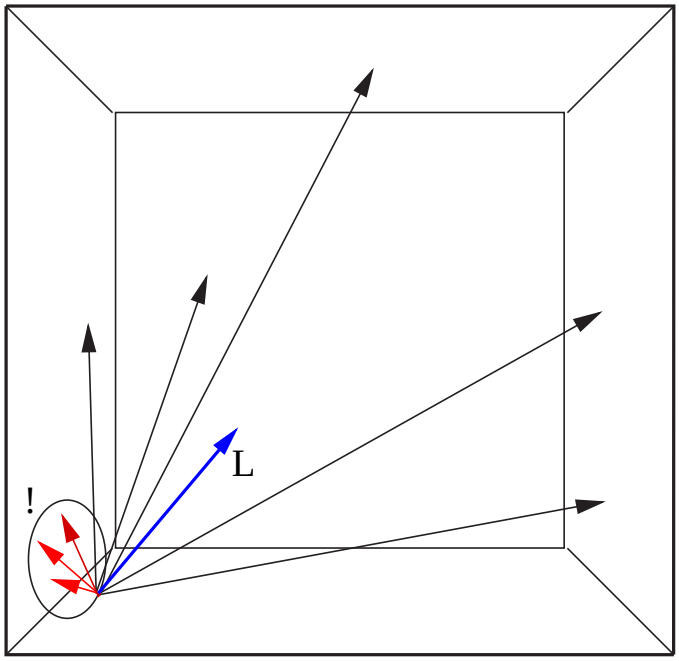


Figure 8.5: A final gather for a point near a corner. Many rays will hit the close-by, abutting surface. The errors in the radiance reconstruction in the encircled area may be visible in the final gathering result. Therefore, secondary final gathers are performed for rays that only travel a very small distance.

Jensen identified this problem [47] and uses the accurate estimate again when the distance traveled by a final gather ray is under a certain threshold. In the close-by hit points a *secondary* final gather is performed[3].

Looking at how the different stored radiance solutions (SPARs) are used, the paths covered in the final image can be identified (ignoring the secondary final gathers). Three stored solutions are used: the caustic map, the global map, and the self-emitted radiance (which is also considered as 'stored' radiance, see also chapter 4):

- **Caustic map:** This map covers $LS^+(D|G)$ paths, which are read by eye paths that may have been scattered specularly. The total transport is $LS^+(D|G)S^*E$.
- **Self-emitted radiance:** The self-emitted radiance is used for the direct light in the accurate estimate and when a specular path hits a light source directly: $\mathsf{L(D|G)S^*E + LS^*E}$ paths are covered.
- **Global map:** The global map stores $\mathsf{LX^*(D|G)}$ paths and is used after final gathering ($\mathsf{(S^*(D|G)S^*E)}$ paths). The paths covered are: $\mathsf{LX^*(D|G)\ S^*(D|G)S^*E}$.

Summation of these contributions shows that the rendering via photon mapping covers all illumination exactly once. A complete global illumination solution is obtained.

[3] In figure 8.5 the problem is exaggerated for illustration purposes: in practice, only a very small region near the corner will require a secondary final gather.

8.5 Optimizations

The most expensive part in photon mapping is the rendering pass, and more specifically the final gathering. Accurately sampling the incoming radiance can require up to a few thousand final gather rays. Each such ray will initiate a query in the global map. Typically more than 95% of the rendering time is spent in final gathering. Therefore, many optimizations have been added to the rendering pass to speed up this process.

Several of these are aimed at speeding up the photon map queries (§8.5.1, §8.5.2), while others try to decrease the number of expensive final gathers (§8.5.3). These optimizations are crucial for an efficient algorithm. We will only briefly outline the optimizations, as our work focuses more on the construction of photon maps. Additional information can be found in the supplied references.

8.5.1 Maximum search radius

Specifying a maximum search radius when searching the kd-tree for nearest photons can greatly speed up the query. In sparse regions, where the photon density is very low, large parts of the kd-tree will be searched in order to find the nearest photons. Since the nearest photons will span a large area, the radiance estimate will be very low, while the bias may be relatively high (e.g., when photons from a high density region are erroneously included).

A good maximum search radius can prevent searching unnecessary parts of the tree. However, when the radius is set too small, not enough photons will be found, and the estimate may become inaccurate, while a large radius may not improve the query time.

We use a heuristic to automatically determine a suitable maximum radius. The radiance estimate for M nearest photons is given by (8.2):

$$\hat{L} = \frac{1}{N} \frac{\sum_{i=1}^{M} \phi_i f_s(\mathbf{x}, \omega_i \rightarrow \omega_o)}{\pi d_M^2(\mathbf{x})} .$$

If this estimate results in a very low radiance, we should have decreased the search radius. The question now is when a radiance value can be labeled as low. A sort of *ambient* radiance estimate can be constructed by considering *all* N_Φ photons in the photon map. The ambient radiance is given by

$$\hat{L}_{\text{amb}} = \frac{1}{N} \frac{\sum_{i=1}^{N_\Phi} \phi_i f_s(\mathbf{x}, \omega_i \rightarrow \omega_o)}{\pi d_{N_\Phi}^2} ,$$

where d_{N_Φ} is the radius of the sphere enclosing all photons.

A radiance estimate is labeled as low when it is below a certain small percentage α of the ambient radiance. The maximum radius is reached when

$$d_{\max} \Leftrightarrow \hat{L} = \alpha \hat{L}_{\text{amb}} .$$

We simplify both estimates by replacing ϕ_i with an average photon power ϕ_{avg} and by replacing the BSDF

evaluation by an average diffuse BRDF ρ_{avg}/π. The equality becomes:

$$\frac{1}{N}\frac{M\cdot\phi_{avg}\rho_{avg}/\pi}{\pi d_{max}^2} = \alpha\cdot\frac{1}{N}\frac{N_\Phi\cdot\phi_{avg}\rho_{avg}/\pi}{\pi d_{N_\Phi}^2}.$$

Many factors cancel out, leading to the following expression for the maximum radius:

$$d_{max} = \sqrt{\frac{M\cdot d_{N_\Phi}^2}{\alpha N_\Phi}}.$$

This maximum radius can now be supplied with the photon queries

Christensen proposes a slightly different heuristic [P9], but it is based on the same idea: deriving the maximum radius through a radiance threshold. His heuristic is given by

$$d_{max} = \frac{1}{\pi}\sqrt{\frac{M\phi_{max}}{L_t}},$$

with ϕ_{max} the maximum photon power present in the map, and L_t a user defined radiance threshold. A minor difference is that the radiance threshold depends on the scene (i.e., on the total self-emitted radiance), while our percentage α does not. Both heuristics perform well and show the same square root dependence on M.

The speedup is especially noticeable with caustic maps, since these typically have many regions of low density. For balanced kd-trees, visualizing the caustic map can be easily twice as fast, while for unbalanced trees the speed-up can be even better.

8.5.2 Irradiance precomputation

The radiance estimate in the global map can tolerate quite some error, due to the final gathering. Christensen [20] recognizes this and proposes a preprocessing step that precomputes irradiance with the global map in a limited number of positions (namely the photon positions).

After the construction of the photon map, irradiance is precomputed for all photon positions and stored with the photon. During rendering, the diffusely reflected radiance can now be estimated by just looking for the nearest photon and multiplying its precomputed irradiance value with the diffuse BRDF.

Irradiance, however, depends on the normal of the surface, as it is an integral over the upper hemisphere. Therefore, the normal of the surface that was hit by a photon, is also stored with that photon. The query for a nearest 'irradiance' photon will only accept photons that have a similar normal, ensuring a good irradiance estimate. The resulting radiance reconstruction shows the Voronoi regions of the irradiance photons.

Storing both the irradiance and the normal with a photon increases memory demand, but the speed-up is large, because only the single nearest photon has to be located. Christensen reports speed-up factors of up to 6, which was confirmed by our implementation and experiments.

For the caustic map, this approach is not used, because a higher accuracy is needed for the direct visualization.

For more information, refer to [20, P10] or to [9] for an implementation.

8.5.3 Irradiance caching

While the two previous techniques try to accelerate the photon queries, the irradiance caching optimization tries to reduce the number of final gathers. Irradiance caching was presented by Ward et al. in 1988 [126] as a general acceleration technique for computing indirect diffuse illumination in stochastic ray tracing.

The main idea is to accurately compute a final gather in a selected number of points in the scene only, and to interpolate these results for all points in between. The high level algorithm is shown in algorithm 4.

The details on the caching and interpolation can be found in [126, 67]. Our implementation was inspired by the implementation in the freely available Radiance system [124, 14].

By computing the gradient of the irradiance in the sample points with respect to a change in position or orientation, a better interpolation can be performed without much extra work [125].

Since irradiance is cached, the specific distribution of the incoming radiance is lost, and only diffuse radiance can be computed using this method. The irradiance multiplied by the diffuse BSDF results in the scattered diffuse radiance.

The speed-up obtained with irradiance caching can be enormous. Depending on the accuracy, which has to be chosen by the user, only a small fraction (e.g., 1%) of the indirect diffuse illumination evaluations needs to be done with an expensive final gather. Since final gathering dominates the computation time, this can lead to speed-up factors of 100 or even more, depending on the scene.

However, the accuracy has to be chosen carefully, otherwise interpolation artefacts may become visible. The results in the next chapter on density control do not use irradiance caching in order to make certain that these artefacts do not interfere with our comparisons.

8.6 Conclusion

Photon mapping provides an elegant algorithm to compute global illumination in complex scenes. The photon storage is independent of the geometrical complexity of the scene, which is one of the important advantages over finite element methods.

Algorithm 4 Irradiance caching: high level overview

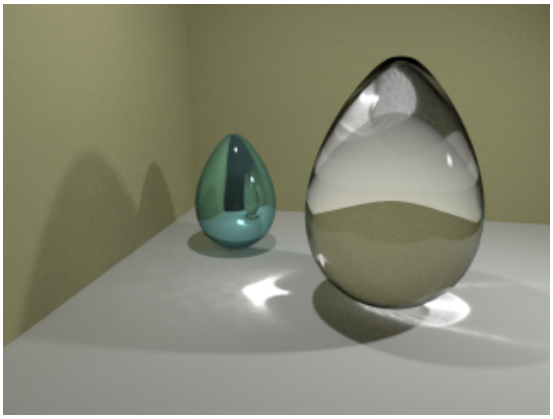
- **Compute indirect diffuse illumination in x:**
  - Find all cached irradiance points that are valid in **x**. The validity radius is determined by a user-defined accuracy and the scene geometry (points close to other geometry, e.g., in corners, will have a smaller validity radius, because the irradiance will change more rapidly in these regions). This radius is computed together with the accurate final gather.
  - If the number found $<$ user defined minimum (Ward uses 1, we use 3 for a better interpolation)
    - Generate a new point in the cache by an accurate final gather.
  - else interpolate between all the cached values

Rendering with photon maps requires a final gather and rather expensive nearest neighbor queries, but several intelligent optimizations reduce the rendering time dramatically. We described several optimizations, including a novel maximum radius heuristic, that we have all implemented into RENDERPARK. Tests with several scenes showed a speed-up factor of up to 300.

But several other optimizations are possible, many of which are applicable to other global illumination algorithms as well. A few examples are parallel computations, the use of quasi-Monte Carlo sequences to obtain a more even distribution of photons in the scene, and the use of shadow photons to accelerate the computation of shadows.

Most of the optimizations are targeted towards the rendering pass, since that is the computationally most expensive part. In the next chapter we will focus on the construction of photon maps, in order to reduce memory requirements and to introduce some level of error control into the photon map construction.

We conclude this chapter with a few examples. Figure 8.6 shows a few scenes rendered with photon mapping in RENDERPARK. Note the indirect illumination and the caustic effects.



- Eggs -

- Glass knot -

- Earth, wind, fire, water and photons -

Figure 8.6: Three images rendered using photon maps in RENDERPARK. Note the caustics and indirect illumination.

9 Density control for photon maps

In this chapter we will focus on techniques to construct memory-efficient photon maps. A technique is presented to control the density of photon maps according to a certain target density criterion. Such a criterion is derived from an initial eye-based pass that computes the importance of each point in the scene with respect to the viewpoint. The method results in fewer stored photons and leads the way to a full error controlled photon mapping algorithm.

9.1 Introduction

While very realistic images can be computed efficiently with photon maps, as we have seen in the previous chapter, photon mapping is not (yet) the perfect global illumination algorithm. Several difficulties still need to be addressed:

- The accuracy of the radiance reconstruction with a photon map is controlled by two parameters: the number of light paths shot in the particle tracing pass (this determines the number of photons in the map), and the number of nearest photons used in the reconstruction. A more accurate radiance solution is obtained when both the number of light paths and the number of nearest photons is increased. It is, however, not obvious how large these numbers should be in order to obtain a sufficiently accurate photon map. The required accuracy for the photon maps depends on the illumination in a specific scene and on the viewing parameters in that scene. Hence, the parameters have to be determined individually for each scene.

  Currently these parameters have to be chosen by the user. This requires an experienced user who understands the techniques behind photon maps and the radiance reconstruction. It would be preferable to have an automatic procedure to determine these parameters. Developing such a procedure requires a full error analysis of the photon map, which is very difficult.

- Another important issue with photon maps is memory utilization. Storing all the individual photons can take a lot of memory, even when a single photon requires only around 20 bytes. Especially complex, accurate caustics or large scenes require many photons. Since many nearest photon queries are performed during the rendering pass, all photons should be kept in main memory for fast access. Storage on a hard-disk, as is done in density estimation [93], would slow down the rendering considerably.

  During photon tracing, some regions in a scene, for example the bright parts in caustics or unimportant parts of the scene, may already have received enough photons while other regions still need more.

Reducing the number of stored photons in these over-dense regions, without sacrificing accuracy, can reduce the memory requirements of photon maps significantly.

The density control framework, which we present in this chapter, offers some answers to these problems. First, a method is presented to control the local density of a photon map based on a required or target density criterion. When a photon arrives in a region that already has a sufficient density, it will not be stored, but its power will be distributed among nearby photons in the map. This approach has two main advantages:

- The density of photon maps can be controlled locally in the scene. Less photons are stored in over-dense or unimportant regions. This can reduce memory requirements quite effectively.

- The concept of a required or target density offers an interesting framework for error control in photon maps. Since the density of photon maps is related to their accuracy, a high density should be chosen for important regions in the scene. The target density can be chosen arbitrarily and can be based on principles like view importance, relative error, visual masking by textures, ...

  We derive a target density criterion based on importance. In a new, initial pass, importance maps are constructed that indicate how important regions in the scene are with respect to the viewer. A simple heuristic relates importance to the target density of the photon map.

Results obtained with the density control framework show a significant reduction in the number of stored photons, both for the caustic and the global map. But maybe even more important is that the density control framework, together with importance, leads the way to error control. Accurate error control will eventually lead to a fully automatic photon map construction, where the user does not have to guess an appropriate number of photons in the map or the number of nearest photons in the reconstruction. In our opinion, this is the biggest remaining challenge in the photon mapping method.

Section 9.2 introduces and analyzes our density control technique. In §9.3 an error analysis for rendering with photon maps is developed, which leads to an importance based target density criterion. The practical computation of importance requires a new, initial pass that computes importance maps (§9.4). We present two alternatives for computing importance: one based on nearest neighbor density estimation, similar to the photon map, and another based on path differentials. Results of the density control for both the global and the caustic map are presented in §9.5. We finish the chapter with a comparison to other approaches for controlling the photon map density (§9.6), draw some conclusions and indicate interesting directions for future research towards full error control (§9.7).

An overview of the resulting importance driven photon mapping method developed in this chapter is given in algorithm 5.

Some parts of this research were previously published in [P6] and [P9, P10].

Algorithm 5 Importance driven photon mapping with density control: overview of the algorithm

```
1 Compute an importance map by tracing eye paths
    (importons) from the viewer (gives a target density)
2 while(photons to trace/store)
    trace photon
    if (density of photon map at this photon position <
        target density)
      store photon
    else
      distribute photon power over nearest neighbours
3 Render an image as with the standard photon map method
  (but with less photons stored in the maps)
```

9.2 Density control framework

9.2.1 Overview

The goal of density control is to adapt the actual density of a photon map to a certain given target or required density (§9.2.2). When tracing a new photon, the current density of the photon map will be evaluated (§9.2.3) and compared to the target density. If the density is too low, the photon will be stored.

If the current density is already sufficient, we do not want to store the photon. However, we cannot just discard it, because energy would be lost, leading to an incorrect radiance solution. We account for the discarded photon by distributing its power over neighboring photons (§9.2.4). This ensures an energy balance and keeps the photon powers in the map homogeneous, which is very important for a good radiance reconstruction (§9.2.5). An analysis of the redistribution results and some improvements are presented in §9.2.6.

9.2.2 Target density

Density control requires the specification of a target density throughout the whole scene. The target density in a point x will be denoted by $D_{\text{tar}}(\mathbf{x})$.

The results in this section will all use a hand-picked constant target density. In §9.3 we will derive a target density criterion based on importance.

Note that our definition of the target density does not include the direction. An alternative definition could take into account the directions of the photons when estimating the current density. This would require an extension of the target density to $D_{\text{tar}}(\mathbf{x}, \omega)$. Since both estimating the current density and specifying the target density would be more complicated, we currently use a target density that is independent of the direction. For diffuse surfaces this makes no difference, but for highly glossy surfaces a directional density might give better results.

9.2.3 Current density

Let $\Phi_j(\mathbf{x}, \omega_j, \phi_j)$ be a new photon that arrives in $\mathbf{x}$ on a surface, with an incoming direction ω_j and a power ϕ_j. To determine whether the photon needs to be stored, the current density is estimated by locating a number of nearest photons. For M nearest photons, the current density is given by

$$D_{\text{cur}}(\mathbf{x}) = \frac{M}{\pi d_M^2(\mathbf{x})}\,.$$

If $D_{\text{cur}}(\mathbf{x}) < D_{\text{tar}}(\mathbf{x})$, the photon is stored; if not, its power is distributed.

Note that this estimate requires a photon map query *during* the construction of the photon map. Therefore, our implementation stores photons directly in an unbalanced kd-tree during the construction, so that the query is efficient.

9.2.4 Photon redistribution

When a photon is not stored, its power must be accounted for. This can be done by distributing the power over the nearest photons.

The radiance reconstruction in $\mathbf{x}$, *including* the newly arrived photon j, is given by

$$\hat{L}(\mathbf{x} \to \omega_o) = \frac{1}{N} \frac{\sum_{i=1}^{M} \phi_i\, f_s(\mathbf{x}, \omega_i \to \omega_o) \;+\; \phi_j\, f_s(\mathbf{x}, \omega_j \to \omega_o)}{\pi d_{M+1}^2(\mathbf{x})}\,.$$

Note that the radius $d_M(\mathbf{x})$ without photon Φ_j stored, is equal to the radius $d_{M+1}(\mathbf{x})$ with Φ_j stored, because the photon is located precisely in $\mathbf{x}$.

Since the photon is not stored, the power of the other photons must be adjusted in such a way that the reconstruction in $\mathbf{x}$ would deliver the same (or at least a similar) result:

$$\hat{L}(\mathbf{x} \to \omega_o) = \frac{1}{N} \frac{\sum_{i=1}^{M} (\phi_i + \Delta\phi_{i,j})\, f_s(\mathbf{x}, \omega_i \to \omega_o)}{\pi d_M^2(\mathbf{x})}\,,$$

where $\Delta\phi_{i,j}$ denotes the power adjustment of photon Φ_i due to not storing Φ_j.

Different choices for $\Delta\phi_{i,j}$ can be made depending on the BSDF f_s, the incoming direction, or the distance of $\mathbf{x}$ to a photon Φ_i:

- **BSDF $f_s(\mathbf{x}, \omega_i \to \omega_o)$:** To get an equal reconstruction $\hat{L}(\mathbf{x} \to \omega_o)$ in $\mathbf{x}$, $\Delta\phi_{i,j}$ should be zero for photons that have $f_s(\mathbf{x}, \omega_i \to \omega_o) = 0$: Since the BSDF is zero, these photons can not contribute to $\hat{L}(\mathbf{x} \to \omega_o)$. Any power assigned to such photons would not be taken into account in the radiance reconstruction in $\mathbf{x}$.

  Currently, we test the angle between the photon direction ω_i and the normal $\mathbf{N}_x$ in $\mathbf{x}$ to determine whether $\Delta\phi_{i,j}$ should be zero (i.e., for a non-transparent material, $\Delta\phi_{i,j} = 0$ when $\cos(\omega_i, \mathbf{N}_x) \leq 0$).

- **Incoming direction:** Another approach would be to choose a larger delta for photons with a direction similar to the distributed photon. This might be better for non-diffuse BSDFs, but at the cost of a

less smoothly varying photon power: photons with different incoming directions may accumulate a varying amount of energy.

- **Distance to x:** The distribution of the photon power can be seen as applying a low-pass filter (or as splatting). The dependence of $\Delta\phi_{i,j}$ on the distance to **x** is determined by the filter kernel.

  We distribute the power equally over the affected photons (a box filter) to keep the photon powers homogeneous which, as said, is beneficial for the reconstruction.

So to summarize, we choose:

$$\forall i, cos(\omega_i, \mathbf{N}_x) > 0 : \Delta\phi_{i,j} = \phi_j / M',$$

with M' the number of photons that have a positive cosine with respect to the normal.

For diffuse surfaces, this choice results in an equal reconstruction in **x**, whether the photon is stored or not. For glossy surfaces, the BSDF evaluation of the other photons may be different from the BSDF evaluation of the discarded photon. This will result in a difference in the reconstructed radiance. This might be improved by applying a directionally dependent redistribution on non-diffuse surfaces.

Of course, even in the diffuse case, the radiance estimate at locations other than **x** will give a slightly modified result. But since the current density is high enough anyway (otherwise the photon would have been stored), this averaging can be expected not to introduce artefacts.

9.2.5 Russian roulette storage

We have also tried an alternative way to control the density of photon maps based on Russian roulette.

A strictly positive acceptance probability P_{acc} is defined as a function of the ratio of current and target density (e.g., $P_{\text{acc}} = \max(1 - \frac{D_{\text{cur}}(\mathbf{x})}{D_{\text{tar}}(\mathbf{x})}, 0.01)$). Russian roulette draws a random number, and if it is smaller than P_{acc}, the photon is stored; otherwise it is discarded. When a photon is stored, its power has to be multiplied by $1/P_{\text{acc}}$ to keep an energy balance.

However, when the current density approaches the target density, the acceptance probability becomes very small. In the rare event that a photon is accepted, the power is multiplied by a large factor and becomes much larger than the power of photons stored earlier. The resulting mix of high and low powered photons causes a high variance in the radiance estimate. Therefore, we abandoned this approach and developed the redistribution.

9.2.6 Results and analysis

In this section some results of the redistribution will be analyzed.

A simple scene is used for which a global photon map is computed: The scene consists of a ground plane and a red blocker plane perpendicular to it. A single small light source illuminates the scene. The

global map is always visualized directly to clearly demonstrate the effects of the redistribution. Unless stated otherwise, all images use 80 nearest photons in the reconstruction and also use the Epanechnikov filter (see §8.3.3.4).

These results were generated on a 1Ghz Pentium III in a 256MB Dell Inspiron 8000 portable.

9.2.6.1 Constant target density

A first example compares the standard photon map construction with a density controlled construction using a constant target density throughout the scene. The results are shown in figure 9.1.

The top row (a) shows a standard photon map construction with around 65.000 photons in the map. For this scene, this required about 120.000 light paths, since many rays traced from the light source miss the scene. The bottom row (c) also uses a standard photon map with around 400.000 stored photons.

The middle row (b) shows a density controlled solution. The target density was set to 1500 photons per square meter. The area of the complete scene is about $48m^2$. Note that the target density automatically defines an upper limit for the number of stored photons ($\pm$72.000 photons in this case). For the image as many light paths were traced as for the solution with 400.000 photons. Instead of 400.000 stored photons, the density control allowed only around 55.000 photons to be stored.

The comparison of the radiance reconstruction brings up some interesting points:

- Overall the illumination is very similar, which is logical because the redistribution was designed to maintain energy balance.
- The variance in (c) using 400.000 photons is not reduced compared to (a), because the same number of nearest neighbors is used (80 photons). Note especially the low frequency noise in the red color bleeding on the floor, which is caused by a mixture of pure red and white photons. The bias in (c) on the other hand is reduced by the higher number of stored photons, which is evident in the less blurry shadow boundary.
- The variance in (b) is lower than both (c) and (a). This is due to the redistribution. Excess photons are distributed over the neighbors, which, in this case, causes red photons to be distributed over white ones and vice versa. This leads to more homogeneous photon powers and reduces the noise due to the color variation of the photons (In the end, photons will reach an equilibrium mix of red and white). This is a fortunate side effect of the redistribution.
- The bias in (b) is about the same as for image (a) because an equal number of photons is stored. However, for scenes where the target density is low compared to the size of the geometrical detail, the redistribution can cause a larger bias. This will be illustrated in a later example (§9.5.2).

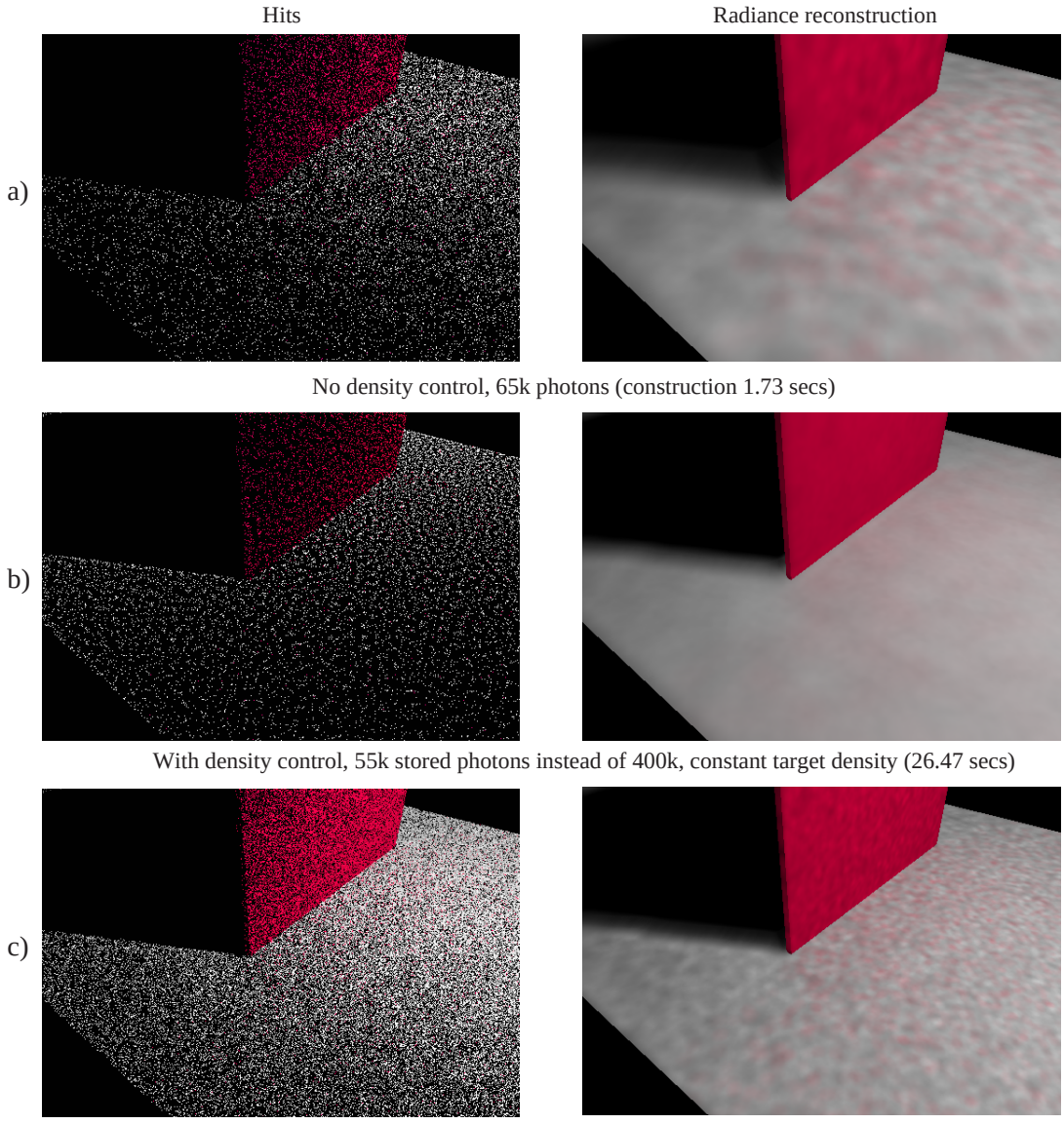


Figure 9.1: Density control results: The top and bottom row show the hits and radiance reconstruction for a photon map constructed without density control with a varying number of photons. The middle row used a constant required density. Overall the radiance reconstruction is similar. The density control even causes a reduction in the variance, although a smaller number of photons are stored.

The time to build the density controlled photon map (26 secs) is long compared to the other photon maps (1.7 secs for (a), 11.3 secs for (b)). It takes about twice as long as the photon map in (c) for which the same number of light paths were traced. This is because with density control, each photon requires a query for the nearest photons in the current photon map in order to estimate the current density. For redistribution the same nearest photons are used, so that only one query is necessary. For these images we used 20 nearest photons to estimate the density and for the redistribution. Using more nearest photons takes even longer, but can also introduce additional bias (see §9.2.6.2).

As was mentioned, the number of stored photons in the density controlled map is 72.000 when the target density is reached everywhere in the scene. The time it takes to reach this maximum number can be terribly long. In the example scene, the region in shadow does receive photons through reflection from the floor onto the back of the red blocker, but only very few.

Instead of waiting until all regions have reached the target density, it is better to test the percentage of photons that is being added as new to the photon map. For example, we emit photons in batches of about 10000 photons. When the number of newly stored photons is a small percentage of that, the photon construction phase is ended. Some regions will not have reached the target yet, but they are very dark anyway as the percentage of photons arriving there is proportional to the received radiance. This provides a convenient stopping criterion for the photon map construction.

9.2.6.2 Target density discontinuity

Since the redistribution uses the nearest neighbors to distribute the power, the same problems with discontinuities can be expected as with NN density estimation. This section analyzes what happens with discontinuities in the target density. The next section looks at discontinuities in the photon density.

Figure 9.2 shows the result of a discontinuity in the target density (left column). For the front half of the scene, $D_{\text{tar}} = 1500$, while the rear half has $D_{\text{tar}} = 100$. This causes a sharp discontinuity as can be seen in the bottom left image.

The photon map density adapts nicely to the target density (middle), but the radiance reconstruction shows a clear overestimation at the dense side of the discontinuity (top). This is caused by redistribution of photons that arrive in the low density region: the nearest photons include several lower powered photons in the high density region that receive a part of the power. Many more photons are redistributed in the low density region, and cause a significant power leak into the high density region.

This overshooting effect leads to the following recommendations:

- To prevent excessive bias, a small number of photons should be used in the redistribution. We use the 20 nearest photons, also in these images, but when more photons are used, the bias at the discontinuity increases further.

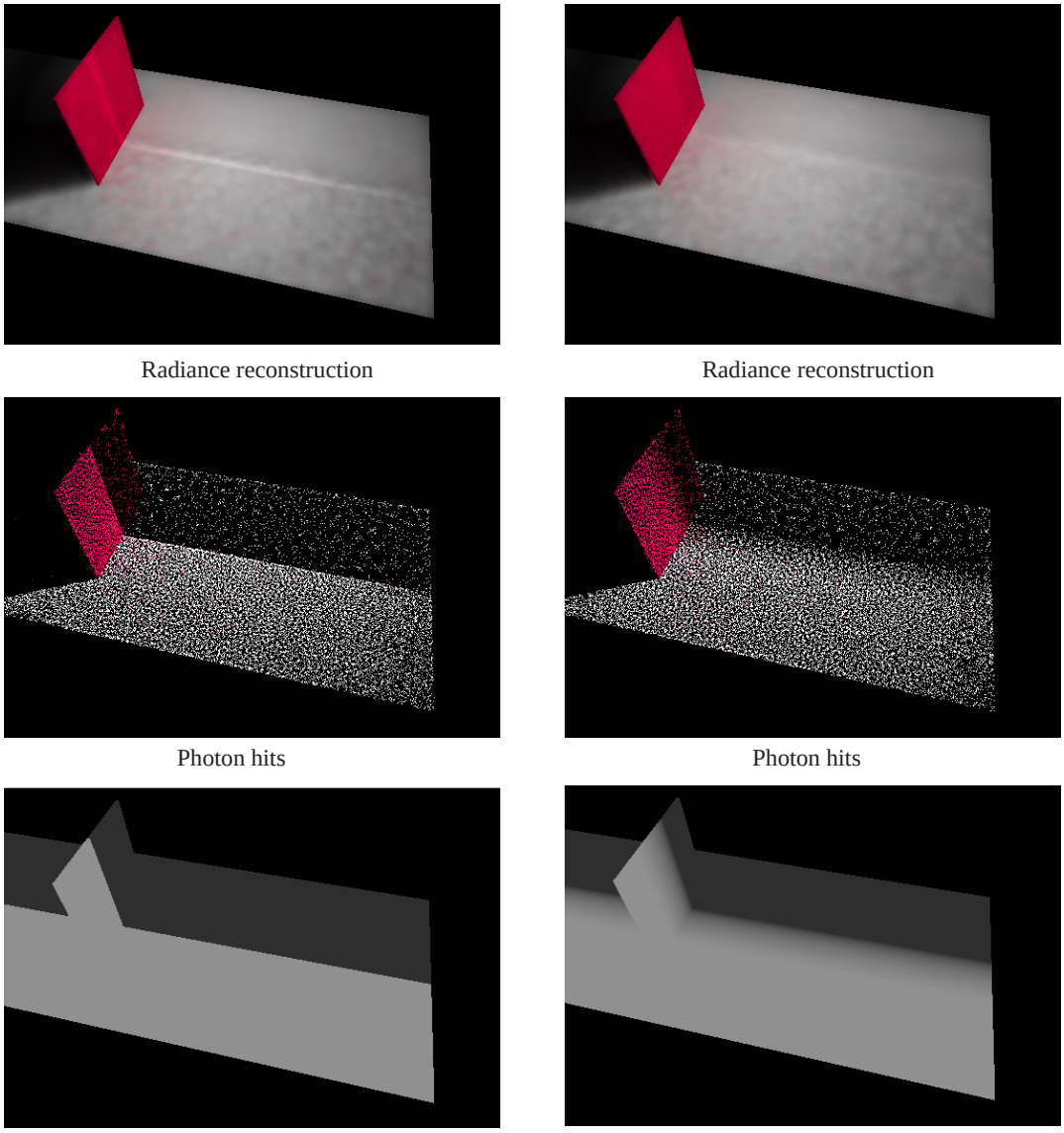


Figure 9.2: Density control with a discontinuity in the target density. The sharper the discontinuity, the stronger the overshooting effect of light leaking into the higher density region.

- Discontinuities in the target density should be avoided. The right column in figure 9.2 shows a similar example, but now the target density changes gradually from high to low. The radiance reconstruction shows a much lower bias (top right).

We have experimented with several alternatives to the uniform redistribution in order to reduce this bias:

Kernel weighted redistribution Similar to using a filter kernel in density estimation, we can use a kernel to weight the power adjustments of the nearest photons. An Epanechnikov kernel results in the following redistribution:

$$\forall i, cos(\omega_i, \mathbf{N}_x) > 0 : \Delta\phi_{i,j} = \phi_j \cdot \frac{\|\mathbf{x}_i - \mathbf{x}\|^2}{S},$$

where S is a normalization factor so that all energy (ϕ_j) is accounted for:

$$S = \sum_{i}^{cos(\omega_i, \mathbf{N}_\mathbf{x})>0} \|\mathbf{x}_i - \mathbf{x}\|^2.$$

We have implemented this strategy, but the bias reduction near the hard discontinuity is small. A photon close to the high density boundary, will find its nearest photons in the high density area, leading to a relatively large kernel evaluation for these photons. As a result, a large part of the power is still added to these boundary photons.

Target density weighted redistribution A better weighted redistribution can be obtained by taking into account the target density of already stored photons. The idea is that in the limit, when the target density is reached everywhere and all new photons are distributed, the target density at each stored photon position also indicates the actual density of the photon map. As shown in figure 9.3 the distribution of the photon power should be inversely proportional to the target density: a larger part of the power should be assigned to the sparser photons where the target density is lower.

The following weighted distribution is used:

$$\forall i, cos(\omega_i, \mathbf{N}_x) > 0 : \Delta\phi_{i,j} = \phi_j \cdot \frac{1}{D_{\text{tar}}(\mathbf{x}_i)} \cdot \frac{1}{S},$$

where S is the normalization factor: $S = \sum_i^{cos(\omega_i, \mathbf{N}_x)>0} \frac{1}{D_{\text{tar}}(\mathbf{x}_i)}$ and $D_{\text{tar}}(\mathbf{x}_i)$ is the target density at the photon position $\mathbf{x}_i$.

In a homogeneous target density region, where all $D_{\text{tar}}(\mathbf{x}_i)$ are equal, the redistribution is uniform over the nearest photons, but near discontinuities the distribution is non-uniform.

Figure 9.4 compares the results using homogeneous and target density weighted distribution. The bias in the high density region is significantly reduced. The variance in the low density region is slightly higher near the discontinuity (some darker spots), because the estimate contains a mixture of low powered photons (high density) and high powered photons (low density).

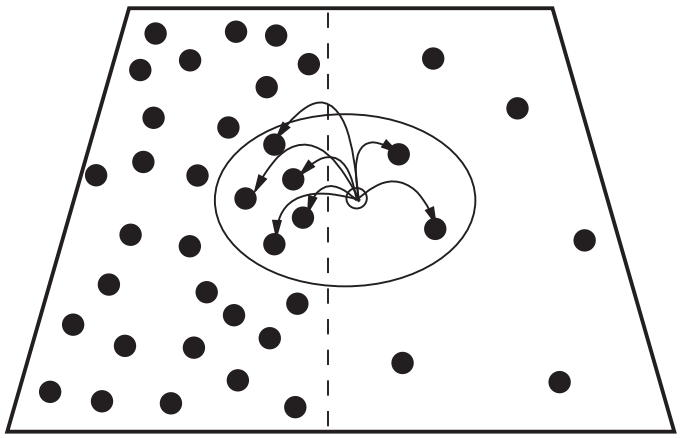
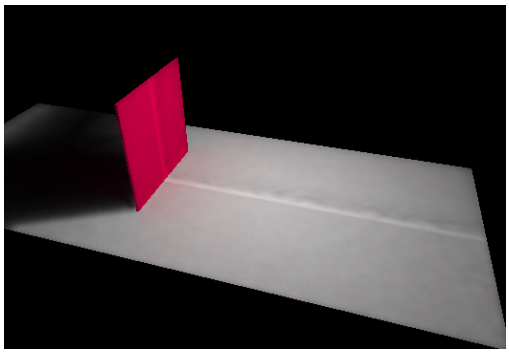
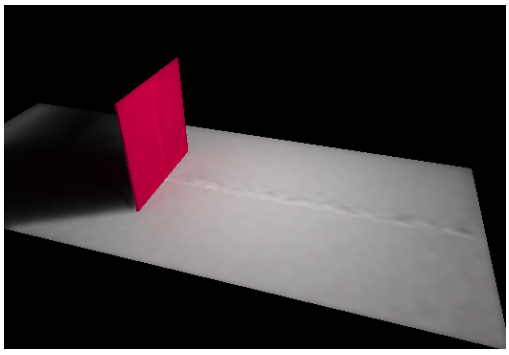


Figure 9.3: Density control with a discontinuity in the target density. The photons in the lower density region should get a bigger portion of the power that is distributed, because the resulting photon power is inversely proportional to the photon density.



Even redistribution



Weighted redistribution

Figure 9.4: Density control with a discontinuity in the target density. Using a target density weighted redistribution reduces the overshooting significantly.

The weighted distribution requires the target density in each stored photon position. Re-evaluating $D_{\text{tar}}(\mathbf{x}_i)$ each time becomes too costly for more advanced target density criteria, such as the importance driven criterion discussed further on. In our implementation, we store the target density with each photon, which requires an extra floating point number in the photon data structure.

9.2.6.3 Photon density discontinuity

Besides target density discontinuities, the density of photons arriving in a region may also contain discontinuities. This happens when the radiance function itself is discontinuous, namely at sharp shadow edges and object boundaries. In the former case the radiance reconstruction results in a blurring of the shadow edges (§8.3.3.4), while the latter case causes boundary bias (§8.3.3.3). Both effects can be reduced by shooting more photons.

Near the discontinuity (within the object or at the bright side of the shadow boundary) most of the nearest photons are located at one side of the reconstruction point, which results in an underestimation of the local photon map density.

With density control, the current density is also estimated by locating the nearest photons, and shows the same underestimation. Since photons are stored until the target density is reached, more photons will be stored near the radiance discontinuities. In turn this will lower the boundary bias, because of the higher density near the boundary.

This is a fortunate side-effect of using the same nearest-photon procedure to estimate the current density and in the actual radiance reconstruction. The effect can be seen at polygon boundaries and shadow edges in the photon hit images in figures 9.1 and 9.2.

The same effect also occurs near target density discontinuities (figure 9.2), but is less beneficial in this case. This also stresses the need for smooth target densities and the weighted distribution.

9.2.7 Conclusion

Density control for photon maps provides an interesting framework for adapting the density of a photon map to a desired target density criterion. Photons that arrive in regions that already have a sufficient photon density, will not be stored anymore. The photon power is redistributed over the neighbors to conserve energy. We have analyzed several redistribution strategies, taking into account possible discontinuities.

The target density limits the maximum number of photons in the photon map. Therefore, one can keep shooting photons until difficult regions have a sufficient density, without worrying about excessive memory usage in other regions.

The resulting photon maps can be used in the same way as normal photon maps. The photon powers are homogeneous, which is necessary for a good reconstruction.

9.3 An importance driven target density criterion

With density control the photon map density can be adapted to an arbitrary target density function. In this section a target density criterion based on importance will be derived.

When a certain view of a scene is rendered, a photon map does not need a constant accuracy throughout the whole scene. For example, a caustic far away does not need to be as detailed as one close to the viewer. Another example is final gathering where a more accurate reconstruction from the global map is required for close-by surfaces. In this section an error analysis of the rendering pass will provide a heuristic for the error tolerated in the photon maps for a particular view (§9.3.1). The error tolerated will then be related to the target density of the photon maps (§9.3.2). Using this target density criterion with the density control from the previous section, we can construct importance driven photon maps.

9.3.1 Error analysis

The pixel flux, computed in the final ray tracing pass, consists of several separate parts: direct illumination, for which the light sources are sampled directly, caustics, using the caustic map, and indirect illumination from the global map.

We are interested in the error in a pixel due to reconstruction errors in both the caustic and the global map. The error analysis is given for the caustic map, but it is similar for the global map.

The caustic map is used for directly visible surfaces and also after one or more specular reflections or refractions. Let the radiance due to caustics be L_c; the contribution to the pixel flux is then defined by all $(\mathsf{L_c S^* E})$ paths. By defining the *caustic importance function*, W_c, as the directional importance function due to these $(\mathsf{S^* E})$ paths, the caustic contribution to the pixel flux is given by

$$\Phi_{\text{pix},c} = \int_{A_s} \int_{\Omega_{2\pi}} W_c(\mathbf{x} \leftarrow \omega) L_c(\mathbf{x} \rightarrow \omega) \; \mathrm{d}_\perp\omega \; \mathrm{d}A \,, \tag{9.1}$$

with A_s the total surface area in the scene

Note that:

- The specific integral (9.1) is not actually computed in the rendering pass, as no explicit representation of W_c is constructed. Instead eye paths are traced and L_c is evaluated for these paths.
- The caustic importance function W_c is defined by the self-emitted directional importance function W_e and the eye paths that are covered $(\mathsf{S^* E})$.

  Recall that for our pinhole camera model, W_e is defined as follows for a certain pixel (see §2.6):

$$W_\mathrm{e}(\mathbf{x} \rightarrow y) = \begin{cases} 1 \cdot \delta(\mathbf{x} - \text{eye}) & \text{when } y \in A_{\text{pix}} \\ 0 \quad \text{otherwise} \end{cases}$$

The importance Γ on a point in a pixel is the integral of the self-emitted directional importance over the aperture: $\Gamma(\mathbf{x}_{\text{pix}}) = 1$. The importance flux emitted through a complete pixel is $\Psi_{\text{pix}} = \int_{A_{\text{pix}}} \Gamma(\mathbf{x}_{\text{pix}})\,\mathrm{d}A = A_{\text{pix}}$. These quantities will be used later on in the error analysis.

The caustic radiance reconstruction from the caustic photon map is only approximate. The reconstruction error ΔL_c causes an error $\Delta\Phi_{\text{pix},c}$ in the pixel flux:

$$\Phi_{\text{pix},c} + \Delta\Phi_{\text{pix},c} = \int_{A_s}\int_{\Omega_{2\pi}} W_c(\mathbf{x} \leftarrow \omega)(L_c(\mathbf{x} \rightarrow \omega) + \Delta L_c(\mathbf{x} \rightarrow \omega))\ \mathrm{d}_\perp\omega\ \mathrm{d}A$$

$$\Rightarrow \Delta\Phi_{\text{pix},c} = \int_{A_s}\int_{\Omega_{2\pi}} W_c(\mathbf{x} \leftarrow \omega)\Delta L_c(\mathbf{x} \rightarrow \omega)\ \mathrm{d}_\perp\omega\ \mathrm{d}A\,.$$

In this error analysis we will assume surfaces where the caustics are stored, to be diffuse (L_c independent of direction), but for storage on glossy materials a directional error estimate might be better. The error can now be rewritten as follows:

$$\Delta\Phi_{\text{pix},c} = \int_{A_s} \Big(\int_{\Omega_{2\pi}} W_c(\mathbf{x} \leftarrow \omega)\ \mathrm{d}_\perp\omega\ \Big)\Delta L_c(\mathbf{x})\ \mathrm{d}A\,.$$

The integral over $\Omega_{2\pi}$ corresponds to the importance Γ_c (equivalent to irradiance) in $\mathbf{x}$:

$$\Delta\Phi_{\text{pix},c} = \int_A \Gamma_c(\mathbf{x})\Delta L_c(\mathbf{x})\ \mathrm{d}A\,. \tag{9.2}$$

This equation formulates the total pixel error in terms of the importance and the reconstruction error. Since we are interested in a heuristic for the reconstruction error itself, the integral still has to be removed.

The portion of the pixel error due to one particular position $\mathbf{x}$, is given by the integrand of equation (9.2):

$$\Delta\Phi_{\text{pix},c}(\mathbf{x}) = \Gamma_c(\mathbf{x})\Delta L_c(\mathbf{x})\,.$$

Bounding this error by a maximum $\Delta\Phi^{\max}_{\text{pix},c}(\mathbf{x})$ gives a bound for the reconstruction error:

$$\Delta L_c(\mathbf{x}) \le \Delta\Phi^{\max}_{\text{pix},c}(\mathbf{x})\ /\ \Gamma_c(\mathbf{x})\,.$$

If each position $\mathbf{x}$ in the scene is allowed to contribute an equal amount to the pixel error, the reconstruction error can be bounded by

$$\Delta L_c(\mathbf{x}) \le \frac{\Delta\Phi^{\max}_{\text{pix},c}}{A_s}\ /\ \Gamma_c(\mathbf{x})\,, \tag{9.3}$$

with $\Delta\Phi^{\max}_{\text{pix},c} = \int_{A_s} \Delta\Phi^{\max}_{\text{pix},c}(\mathbf{x})\ \mathrm{d}A$, and A_s the total area in the scene.

From this analysis we learn that the ratio between the pixel flux error and the reconstruction error is given by the importance. If we can compute Γ_c throughout the scene (which will be accomplished with a caustic importance map in §9.4) and if we can derive a useful relationship between the reconstruction error and the photon map density, then we can use that to tune the storage of photons in the caustic map.

A similar error analysis can be performed for the global map. The difference lies in how the reconstructed radiance L_g from the global map is used in the final pass, namely after a diffuse or moderately

glossy bounce. The contributing paths during rendering are $(\mathsf{L_g(S^*)(D|G)(S|D_{tc})E})$, with $\mathsf{D_{tc}}$ meaning a diffuse or glossy bounce that reaches a surface under the distance threshold (too close-by). This results in a different importance Γ_g over the surfaces and requires a separate global importance map.

9.3.2 Target density criterion

Equation (9.3) relates the reconstruction error to the error made in the pixel flux. The next step is to relate the reconstruction error with the target density of the photon map. This is a very difficult problem. The reconstruction is a form of nearest neighbor density estimation, for which a detailed error analysis is difficult and has only been investigated for very specific cases [98].

9.3.2.1 Error-density relation

We assume a simple inverse linear relationship between the error and the density (D_{cur}):

$$\Delta L_c(\mathbf{x}) = \frac{C}{D_{\text{cur}}(\mathbf{x})}\,, \tag{9.4}$$

where C is the constant of proportionality. When the density increases, the error is assumed to decrease proportionally.

We have obtained good results using this simple, yet adequate heuristic, but finding a better error-density relation is definitely an important area for future research.

9.3.2.2 A practical target density criterion

Both the relation between the reconstruction error and the importance (eq. 9.3) and the relation between the error and the density (eq. 9.4) contain a proportionality constant. A good choice for these constants is not obvious.

Combining the two equations into a direct relationship between the density and the importance provides a more intuitive choice of the constant.

Substituting (9.4) into (9.3) and combining the two constants with the total scene area gives the following relation:

$$D_{\text{cur}}(\mathbf{x}) \geq C'\Gamma_c(\mathbf{x})\,,$$

where $C' = A_s \cdot C/\Delta\Phi^{\max}_{\text{pix},c}$.

In [P9] we presented an intuitive approach to determine C' based on the importance on the image plane:

- The importance of a position x_{scr} on the screen is $\Gamma(x_{\text{scr}}) = 1$. (As mentioned before, this follows from the definition of W_e and the fact that we use a pinhole camera).
- A *target pixel density* is chosen as: M_t/A_{pix}. This density corresponds to a unit importance. The user can choose the number of target photons M_t per pixel. A higher number leads to a denser photon map and a higher accuracy.

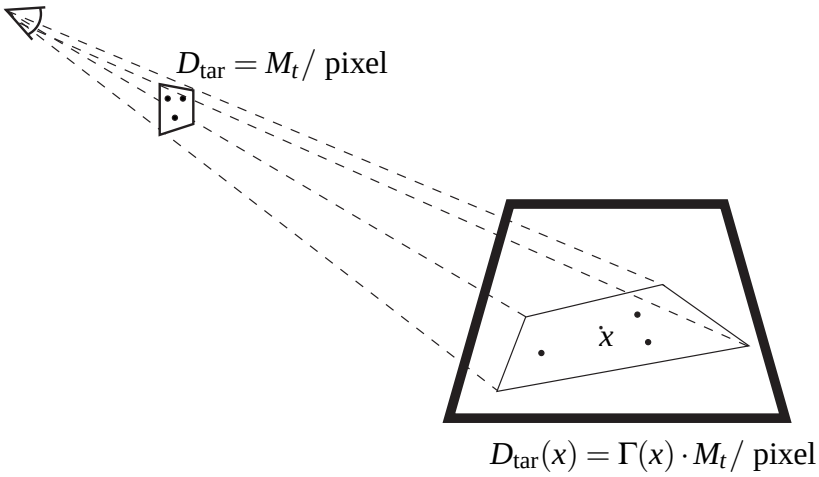


Figure 9.5: The target density is proportional to the importance in the scene. The proportionality constant is defined by choosing a number of required photons per pixel (where the importance is 1), which is then scaled by the importance $\Gamma(\mathbf{x})$ to get the target density in $\mathbf{x}$.

- Given the importance on a position $\mathbf{x}$ in the scene, the target density is set to

$$D_{\text{tar}}(\mathbf{x}) = \Gamma(\mathbf{x}) \times M_t / A_{\text{pix}}\,, \tag{9.5}$$

  which gives the final target density heuristic.

This choice for the constant is best explained by a caustic map example: M_t determines the number of caustic photons we want to be contained in one pixel. The projection of the pixel onto the scene is given by the importance in the scene (on the directly visible surface). The target density in the scene is thus determined by M_t photons in the pixel projection and is given by equation (9.5). This heuristic is illustrated in figure 9.5.

A few remarks about the heuristic:

- Although the accuracy factor M_t still has to be chosen by the user, it is much less dependent on a particular scene than the number of emitted photons that had to be chosen before.
- For the global map, we use values for M_t around 1 or 2, meaning that we want 1 or 2 photons per pixel in the photon map.
- The caustic photon map is visualized directly and needs to be more accurate. We use $M_t = 25$ in our examples.
- The accuracy of the reconstruction is also dependent on the number of nearest photons used. One might make M_t dependent on this number also, for example by choosing M_t as a user-defined fraction of the number of nearest photons used.

One thing remains to be worked out before importance driven photon maps can be constructed: The

importance $\Gamma(\mathbf{x})$ must be known for every position in the scene. This can be done by initially computing importance maps as discussed in the next section.

9.4 Importance maps

To evaluate the target density criterion, the importance must be known for each point in the scene. In this section we will present two different ways to compute importance: one based on *importons*, the adjoint of photons (§9.4.2), and the second one based on path differentials (§9.4.3). To explain the difference between these two approaches, we first need to make the distinction between pixel error and screen error.

9.4.1 Pixel error versus screen error

The target density heuristic (9.5) was derived for a single pixel. Each pixel defines a different importance function and leads to a different target density in the scene.

A combined heuristic that considers all pixels can be defined by choosing an error metric over the different pixels. Two error metrics will be of particular interest: the maximum error that takes the maximum over all pixel errors and the mean error which leads to an average error over the whole screen.

9.4.1.1 Maximum pixel error

Each pixel defines a different importance $\Gamma_{\text{pix}}(\mathbf{x})$ in the scene that determines the impact of the radiance reconstruction error in $\mathbf{x}$ on the error in the pixel flux.

The maximum pixel error metric chooses the target density $D_{\text{tar}}(\mathbf{x})$ by taking the maximum importance in $\mathbf{x}$ over all pixels:

$$\Gamma_{\text{max}}(\mathbf{x}) = \max_{\text{pix}} \left(\Gamma_{\text{pix}}(\mathbf{x})\right) ,$$

and evaluating the heuristic (9.5) with this maximum importance. This error metric ensures that the error in each individual pixel will be bounded, or in other words, that the photon map density in a point will be high enough for the pixel that is most influenced by that point.

9.4.1.2 Screen error

Alternatively, one can also take the average of the different pixel importances to determine the target density:

$$\Gamma_{\text{avg}}(\mathbf{x}) = \frac{1}{N_{\text{pix}}} \sum_{\text{pix}} \Gamma_{\text{pix}}(\mathbf{x}) .$$

The sum of the importance of all pixels, corresponds to the importance for the whole screen at once:

$$\Gamma_{\text{avg}}(\mathbf{x}) = \frac{1}{N_{\text{pix}}} \Gamma_{\text{scr}}(\mathbf{x}) .$$

Using the screen importance to determine the target density, bounds the error through the screen as a whole. Individual pixels, however, still may exhibit a large error.

The advantage of using screen importance is that it is easier to compute. This is why other approaches that use importance, always have used the screen importance [81, 58].

9.4.2 Importons

A first approach for computing importance in the scene is building an importance map by shooting *importons* from the camera into the scene. The construction of an importance map is very similar to the construction of a normal photon map, but *importons* are used instead of photons. Peter and Pietrek [81] were the first to use the importance equivalent of photons to build importance maps.

An importon has the same properties as a photon: a position, a direction and an importance flux, which is equivalent to the power of a photon. An emitted importon is created by sampling a uniform point on a pixel or on the screen. The starting position of the importon is the camera position itself; the direction is given by the normalized vector from the camera to the sampled image point.

The importance of a single importon is derived from the total emitted importance flux and the number of emitted importons. The total emitted importance flux for one pixel is $\Psi_{\text{pix}} = A_{\text{pix}}$. For the whole screen this is $\Psi_{\text{scr}} = A_{\text{scr}}$.

If N importons are shot into the scene, the importance of a single importon i is given by $\Psi_i = \Psi/N$. Scattering of importons (and the corresponding power adjustment) is the same as the scattering of photons. Note that, due to scattering, an emitted importon (an eye path) may result in several stored importons, just as with photons.

Importons are only stored on glossy or diffuse surfaces, since photons will also be stored on these surfaces only. Two importance maps are needed: one for the caustic map and one for the global map. Depending on the history of the importon, it is added to the caustic or global importance map. The importance paths must mimic exactly the paths traced in the final rendering pass:

- If no diffuse or glossy $(\mathsf{D}|\mathsf{G})$ bounce was made before (or if the distance between the bounce and the subsequent hit point is too small), then the global map will *not* be used for radiance reconstruction in the final pass. These importons must be stored in the caustic importance map. For example importons that hit surfaces directly from the eye, corresponding to direct importance, will be stored in this map.

- Once an acceptable $(\mathsf{D}|\mathsf{G})$ bounce is made, the importon is stored in the global importance map, because in the final pass the same eye path would use the global photon map. Note that further scattering of the importon should only include specular bounces, because further $(\mathsf{D}|\mathsf{G})$ bounces will never be generated in the final rendering pass.

During importance map construction, the importons are stored in an array. Afterwards, this array is transformed into a balanced kd-tree for faster access during photon map construction.

Importance reconstruction To evaluate the required density, we need to reconstruct importance from the importance map. The reconstruction of importance corresponds to the reconstruction of *irradiance* from a photon map. The M nearest importons are located and the importance is estimated as follows:

$$\Gamma(\mathbf{x}) \approx \frac{\sum_{i=1}^{M} \Psi_i}{\pi d_M^2(\mathbf{x})}. \tag{9.6}$$

Pixel versus screen importance As mentioned before, both the maximum of the pixel importances or the total screen importance can be used to determine the target density.

The maximum pixel error metric would require a separate importance map for each pixel, because the *maximum* of the reconstructed importances is used to determine the required density at a certain point in the scene. Computing an importance map for each pixel, however, is totally infeasible as it would require emitting quite a number of importons *per pixel*, consuming too much time and memory. Therefore, it is common practice to compute the screen importance and use the screen error metric.

Using the total screen error, a single importance map for the whole screen at once is sufficient. This leads to a single caustic importance map and a single global importance map.

9.4.3 Path differentials

Since bounding each pixel to a maximum error ensures accuracy over the whole image, we have been looking for ways to compute this without requiring an importance map per pixel.

In [P6] we computed a point based importance. Instead of enlarging the pixel to the full screen, it was diminished to an infinitely small area. Importance was computed for individual points on the image. While this gave quite good results for paths of length 2 ($\mathsf{L_g(D|G)E}$ paths), it was hard to generalize to longer paths that include specular bounces.

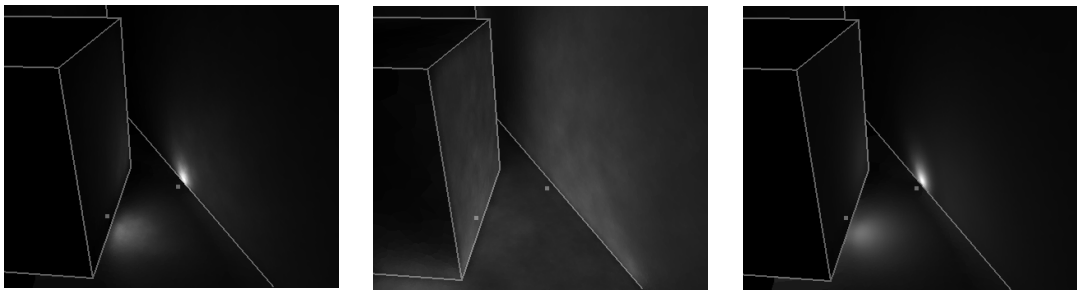
Later we developed path differentials, that were discussed in detail in chapter 7. Path differentials provide a convenient way to estimate the pixel importance for individual eye paths.

Using path differentials, the importance $\Gamma(\mathbf{x})$ for a vertex in a path can estimated as the self-emitted importance flux through a pixel, divided by the area of the footprint $\mathcal{F}_x$:

$$\Gamma(\mathbf{x}) \approx \Gamma_{\text{pd}}(\mathbf{x}) = \frac{\Psi_{\text{pix}}}{A(\mathcal{F}_x)}.$$

For example, the footprint of a path in its image plane vertex $\mathbf{x}_{\text{pix}}$ has an area A_{pix} (see chapter 7). The self-emitted importance flux is also A_{pix}, so that the importance $\Gamma_{\text{pd}}(\mathbf{x}_{\text{pix}}) = 1$ as expected.

The perturbation intervals must all be set to one, as we are only interested in the change of local density of the path. The total self-emitted importance flux is associated with an elementary volume $\mathrm{d}\mathbf{u}$ around



a) Pixel importance b) Screen importance c) Path differential based importance

Figure 9.6: Comparison of pixel and screen importance (for the global photon map): (a) shows the importance for 2 indicated pixels (b) shows the importance for the complete screen. The screen importance underestimates the importance on nearby walls. (c) shows a path based importance computed from path differentials. A close match with (a) is obtained.

a point $\mathbf{u}$ in path space. The footprint indicates the change of path density through the scaling of the elementary intervals $d\mathbf{u}_k$ by the partial derivatives.

Construction The construction of an importance map based on path differentials, proceeds along the same lines as an importons based importance map. Several eye paths are traced through the screen and for each hit point the importance estimate Γ_{pd} is evaluated. Note that the importance is estimated from a single path.

The eye paths are constructed in exactly the same way as for importons. In practice, we store both the importon weight Ψ_i and the path differential importance Γ_{pd} within the same importance map, so that two importance estimates are computed at once.

The importance reconstruction locates the nearest importons i around a point $\mathbf{x}$, and estimates $\Gamma_{\text{pd}}(\mathbf{x})$ as the maximum of $\Gamma_{\text{pd}}(\mathbf{x}_i)$.

9.4.4 Comparison

We have compared the different methods to compute importance in a scene: pixel importance and screen importance (both by computing an importance map) and importance based on path differentials.

Figure 9.6 shows the comparison for the indirect importance that is to be used in the target density of the global photon map. This image is a close-up of the classic Cornell box scene (part of a cube, the floor, and the right wall are visible).

- (a): The pixel importance was computed for two pixels. For each pixel a different importance map was constructed by emitting a different set of importons. The importance in the scene is set to the maximum of the importance over the (few) pixels.

  Abutting surfaces near the projection of the pixel in the scene, show a very high importance. This is logical, because those parts have a large influence on the indirect illumination of the respective pixels. All other areas show a much lower importance.

- (b): The screen importance was computed with a single importance map by shooting importons through the whole screen. The self-emitted importance flux (A_{scr}) is much larger than the emitted importance flux for the other images ($2 \times A_{\text{pix}}$). The importance was scaled to match the emitted importance of the other images.

  The screen importance shows a much lower importance near the pixels of interest, because an average is taken over all pixels. These regions are unimportant for many other pixels, leading to a lower average importance. Regions that are relatively important for many pixels in the image (the cube face and the wall facing each other) show a higher importance.

- (c): The result using the path differential estimate shows a very good match with the actual pixel importance. Because path differentials can estimate the pixel importance for a single path, it is no longer needed to construct an importance map for each pixel separately.

  This shows another interesting application of path differentials. In fact, the need for a path based importance estimate during our work on density control [P6], has initiated our later work on path differentials.

9.4.5 Practical computation of importance maps

9.4.5.1 Path differential importance

The estimate $\Gamma_{\text{pd}}(\mathbf{x})$ is taken as the maximum of the footprint importance over a number of nearest importons. This maximum, however, can cause rather abrupt changes in the target density, for example when an importon with a very high importance is added to or removed from the nearest importons. Such abrupt changes can cause some redistribution bias (§9.2.6.2). Therefore, in practice an average is taken over several nearest pixel importances[1].

Some eye paths that are refracted or reflected specularly may get extremely focused, so that the associated footprint becomes very small. This results in a huge importance and, thus, a huge target density that can never be met. To relieve this problem, the path differential estimate ignores importons with a footprint that is much smaller than the average of all importon footprints. In practice, footprints that are 100 times smaller than the average are ignored (these would require a density a 100 times higher than the average). The average footprint is accumulated during the construction of the importance map.

9.4.5.2 Combined importance

Pixel importance stresses the importance of nearby surfaces for individual pixels, while the screen importance computes an overall importance for the whole image. In our experiments both estimates performed

[1] Note that this averaging does not lead to a screen based importance, because the importons near a point $\mathbf{x}$ are not uniformly distributed over the image plane. Near $\mathbf{x}$ more importons will be located that *are* important in $\mathbf{x}$ (and that will have a smaller footprint).

quite well, thus if an implementation of path differentials is not available, using the screen importance alone also gives adequate results.

In practice, to get the best of both, we usually combine the pixel importance estimate using path differentials with the normal screen importance estimation. We store both footprint and screen importance information in the importon. Importance is estimated using both methods and a weighted average is taken.

The weights are chosen so that the average screen importance matches the average pixel importance:

$$\Gamma(\mathbf{x}) = 0.5(\Gamma_{\text{scr}}(\mathbf{x}) + \Gamma_{\text{pd}}(\mathbf{x})\frac{\Gamma_{\text{scr}}^{(avg)}}{\Gamma_{\text{pd}}^{(avg)}}).$$

The averages are computed when precomputing importances at the importon locations (see §9.4.5.3)

While this is a rather arbitrary combination, it provides a fairly robust estimate for the required density. Nevertheless, using solely the path differentials or even the screen importance, still can give satisfactory results.

9.4.5.3 Precomputation of importance

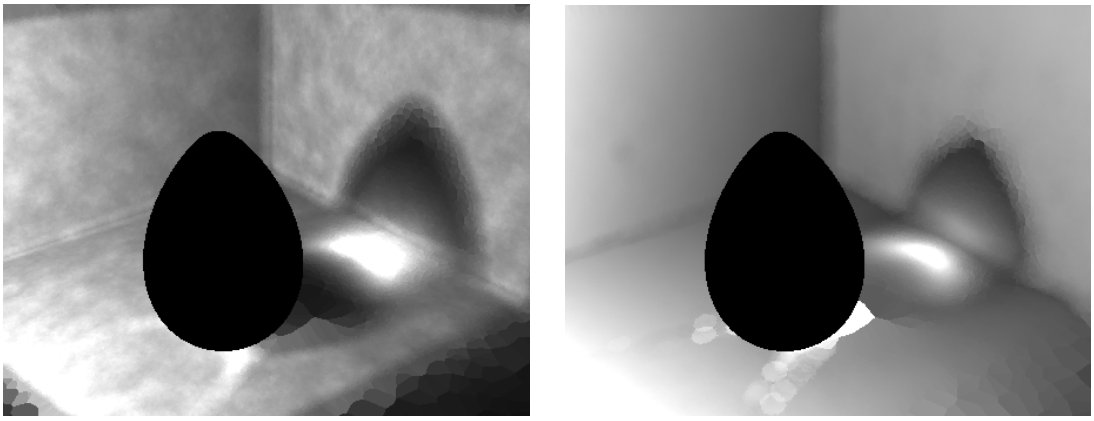
The importance will determine the local density of the photon map. A smoothly varying density is beneficial for the photon map reconstruction and for the redistribution that is used by the density control (§9.2.6.2). This suggests using a large number of nearest importons to get a smoothly changing importance solution. Normally we use around 200 importons in the estimate.

The target density must be evaluated for each photon, whether stored or not. Since locating the 200 nearest importons is expensive, and since the number of photons (= the number of queries) is usually much larger than the number of importons, we precompute importance at the importon locations as proposed by Christensen [20] (see also §8.5.2). However the precomputation is not necessary on *all* importon locations (Christensen suggests 1/4th of the photons/importons). During photon tracing only the precomputed importance is used, and the other importons can be discarded to save memory.

9.4.6 Conclusion: importance maps

We have discussed two methods to compute the importance (and the target density) throughout the scene: one based on the density of importons, the other based on path differentials.

Both estimates can be computed at once by a new initial pass that traces eye paths through the scene. In practice we use a combination of both estimates to evaluate the target density in the scene. Together with density control for photon map construction (§9.2), we now have a practical method for importance driven photon map construction.



(a) (b)

Figure 9.7: Required density of the caustic map (seen from an alternate camera, that was slightly shifted to the right). Note the 'importance caustic' that indicates the region that is magnified by the egg. (a) shows screen importance, (b) shows footprint based importance.

9.5 Results

In this section we will present and analyze several results obtained using importance driven photon map construction. Note that the rendering pass does not have to change when using importance driven photon maps.

We added the importance driven construction of photon maps to our photon map implementation in RENDERPARK [9]. The extra code required for constructing the importance map and the redistribution on top of a normal photon map (and path differential) implementation is limited.

The following results were computed on a 1GHz AMD Athlon PC with 256MB SDR SDRAM memory. We will present results for the caustic map and global map separately.

9.5.1 Caustic map

To test importance driven density control for the caustic map, we used a room with a large glass egg in it. It is lit by two light sources.

Importance maps were computed for the view shown in figure 9.9. The caustic importance map contained 100.000 importons. Importance was precomputed for each importon location using the 200 nearest importons. Tracing the importons took around 3 seconds, precomputation and kd-tree balancing took 15 seconds. The screen importance and footprint importance are computed simultaneously and both are stored in the importon.

For the target density, the accuracy scale factor (M_t) was set to 25. This is higher than for the global maps, because the caustic map is visualized directly and a higher density is needed.

Figure 9.7 shows the required density as seen from an alternate viewpoint. Figure (a) shows the screen importance estimate and (b) the footprint estimate. Some interesting observations can be made:

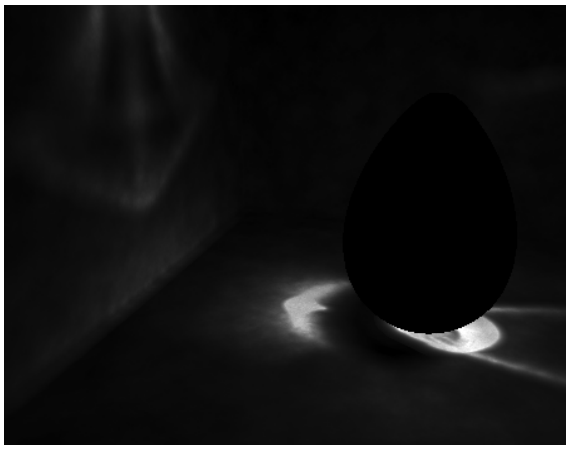


Figure 9.8: Resulting density of the caustic map (seen from the original viewpoint). Bright parts of the caustic are relatively dark because the required density is reached and no extra photons are stored. (200000 stored photons in total)

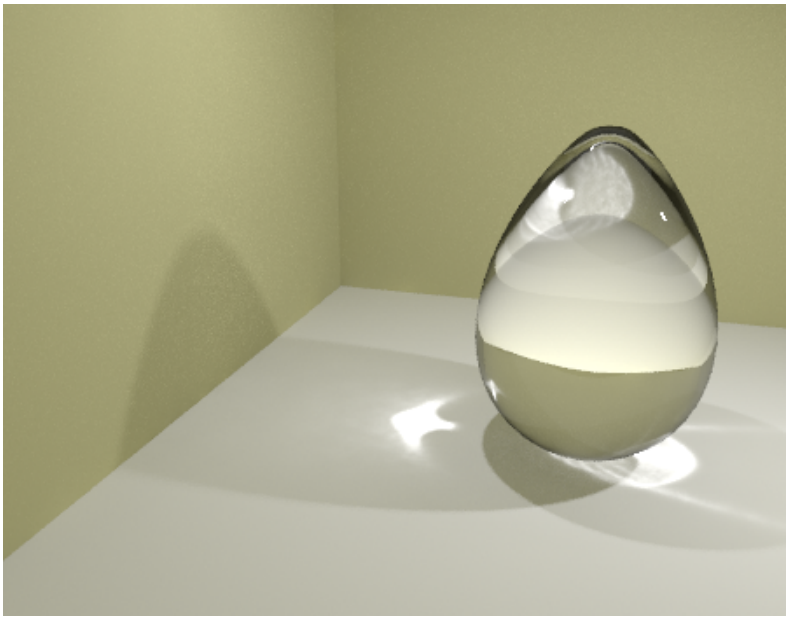


Figure 9.9: Final rendering of the egg scene.

- Importance is focused through the egg on the ground. This results in a sort of importance caustic. This is the part of the scene that is magnified most by the egg as can be seen in the final rendering in figure 9.9.

- The screen importance (a) shows a much higher variance. This is because of density variations in the importance map, that show up in the importance estimate. When using the footprint, each importon carries an importance estimate and the density is not used.

- Some importons had very small footprints that are caused by eye paths that are extremely focused by the glass egg. Although few of these paths occur, the small footprints cause 'spike circles' in the required density estimate. As mentioned in §9.4.5.1 we remove the worst spikes by discarding extremely small footprints.

To compute the caustic photon map, we used the average of screen and footprint importance. Figure 9.8 shows the resulting density of the caustic map. Around 200.000 photons are stored in the map. Without density control 400.000 would have been stored at this point in time. If more photons are shot, the difference grows because more regions reach the target density, and fewer and fewer photons are stored in the caustic map. The gain is the largest in the bright parts of the caustic, where a large fraction of the photons arrive and the target density is reached early on.

Unstored photons were distributed over 20 neighbors. These neighbors are also used to determine the current density.

Tracing and storing the photons took about 140 seconds. This is slower than tracing the 400.000 photons without density control (about 100 seconds), because of a lookup in the importance map (quite fast due to the precomputation) and a lookup in the current photon map to determine the current density (slower as the tree is not yet balanced). Compared to the final rendering time this difference is negligible.

Figure 9.9 shows the final rendering of the scene. Full global illumination is computed using a global photon map. The final rendering took around 80 minutes (without irradiance caching. With irradiance caching, this image can be computed in a few minutes)

Note that, in the end, no photons will be stored anymore when all lit regions have reached the required density. This is an interesting advantage of the redistribution: We can just keep shooting photons until the density in difficult parts of a caustic is sufficient without worrying about the many photons that would be stored in the bright, 'easy' parts. In practice, we end the photon tracing when just a tiny fraction of photons are added as new to the map.

9.5.2 Global map

To show the benefits of importance for the global photon map we use an office scene with several desks and light sources. The camera is looking towards a single desk with a glossy pad and some photo stands. The view can be seen in the final rendered figure 9.13.

Figure 9.10 shows the required density. The average of screen and footprint importance was used with an accuracy scale factor $M_t = 2$. The glossy pad causes a high required density on the photo stands and part of the wall, because the global map is visualized directly after the glossy reflection. 80.000 importons were stored and importance was precomputed for the importon locations using the 200 nearest importons.

Figure 9.11 (a) shows the resulting density of the global map. 57.000 photons are stored, resulting in a fairly good match with the required density. The 20 nearest photons were used for redistribution. Without density control, 400000 photons would have been stored (density shown in figure 9.11 (b)).

Note that some parts did not yet reach the required density. The final rendering, however, shows no artefacts. This indicates that the accuracy could be set even lower.

Figures 9.12 (a) and (b) show a direct visualization of the global map respectively with and without density control. Diffuse irradiance was precomputed on the photon positions (§8.5.2). The 80 nearest photons were used in the radiance estimate.

While the overall illumination is similar, a coarser solution can be seen in unimportant parts of the density controlled image. Note that these parts have a low variance due to the redistribution, but the bias or blurring is higher (e.g., blurry shadow boundaries under the tables).

Figure 9.13 shows the final rendering using the density controlled map. The final rendering using the 400.000 photons is not shown as no visible differences could be seen. The rendering took 90 minutes, which is much more than the time needed for the importance and photon map construction (although irradiance caching would reduce the rendering time significantly).

Note that a fairly open scene was used and that even in such scenes much can be gained by using visual importance. Typical importance examples, such as a maze or a viewer standing in one room of a large building would give even better results. As with all importance driven algorithms, the gain can be arbitrary by choosing a large enough scene, where only a small fraction is visible.

Also note that the memory taken up by the importance maps, can be released when the photon map construction is finished. This free memory can, for example, be used for storing irradiance caching nodes in the rendering pass.

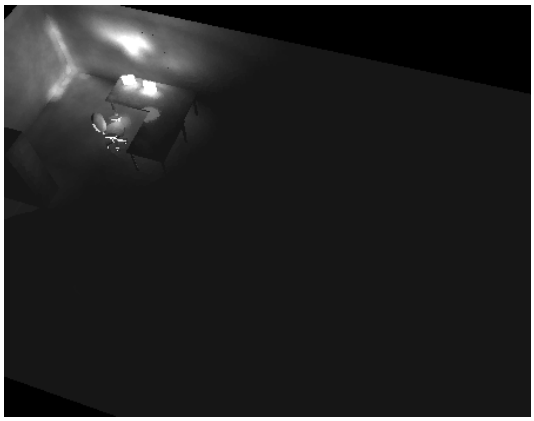
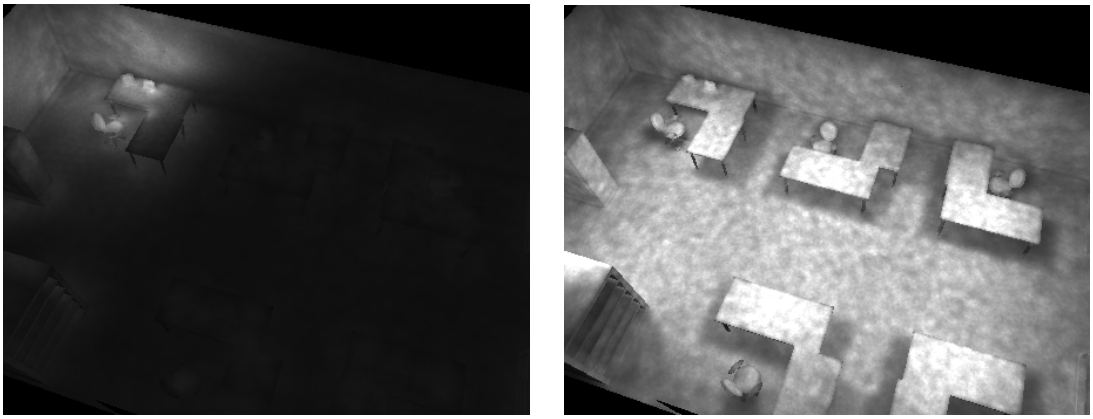
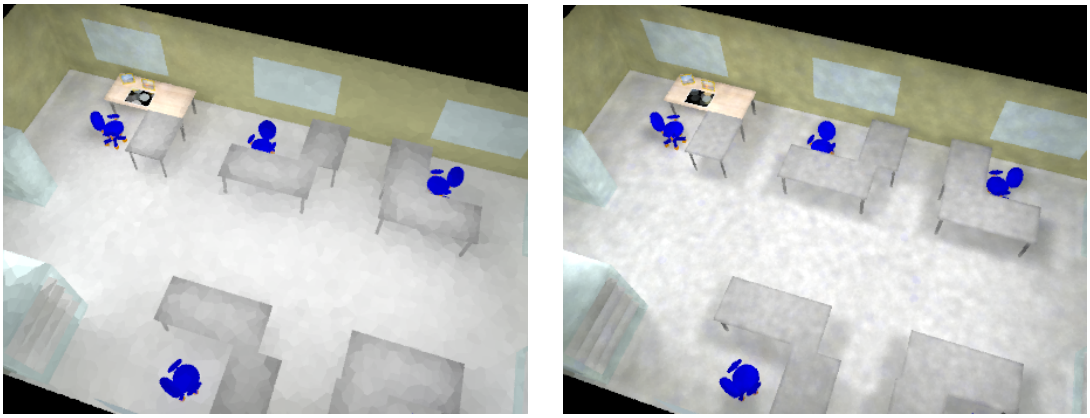


Figure 9.10: Required density of the global map. The glossy pad causes a high required density on the photo stands and part of the wall. Abutting surfaces also require a higher density.



(a) (b)

Figure 9.11: Overview of the resulting density of the global photon map. (a) uses density control, (b) does not.



(a) (b)

Figure 9.12: Direct visualization of the global photon map. This overview shows a coarser solution in unimportant regions when using density control (a, 57000 photons). Overall illumination is of course the same as without density control (b, 400000 photons).

Figure 9.13: Final rendered image of the the office scene. The camera is looking towards one desk in the office.

9.6 A comparison to other approaches for importance driven photon map construction

Other approaches for importance driven photon map construction have been proposed. In this section we will compare these to our method.

Importance has been used in other rendering algorithms, before it was applied to photon maps. The first application of importance in a graphics algorithm was by Smits et al. [102] who used importance to drive the refinement in a hierarchical radiosity algorithm. Christensen [19] extended this principle to a non-diffuse finite element method.

Pattanaik and Mudur [80] applied importance (or potential) to particle tracing radiosity. By estimating importance on light sources, more particles could be directed towards relevant regions in the scene. This idea was further explored by Dutré et al. [29] who also used the estimated importance to drive the sampling of directions on surfaces during particle tracing.

In photon mapping, three other approaches, besides ours, have been presented. All the methods aim at a more efficient construction of the photon map using the importance information.

9.6.1 Peter and Pietrek

In ’98, Peter and Pietrek [81] presented an importance driven three-pass algorithm for photon map construction. In a first pass ‘importons’ are shot from the camera into the scene to construct an importance map. This map is used to gather importance on light sources. In a second pass photons are emitted into the

scene based on the light source importance. When a photon is reflected the importance map is also used to guide the photons towards important regions. The third pass consists of path tracing, where the photon map is only used to construct a pdf for optimized direction sampling, but it is not used directly for illumination reconstruction.

While photons are effectively concentrated in relevant parts of the scene, the method results in a photon map that has a mixture of high and low powered photons. This results in an increased variance when one would reconstruct radiance by locating the nearest photons. The radius of influence of a few high powered photons would be clearly visible in the reconstruction, because they would increase the estimate by a large amount.

Therefore, the number of nearest photons used in the reconstruction would have to be be very high in order to reduce variance. This prevents straight application of the technique to the direct use of a global and caustic map.

9.6.2 Keller and Wald

Keller and Wald [58] propose a different technique to control the density of photon maps:

Photons are emitted as in the standard photon mapping method. When a photon hits a diffuse or glossy surface in $\mathbf{x}$, it is not blindly stored but a discrete storage probability P_Γ is determined based on the (screen) importance in $\mathbf{x}$. A low importance results in a low probability P_Γ and few photons will be stored in this area.

If a photon is stored, the power of the photon is multiplied by $1/P_\Gamma$ to ensure energy conservation. Since P_Γ must be smaller than 1, importance values are scaled down by a certain factor (e.g., the maximum importance) and cut off at 1. The resulting density of the photon map is proportional to the importance multiplied by the irradiance: $\Gamma(\mathbf{x})E(\mathbf{x})$.

This method results in a homogeneous photon map, because the photon powers are scaled with a factor $1/P_\Gamma$, which is dependent on position only. If the importance changes smoothly throughout the scene, then so will the power of the photons.

The biggest drawback of this method, is that the number of photons in the photon map is not limited. A bright area that also has a high importance, will keep storing the many photons that arrive. Clearly, this is not necessary once a certain photon map accuracy is reached.

Our method does limit the number of photons in the photon map, which allows many photons to be shot until less bright, but important regions reach a sufficient density. An additional advantage of our method is that no photons are simply discarded, but are redistributed, which results in a variance reduction due to the averaging of photon powers. This leads to a more smoothly varying radiance reconstruction, especially in less important regions, where the higher bias can be tolerated. A lower variance in the radiance

reconstruction, reduces the number of final gather rays required.

9.6.3 Christensen

Both Keller and Wald's method and ours do not change the way photons are emitted, only the storage method is changed. It would be interesting to also emit photons in important directions, so that less photons need to be traced.

Christensen suggests an adaptation of Peter and Pietrek's method that emits fewer photons, but also keeps the photon powers homogeneous [P9]. An importance map is built first using the importons method.

During photon map construction, the (hemi)sphere around each light source is subdivided in a number of strata. For each stratum a small number of 'feeler' photons (with a high power) is sent out to evaluate the importance of the stratum. If none of the photons reach an important area, the photons are stored.

However, if one of these high powered photons does reach an important area, it should not be stored. Therefore, the stratum is labeled important, and a large number of low powered photons are emitted. Storage of these photons is directed by a storage probability based on importance (similar to Keller and Wald), because it is possible that the low powered photons reach unimportant areas. Instead, our storage method based on redistribution could also be used here, effectively limiting the total number of photons.

Christensen's method to guide more photons to important regions, would be an interesting addition to our density control framework. While it would not really change the accuracy or the amount of photons in the photon maps, it would reduce the time to construct the maps (although this time is much lower than the rendering time, except perhaps for vast, complex scenes).

9.7 Conclusion

In this chapter we presented a framework for controlling the density in photon maps. The framework has two main components: a method to adapt the photon map density to a target density that can be arbitrarily chosen, and a criterion to determine such a target density from an importance solution for a particular view.

The result is a three-pass algorithm. A new initial pass computes an importance solution in the scene. We presented two alternatives to compute importance: the classic method based on 'importons', that are the importance equivalent of photons, and a method based on path differentials. Path differentials allow to estimate pixel importance from a single eye-path, and thus allow to estimate pixel importance in the scene, which would be intractable using the importons method.

The result of the framework is the construction of more memory-efficient photon maps. Far fewer photons need to be stored compared to a standard photon map construction.

An even more important consequence, however, is that the density control framework leads the way to fully error controlled photon maps. In our implementation we have assumed a simple relationship between

the reconstruction error in a photon map and its (target) density. When a better relationship can be derived, for example based on perceptual principles or a better error analysis of nearest neighbor density estimation, our method will allow precise error control while storing just enough photons. This will lead to a photon map construction that does not require the user to choose the number of photons to emit or even the number of nearest photons to use in the reconstruction. Since these two parameters depend on the scene, choosing them currently requires knowledge about the behavior of the illumination in the scene, and an understanding of the internals of the photon mapping algorithm. An automatic procedure to determine these parameters would be preferable.

In its present state, density control relieves the user from choosing the number of photons in the photon map (but not yet from choosing the number of nearest photons). The accuracy parameter M_t in our framework is quite independent of the scene, and is thus easier to determine (the first values we tried for the caustic and global map worked well).

The density control framework also preserves all the advantages of the original photon mapping algorithm: full global illumination is computed and the storage does not require a meshing of the scene.

Due to the limited amount of extra implementation (besides perhaps the optional path differentials), we believe density control is an interesting addition to any existing photon mapping implementation.

Extensions and improvements

Still, many things in the density control framework can be improved and extended. We will highlight a few interesting directions for future research.

- Currently the error due to the caustic or global map reconstruction is estimated independently of any other illumination. Strong direct light for example can mask errors in the caustics or indirect illumination, so that a lower accuracy could be allowed. For example in [P6], we experimented with a very simple convergence heuristic for caustic maps, that takes into account the other illumination in the scene (estimated from the global map).

  One could also take into account surface texture, as illumination errors are less visible on high frequency textures [83].

- The footprint estimate, computed using path differentials, could be used for other purposes:

  – Compute the footprint of photons and distribute them accordingly.

  – It could be used for eye paths in the rendering pass to determine the area over which photons must be considered for illumination reconstruction. If too few photons are found, a secondary final gather can be initiated. This might also help answering the question of how many photons to use in the reconstruction.

- The error analysis and redistribution were both targeted at diffuse illumination. For highly glossy materials a directional target density criterion and redistribution method might deliver better results. A simple method, for example, could be to just increase the target density by a factor based on the glossiness of the material (e.g., multiply the target density by some function of the Phong exponent).

10 Conclusion

This chapter concludes this dissertation. The main topics are summarized in §10.1. Original contributions of this work are given in §10.2 (in a little more detail than in chapter 1). We end this chapter by indicating some interesting directions for future research (§10.3).

10.1 Summary

This dissertation has focused on Monte Carlo techniques and multi-pass algorithms for global illumination computations. After some introduction on global illumination, general Monte Carlo integration and multi-pass methods, we have presented a number of techniques that allow the development of more advanced, robust, and efficient multi-pass algorithms.

A first part of our work was concerned with multi-pass methods in general. Separation of light transport over different algorithms is key in all multi-pass methods. Commonly, parts of the light transport are described by a corresponding regular expression. We have inverted this idea by deriving the path evaluation directly from a user-defined regular expression. With this, separation is easy, and can be tuned depending on the specific multi-pass configuration.

Extreme separation, however, becomes difficult. Small illumination features may have to be separated for optimal performance. This has led to the development of weighted multi-pass methods, that use weighting instead of separation. Weighted multi-pass methods are able to preserve the strengths of separate methods even within overlapping transport. The techniques from the first part were demonstrated using a combination of bidirectional path tracing and radiosity.

A second part in this dissertation deals with path differentials. The tracing of a path corresponds to taking a point sample in path space. While point samples are convenient —they just require tracing infinitely thin rays—, nothing is known about the neighborhood of the path. Path differentials address this problem by computing a region of influence, a footprint, of a path through the computation of partial derivatives. Some work had to be invested in differentiating the sampling procedures and in finding good perturbation interval heuristics, but the resulting footprint estimation is convenient and very useful for many global illumination algorithms. This was demonstrated by several applications: local texture filtering, a refinement oracle for hierarchical particle tracing radiosity, and the computation of importance maps.

A final part of this work looked at photon mapping, an interesting and popular multi-pass method. We have first analyzed the standard photon mapping method and identified some difficulties with the photon map construction: a large memory usage and some scene-dependent parameters (the total number of pho-

tons, the number of nearest photons in the reconstruction), which are not obvious to determine, especially for inexperienced users. We developed a density control framework that addresses some of these difficulties. By redistribution of unstored photons, the actual density can be adapted to any given target density function. A target density criterion that is based on the importance of regions in the scene with respect to the viewer, ensures that all regions store the right amount of photons. The result is that far fewer photons need to be stored and that choosing the number of photons is not as stringent as before.

These three parts together provide several new and general techniques that help the development of better, more robust Monte Carlo methods for multi-pass global illumination.

10.2 Original contributions

This section gives an overview of all the original contributions that were presented in this dissertation. Some contributions have been published before in some of our papers, some others were presented for the first time in this text. The contributions are organized by chapter.

Chapter 5

- The idea and technique to derive the path evaluation directly from the regular expressions that describe the partial light transport is new. It enables a versatile separation of light transport in image-space passes that are based on path sampling. It has led to more advanced multi-pass configurations.
- For the first time, bidirectional path tracing was tightly incorporated into a multi-pass method. This was a direct result of the regular expression based evaluation.

Chapter 6

- A new Monte Carlo variance reduction technique that extends multiple importance sampling was proposed. The technique allows a weighted combination of separate estimators that estimate the same integral, but that use a different amount of precomputed information.
- Provably good heuristics were derived for the weighting functions used in this extension of multiple importance sampling.
- The theory was applied to multi-pass methods. For the first time a weighted multi-pass method was proposed. The weighting automatically preserves the strengths of the different methods that are combined, even within the overlapping transport. A specific weighted combination of bidirectional path tracing and radiosity was presented, that improves on the non-weighted combination.

Chapter 7

Path differentials significantly extend ray differentials as presented by Igehy in [44]. While ray differentials are limited to classical ray tracing only, path differentials can be applied to arbitrary Monte Carlo sampled paths and support area light sources, general BSDFs, and just about anything else for which a sampling procedure can be defined. All global illumination algorithms based on path sampling can benefit from path differentials, considerably extending the field of applications compared to ray differentials.

Specific contributions concerning path differentials are the following:

- A precise definition of the notion of a path footprint was given for sampled paths that can depend on any number of variables. An alternative definition based on convolution was also given, but not further explored.
- A first order Taylor approximation of a footprint was constructed for any vertex or direction in a path; it is based on partial derivatives and differential vectors. A simple procedure that constructs a convenient representation of the footprint, was derived.
- Several heuristics were introduced in order to determine an appropriate neighborhood around a sample point in path space, in order to ensure coherence over the footprint. These heuristics are vital for a good noise versus bias trade-off. A combined heuristic that performs well in all our applications was based on the number of samples, the path gradient, and the second order derivatives.
- An application for local texture filtering in stochastic ray tracing was presented. Noise due to texture variation is reduced by filtering over the footprint.
- A second application presents a refinement oracle for particle tracing radiosity. This is the first refinement oracle that can determine a suitable subdivision level for a single path.

Chapter 8

This chapter describes the standard photon mapping method and several optimizations. Some minor contributions in this chapter are the following:

- An alternative formulation of the radiance reconstruction from photon maps as a standard Monte Carlo estimator.
- A novel heuristic for automatically determining a maximum search radius for nearest photon queries.

Chapter 9

The density control framework made several contributions to the photon mapping method:

- Redistribution of photons was introduced in order to adapt the actual density of photon maps to an arbitrary target density criterion. An important effect of the density control is that the total number of photons in the map is limited.
- An improved redistribution was proposed (compared to our previously published work), that reduces bias in case of discontinuities in the target density.
- From an error analysis of the rendering pass, an importance based target density heuristic was derived. The difference with other importance based photon mapping methods is that this heuristic specifies an absolute target density in terms of a single accuracy parameter.
- A new method to estimate importance throughout the scene was (easily) derived from path differentials. Importance can be estimated from a single path, allowing for a pixel-based importance estimate instead of the common screen-based importance, that has to consider the average error through the whole screen at once in order to be practical.

The resulting three-pass importance driven photon mapping algorithm results in far fewer stored photons, and introduces some level of automatic error control in photon mapping.

RenderPark

During the research conducted for this dissertation, all the techniques have been implemented in RENDERPARK [9]. The source code of RENDERPARK is freely available to the rendering community, and it is being used by other people for research and educational purposes. Therefore, RENDERPARK in itself is definitely an important contribution of this dissertation.

10.3 Directions for future research

From all the previous global illumination research it has become apparent that the most efficient and robust algorithms combine the strengths of several different techniques. This is obvious in multi-pass methods, as demonstrated, for example, in the intelligent combination of algorithms and optimizations in photon mapping. It is unlikely that a new single-pass algorithm will replace all available algorithms, optimization techniques, and their combinations, to form the ultimate global illumination algorithm.

Therefore, in this work, we have been looking for general techniques to combine and improve the wealth of algorithms available. The most interesting directions for future research, with respect to the work in this dissertation, would be the development of improved techniques, and the application and combination of the techniques in more global illumination algorithms.

10.3.1 Improved techniques

Although the techniques we presented can be and were all used in practice, several of them still have room for further improvement. Most of the extensions and improvements have been mentioned in the respective chapters. Here, we will just repeat a few that we consider the most important:

- The flexibility and wide application of path differentials invites more research to improve them further. Future research could focus on improved heuristics for the perturbation intervals to ensure a better coherence (e.g., fully integrated second order derivatives, selective visibility tests), on non-constant footprints (convolution footprint, higher order gradients), and on efficiency (reducing the number of differential vectors before scattering).

- Density control for photon maps could benefit from a directionally dependent redistribution and a directional target density criterion to better handle glossy surfaces. Also, a better relation between the target density and the reconstruction error should be derived, perhaps based on perceptual issues or on a detailed error analysis of nearest neighbor density estimation.

10.3.2 More applications and combinations

Most of the techniques we presented are applicable to more global illumination algorithms than we have demonstrated. Therefore, we foresee that they will find application in other algorithms.

The weighted multi-pass theory can be applied to other multi-pass methods. This is, however, not an easy task because the separation and different sampling techniques must be analyzed carefully, and the derivation of the weighting heuristics is non-trivial. The application that we presented nevertheless shows that it is indeed possible and that the benefits are substantial. It would be interesting to apply the weighting to the photon mapping method, which could result in an automatic separation of light transport into a directly and indirectly visualized photon map, instead of the current fixed separation that only stores caustics in the directly visualized map.

Path differentials form a really practical tool. The applications that we have developed with path differentials were easy to develop and to implement. More applications await, some of which we indicated in the conclusions of chapter 7. Again, photon mapping would be a candidate for further application of path differentials, not only in the initial, importance computation pass but also during photon map construction and rendering.

There are also other interesting techniques and tools that could easily be combined with our methods. For example, quasi-Monte Carlo methods exhibit a superior performance over the standard Monte Carlo sampling that we have used. Adaptive sampling, a parallel implementation, or coherent ray tracing can give very good speed-ups. Adding these techniques is not difficult and they would further increase the efficiency

of our methods.

10.3.3 Error control

A more general direction for future research is the development of accurate error control for global illumination algorithms such as presented in this work. Weighted multi-pass methods, path differentials, as well as density control for photon maps, they all provide a framework where the error can be controlled. We have used heuristics to minimize the error or to trade bias for noise. While they were shown to work well, they still are heuristics. Accurate error estimates and error bounds would not only give better solutions and heuristics, but would also tell us exactly how good our solutions are. Most of the estimators we presented are consistent (e.g., path differential applications and photon maps) for which error estimation is rather difficult. Full error control will not be obtained easily, but it is very promising.

Recent work on error control based on perceptual principles might also contribute to this goal.

The ultimate rendering algorithm

The goal of any global illumination researcher is to find the ultimate rendering algorithm, that computes images quickly and accurately for any imaginable scene.

I am no different, and I've been haunted in my dreams by visions of a real-time fully error controlled weighted multi-pass photon mapping method in which path differentials perfectly estimate all the necessary parameters. I still think this is a good way to go, and this work has taken some steps in this direction. Should 'the perfect algorithm' be found, I believe it will consist of a combination of different techniques that each have stood the test of time. We're not there yet, but a Ph.D. has to finish at some point in time. For me, that point in time being now…

Publications

[P1] Philip Dutré, Philippe Bekaert, Frank Suykens and Yves D. Willems. “Bidirectional Radiosity” In J. Dorsey and Ph. Slusallek, eds., *Rendering Techniques ’97 (Proceedings of the 8th Eurographics Workshop on Rendering)*, St.-Etienne, France, pages 205–216, Springer-Verlag, June 1997.

[P2] Philip Dutré, Frank Suykens and Yves D. Willems “Optimized Monte Carlo Path Generation Using Genetic Algorithms” Technical Report CW267, Department of Computer Science, K.U. Leuven, May 1998. 14 pages.

[P3] Frank Suykens and Yves D. Willems. “Combining Bidirectional Path Tracing and Multipass Rendering” In V.Skala, editor, *Seventh International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG ’99)*, pages 265–272, Plzen, Czech Republic, Februari 1999.

[P4] Frank Suykens and Yves D. Willems. “Weighted Multipass Methods for Global Illumination” In *Computer Graphics Forum* 18(3), pages C-209–C-220, Blackwell Publishers, UK, September 1999.

[P5] Frank Suykens and Yves D. Willems. “Adaptive Filtering for Progressive Monte Carlo Image Rendering” In V.Skala, editor, *Eighth International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG 2000)*, pages 220–227, Plzen, Czech Republic, Februari 2000.

[P6] Frank Suykens and Yves D. Willems. “Density Control for Photon Maps” In B. Peroche and H. Rushmeier, eds., *Rendering Techniques 2000 (Proceedings of the 11th Eurographics Workshop on Rendering)*, Brno, Czech Republic, pages 23–34, Springer-Verlag, June 2000.

[P7] Frank Suykens and Yves D. Willems “Path Differentials and Applications” Technical Report CW307, Department of Computer Science, K.U. Leuven, May 2001. 21 pages.

[P8] Frank Suykens and Yves D. Willems. “Path differentials and Applications” In S.J. Gortler and K. Myszkowski, eds., *Rendering Techniques 2001 (Proceedings of the 12th Eurographics Workshop on Rendering)*, London, UK, pages 257–268, Springer-Verlag, June 2001.

[P9] Henrik Wann Jensen, Per H. Christensen and Frank Suykens. “A Practical Guide to Global Illumination Using Photon Mapping” *SIGGRAPH 2001 Course Notes, Course 38*, Los Angeles, USA, 140 pages, ACM SIGGRAPH, August 2001.

[P10] Henrik Wann Jensen, Per H. Christensen, Toshi Kato and Frank Suykens. “A Practical Guide to Global Illumination Using Photon Mapping” *SIGGRAPH 2002 Course Notes, Course 43*, San Antonio, USA, 205 pages, ACM SIGGRAPH, July 2002.

Bibliography

[1] John Amanatides. Ray Tracing with Cones. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, volume 18, pages 129–135, July 1984.

[2] Görgy Antal, László Szirmay-Kalos, and Ferenc Csonka. Weighted Multi-pass Method Based on Stochastic Iteration and Random Walk Methods. In *10th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG 2002)*, Plzen, Czech Republic, February 2002. University of West Bohemia.

[3] James R. Arvo. Backward Ray Tracing. In *ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, volume 12, August 1986.

[4] James R. Arvo. *Analytic Methods for Simulated Light Transport*. Ph.D. thesis, Yale University, New Haven, CT, December 1995.

[5] Philippe Bekaert. *Hierarchical and stochastic algorithms for radiosity*. PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, December 1999. 260+xi pages.

[6] Philippe Bekaert. *Personal Communication*. 2001.

[7] Philippe Bekaert, Philip Dutré, and Yves D. Willems. Final radiosity gather step using a Monte Carlo technique with optimal importance sampling. Report CW 275, Department of Computer Science, K.U.Leuven, Leuven, Belgium, November 1998.

[8] Philippe Bekaert, László Neumann, Attila Neumann, Mateu Sbert, and Yves D. Willems. Hierarchical Monte Carlo Radiosity. In George Drettakis and Nelson Max, editors, *Rendering Techniques '98 (Proceedings of the 9th Eurographics Workshop on Rendering)*, pages 259–268. Springer-Verlag Wien New York, 1998.

[9] Philippe Bekaert and Frank Suykens. *RenderPark, a physically based rendering tool*. K.U. Leuven, Belgium, http://www.renderpark.be, 1996-2002.

[10] Philippe Bekaert and Yves D. Willems. Error Control for Radiosity. In X. Pueyo and P. Shröder, editors, *Rendering Techniques '96 (Proceedings of the 7th Eurographics Workshop on Rendering)*, pages 153–164. Springer-Verlag Wien New York, June 1996.

[11] Jon L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[12] Mark A. Bolin and Gary W. Meyer. An Error Metric for Monte Carlo Ray Tracing. In Julie Dorsey and Philipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the 8th Eurographics Workshop on Rendering)*, pages 57–68. Springer Wien New York, June 1997.

[13] Nikolai P. Buslenko, D. I. Golenko, Yu. A. Shreider, Ilya M. Sobol, and V. G. Sragovich. *The Monte Carlo Method – The Method of Statistical Trials*. Pergamon Press, 1962. Translated from the Russian by G. J. Tee, 1966.

[14] Rachel Chadwell, Raphael Compagnon, Chas Ehrlich, Danny Fuller, and Greg Ward Larson. *Radiance*. Berkeley Lab, California, USA, http://radsite.lbl.gov/radiance/HOME.html.

[15] Min Chen. Perturbation Methods for Image Synthesis. M.Sc. thesis, Computer Graphics Group, California Institute of Technology, Pasadena, CA, June 1999.

[16] Min Chen and James R. Arvo. Perturbation Methods for Interactive Specular Reflections. In Hans Hagen, editor, *IEEE Transactions on Visualization and Computer Graphics*, volume 6 (3), pages 253–264. IEEE Computer Society, 2000.

[17] Min Chen and James R. Arvo. Theory and application of specular path perturbation. In *ACM Transactions on Graphics*, volume 19(4), pages 246–278. ACM Press, 2000.

[18] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. A Progressive Multi-Pass Method for Global Illumination. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25, pages 164–174, July 1991.

[19] Per H. Christensen. *Hierarchical Techniques for Glossy Global Illumination*. PhD thesis, Department of Computer Science and Engineering, University of Washington, Seattle, Washington, 1995.

[20] Per H. Christensen. Faster Photon Map Global Illumination. *Journal of Graphics Tools: JGT*, 4(3):1–10, 1999.

[21] Per H. Christensen, Eric J. Stollnitz, and David H. Salesin. Global Illumination of Glossy Environments Using Wavelets and Importance. *ACM Transactions on Graphics*, 15(1):37–71, January 1996.

[22] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993.

[23] Steven Collins. Adaptive Splatting for Specular to Diffuse Light Transport. In *Photorealistic Rendering Techniques (Proceedings of the 5th Eurographics Workshop on Rendering)*, pages 119–135, Darmstadt, Germany, June 1994. Springer-Verlag Wien New York.

[24] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed Ray Tracing. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, volume 18, pages 137–145, July 1984.

[25] Mark A. Z. Dippe and Erling Henry Wold. Stochastic Sampling: Theory and Application. In George W. Zobrist, editor, *Progress in Computer Graphics*. Ablex Publishing, Norwood, NJ, 1991.

[26] Frédo Durand, George Drettakis, and Claude Puech. The Visibility Skeleton: A Powerful and Efficient Multi-Purpose Global Visibility Tool. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings*, volume 31, pages 89–100, August 1997.

[27] Philip Dutré. *Mathematical Frameworks and Monte Carlo Algorithms for Global Illumination in Computer Graphics*. PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium, September 1996.

[28] Philip Dutré, Eric P. Lafortune, and Yves D. Willems. Monte Carlo Light Tracing with Direct Pixel Contributions. In H. P. Santo, editor, *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 128–137, Alvor, Portugal, December 1993.

[29] Philip Dutré and Yves D. Willems. Potential-Driven Monte Carlo Particle Tracing for Diffuse Environments with Adaptive Probability Density Functions. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the 6th Eurographics Workshop on Rendering)*, pages 306–315. Springer-Verlag Wien New York, 1995.

[30] Gershon Elber. Low Cost Illumination Computation using an Approximation of Light Wavefronts. In *Computer Graphics (ACM SIGGRAPH '94 Proceedings)*, volume 28, pages 335–342, July 1994.

[31] Jon D. Genetti, Dan Gordon, and Glen N. Williams. Adaptive Supersampling in Object Space Using Pyramidal Rays. *Computer Graphics Forum*, 17(1):29–54, 1998. ISSN 1067-7055.

[32] Andrew S. Glassner. A Model for Fluorescence and Phosphorescence. In Peter Shirley, Georgios Sakas, and Stefan Müller, editors, *Photorealistic Rendering Techniques (Proceedings of the 5th Eurographics Workshop on Rendering)*, pages 60–70. Springer-Verlag Wien New York, 1994.

[33] Ned Greene and Paul S. Heckbert. Creating Raster Omnimax Images From Multiple Perspective Views Using the Elliptical Weighted Average Filter. *IEEE Computer Graphics and Applications*, 6(6):21–27, June 1986.

[34] John M. Hammerlsey and David C. Handscomb. *The Monte Carlo Method*. Cambridge University Press, 1964.

[35] Pat Hanrahan, David Salzman, and Larry Aupperle. A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.

[36] Xiao D. He, Kenneth E. Torrance, Francois X. Sillion, and Donald P. Greenberg. A Comprehensive Physical Model for Light Reflection. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25, pages 175–186, July 1991.

[37] Paul S. Heckbert. Texture Mapping Polygons in Perspective. TM 13, NYIT Computer Graphics Lab, April 1983.

[38] Paul S. Heckbert. Survey of Texture Mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.

[39] Paul S. Heckbert. Fundamentals of Texture Mapping and Image Warping. M.Sc. thesis, UCB/CSD 89/516, CS Division, U.C. Berkeley, Berkeley, CA, June 1989.

[40] Paul S. Heckbert. Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, volume 24, pages 145–154, August 1990.

[41] Paul S. Heckbert. Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, volume 24, pages 145–154, August 1990.

[42] Paul S. Heckbert and Pat Hanrahan. Beam Tracing Polygonal Objects. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, volume 18, pages 119–127, July 1984.

[43] Homan Igehy. *Addendum to Tracing Ray Differentials*. Stanford University, http://graphics.stanford.edu/papers/trd/addendum.html, 1999.

[44] Homan Igehy. Tracing ray differentials. In *Computer Graphics (ACM SIGGRAPH '99 Proceedings)*, volume 33, pages 179–186, 1999.

[45] Dave S. Immel, Michael F. Cohen, and Donald P. Greenberg. A Radiosity Method for Non-Diffuse Environments. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, volume 20, pages 133–142, August 1986.

[46] Henrik Wann Jensen. Importance Driven Path Tracing Using the Photon Map. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 326–335. Springer-Verlag Wien New York, 1995.

[47] Henrik Wann Jensen. Global Illumination Using Photon Maps. In X. Pueyo and P. Scröder, editors, *Rendering Techniques '96 (Proceedings of the 7th Eurographics Workshop on Rendering)*, pages 21–30. Springer-Verlag, Wien New York, 1996.

[48] Henrik Wann Jensen. Rendering Caustics on Non-Lambertian Surfaces. In *Proceedings of Graphics Interface '96*, pages 116–121, San Francisco, CA, May 1996. Morgan Kaufmann.

[49] Henrik Wann Jensen. A Practical Guide to Global Illumination Using Photon Maps. In *SIGGRAPH 2000 Course Notes*. Association for Computing Machinery, ACM SIGGRAPH, August 2000. Course 8.

[50] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Natick, MA, 2001.

[51] Henrik Wann Jensen and Niels J. Christensen. Efficiently Rendering Shadows Using the Photon Map. In Harold P. Santo, editor, *Edugraphics + Compugraphics Proceedings*, pages 285–291, Portugal, December 1995. GRASP- Graphic Science Promotions and Publications.

[52] Henrik Wann Jensen and Niels J. Christensen. Photon Maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects. *Computers & Graphics*, 19(2):215–224, 1995.

[53] Henrik Wann Jensen and Per H. Christensen. Efficient Simulation of Light Transport in Scenes with Participating Media Using Photon Maps. In *Computer Graphics (ACM SIGGRAPH '98 Proceedings)*, volume 32, pages 311–320, July 1998.

[54] James T. Kajiya. The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, August 1986.

[55] Malvin H. Kalos and Paula A. Whitlock. *Monte Carlo methods, Volume I: Basics*. J. Wiley and sons, 1986.

[56] Alexander Keller. Instant Radiosity. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 49–56, August 1997.

[57] Alexander Keller. *Quasi-Monte Carlo Methods for Photorealistic Image Synthesis*. PhD thesis, University of Kaiserlautern, June 1997. ISBN 3-8265-3330-5.

[58] Alexander Keller and Ingo Wald. Efficient Importance Sampling Techniques for the Photon Map. In *Proceedings of Vision, Modelling and Visualization 2000*, pages 271–279, Saarbruecken, Germany, November 2000.

[59] Craig Kolb, Pat Hanrahan, and Don Mitchell. A Realistic Camera Model for Computer Graphics. In *Computer Graphics (ACM SIGGRAPH 95 Proceedings)*, pages 317–324, August 1995.

[60] Eric P. Lafortune. *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*. Ph.D. thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, February 1996.

[61] Eric P. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-Linear Approximation of Reflectance Functions. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 117–126, 1997.

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

[63] Eric P. Lafortune and Yves D. Willems. A Theoretical Framework for Physically Based Rendering. *Computer Graphics Forum*, 13(2):97–107, June 1994.

[64] Eric P. Lafortune and Yves D. Willems. Using the Modified Phong BRDF for Physically Based Rendering. Technical Report CW197, Department of Computer Science, K.U.Leuven, Leuven, Belgium, November 1994.

[65] Eric P. Lafortune and Yves D. Willems. A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 11–20, New York, NY, 1995. Springer-Verlag.

[66] Eric P. Lafortune and Yves D. Willems. Reducing the Number of Shadow Rays in Bidirectional Path Tracing. In V. Skala, editor, *Proceedings of the Winter School of Computer Graphics and CAD Systems '95*, pages 384–392, Plzen, Czech Republic, February 1995. University of West Bohemia.

[67] Greg Ward Larson and Rob Shakespeare. *Rendering with Radiance: The Art and Science of Lighting Visualization*. Morgan Kaufmann, San Francisco, CA, 1998. ISBN 1-55860-499-5.

[68] Mark E. Lee, Richard A. Redner, and Samuel P. Uselton. Statistically Optimized Sampling for Distributed Ray Tracing. In *Computer Graphics (ACM SIGGRAPH '85 Proceedings)*, volume 19, pages 61–67, July 1985.

[69] Robert R. Lewis. Making Shaders More Physically Plausible. In Michael Cohen, Claude Puech, and Franǫis Sillion, editors, *Fourth Eurographics Workshop on Rendering*, pages 47–62, Paris, France, June 1993.

[70] Daniel Lischinski, Filippo Tampieri, and Donald P. Greenberg. Combining Hierarchical Radiosity and Discontinuity Meshing. In *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, volume 27, pages 199–208, 1993.

[71] Joachim Loos, Philipp Slusallek, and Hans-Peter Seidel. Using Wavefront Tracing for the Visualization and Optimization of Progressive Lenses. In David Duke, Sabine Coquillart, and Toby Howard, editors, *Computer Graphics Forum*, volume 17(3), pages 255–265. Eurographics Association, 1998.

[72] mental images. *Mental Ray*. Germany, http://www.mentalimages.com.

[73] Nicholas C. Metropolis, Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

[74] Don Mitchell and Pat Hanrahan. Illumination from curved reflectors. In *Computer Graphics (ACM SIGGRAPH '92 Proceedings)*, volume 26, pages 283–291, July 1992.

[75] Karol Myszkowski. Lighting Reconstruction Using Fast and Adaptive Density Estimation Techniques. In Julie Dorsey and Philipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the 8th Eurographics Workshop on Rendering)*, pages 251–262. Springer Wien New York, 1997.

[76] Art B. Owen. Orthogonal Arrays for Computer Experiments, Integration and Visualization. *Statistica Sinica*, 2(2):439–452, July 1992.

[77] Art B. Owen and Yi Zhou. Safe and effective importance sampling. Technical report, Stanford University, March 1999. http://www-stat.stanford.edu/ owen/reports/.

[78] Sumanta N. Pattanaik. *Computational Methods for Global Illumination and Visualisation of Complex 3D Environments*. Ph.D. thesis, Birla Institute of Technology and Science, Computer Science Department, Pilani, India, February 1993.

[79] Sumanta N. Pattanaik and Sudhir P. Mudur. Computation of Global Illumination by Monte Carlo Simulation of the Particle Model of Light. In Alan Chalmers and Derek Paddon, editors, *Third Eurographics Workshop on Rendering*, pages 71–83, Bristol, UK, May 1992.

[80] Sumanta N. Pattanaik and Sudhir P. Mudur. Adjoint Equations and Random Walks for Illumination Computation. *ACM Transactions on Graphics*, 14(1):77–102, January 1995.

[81] Ingmar Peter and Georg Pietrek. Importance Driven Construction of Photon Maps. In G. Drettakis and N. Max, editors, *Rendering Techniques '98 (Proceedings of the 9th Eurographics Workshop on Rendering)*, pages 269–280. Springer Wien New York, 1998.

[82] Jackson Pope and Alan Chalmers. Improving the Hierarchical Stochastic Radiosity Algorithm. In *Eighth International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG 2000)*, pages 244–251, Plzen, Czech Republic, February 2000. University of West Bohemia.

[83] Mahesh Ramasubramanian, Sumanta N. Pattanaik, and Donald P. Greenberg. A Perceptually Based Physical Error Metric for Realistic Image Synthesis. In *Computer Graphics (ACM SIGGRAPH '99 Proceedings)*, volume 33, pages 73–82, August 1999.

[84] Holly E. Rushmeier, Charles Patterson, and Aravindan Veerasamy. Geometric Simplification for Indirect Illumination Calculations. In *Proceedings of Graphics Interface '93*, pages 227–236, San Francisco, CA, May 1993. Morgan Kaufmann.

[85] Holly E. Rushmeier and Kenneth E. Torrance. The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium. In *Computer Graphics (ACM SIGGRAPH '87 Proceedings)*, volume 21, pages 293–302, July 1987.

[86] Satunino L. Salas and Einar Hille. *Calculus, One and Several Variables, 6th edition*. John Wiley and Sons, New York, NY, 1990.

[87] Stephan Schaefer. Hierarchical Radiosity on Curved Surfaces. In Julie Dorsey and Philipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 187–192. Springer Wien New York, 1997.

[88] Annette Scheel, Marc Stamminger, and Hans-Peter Seidel. Thrifty Final Gather Radiosity. In S.J. Gortler and K. Myszkowski, editors, *Rendering Techniques 2001 (Proceedings of the Twelfth Eurographics Workshop on Rendering)*. Springer Wien New York, 2001.

[89] Andreas Schilling, Günter Knittel, and Wolfgang Strasser. Texram: a Smart Memory for Texturing. *IEEE Computer Graphics and Applications*, 16(3):32–41, May 1996.

[90] Mikio Shinya, Tokiichiro Takahashi, and Seiichiro Naito. Principles and Applications of Pencil Tracing. In *Computer Graphics (ACM SIGGRAPH '87 Proceedings)*, volume 21, pages 45–54, July 1987.

[91] Peter Shirley. A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes. In *Proceedings of Graphics Interface '90*, pages 205–212, San Francisco, CA, May 1990. Morgan Kaufmann.

[92] Peter Shirley. Discrepancy as a Quality Measure for Sample Distributions. In *Proceedings of Eurographics '91*, pages 183–94. Elsevier Science Publishers, Amsterdam, North-Holland, September 1991.

[93] Peter Shirley, Bretton Wade, Philip M. Hubbard, David Zareski, Bruce Walter, and Donald P. Greenberg. Global Illumination via Density Estimation. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 219–230. Springer-Verlag Wien New York, 1995.

[94] Peter Shirley, Chang Yaw Wang, and Kurt Zimmerman. Monte Carlo Techniques for Direct Lighting Calculations. *ACM Transactions on Graphics*, 15(1):1–36, January 1996.

[95] Francois Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. A Global Illumination Solution for General Reflectance Distributions. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25, pages 187–196, July 1991.

[96] Francois Sillion, George Drettakis, and Cyril Soler. A Clustering Algorithm for Radiance Calculation in General Environments. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 196–205, New York, NY, 1995. Springer-Verlag.

[97] Francois Sillion and Claude Puech. A General Two-Pass Method Integrating Specular and Diffuse Reflection. In *Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, volume 23, pages 335–344, July 1989.

[98] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapmann and Hall, New York, NY, 1986.

[99] Philipp Slusallek, Marc Stamminger, Wolfgang Heidrich, Jan-Christian Popp, and Hans-Peter Seidel. Composite Lighting Simulations with Lighting Networks. *IEEE Computer Graphics and Applications*, 18(2):22–31, March/April 1998.

[100] Philipp Slusallek, Marc Stamminger, and Hans-Peter Seidel. Lighting Networks: A New Approach for Designing Lighting Algorithms. In *Graphics Interface '98*, pages 17–25, San Francisco, CA, June 1998. Morgan-Kaufmann.

[101] Brian Smits, James R. Arvo, and Donald Greenberg. A Clustering Algorithm for Radiosity in Complex Environments. In *Computer Graphics (ACM SIGGRAPH '94 Proceedings)*, volume 28, pages 435–442, 1994.

[102] Brian E. Smits, James R. Arvo, and David H. Salesin. An Importance-Driven Radiosity Algorithm. In *Computer Graphics (ACM SIGGRAPH '92 Proceedings)*, volume 26, pages 273–282, July 1992.

[103] Jerome Spanier and Ely M. Gelbard. *Monte Carlo Principles and Neutron Transport Problems*. Addison-Wesley, 1969.

[104] Jos Stam. Diffraction shaders. In *Computer Graphics (ACM SIGGRAPH '99 Proceedings)*, volume 33, pages 101–110, August 1999.

[105] Jos Stam. *MAYA, Caustics plugin*. Alias Wavefront, Toronto, Canada, www.aliaswavefront.com/en/Community/Learn/how_tos/rendering/caustics/, 2000.

[106] Orestes N. Stavroudis. *The optics of rays, wavefronts, and caustics (vol. 38 of Pure and applied physics)*. Academic Press, New York, London, 1972.

[107] Wolfgang Sturzlinger and Rui Bastos. Interactive Rendering of Globally Illuminated Glossy Scenes. In Julie Dorsey and Philipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 93–102, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.

[108] László Szirmay-Kalos. *Photorealistic Image Synthesis Using Ray-Bundles*. PhD thesis, Department of Control Engineering and Information Technology, Technical University of Budapest, Budapest, Hungary, 2000.

[109] László Szirmay-Kalos and Werner Purgathofer. Analysis of the Quasi-Monte Carlo Integration of the Rendering Equation. In V. Skala, editor, *7th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG 1999)*, 1999.

[110] David C. Tannenbaum, Peter Tannenbaum, and Michael J. Wozny. Polarization and Birefringency Considerations in Rendering. In *Computer Graphics (ACM SIGGRAPH '94 Proceedings)*, volume 28, pages 221–222, July 1994.

[111] Robert F. Tobler, Alexander Wilkie, Martin Feda, and Werner Purgathofer. A Hierarchical Subdivision Algorithm for Stochastic Radiosity Methods. In Julie Dorsey and Philipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 193–204. Springer Wien New York, 1997.

[112] Greg Turk. Generating Random Points in Triangles. In Andrew Glassner, editor, *Graphics Gems I*, pages 24–28. Academic Press Professional, Boston, MA, 1990.

[113] Square USA. *Kilauea Parallell Renderer*. Honolulu, Hawaii, http://www.squareusa.com/kilauea/index.html, 1998-2002.

[114] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford university, Department of Computer Science, Stanford, CA, December 1997.

[115] Eric Veach and Leonidas J. Guibas. Bidirectional Estimators for Light Transport. In G. Sakas, P. Shirley, and S. Müller, editors, *Photorealistic Rendering Techniques (Proceedings of the fifth Eurographics Workshop on Rendering*, pages 147–162, Darmstadt, Germany, June 1994. Springer.

[116] Eric Veach and Leonidas J. Guibas. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Computer Graphics (ACM SIGGRAPH '95 Proceedings)*, volume 29, pages 419–428, 1995.

[117] Eric Veach and Leonidas J. Guibas. Metropolis Light Transport. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 65–76, 1997.

[118] Bretton S. Wade. Kernel Based Density Estimation for Global Illumination. M.Sc. thesis, Program of Computer Graphics, Cornell University, Ithaca, NY, January 1996.

[119] Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. Interactive Global Illumination. In P. Debevec and S. Gibson, editors, *Rendering Techniques 2002 (Proceedings of the 13th Eurographics Workshop on Rendering)*, pages 9–20. Blackwell, June 2002.

[120] John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods. In *Computer Graphics (ACM SIGGRAPH '87 Proceedings)*, volume 21, pages 311–320, July 1987.

[121] Bruce Walter. *Density Estimation Techniques for Global Illumination*. PhD thesis, Program of Computer Graphics, Cornell University, Ithaca, NY, August 1998.

[122] Bruce Walter, Philip M. Hubbard, Peter Shirley, and Donald P. Greenberg. Global Illumination Using Local Linear Density Estimation. *ACM Transactions on Graphics*, 16(3):217–259, July 1997.

[123] Gregory J. Ward. Measuring and Modeling Anisotropic Reflection. In *Computer Graphics (ACM SIGGRAPH '92 Proceedings)*, volume 26, pages 265–272, July 1992.

[124] Gregory J. Ward. The RADIANCE Lighting Simulation and Rendering System. In *Computer Graphics (ACM SIGGRAPH '94 Proceedings)*, volume 28 of *Computer Graphics Proceedings, Annual Conference Series*, pages 459–472, July 1994.

[125] Gregory J. Ward and Paul S. Heckbert. Irradiance Gradients. In Alan Chalmers and Derek Paddon, editors, *Third Eurographics Workshop on Rendering*, pages 85–98, Bristol, UK, May 1992.

[126] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A Ray Tracing Solution for Diffuse Interreflection. In *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, volume 22, pages 85–92, August 1988.

[127] John E. Warnock. A Hidden-Surface Algorithm for Computer Generated Half-Tone Pictures. Technical Report TR 4–15, NTIS AD-733 671, University of Utah, Computer Science Department, 1969.

[128] Kevin Weiler and Peter Atherton. Hidden surface removal using polygonal area sorting. In *Computer Graphics (ACM SIGGRAPH '77 Proceedings)*, volume 11, August 1977.

[129] Lance Williams. Pyramidal Parametrics. In *Computer Graphics (ACM SIGGRAPH '83 Proceedings)*, volume 17, pages 1–11, July 1983.

[130] Harold R. Zatz. Galerkin Radiosity: A Higher Order Solution Method for Global Illumination. In *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, volume 27, pages 213–220, August 1993.