SCALABLE AND HETEROGENEOUS RENDERING OF SUBSURFACE SCATTERING MATERIALS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Adam Joseph Arbree

August 2009

SCALABLE AND HETEROGENEOUS RENDERING OF SUBSURFACE SCATTERING MATERIALS

Adam Joseph Arbree, Ph.D.

Cornell University 2009

In many natural materials, such as skin, minerals, and plastics, light scatters inside the material and gives them their distinctive appearance. The accurate reproduction of these materials requires new rendering algorithms that can simulate these subsurface interactions. Unfortunately, adding subsurface scattering dramatically increases the rendering cost. To achieve efficiency, recent approaches have used an approximate scattering model and have two significant limitations: they scale poorly to complex scenes and they are limited to homogeneous materials. This thesis proposes two new algorithms without these limitations.

The first is a scalable, subsurface renderer for homogeneous scattering. Using a canonical model of subsurface light paths, the new algorithm can judiciously determine a small set of important paths. By clustering the unimportant paths and approximating the contributions of these clusters, the new algorithm significantly reduces computation. In complex scenes, this new approach can achieve up to a three hundred fold speedup over the most efficient previous algorithms.

The second is the first, general, efficient and high-quality renderer for heterogeneous subsurface scattering. It based on a carefully derived formulation of the heterogeneous scattering problem using the diffusion equation and it solves that problem quickly and accurately using the finite element method. The new algorithm is designed for high-quality rendering applications producing, in minutes, images nearly identical to exact solutions produced in hours.

BIOGRAPHICAL SKETCH

Adam Joseph Arbree was born in Jupiter, Florida on Thursday, November 15th 1979. With his parents Ray and Kathi and, after a short delay, his sister Autumn, he spent his first 18 years a child. Life was good. Unfortunately, at age 18, changing political climates inside the Arbree household meant that he could no longer pursue childhood as a full-time profession. Following his other interests, he began attending the University of Florida in his spare time. There he received double Bachelors in Physics and Computer Science (with a Mathematics Minor on the side).

However, of much greater import during this period, he began a lifelong relationship with his wife—a beautiful and intelligent young woman named Bethany. Starting out with a 45-day car ride across much the western United States and Canada was—depending on whom you ask, them or people who dislike fun—either the best or worst decision of their young lives. But, regardless of opinion, the experience cemented their love for each other and in August of 2003 they settled down in Ithaca, New York to pursue their mutual dreams.

Citing a general dissatisfaction with his current reality, Adam dreamed of creating some more interesting versions. To this end, Adam enrolled at Cornell University for training in synthetic reality generation. Unfortunately, lacking in both imagination and artistic talent, he had to come to his dream the hard way. Through many thousands of hours of work, detailed investigations and perhaps too much coffee, Adam built machine that could display a handful of images of one particular dream-scape. Unfortunately, it is populated mainly by statuettes on tables. However, since these images have never been seen before, in July 2009 Cornell decided his efforts merited a "Degree of Doctor of Philosophy". Pleased with his initial success, Adam is currently writing a short, humorous biographical sketch of the experience.

For Mom and Dad

For as long as I can remember, I have wanted to do what I have just finished;
for longer than that, I remember your confidence that I could.

That confidence and your love are two of the greatest gifts of my life.

ACKNOWLEDGEMENTS

I'll be honest; writing this thesis was hard. However, it was nothing compared to writing these acknowledgments. There are so many people I want to thank for this work.

To begin I want to thank my adviser, Kavita Bala. She has a contagious zest for her work and encourages all around her to pursue new and interesting ideas with vigor. Throughout my research, she gave me the freedom to work on the projects that interested me and she was always supporting and trusting of my decisions. However, most importantly, she pushed back precisely when I needed it. Her advice has always been frank, well-founded and struck to the core of what, I realize in hindsight, that I needed most to hear. Whatever I become as a professional, I will think back on my time at Cornell and I will remember, in deep gratitude, her guidance. I am proud to have been her student.

Next I want to thank Bruce Walter for the hundreds of hours of input on my ideas, the innumerable discussions that taught me much of what I know about rendering, the enormous effort he put into the code that forms the core of my own renderer and, last but not least, the many, many free cups of Illy coffee. I know that without your effort I could not have accomplished this dissertation.

Steve Marschner deserves considerable credit for the following: buying a very expensive robot and then, immediately after unpacking and assembling it, giving a untrained, first-year Ph.D. student free license to program and play with it. Of course, I am sure to Steve, this is exactly the way things should be and that is precisely why he was such fun to work with. Of course, expensive toys aside, I want to thank Steve most for his invaluable advice and the time he took for so many discussions about my work.

I also want to heartily thank the other members of my committee: Charlie Van Loan, Alexander Vladimirsky and Lars Wahlbin. Their help, generous efforts on my behalf, insight and comments were essential to my success.

To Miloš Hašan and Ganesh Ramanarayanan, thank you for years of listening to me talk about my work, thought provoking criticism and intelligent insights. To Sebastian Fernandez, Jon Moon, Edgar Velazquez-Armendariz, Tim Condon, Todd Harvey, Konstantin Shkurko, Jaroslav Křivánek and all the other folks in CS Graphics and at the PCG, thank you for many enlightening conversations and your generally wonderful company.

Four new friends, Connie Etter, Brent Olsen, and Lori and Chris Klivak, have gone a long way to make the last few years some of the best in my life. All of you has given me a unique gift: a reminder to sit back, to laugh and to just have fun. I anticipated our nights together more eagerly than Christmas. Rarely have I had such a good time. Four old friends, John Leftwich, Sabrina McGraw, and Chuck and Jenny Tucker, I thank every chance I get. They are some of the most wonderful people I have ever known. My years with them have helped make me who I am.

Over the last few years and without a second glance, the Bloomstons—Harvey, Sue, Adam, Kim, Dustin and Jackson—have welcomed me into their family. In every moment that I have spent with them, they have been kind, generous and fun. I love you all and I am enormously proud of the love you give me. To Kim and Adam specifically, thank you everything you have done this last year. You helped me forget, at least for short weekends sleeping on your couches, that Bethany was so far away.

To my sister Autumn Arbree: of my time here, one of the few things I regret is that I had to be so far away from you. I love you because you are so uniquely like me and you understand so many things that no one else does. You are one of the few people I know will always be there. To my Mom and Dad, Ray and Kathi Arbree: I love you. I dedicated this thesis to you because completing it was difficult and I know that, without being your son, I never would have finished. All of the things in myself that I most value are yours. As proud as I am of this thesis, nothing compares to the pride I have in being your son.

To Bethany Bloomston: in every moment I spare, in every thought I have, in every action I take, I try—I try as hard as I can—to find the perfect way to tell you how, in all the myriad ways you do, you make my life wonderful. Nothing even begins to describe the richness, joy, hope, beauty, intelligence, honesty, laughter, excitement and fun you have brought to every moment of my life. Without you, nothing about me would be the same. When I say I love you—this time, all the past times and all the times yet to come—this is what I mean.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Basic Parameters		
$\mathbf{x}$	3D position vector	
$\vec{\omega}$	3D direction vector	
$\vec{n}$	Surface normal vector	
Ω	Domain of volume scattering	
$\partial\Omega$	Boundary of Ω	

Radiance Functions		
$L(\mathbf{x},\vec{\omega})$	Radiance	
$L^{\text{cam}}(\mathbf{x})$	Radiance at $\mathbf{x}$ traveling towards the camera	
$L_d(\mathbf{x},\vec{\omega})$	Diffuse radiance	
$L_{ri}(\mathbf{x},\vec{\omega})$	Reduced intensity radiance	
L_{pixel}	Radiance integrated over the area of a pixel	
$\phi(\mathbf{x})$	Radiant fluence; 0^{th} moment of $L(\mathbf{x},\vec{\omega})$	
$\vec{E}(\mathbf{x})$	Vector irradiance; 1^{st} moment of $L(\mathbf{x},\vec{\omega})$	

Surface Reflectance Properties		
$f_r(\mathbf{x},\vec{\omega},\vec{\omega}')$	BRDF function	
$S(\mathbf{x},\vec{\omega},\mathbf{x}',\vec{\omega}')$	BSSDRF function	
$R(\lVert\mathbf{x}-\mathbf{x}'\rVert)$	Dipole BSSDRF function	
η	Relative index of refraction	
$F_r(\eta,\vec{\omega})$	Fresnel reflection coefficient	
$F_t(\eta,\vec{\omega})$	Fresnel transmission coefficient	
$F_{dr}(\eta)$	Average Fresnel reflectance	
$F_{dt}(\eta)$	Average Fresnel transmittance	
$A(\eta)$	Surface Fresnel factor	

Volume Scattering Properties		
$p(\vec{\omega},\vec{\omega}')$	Phase function	
μ	0^{th} moment of $p(\vec{\omega},\vec{\omega}')$	
$\sigma_a(\mathbf{x})$	Absorption coefficient	
$\sigma_s(\mathbf{x})$	Scattering coefficient	
$\sigma_t(\mathbf{x})$	Total extinction coefficient	$= \sigma_s(\mathbf{x}) + \sigma_a(\mathbf{x})$
$l(\mathbf{x})$	Mean free path	$= {}^{1}/_{\sigma_t(\mathbf{x})}$

$\alpha(\mathbf{x})$	Albedo	$= {}^{\sigma_s(\mathbf{x})}/_{\sigma_t(\mathbf{x})}$
$\sigma_{sr}(\mathbf{x})$	Reduced scattering coefficient	$= (1-\mu)\sigma_s(\mathbf{x})$
$\sigma_{tr}(\mathbf{x})$	Reduced extinction coefficient	$= \sigma_{sr}(\mathbf{x}) + \sigma_a(\mathbf{x})$
$\sigma_{et}(\mathbf{x})$	Effective transport coefficient	$= \sqrt{3\sigma_a(\mathbf{x})\sigma_t(\mathbf{x})}$
$\kappa_d(\mathbf{x})$	Diffusion coefficient	$= {}^{1}/_{3\sigma_{tr}(\mathbf{x})}$

Scalable Homogeneous Algorithm

$\mathcal{E}$	Region of the scene visible through a given pixel
$\mathbf{e}$	Eye sample
E_k	Weight of the k^{th} eye sample
$\mathbb{E}$	Set of all eye samples
$\mathbb{C}_{\mathbf{e}}$	A cluster of eye sample
$\mathbb{E}_{\mathbb{C}}$	Set of all eye sample clusters
$\mathbf{l}$	Light sample
I_i	Weight of the i^{th} light sample
$\mathbb{L}$	Set of all light samples
$\mathbb{C}_{\mathbf{l}}$	A cluster of light samples
$\mathbb{L}_{\mathbb{C}}$	Set of all light sample clusters
$\gamma_i(\vec{\omega})$	Directional emittance function of the i^{th} light sample
$\mathbf{b}$	Irradiance sample
B_j	Weight of the j^{th} irradiance sample
$\mathbb{B}$	Set of all irradiance samples
$\mathbb{C}_{\mathbf{b}}$	A cluster of irradiance samples
$\mathbb{B}_{\mathbb{C}}$	Set of all irradiance sample clusters
$\mathcal{B}_j$	Total irradiance at the j^{th} irradiance sample
$\mathbb{T}$	Set of all sample triples
$\mathbb{C}_{\mathbf{t}}$	A cluster of sample triples
$\mathbb{T}_{\mathbb{C}}$	Set of all sample triple clusters
F_{ji}	Irradiance link weight
R_{kj}	BSSRDF link weight

Heterogeneous Finite Element Algorithm

$Q(\mathbf{x},\vec{\omega})$	Light source function
$Q^0(\mathbf{x})$	0th moment of $Q(\mathbf{x},\vec{\omega})$

$Q^1(\mathbf{x})$	1st moment of $Q(\mathbf{x},\vec{\omega})$
$Q_{ri}(\mathbf{x},\vec{\omega})$	Reduced intensity source function
$Q^0_{ri}(\mathbf{x})$	0th moment of $Q_{ri}(\mathbf{x},\vec{\omega})$
$Q^1_{ri}(\mathbf{x})$	1st moment of $Q_{ri}(\mathbf{x},\vec{\omega})$
$\Gamma_s(\mathbf{x})$	Refracted incident exterior boundary flux
$\Gamma^{\text{in}}_d(\mathbf{x})$	Inward diffuse boundary flux
$\Gamma^{\text{ref}}_d(\mathbf{x})$	Reflected diffuse boundary flux
F	FEM matrix
D	Component of F
M	Component of F
S	Component of F
$\vec{\mathsf{a}}$	FEM coefficient vector
$\vec{\mathsf{r}}$	FEM right hand side vector
$\vec{\mathsf{q}}$	Component of $\vec{\mathsf{r}}$
$\vec{\mathsf{g}}$	Component of $\vec{\mathsf{r}}$
$\mathbb{H}$	Hilbert function space
$\mathcal{H}$	A bilinear form on functions in $\mathbb{H}$
$\mathcal{F}$	A linear form on functions in $\mathbb{H}$

Discontinuous Galerkin Finite Element Method

Ω_i	Subregion of Ω
N_e	Number of Ω_i
$\mathbb{H}^{\text{dg}}_{\phi}$	Function space containing $\phi(\mathbf{x})$
$\mathbb{H}^{\text{dg}}_{\vec{E}}$	Function space containing $\vec{E}(\mathbf{x})$
f_{ij}	Face between the i^{th} and j^{th} elements
$\mathcal{F}_{\mathcal{I}}$	Set of all interior faces
$\vec{E}^*$	Interior face $\phi(\mathbf{x})$ estimate
κ^*_d	Interior face $\vec{E}(\mathbf{x})$ estimate
η	Discontinuous penalty coefficient
E_{ij}	Discontinuous matrix penalty term

CHAPTER 1

INTRODUCTION

In the last two decades applications using synthetic, computer generated (CG) imagery have proliferated dramatically. No visual medium has been unaffected. Almost every movie, television show or magazine cover contains some synthetically generated content. Fully computer animated movies have almost completely replaced 2D animation and the most recent personal video game consoles provide fully interactive, real-time experiences whose quality rivals the best CG cinema from a decade past. All of these advances were made possible by a continuous stream of improvements in computer rendering algorithms. However, most of these improvements rest on fundamental reflectance approximations that exclude important classes of materials. The exclusion of the class of translucent, *subsurface scattering*, materials is particularly limiting since it contains most organic materials including skin, leaves and foods as well as many common inorganic materials such as minerals and plastics. Because of their prevalence in everyday scenes, efficient and accurate subsurface rendering algorithms are an essential next step towards realistic rendering.

Rendering algorithms compute an image by simulating the interaction of light with the objects and materials in the scene. Since light makes orders-of-magnitude more interactions with subsurface scattering materials than without, introducing subsurface scattering into a scene dramatically increases the cost of the rendering computation. Until this thesis, almost all practical subsurface rendering algorithms addressed this cost explosion by replacing the simulation of subsurface interactions with the evaluation of a single *bidirectional surface scattering reflectance distribution function* (BSSRDF), the dipole diffusion BSSRDF. Unfortunately, using the dipole diffusion BSSRDF has two limitations: the algorithms are limited to homogeneous

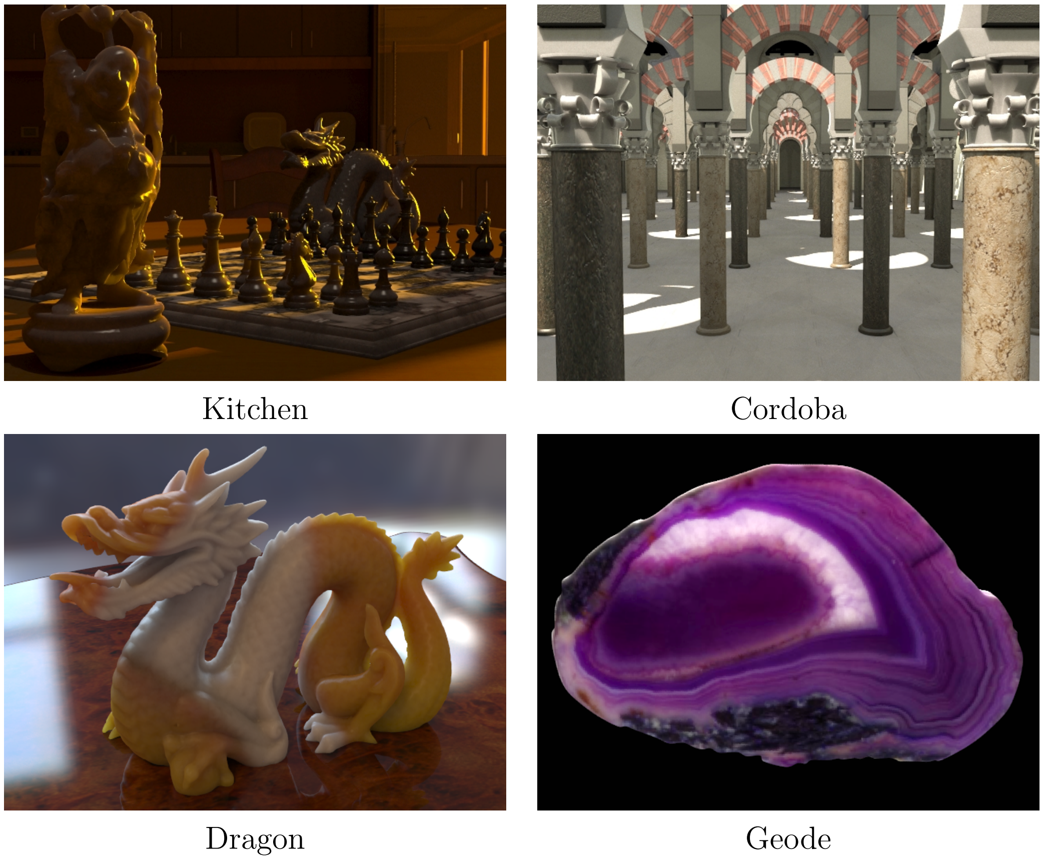


Figure 1.1: Top row: Large scenes with many subsurface scattering objects. Bottom row: Complex heterogeneously scattering materials rendered quickly and accurately.

materials and they scale poorly on large scenes. This thesis extends the state-of-the-art in subsurface rendering by introducing two new subsurface rendering algorithms that lift these limitations. The first algorithm is a scalable dipole diffusion rendering algorithm. It introduces a new, adaptive hierarchical representation of the subsurface computation that allows the algorithm to better focus its effort on the parts of the calculation that contribute to the image. This better distribution of effort allows the new algorithm to render larger scenes with more detailed geometry and more complicated lighting effects than were previously possible. The second algorithm is the first efficient and accurate simulation algorithm for general, complex, heterogeneous materials. Together these algorithms address the two basic challenges of subsurface rendering, significantly advance rendering technology and

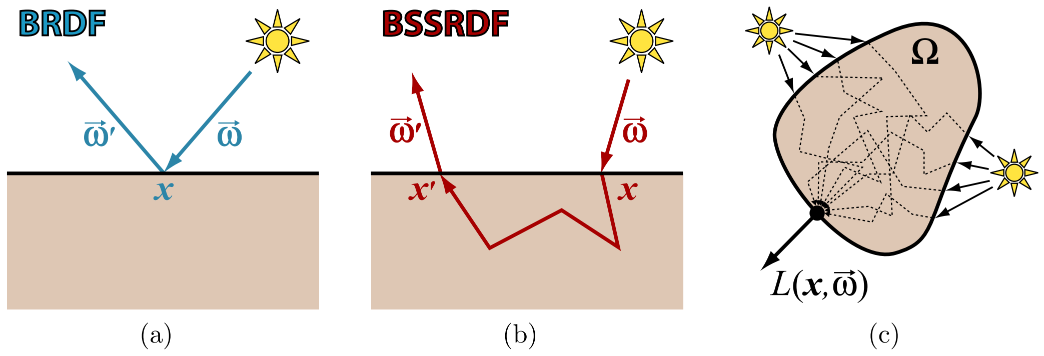


Figure 1.2: (a) Surface material interaction as described by a BRDF; (b) subsurface material interaction as described by a BSSRDF; and (c) basic problem of subsurface rendering.

make it possible to render new types of scenes and materials (see Figure 1.1).

1.1 Surface vs. Subsurface Rendering

To highlight the contributions of these new algorithms, this section discusses why subsurface scattering makes the rendering problem more difficult. Rendering algorithms simulate the propagation of light in synthetic scenes. They require four inputs:

- The position and emission function of each light source;
- The geometry of each object;
- The reflectance function for each material; and
- The position and focal parameters of the camera.

To compute the light that arrives at the camera, a rendering algorithm propagates the light from the sources simulating its interactions with the geometry and materials. For a given image, the algorithm performs this calculation pixel by pixel. Through the pixel, the algorithm computes **x**, the first visible surface point, and then computes the light, or radiance[1] $L(\mathbf{x}, \vec{\omega})$, that leaves $\mathbf{x}$ in $\vec{\omega}$ the direction towards the camera aperture. Compared to surface rendering, accurate subsurface rendering requires a more general model of light/material interactions that significantly increases the cost of computing $L(\mathbf{x}, \vec{\omega})$.

1.1.1 Surface Rendering Model

The reflectance of simple materials can be accurately modeled by a *bidirectional reflectance distribution function* (BRDF) (see Figure 1.2(a)). In the BRDF model light is both incident and reflected at a single point $\mathbf{x}$. The BRDF describes the probability that at $\mathbf{x}$ the light will reflect from direction $\vec{\omega}$ into direction $\vec{\omega}'$. Using a BRDF, $L(\mathbf{x}, \vec{\omega})$ is defined by the surface scattering integral [Kaj86]

$$L(\mathbf{x}, \vec{\omega}) = \int\limits_{(\vec{n}\cdot\vec{\omega}')>0} f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}') L(\mathbf{x}, \vec{\omega}')(\vec{\omega}' \cdot \vec{n})\, d\vec{\omega}' \tag{1.1}$$

where $f_r(\mathbf{x}, \vec{\omega}, \vec{\omega}')$ is the BRDF and $\vec{n}$ is the surface normal vector at $\mathbf{x}$.

1.1.2 Subsurface Rendering Model

Unfortunately, BRDF models fail for subsurface scattering materials. When the material contains subsurface scattering, the light incident on the material at $\mathbf{x}$ and $\vec{\omega}$ enters the object volume, is scattered and absorbed, and then exits the material in a separate position $\mathbf{x}'$ and direction $\vec{\omega}'$ (see Figure 1.2(b)). For subsurface scattering materials, accurate reproduction of appearance requires simulating these interior material interactions. Since the BRDF is insufficient, subsurface algorithms

[1]It is standard in computer graphics literature to omit the common dependence on the wavelength of the light in all terms. To keep the notation consistent, this thesis follows this standard. Unless otherwise noted, any quantity related to light transport is implicitly wavelength dependent and any computations using these quantities—everything described in this thesis—must be repeated once per wavelength of interest. Typically three wavelengths are chosen, one red, one green and one blue.

use a generalization of the BRDF, the *bidirectional surface scattering reflectance distribution function* (BSSRDF) [NRH*77]. Analogous to Equation (1.1), subsurface rendering is defined by the volumetric scattering integral [JMLH01] (see Figure 1.2(c))

$$L(\mathbf{x},\vec{\omega}) = \int\limits_{\partial\Omega} \int\limits_{4\pi} S(\mathbf{x},\vec{\omega},\mathbf{x}',\vec{\omega}')L(\mathbf{x},\vec{\omega}')(\vec{\omega}' \cdot \vec{n})\, d\vec{\omega}'\, d\mathbf{x}' \tag{1.2}$$

where $S(\mathbf{x},\vec{\omega},\mathbf{x}',\vec{\omega}')$ is the BSSRDF and $\partial\Omega$ is the boundary of the scattering domain Ω.

1.2 Problems Solved by this Thesis

The differences between Equation (1.1) and Equation (1.2) succinctly illustrate the two fundamental problems introduced and solved by this thesis.

Problem: Rapid Growth in Recursive Evaluations Since subsurface renderers must integrate over a larger, more complex domain, the boundary of an arbitrary model, they make significantly more recursive evaluations of the integrand radiance. As the subsurface objects grow in size and number, the total cost of these recursive radiance evaluations can become prohibitive.

Solution: Scalable, Hierarchical Evaluation Algorithm By intelligently determining the number and accuracy of the recursive radiance computations, a new scalable rendering algorithm can, while preserving image quality, prune unnecessary evaluations and approximate less important ones. For small scenes the algorithm performs no worse than the most efficient subsurface algorithms but for large scenes it offers dramatic performance improvements.

Problem: Heterogeneous BSSRDFs Many BRDFs are represented by simple, analytic functions [MPBM03]. However, the BSSRDF is usually a complicated function of the material and geometry and often lacks a convenient analytic form. Accurate evaluation of the BSSRDF requires a secondary simulation of the light scattering within the material. For complex, general heterogeneous materials, no efficient and accurate simulation algorithm has been developed.

Solution: Accurate Finite Element (FE) Algorithm By carefully formulating an approximate heterogeneous scattering problem using the diffusion equation and solving that problem accurately, a new FE algorithm can solve complex subsurface scattering problems in a few minutes producing images comparable to exact solutions that require hours of computation.

The next two sections present an overview of these problems and solutions.

1.3 Scalable Subsurface Rendering

Scalable rendering exploits a fundamental property of images. Roughly, for a given size, the amount of information an image can contain is fixed. If the scene in the image becomes larger or more complex then the relative image importance of some parts of the scene must go down. If image importance was distributed equally, scalable rendering would be easy. Unfortunately, importance is a complex function of the image, the scene and the rendering algorithm that often defies heuristic definition. To tackle this problem, scalable rendering tries to adaptively detect importance by performing progressively more refined calculations that home in on important regions. This thesis demonstrates that scalable rendering is more difficult with subsurface materials and that current algorithms are not scalable. Then, it develops the first major contribution of this work: a scalable rendering algorithm for homogeneous materials based on the dipole diffusion BSSRDF.

Introducing subsurface scattering simultaneously increases the relative differences in importance between regions of the scene and makes them harder to detect.

This is intuitive. For example, consider Figure 1.2(c) that illustrates the integral in Equation (1.2). The integral will contain contributions from the entire surface of the object. But, clearly, most of the contributions will come from a smaller subset of surface regions *optically close* to $\mathbf{x}$. A region is optically close if it is either physically close or the material between the region and the goal is less dense. Because the BSSRDF, a complex function material and geometry, determines whether a region is optically close, it is challenging for rendering algorithms to determine which regions are important.

The new algorithm developed in this thesis addresses this difficult problem. By introducing a new unified, hierarchical representation of subsurface light paths, called triple clusters, the new scalable algorithm can adaptively determine the importance distribution within the subsurface integral. In tests presented in Chapter 4 the new scalable algorithm reduces subsurface rendering cost in the Cordoba scene (see Figure 1.1, upper right) by a factor of 300.

1.4 Heterogeneous Subsurface Rendering

For many years, subsurface rendering had a large quality/performance gap. On one hand there are exact algorithms for rendering high quality images of arbitrary subsurface materials; however, these rendering algorithms are sufficiently costly to be impractical. On the other hand, approximate, homogeneous scattering can be computed in much less time by using the dipole diffusion BSSRDF. The new finite element (FE) algorithm presented in this thesis bridges this gap. It can render complex heterogeneous scattering in a few minutes with quality comparable to the exact algorithms that take many hours (see Figure 1.1, bottom row).

The thesis presents the first efficient and accurate solution for general heterogeneous subsurface scattering. To achieve efficiency, the algorithm uses an approximate scattering model, the *diffusion equation* (DE), and solves the equation quickly and accurately using the finite element (FE) method. The contributions of this algorithm are two-fold:

1. It demonstrates that a careful formulation of the diffusion problem, including a description of a new, boundary source model, obtains accuracies useful for high-quality rendering applications.
2. It presents a FE solution for this problem describing in detail how to efficiently discretize, assemble, solve and refine that solution.

These contributions reduce the difficult heterogeneous subsurface scattering problem to a new, simple, four-step algorithm that works well for a wide range of scattering materials; solves for scattering in arbitrary surface geometry; produces images comparable to exact algorithms; and requires only a few minutes per image.

1.5 Thesis Organization

The rest of the thesis motivates, derives and reviews these new advanced algorithms for subsurface rendering. Chapter 2 introduces the basic scattering physics that underlie the two algorithms. Chapter 3 discusses the limitations of previous solutions. Chapters 4 and 5 describe and analyze the scalable, dipole diffusion algorithm and the accurate heterogeneous algorithm respectively. Then, by using a different finite element solution, the discontinuous Galerkin method, Chapter 6 discusses why the heterogeneous solver in Chapter 5 is a particularly accurate rendering solution. Finally, Chapter 7 reviews the contributions of all these works, discusses their limitations and proposes remaining problems at the forefront of subsurface rendering research.

CHAPTER 2

SUBSURFACE SCATTERING THEORY

This chapter introduces the three models of subsurface scattering used by most rendering algorithms.

- The *volumetric radiative transfer equation* (VRTE) accurately models the physics of subsurface scattering;
- The *diffusion equation* (DE) approximates the VRTE by assuming many random scattering events erase all but the lowest order directional properties from the radiance;
- The *dipole diffusion BSSRDF* solves the DE approximately in the specific case of a semi-infinite slab filled with an isotropically and homogeneously scattering medium.

Classes of rendering algorithms using each of these models are progressively less accurate but faster. Monte Carlo (MC) path tracing algorithms use the VRTE to compute exact photon paths—one at a time. For any scene and material, given sufficient time, these algorithms compute images of the highest quality; however, at best, they require hours of computation and are generally considered impractical. Practical rendering algorithms rely on one of the two approximate models. Chapter 5 demonstrates, for the first time, that for many materials the DE is almost as accurate as path tracing and can be solved in a few minutes. However, for many applications, this is still too expensive. Using the dipole diffusion BSSRDF, many algorithms limit themselves to homogeneous materials but then approximate scattering very fast, even in real-time. Unfortunately, despite the low cost of the dipole diffusion BSSRDF, the structure of the basic dipole rendering algorithms causes them to perform poorly on large, complex scenes. Chapter 4 solves this problem with a new scalable, dipole diffusion rendering algorithm that efficiently renders large, complex scenes beyond the limits of all previous methods.

The development of the algorithms in Chapters 4 and 5 is closely linked to the mathematics of the diffusion equation and the dipole diffusion BSSRDF. To place these later discussions in context, this chapter presents the derivations of each of these models and highlights the details related to the future discussion. The two derivations in this chapter closely follow the work from scattering physics by Ishimaru [Ish78] who originally derived both the DE and the dipole diffusion BSSRDF; however, the notation matches the standard introduced to computer graphics by Jensen et al. [JMLH01].

The second section in this chapter, Section 2.2, on the diffusion equation is the longest and most relevant to later discussions. The section has two main subsections. The first derives the diffusion equation. Besides the DE itself, the most important consequence of this derivation is the *reduced intensity source* which models radiance that violates the fundamental assumption of the DE, the diffusion approximation (DA). The second subsection describes how to use the DE to create a basic rendering algorithm. Since both of the algorithms in this thesis follow this basic structure, this discussion is the most important in the chapter. It has three goals.

1. It solves the basic problem of computing the radiance values needed for rendering from a solution to the DE.

2. It discusses how to derive the DE's boundary condition. The significant results are the *Robin boundary condition* and the *extrapolated boundary condition*.

3. It introduces two models of the reduced intensity source: the *boundary source model* and *embedded source model*.

The dipole diffusion BSSRDF, used by the scalable algorithm in Chapter 4, is derived from the extrapolated boundary condition and the embedded source model. In Chapter 5, the more accurate Robin boundary condition and boundary source model are combined to derive the *diffusive source boundary condition*—an accurate heterogeneous boundary condition used by the new FEM rendering algorithm (see Section 5.2.2). The rest of this chapter proceeds along the sequence of approximations: VRTE to DE to dipole diffusion BSSRDF. Section 2.1 introduces the basic parameters of a randomly scattering medium and then uses them to define the VRTE; Section 2.2 derives the DE; and, finally, Section 2.3 derives the dipole diffusion BSSRDF.

2.1 Basic Scattering Physics

This first section introduces the basic physics of scattering. The physical model of most materials is the *random medium*. A random medium contains a collection of particles randomly distributed and having random sizes and shapes. The model implicitly assumes that the particles are much larger than the wavelength of the light scattered and therefore it ignores quantum effects, such as diffraction. Because in most physical problems the number of particles is essentially infinite, it is practically impossible to consider modeling particles individually. Instead, the aggregate collection of particles is defined by the probability of interactions of light with the medium. The goal of this section is two-fold: to introduce the basic parameters of a random media used throughout this thesis and to introduce the equation that governs the physical behavior of light in such a medium, the volumetric radiative transfer equation.

2.1.1 Parameters of a Random Media

A random medium is completely specified by three functions.

Absorption coefficient $\sigma_a(\mathbf{x})$ has units of inverse distance (typically mm^{-1}) and describes the expected number of absorption interactions per unit distance traveled.

Scattering coefficient $\sigma_s(\mathbf{x})$ has the same units and describes the expected number of scattering interactions per unit distance traveled.

Phase Function $p(\vec{\omega}, \vec{\omega}')$ has no units. It describes the probability that light scatters from direction $\vec{\omega}$ into direction $\vec{\omega}'$ during a scattering event. It could vary by position but this thesis assumes that $p(\vec{\omega}, \vec{\omega}')$ is constant throughout the entire scattering domain. The DE derivation (see Section 2.2 and Appendix A) requires that the phase function be normalized and depend only on the angle between $\vec{\omega}$ and $\vec{\omega}'$, i.e.

$$\int_{4\pi} p(\vec{\omega}, \vec{\omega}')\, d\vec{\omega}' = 1 \tag{2.1}$$

$$p(\vec{\omega}, \vec{\omega}') = p(\vec{\omega} \cdot \vec{\omega}') \tag{2.2}$$

From these three basic parameters, several additional parameters, referenced throughout this thesis, can be derived:

Mean cosine of the phase function μ is the probability weighted average of the cosine of the angle between the light's current direction and its scattered direction:

$$\mu = \int_{4\pi} p(\vec{\omega}, \vec{\omega}')(\vec{\omega} \cdot \vec{\omega}')\, d\vec{\omega}' \tag{2.3}$$

If μ is positive, the material is said to be *forward scattering* since light tends to scatter into directions closer to its current direction. Analogously, if μ

is negative, the material is *backward scattering.* If $\mu = 0$ the material has *isotropic scattering.* The a common normalized isotropic phase function is the constant, $p(\vec{\omega}, \vec{\omega}') = {}^1\!/_{4\pi}$.

Total extinction coefficient $\sigma_t(\mathbf{x}) = \sigma_s(\mathbf{x}) + \sigma_a(\mathbf{x})$ is the total number of interactions, both absorption and scattering, per unit distance traveled.

Mean free path $l(\mathbf{x}) = {}^1\!/_{\sigma_t(\mathbf{x})}$ is the average distance light travels before interacting. The mean free path measures the optical thickness, or relative transparency, of a material.

Albedo $\alpha(\mathbf{x}) = {}^{\sigma_s(\mathbf{x})}\!/_{\sigma_t(\mathbf{x})}$ is the fraction of all interactions that scatter. Lower albedo materials are more absorptive and appear darker. Most materials have a different albedo for each wavelength and the relative differences in absorption give the material its color.

Reduced scattering coefficient $\sigma_{sr}(\mathbf{x}) = (1 - \mu)\sigma_s(\mathbf{x})$ gives the scattering coefficient of an isotropically scattering material that, in the limit of many scattering events, has approximately the same properties as a non-isotropically scattering material whose phase function has a mean cosine of μ. This transformation is commonly used to apply scattering models valid only for isotropic materials to non-isotropic ones. The transformation leaves the absorption coefficient unchanged but it introduces the *reduced total extinction coefficient* $\sigma_{tr}(\mathbf{x}) = \sigma_{sr}(\mathbf{x}) + \sigma_a(\mathbf{x})$.

Diffusion coefficient $\kappa_d(\mathbf{x}) = {}^1\!/_{3\sigma_{tr}(\mathbf{x})}$ is a derived material property that results from the derivation of the diffusion equation (see Equation (2.15)).

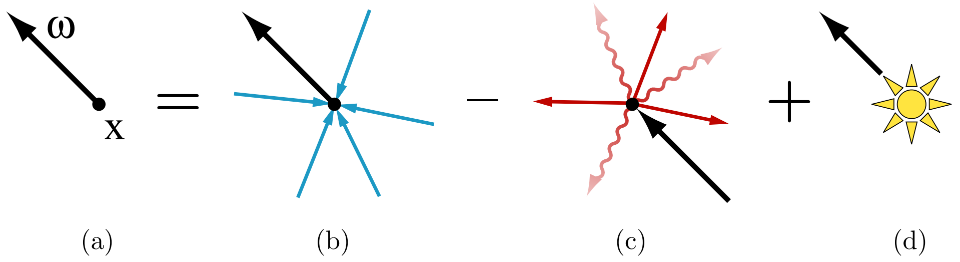


Figure 2.1: (a) The differential radiance $(\vec{\omega} \cdot \vec{\nabla})L(\mathbf{x}, \vec{\omega})$ emitted from $\mathbf{x}$ in direction $\vec{\omega}$ is formed from three components: (b) the light that scatters into $\vec{\omega}$ at $\mathbf{x}$ minus (c) the light lost to absorption and scattering at x plus (d) the emission of the media in direction $\vec{\omega}$.

2.1.2 Volumetric Radiative Transfer Equation

Given the parameters above, the scattering of light in a random medium is exactly defined by the volumetric radiative transfer equation (VRTE) [Ish78]. It defines the differential radiance $(\vec{\omega} \cdot \vec{\nabla})L(\mathbf{x}, \vec{\omega})$ at each point in the medium.

$$(\vec{\omega} \cdot \vec{\nabla})L(\mathbf{x}, \vec{\omega}) = \sigma_s(\mathbf{x}) \int_{4\pi} p(\vec{\omega}, \vec{\omega}')L(\mathbf{x}, \vec{\omega}')\, d\vec{\omega}' - \sigma_t(\mathbf{x})L(\mathbf{x}, \vec{\omega}) + Q(\mathbf{x}, \vec{\omega}) \qquad (2.4)$$

In Equation (2.4), $Q(\mathbf{x}, \vec{\omega})$ is the source function. On the boundary it describes any light incident on the medium from external light sources and within the medium it describes any emission from the medium itself. Figure 2.1 illustrates how the three terms in the VRTE respectively account for the three basic scattering phenomenon. Changes in the radiance at $\mathbf{x}$ in $\vec{\omega}$ can arise because either:

Figure 2.1(b) light in-scatters into $\vec{\omega}$ from other directions;

Figure 2.1(c) light is lost to absorption or out-scatter into other directions; or

Figure 2.1(d) light is emitted in $\vec{\omega}$ by the source.

2.1.3 Basic Path Tracing

Path tracing algorithms use the VRTE to generate photon paths. Energy conservation guarantees that all light interactions are reciprocal so the paths can be traced either forwards or backwards using the same algorithm. Path tracing algorithms generally start from the camera and work backwards. At each step the algorithm has a partial path and it uses it to transmit any radiance emitted from its current position back to the camera. Then it extends the path by one segment by sampling the first two terms of the VRTE. The algorithm randomly samples the albedo (the ratio of the two terms) to determine if the light scatters or is absorbed. If it is absorbed, the simulation of the path stops. If it scattered, the first term is sampled to create a new path segment. Repeating this process many times creates a single path and tracing billions of such paths creates an image. Unfortunately, this path-by-path simulation usually requires many hours of computation. The diffusion equation, derived in the next section, reduces this cost by modeling the bulk effect of many photon paths together.

2.2 Diffusion Equation

Almost all subsurface rendering algorithms rely on the diffusion equation (DE); either directly or indirectly through the dipole diffusion BSSRDF. The diffusion equation simplifies the scattering computation by assuming the light randomly scatters in the medium enough to erase all but the lowest order angular dependence from the radiance. By idealizing the radiance function as first order angular expansion, the VRTE can reduced to a constraint on the total radiant power, the *fluence*, at each point in the medium. This constraint is the diffusion equation. Because the approximations required to derive the DE have fundamental consequences on both of the rendering algorithms in this thesis, this section provides an in-depth introduction to DE theory. This section has two parts. The first part motivates the diffusion approximation (DA), the basic simplifying assumption of the DE, and then uses the diffusion approximation to derive the DE. The second part, the most important part of this chapter, discusses how to use the DE for rendering.

2.2.1 Derivation

The derivation of the DE has three steps:

Step 1 introduces the diffusion approximation;

Step 2 restricts the problem by introducing the reduced intensity source to model the radiance that violates the diffusion approximation; and

Step 3 substitutes the diffusion approximation into the restricted problem to produce the DE.

2.2.1.1 Step 1: Diffusion Approximation

Because light interacts with the material many times, materials with high albedo and short mean free paths are the most expensive to render using path tracing. However, for these materials, the DE is most accurate because it leverages a consequence of all this scattering: frequent random scattering smooths the aggregate angular radiance distribution. As the scale the scattering domain Ω grows relative to the mean free path l, one can assume some limit beyond which sufficient scattering will erase almost all of the angular properties of the source $Q(\mathbf{x}, \vec{\omega})$. The diffusion approximation is the specific assumption that the final aggregate radiance is linear in its 0^{th} and 1^{st} angular moments, the scalar fluence $\phi(\mathbf{x})$ and the vector irradiance $\vec{E}(\mathbf{x})$.

$$\phi(\mathbf{x}) = \int_{4\pi} L(\mathbf{x}, \vec{\omega}')\, d\vec{\omega} \tag{2.5}$$

$$\vec{E}(\mathbf{x}) = \int_{4\pi} L(\mathbf{x}, \vec{\omega}') \cdot \vec{\omega}'\, d\vec{\omega} \tag{2.6}$$

$$L(\mathbf{x}, \vec{\omega}) = \frac{1}{4\pi}\phi(\mathbf{x}) + \frac{3}{4\pi}\vec{\omega} \cdot \vec{E}(\mathbf{x}) \tag{2.7}$$

2.2.1.2 Step 2: Reduced Intensity Source

Unfortunately, the DA cannot be applied to all radiance in the medium. Sufficiently near the boundary or the interior sources, the source radiance will not yet have scattered and this radiance might contain non-linear components. To apply the DA more accurately, this yet-to-have-scattered source radiance, called the reduced intensity radiance $L_{ri}(\mathbf{x},\vec{\omega})$, is separated from from the remaining, diffusive radiance $L_d(\mathbf{x},\vec{\omega})$.

$$L(\mathbf{x},\vec{\omega}) = L_d(\mathbf{x},\vec{\omega}) + L_{ri}(\mathbf{x},\vec{\omega}) \tag{2.8}$$

Since the DA does not apply to $L_{ri}(\mathbf{x},\vec{\omega})$, it is removed from the problem by assuming that a new input source function can be supplied that gives diffusive radiance out-scattered by $L_{ri}(\mathbf{x},\vec{\omega})$. This new term is called the reduced intensity source $Q_{ri}(\mathbf{x},\vec{\omega})$.

$$Q_{ri}(\mathbf{x},\vec{\omega}) = \sigma_s(\mathbf{x}) \int\limits_{4\pi} p(\vec{\omega},\vec{\omega}') L_{ri}(\mathbf{x},\vec{\omega}')\, d\vec{\omega}' \tag{2.9}$$

Using these two equations, the VRTE can be restricted only to $L_d(\mathbf{x},\vec{\omega})$. First substitute Equation (2.8) into Equation (2.4) to yield

$$\begin{aligned}\big(\vec{\omega}\cdot\vec{\nabla}\big)L_d(\mathbf{x},\vec{\omega}) + \big(\vec{\omega}\cdot\vec{\nabla}\big)L_{ri}(\mathbf{x},\vec{\omega}) = \\ \sigma_s(\mathbf{x}) \int\limits_{4\pi} p(\vec{\omega},\vec{\omega}') \left[L_d(\mathbf{x},\vec{\omega}') + L_{ri}(\mathbf{x},\vec{\omega}')\right]\, d\vec{\omega}' \\ - \sigma_t(\mathbf{x})L_d(\mathbf{x},\vec{\omega}) - \sigma_t(\mathbf{x})L_{ri}(\mathbf{x},\vec{\omega}) + Q(\mathbf{x},\vec{\omega})\end{aligned} \tag{2.10}$$

Next, because $L_{ri}(\mathbf{x},\vec{\omega})$ only includes light that has not scattered, differential changes to $L_{ri}(\mathbf{x},\vec{\omega})$ can only occur due to losses from absorption and outscatter. Thus, it satisfies

$$\big(\vec{\omega}\cdot\vec{\nabla}\big)L_{ri}(\mathbf{x},\vec{\omega}) = -\sigma_t(\mathbf{x})L_{ri}(\mathbf{x},\vec{\omega}) \tag{2.11}$$

Then, by substituting Equations (2.11) and (2.9), Equation (2.10) becomes dependent only on $L_d(\mathbf{x}, \vec{\omega})$

$$\big(\vec{\omega} \cdot \vec{\nabla}\big) L_d(\mathbf{x}, \vec{\omega}) = \sigma_s(\mathbf{x}) \int\limits_{4\pi} p(\vec{\omega}, \vec{\omega}') L_d(\mathbf{x}, \vec{\omega}') \, d\vec{\omega}'$$

$$- \sigma_t(\mathbf{x}) L_d(\mathbf{x}, \vec{\omega}) + Q_{ri}(\mathbf{x}, \vec{\omega}) + Q(\mathbf{x}, \vec{\omega}) \quad (2.12)$$

2.2.1.3 Step 3: Substituting the Diffusion Approximation

Assuming that $L_d(\mathbf{x}, \vec{\omega})$ in Equation (2.12) satisfies the diffusion approximation (Equation (2.7)) yields the diffusion equation. Algebraically, this process has three steps. The first two define separate equations that relate $\phi(\mathbf{x})$ and $\vec{E}(\mathbf{x})$ by computing integrals of Equation (2.12). The first is created by equating the 0th moments of the terms in Equation (2.12) and the second is created by substituting Equation (2.7) into Equation (2.12) and then equating the 1st moments of the resulting terms. The two resulting equations are (see Appendix A for the algebraic details):

$$\vec{\nabla} \cdot \vec{E}(\mathbf{x}) = -\sigma_a(\mathbf{x})\phi(\mathbf{x}) + Q^0_{ri}(\mathbf{x}) + Q^0(\mathbf{x}) \quad (2.13)$$

$$\vec{\nabla}\phi(\mathbf{x}) = -3\sigma_{tr}(\mathbf{x})\vec{E}(\mathbf{x}) + 3Q^1_{ri}(\mathbf{x}) + 3Q^1(\mathbf{x}) \quad (2.14)$$

These equations introduce several new terms. Four are the 0th and 1st moments of the two source terms, $Q^0(\mathbf{x})$, $Q^0_{ri}(\mathbf{x})$, $Q^1(\mathbf{x})$ and $Q^1_{ri}(\mathbf{x})$, and the last is the reduced total extinction coefficient $\sigma_{tr}(\mathbf{x})$ discussed in Section 2.1.1. The final step in the derivation of the DE makes one further approximation: that the source terms are isotropic, i.e. $Q^1(\mathbf{x}) \equiv 0 \equiv Q^1_{ri}(\mathbf{x})$. With this simplifying assumption, Equation (2.14) can be used to eliminate $\vec{E}(\mathbf{x})$ from Equation (2.13) to yield the DE.

$$-\vec{\nabla} \cdot \Big(\kappa_d(\mathbf{x})\vec{\nabla}\phi(\mathbf{x})\Big) + \sigma_a(\mathbf{x})\phi(\mathbf{x}) = Q^0(\mathbf{x}) + Q^0_{ri}(\mathbf{x}) \quad (2.15)$$

In Equation (2.15), $\kappa_d(\mathbf{x})$ is the diffusion coefficient (see Section 2.1.1).

2.2.2 Rendering using the Diffusion Equation

Having derived the diffusion equation, the discussion turns to the most important topic in this chapter: using the DE for rendering. Rendering using the DE is a three step process.

Step 1 Estimate $Q_{ri}(\mathbf{x}, \vec{\omega})$

Step 2 Use $Q_{ri}(\mathbf{x}, \vec{\omega})$ to solve the Equation (2.15) for $\phi(\mathbf{x})$

Step 3 Compute $L(\mathbf{x}, \vec{\omega})$ from $\phi(\mathbf{x})$ using Equation (2.18)

Unfortunately, a useful rendering algorithm cannot be created from the derivation in the previous section alone. Implementing a rendering algorithm requires solving three additional problems.

Problem 1 Even for accurate solutions of the DE, its approximations can introduce objectionable artifacts in images. To hide these errors, the radiance must be carefully computed from the fluence. Section 2.2.2.1 describes the common rendering computation.

Problem 2 The solution to the DE is only fixed with the addition of a boundary condition specifying $\phi(\mathbf{x})$ on the medium's surface. Unfortunately, the physical boundary condition does not satisfy the DA and an approximate condition must be chosen. Section 2.2.2.2 describes three common choices: Dirichlet, Robin and extrapolated [SAHD95, HST*94]. The Dirichlet condition is used to derive the extrapolated condition which is then used to derive the dipole diffusion BSSRDF in Section 2.3. The Robin condition is the basis for the diffusive source boundary condition used by the FE algorithm in Chapter 5.

Problem 3 The reduced intensity source $Q_{ri}(\mathbf{x}, \vec{\omega})$ is difficult to compute accurately and an approximate model of the source must be chosen as well. Section 2.2.2.3 describes two choices: the boundary source model and the embedded

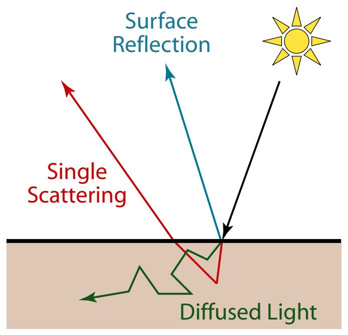


Figure 2.2: The incident light on the surface is split into three components, surface reflection, single scattering and diffused light.

source model. The embedded source model is used to derive the dipole diffusion BSSRDF and the more accurate boundary source model is used in Chapter 5.

2.2.2.1 Problem 1: Radiance Computation

For rendering, the surface radiance calculation must address two issues: *single scattering* and approximation errors. First, the effects of single scattering, the light that scatters exactly once in the medium, are often visually important to image appearance. More accurate images can be produced if, before the DE is applied, the incident light is split into the three components shown in Figure 2.2: the surface reflection, the single scattering component and the remaining *multiple scattering* component. The first two components are computed using more accurate algorithms [HK93] and the DE is applied only to the multiple scattering. This change alters the definition of the reduced intensity radiance $L_{ri}(\mathbf{x}, \vec{\omega})$ and Equation (2.11) becomes invalid. Consequently, the diffusion equation may no longer hold. However, this is generally ignored. Subtracting the total power of the single scattered radiance from the total power of $Q_{ri}(\mathbf{x}, \vec{\omega})$ ensures that this approximate computation conserves energy.

Second, since the DA only approximates the diffusive scattering, the a solution to the DE may contain non-physical 1st order, angular variation resulting entirely from the DA. For a rendering application, it is preferable to smooth this variation even if the smoothing is inaccurate in some cases. Moreover, in the limit of infinite scattering, the DA solution converges to the smoothed solution [Ish78]. For rendering, the surface radiance is approximated by the angular average of the diffuse radiance projected onto the interior boundary of the medium

$$L(\mathbf{x}, \vec{\omega}) = \frac{F_t(\eta, \vec{\omega})}{\pi} \int\limits_{(\vec{n}\cdot\vec{\omega})>0} L_d(\mathbf{x}, \vec{\omega})(\vec{\omega} \cdot \vec{n})\; d\vec{\omega} \tag{2.16}$$

This can be simplified by the substitution of the DA into Equation (2.16). For this Ishimaru [Ish78] derives a useful identity. If $L_d(\mathbf{x}, \vec{\omega})$ satisfies the DA then

$$\int\limits_{(\vec{s}\cdot\vec{\omega})>0} L_d(\mathbf{x}, \vec{\omega})(\vec{s} \cdot \vec{\omega})\; d\vec{\omega} = \frac{\phi(\mathbf{x})}{4} - \frac{\kappa_d(\mathbf{x})(\vec{s} \cdot \vec{\nabla})\phi(\mathbf{x})}{2} \tag{2.17}$$

Using Equation (2.17), Equation (2.16) becomes

$$L(\mathbf{x}, \vec{\omega}) = \frac{F_t(\eta, \vec{\omega})}{4\pi} \Big(\phi(\mathbf{x}) - 2\kappa_d(\mathbf{x})\big(\vec{n} \cdot \vec{\nabla}\big)\phi(\mathbf{x})\Big) \tag{2.18}$$

Finally, with Equation (2.18), a rendering algorithm can convert the fluence solution of the DE to radiance.

2.2.2.2 Problem 2: Boundary Conditions

Solving the DE requires an additional boundary condition that fixes the solution fluence on the surface of the medium. The physical condition should be derived analogous to the DE: scattering physics imposes an exact condition on the radiance at the boundary and then the DA is applied to derive a condition on the fluence.

Physical Boundary Condition The DE models the diffusive radiance $L_d(\mathbf{x}, \vec{\omega})$ and, by definition, this radiance must have scattered at least once in the medium. At the boundary it is not possible for light to have previously scattered so, physically, $L_d(\mathbf{x}, \vec{\omega})$ must be zero along all inward directions on $\partial\Omega$, i.e.

$$\forall(\vec{n} \cdot \vec{\omega}) < 0,\ L_d(\mathbf{x}, \vec{\omega}) = 0 \tag{2.19}$$

Since this boundary condition is discontinuous between the inward and outward directions, no function can simultaneously satisfy both this boundary condition and the DA. Thus, unfortunately, when using the DE, the physical condition can only be approximately satisfied. Three possible choices, Dirichlet, Robin and boundary extrapolation, are discussed here [SAHD95, HST*94].

Dirichlet Boundary Condition The Dirichlet condition is the simplest. It sets the fluence to zero on the boundary, $\phi(\mathbf{x}) \equiv 0$. Depending on how the DE is solved, the Dirichlet condition may or may not significantly effect the accuracy of the global solution. For applications other than rendering, such as medical imaging and scattering physics that are often concerned with radiance estimates far from the boundary, the Dirichlet condition works well [SAHD95, HST*94]. However, for the obvious reason that the boundary is the only externally visible part of the medium, the radiance estimates required by rendering algorithms are generally restricted to the boundary. Since the boundary condition has a large effect on accuracy in these regions, the Dirichlet condition cannot be used for rendering.

Robin Boundary Condition The physical boundary condition can be made compatible with the DA by approximating it with a new condition on only $\Gamma_d^{\text{in}}(\mathbf{x})$, the *net* inward radiant flux at the boundary.

$$\Gamma_d^{\text{in}}(\mathbf{x}) = \int\limits_{(\vec{n}\cdot\vec{\omega})<0} L_d(\mathbf{x}, \vec{\omega})(\vec{\omega} \cdot \vec{n})\, d\vec{\omega} \tag{2.20}$$

Robin boundary conditions are the general class of boundary conditions that use the net flux approximation and, depending on the requirements imposed, they can be used to generate accurate solutions (see [SAHD95, HST*94] for a 2D analysis and Chapter 5 for analysis for rendering applications). The Robin boundary condition

$$\phi(\mathbf{x}) + 2A(\eta)\kappa_d(\mathbf{x})\big(\vec{n}\cdot\vec{\nabla}\big)\phi(\mathbf{x}) = 0 \tag{2.21}$$

is commonly used by previous rendering algorithms [Sta95, JMLH01, HMBR05, LPT05, WZT*08]. It ensures that the net inward flux approximately matches the net outward flux reflected by Fresnel interface effects at the boundary. The factor

$$A(\eta) = \frac{1+F_{dr}(\eta)}{1-F_{dr}(\eta)} \tag{2.22}$$

$$F_{dr}(\eta) = \int\limits_{(\vec{n}\cdot\vec{\omega})>0} F_r(\eta,\vec{\omega})(\vec{\omega}\cdot\vec{n})\, d\vec{\omega} \tag{2.23}$$

approximately models these Fresnel effects. Here η is the relative index of refraction between the interior and exterior media and $F_{dr}(\eta)$ is the hemispherical average of the Fresnel reflection coefficient $F_r(\eta,\vec{\omega})$. The corresponding Fresnel transmission coefficients $F_{dt}(\eta) = 1 - F_{dr}(\eta)$ and $F_t(\eta,\vec{\omega}) = 1 - F_r(\eta,\vec{\omega})$ are also used throughout this thesis. In scattering physics, different Fresnel models result in different values for $A(\eta)$ (see [SAHD95] for a review). Equation (2.22) was originally derived in [GFB83]. The derivation of Equation (2.21) is discussed in Section 5.2 as part of the derivation of the more accurate, diffusive source boundary condition.

Boundary Extrapolation Boundary extrapolation assumes that the solution to the DE in the original domain Ω with the Robin boundary condition is equivalent to the projection of a solution from a larger, extrapolated domain with the Dirichlet boundary condition. Determining the position of the new boundary has two steps. First the fluence is extrapolated along the normal a distance δ outside Ω using a 1st order Taylor series centered at $\mathbf{x} \in \partial\Omega$.

$$\phi(\mathbf{x}+\delta\vec{n}) = \phi(\mathbf{x}) + \delta\big(\vec{n}\cdot\vec{\nabla}\big)\phi(\mathbf{x}) \tag{2.24}$$

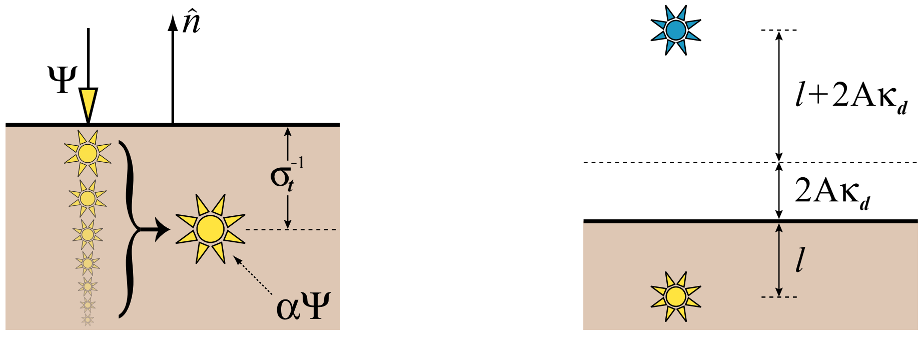


Figure 2.3: (a) The reduced intensity source for a single collimated beam of normally incident light with power Ψ can be approximated by a single point source embedded one mean free path $l = \sigma_t^{-1}$ in the material with power $\alpha\Psi$ where α is the albedo of the medium; (b) the dipole solution to the diffusion equation (DE) using the embedded source model and a second negative point source to satisfy the extrapolated boundary condition.

Next, the Robin boundary condition (Equation (2.21)) is used to eliminate the gradient term in Equation (2.24).

$$\phi(\mathbf{x} + \delta\vec{n}) = \left(1 - \frac{\delta}{2A(\eta)\kappa_d(\mathbf{x})}\right)\phi(\mathbf{x}) \tag{2.25}$$

Equation (2.25) implies that $\phi(\mathbf{x}+\delta\vec{n})$ will equal zero on an extrapolated, boundary normally projected outwards a distance $\delta = 2A(\eta)\kappa_d(\mathbf{x})$ from $\partial\Omega$. The derivation of the dipole diffusion BSSRDF uses boundary extrapolation (see Section 2.3).

2.2.2.3 Problem 3: Source Models

This final section discusses models for the reduced intensity source $Q_{ri}(\mathbf{x}, \vec{\omega})$. At any particular point, the exact value of $Q_{ri}(\mathbf{x}, \vec{\omega})$ could be computed by path tracing however this can be very expensive. By modeling the reduced intensity approximately this cost can be reduced. Two models are discussed here: the embedded source model, used by the dipole diffusion BSSRDF, and the boundary source model, used by the new FE algorithm in Chapter 5.

Embedded Source Model The embedded source model positions point sources inside the media whose emittance approximates the reduced intensity source. The model is motivated by the particular case, illustrated in Figure 2.3(a), of a collimated beam of normally incident radiance arriving on a semi-infinite slab of isotropically scattering[1], homogeneous material. In this case, a single point source closely matches the exact solution. To use the model for any other problem, the problem is approximated locally by an instance of the semi-infinite, homogeneous case.

Let $\mathbf{x}$ be a point on the boundary and $\Psi(\mathbf{x}) = L(\mathbf{x}, -\vec{n})$ be the radiance of the collimated incident beam then the reduced intensity source is exactly

$$Q_{ri}(\mathbf{x} - \delta\vec{n}, \vec{\omega}) = \frac{\sigma_s \Psi(\mathbf{x})}{4\pi} e^{-\sigma_t \delta} \tag{2.26}$$

where δ is some distance inward along the normal from $\mathbf{x}$. As illustrated in Figure 2.3(a), this source is an exponentially-decreasing linear source that lies along $-\vec{n}$. The total power of this source is

$$Q_{ri}^{\text{total}} = \int_0^\infty \int_{4\pi} Q_{ri}(\mathbf{x} - \delta\vec{n}, \vec{\omega})\ d\vec{\omega}\ d\delta = \frac{\sigma_s \Psi(\mathbf{x})}{\sigma_t} = \alpha\Psi(\mathbf{x}) \tag{2.27}$$

where α is the albedo of the material. The embedded source model approximates this linear source with a single, isotropic point source of equal power. The optimal location for the point source is the power-weighted average depth of the linear source in the media.

$$\frac{1}{Q_{ri}^{\text{total}}} \int_0^\infty \mathbf{x} \int_{4\pi} Q_{ri}(\mathbf{x} - \delta\vec{n}, \vec{\omega})\ d\vec{\omega}\ dd = \frac{1}{Q_{ri}^{\text{total}}} \frac{\sigma_s \Psi(\mathbf{x})}{\sigma_t^2} = \frac{1}{\sigma_t} \tag{2.28}$$

To use the embedded source model in more complicated cases, the radiance at the surface is sampled at many points. At each point, the total incident radiance is

[1]This restriction is relatively minor since, when using the DE, non-isotropic materials are approximated by isotropic ones using reduced scattering coefficients (see Section 2.1.1).

approximated by a single collimated beam with power

$$\Psi(\mathbf{x}) = \int_{4\pi} L(\mathbf{x},\vec{\omega}) F_t(\eta,\vec{\omega})(\vec{\omega}\cdot\vec{n})\, d\vec{\omega} \tag{2.29}$$

At each sample, an embedded source is created by assuming the local media is a homogeneous, semi-infinite slab with the scattering parameters of the surface sample point. If many samples are created, the resulting model is a reasonable estimate of the real reduced intensity source [DJ07].

Boundary Source Model Unlike the embedded source model, the boundary source model does not try to create a volumetric representation of the reduced intensity source. Instead it approximates $Q_{ri}(\mathbf{x},\vec{\omega})$ by assuming it arrives directly at the boundary. Because the reduced intensity source decays exponentially away from the boundary, this is reasonable since most of the power in $Q_{ri}(\mathbf{x},\vec{\omega})$ lies very near the boundary compared to the size of the domain. For example, in the collimated example in the last section, the powered weighted average depth of the infinite line source was only a single mean free path. Once the reduced intensity source is assumed to lie only on the boundary, it can be incorporated into the boundary condition. Chapter 5 develops an accurate method of solving the DE. An essential part of that solution is the diffusive boundary source condition (Equation (5.5)) which combines the boundary source model with a Robin boundary condition (Equation (2.21)) to create an accurate, DE formulation for complex heterogeneous materials.

2.3 Derivation of the Dipole Diffusion BSSRDF

Because it is used by most previous subsurface rendering algorithms and the new scalable algorithm in Chapter 4, this last section derives the dipole diffusion

BSSRDF. The dipole diffusion BSSRDF is based on a solution to the diffusion equation for the simplified configuration already discussed for the embedded source model (Section 2.2.2.3): a collimated, normally incident beam illuminating a semi-infinite, isotropic, homogeneous slab. Given the theory already discussed in the previous section, the derivation of the dipole diffusion BSSRDF has only three additional steps:

1. The DE is solved for a single point source—the embedded, reduced intensity source approximation of the collimated beam—in an infinite medium.
2. This solution is converted into the semi-infinite solution by using the extrapolated boundary condition and a second, negative pseudo-source.
3. The semi-infinite solution is differentiated and converted into an approximate BSSRDF.

The first step references Ishimaru who derives that, for an infinite, isotropic, homogeneous material, the DE has a analytic solution for the embedded point source:

$$\phi(\mathbf{x}) = \frac{\Psi(\mathbf{x})}{4\pi\kappa_d} \frac{e^{-\sigma_{et} r(\mathbf{x})}}{r(\mathbf{x})} \tag{2.30}$$

where $\Psi(\mathbf{x}')$ is the incident radiance that generates the embedded source at $\mathbf{x}$, $\sigma_{et} = \sqrt{3\sigma_a\sigma_t}$ is the *effective transport coefficient* and $r(\mathbf{x})$ is the distance from the point source to $\mathbf{x}$. The second step imposes a boundary condition on the infinite solution to create a semi-infinite solution. Using the extrapolated boundary condition (see Section 2.2.2.2), this is trivial: a second, negative power pseudo-source is positioned above the boundary as illustrated in Figure 2.3(b). Since all points on the extrapolated boundary are equally distant from both sources, the fluence on the extrapolated boundary is zero as required. Thus, the solution to the semi-infinite case is the composition of the two point source solutions (i.e. a *dipole* *source*).

$$\phi(\mathbf{x}) = \frac{\alpha\Psi(\mathbf{x})}{4\pi\kappa_d}\left[\frac{e^{-\sigma_{et}r_p(\mathbf{x})}}{r_p(\mathbf{x})} - \frac{e^{-\sigma_{et}r_n(\mathbf{x})}}{r_n(\mathbf{x})}\right] \tag{2.31}$$

$r_p(\mathbf{x})$ and $r_n(\mathbf{x})$ are the distances from $\mathbf{x}$ to the positive and negative sources respectively. Finally the third step uses Equation (2.31) to create an approximate BSSRDF. A BSSRDF is an estimate the fraction of radiance incident at $\mathbf{x}'$ from $\vec{\omega}'$ that leaves the medium at $\mathbf{x}$ in $\vec{\omega}$. Since at both the inward and outward interfaces the radiance will experience Fresnel interface effects, a basic BSSRDF has the form

$$S(\mathbf{x}, \vec{\omega}, \mathbf{x}', \vec{\omega}') = F_t(\eta, \vec{\omega})R(\mathbf{x}, \vec{\omega}, \mathbf{x}', \vec{\omega}')F_t(\eta, \vec{\omega}) \tag{2.32}$$

where $R(\mathbf{x}, \vec{\omega}, \mathbf{x}', \vec{\omega}')$ is the ratio of internal subsurface transmittance from entry to exit. The dipole diffusion BSSRDF estimates $R(\mathbf{x}, \vec{\omega}, \mathbf{x}', \vec{\omega}')$ as ratio of the exitant normal gradient of the fluence to the incident, differential embedded source power.

$$R(\mathbf{x}, \vec{\omega}, \mathbf{x}', \vec{\omega}') = \frac{-\kappa_d(\vec{n} \cdot \vec{\nabla})\phi(\mathbf{x})}{d\Psi(\mathbf{x})} \tag{2.33}$$

This ratio can be directly computed by differentiating Equation (2.31) and rearranging terms. In the end the result, the dipole diffusion BSSRDF, only depends on $r = \left\|\mathbf{x} - \mathbf{x}'\right\|$

$$R(\mathbf{x}, \vec{\omega}, \mathbf{x}', \vec{\omega}') = R(r) = \frac{\alpha}{4\pi}\left[C_1\frac{e^{-\sigma_{et}r_p(r)}}{r_p(r)^2} + C_2\frac{e^{-\sigma_{et}r_n(r)}}{r_n(r)^2}\right] \tag{2.34}$$

where

$$\begin{aligned} z_p &= \frac{1}{\sigma_{tr}} & z_n &= z_p + 4A(\eta)\kappa_d \\ r_p(r) &= \sqrt{r^2 + z_p^2} & r_n(r) &= \sqrt{r^2 + z_n^2} \\ C_1 &= z_p\left(\sigma_{et} + \frac{1}{r_p(\mathbf{x})}\right) & C_2 &= z_n\left(\sigma_{et} + \frac{1}{r_n(\mathbf{x})}\right) \end{aligned}$$

2.4 Summary

The goal of the chapter was to give a complete introduction to the scattering theory used by rendering algorithms. The volumetric radiative transfer equation exactly models light transport (Section 2.1) but Monte Carlo path tracing algorithms that solve it are too expensive for practical applications. Most previous work, surveyed in the next chapter, and the algorithms in this thesis rely on an approximate model of subsurface scattering, the diffusion equation, derived in Section 2.2. The most important part of this chapter, Section 2.2.2, introduced the three steps that outline a basic rendering algorithm using the diffusion equation. Each of the new algorithms in this thesis extends this basic algorithm to larger and more difficult problems. Chapter 4 demonstrates that, even when the DE solution is approximated cheaply be the dipole diffusion BSSRDF (Section 2.3), the required pre-computation of the reduced intensity source $Q_{ri}(\mathbf{x}, \vec{\omega})$ can become an impractical bottleneck in large complex scenes. The new scalable algorithm solves this problem by trying to detect when source computation is essential and compute the reduced intensity source only when necessary. Then, Chapter 5 develops the first, accurate and efficient algorithm for solving the diffusion equation in general, complex heterogeneous materials. It demonstrates that for many materials, the new heterogeneous algorithm is comparable to the accuracy of path tracing with orders of magnitude savings in cost. An essential part of this new heterogeneous algorithm is the diffusive source boundary condition that combines the Robin boundary condition (Section 2.2.2.2) with the boundary source model (Section 2.2.2.3) to create an accurate model of heterogeneous boundary effects.

CHAPTER 3

PREVIOUS WORK

Previous work in subsurface rendering can be divided into three categories: Monte Carlo (MC) algorithms, dipole diffusion algorithms and heterogeneous algorithms. The goal of this chapter is to identity the limits of these previous algorithms to highlight the contributions of the two new algorithms presented in the next chapters. To begin this discussion, it is useful to summarize the limitations of each class as a whole.

3.1 Limitations of Previous Work

Monte Carlo algorithms are limited by their path-by-path approach. Regardless of whether an image contains subsurface scattering or not, a path tracing algorithm must trace a minimum number of paths to ensure the image is accurate. When subsurface scattering is present, tracing a single path requires simulating hundreds of subsurface interactions. These new simulation steps sufficiently raise costs such that all path-based MC algorithms, even those that trace close to the minimum number of paths, become impractically expensive. Since the path-by-path approach is inherent to MC algorithms, practical considerations have pushed most research towards approximate approaches, like the dipole diffusion BSSRDF, that can solve for the aggregate behavior of many paths.

However, dipole diffusion algorithms have limited scalability. Recalling the basic diffusion equation rendering algorithm discussed in Section 2.2.2, a dipole renderer must first compute the reduced intensity source then solve the DE (using the dipole diffusion BSSRDF) and compute an outgoing radiance estimate. In all previous work, this algorithm is implemented in one of two ways. The first, based on [JMLH01], computes an estimate of the reduced intensity source for each outgoing radiance calculation. The second, based on [JB02], leverages the independence of the reduced intensity source from any particular radiance calculation to avoid this re-computation. The improved algorithm has two passes. The first pass pre-computes the reduced intensity source once and the second pass reuses it many times for all outgoing radiance estimates. Unfortunately, the cost of both algorithms significantly increases as the scene grows in size and complexity. In the first approach, the effort wasted on repeated source calculations only grows. In the two-pass algorithms, the cost of the initial source pre-computation becomes expensive. The problem is that basic, two-pass algorithms conservatively compute an excessively, detailed representation of the reduced intensity source. The new scalable algorithm presented in Chapter 4 fixes this scalability limitation. It both avoids repeated source computations and judiciously evaluates the reduced intensity source to ensure that no extra source estimates are performed.

Finally, there is no previous algorithm for the high-quality rendering of general, physical models of heterogeneous materials that is faster than path tracing. Previous approaches for heterogeneous materials are capture and re-render systems. These systems use photographs of real objects to record models of the subsurface scattering. Using these recorded models, these systems can re-render the captured materials in new environments. However, the capture process corrects for inaccuracies inherent in the re-rendering algorithms by altering the recorded model parameters. As a consequence, these rendering algorithms can produce high-quality images but only of the recorded materials. The most recent work in this category, Wang et al., tried to avoid this problem. However, as will be discussed in detail in Section 5.5.6, their approach limits the scattering geometry and produces lower quality images when not rendering captured materials. The new finite element algorithm presented in Chapter 5 is the first, general, high-quality and practical rendering algorithm for heterogeneous materials.

The rest of this chapter has four sections. The first reviews Monte Carlo algorithms and the second reviews dipole diffusion algorithms. The third section discusses previous work on heterogeneous rendering including an overview of work from medical imaging that addresses a similar problem. Finally, the chapter closes with a summary of the key elements of this work relevant to the new algorithms presented in the rest of this thesis.

3.2 Monte Carlo Algorithms

Because solving the volumetric radiative transfer equation requires only simple extensions to basic surface path tracing algorithms, the earliest subsurface rendering algorithms were path tracers. Hanrahan and Krueger [HK93] used path tracing to render layered translucent materials, such as leaves and skin, and Jensen et al. [JLD99] used it to simulate the appearance of wet materials. Pharr and Hanrahan [PH00] later formalized the volumetric path tracing algorithm using operator theory.

Two algorithms, photon mapping [JC98, DEJ*99] and Metropolis sampling [PKK00], reduce the cost of the basic path tracer. Photon mapping accelerates rendering by caching and reusing some path segments [JC98, DEJ*99]. The photon mapping algorithm has two stages. First it generates long paths starting at the light sources and stores the position and radiance of all their interactions in a large index called the *photon map*. Then short paths are generated from the camera. The radiance arriving at the end of each short path can be estimated using the photon map and then added directly to the image. By reusing a relatively small number of long paths, the total number of simulated interactions is dramatically reduced.

Metropolis sampling focuses effort on the simulation of paths that contribute most to the image [PKK00]. The algorithm randomly samples a starting path and then mutates the path to generate new ones. The size and probability of the mutation depend on the amount of radiance that travels along the original path. When the algorithm finds important paths, it saves time by sampling the nearby paths densely to quickly add radiance to the image. Unfortunately, neither of these techniques improves performance enough to make Monte Carlo algorithms practical for subsurface rendering.

3.3 Dipole Diffusion Algorithms

The majority of subsurface rendering algorithms are based on the dipole diffusion BSSRDF. All of these algorithms rely on the results from two papers: one by Jensen et al. [JMLH01] and a follow-up work a year later by Jensen and Buhler [JB02]. When reading this section it is important to keep in mind, that none of the following algorithms address the scalability problem solved in the next chapter. Though one algorithm may be more accurate or more efficient than another, the scalability of these dipole algorithms is fundamentally limited by their structure. For any algorithm below, there is a sufficiently large or complex scene where it becomes impractical. The new scalable algorithm presented in this thesis solves this problem by using a new unified rendering structure that remains practical even if the scene grows in size and complexity. The rest of this section breaks review of dipole diffusion algorithms into three subsections. The first subsection discusses the basic algorithms [JMLH01, JB02] and then the next two subsections respectively discuss improvements in the accuracy and the speed of these two basic approaches.

3.3.1 Basic Algorithms

Jensen et al. [JMLH01] introduced the dipole diffusion BSSRDF and presented an accelerated path tracer that used it. Their algorithm worked just like a basic path tracer. However, when it encountered a subsurface scattering material, instead of simulating the subsurface interactions, their algorithm sampled the BSSRDF. Since it avoided the simulation of many subsurface interactions, their path tracer was much faster than previous, pure Monte Carlo algorithms.

However, Jensen and Buhler [JB02] realized that a significant part of the computation could be cached and reused. To compute the light that scatters through the material, the dipole path tracer must recursively estimate the radiance incident on the material's surface. Jensen and Buhler realized that, since these recursive calculations are independent of any subsurface path, they could be computed once and reused. Their two-pass algorithm pre-computes the radiance at many surface samples and then gathers the subsurface scattering from these samples in the second pass. Because of this reuse and an optimization that clusters similar samples during the radiance calculations in the second pass, their algorithm is very fast for simple scenes and is used in many rendering applications. However, as will be discussed in detail in Chapter 4, the algorithm becomes inefficient as scenes grow larger.

3.3.2 More Accurate Algorithms

The accuracy of these original dipole renderers has been improved by three techniques: a new thin-layer BSSRDF, an improved embedded source model called photon diffusion, and hybrid path tracing and dipole diffusion algorithms.

3.3.2.1 Thin-layer BSSRDF

Donner and Jensen [DJ05] created a new BSSRDF, similar to the basic dipole diffusion BSSRDF, but more accurate for thin-layer geometries such as skin and leaves. To create the original dipole diffusion BSSRDF (see Section 2.3), the derivation converted a solution for an infinite medium into a solution for a semi-infinite medium by introducing a second, pseudo-source to satisfy the boundary condition. In the same way, the dipole BSSRDF can be converted into a thin-layer BSSRDF by introducing yet more sources to satisfy the second boundary condition at the bottom of the layer. Donner and Jensen used this new BSSRDF to implement a new accelerated path tracer for rendering layered, skin-like materials.

3.3.2.2 Photon Diffusion

Photon diffusion [DJ07] improves the accuracy of the embedded source model. The basic dipole solution used an embedded source model that converts an incoming collimated beam into a single dipole source. The photon diffusion algorithm converts the incoming beam into several dipole sources. Because the new source has a larger spatial extent, the representation is a better model for cases where the beam is not normally incident. Additionally the algorithm can approximate volume shadows by blocking the generation of some of the dipole sources and handle paths that re-enter the subsurface material several times by allowing the source generation algorithm to exit and re-enter to the material as well. Their rendering method is the same two-pass algorithm as [JB02]. All the embedded sources are generated in an initial pass and the second pass gathers the subsurface contributions from these sources to produce an image.

3.3.2.3 Hybrid Algorithms

Finally, two methods [LPT05, CTW*04] combine accurate subsurface path tracing with the dipole diffusion to create hybrid methods. Both algorithms divide the subsurface scattering object into two regions: a thin, outer shell and an inner core. The algorithms use accurate, subsurface path tracing in the outer shell but approximate transport in the inner core using the dipole diffusion BSSRDF. Li et al. [LPT05] demonstrate that this thin-shell model can significantly improve quality when rendering thin geometry. Chen et al. [CTW*04] model a limited type of tile-able heterogeneity in the thin-shell by first pre-computing a shell texture function (STF), a path-traced BSSRDF for the heterogeneous tile. The expensive heterogeneous path tracing in the thin shell is accelerated by sampling the STF to directly step from tile to tile.

3.3.3 Faster Algorithms

Fast dipole diffusion algorithms are designed for specialized hardware, the graphics processing unit (GPU). Using the GPU these algorithms can render images fast enough to support real-time interactions with the scene. There are two basic methods: pre-computed radiance transfer (PRT) methods and interactive two-pass methods.

3.3.3.1 Pre-computed Radiance Transfer

Pre-computed radiance transfer algorithms estimate the transfer function [SKS02, WTL05]. The transfer function describes the complete radiance transport from light sources through the scene to the outgoing radiance from all surfaces. PRT algorithms pre-compute an estimate of the transfer function and store it in a spherical basis. Using this pre-computed data, an image can be computed quickly by convolving the transfer function with the light source function using the GPU. The initial pre-computation is not interactive. It estimates the transfer function using the two-pass dipole diffusion algorithm [JB02]. But, given a pre-computed transfer function, the lighting can be updated in real-time. In a later work, Sloan et al. [SLS05] showed that deformable, dipole diffusion transfer functions can be created with extra pre-computation.

3.3.3.2 Interactive Two-pass Algorithms

There have been many real-time implementation of the basic two-pass algorithm [JB02] using the GPU [MKB*03b, HBV03, MKB*03a, CHH03, HV04, DS03, LGB*02]. The basic outline of all these algorithms is the same. Given a subsurface scattering object, they choose a fixed set of surface samples where the outgoing subsurface radiance will be computed. Then they link each outgoing sample to a small set of incoming surface samples and the algorithm estimates the dipole diffusion BSSRDF for each link. To compute the image, these algorithms compute the radiance at all the incoming samples using a fast GPU algorithm and then they use the links and BSSRDF values to transfer the radiance at the incoming samples to the outgoing samples.

These algorithms fall into two groups. The first group [LGB*02, CHH03, HBV03, HV04] require fixed scattering geometry. In this case the links and BSSRDF values never change and can be pre-computed and reused. Since link selection is not part of the real-time computation, these algorithms can improve quality by spending more effort selecting useful links and computing the BSSRDF estimates more accurately. The second group [DS03, MKB*03b, MKB*03a] supports full real-time, deformation of the geometry and alteration of the material terms. For these algorithms, the links and the BSSRDF values must be computed interactively each frame so the subsurface scattering computations are performed more approximately. However, because of this, they can support fully dynamic scenes and require no expensive pre-computation.

3.4 Heterogeneous Materials

The most recent works in subsurface rendering address the difficult problem of heterogeneous scattering. The new finite element algorithm in Chapter 5 is the first general rendering algorithm in this class. Given a generic, physical description of a heterogeneous scattering material, either measured or synthetic, it produces in a few minutes a high-quality image comparable to Monte Carlo methods that take hours. All of the previous algorithms discussed in this section are capture and re-render systems. They use photographs of real objects to capture material models that can then be re-rendered in novel environments. They all produce high-quality images, but they rely on capture process to correct rendering inaccuracies. They either cannot render, or cannot render accurately, general material models.

This section will discuss these previous works in three subsections. The first subsection discusses algorithms that capture materials using a specialized material model that can only be created through capture process itself. The second subsection discusses the most recent and ambitious work, Wang et al. [WZT*08], that captures materials using a general model, but cannot render general materials accurately. The final section discusses work outside of computer graphics in the medical imaging field of optical tomography that solve problems similar to the one addressed in Chapter 5.

3.4.1 Specialized Capture Algorithms

Two capture systems represent the captured material by a compressed collection of photographs. Goesele et al. [GLL*04] use a video camera to record thousands of images real objects. Then they compress this data into a smaller index of transport coefficients within the object and material. They can then re-render the original object in novel lighting environments by querying this index to calculate the scattering between different regions. Peers et al. [PvBM*06] present a similar technique but they factor out the geometry from the captured data so the measured material can be inserted into novel objects.

Tong et al. [TWL*05] use the dipole diffusion BSSRDF to approximate captured heterogeneous materials. Their rendering algorithm is essentially the two-pass algorithm [JB02] from the previous section, but they modulate the incoming surface radiance samples and the outgoing scattered radiance estimates using two separate texture images. In the capture process, they compute these images so that the renderer reproduces photographs of real objects. Then they can approximate these materials in new geometry by mapping these textures onto the new object's surface.

Finally, an ambitious recent work by Donner et al. [DWd*08] uses a similar technique to capture and model human skin. Their capture method is based on the thin layer BSSRDF [DJ05] described above (see Section 3.3.2.1). Their algorithm models human skin as a series of thin layers and, within each layer, they approximate scattering homogeneously using the thin-layer BSSRDF. To add the heterogeneous appearance of skin, they modulate the scattering between layers using detailed, captured textures. The result is a very detailed model of the skin material that can be applied to any object. Their renderings are very high-quality but their model is valid only for layered skin-like geometries.

3.4.2 Generalized Capture

Wang et al. [WZT*08] generalizes these specialized capture algorithms. The main component of their system is a fast finite difference solution of the heterogeneous diffusion equation. This finite difference method was originally proposed by Stam [Sta95] and implemented for homogeneous materials by Haber et al. [HMBR05]. Their capture system uses the adjoint conjugate gradient method to solve for an input grid such that their finite difference rendering algorithm reproduces a series of photographs of a real object. This capture method requires solving many instances of the forward rendering problem and, to make this practical, they implement the finite difference solver using the GPU. Their end result is an interactive system that can redisplay and edit the captured materials in real-time. Their system can apply the captured materials to new objects by warping the material grid into a new geometry.

The finite difference renderer from Wang et al. is closely related to the new heterogeneous finite element algorithm presented in Chapter 5. But, as a general rendering solution for heterogeneous materials, the analysis in that chapter will reveal two significant limitations. First, the warping operation required to deform the finite difference grid into novel geometry is expensive to compute and cannot be easily applied to all models without introducing error. Moreover, the warping is expensive; Wang et al. note that good grid generation requires manual fitting. Second, and most importantly, the finite difference algorithm has limited accuracy. For the original capture and re-render system this is acceptable because these accuracy issues are naturally corrected by the capture process. However, they fundamentally limit the quality of the solution for general materials.

3.4.3 Optical Tomography

The finite element method and the diffusion equation, the basic tools used to develop the new heterogeneous rendering algorithm in Chapter 5, are also general tools of mathematics and physics. Outside of computer graphics, many other fields have considered similar scattering problems. The closest work is the medical imaging field of optical tomography. The optical tomography problem is similar to the capture problem addressed by Wang et al.: given a series of photographs of human tissue lit by visible light determine the scattering properties of the tissue. All the techniques used in subsurface rendering have also been used in optical tomography (see [GHA05, Kol01] for a review of the state of the art). However, the optical tomography problem is different from the rendering problem in three ways. First, in optical tomography, the scattering solvers are interested in accurate solutions at only a few points whereas a rendering algorithm needs accurate solutions across a whole image. Second, rendering algorithms must work for a wide range of material properties whereas their counterparts must consider only the small range of parameters found in human tissue. Finally, optical tomography applications are much less concerned with efficiency. The goal of the analysis in Chapter 5 is to demonstrate that the new finite element algorithm is an efficient, high-quality solution for the general class of rendering applications.

3.5 Summary

This chapter highlights two limitations of previous work in subsurface rendering. First, dipole diffusion algorithms rely on a basic two-pass structure that limits their scalability. Though this structure is useful for rapidly rendering small scenes and many algorithms have considered more accurate and more efficient versions of this basic approach, the next chapter will demonstrate that there are always larger scenes where these techniques become impractical. The new scalable algorithm corrects this problem with a new unified rendering structure that performs well in the large, complex scenes difficult for previous work. Second, the recent advances in heterogeneous rendering are limited to capture and re-rendering applications that cannot render general, physical material descriptions. The most advanced of these algorithms, Wang et al., uses a finite difference technique to partially overcome this limitation. However, the analysis in Chapter 5 will demonstrate that this algorithm is difficult to apply to general geometric models and has limited accuracy as a solution for general heterogeneous materials. The new finite element algorithm in this thesis is the first efficient, general solution for the high-quality rendering of heterogeneous subsurface scattering.

CHAPTER 4

SCALABLE HOMOGENEOUS SCATTERING

At its core a rendering algorithm computes an integral. Scalable rendering algorithms increase performance by approximating these integrals. They first determine the relative importance of different parts of the integration domain and then they balance performance and accuracy by estimating the contributions from unimportant regions more approximately. Since they carefully match computational effort to relative importance, scalable algorithms can compute complex integrals with little increase in cost. Thus they can render larger scenes with more detailed geometry and more complex lighting. However, for subsurface rendering, the BSSRDF makes determining the relative importance of different regions of the subsurface rendering integral a challenge. This chapter addresses this challenge with a new scalable, dipole diffusion rendering algorithm for large, complex scenes with subsurface scattering materials.

4.1 Problem

To make the problem solved by the new algorithm more clear, it is useful to consider the most commonly-used and efficient dipole diffusion rendering algorithm in previous work: Jensen et al.'s two-pass method [JB02]. In an initial pass, this method computes an estimate of the reduced intensity source by sampling the radiance at many points on the scattering object's surface. Then in a second pass, the algorithm gathers the subsurface scattering from these samples by using the dipole diffusion BSSRDF. If most of the samples computed in the first pass are used when gathering scattering in the second pass, the two-pass algorithm is efficient. However, sometimes the algorithm computes extra less important samples. If it computes too many, the two-pass pre-computation becomes expensive.

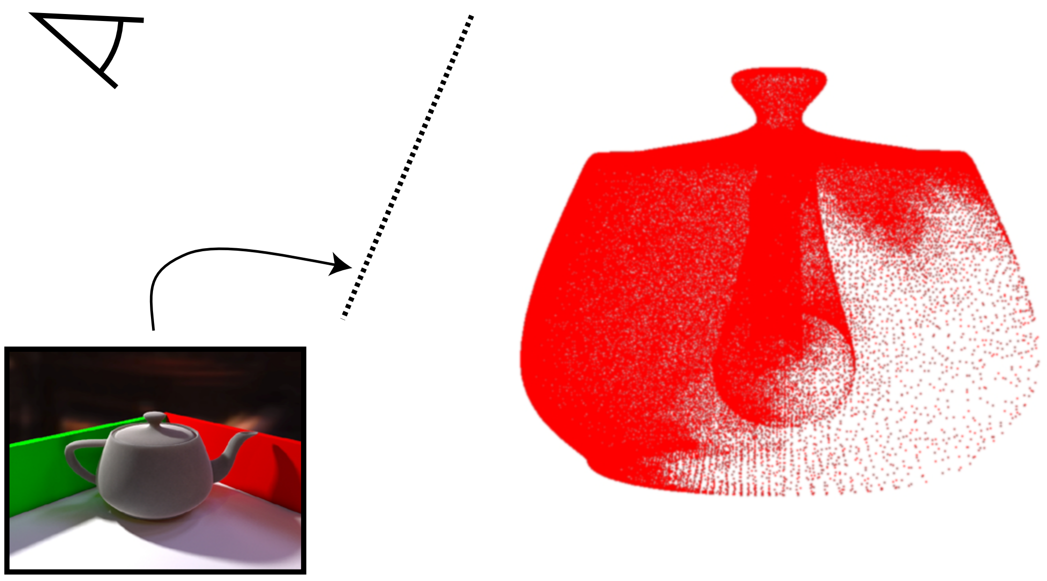


Figure 4.1: Red dots indicate the pre-computed reduced intensity source samples that were needed to compute the image of the teapot in the lower left using the two-pass dipole diffusion algorithm by Jensen and Buhler [JB02]. Notice that even though the samples are pre-computed uniformly across the entire surface, only a subset of these samples are ultimately used.

Figure 4.1 shows, as a red dot, every reduced intensity source sample used during the second pass when rendering the teapot image in the lower left-hand corner. As illustrated by the sparse distribution of dots, there are regions, such as back-facing areas or surfaces occluded in the camera, where fewer reduced intensity source samples are actually needed. However, because the algorithm works in two passes, it cannot identify these regions and it must initially, pre-compute the radiance in all regions equally. This can waste considerable effort in large, complex scenes where the algorithm's cost becomes dominated by an impractical, initial pre-computation. The new unified, scalable, dipole diffusion rendering algorithm described in this chapter combines the two-passes. It avoids unnecessary pre-computations and can therefore render large scenes that were impractical when using the basic two-pass approach.

The rest of this chapter has four sections. The first section introduces the basics of scalable rendering by introducing *Lightcuts* (LC) [WFA*05] and *Multidimensional Lightcuts* (MDLC) [WABG06], previous scalable rendering algorithms for problems without subsurface scattering. The second section builds upon these methods to create the new scalable subsurface rendering algorithm and then Section 4.4 discusses additional details required to implement the new algorithm efficiently. Finally, the last section in the chapter analyzes the scalability of the new algorithm on a sequence of progressively larger and more complex scenes. This analysis demonstrates that the new algorithm significantly improves performance when rendering large scenes with detailed subsurface scattering geometry and complex lighting.

4.2 Introduction to Scalable Rendering

In the introduction of this thesis, Section 1.1 discussed the basic, pixel-by-pixel rendering process. Through each pixel in an image, some region of the scene $\mathcal{E}$ is visible. To compute the value of a pixel, the rendering algorithm computes several *eye samples* $\mathbf{e}$ within $\mathcal{E}$ and computes the radiance $L^{\text{cam}}(\mathbf{e})$ that leaves each eye sample towards the camera. At each eye sample $\mathbf{e}$, this radiance equals

$$L^{\text{cam}}(\mathbf{e}) = \int\limits_{(\vec{n}\cdot\vec{\omega})>0} f_r^{\text{cam}}(\mathbf{e},\vec{\omega}) L(\mathbf{e},\vec{\omega})(\vec{\omega}\cdot\vec{n})\, d\vec{\omega} \tag{4.1}$$

where $f_r^{\text{cam}}(\mathbf{e},\vec{\omega})$ is a restricted bidirectional reflectance distribution function (BRDF) that gives the fraction of the radiance incident from $\vec{\omega}$ at $\mathbf{e}$ that reflects towards the camera.

The Lightcuts (LC) algorithm estimates Equation (4.1) by approximating the radiance function as a large collection of *light samples* $\mathbf{l}$. With the light samples, the integral reduces to a sum. However, instead of computing this large sum directly, which would be prohibitively expensive, the Lightcuts algorithm estimates this sum by partitioning the light samples into clusters and approximating the sum within each cluster. A valid non-overlapping partitioning, a *cut*, is created by traversing a pre-computed hierarchy of potential clusters.

Multidimensional Lightcuts (MDLC) generalizes the Lightcuts algorithm. Instead of estimating the radiance at only a single eye sample, MDLC estimates the integrated radiance over the entire pixel domain $\mathcal{E}$ in a unified computation. For this problem, there will be several eye samples and, instead of clustering individual light samples, MDLC clusters pairs of eye and light samples. Similarly, the new scalable subsurface rendering algorithm generalizes the MDLC algorithm. The new algorithm introduces a new set of samples, *irradiance samples* and clusters triples of eye, irradiance and light samples. To build up to this new algorithm, this section first reviews the LC and MDLC algorithms.

4.2.1 Lightcuts

The basic Lightcuts algorithm has four steps.

1. Discretize the radiance function into light samples
2. Estimate Equation (4.1) as a sum over light samples
3. Estimate this sum by another, much smaller, sum over clusters of samples
4. Choose a valid clustering of light samples by selecting a lightcut

First, an efficient light sample generation process is an essential part of the Lightcuts algorithm; however, describing this algorithm requires introducing specific light source models that are not relevant anywhere else in this thesis. Thus here it is only briefly described. The basic sampling algorithm generates samples directly on all light sources and the environment map [Deb02]. Afterward, the algorithm traces photons outwards from these initial samples to generate additional indirect lighting samples that approximate the radiance in all other areas of the scene.

These generated light samples are not modeled uniformly but have slightly different traits depending on their method of generation. Since these distinctions have only computational importance, this chapter uses a more general and unified light sample model with in [WFA*05]. Each light sample has a radiant intensity I and, given an eye sample $\mathbf{e}$ and a light sample $\mathbf{l}$, the form factor $F_{\mathbf{el}}$ between them can be calculated. The form factor gives the fraction of the radiant intensity emitted by l that is reflected at $\mathbf{e}$ towards the camera. For more details on the specifics of the light samples see the original work by Walter et al. [WFA*05]. For the rest of this chapter, it is assumed that a useful set of light samples can be generated in a few seconds during an initial pre-process.

Moving on to the second step, given the light samples, the radiance $L^{\text{cam}}(\mathbf{e})$ in Equation (4.1) can be expressed as a sum over the set of all light samples $\mathbb{L}$

$$L^{\text{cam}}(\mathbf{e}) = \sum_{i \in \mathbb{L}} F_{\mathbf{e}i} I_i \tag{4.2}$$

For the next step, assume that all light samples $\mathbb{L}$ are partitioned into a set $\mathbb{L}_{\mathbb{C}}$ of clusters $\mathbb{C}_{\mathbf{l}}$ of light samples. Then the sum in Equation (4.2) can be approximated using these clusters by reusing a single form factor for all the samples in each cluster.

$$L^{\text{cam}}(\mathbf{e}) = \sum_{\mathbb{C}_{\mathbf{l}} \in \mathbb{L}_{\mathbb{C}}} F_{\mathbf{el}} \Big[\sum_{i \in \mathbb{C}_{\mathbf{l}}} I_i \Big] \tag{4.3}$$

The chosen form factor $F_{\mathbf{el}}$ is the form factor between the eye sample and the single *representative* light sample $\mathbf{l}$ of the cluster. To illustrate the efficiency difference between the preceding two sums, note that in typical scenes there are usually several hundred thousand light samples; however, for most eye samples, only a few thousand clusters are required to compute an accurate estimate using Equation

(4.3). Since the total radiant intensity of all possible clusters can be pre-computed and stored, the cost using Equation (4.3) is usually orders-of-magnitude less than the cost using Equation (4.2).

The key to this performance gain is the selection of a good partitioning of the light samples into clusters. Lightcuts accomplishes this using a binary tree of potential clusters. This tree is created at the beginning of rendering along with the original light samples. The LC algorithm chooses a partition by selecting a cut through this tree. A cut is a set of nodes such that, along every path from the root to a leaf, exactly one node on the path lies in the cut. Because of this property, every cut is a valid non-overlapping partitioning of samples into clusters.

To select a cut, the LC algorithm starts with the trivial one, the root cluster of the cluster tree. It then iteratively refines the cut by removing the highest error cluster and replacing it with its two children clusters from the cluster hierarchy. Of course, this requires an estimate of the error of each cluster's approximation. Because the contribution of a cluster to the sum cannot be negative, the error caused by the cluster approximation must be less than the maximum possible contribution of the cluster. This maximum contribution can be conservatively estimated by replacing the form factor term in Equation (4.3) by its upper bound.

$$\left\| L_{\mathbb{C}}^{\text{true}} - L_{\mathbb{C}}^{\text{est}} \right\| \leq F_{\mathbf{el}}^{(ub)} \left[\sum_{i \in \mathbb{C}_{\mathbf{l}}} I_i \right] \tag{4.4}$$

The cut refinement iteration ends when the cut satisfies a perceptual metric called Weber's law [Bla72]. Weber's law notes that humans cannot recognize differences in intensity less than a 2% of the baseline intensity. Thus the LC algorithm stops when the error of the highest error node is less than 2% of the estimate of $L^{\text{cam}}(\mathbf{e})$ using the current cut.

4.2.2 Multidimensional Lightcuts

The problem with the basic LC algorithm above is that a single pixel computation may require radiance estimates at many eye samples. When each is computed separately, the perceptual metric is less efficient. For example, consider a pixel that requires 16 eye samples for a good estimate of the final pixel color. In this case the maximum error the metric allows for each individual cut is $1/16^{\text{th}}$ the error allowed in the sum of all eye sample cuts. Potentially, computing shorter, less accurate individual cuts may not have a perceptual effect on the final pixel value. Multidimensional Lightcuts (MDLC) is a unified algorithm for combining these individual cut computations. The MDLC algorithm has the same four-step structure as the one described for LC in the previous section, except that now, before each pixel computation a set $\mathbb{E}$ of eye samples is created and, instead of clustering the light samples individually, the MDLC algorithm clusters pairs of eye and light samples. MDLC and LC use the same light sample generation algorithm, so the discussion here starts with step 2 where they differ.

Given the eye samples, the total pixel radiance L_{pixel} can be computed by summing the contributions of all eye and light sample pairs

$$L_{\text{pixel}} = \sum_{(j,i)\in\mathbb{E}\times\mathbb{L}} E_j F_{ji} I_i \tag{4.5}$$

In Equation (4.5), E_j is the weight of the j^{th} eye sample in the pixel estimate. The next step assumes that the set $\mathbb{E}\times\mathbb{L}$ of pairs can be partitioned into a set of clusters. Each cluster of pairs can be represented by a pair $(\mathbb{C}_{\mathbf{e}}, \mathbb{C}_{\mathbf{l}})$ of a cluster of eye samples $\mathbb{C}_{\mathbf{e}}$ and a cluster of light samples $\mathbb{C}_{\mathbf{l}}$. Given a partitioning of all pairs into clusters $\mathbb{P}_{\mathbb{C}}$, the larger sum (Equation (4.5)) can be approximated by a sum over pair clusters which uses the same form factor $F_{\mathbf{el}}$ for all pairs in each cluster.

$$L_{\text{pixel}} = \sum_{(\mathbb{C}_{\mathbf{e}},\mathbb{C}_{\mathbf{l}})\in\mathbb{P}_{\mathbb{C}}} F_{\mathbf{el}} \sum_{\substack{j\in\mathbb{C}_{\mathbf{e}}\\ i\in\mathbb{C}_{\mathbf{l}}}} E_j I_i \tag{4.6}$$

Like LC, the chosen form factor $F_{\mathbf{el}}$ is the factor between a *representative* eye sample/light sample pair $(\mathbf{e}, \mathbf{l}) \in \mathbb{C}_{\mathbf{e}} \times \mathbb{C}_{\mathbf{l}}$. Also like LC, the potential eye and light clusters—used to create the pair clusters—are pre-computed and form separate binary cluster hierarchies. For each pre-computed cluster, the total value E_j and I_i can be computed and stored. Then Equation (4.6) can be further approximated by replacing the interior sum of products with a product of these pre-computed sums

$$L_{\text{pixel}} = \sum_{(\mathbb{C}_{\mathbf{e}},\mathbb{C}_{\mathbf{l}})\in\mathbb{P}_{\mathbb{C}}} F_{\mathbf{el}} \left[\sum_{j\in\mathbb{C}_{\mathbf{e}}} E_j\right] \left[\sum_{i\in\mathbb{C}_{\mathbf{l}}} I_i\right] \tag{4.7}$$

In this form, the cost of a cluster evaluation reduces to the cost of evaluating the single representative pair. Amazingly, for many scenes, the number of pair clusters required by MDLC to compute the entire pixel integral is roughly equal to the number of light sample clusters required by LC to estimate the radiance at a single sample. Since the two cluster estimates have equal cost, MDLC can be dramatically faster.

To select its clustering, MDLC also computes a cut by refinement but, instead of a single cluster tree, the cut is through the implicit Cartesian product graph of the two individual sample cluster trees. The product graph need not be explicitly created to perform this traversal; the cut can be computed by walking the two cluster trees simultaneously. MDLC starts with the cut containing only the root pair cluster and then it iteratively refines the cut by removing the pair cluster with the highest error. During each refinement step, it replaces the removed pair cluster with two new pair clusters formed by replacing either the eye cluster or the light cluster with its hierarchy's children. Like LC, the stopping criteria is the same

2% perceptual metric and the error of a cluster is bounded by an estimate of its maximum contribution.

$$\left\|L_{\mathbb{C}}^{\text{true}} - L_{\mathbb{C}}^{\text{est}}\right\| \leq F_{\mathbf{el}}^{(ub)} \left[\sum_{j \in \mathbb{C}_{\mathbf{e}}} E_j\right] \left[\sum_{i \in \mathbb{C}_{\mathbf{l}}} I_i\right] \tag{4.8}$$

Since in Equation (4.8) the bound is over a cluster of pairs instead of just a cluster of light samples, the computation of $F_{\mathbf{el}}^{(ub)}$ changes in MDLC but the details are left to the original paper [WABG06].

4.3 Scalable Subsurface Algorithm

The new scalable subsurface rendering algorithm has a basic outline analogous to both MDLC and LC. The new algorithm proceeds in five steps by:

1. discretizing the integral using three types of samples: eye samples $\mathbf{e}$, irradiance samples $\mathbf{b}$ and light samples $\mathbf{l}$;
2. building three binary hierarchical clusterings of each type of sample;
3. finding an accurate partitioning of complete eye-subsurface-light paths, represented as a set of triple clusters $(\mathbb{C}_{\mathbf{e}}, \mathbb{C}_{\mathbf{b}}, \mathbb{C}_{\mathbf{l}})$, by refining a cut through an implicit hierarchy of potential triple clusters;
4. efficiently computing a bounded-error estimate of the integrated subsurface scattering by using Equation (4.15) to approximate the contribution of each triple cluster in the cut; and
5. caching the expensive computation of the irradiance sample to light sample form factors of between pixels using a form factor cache.

This section describes the new algorithm in four parts. The first discusses the discretization of the subsurface scattering integral and the second describes how

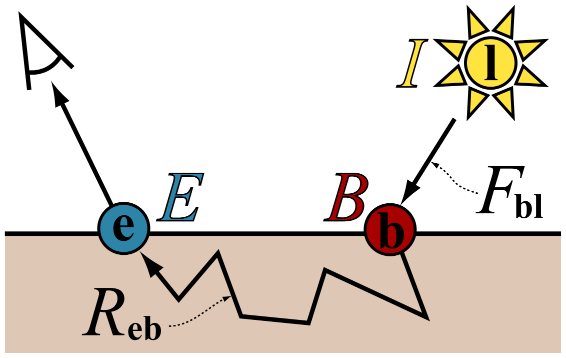


Figure 4.2: An eye-subsurface-light path can be represented by three points—a blue eye sample $\mathbf{e}$, a red irradiance point $\mathbf{b}_j$ and a yellow light point $\mathbf{l}_i$—and two links—the irradiance link $F_{\mathbf{bl}}$ and the BSSRDF link $R_{\mathbf{eb}}$.

to convert the integral into a sum using the discretized samples. This second section also describes how to efficiently approximate this sum with clusters of sample triples. Comparing the size of triple cluster summation to other summation approximations used by previous algorithms, this discussion demonstrates why triple clustering is essential for scalable rendering. The third section describes how to select an error-bounded clustering of triples and finally the last part discusses the new irradiance form factor cache. This cache is a critical component of the new algorithm that shares expensive sub-calculations between estimates for different pixels.

4.3.1 Domain Discretization

Like MDLC, the new scalable algorithm integrates the radiance over a pixel. Using Equation (1.2), this integral is

$$L_{\text{pixel}} = \int_{\mathcal{E}} \int_{\partial\Omega} \int_{4\pi} S(\mathbf{x}, \vec{\omega}, \mathbf{x}', \vec{\omega}') L(\mathbf{x}, \vec{\omega}')(\vec{\omega}' \cdot \vec{n})\, d\vec{\omega}'\, d\mathbf{x}'\, d\mathbf{x} \qquad (4.9)$$

The integral domain of Equation (4.9) represents the set of paths, illustrated in Figure 4.2, that leave the eye, pass through the subsurface of the object and leave to travel towards any light. Since the new algorithm uses the dipole diffusion BSSRDF, each of these paths can be represented by three samples. Like MDLC, the blue eye samples $\mathbf{e}$ discretize the area visible through the pixel $\mathcal{E}$ and the radiance is discretized by yellow isotropically emitting light samples $\mathbf{l}$. However, the subsurface integral additionally requires *irradiance samples*. The red irradiance samples $\mathbf{b}$ discretize the surface $\partial\Omega$ of the subsurface scattering object. The sets of all samples of each type are respectively the eye sample set $\mathbb{E}$, the light sample set $\mathbb{L}$ and the irradiance sample set $\mathbb{B}$. The discussion below describes how to partition each of these sets into a set ($\mathbb{E}_\mathbb{C}$, $\mathbb{B}_\mathbb{C}$, $\mathbb{L}_\mathbb{C}$ respectively) of clusters ($\mathbb{C}_\mathbf{e}$, $\mathbb{C}_\mathbf{b}$, $\mathbb{C}_\mathbf{l}$ respectively) of samples.

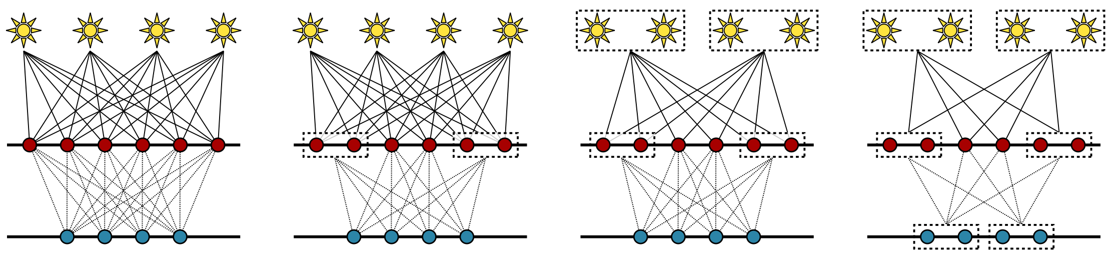
The radiance from a single sample triple $(\mathbf{e}, \mathbf{b}, \mathbf{l})$ is expressed with five terms also illustrated in Figure 4.2. First, each of the three sample types has a strength that represents the sample's relative importance.

- The eye sample's strength E is its fraction of the pixel area times the incoming Fresnel term $F_t(\eta, \vec{\omega})$.
- The irradiance sample's strength B is its fraction of the scattering surface's area times the outgoing Fresnel term $F_{dt}(\eta)$[1].
- The light sample's strength I_i is the intensity of sample $\mathbf{l}_i$.

Second, each of the two links between samples has a weight.

- The irradiance link weight $F_{\mathbf{bl}}$ is the point to point form factor from Section 4.2.1
- The BSSRDF link weight $R_{\mathbf{eb}}$ is the dipole BSSRDF (Equation (2.32)) between $\mathbf{e}$ and $\mathbf{b}$.

[1]The irradiance strength uses the average Fresnel reflectance $F_{dt}(\eta)$ because the dipole diffusion BSSRDF discards the surface outgoing direction.


(a) NTP (b) HTP (c) HTPwLC (d) Triples

Algorithm	Equation		Evaluations (in millions)	
	1st Pass [Irradiance]	2nd Pass [BSSRDF]	1st Pass [Irradiance]	2nd Pass [BSSRDF]
NTP	$O(\lvert\mathbb{B} \times \mathbb{L}\rvert)$	$O(\lvert\mathbb{E} \times \mathbb{B}\rvert)$	3,200,000	19,200,000
HTP	$O(\lvert\mathbb{B} \times \mathbb{L}\rvert)$	$O(\lvert\mathbb{E} \times \mathbb{B}_{\mathbb{C}}\rvert)$	3,200,000	1,500
HTPwLC	$O(\lvert\mathbb{B} \times \mathbb{L}_{\mathbb{C}}\rvert)$	$O(\lvert\mathbb{E} \times \mathbb{B}_{\mathbb{C}}\rvert)$	128,000	1,500
Triples	$O(\lvert\mathbb{E}_{\mathbb{C}} \times \mathbb{T}_{\mathbb{C}}\rvert)$		1,800	

Figure 4.3: Top Row: four different methods of computing L_{pixel} using 4 light samples (yellow; top), 6 irradiance samples (red; middle) and 4 eye samples (blue; bottom). (a) Naïve Two-pass (NTP); (b) Hierarchical Two-pass (HTP); (c) Hierarchical Two-pass with LightCuts (HTPwLC); and (d) Unified Triples. Table: Scalability of the four summation algorithms on the Cordoba scene where $|\mathbb{E}| \approx 300,000$, $|\mathbb{B}| \approx 64,000,000$, $|\mathbb{L}| \approx 50,000$, $|\mathbb{B}_{\mathbb{C}}| \approx 5,000$, $|\mathbb{L}_{\mathbb{C}}| \approx 2,000$, $|\mathbb{E}_{\mathbb{C}}| \approx 300,000$ and $|\mathbb{T}_{\mathbb{C}}| \approx 6,000$.

4.3.2 Summation

Using the discretization above, Equation (4.9) can be rewritten as a sum

$$L_{\text{pixel}} = \sum_{(k,j)\in\mathbb{E}\times\mathbb{B}} E_k R_{kj} B_j \left[\sum_{i\in\mathbb{L}} F_{ji} I_i \right] \tag{4.10}$$

This section builds up the new clustered sum used by the new scalable algorithm by considering four progressively more efficient approximations of Equation (4.10). These methods are illustrated in Figure 4.3. However, as a prelude to this discussion, consider the table in Figure 4.3 which summarizes the cost of rendering the large Cordoba scene (see Figure 4.4) using each of the four algorithms. Except for the new unified triple algorithm, all the summation methods discussed below are two-pass algorithms where the first pass evaluates all of the irradiance links and the second pass evaluates all of the BSSRDF links. The total number of each type of evaluation is given in the rightmost two columns of Figure 4.3.

Regardless of their methods, the table highlights the fundamental problem with two-pass algorithms. They are extremely efficient at decreasing the cost of the second pass. They reduce the number of BSSRDF link evaluations by several orders of magnitude. However, they do nothing to reduce the cost of the first pass. In large scene like Cordoba, where potentially trillions of irradiance links must be evaluated, the first pass represents essentially all of the cost. Even using Lightcuts (see Section 4.2.1) to reduce the cost of individual computations in the first pass, a two-pass algorithm still requires over 100 billion irradiance link evaluations. However, the new unified algorithm, by clustering complete eye-subsurface-light paths, is able to recognize that only a small fraction of those, 1.8 billion, make important contributions to the image. The practical effect is to reduce image cost from hours to minutes.

4.3.2.1 Naïve Two-pass

The naïve two-pass algorithm (Figure 4.3(a)) recognizes that the value of the inner summation in Equation (4.10) depends only on the irradiance sample and the light samples and its value can be precomputed, stored and re-used for all pixels.

$$1^{\text{st}} \text{ pass:} \qquad \forall j \in \mathbb{B}, \qquad \mathcal{B}_j = B_j \sum_{i \in \mathbb{L}} F_{ji} I_i \tag{4.11}$$

$$2^{\text{nd}} \text{ pass:} \qquad L_{\text{pixel}} = \sum_{(k,j) \in \mathbb{E} \times \mathbb{B}} E_k R_{kj} \mathcal{B}_j \tag{4.12}$$

4.3.2.2 Hierarchical Two-pass

The previous work by Jensen et al. [JB02] developed the naïve two-pass algorithm and then introduced a clustering algorithm to accelerate the second pass (see Figure

4.3(b)). They pre-computed a hierarchical set of clusters of irradiance samples and, using this hierarchy, they selected set of irradiance clusters $\mathbb{B}_{\mathbb{C}}$ for each eye sample. Then they approximated Equation (4.12) by reusing a single BSSRDF weight for all the irradiance samples in the cluster.

$$L_{\text{pixel}} = \sum_{\substack{k \in \mathbb{E} \\ \mathbb{C}_{\mathbf{b}} \in \mathbb{B}_{\mathbb{C}}}} E_k R_{k\mathbf{c}} \left[\sum_{j \in \mathbb{C}_{\mathbf{b}}} \mathcal{B}_j \right] \tag{4.13}$$

In Equation (4.13), $R_{k\mathbf{c}}$ is the BSSRDF between the k^{th} eye sample and the centroid of the irradiance samples in $\mathbb{C}_{\mathbf{b}}$. Since the inner sums can be pre-computed and stored with each potential irradiance cluster, Equation (4.13) reduces the cost of the second pass from $O(|\mathbb{E} \times \mathbb{B}|)$ to $O(|\mathbb{E} \times \mathbb{B}_{\mathbb{C}}|)$.

4.3.2.3 Hierarchical Two-pass with Lightcuts

Next, in much the same way that Jensen et al. accelerated the second pass, the clustered Lightcuts approximation (Equation (4.3)) can be used to make each computation in the first pass more efficient (see Figure 4.3(c)).

$$\mathcal{B}_j = B_j \sum_{\mathbb{C}_{\mathbf{l}} \in \mathbb{L}_{\mathbb{C}}} F_{j\mathbf{l}} \left[\sum_{i \in \mathbb{C}_{\mathbf{l}}} I_i \right] \tag{4.14}$$

Here $F_{j\mathbf{l}}$ is the irradiance link weight between the j^{th} irradiance sample and the representative light of $\mathbb{C}_{\mathbf{l}}$. Again, since the inner sum can be pre-computed, this reduces the cost of the first pass from $O(|\mathbb{B} \times \mathbb{L}|)$ to $O(|\mathbb{B} \times \mathbb{L}_{\mathbb{C}}|)$ but unfortunately, this is insufficient. The issue is that the two cluster selection operations, the light cluster selection in the first pass and the irradiance cluster selection in the second pass, are done independently.

4.3.2.4 Unified Triples

The new unified algorithm (see Figure 4.3(d)) clusters triples of samples $(\mathbf{e}, \mathbf{b}, \mathbf{l})$ each representing a complete eye-subsurface-light path. By unifying the two different clustering operations, the relative importance of both the BSSRDF link and the irradiance link can be considered simultaneously when partitioning the paths into clusters. Having this unified knowledge, allows the algorithm to choose a much smaller set of cluster triples without sacrificing the quality of the resulting estimate.

A cluster of eye-subsurface-light paths is represented with a triple of clusters $(\mathbb{C}_{\mathbf{e}}, \mathbb{C}_{\mathbf{b}}, \mathbb{C}_{\mathbf{l}})$. Given a set of triple clusters $\mathbb{T}_{\mathbb{C}}$, Equation (4.10) finally becomes

$$L_{\text{pixel}} = \sum_{(\mathbb{C}_{\mathbf{e}},\mathbb{C}_{\mathbf{b}},\mathbb{C}_{\mathbf{l}})\in\mathbb{T}_{\mathbb{C}}} R_{\mathbf{eb}} F_{\mathbf{bl}} \left[\sum_{k\in\mathbb{C}_{\mathbf{e}}} E_k\right] \left[\sum_{j\in\mathbb{C}_{\mathbf{b}}} B_j\right] \left[\sum_{i\in\mathbb{C}_{\mathbf{l}}} I_i\right] \tag{4.15}$$

where $R_{\mathbf{eb}}$ and $F_{\mathbf{bl}}$ are the link weights for a representative triple $(\mathbf{e}, \mathbf{b}, \mathbf{l}) \in \mathbb{C}_{\mathbf{e}} \times \mathbb{C}_{\mathbf{b}} \times \mathbb{C}_{\mathbf{l}}$. Like LC and MDLC, the new algorithm pre-computes hierarchical clusters of each type of sample and, with these clusters, the inner sums can be pre-computed and stored. Thus, the cost estimating each triple cluster reduces to the cost of evaluating a single sample triple.

4.3.3 Cut Selection

Given Equation (4.15), the new algorithm simply requires an efficient method of computing a set of triple clusters for each pixel. Like MDLC, the algorithm computes this set by traversing a implicit hierarchy of potential clusters. Separate hierarchical, binary clusterings are created for the eye, irradiance and light samples. The eye sample clustering is computed once per pixel while the others are computed once per image. To chose a clustering, the algorithm creates a cut through the implicit Cartesian product graph formed by these three binary cluster trees. The process is analogous to MDLC (see Section 4.2.2). It starts with the cut containing the root triple cluster, the triple cluster that contains the roots of the three binary trees. Then it iteratively removes and refines the highest error triple cluster currently in the cut. Refining a triple cluster replaces it in the cut with two smaller clusters formed by switching one of the member sub-clusters with each of its two children from its respective hierarchy. A heuristic chooses which sub-cluster to refine at each stage (see Section 4.4.3). During cut selection, a running estimate of Equation (4.15) is maintained using the triple clusters currently in the cut. As in LC and MDLC, refinement stops when the error of the highest error triple cluster in the cut falls below a 2% of the current cut's estimate. Since the true contribution of a triple cluster is a positive number, the error of a triple cluster estimate can be conservatively bounded by the upper bound of its contribution estimate (compare to Equation (4.15)).

$$\left\| L_{\mathbb{C}}^{\text{true}} - L_{\mathbb{C}}^{\text{est}} \right\| = \leq R_{\mathbf{eb}}^{(ub)} F_{\mathbf{bl}}^{(ub)} \left[\sum_{k \in \mathbb{C}_{\mathbf{e}}} E_k \right] \left[\sum_{j \in \mathbb{C}_{\mathbf{b}}} B_j \right] \left[\sum_{i \in \mathbb{C}_{\mathbf{l}}} I_i \right] \tag{4.16}$$

Since the dipole BSSRDF monotonically decreases with distance, $R_{\mathbf{eb}}^{(ub)}$ can be conservatively estimated as the BSSRDF for the smallest possible distance between any eye sample in $\mathbb{C}_{\mathbf{e}}$ and any irradiance sample in $\mathbb{C}_{\mathbf{b}}$ and $F_{\mathbf{bl}}^{(ub)}$ can be bounded using the techniques from [WABG06].

4.3.4 Irradiance Link Form Factor Cache

The most significant cost of the new algorithm is the evaluation of the irradiance link form factor weights F. Each of these evaluations requires an expensive visibility check between the irradiance sample and light sample. However, since these weights do not depend on the eye sample, they can be cached and reused between different pixels. This form factor cache dramatically improves performance. However, because the cache is accessed every time the algorithm refines the current cut, the implementation must be as efficient as possible.

To ensure this efficiency, the cache is built to mirror the order different irradiance links are generated by the cut selection algorithm. For example, consider the root triple cluster $(\mathbb{C}_{\mathbf{e}}, \mathbb{C}_{\mathbf{b}}, \mathbb{C}_{\mathbf{l}})$. The estimate for this cluster implicitly includes an estimate for the irradiance link between $\mathbb{C}_{\mathbf{b}}$ and $\mathbb{C}_{\mathbf{l}}$. This estimate is stored in a new root cache node. Since the irradiance and light cluster trees are reused for all pixels, this root cache node can be used as the cache starting point for all pixels. From the root cut, there are three possible refinement choices, splitting the eye, irradiance or light cluster. Splitting the eye cluster does not effect the irradiance link estimate and the root cache node covers these cases. However, splitting either the irradiance or light cluster creates two new irradiance form factor estimates and, therefore, the root cache node has four potential children. If each triple cluster in the cut stores a pointer to the cache node it uses, the cache refinement algorithm can simultaneously traverse this natural quad-tree of cache nodes. The refinement algorithm builds the cache tree lazily. Anytime a cache node is requested, but does not exist, the algorithm creates a new cache node and stores the appropriate form factor within it.

Without an eviction heuristic, the form factor cache would quickly grow prohibitively large. Fortunately, the cache accesses tend to be local. At any time, most of the nodes in the cache store form factors relevant to a small region of pixels nearby the current computation. Nearby pixels frequently reuse these form factors, but as the rendering progresses to more distant regions, the form factors for small cluster triples are no longer needed. After testing several cache eviction algorithms, cache deletion was empirically determined to be the most efficient eviction strategy. The refinement algorithm fixes the total number of nodes allowed in the cache (1,000,000 is a reasonable choice) and simply discards them when the cache is full and lets the algorithm lazily rebuild the cache as necessary.

4.4 Implementation Details

This section discuss additional details essential for implementing the algorithm described in the last section.

4.4.1 Surface, Single and Multiple Scattering

The dipole diffusion BSSRDF is an approximate method of representing subsurface scattering. When using the dipole diffusion BSSRDF, rendering quality can be significantly improved if the BSSRDF is split into three components, the surface scattering, the single scattering and the multiple scattering. As discussed in Section 2.2.2.1, these different components can be rendered separately. The test implementation uses MDLC for the surface component, an analytic BRDF estimate for the single scattering [HK93] and the new scalable, algorithm for the multiple scattering.

4.4.2 Representative Selection

When computing a triple cluster's contribution, it is important to chose the representative triple carefully. If the cut contains many triple clusters with similar representatives, the resulting estimate might have low error, but the switch from one fixed set of representatives to another between neighboring pixels can result in aliasing in the final image. Borrowing a technique from MDLC, the new algorithm stores multiple representatives per cluster. Each time a cluster's contribution is evaluated, a new representative is randomly selected from the pool in the cluster. Random representative selection makes it unlikely that nearby integral evaluations will choose the same representatives and aliasing resulting from correlated representative selection is avoided.

However, multiple representatives complicate the irradiance form factor cache. A weight in stored in the cache implicitly fixes the choice of the representative triple used to compute it. Caching multiple weights per cluster would be prohibitively expensive and significantly reduce the cache's utility. However, within a triple cluster, it is not necessary to use the same irradiance representative to evaluate both the representative BSSRDF and irradiance links. By using different representatives, the algorithm can select from randomized representatives for BSSRDF link evaluations while still caching weights for fixed representatives in the form factor cache. The randomization of just the BSSRDF representatives is sufficient to avoid aliasing.

4.4.3 Triple Cluster Refinement

During cut selection, the algorithm must choose how to refine a given triple cluster. The three choices are refining the eye sub-cluster, the irradiance sub-cluster or the light sub-cluster. The ideal choice is the sub-cluster that would ultimately produce the smallest, and thus cheapest, final cut. However, since this cannot be easily determined, a heuristic choice must be made. As will be discussed in Section 4.4.3.2, refining the eye cluster is not usually necessary so the next section focuses first on the choice between the irradiance and light clusters.

4.4.3.1 Irradiance/Light Refinement Heuristics

The heuristic choice between refining the light or irradiance clusters is motivated by two factors:

- since the error in BSSRDF weight decreases exponentially with distance, splitting the irradiance cluster tends to reduce error faster—isolating small, high-contribution triples near the eye samples—than splitting the light cluster which causes only a linear decrease in error; however,

- light cluster refinement is required to identify the visible light sources.

A corollary of these two competing factors is that long chains of the same refinement are poor choices. Based on these factors, a good heuristic applies the following four tests in order to determine how to refine a cluster with the representative triple $(\mathbf{e}, \mathbf{b}, \mathbf{l})$

1. If the irradiance cluster intersects the eye cluster, split the irradiance cluster.
2. If the last m consecutive refinements have made the same choice, split the opposite cluster.
3. To ensures that light clusters with potentially large contributions are subdivided early, if

   $$F_{\mathbf{bl}}^{(nv)} I_i > \alpha * W \tag{4.17}$$

   split the light cluster. In Equation (4.17), W is the image white point and $F_{\mathbf{bl}}^{(nv)}$ is the value of the representative irradiance link weight without the visibility term.
4. If

   $$R_{\mathbf{eb}} > \beta * F_{\mathbf{bl}}^{(nv)} \tag{4.18}$$

   refine the irradiance cluster otherwise refine the light cluster. This last test attempts to estimate which representative link term contributes most to the current error and refines the appropriate cluster accordingly.

The values of m, α and β are user defined parameters that adjust the relative importance of the various heuristics. Respectively, 8, 10 and 1 were used for all results in Section 4.5.

4.4.3.2 Eye Cluster Refinement

Using the dipole diffusion BSSRDF requires the assumption that the scattering media has a high albedo and a short mean free path. As a consequence, light in the media tends to diffuse throughout the volume and the resulting radiance distribution becomes smooth. This smoothness means that the multiply scattered radiance can be sampled at a much lower frequency than the surface reflection which tends to have high frequency shadow and geometry edges. This project compares the scalability of the new algorithm with previous algorithms. For this goal, super-sampling the eye samples would unfairly degrade the performance of the previous algorithms since they cannot cluster these extra eye samples. For the results presented, one eye sample per pixel is sufficient for both the previous and new methods.

However, the new algorithm is agnostic to extra eye samples. When eye sample clusters are present, the heuristics in the previous section are still applied. However, whenever these heuristics would choose to refine the irradiance cluster, the algorithm instead refines the the larger of the eye and irradiance clusters. Since the eye samples from a single pixel are usually tightly packed together, adding eye clusters changes the refinement strategy little.

4.4.4 Irradiance Sample Generation

To generate the irradiance samples, the algorithm must generate a smooth, uniformly distributed set of point samples across the entire surface of the scattering geometry. In previous work [JB02], these samples were computed using an energy based point repulsion algorithm developed by Turk [Tur92]. However for large meshes, that algorithm required several hours to converge to a steady state distribution. For the tests in the next section, a simpler method—based on Poisson sampling by dart throwing [Mit87, Mit91]—produced equal quality images and required only a few minutes of computation.

A 2D Poisson sampling is a random sampling satisfying the Poisson condition: no sample can lie within a fixed radius of any other sample. For irradiance sample generation, this radius is the mean free path of the material. The Poisson dart throwing algorithm randomly generates surface samples, one at a time, and discards any sample that violates the Poisson condition. However, near the end of sample generation, true Poisson dart throwing algorithms may generate many samples before finding a valid one and they may even have to backtrack. These cases make true dart throwing expensive. However, for the purposes of irradiance sample generation, it is sufficient to use a faster approximate algorithm [Mit87, Mit91].

In the set of trial samples generated since the last valid sample was found, the sampling algorithm remembers the discarded sample farthest away from all valid samples. If sufficiently many discards are made before a new valid sample is found, this "best" discarded sample is added even if it violates the Poisson condition. This fast algorithm works quickly, generating sets of many thousands of samples in less than a minute. Further, for all the tests here, on average, only 20 discards were made per valid sample generated and there was no need to add a invalid sample when up to 10,000 consecutive discards were allowed.

4.5 Analysis

To close this chapter, the new scalable algorithm was tested on a series of three progressively larger scenes illustrated in Figure 4.4.

Teapot contains a solid marble teapot (scattering parameters from [JMLH01]) on a small table lit by an area key light and the Grace Cathedral environment map [Deb02].

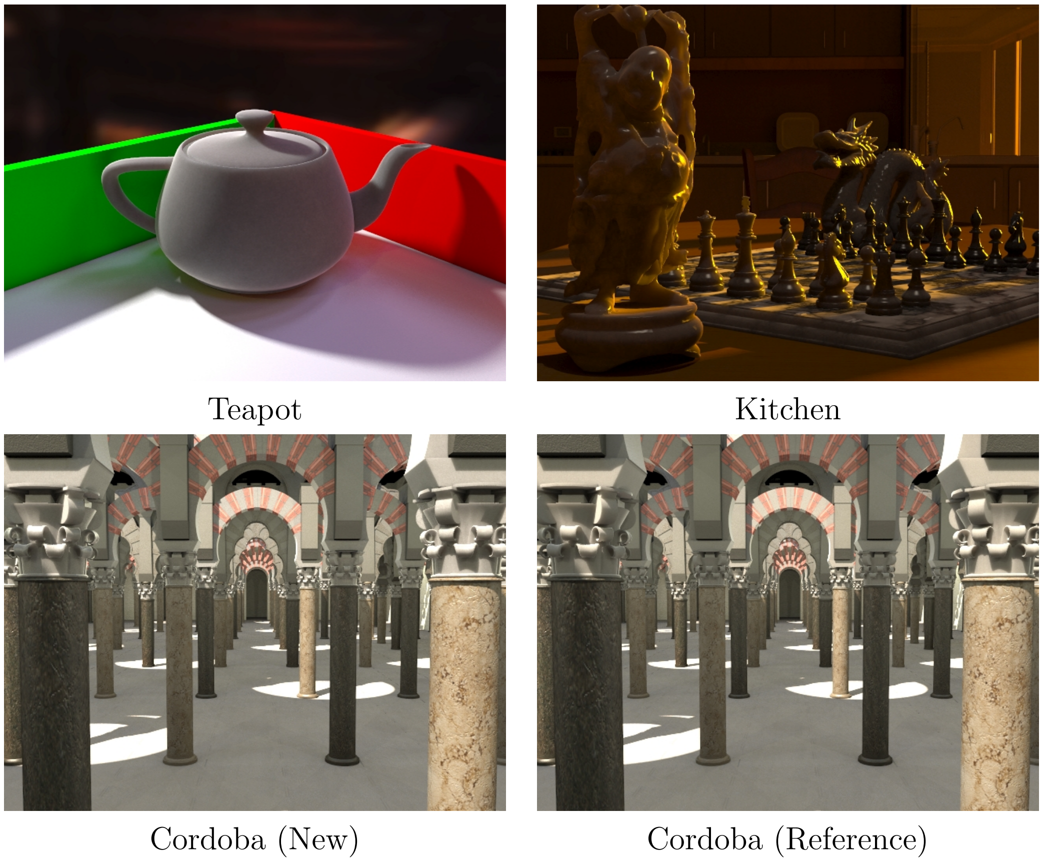


Figure 4.4: Result images for the three test scenes: Teapot, Chess and Cordoba. The lower row compares our new result (left) to a reference rendering using [JB02].

Kitchen includes several white and black marble objects on a table lit by several small recessed area lights and a sun/sky model shining through several large windows out of frame.

Mezquita de Cordoba shows a colonnade similar to the famous Spanish mosque. The capitals of each column and every other brick in the upper archways has been rendered using a translucent marble material (see Figure 4.5 for a closeup view). The scene is lit both by several large holes in the ceiling that pass in sun/sky illumination and from a grid of 90 point lights positioned near the ceiling. This final scene was designed to demonstrate the type of large complex rendering only possible with the new scalable algorithm.

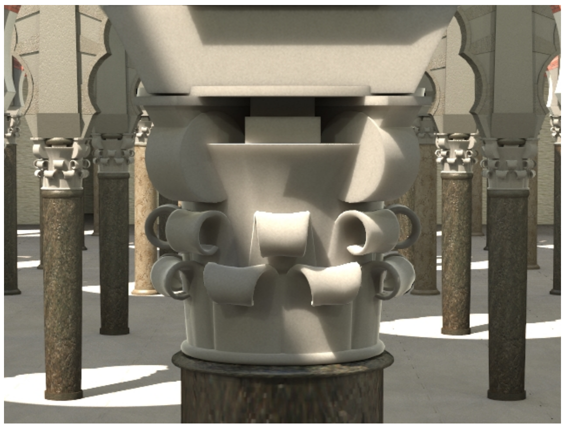


Figure 4.5: Closeup of the marble capitals in the Mezquita de Cordoba model.

For each scene, Table 4.1(a) summarizes the number of polygons, irradiance samples and light samples used for rendering.

4.5.1 Rendering Configuration

This analysis compares a reference implementation of the two-pass method [JB02] to the new scalable renderer. To make the comparison as fairly as possible, the reference algorithm uses Lightcuts to pre-compute the irradiance during the first pass. In both algorithms the reflected surface component is computed using Multidimensional Lightcuts [WABG06]. All images are 640x480 pixels in resolution. The Teapot and Kitchen images were computed using a dual-core 3GHz Pentium 4 with 2GB RAM and the Cordoba scene was rendered on a cluster of sixteen 1.7GHz Pentium 3s with 1GB of RAM.

For all images, for the surface rendering, each pixel was 32x super-sampled; however, as noted Section 4.4.3.2 this is unnecessary for the multiple scattering. For both algorithms, the multiple scattering is only estimated at the eye sample closest to the centroid of the samples in each pixel. The single scattering term was approximated by a BRDF [HK93] in both algorithms and an initial culling operation ensures that neither algorithm pre-computes any information for objects

Table 4.1: Model size, rendering statistics and costs of the new unified scalable renderer for its three test scenes.

(a) Number of polygons, irradiance samples and light samples.

Model	Polygons	Irrad. Samples	Light Samples
Teapot	16,422	173,671	53,128
Kitchen	1,238,126	3,540,428	53,896
Cordoba	2,070,732	64,483,644	53,128

(b) Left side: total number of irradiance links evaluated for the reference and new algorithms and the percentage saved by the new algorithm. Right side: Average cut size, average irradiance link evaluations per cut and form factor cache hit rate.

Model	Irrad. Links			FF Cache Performance		
	Reference	New	% Saved	Ave. Cut	Misses	FF Hit %
Teapot	53M	12M	77.3	3,589	243	99.3
Kitchen	1,414M	62M	95.6	3,360	405	87.9
Cordoba	91,100M	304M	99.6	6,306	6,220	1.3

(c) Rendering costs for new algorithm compared to the reference algorithm. For each algorithm, the columns present the time for the subsurface computation, the total image time and the percent of that total the algorithm required for the subsurface. The new algorithm additionally reports the speedup of the subsurface computation obtained using the new approach. Rows marked with † are reported for two 3 Ghz processors and those marked with ‡ are reported for sixteen 1.75 Ghz processors.

Model	Reference			New			
	SS	Total	% SS	SS	Total	% SS	SS Speedup
Teapot†	321s	641s	50.0	318s	618s	48.2	1.0x
Kitchen†	5,179s	6,727s	74.5	960s	2,679s	35.8	5.4x
Cordoba‡	50,276s	52,717s	97.8	166s	1,258s	13.2	300.0x

not visible in the image. In both implementations, the evaluation of the diffuse BSSRDF term R had a significant cost. However, interpolating R from a table of 100,000 pre-computed values introduced no visible differences and this look-up table was used by both renders.

4.5.2 Discussion

Tables 4.1(b) and 4.1(c) highlight three significant benefits of the new scalable rendering algorithm: the reduction irradiance link computations, the overall speedup and the performance of the irradiance form factor cache.

4.5.2.1 Irradiance Link Savings

As shown in the left side of Table 4.1(b), the new algorithm is able to dramatically reduce the number of irradiance link evaluations. Since these evaluations are the only part of the rendering computation that increases with the number of polygons in the scene, their savings rate closely estimates the ratio of cost growth between the two algorithms as tessellation increases. For example all other things being equal, if each polygon in Cordoba were subdivided in place, one would expect the increase in cost of the new algorithm to be roughly 1% of the increase of the two-pass method.

4.5.2.2 Overall Scalability

Of course, the most important comparison is the overall savings in rendering time. Table 4.1(b) summarizes rendering costs for both algorithms. Since only a fraction of the total time depends on the subsurface rendering algorithm, Table 4.1(b) reports the cost of the subsurface computation separate from the total cost of the image. Comparing these subsurface costs reveals the significant cost benefit of a scalable subsurface rendering algorithm. For the two-pass algorithm, the cost of the subsurface computation grows quickly with complexity becoming almost 100% of the almost 15 hour computation for the Cordoba scene. Comparatively, the relative cost of adding subsurface components to complex scenes decreases with the scalable renderer. The result is a 300x decrease in subsurface rendering costs:

the difference between a practical and impractical rendering algorithm.

4.5.2.3 Form Factor Cache Performance

Finally, the form factor cache performs well. The right side of Table 4.1(b) shows the average number of triples evaluated per pixel per image, the average number of irradiance link evaluations per pixel and their ratio, the hit rate of the form factor cache. Ignoring the Cordoba result for a moment, the high hit rates for the Teapot and Kitchen scenes demonstrate that, when pixels are spatially similar, the form factor cache provides significant reuse and savings. However, when the scene has a large spatial extent, like Cordoba, the spatial distance between pixels grows and the set of triples that contributes to each pixel becomes isolated from its neighbors. When this happens the cache hit rate declines because there is less and less potential for shared computation. However, this is precisely the case when the overall scalability of the unified triple algorithm provides the most benefit. In this case, the unified algorithm saves significant effort by finding just the triples required to calculate these isolated pixel integrals and avoids a pre-computation over the entire surface.

4.6 Summary

This section described a new, scalable, dipole diffusion rendering algorithm for homogeneous subsurface scattering materials. On large, complex scenes, the new algorithm reduces the cost of including subsurface scattering by a factor 300 (Section 4.5.2.2) while remaining robust to increases in scene tessellation (Section 4.5.2.1) and, using a new form factor cache, ensuring that repeated calculations are shared efficiently between pixels(Section 4.5.2.3). To achieve these dramatic improvements, the algorithm builds on Lightcuts and Multidimensional Lightcuts (Section 4.2) to compute approximations of subsurface integrals by modeling complete eye-subsurface-light paths with triples of samples (Section 4.3.1). By partitioning these triples into clusters, the new algorithm can compute an error-bounded estimate of the true sum by selecting a cut through a hierarchy of potential clusterings (Section 4.3.3). Because it has complete information about full subsurface paths, the new triple cluster approximation has fundamentally better scalability that previous two-pass algorithms that are forced to consider only sections of the subsurface paths at a time (Section 4.3.2).

CHAPTER 5
HETEROGENEOUS SUBSURFACE SCATTERING

Previous work on subsurface rendering is clustered at the extremes. On one end lie Monte Carlo algorithms that can render any subsurface scattering material with arbitrary accuracy but are prohibitively expensive. At the other end lie dipole diffusion algorithms that are fast, even real-time, but are restricted to approximating only homogeneous materials. This research seeks a solution in between: an algorithm that can accurately render a general class of heterogeneously scattering materials while remaining practically efficient. To simultaneous satisfy these competing goals, the new algorithm represents heterogeneous scattering approximately using the diffusion equation; but, within the limits of that representation, the algorithm solves the problem as accurately as possible. This algorithm makes two contributions

1. a careful formulation of the heterogeneous DE problem introducing the diffusive source boundary condition (DSBC) to model the reduced intensity source; and
2. an finite element (FE) algorithm for solving this problem efficiently and accurately.

Previous rendering algorithms for heterogeneous subsurface scattering are limited, capture and re-render systems. Because these systems can rely on the capture process to correct for rendering inaccuracies, these systems can render high quality images but only for the captured materials. Further these systems generally use specialized material representations that are difficult to create for general materials. However, one ambitious system, Wang et al. [WZT*08], begins to overcome these limitations by measuring grids of approximately accurate, physical scattering parameters. Because their algorithm renders these grids quickly using an interactive, finite difference (FD) solver implemented on fast rendering hardware, the graphics processing unit (GPU), their system allows the user to interactively edit the captured models. Despite these advantages, the system has limited utility as a general rendering system for all materials. Wang et al.'s approach has two drawbacks. First, the finite difference algorithm requires the generation of a special warped grid, a PolyGrid mesh [THCM04], that is expensive or impossible to create inside of arbitrary scattering geometry. Second, the finite difference algorithm is inaccurate. The new finite element algorithm presented here has neither of these limitations. It is the first method of rendering, general heterogeneous subsurface scattering that:

- works well for a wide range of materials;
- computes solutions in arbitrary surface geometry;
- produces images comparable to MC algorithms requiring hours; and
- requires only a few minutes of computation per image.

5.1 Chapter Overview

To outline this chapter, its useful to state the mathematical goal of the finite element (FE) subsurface algorithm. Mathematically, the diffusion equation (DE) is a 2^{nd}-order, elliptic, partial differential equation (PDE) and the rendering algorithm requires its solution. The FE method is a general mathematical framework for approximating the solution of these PDEs. To use the FE method, one first chooses a space of functions $\mathbb{H}$ and then FE theory describes how to find the closest approximation in $\mathbb{H}$ to the solution of the PDE. If $\mathbb{H}$ has a finite basis—i.e. it is represented by a finite number of elements—then the closest approximating function can be computed by solving a linear system of equations.

The first half of this chapter derives this linear system for the heterogeneous diffusion problem. This derivation has two parts. First Section 5.2, as outlined in Section 2.2.2, discusses choosing a boundary condition and a computationally efficient model of the reduced intensity source. That section demonstrates that the best choice, balancing both accuracy and efficiency, is the diffusive source boundary condition (DSBC). Second Section 5.3 solves this heterogeneous diffusion problem using the FE method. The result is Equation (5.16). Though this derivation takes several pages, a fundamental advantage of the FE solution is that the resulting algorithm can be summarized in four steps:

Step 1 Create a basis for functions defined in the scattering domain by discretizing the scattering volume

Step 2 Construct the linear system in Equation (5.16)

Step 3 Solve this system

Step 4 Render the image and, whenever needed, use Equation (5.7) to compute the outgoing subsurface scattering from the surface

The second half of this chapter describes and then analyzes a prototype implementation of this four-step algorithm. Section 5.4 enumerates the essential implementation choices. Its two main results are the pseudo-code for the assembly of the linear system (see Figure 5.4) and a method for creating a FE basis that adaptively captures detail in the scattering material and geometry (Section 5.4.5). Finally, Section 5.5 analyzes the accuracy and performance of this prototype. This analysis has three main goals.

1. It demonstrates that the FE algorithm is general and efficient. For each of a wide range of scenes, the algorithm produces high-quality solutions in only a few minutes.

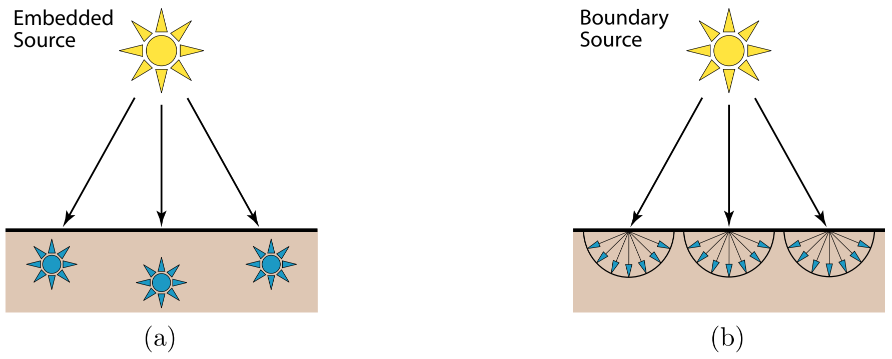


Figure 5.1: Two methods of approximating the reduced intensity source $Q_{ri}(\mathbf{x}, \vec{\omega})$. The embedded source model (a) approximates $Q_{ri}(\mathbf{x}, \vec{\omega})$ with point sources. (b) The boundary source model, approximates the source as a diffusive flux arriving at the boundary.

2. Though approximate, these images are nearly identical to accurate images produced by MC algorithms requiring hours of computation.

3. The four-step algorithm significantly improves upon the most ambitious previous approach, Wang et al., in both quality and generality.

5.2 Heterogeneous Diffusion Problem

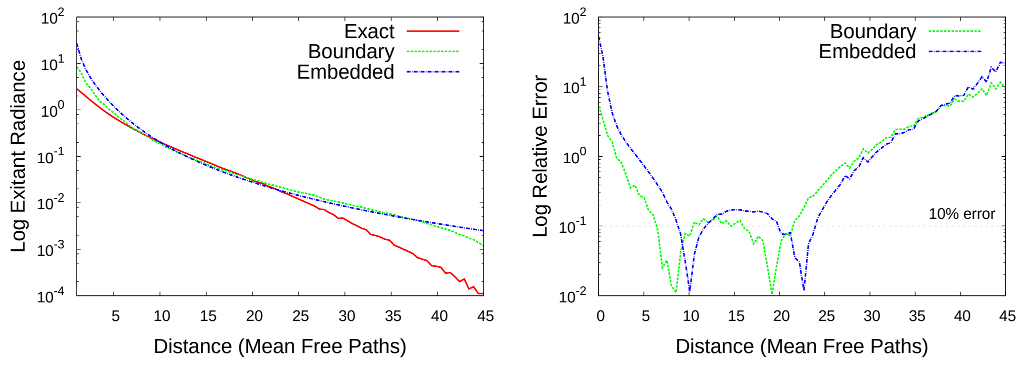
As described in Section 2.2.2, rendering using the diffusion equation requires approximating the boundary condition and the reduced intensity source. While previous work has shown the Robin boundary condition to be most accurate [SAHD95], the choice of the source model—between the embedded source model and the boundary source model—is more difficult. This section describes why, for efficiency and accuracy, the best choice is the boundary source model. Then it combines the boundary source model with the Robin boundary condition to derive the diffusive source boundary condition (DSBC). Together the diffusion equation (Equation (2.15)) and the DSBC (Equation (5.5)) completely specify the heterogeneous scattering problem.

5.2.1 Comparing Source Models

Illustrated in Figures 5.1(a) and 5.1(b) are the two models of the reduced intensity source. The embedded source model represents the source by placing point sources into the medium and the boundary source model represents the source as a diffusive flux arriving at the boundary. Compared to the embedded source model, the boundary source model has significant advantages for the heterogeneous rendering problem.

First, the boundary source model is cheaper to compute. Both the embedded source model and the boundary source model sample the surface and calculate the incoming radiance at these samples. However, the two source models must represent different functions with these samples. The boundary source model represents the incident radiance as it is distributed on the surface while the embedded source model represents this radiance after it has become distributed in the outer layer of the medium. For homogeneous materials, this difference is slight. However, in complex heterogeneous materials the embedded source model must represent additional, high-frequency features derived from the material parameters that are not present in the incident radiance itself. Thus it can require more samples to reach the same accuracy. This increased cost also has quality implications. A rendering algorithm cannot determine a perfect sampling density a priori, it must estimate one. The higher frequency components of the embedded source model make this estimate less accurate, potentially decreasing the overall accuracy of the model.

Of course, if the individual embedded sources were significantly more accurate, these computational disadvantages might balance out. However, this is not the case. Figure 5.2 compares the two source models on the scene used to derive the embedded model: a collimated beam normally incident on a semi-infinite,



(a) (b)

Figure 5.2: (a) Exact solution (red) compared to boundary source solution (green) and embedded source solution (green) for the problem of a column source incident on a homogeneous, semi-infinite slab. (b) Log scale plot of the relative error of boundary source and embedded source models.

homogeneously scattering slab. The figure shows the true solution as computed by MC path tracing compared with the FE solution using the boundary source model and the analytic, dipole solution of the embedded source model. Since the FE algorithm cannot model a semi-infinite slab, it approximates one by a cube 100 mean free paths across.

Figure 5.2(b) plots the relative error of both approaches. As outlined in Section 2.2, there are many sources of error in any model based on the diffusion equation and they are all evident on the figure. Both near the source, where the diffusion equation breaks down, and far from the source, where the radiance values are very small, there are high errors for both methods. In the critical middle region between 5 and 25 mean free paths (between the troughs where the approximations intersect the true solution), both methods have a more reasonable 10% error. Though in this region the two methods perform roughly equally, this configuration defines the embedded source model and it's error should be lowest for this case. Given that the boundary source model performs equally well and it has the performance advantages discussed above, it it clearly the best choice.

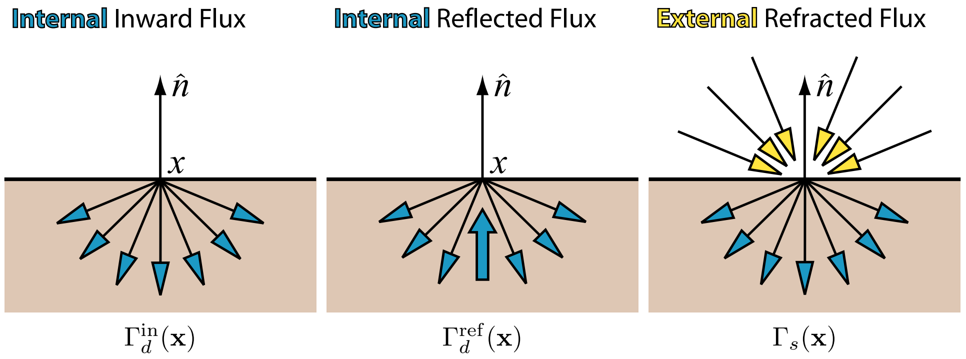


Figure 5.3: Diagrams illustrating the three components of the diffusive source boundary condition (Equation (5.5)). The condition forces $\Gamma_d^{\text{in}}(\mathbf{x})$, the internal inward flux at the boundary, to be equal to the sum of $\Gamma_d^{\text{ref}}(\mathbf{x})$, the internal flux reflected at the boundary, and $\Gamma_s(\mathbf{x})$ the exterior light refracted into the material.

5.2.2 Diffusive Source Boundary Condition Derivation

The boundary source model adds a surface flux to the Robin boundary condition (see Section 2.2.2.2) resulting in the diffusion source boundary condition (DSBC). At every boundary point $\mathbf{x}$, it imposes the relationship

$$\Gamma_d^{\text{in}}(\mathbf{x}) = \Gamma_d^{\text{ref}}(\mathbf{x}) + \Gamma_s(\mathbf{x}) \tag{5.1}$$

between three boundary fluxes (see Figure 5.3): $\Gamma_d^{\text{in}}(\mathbf{x})$ the internal inward diffuse flux of the solution; $\Gamma_d^{\text{ref}}(\mathbf{x})$ internal diffuse flux reflected at the boundary; and $\Gamma_s(\mathbf{x})$ the incoming flux refracted from external sources. Using Figure 5.3 as a guide each of these fluxes is computed by integrating over the blue arrows in each sub-figure.

$$\Gamma_d^{\text{in}}(\mathbf{x}) = \int_{(\vec{n}\cdot\vec{\omega})<0} L_d(\mathbf{x},\vec{\omega})(-\vec{n}\cdot\vec{\omega})\, d\vec{\omega} \tag{5.2}$$

$$\Gamma_d^{\text{ref}}(\mathbf{x}) = F_{dr}(\eta)\int_{(\vec{n}\cdot\vec{\omega})<0} L_d(\mathbf{x},-\vec{\omega})(-\vec{n}\cdot\vec{\omega})\, d\vec{\omega} \tag{5.3}$$

$$\Gamma_s(\mathbf{x}) = e^{-\frac{\sigma_a(\mathbf{x})}{\sigma_s(\mathbf{x})}}\int_{(\vec{n}\cdot\vec{\omega})>0} F_t(\eta,\vec{\omega})L(\mathbf{x},-\vec{\omega})(\vec{n}\cdot\vec{\omega})\, d\vec{\omega} \tag{5.4}$$

When creating these expressions, one must be careful to use a consistent definition of $\vec{\omega}$ and $\vec{n}$ to ensure that resulting boundary condition has the correct signs. To be consistent in all three equations, $\vec{n}$ always points out of the material and $\vec{\omega}$ always points away from $\mathbf{x}$. Equation (5.3) computes the total reflected internal diffuse flux by scaling the total internal diffuse flux incident on the boundary by the average Fresnel reflectance coefficient $F_{dr}(\eta)$. Equation (5.4) converts incident, external radiance into refracted, internal, diffuse radiance by scaling the incoming light by Fresnel transmittance coefficient $F_t(\eta, \vec{\omega})$ and an exponential term that approximates the absorption that occurs before the light becomes diffusive.

The final two steps of the derivation first substitute Equations (5.2) and (5.3) into Equation (5.1) and then substitute the diffusion approximation (Equation (2.7)) for $L_d(\mathbf{x}, \vec{\omega})$. The algebra of this derivation is presented in Appendix B. The result is the diffusive source boundary condition

$$\phi(\mathbf{x}) + 2A(\eta)\kappa_d(\mathbf{x})\big(\vec{n} \cdot \vec{\nabla}\big)\phi(\mathbf{x}) = \frac{4}{F_{dt}(\eta)}\Gamma_s(\mathbf{x}) \tag{5.5}$$

where the coefficient terms are the same as in the original Robin boundary condition (see Section 2.2.2.2). Because the DSBC models the reduced intensity source, the $Q_{ri}(\mathbf{x}, \vec{\omega})$ term can be dropped from the DE resulting in

$$-\vec{\nabla} \cdot \Big(\kappa_d(\mathbf{x})\vec{\nabla}\phi(\mathbf{x})\Big) + \sigma_a(\mathbf{x})\phi(\mathbf{x}) = Q^0(\mathbf{x}) \tag{5.6}$$

To close this section, note that the DSBC also simplifies how radiance is computed from fluence when solving the DE. Equation (2.18) gives the basic relationship but Equation (5.5) can now be used to eliminate the normal derivative.

$$L(\mathbf{x}, \vec{\omega}) = \frac{F_t(\eta, \vec{\omega})}{4\pi}\left[\left(1 + \frac{1}{A(\eta)}\right)\phi(\mathbf{x}) - \frac{4}{F_{dt}(\eta)A(\eta)}\Gamma_s(\mathbf{x})\right] \tag{5.7}$$

5.3 Finite Element Solution

This section solves the heterogeneous scattering problem, represented by Equations (5.6) and (5.5), derived in the last section using the finite element method. As described in Section 5.1, FE algorithms solve a PDE by fixing a space of trial solutions $\mathbb{H}$ and searching in that space for a function that best approximates the solution to the PDE. This method is motivated by the Lax-Milgram Theorem from functional analysis [Eva98].

Lax-Milgram Theorem. *Let $\mathbb{H}$ be a Hilbert space. Given a bilinear functional $\mathcal{H} : \mathbb{H} \times \mathbb{H} \to \mathbb{R}$ that is bounded and coercive, i.e. there exist constants $\alpha, \beta > 0$*

$$\forall u, v \in \mathbb{H}, \left|\mathcal{H}[u, v]\right| \leq \alpha \left\|u\right\| \left\|v\right\| \tag{5.8}$$

$$\forall u \in \mathbb{H}, \quad \beta \left\|u\right\| \leq \mathcal{H}[u, u] \tag{5.9}$$

and a linear functional $\mathcal{F} : \mathbb{H} \to \mathbb{R}$ there is exactly one function $v \in \mathbb{H}$ such that

$$\forall u \in \mathbb{H}, \ \mathcal{H}[u, v] = \mathcal{F}[u] \tag{5.10}$$

The Lax-Milgram theorem can be used to solve PDEs. The process converts the PDE into its *weak form*, a bilinear form and a corresponding linear form to which the Lax-Milgram theorem applies. The weak form generalizes the solution to the PDE. For the weak form, the function predicted by Lax-Milgram is either the solution to the PDE or the closest function in $\mathbb{H}$, in terms of its norm, to that solution. Conveniently, the Lax-Milgram theorem guarantees that such a weak solution exists and is unique. The FE method solves the weak form in the special case that $\mathbb{H}$ has a finite basis. In this case, the weak solution can be computed by solving a linear system. This section derives that system, Equation (5.16), for the diffusion equation (Equation (5.6)) and the diffusive source boundary condition (Equation (5.6)) . To make the notation in this section more compact, the independent spatial variable $\mathbf{x}$ has been omitted in all functions and η has been removed from all Fresnel terms.

5.3.1 Derivation of the Weak Form

Assume that $\mathbb{H}$ is an arbitrary space of functions and let $\theta \in \mathbb{H}$ be any function. Start by multiplying Equation (5.6) by θ and integrating over the entire scattering volume Ω.

$$-\int_{\Omega} \left[\vec{\nabla} \cdot (\kappa_d(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}))\right] \theta \, d\mathbf{x} + \int_{\Omega} \sigma_a \phi\theta \, d\mathbf{x} = \int_{\Omega} Q^0 \theta \, d\mathbf{x} \tag{5.11}$$

Taking the left-hand side as the bilinear form and the right-hand side as the linear form, Equation (5.11) is nearly what is required by Lax-Milgram. However, the first term is not yet in a convenient form. This can be corrected in two steps. First, using the divergence theorem,

Divergence Theorem. *Let* $\mathbf{v}$ *by any vector function and* u *be any scalar function then*

$$\int_{\Omega} \vec{\nabla} u \cdot \mathbf{v} \, d\mathbf{x} = \int_{\partial\Omega} u(\mathbf{v} \cdot \vec{n}) \, d\mathbf{x} - \int_{\Omega} u(\vec{\nabla} \cdot \mathbf{v}) \, d\mathbf{x} \tag{5.12}$$

integrate the first term by parts.

$$\int_{\Omega} \kappa_d \vec{\nabla}\phi \cdot \vec{\nabla}\theta \, d\mathbf{x} - \int_{\partial\Omega} \kappa_d (\vec{n} \cdot \vec{\nabla})\phi\theta \, d\mathbf{x} + \int_{\Omega} \sigma_a \phi\theta \, d\mathbf{x} = \int_{\Omega} Q^0 \theta \, d\mathbf{x} \tag{5.13}$$

Second, impose the DSBC (Equation (5.5)) by using it to eliminate the $\kappa_d(\vec{n} \cdot \vec{\nabla})\phi$ term in Equation (5.13). This gives the weak form.

Weak Form. *Find* ϕ *such that*

$$\forall\, \theta \in \mathbb{H}, \quad \int_{\Omega} \kappa_d \vec{\nabla}\phi \cdot \vec{\nabla}\theta \, d\mathbf{x} + \int_{\Omega} \sigma_a \phi\theta \, d\mathbf{x} + \frac{1}{2A} \int_{\partial\Omega} \phi\theta \, d\mathbf{x} = \int_{\Omega} Q^0 \theta \, d\mathbf{x} + \frac{2}{AF_{dt}} \int_{\partial\Omega} \Gamma_s \theta \, d\mathbf{x} \tag{5.14}$$

5.3.2 Derivation of the Matrix Equation

The final step of the FE solution converts Equation (5.14) into a linear system. To do this, first assume that $\mathbb{H}$ has a finite basis:

$$\mathcal{B}(\mathbf{x}) = \left\{\beta_0(\mathbf{x}), \beta_1(\mathbf{x}), \ldots, \beta_{n-1}(\mathbf{x})\right\}$$

Then the weak form is equivalent to ensuring that Equation (5.14) holds for each $\beta_i \in \mathcal{B}$. This is a system of $|\mathcal{B}|$ equations for ϕ.

$$\int\limits_{\Omega} \kappa_d \vec{\nabla}\phi \cdot \vec{\nabla}\beta_i \; d\mathbf{x} + \int\limits_{\Omega} \sigma_a \phi \beta_i \; d\mathbf{x} + \frac{1}{2A} \int\limits_{\partial\Omega} \phi \beta_i \; d\mathbf{x} = \int\limits_{\Omega} Q^0 \beta_i \; d\mathbf{x} + \frac{2}{AF} \int\limits_{\partial\Omega} \Gamma_s \beta_i \; d\mathbf{x} \qquad \forall \beta_i \in \mathcal{B} \quad (5.15)$$

However, by Lax-Milgram, $\phi \in \mathbb{H}$ and there exist some constants a_i such that $\phi = \sum_{i=0}^{n-1} a_i \beta_i$. Substituting this expression into Equation (5.15) results in an system of linear equations for the coefficient vector $\vec{\mathsf{a}}$ of ϕ.

Matrix Equation.

$$\mathsf{F}\vec{\mathsf{a}} = \vec{\mathsf{r}}$$

$$(\mathsf{D} + \mathsf{M} + \mathsf{S})\vec{\mathsf{a}} = (\vec{\mathsf{q}} + \vec{\mathsf{g}}) \qquad (5.16)$$

where

$$\mathsf{D}_{ij} = \int\limits_{\Omega} \kappa_d \vec{\nabla}\beta_i \cdot \vec{\nabla}\beta_j \; d\mathbf{x} \qquad \vec{\mathsf{q}}_i = \int\limits_{\Omega} Q^0 \beta_i \; d\mathbf{x}$$

$$\mathsf{M}_{ij} = \int\limits_{\Omega} \sigma_a \beta_i \beta_j \; d\mathbf{x} \qquad \vec{\mathsf{g}}_i = \frac{2}{AF_{dt}} \int\limits_{\partial\Omega} \Gamma_s \beta_i \; d\mathbf{x}$$

$$\mathsf{S}_{ij} = \frac{1}{2A} \int\limits_{\partial\Omega} \beta_i \beta_j \; d\mathbf{x}$$

5.4 Implementation

The next section analyzes a prototype implementation of the four-step algorithm discussed in Section 5.1. To describe that implementation, this chapter answers five questions:

1. How to mesh the domain to build a basis?
2. How to assemble Equation (5.16)?
3. How to solve the linear system?
4. How to prepare the material and source functions to avoid aliasing errors?
5. How to adaptively refine the mesh to ensure accuracy?

5.4.1 Tetrahedral Basis

The prototype implementation uses a piecewise-linear basis on a tetrahedral mesh. For this basis, the support of each basis function lies only in the one ring of tetrahedra sharing a central, support vertex. This limited support ensures that the FE matrix will be sparse and efficient to solve. The 2D analogue of these functions are commonly called “tent” basis functions. Tetrahedral elements are particularly advantageous because there are many, well-studied algorithms for quickly generating a high-quality, tetrahedral mesh within a triangular surface mesh. For other cell types, this is a more difficult problem. The prototype uses Tetgen [SG05] to generate its meshes.

5.4.2 Assembly of the Finite Element System

Given a tetrahedral mesh of the domain, Equation (5.16) can be written as a sum of integrals over the tetrahedra volumes and faces. For example, if $\mathcal{T}$ is the set of

```
SparseMatrix f_mat;
Vector r_vec;

matrix.zero();
rhs.zero();
foreach Tet t in mesh {
  foreach QuadPt pt in Tet {
    foreach Basis i in Tet {
      foreach Basis j in Tet {
        f_mat[i,j]
          += pt.wt*Kd(pt)*dot(grad(i,pt),(grad(j,pt)));
        f_mat[i,j]
          += pt.wt*sigA(pt)*value(i,pt)*value(j,pt);
      }
      r_vec[i] += pt.wt*src(pt)*value(i,pt);
  }}

  foreach Face f of Tet {
    if(f on boundary) {
      foreach QuadPt pt on f {
        foreach Basis i in Tet {
          foreach Basis j in Tet {
            f_mat[i,j]
              += (0.5/A)*pt.wt*value(i,pt)*value(j,pt);
          }
          r_vec[i] += (2/A)*pt.wt*gamma(pt)*value(i,pt);
}}}}}
```

Figure 5.4: Pseudo-code for the assembly algorithm. Here `Kd(pt)`, `sigA(pt)`, `src(pt)` and `gamma(pt)` are functions that return the values of $\kappa_d(\mathbf{x})$, $\sigma_a(\mathbf{x})$, $Q^0(\mathbf{x})$ and $\Gamma_s(\mathbf{x})$ respectively. The functions `grad(i,pt)` and `value(i,pt)` return the gradient and value respectively of the i^{th} basis function. Finally, `dot` computes a dot product and `pt.wt` is the weight of the quadrature point.

all tetrahedra and Ω_t is the volume of tetrahedron t, an entry in D_{ij} can be written as

$$\mathsf{D}_{ij} = \sum_{t \in \mathcal{T}} \int_{\Omega_t} \kappa_d \vec{\nabla}\beta_i \cdot \vec{\nabla}\beta_j \; d\mathbf{x} \tag{5.17}$$

However, the terms in this sum are only non-zero if the supports of β_i and β_j overlap. For the piecewise-linear basis, this happens only if vertices i and j are part of the same tetrahedron. Given this restricted domain of integration, the assembly of Equation (5.16) can be expressed as a loop over all tetrahedra. Further, in practice, integrals like Equation (5.17) are computed approximately using quadrature. In this case, computing each integral term reduces to computing the sum of the values of the integrand at a series of quadrature points. The results in Section 5.5 use 2^{nd} order 3D Gaussian quadrature. Combining these facts the system can be assembled by a small set of nested loops over tetrahedra, quadrature points and basis functions. Figure 5.4 contains the pseudo-code for the complete assembly algorithm.

5.4.3 Material and Source Projection

A side affect of the assembly process is the projection of the scattering parameters, $\kappa_d(\mathbf{x})$ and $\sigma_a(\mathbf{x})$, and the source functions, $\Gamma_s(\mathbf{x})$ and $Q^0(\mathbf{x})$, onto the FE basis. Since this operation uses a regular spacing of quadrature points, high frequencies in the material or source terms can cause aliasing in the projection and produce artifacts. To avoid this, the source and material terms are computed only at the vertices of the tetrahedral grid and interpolated within each cell. If the basis can represent the final solution to the scattering problem, this is a small approximation since the sub-cell detail causing the aliasing would ultimately be blurred, correctly, by the process of scattering.

5.4.4 Solving the Linear System

Satisfying the conditions of the Lax-Milgram theorem (Equation (5.10)) indirectly requires that the material parameters and source functions in the DE be bounded and have bounded 1$^{\text{st}}$ derivatives.[1] If these conditions are violated, the resulting FE matrix may not exist or be invertible. Moreover the approximation of the integrals by quadrature may result in a singular matrix even if the exact matrix can be inverted. Using the pre-projection discussed above will avoid both of these issues, but even in other cases, the FE method was empirically robust. For all tests of the prototype, including tests with discontinuous material and source functions, the matrix was non-singular. To actually perform the inversion, any sparse matrix algorithm could be used. The prototype uses the conjugate gradient algorithm with the symmetric successive over-relaxation (SSOR) preconditioner.

5.4.5 Adaptive Refinement

The last topic in this section is adaptive mesh refinement. There are two important issues: non-conforming meshes resulting from refinement and the refinement heuristics themselves.

5.4.5.1 Non-conforming Meshes

Adaptively refining a mesh creates T-junction vertices when neighboring tetrahedra have differing levels of refinement. These vertices are a problem because, when present, the span of the resulting basis contains discontinuous functions. These discontinuous functions violate implicit assumptions required to use the divergence theorem (see Equation (5.13)) to derive the weak form. With T-junctions, continuity

[1]This condition stronger than necessary. The functions need only be 1$^{\text{st}}$-order Sobelev functions but introducing Sobelev spaces is beyond the scope of this thesis. See [Eva98] for more details.

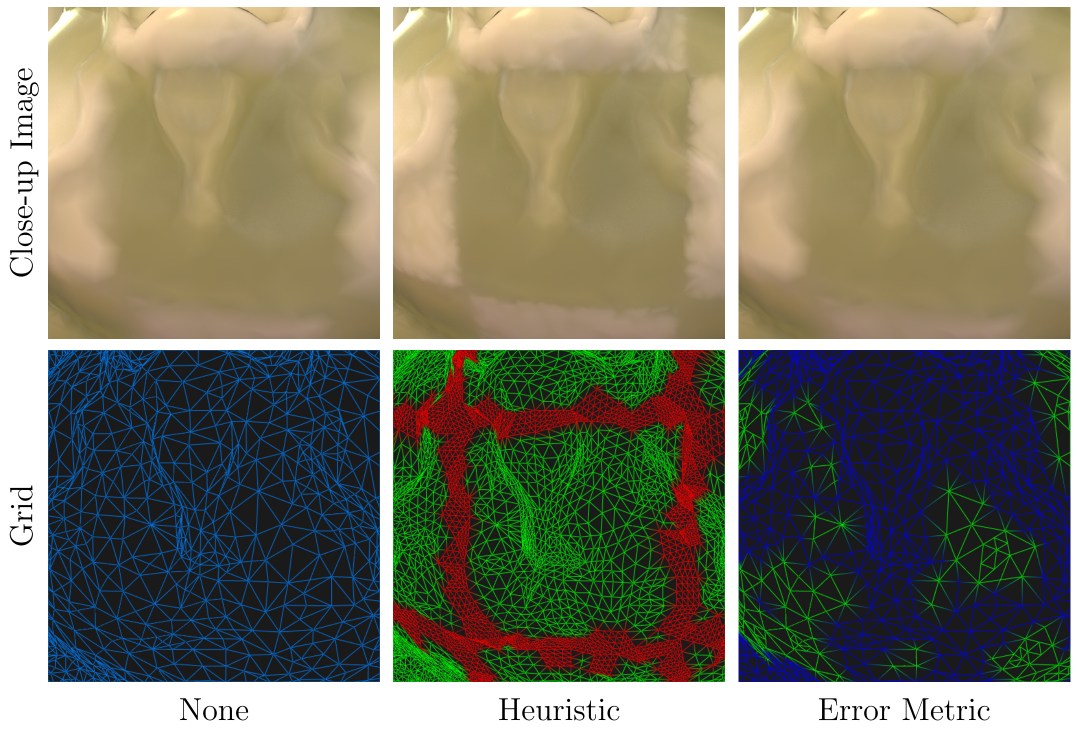


Figure 5.5: Evaluation of refinement metrics. Top row: Close-up images of the Buddha result: no refinement (left), heuristic refinement (center) and FE error metric refinement (right). Bottom row: visualization of the grid refinement levels for each image. Unrefined tetrahedra are blue, once refined tetrahedra are green and twice refined tetrahedra are red.

must be imposed by constraining the coefficients of the t-junction basis functions. Creating and enforcing these constraints is a complex, but solved problem in finite element analysis beyond the scope of this thesis (for a summary of a modern implementation see [BKH07]). The prototype uses LibMesh [KPSC06] which generates of these constraints automatically.

5.4.5.2 Refinement Heuristics

The prototype uses two simple refinement heuristics. It initially refines the mesh using the following conditions in order:

1. refine, once, all the tetrahedra visible from the camera; and then
2. refine, once, those tetrahedra with large differences in scattering parameters.

To implement the second condition, $\sigma_a(\mathbf{x})$ and $\sigma_s(\mathbf{x})$ are computed at all the vertices of the mesh. Then, for each tetrahedron, the largest percentage difference amongst all pairs of vertices is determined. All tetrahedra with a local difference greater than one deviation above the mean are refined.

In developing the FE algorithm, a more rigorous, mathematically-motivated error metric [dSRGKZB83] was also tested as a refinement heuristic. Given an initial solution on a coarse mesh, the metric bounds the error of each element and, using the metric, the radiance solution can be iteratively solved and refined. Unfortunately, for the particular problem of subsurface rendering, these metrics were less useful and much more expensive than the ad-hoc heuristics above. The heuristics have an advantage because they are chosen to exploit two features specific to the subsurface rendering problem. One, the solution tends to have large gradients near the boundary that are hard to represent in any basis [HST*94] and two, the solution will only be queried on the visible surface of the mesh. These advantages are illustrated in Figure 5.5. In the figure, the top row shows three close-up images of the Buddha model (see Section 5.5): a scene that particularly requires refinement because the material's sharp features. The bottom row shows the refined mesh used for each image. Comparing the unrefined image (left) to refined images rendered using equal-sized meshes created with both the error metric (left) and the heuristic refinement (center), it is clear that the heuristics better align with the material edges and produce a higher quality solution.

5.5 Analysis

This section demonstrates that the finite element algorithm is an efficient and accurate rendering method for general heterogeneous subsurface scattering problems. The analysis in this section assesses the results of the new renderer on a set of four

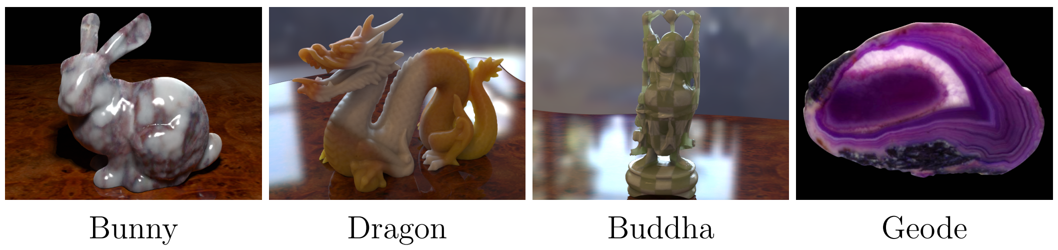


Figure 5.6: Four test scenes used to analyze the heterogeneous FE algorithm

distinct test scenes. This section has six parts. The first three describe the results themselves. In order, they describe the test scenes, the details of the rendering computation and the synthesis of the material parameters. The last three sections use these results to highlight the advantages of the new renderer. Section 5.5.4 itemizes the costs of the FE algorithm and notes that the new algorithm adds at most 3 minutes to rendering cost; Section 5.5.5 demonstrates that the test images are nearly identical to exact images produced with a MC path tracer; and Section 5.5.6 discusses the improvements the new algorithm makes over the most advanced previous method, Wang et al. [WZT*08].

5.5.1 Scenes

The prototype implementation was tested using the scenes shown in Figure 5.6. Each has a different geometry and material. The test scenes were chosen to demonstrate the range of effects a general, high-quality heterogeneous solver can reproduce.

Bunny uses a marble texture to simulate scattering in a complex, aggregate material. The bunny scene uses a smaller mesh, simple lighting and its rendering costs are presented with and without expensive global illumination. This scene emphasizes that, especially for simple lighting, the FE algorithm only adds a few seconds to the image cost.

Dragon is modeled using an optically thinner material similar to translucent plastic. This model shows that our solution can capture smooth changes in color and opacity.

Buddha contains a checkerboard of homogeneous marble and jade-like materials. The material is difficult because the solver must simultaneously capture the sharp edges in the material properties but also correctly simulate the smooth translucency in thinner geometry.

Geode pushes the limits of our FE algorithm. It demonstrates that the FE approach can easily scale to capture complex, high-frequency scattering in difficult lighting environments. In the Geode, the renderer is able to reproduce fine detail in the subsurface scattering even in the difficult case where all light passes through the material.

5.5.2 Details of the Rendering Computation

All results were generated on a 8 x 2.66Ghz Xeon workstation with 8GB of RAM. Tables 5.1(a) and 5.1(b) summarize the model size, mesh creation costs and rendering time for each scene. Of the results presented in Table 5.1, all costs are fully parallelized on all cores except the mesh creation, the mesh refinement and the FE matrix solution costs. For these tasks, parallel implementations were not available and these times are presented for a single core. The images are computed as three components (as described in Section 2.2.2.1): the surface reflection, single scattering and multiple scattering. The multiple scattering is solved using the new FE algorithm. The surface and single scattering are computed using a combination of Multidimensional Lightcuts (MDLC) [WABG06] and an analytical single scattering approximation [HK93]. The implementation is split between Java, which provides an implementation of MDLC, and C++ which

Table 5.1: Summary of the rendering costs and model parameters for our four test scenes: Bunny, Dragon, Buddha and Geode. Operations marked with † are run on a single processor.

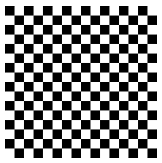
(a) Mesh sizes, generation cost and refinement cost. Costs range from 40-163 seconds.

Model	Initial Tets	Time†	Refined Tets	Time†
Bunny	427,918	40.8s	427,918	0s
Dragon	375,919	59.5s	1,084,599	73s
Buddha	429,158	79.0s	1,369,965	103s
Geode	849,763	60.0s	1,317,804	103s

(b) Rendering costs by model and category. Rendering times vary from 1 to 6 minutes.

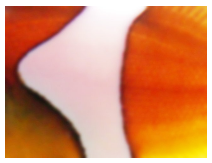
Model	Base Costs			FE Costs			Total
	Source	Surface	%	Assembly	Solve†	%	
Bunny (no GI)	4s	32s	72	9s	5s	28	50s
Bunny	17s	43s	81	9s	5s	19	74s
Dragon	28s	59s	45	35s	71s	55	193s
Buddha	29s	40s	36	44s	137s	72	250s
Geode	152s	108s	78	38s	88s	32	386s

provides an interface to the LibMesh [KPSC06] library. The LibMesh library is used only for the matrix solver and to iterate over the tetrahedra in the mesh. All other operations are performed outside of the library. LibMesh uses a basic linear algebra subprogram (BLAS) [LHKK79] implementation for all its linear algorithm operations. All images are 640x480 pixels. The Geode model is lit by a small area source, Buddha and Dragon are lit by the Kitchen environment map [Deb02] and the Bunny is lit by a small spherical source. All images, unless otherwise noted, include global illumination approximated by 100,000 virtual indirect sources.


(a)

(b)

(c)


(d)

(e)

Model & Param.		Range Min	Range Max	Base Scale	Tex.
Dragon	σ_a	(.05, .05, .05)	(1.0, 1.0, 1.0)	(0.75, 1.25, 1.75)	(d)
Dragon	σ_s	(.25, .25, .25)	(1.0, 1.0, 1.0)	(16.7, 16.7, 16.7)	(e)
Geode	σ_a	(.01, .01, .01)	(1.0, 1.0, 1.0)	(5.0, 5.0, 5.0)	(c)
Geode	σ_s	constant	constant	(5.0, 5.0, 5.0)	–
Buddha	σ_a	constant	constant	(1.63, 1.18, 4.5)	–
Buddha	σ_s	(.05, .05, .05)	(1.0, 1.0, 1.0)	(16.7, 16.7, 16.7)	(a)
Bunny	σ_a	(.05, .05, .05)	(3.5, 3.5, 3.5)	(7.8, 7.8, 7.8)	(b)
Bunny	σ_s	(.60, .60, .60)	(1.0, 1.0, 1.0)	(13.1, 15.7, 18.0)	(b)

Figure 5.7: Parameters used to procedurally generate the volume scattering textures.

5.5.3 Material Parameters

The material models used in the test scenes were synthesized by orthographically projecting an image through the scattering geometry and using the pixel values to determine values for the scattering coefficients σ_a and σ_s. The parameters used in this generation, as well as the source images, are provided in Figure 5.7. Each material coefficient is specified by three colors and an image. The first two colors, range min and max, are used to rescale the image to the dynamic range of the scattering parameter and the last color sets the parameter's overall scale. To compute a scattering parameter at a particular point, one finds the corresponding pixel in the rescaled image, inverts it and multiplies by the base scale. The inversion is necessary because the material parameters specify how much light is removed during scattering whereas, in the image, the colors specify how much light should be added to each pixel.

5.5.4 Algorithm Costs

Table 5.1 breaks down the cost of the new FE algorithm by component. From new scene to final image, the system requires between 2 and 10 minutes. This cost can be roughly divided into 2 parts. Table 5.1(a) gives mesh sizes and the costs of mesh generation and refinement. Because the FE algorithm is agnostic to how the scattering domain is discretized, it can use the most efficient meshing algorithms available. For unstructured triangular surface meshes, tetrahedralization is fast (1-2 minutes for all examples) and produces a high quality discretization [SG05]. Table 5.1(b) gives the cost to render each image after the mesh has been computed. For the four test scenes, these costs total between 50s and 6 minutes. These costs are further split into two categories. Depending only on the surface rendering algorithm and the scene's lighting, the base costs, computing the boundary source and rendering the surface component, are independent of the subsurface rendering algorithm. Only the costs of assembling Equation (5.16) and solving it are inherent to the new FE algorithm. For all test scenes, these costs total 3 minutes or less and, except for the Buddha image, account for half or less of the rendering cost. This compares with the base cost of rendering high-quality images without subsurface scattering and for a simple scenes where this base cost is smaller—like Bunny without complex global illumination (see Table 5.1(b))—the new algorithm adds only 15s to the total render time.

5.5.5 Comparison with Monte Carlo

To test the quality of the new algorithm, Figure 5.8 presents the images produced by the FE algorithm side-by-side with the same images produced by an exact, MC path tracer. Compared to previous work in subsurface rendering, there is impressive agreement between these two sets of images. However, neither set of

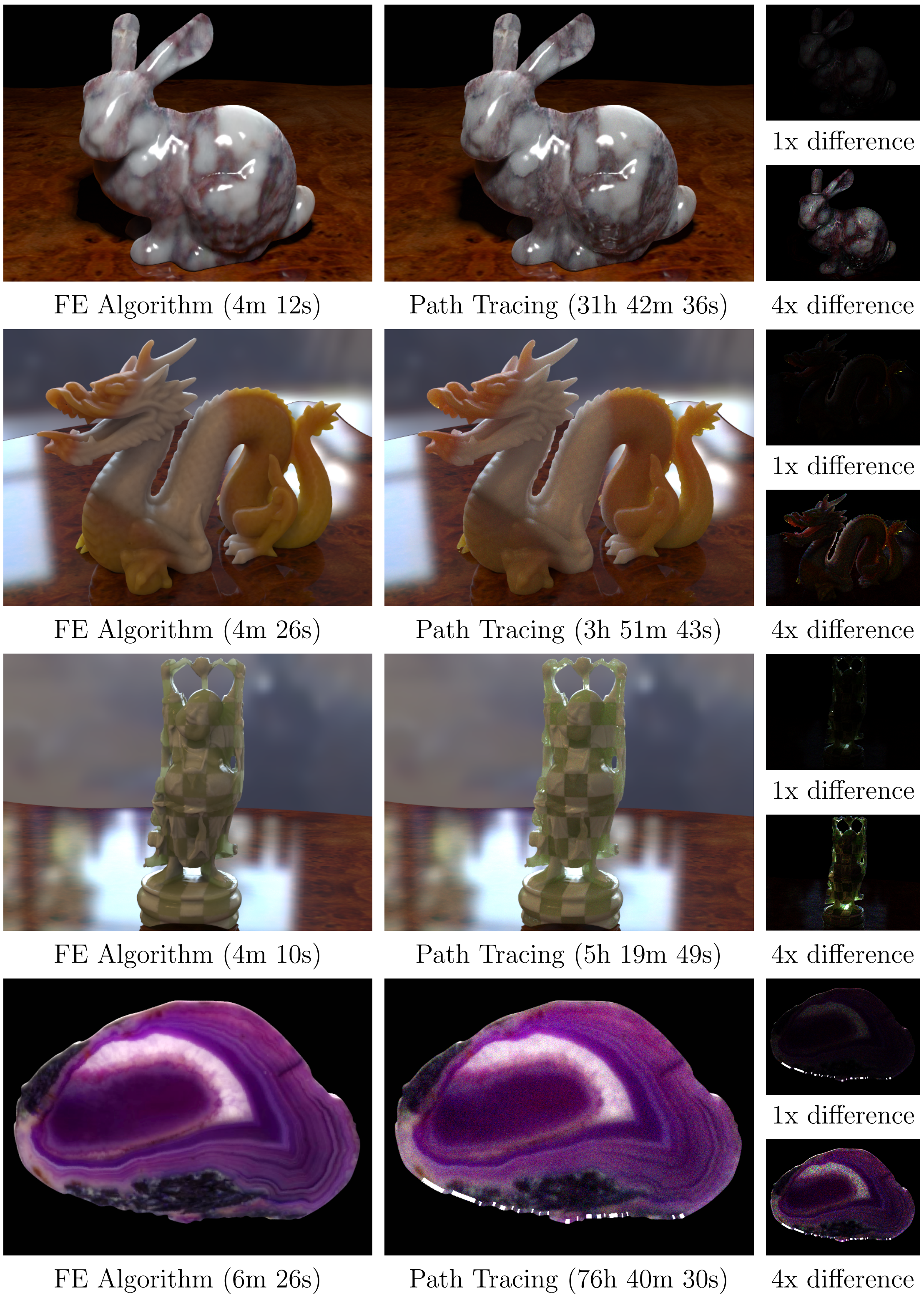
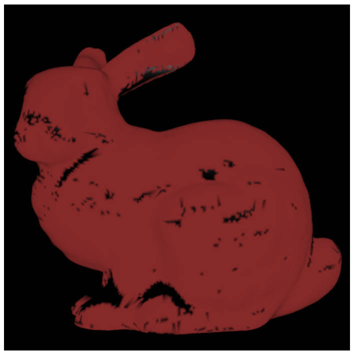
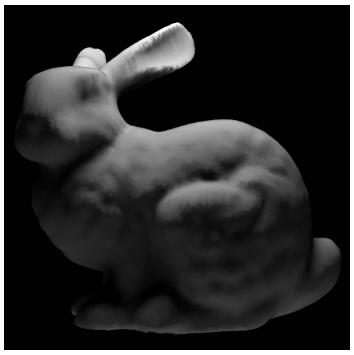


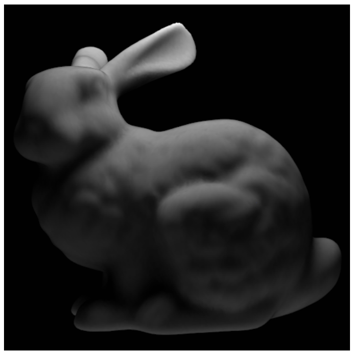
Figure 5.8: Comparison of the results of the new FE algorithm (right column) to exact path traced references (center) column with 1x and 4x magnified difference images (left column).

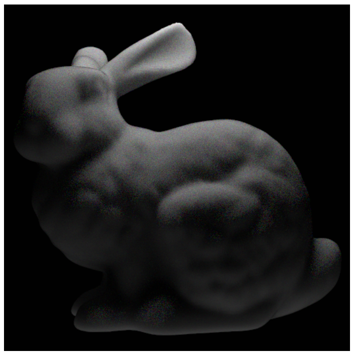
images is perfect. As noted throughout this thesis, the path tracing algorithm is impractically expensive. To avoid excessive computation, the path tracing was performed progressively and was stopped as soon as the noise was fell below a level permitting a reasonable comparison. Computing noise-free images would at least double the multiple-hour cost of the path traced references. As expected, the new algorithm has a considerable advantage in performance. It generates noise-free images in a few minutes.

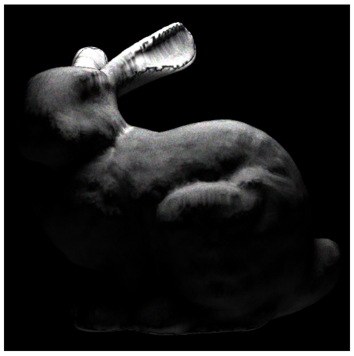
To facilitate the quality comparison, absolute error images are provided. These error images are nearly black so 4x magnified versions help reveal three differences. First, particularly prominent in the Buddha and Geode images, the path tracer is able to capture highlights from caustic paths that do not scatter within the material. There is currently no method for the FE algorithm to simulate these paths. Secondly, because the diffusion equation treats all diffusive radiance as nearly isotropic, it tends to overestimate scattering in thin geometry and near the surface. The effect is to diminish the contribution of low order scattering events. This slightly darkens regions when they are lit from behind as in the optically thinner parts of the Dragon and Geode and slightly lightens highly absorptive regions viewed directly, as in the darker checkers on the Buddha. Finally, because it projects the final solution onto a mesh basis, the FE algorithm slightly blurs the solutions. This is most evident in the Bunny whose marble material contains considerable high-resolution detail. However, overall, the differences are small and mostly due to fundamental limitations of the DE. Given the orders-of-magnitude difference in performance, these results are compelling evidence that the FE algorithm produces results acceptable even for high-quality rendering applications.


(a) FD original


(b) FD corrected


(c) FE


(d) MC


(e) 4x FD v MC error

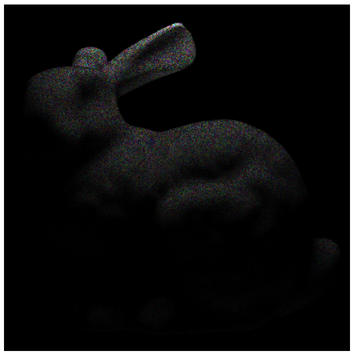

(f) 4x FE v MC error

Figure 5.9: Images of a constant scattering white bunny lit by two area lights, fill below and key above: (a) as described in Wang et al. with negative radiance areas highlighted in red; (b) Wang et al. corrected using derivation in Section 5.2; (c) our FE algorithm; (d) Monte Carlo reference; (e) 4x absolute error of (b); and (f) 4x absolute error of (c).

5.5.6 Comparison to Wang et al.

Finally, the new renderer is compared directly to the next best previous approach, Wang et al. [WZT*08]. Since the focus of the new FE algorithm is quality, rather than performance, the comparison is made to a software version of their iterative finite difference (FD) algorithm. The software renderer can dispense with several lower quality, performance optimizations required for interactive performance of the original renderer. Specifically, it does not use an approximate multi-resolution solution method. The software renderer performs each FD update step fully and iterates until the solution converges. To facilitate this comparison, Wang et al.

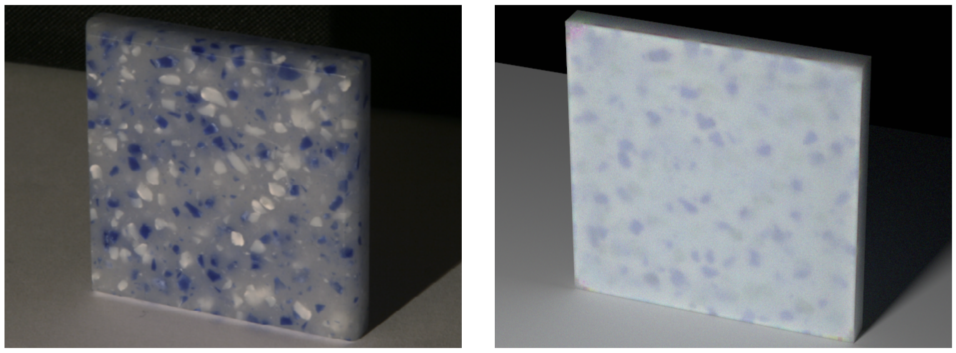


Figure 5.10: Comparison of the photograph artificial stone slab captured by Wang et al. to a path traced rendering of the resulting material parameters. Wang et al. have confirmed that this comparison is accurate.

kindly provided their measured material data and a PolyGrid [THCM04] bunny model. As an initial test, several images of the bunny with a white, homogeneously scattering material (see Figure 5.9) were created.

Unfortunately, the formulation of the heterogeneous scattering problem solved in the original work is not physically accurate. This causes the algorithm to sometimes compute negative radiance. As shown in red in Figure 5.9(a), this happens almost everywhere on the homogeneous bunny. In the original work, this error was likely corrected by the computation of the material parameters during the capture optimization and, as a result, this problem did not manifest itself in any of the authors' original results. As evidence of this capture correction, Figure 5.10 compares a path traced rendering of the captured artificial stone material with a photograph of the original object. The lack of saturation in the MC result suggests that the acquired parameters have significantly less absorption, making the image brighter. The authors of [WZT*08] have confirmed the mismatch in Figure 5.10.

In order to further discuss the FD approach, the rest of the results in this section were generated with a corrected FD renderer uses the correct formulation described in Section 5.2. The remaining images in Figure 5.9 directly compare, for meshes of equal size, Wang et al.'s FD algorithm, the new FE algorithm and a path traced reference (Figures 5.9(b), 5.9(c) and 5.9(d) respectively). Figures 5.9(e) and 5.9(f) display the error of the FD and FE methods respectively. Since both algorithms depend on the DE, neither solution can produce an exact answer. However, because the FD algorithm relies on a special PolyGrid mesh [THCM04], it has at least three additional sources of error.

1. The boundary condition is enforced only approximately by using special, smaller PolyGrid cells on the boundary.
2. The overall distortion of the PolyGrid can be only approximately modeled during the FD solution.
3. Creating the uniformly connected PolyGrid required of the FD solver requires deleting some nodes to along all boundary edges of the grid. This introduces error in the solution near these edges and especially at the grid corners.

Not only does the PolyGrid introduce error, it is expensive to create. The costs of mesh construction were not presented in [WZT*08] but the authors noted that the construction required optimizations performed by hand. Using the FE algorithm, mesh generation is automated, requires only a few minutes of computation (see Table 5.1(a)) and does not have these accuracy issues.

The iterative FD algorithm can also be unstable for certain materials. In Figure 5.11 the Bunny model is rendered with three similar materials using both the FD and FE algorithm. The first material (left) is homogeneous and the two methods are mostly in agreement. However, in the center and right columns, a checker board is introduced by scaling the mean free path (MFP) of the material in alternating sections. As this happens, the FD method diverges. In the middle and right columns the MFP has been reduced by factors of 3x and 4x respectively. For these cases, the FD algorithm was stopped after 500 iterations before the fluence begins

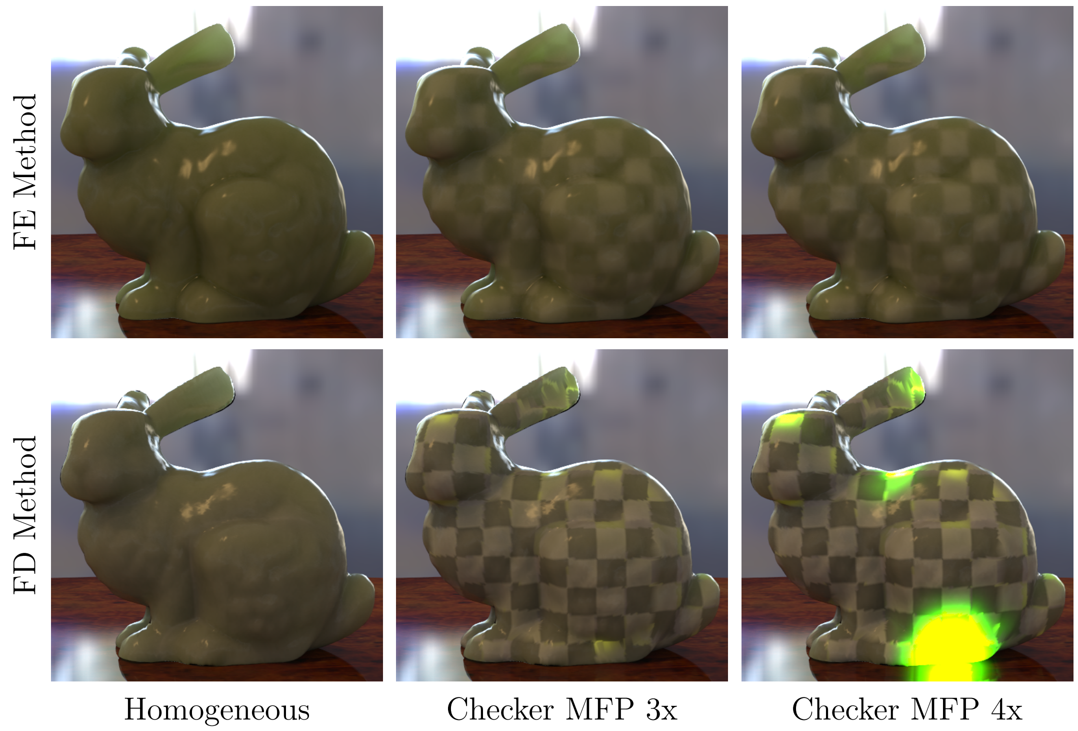


Figure 5.11: Demonstration of FD method's divergence. The bunny's material is varied from a homogeneous green material (left) towards the material used in the Buddha (see Figure 5.6) by scaling the scattering coefficient in the lighter squares. The FD algorithm diverges beyond a 3x scale (middle; see spots in ear, head and foot) and prominently at a 4x scale (right).

to overflow. In both images, there is divergence is beginning in the ears, head and foot. Unlike the Wang et al.'s FD algorithm, the new algorithm can handle this material well. The Buddha model (see Figure 5.8) uses the same material with a relative MFP scale of 20x.

Finally, Figure 5.12 directly compares the FD algorithm (top right), the FE algorithm (top center) and the path traced reference (top left) for the Bunny scene. For this example, the FD algorithm fails to capture most of the detail in the marble material, has meshing errors visible around the tail and is clearly less accurate.

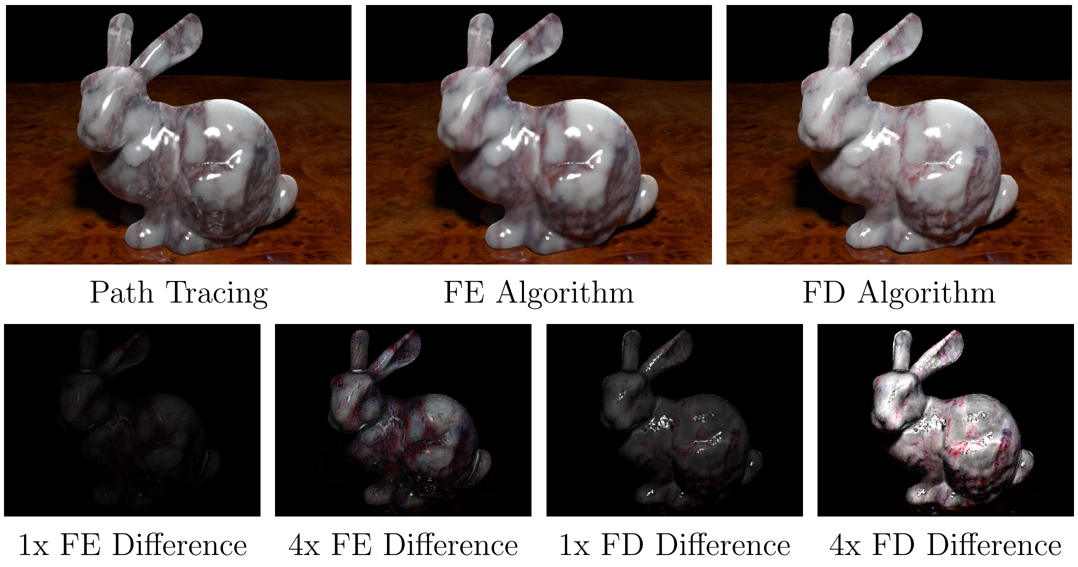


Figure 5.12: Comparison of path traced reference with FE and FD solutions. Top row: Path traced reference; middle row: FE algorithm (left) and FD algorithm (right); and bottom row: 1x and 4x absolute differences for FE algorithm (left side) and 1x and 4x absolute differences for FD algorithm (right side)

5.6 Summary

This chapter presented the first efficient, general, high-quality algorithm for rendering complex heterogeneous materials. It solves a carefully derived heterogeneous subsurface scattering problem (Section 5.2) accurately using the finite element method (Section 5.3). Using this solution, subsurface scattering can be computed by a simple four-step algorithm (Section 5.1). To validate this new algorithm as a solution for high-quality rendering applications, a prototype implementation (Section 5.4) was tested on a series of four difficult scenes (Section 5.5). The results demonstrate that the new algorithm can render images in a few minutes (Section 5.5.4) that are nearly identical to accurate images produced, in hours, by exact path tracing algorithms (Section 5.5.5) and significantly improve upon the quality and generality of the best previous method (Section 5.5.6).

CHAPTER 6

DISCONTINUOUS GALERKIN ALGORITHM

The previous chapter derived and analyzed a particular finite element solution to the heterogeneous diffusion equation (DE) with the diffusive source boundary condition (DSBC). Though this solution renders high quality images efficiently and accurately, an obvious next question asks whether a more sophisticated finite element formulation might yield an improved rendering algorithm. During the development of this thesis, one particular alternative forumulation, the discontinuous Galerkin (DG) finite element (FE) method, was derived and tested in detail. Unlike the FE solution in the previous chapter, which requires that the FE basis span only continuous functions, the DG FE method can find discontinuous solutions to the DE. Because of this, it was hypothesized that the DG method might produce more accurate answers to problems with highly discontinuous materials, such as in the Buddha (see Figure 5.6).

However, because of the diffusion approximation, using the DE as a scattering model adds some minimum error to all images. Improved finite element solutions of the DE are useful only if they decrease other sources of error significantly relative to this minimum threshold. Surprisingly, the continuous FE algorithm, presented in the previous, chapter produced images close to this threshold even for discontinuous materials. Because of the unexpected accuracy of the continuous solution, tests using the DG method did not considerably improve rendering quality. Since the DG algorithm is considerably more expensive than its continuous counterpart, its utility for rendering is limited. Though this result is negative, analyzing the DG method here helps place an upper bound on the accuracy required of finite element solutions in subsurface rendering applications. Looking forward, future work should focus only on matching the accuracy of the continuous renderer which seems, at least empirically, to achieve close to the minimum error provided by the DE. Spending more effort on the finite element solution, such as by using the more expensive DG FE method, is wasteful.

6.1 Discontinuous Galerkin Overview

At a high level, the DG algorithm makes only one change to the continuous algorithm: the continuous algorithm uses basis functions that overlap several tetrahedra while the discontinuous algorithm uses basis functions that are restricted to a single tetrahedron. However, this change has an important consequence. In the continuous case, the overlap blends the local solution in each tetrahedron with the solutions in its neighbors. This blending, by construction, ensures that the final solution is continuous. However, by making the the basis functions private, the DG algorithm can find solutions with certain limited discontinuities. The DG solutions are continuous within each tetrahedron but, since each has its own separate basis functions, the overall solution can be discontinuous between tetrahedra. Conveniently, any continuous overlapping basis can be converted into a corresponding discontinuous basis by duplicating the continuous basis function once for each tetrahedra it overlaps. One copy is designated a private basis function in. By using this conversion, the DG method shares the same types of basis functions—and consequently can share meshes and mesh refinement methods—with the continuous FE algorithm.

The similarity of the basis functions and meshes means that there is essentially no difference between the mechanics of the continuous and discontinuous algorithms. Both use the same four-step structure outlined in Section 5.1: build a mesh, construct a linear system, solve the system and query the radiance as needed. The only change is that discontinuous method constructs a different (and larger; due to the basis function duplication) linear system. The mesh, the matrix solver and the query function all remain the same. Moreover the new linear system is still a single closed form equation (Equation (6.1)) that is assembled by iterating over the mesh tetrahedra and their faces. Thus, relative to the continuous algorithm, the new DG algorithm can be completely described by introducing the new DG linear system and a new assembly method (see Section 5.4.2 for the continuous case) that constructs it. The next section describes these elements of the DG formulation and then Section 6.3 analyzes the limitations of the DG solution for rendering applications.

6.2 Discontinuous Matrix Equation and Assembly

Because derivation of the DG matrix equation is much longer than the continuous case, the goal here is only to introduce the features of the matrix equation necessary to analyze its limitations as a rendering solution. The mathematical details are presented separately in Appendix C. There are four parts of this discussion. The first introduces the problem of trivial solutions that make the DG derivation more involved. The second outlines the major steps of the complete derivation and introduces the new DG matrix, Equation (6.1). The third describes matrix assembly and the fourth discusses two important properties of Equation (6.1).

6.2.1 Trivial Discontinuous Solutions

In the continuous FE solution, discussed in Section 5.3, the Lax-Milgram theorem proved that there is exactly one weak solution to every 2nd-order, elliptic PDE (like the DE). However, in a general sense, this is not true in the discontinuous case. To understand why, recall that a FE algorithm finds the weak solution by searching a chosen space of functions $\mathbb{H}$ for a special function that best satisfies both the DE and its boundary condition, the DSBC. In the DG method, $\mathbb{H}$ is expanded. The larger space $\mathbb{H}^{\mathrm{dg}}$ includes $\mathbb{H}$ but additionally contains a larger set of discontinuous functions. Because of its larger size, in $\mathbb{H}^{\mathrm{dg}}$ another solution can be constructed: use the continuous solution on the boundary to satisfy the DSBC and then immediately jump discontinuously to the zero function everywhere else. Since the zero function always satisfies the DE, this is a mathematically valid, but physically useless, solution to the FE problem. To avoid finding these trivial solutions, the derivation of the DG weak form must penalize discontinuous jumps in potential solutions. If these penalties are chosen carefully they exclude useless trivial solutions but allow the DG formulation to still find novel solutions that cannot be found by the continuous FE method. The next subsection outlines how the DG matrix equation is derived by choosing these penalties.

6.2.2 Derivation Overview

The derivation of the DG method has five basic steps:

Step 1: Diffusion System

To generate a well defined solution, jumps in both the fluence $\phi(\mathbf{x})$ and its gradient, the vector irradiance $\vec{E}(\mathbf{x})$, must be penalized. To allow these penalties to be applied, the DE must first be reformulated as a system of two PDEs, one for $\phi(\mathbf{x})$ and one for $\vec{E}(\mathbf{x})$, so these jumps can be made explicit.

Step 2: Weak System

Analogous to the weak form in the continuous case (see Section 5.3.1), a weak system is created for these two simultaneous PDEs. This process explicitly creates the two jump terms that will later be penalized.

Step 3: Primal Form

Because a weak system is inconvenient for the final discretization, this step is converts the weak system back into a single equation called the *primal form*.

Step 4: Interior Penalty Primal Form

A new primal form, the *interior penalty primal form*, is created from the basic form by choosing a specific penalty function for each of the jump terms.

Step 5: Discontinuous Galerkin Matrix Equation

The final DG matrix equation is derived by discretizing the interior penalty primal form using a finite basis.

As mentioned above, the mathematical details of the derivation are presented in Appendix C. Here only the final result is presented.

Discontinuous Galerkin Matrix Equation.

$$(\mathsf{D} + \mathsf{M} + \mathsf{S} + \mathsf{E})\vec{\mathsf{a}} = \vec{\mathsf{q}} + \vec{\mathsf{g}} \tag{6.1}$$

where

$$\mathsf{D}_{ij} = \int\limits_{\Omega} \kappa_d \vec{\nabla}_{\mathsf{p}}\, \beta_i \cdot \vec{\nabla}_{\mathsf{p}}\, \beta_j \; d\mathbf{x}; \qquad \mathsf{M}_{ij} = \int\limits_{\Omega} \sigma_a \beta_i \beta_j \; d\mathbf{x}; \qquad \mathsf{S}_{ij} = \frac{1}{2A} \int\limits_{\partial\Omega} \beta_i \beta_j \; d\mathbf{x};$$

$$\mathsf{E}_{ij} = -\int\limits_{\mathcal{F}_{\mathcal{I}}} \tfrac{1}{2}\kappa_d^* \big(\vec{\nabla}_{\mathsf{p}}\, \beta_j \cdot \beta_i \vec{n}_i + \vec{\nabla}_{\mathsf{p}}\, \beta_i \cdot \beta_j \vec{n}_j - \eta \beta_i \vec{n}_i \cdot \beta_j \vec{n}_j \big) \; d\mathbf{x};$$

$$\vec{\mathsf{q}}_i = \int\limits_{\Omega} Q^0 \beta_i \; d\mathbf{x}; \qquad \vec{\mathsf{g}}_i = \frac{2}{AF_{dt}} \int\limits_{\partial\Omega} \Gamma_s \beta_i \; d\mathbf{x}$$

6.2.3 Assembly

To assemble Equation (6.1), start with the basic assembly algorithm as in Figure 5.4 then add an pseudoelse clause to the `if` statement in the lower loop section (see Figure 6.1). In this clause, the assembly must iterate over all pairs of basis

```
else {
  BasisSet leftSet = left(f);
  BasisSet rightSet = right(f);
  foreach Basis i in leftSet {
    foreach Basis j in rightSet {
      Area area = overlap(f,i,j);
      if(area.size() == 0)
        continue;
      foreach QuadPt pt in area
        f_mat[i,j] += jumpterm(pt,i,j);
}}}
```

Figure 6.1: Additional pseudo-code for discontinuous Galerkin matrix assembly. This code should be inserted in as the `else` clause in the if statement in Figure 5.4. Here `left` and `right` are functions that return sets of basis functions that project onto a face from the left and right sides respectively; `overlap` is a function that returns the overlap region between the projections on a face of a pair of basis functions; and `jumpterm` is a function that returns the value E_{ij} integrand in Equation (6.1).

functions that have overlapping, non-zero projections on the current face. It is important to note that this operation is not trivial. Because of adaptive refinement, given a face of a single element, the opposite side of the face may project onto only part of an element face or many element faces if the levels of refinement of the current element and its neighbors are different. However, once the area of overlap is computed for each pair, the E_{ij} term is easily estimated by quadrature. To assemble Equation (6.1), start with the basic assembly algorithm as in Figure 5.4 then add an pseudoelse clause to the `if` statement in the lower loop section (see Figure 6.1). In this clause, the assembly must iterate over all pairs of basis functions that have overlapping, non-zero projections on the current face. It is important to note that this operation is not trivial. Because of adaptive refinement, given a face of a single element, the opposite side of the face may project onto only part of an element face or many element faces if the levels of refinement of the current element and its neighbors are different. However, once the area of overlap is computed for each pair, the E_{ij} term is easily estimated by quadrature.

6.2.4 Properties of the Discontinuous Matrix

Before the analysis in the next section, it is important to highlight two essential properties the final matrix equation. First, notice that the DG equation subsumes the continuous equation (Equation (5.16)). Except for the addition of the E_{ij} term, the two equations are identical. Intuitively this should be true because the basic terms model the proper physical behavior required within each element and along the boundary. The new term E_{ij} describes the penalty of the discontinuous jumps along the boundaries of the elements in the mesh (see Section C.3 for the definitions of the operators in E_{ij}) allowing discontinuous solutions.

Second, the most important property of Equation (6.1) is the penalty coefficient η in E_{ij}. The penalty coefficient trades the stability of the matrix solution for flexibility of the DG method. If the penalty coefficient is zero, Equation (6.1) is singular. If the penalty coefficient is small but non-zero, the DG method can find solutions with large discontinuities but the matrix may not be stable (i.e. the condition number of Equation (6.1) tends to infinity as $\eta \to 0$). Finally, as $\eta \to \infty$, the solution to Equation (6.1) converges to the continuous solution (Equation (5.16)). As will be discussed in the next section, the limits of the penalty coefficient are an important limiting factor of the DG formulation itself.

6.3 Summary and Analysis

Given the above, the continuous finite element system from the previous chapter can be altered to solve the DG diffusion problem. Figure 6.2 illustrates a typical result of the DG algorithm. The example shows a cube of checkerboard scattering

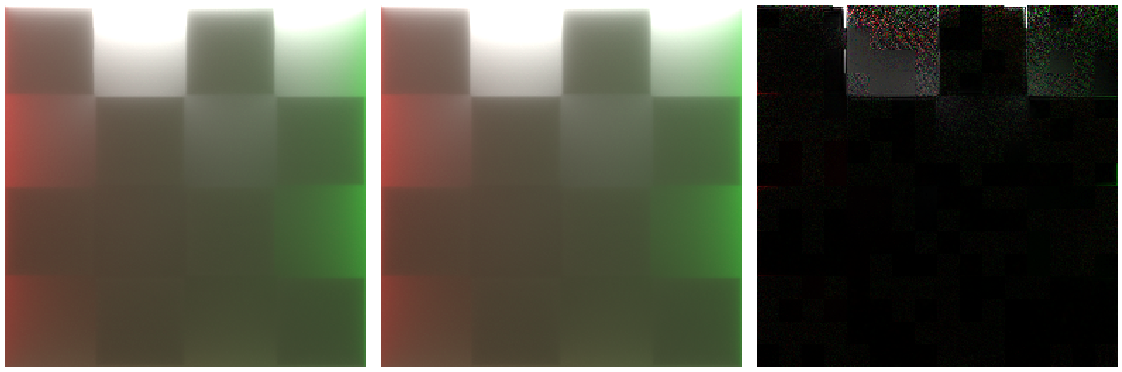


Figure 6.2: Comparison of continuous (left) and discontinuous Galerkin (middle) finite element solutions to the diffusion equation. Even the 10x difference between the two images (right) is negligible.

material lit be red, white and green lights. The images were produced using a special mesh so that the interior mesh faces exactly align with the checkerboard material. However, though this example was specifically chosen to highlight any advantages of the DG solution, the results of the continuous and discontinuous algorithms are nearly identical (see the 10x difference image). Given that the DG image took almost 8x longer to solve, the typically close agreement between these images argues against using the DG method for rendering.

The original hypothesis, that the DG method would produce better images, has two faults: the limitations of the DG method and the advantages of the continuous method. First, as noted in Section 6.2.4, the DG method can solve for discontinuous solutions but only at the cost of stability. For example, altering the penalty coefficient to make the discontinuities in Figure 6.2 sharper causes the conjugate gradient solver to diverge. To be useful for a large range of scenes, the penalty coefficient must be set sufficiently high to avoid this first problem, but this limits the range of new and interesting discontinuous solutions that can be found. Ongoing research in numerical methods is attempting to solve this problem with better penalty functions[1] and localized choices of penalty coefficients [KK05]. The second fault of the DG hypothesis is simply that the continuous FE formulation produces better images than expected. Because they violate the basic diffusion approximation, discontinuous functions tend to be poor solutions to the diffusion equation. Because of this, even when the material is discontinuous, the diffusion equation often favors continuous solutions that can be found by either algorithm.

This last result is a compelling promotion for the algorithm in Chapter 5. Despite a more general matrix equation (Section 6.2.4 and Appendix C) and more basis functions (Section 6.1), using the the DG method cannot significantly improve upon the simple, accurate and efficient result using continuous finite elements.

[1]As compared to the choice, Equations (C.25) and (C.26), made when deriving Equation (6.1).

CHAPTER 7

FUTURE WORK AND CONCLUSIONS

The two new algorithms in this thesis address the two most significant limitations of previous approaches to subsurface rendering: scalability and heterogeneity. By intelligently clustering complete subsurface light paths using sample triples, the first algorithm, a new scalable, homogeneous renderer, is able to reduce the cost of adding subsurface scattering to complex scenes three hundredfold. Then the second algorithm, a new finite element (FE) rendering method, enabled the first general, efficient and high-quality rendering system for complex, heterogeneous materials. It can render, in minutes, images nearly identical to exact images taking hours. This final chapter reviews the advantages of these two new algorithms in two sections. The first looks forward and discusses how the new ideas from this thesis can be used to create even more efficient, scalable and accurate renderers. The second looks backwards and concludes with a summary of the novel contributions of the works presented here.

7.1 Limitations and Future Work

The most promising future work in subsurface rendering lies in heterogeneous rendering algorithms. Homogeneous rendering using the dipole diffusion BSSRDF is a well-studied problem. With the introduction of the new scalable algorithm, renderers can both approximate homogeneous materials in real-time [MKB*03b, HBV03, MKB*03a, CHH03, HV04, DS03, LGB*02] as well as efficiently render large, complex scenes with many hundreds of subsurface scattering objects. However, the dipole diffusion model has it obvious limitations and there is a continuing need for more advanced models that can render more complex and realistic materials. The new FE algorithm is an ideal starting point for these new applications since, in its initial tests, it was accurate enough for even high-quality rendering and it was efficient. Moreover, it is backed by a rich set of mathematical tools that can be applied to extend the basic algorithm further in both efficiency and accuracy. This section discusses some of these possible extensions as well as some of the limitations of each new algorithm. This section has two subsections. First a short subsection discusses small extensions of the scalable homogeneous renderer while the second subsection discusses a range of new topics introduced by the FE renderer.

7.1.1 Scalable Homogeneous Rendering

Compared to the less scalable, two-pass methods, the ability of the new unified algorithm to focus effort on only the important subsurface light paths leads to significant increases in performance. However, the overall efficiency of the new algorithm strongly depends on the speed and accuracy of its adaptive cut refinement. Though the refinement heuristics chosen in Section 4.4.3 attempt to find the smallest cut as fast as possible and they worked well in all tests, they are still ad-hoc choices limited by two factors.

First, the error bounds—the basis for the refinement choices—are very conservative. In the initial stages of refinement, all clusters have high error and the algorithm essentially chooses refinements randomly. Efficient and accurate estimates for these bounds could significantly increase refinement performance. Second, the refinement heuristics themselves can be more advanced. Currently, the refinement algorithm bases all choices solely on the properties of the cluster to be refined. However, correlation between clusters in the cut is a new problem introduced by sample triples. Repeated refinement of, for example, the irradiance cluster leads to many triple clusters with different irradiance sample clusters but the same light sample cluster. This type of cut can be particularly problematic if the light cluster is a poor approximation of the local lighting. By considering all the clusters in the cut and the relationships between them, it may be possible to explore the space of potential cuts more efficiently and avoid these correlation problems.

7.1.2 Heterogeneous Rendering

The results in Chapter 5 demonstrated that the finite element (FE) method is a useful tool for solving general, efficient, high-quality and heterogeneous subsurface scattering problems. Then, in Chapter 6, the analysis of the limitations of the discontinuous Galerkin (DG) FE formulation suggested that the accuracy of the continuous formulation presented in Chapter 5 might be empirically close to the limit imposed by using the diffusion equation (DE). Looking forward, solving subsurface scattering using finite elements provides a framework for investigating a range of subsurface rendering algorithms. However, as suggested in Chapter 6, there is a floor beyond which further improvement in the FE solution to the DE has no effect on image quality. Beyond this limit, quality can only be further improved by restricting use of the DE to only those parts of the subsurface simulation where it is most accurate.

7.1.2.1 Restricted Diffusive Simulation

In subsurface rendering, the solution need only be accurate on the boundary of the object. Unfortunately, this is precisely the region where the DE is most fundamentally limited. Both the source model and the boundary condition introduce additional approximations near the edge of the material. Taking a lesson from previous work [LPT05, CTW*04], a subsurface rendering algorithm could be made more accurate by using a different simulation method in a thin shell near the boundary and a more approximate diffusion-based method in the interior regions.

However, in heterogeneous materials, a thin shell algorithm must overcome two additional problems. First, the necessary shell thickness must vary with the material parameters making it more difficult to choose when to switch between accurate shell simulation and less accurate interior simulation. Second, the shell algorithm must not become the simulation bottle neck. For example, using path tracing in the thin shell as in [LPT05] would likely be impractical. One promising method of avoiding this problem would be to couple two FE solutions, a more accurate one in the thin shell and a less accurate one in the interior.

7.1.2.2 Heterogeneous Scalability

Unfortunately, the new FE algorithm has the same scalability limitation as previous, homogeneous two-pass algorithms. In both cases, the algorithms pre-compute a detailed representation of the incoming reduced intensity source and using that representation solve for the subsurface scattering in a second step. As discussed in Chapter 4, this structure prevents scalability because, as the scene grows in size, the initial source computation becomes prohibitively expensive. In the homogeneous case, scalability was ensured by isolating only a fraction of the subsurface paths within a single object that needed to be simulated. However, the FE algorithm solves for all paths at once in a single computation so this approach is impossible. Thus, solving this scalability problem in FE algorithms is a formidable future challenge.

7.2 Conclusions

The goal of this thesis was to extend the state-of-the-art in subsurface rendering beyond the two fundamental limitations of the dipole diffusion BSSRDF. First, dipole diffusion algorithms use a basic two-pass structure that prevents these algorithms from scaling to large complex scenes with many subsurface scattering objects. Second, the dipole diffusion BSSRDF is fundamentally limited to homogeneous materials. The two new algorithms in this thesis overcame each of these problems respectively.

First, Chapter 4 introduced a new unified and scalable subsurface rendering algorithm that generalized Lightcuts [WFA*05] and Multidimensional Lightcuts [WABG06] to subsurface scattering problems by representing subsurface light paths using triples of pre-computed samples. The new algorithm partitioned these triples into clusters by selecting a cut through an implicit hierarchy of clusters of sample triples. By approximating the contribution of each of the clusters in a cut, the algorithm dramatically reduces the cost of estimating the subsurface scattering in a pixel. It then further ensures efficiency by sharing expensive form factor evaluations between pixel computations by using a new form factor cache. Combining these new approaches, the new scalable algorithm reduced the cost of subsurface simulation by up to a factor of 300 and because of this dramatic performance increase was able to render large complex, scenes with highly detailed geometry and expensive lighting effects that were beyond the reach of previous approaches.

Second, Chapter 5 introduced a new FE algorithm for rendering subsurface scattering in complex heterogeneous materials. In this chapter, the correct formulation of heterogeneous scattering using the diffusion equation was carefully derived and then solved using a continuous FE method. The result was a simple four-step algorithm for heterogeneous rendering. Section 5.5.6 demonstrated that this new FE algorithm is more accurate than even the best previous system [WZT*08] and has no limits on the materials and geometry it can render. Using this algorithm, a prototype rendering system created in minutes images of a range of difficult heterogeneously scattering objects that were nearly identical to exact images requiring hours of simulation. Finally, to promote the accuracy and efficiency of the continuous solver presented in Chapter 5, Chapter 6 discussed why the new algorithm empirically nears the accuracy limit imposed by the diffusion equation.

Together these two algorithms represent significant improvements in the scale, scope and quality of subsurface rendering. They make it now possible to accurately and quickly render a wide range of scenes and include the full range of subsurface scattering materials. Finally, both—especially the finite element algorithm—provide interesting new avenues for solving the difficult problem of subsurface rendering.

APPENDIX A

ADDENDA TO DIFFUSION EQUATION DERIVATION

This appendix gives the extended algebraic derivations of Equations (2.13) and (2.14) quoted in Section 2.2.1.3. Both derivations are manipulations of Equation (2.12)

$$\begin{aligned}\big(\vec{\omega}\cdot\vec{\nabla}\big)L_d(\mathbf{x},\vec{\omega}) = \sigma_s(\mathbf{x})\int\limits_{4\pi} p(\vec{\omega},\vec{\omega}')L_d(\mathbf{x},\vec{\omega}')\,d\vec{\omega}' \\ -\sigma_t(\mathbf{x})L_d(\mathbf{x},\vec{\omega}) + Q_{ri}(\mathbf{x},\vec{\omega}) + Q(\mathbf{x},\vec{\omega})\end{aligned} \tag{A.1}$$

reprinted here for convenience.

A.1 Derivation of Equation (2.13)

Equation (2.13) is derived by equating the 0th moments of the terms in Equation (2.12). Computing those moments yields

$$\begin{aligned}\int\limits_{4\pi}\big(\vec{\omega}\cdot\vec{\nabla}\big)L_d(\mathbf{x},\vec{\omega})\,d\vec{\omega} = \sigma_s(\mathbf{x})\int\limits_{4\pi}\int\limits_{4\pi} p(\vec{\omega},\vec{\omega}')L_d(\mathbf{x},\vec{\omega}')\,d\vec{\omega}'\,d\vec{\omega} \\ -\sigma_t(\mathbf{x})\int\limits_{4\pi} L_d(\mathbf{x},\vec{\omega})\,d\vec{\omega} + Q^0_{ri}(\mathbf{x}) + Q^0(\mathbf{x})\end{aligned} \tag{A.2}$$

First use the definitions of $\phi(\mathbf{x})$ and $\vec{E}(\mathbf{x})$ from Equations (2.5) and (2.6) and rearrange the first term on the right hand side

$$\begin{aligned}\vec{\nabla}\cdot\vec{E}(\mathbf{x}) = \sigma_s(\mathbf{x})\int\limits_{4\pi} L_d(\mathbf{x},\vec{\omega}')\int\limits_{4\pi} p(\vec{\omega},\vec{\omega}')\,d\vec{\omega}\,d\vec{\omega}' \\ -\sigma_t(\mathbf{x})\phi(\mathbf{x}) + Q^0_{ri}(\mathbf{x}) + Q^0(\mathbf{x})\end{aligned} \tag{A.3}$$

Next using the normalization of $p(\vec{\omega}, \vec{\omega}')$ and Equation (2.5)

$$\vec{\nabla} \cdot \vec{E}(\mathbf{x}) = \sigma_s(\mathbf{x})\phi(\mathbf{x}) - \sigma_t(\mathbf{x})\phi(\mathbf{x}) + Q^0_{ri}(\mathbf{x}) + Q^0(\mathbf{x}) \tag{A.4}$$

And finally the definition of $\sigma_t(\mathbf{x}) = \sigma_s(\mathbf{x}) + \sigma_a(\mathbf{x})$ gives Equation (2.13)

$$\vec{\nabla} \cdot \vec{E}(\mathbf{x}) = -\sigma_a(\mathbf{x})\phi(\mathbf{x}) + Q^0_{ri}(\mathbf{x}) + Q^0(\mathbf{x})$$

A.2 Derivation of Equation (2.14)

The derivation of Equation (2.14) is more complicated than Equation (2.13). It has two main parts: the substitution of the DA and equating the 1st moments.

A.2.1 Making the Diffusion Approximation

Direct substitution of Equation (2.7) into Equation (2.12) yields

$$\begin{aligned}
&\frac{1}{4\pi}\big(\vec{\omega} \cdot \vec{\nabla}\big)\phi(\mathbf{x}) + \frac{3}{4\pi}\big(\vec{\omega} \cdot \vec{\nabla}\big)\big(\vec{\omega} \cdot \vec{E}(\mathbf{x})\big) = \\
&\qquad \frac{\sigma_s(\mathbf{x})}{4\pi}\int_{4\pi} p(\vec{\omega}, \vec{\omega}')\phi(\mathbf{x})\, d\vec{\omega}' + \frac{3\sigma_s(\mathbf{x})}{4\pi}\int_{4\pi} p(\vec{\omega}, \vec{\omega}')\big(\vec{\omega}' \cdot \vec{E}(\mathbf{x})\big)\, d\vec{\omega}' \\
&\qquad\qquad - \frac{\sigma_t(\mathbf{x})}{4\pi}\phi(\mathbf{x}) - \frac{3\sigma_t(\mathbf{x})}{4\pi}\big(\vec{\omega} \cdot \vec{E}(\mathbf{x})\big) + Q_{ri}(\mathbf{x}, \vec{\omega}) + Q(\mathbf{x}, \vec{\omega})
\end{aligned} \tag{A.5}$$

Next, simplify the two integral terms on the right hand side. For the first term

$$\frac{\sigma_s(\mathbf{x})}{4\pi}\int_{4\pi} p(\vec{\omega}, \vec{\omega}')\phi(\mathbf{x})\, d\vec{\omega}' = \frac{\sigma_s(\mathbf{x})}{4\pi}\phi(\mathbf{x})\int_{4\pi} p(\vec{\omega}, \vec{\omega}')\, d\vec{\omega}' = \frac{\sigma_s(\mathbf{x})}{4\pi}\phi(\mathbf{x}) \tag{A.6}$$

since the phase function is normalized. Since the phase function depends only on the angle between the two directions (see Equation (2.2)), the second term can be simplified using the μ, the mean cosine of the phase function (see Equation (2.3)).

$$\begin{aligned}\frac{3\sigma_s(\mathbf{x})}{4\pi}\int\limits_{4\pi} p(\vec{\omega},\vec{\omega}')\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big)\,d\vec{\omega}' &= \frac{3\sigma_s(\mathbf{x})}{4\pi}\int\limits_{4\pi} p(\vec{\omega},\vec{\omega}')(\vec{\omega}\cdot\vec{\omega})\big(\vec{\omega}'\cdot\vec{E}(\mathbf{x})\big)\,d\vec{\omega}' \\ &= \frac{3\sigma_s(\mathbf{x})}{4\pi}\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big)\int\limits_{4\pi} p(\vec{\omega},\vec{\omega}')(\vec{\omega}\cdot\vec{\omega}')\,d\vec{\omega}' \\ &= \frac{3\mu\sigma_s(\mathbf{x})}{4\pi}\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big) \qquad \text{(A.7)}\end{aligned}$$

Substitute Equations (A.6) and (A.7) into Equation (A.5) and group the like terms on the right hand side

$$\begin{aligned}&\frac{1}{4\pi}\big(\vec{\omega}\cdot\vec{\nabla}\big)\phi(\mathbf{x}) + \frac{3}{4\pi}\big(\vec{\omega}\cdot\vec{\nabla}\big)\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big) = \\ &\qquad \frac{1}{4\pi}\big[\sigma_s(\mathbf{x}) - \sigma_t(\mathbf{x})\big]\,\phi(\mathbf{x}) + \frac{3}{4\pi}\big[\mu\sigma_s(\mathbf{x}) - \sigma_t(\mathbf{x})\big]\,\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big) \\ &\qquad\qquad + Q_{ri}(\mathbf{x},\vec{\omega}) + Q(\mathbf{x},\vec{\omega}) \qquad \text{(A.8)}\end{aligned}$$

Finally, use the definitions of $\sigma_t(\mathbf{x})$ and $\sigma_{tr}(\mathbf{x}) = (1-\mu)\sigma_s(\mathbf{x}) + \sigma_a(\mathbf{x})$ to simplify the scattering function coefficients.

$$\begin{aligned}&\frac{1}{4\pi}\big(\vec{\omega}\cdot\vec{\nabla}\big)\phi(\mathbf{x}) + \frac{3}{4\pi}\big(\vec{\omega}\cdot\vec{\nabla}\big)\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big) = \\ &\qquad - \frac{\sigma_a(\mathbf{x})}{4\pi}\phi(\mathbf{x}) - \frac{3\sigma_{tr}(\mathbf{x})}{4\pi}\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big) + Q_{ri}(\mathbf{x},\vec{\omega}) + Q(\mathbf{x},\vec{\omega}) \qquad \text{(A.9)}\end{aligned}$$

A.2.2 Computation of 1st moments

Next step is to relate the first moments of the terms in Equation (A.9). First, compute each moment

$$\frac{1}{4\pi}\int\limits_{4\pi}\vec{\omega}\big(\vec{\omega}\cdot\vec{\nabla}\big)\phi(\mathbf{x})\;d\vec{\omega}+\frac{3}{4\pi}\int\limits_{4\pi}\vec{\omega}\big(\vec{\omega}\cdot\vec{\nabla}\big)\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big)\;d\vec{\omega}=$$

$$-\frac{\sigma_a(\mathbf{x})}{4\pi}\int\limits_{4\pi}\vec{\omega}\phi(\mathbf{x})\;d\vec{\omega}-\frac{3\sigma_{tr}(\mathbf{x})}{4\pi}\int\limits_{4\pi}\vec{\omega}\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big)\;d\vec{\omega}+Q^1_{ri}(\mathbf{x})+Q^1(\mathbf{x}) \tag{A.10}$$

Each of the integral terms need simplification and an identity from [Ish78] is useful. Let $\vec{s}$ be any 3D vector, then

$$\int\limits_{4\pi}\vec{\omega}(\vec{\omega}\cdot\vec{s})\;d\vec{\omega}=\frac{4\pi}{3}\vec{s} \tag{A.11}$$

Next, simplify the terms individually from left to right. The first term uses the identity.

$$\frac{1}{4\pi}\int\limits_{4\pi}\vec{\omega}\big(\vec{\omega}\cdot\vec{\nabla}\big)\phi(\mathbf{x})\;d\vec{\omega}=\frac{1}{3}\vec{\nabla}\phi(\mathbf{x}) \tag{A.12}$$

The second term is identically zero.

$$\frac{3}{4\pi}\int\limits_{4\pi}\vec{\omega}\big(\vec{\omega}\cdot\vec{\nabla}\big)\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big)\;d\vec{\omega}=\frac{3}{4\pi}\int\limits_{4\pi}\vec{\omega}\big(\vec{\omega}\cdot\vec{\omega}\big)\big(\vec{\nabla}\cdot\vec{E}(\mathbf{x})\big)\;d\vec{\omega}$$

$$=\frac{3}{4\pi}\big(\vec{\nabla}\cdot\vec{E}(\mathbf{x})\big)\int\limits_{4\pi}\vec{\omega}\;d\vec{\omega}\equiv 0 \tag{A.13}$$

And, for similar reasons, so is the first term on the right hand side.

$$\frac{\sigma_a(\mathbf{x})}{4\pi}\int\limits_{4\pi}\vec{\omega}\phi(\mathbf{x})\;d\vec{\omega}=\frac{\sigma_a(\mathbf{x})}{4\pi}\phi(\mathbf{x})\int\limits_{4\pi}\vec{\omega}\;d\vec{\omega}\equiv 0 \tag{A.14}$$

Lastly, the second term on the right hand side again uses Equation (A.11).

$$\frac{3\sigma_{tr}(\mathbf{x})}{4\pi}\int\limits_{4\pi}\vec{\omega}\big(\vec{\omega}\cdot\vec{E}(\mathbf{x})\big)\;d\vec{\omega}=\sigma_{tr}(\mathbf{x})\vec{E}(\mathbf{x}) \tag{A.15}$$

The final step substitutes Equations (A.12)-(A.15) into Equation (A.10) and multiplies by 3 to derive Equation (2.14).

$$\vec{\nabla}\phi(\mathbf{x}) = -3\sigma_{tr}(\mathbf{x})\vec{E}(\mathbf{x}) + 3Q^1_{ri}(\mathbf{x}) + 3Q^1(\mathbf{x})$$

APPENDIX B

ADDENDA TO THE DIFFUSIVE SOURCE BOUNDARY CONDITION DERIVATION

Deriving the DSBC requires substituting Equations (5.2) and (5.3) into Equation (5.1) and then substituting the diffusion approximation (Equation (2.7)) for $L_d(\mathbf{x}, \vec{\omega})$. The result of the first substitution is

$$\int\limits_{(\vec{n}\cdot\vec{\omega})<0} L_d(\mathbf{x}, \vec{\omega})(-\vec{n} \cdot \vec{\omega})\, d\vec{\omega} = F_{dr}(\eta) \int\limits_{(\vec{n}\cdot\vec{\omega})<0} L_d(\mathbf{x}, -\vec{\omega})(-\vec{n} \cdot \vec{\omega})\, d\vec{\omega} + \Gamma_s(\mathbf{x}) \tag{B.1}$$

Using Equation (2.17), the DA can be applied and the integrals simplified.

$$\frac{\phi(\mathbf{x})}{4} + \frac{\kappa_d(\mathbf{x})(\vec{n} \cdot \vec{\nabla})\phi(\mathbf{x})}{2} = F_{dr}(\eta) \left[\frac{\phi(\mathbf{x})}{4} - \frac{\kappa_d(\mathbf{x})(\vec{n} \cdot \vec{\nabla})\phi(\mathbf{x})}{2}\right] + \Gamma_s(\mathbf{x}) \tag{B.2}$$

Multiply by 4.

$$\phi(\mathbf{x}) + 2\kappa_d(\mathbf{x})\big(\vec{n} \cdot \vec{\nabla}\big)\phi(\mathbf{x}) = F_{dr}(\eta) \Big[\phi(\mathbf{x}) - 2\kappa_d(\mathbf{x})\big(\vec{n} \cdot \vec{\nabla}\big)\phi(\mathbf{x})\Big] + 4\Gamma_s(\mathbf{x}) \tag{B.3}$$

Group like terms.

$$\big(1 - F_{dr}(\eta)\big)\, \phi(\mathbf{x}) + 2\kappa_d(\mathbf{x})\, \big(1 + F_{dr}(\eta)\big)\, \big(\vec{n} \cdot \vec{\nabla}\big)\phi(\mathbf{x}) = 4\Gamma_s(\mathbf{x}) \tag{B.4}$$

Divide by $F_{dt}(\eta) = 1 - F_{dr}(\eta)$.

$$\phi(\mathbf{x}) + 2\kappa_d(\mathbf{x}) \left[\frac{1 + F_{dr}(\eta)}{1 - F_{dr}(\eta)}\right] \big(\vec{n} \cdot \vec{\nabla}\big)\phi(\mathbf{x}) = \frac{4}{F_{dt}(\eta)}\Gamma_s(\mathbf{x}) \tag{B.5}$$

Substitute Equation (2.22) introducing $A(\eta)$ and completing the derivation.

$$\phi(\mathbf{x}) + 2\kappa_d(\mathbf{x})A(\eta)\big(\vec{n} \cdot \vec{\nabla}\big)\phi(\mathbf{x}) = \frac{4}{F_{dt}(\eta)}\Gamma_s(\mathbf{x}) \tag{B.6}$$

APPENDIX C

DERIVATION OF THE DISCONTINUOUS GALERKIN MATRIX EQUATION

This derivation follows the five step outline described in Section 6.1. The discontinuous Galerkin (DG) penalty method used here ensures a unique solution exists [ABCM02]. However, this proof is well beyond the scope of this thesis. Summarizing the one in [ABCM02], this discussion only describes the minimal detail required to derive the DG matrix equation (Equation (6.1)). Refer to the original work for a more robust mathematical discussion.

C.1 Step 1: Diffusion System

In this first step, the dependence on the vector irradiance $\vec{E}(\mathbf{x})$ must be made explicit. Recalling Equation (2.6), $\vec{E}(\mathbf{x})$ can be defined such that

$$\vec{E}(\mathbf{x}) = \kappa_d(\mathbf{x})\vec{\nabla}\phi(\mathbf{x}) \tag{C.1}$$

With this definition, the DE can be rewritten as a coupled system of two PDEs.

Diffusion System.

$$-\vec{\nabla} \cdot \vec{E} + \sigma_a(\mathbf{x})\phi = Q^0 \tag{C.2}$$

$$\vec{E} = \kappa_d\vec{\nabla}\phi \tag{C.3}$$

From these equations to the end of this appendix, all explicit references to the independent spatial variable $\mathbf{x}$ and, for the Fresnel terms, the relative index of refraction η have been dropped to make the equations more concise.

C.2 Step 2: Weak System

Converting the diffusion system into a weak system requires two additional definitions: a domain subdivision and a new function space. First, because the final DG method allows only a limited type of discontinuity—continuous *within* mesh elements but discontinuous *between* mesh elements—the derivation of the DG method is more dependent on the ultimate goal of dividing the solution into a finite basis. Thus here, at the beginning of the derivation, the domain Ω must be initially subdivided into a finite number N_e of elements Ω_i. For this derivation, this subdivision must be non-overlapping, complete (i.e. $\Omega = \bigcup_{i=0}^{N_e} \Omega_i$) and the boundaries between elements must be planar [ABCM02]. Within these constraints, these subdivisions can be arbitrary but a useful specific model is the tetrahedral mesh described in Section 5.4.1. Second, additional function spaces must be defined. In Section 5.3.1, the continuous FE method searched a continuous function space $\mathbb{H}$ for the solution fluence ϕ. In the DG FE method, the fluence can lie in a larger space $\mathbb{H}_{\phi}^{\mathrm{dg}}$ that contains additional discontinuous functions. However, the weak system requires a second vector function space $\mathbb{H}_{\vec{E}}^{\mathrm{dg}}$ that contains all potential solutions for $\vec{E}$. These function spaces must be chosen such that gradient of any function in $\mathbb{H}_{\phi}^{\mathrm{dg}}$ also lies in $\mathbb{H}_{\vec{E}}^{\mathrm{dg}}$.

Next, the derivation of the weak system proceeds by the same process used in Section 5.3.1 but applied separately to both Equations (C.2) and (C.3). However additionally, because of the inter-element discontinuities, it must also be applied separately inside each element of the domain Ω_i. Let $\theta \in \mathbb{H}_{\phi}^{\mathrm{dg}}$ and $\vec{e} \in \mathbb{H}_{\vec{E}}^{\mathrm{dg}}$ be arbitrary functions in each function space. Multiply Equations (C.2) and (C.3) by

θ and $\vec{e}$ respectively and integrate over each Ω_i

$$-\int_{\Omega_i} (\vec{\nabla} \cdot \vec{E})\theta \, d\mathbf{x} + \int_{\Omega_i} \sigma_a \phi \theta \, d\mathbf{x} = \int_{\Omega_i} Q^0 \theta \, d\mathbf{x} \tag{C.4}$$

$$\int_{\Omega_i} \vec{E} \cdot \vec{e} \, d\mathbf{x} = \int_{\Omega_i} \vec{\nabla}\phi \cdot \big(\kappa_d \vec{e}\big) \, d\mathbf{x} \tag{C.5}$$

Now use the divergence theorem (Equation (5.12)) to transform the first term in Equation (C.4) and the last term in Equation (C.5). Then rearranging terms yields the DG weak system.

Weak System. *Find ϕ and $\vec{E}$ such that for all $\Omega_i \in \Omega$, $\theta \in \mathbb{H}^{\text{dg}}_{\phi}$ and $\vec{e} \in \mathbb{H}^{\text{dg}}_{\vec{E}}$*

$$\int_{\Omega_i} \vec{E} \cdot \vec{\nabla}\theta \, d\mathbf{x} = \int_{\partial\Omega_i} \big(\vec{E} \cdot \vec{n}\big)\theta \, d\mathbf{x} - \int_{\Omega_i} \sigma_a \phi \theta \, d\mathbf{x} + \int_{\Omega_i} Q^0 \theta \, d\mathbf{x} \tag{C.6}$$

$$\int_{\Omega_i} \vec{E} \cdot \vec{e} \, d\mathbf{x} = -\int_{\Omega_i} \phi \vec{\nabla} \cdot \big(\kappa_d \vec{e}\big) \, d\mathbf{x} + \int_{\partial\Omega_i} \kappa_d \phi \big(\vec{e} \cdot \vec{n}\big) \, d\mathbf{x} \tag{C.7}$$

In the weak system, the jump terms to be penalized are evident. The two element boundary integral terms, one in each equation, respectively require the values of $\vec{E}$ and $\kappa_d \phi$ on the boundaries of elements. However, in the DG formulation, $\vec{E}$ and ϕ may be discontinuous along these boundaries and, therefore, these terms may not be well defined. To correct this, numerical estimates of these boundary values must be defined—call these estimates $\vec{E}^*$ and ϕ^*_κ—and, as part of these definitions, jump penalties are imposed.

C.3 Step 3: Primal Form

In this section, the dependence on $\vec{E}$ is removed to produce a new weak form dependent only on ϕ. This new weak form is called the *primal form.* This step is the longest in the derivation and will be presented in several subsections. Each intermediate step either introduces some simplifying notation or performs some simplifying algebra. To begin, sum Equations (C.6) and (C.7) over all elements in the domain

$$\sum_{\Omega_i\in\Omega}\left[\int_{\Omega_i}\vec{E}\cdot\vec{\nabla}\theta\ d\mathbf{x}\right]=-\int_{\Omega}\sigma_a\phi\theta\ d\mathbf{x}+\int_{\Omega}Q^0\theta\ d\mathbf{x}+\sum_{\Omega_i\in\Omega}\left[\int_{\partial\Omega_i}\left(\vec{E}^*\cdot\vec{n}\right)\theta\ d\mathbf{x}\right] \tag{C.8}$$

$$\int_{\Omega}\vec{E}\cdot\vec{e}\ d\mathbf{x}=-\sum_{\Omega_i\in\Omega}\left[\int_{\Omega_i}\phi\vec{\nabla}\cdot\left(\kappa_d\vec{e}\right)\ d\mathbf{x}\right]+\sum_{\Omega_i\in\Omega}\left[\int_{\partial\Omega_i}\phi^*_\kappa\left(\vec{e}\cdot\vec{n}\right)\ d\mathbf{x}\right] \tag{C.9}$$

C.3.1 Piecewise Gradient Operator

Since $\vec{E}$ and ϕ may be discontinuous across faces, their gradient and divergence are no longer well defined throughout the entire domain. Thus terms like the first one in Equation (C.8) cannot be converted into a single integral over the whole domain. However, these expressions can be simplified with some new notation: the piecewise gradient $\vec{\nabla}_{\mathsf{p}}$ operator. The $\vec{\nabla}_{\mathsf{p}}$ operator is defined such that for any functions α and $\vec{a}$

$$\int_{\Omega}\vec{\nabla}_{\mathsf{p}}\,\alpha\ d\mathbf{x}=\sum_{\Omega_i\in\Omega}\left[\int_{\Omega_i}\vec{\nabla}\alpha\ d\mathbf{x}\right] \tag{C.10}$$

$$\int_{\Omega}\vec{\nabla}_{\mathsf{p}}\cdot\vec{a}\ d\mathbf{x}=\sum_{\Omega_i\in\Omega}\left[\int_{\Omega_i}\vec{\nabla}\cdot\vec{a}\ d\mathbf{x}\right] \tag{C.11}$$

Using the piecewise gradient Equations (C.8) and (C.9) become

$$\int_{\Omega}\vec{E}\cdot\vec{\nabla}_{\mathsf{p}}\,\theta\ d\mathbf{x}=-\int_{\Omega}\sigma_a\phi\theta\ d\mathbf{x}+\int_{\Omega}Q^0\theta\ d\mathbf{x}+\sum_{\Omega_i\in\Omega}\left[\int_{\partial\Omega_i}\left(\vec{E}^*\cdot\vec{n}\right)\theta\ d\mathbf{x}\right] \tag{C.12}$$

$$\int_{\Omega}\vec{E}\cdot\vec{e}\ d\mathbf{x}=-\int_{\Omega}\phi\vec{\nabla}_{\mathsf{p}}\cdot\left(\kappa_d\vec{e}\right)\ d\mathbf{x}+\sum_{\Omega_i\in\Omega}\left[\int_{\partial\Omega_i}\phi^*_\kappa\left(\vec{e}\cdot\vec{n}\right)\ d\mathbf{x}\right] \tag{C.13}$$

C.3.2 Face Integrals with Jump and Average Operators

Next it is convenient to convert the sums of boundary integrals into integrals over the union of all element faces. This allows the diffusive source boundary condition (Equation (5.5)) to be imposed and simplifies the specification of $\vec{E}^*$ and ϕ^*_κ. Performing the conversion requires introducing the average $\{\cdot\}$ and the jump $[\![\cdot]\!]$ operators for an interior face. If $f_{ij} = \Omega_i \cup \Omega_j$ is the face between Ω_i and Ω_j, then let $\mathcal{F}_\mathcal{I} = \bigcup f_{ij}$ and $\mathcal{F} = \partial\Omega \cup \mathcal{F}_\mathcal{I}$. Then, for f_{ij}, the jump $[\![\cdot]\!]$ and average $\{\cdot\}$ operators are defined for scalar α and vector $\vec{a}$ values as

$$[\![\alpha]\!] = \alpha_i \vec{n}_i + \alpha_j \vec{n}_j \qquad\qquad \{\alpha\} = \tfrac{1}{2}\big(\alpha_i + \alpha_j\big) \tag{C.14}$$

$$[\![\vec{a}]\!] = \vec{a}_i \cdot \vec{n}_i + \vec{a}_j \cdot \vec{n}_j \qquad\qquad \{\vec{a}\} = \tfrac{1}{2}\big(\vec{a}_i + \vec{a}_j\big) \tag{C.15}$$

Since these definitions require each face to have a single normal, their use here requires that the domain elements Ω_i have piecewise planar boundaries. With these definitions, the sums over boundary integrals can be rewritten using an identity (derived in Section C.6): for any $\vec{a}$ and α,

$$\sum_{\Omega_i \in \Omega} \left[\int\limits_{\partial\Omega_i} \alpha \vec{a} \cdot \vec{n}\, d\mathbf{x} \right] = \int\limits_{\partial\Omega} \alpha \vec{a} \cdot \vec{n}\, d\mathbf{x} + \int\limits_{\mathcal{F}_\mathcal{I}} [\![\vec{a}]\!]\{\alpha\} + \{\vec{a}\} \cdot [\![\alpha]\!]\, d\mathbf{x} \tag{C.16}$$

Using Equation (C.16), Equations (C.12) and (C.13) become

$$\int\limits_{\Omega} \vec{E} \cdot \vec{\nabla}_{\mathsf{p}}\, \theta\, d\mathbf{x} = -\int\limits_{\Omega} \sigma_a \phi \theta\, d\mathbf{x} + \int\limits_{\Omega} Q^0 \theta\, d\mathbf{x}$$
$$+ \int\limits_{\partial\Omega} \big(\vec{E}^* \cdot \vec{n}\big)\theta\, d\mathbf{x} + \int\limits_{\mathcal{F}_\mathcal{I}} [\![\vec{E}^*]\!]\{\theta\} + \{\vec{E}^*\} \cdot [\![\theta]\!]\, d\mathbf{x} \tag{C.17}$$

$$\int\limits_{\Omega} \vec{E} \cdot \vec{e}\, d\mathbf{x} = -\int\limits_{\Omega} \phi \vec{\nabla}_{\mathsf{p}} \cdot \big(\kappa_d \vec{e}\big)\, d\mathbf{x}$$
$$+ \int\limits_{\partial\Omega} \phi^*_\kappa \big(\vec{e} \cdot \vec{n}\big)\, d\mathbf{x} + \int\limits_{\mathcal{F}_\mathcal{I}} [\![\vec{e}]\!]\{\phi^*_\kappa\} + \{\vec{e}\} \cdot [\![\phi^*_\kappa]\!]\, d\mathbf{x} \tag{C.18}$$

C.3.3 Piecewise Divergence Theorem

The next step simplifies the first term on the right hand side of Equation (C.18). This requires another identity (also derived in Section C.6 by applying the divergence theorem locally to each element in the domain): for any $\vec{a}$ and α,

$$-\int_{\Omega} \alpha \vec{\nabla}_{\mathsf{p}} \cdot \vec{a} \, d\mathbf{x} = \int_{\Omega} \vec{\nabla}_{\mathsf{p}} \, \alpha \cdot \vec{a} \, d\mathbf{x} - \int_{\partial\Omega} \alpha \vec{a} \cdot \vec{n} \, d\mathbf{x} - \int_{\mathcal{F}_{\mathcal{I}}} [\![\vec{a}]\!] \{\alpha\} + \{\vec{a}\} \cdot [\![\alpha]\!] \, d\mathbf{x} \quad \text{(C.19)}$$

Applying Equation (C.19) to Equation (C.18) yields

$$\int_{\Omega} \vec{E} \cdot \vec{e} \, d\mathbf{x} = \int_{\Omega} \vec{\nabla}_{\mathsf{p}} \, \phi \cdot \left(\kappa_d \vec{e}\right) \, d\mathbf{x} + \int_{\partial\Omega} \left(\phi^*_\kappa - \kappa_d \phi\right)\left(\vec{e} \cdot \vec{n}\right) \, d\mathbf{x}$$
$$+ \int_{\mathcal{F}_{\mathcal{I}}} [\![\vec{e}\,]\!] \{\phi^*_\kappa - \kappa_d \phi\} + \{\vec{e}\,\} \cdot [\![\phi^*_\kappa - \kappa_d \phi]\!] \, d\mathbf{x} \quad \text{(C.20)}$$

C.3.4 Final Substitution

Next, on the boundary of the domain, ϕ^*_κ is well valued and no special definition or penalty need be applied. Thus, on the boundary, let $\phi^*_\kappa = \kappa_d \phi$ and then Equation (C.20) becomes

$$\int_{\Omega} \vec{E} \cdot \vec{e} \, d\mathbf{x} = \int_{\Omega} \vec{\nabla}_{\mathsf{p}} \, \phi \cdot \left(\kappa_d \vec{e}\right) \, d\mathbf{x} + \int_{\mathcal{F}_{\mathcal{I}}} [\![\vec{e}\,]\!] \{\phi^*_\kappa - \kappa_d \phi\} + \{\vec{e}\,\} \cdot [\![\phi^*_\kappa - \kappa_d \phi]\!] \, d\mathbf{x} \quad \text{(C.21)}$$

Now, recalling the original definition of the weak system (see Equations (C.4) and (C.5)), note that Equations (C.17) and (C.21) must hold for all functions $\theta \in \mathbb{H}^{\mathrm{dg}}_{\phi}$ and $\vec{e} \in \mathbb{H}^{\mathrm{dg}}_{\vec{E}}$. Moreover, for the spaces chosen for this derivation (Section C.2), it must be that if $\theta \in \mathbb{H}^{\mathrm{dg}}_{\phi}$ then $\vec{\nabla}_{\mathsf{p}} \, \theta \in \mathbb{H}^{\mathrm{dg}}_{\vec{E}}$. Thus Equation (C.21) must specifically hold when $\vec{e} = \vec{\nabla}_{\mathsf{p}} \, \theta$.

$$\int_{\Omega} \vec{E} \cdot \vec{\nabla}_{\mathsf{p}} \theta \, d\mathbf{x} = \int_{\Omega} \kappa_d \vec{\nabla}_{\mathsf{p}} \phi \cdot \vec{\nabla}_{\mathsf{p}} \theta \, d\mathbf{x} + \int_{\mathcal{F}_{\mathcal{I}}} [\![\vec{\nabla}_{\mathsf{p}} \theta]\!] \{\phi^*_\kappa - \kappa_d \phi\} + \{\vec{\nabla}_{\mathsf{p}} \theta\} \cdot [\![\phi^*_\kappa - \kappa_d \phi]\!] \, d\mathbf{x} \quad \text{(C.22)}$$

Substituting Equation (C.22) into Equation (C.17) and rearranging terms yields the primal form.

Primal Form. *Find ϕ such that for all $\theta \in \mathbb{H}^{\mathrm{dg}}_{\phi}$*

$$\begin{aligned}\int\limits_{\Omega} \kappa_d \vec{\nabla}_{\mathsf{p}}\, \phi \cdot \vec{\nabla}_{\mathsf{p}}\, \theta \; d\mathbf{x} + \int\limits_{\Omega} \sigma_a \phi\theta \; d\mathbf{x} \\ - \int\limits_{\partial\Omega} (\vec{E}^{*} \cdot \vec{n})\theta \; d\mathbf{x} + \int\limits_{\mathcal{F}_{\mathcal{I}}} [\![\vec{\nabla}_{\mathsf{p}}\, \theta]\!]\{\phi^{*}_{\kappa} - \kappa_d\phi\} + \{\vec{\nabla}_{\mathsf{p}}\, \theta\} \cdot [\![\phi^{*}_{\kappa} - \kappa_d\phi]\!] \; d\mathbf{x} \\ - \int\limits_{\mathcal{F}_{\mathcal{I}}} [\![\vec{E}^{*}\,]\!]\{\theta\} + \{\vec{E}^{*}\,\} \cdot [\![\theta]\!] \; d\mathbf{x} = \int\limits_{\Omega} Q^0\theta \; d\mathbf{x} \quad \text{(C.23)}\end{aligned}$$

C.4 Step 4: Interior Penalty Primal Form

This step of the derivation chooses a particular definition and penalty for the two undefined boundary terms, $\vec{E}^{*}$ and ϕ^{*}_{κ}. This choice results in a specific primal form, the *interior penalty* (IP) primal form [ABCM02]. However, before the IP primal form is introduced, the basic primal form can be further simplified. All valid choices for the element boundary terms must be both consistent and conservative. Consistency is ensured if, when the solution fluence is smooth[1], then $[\![\phi^{*}_{\kappa}\,]\!] = [\![\vec{E}^{*}\,]\!] = 0$, $\{\phi^{*}_{\kappa}\,\} = \phi$ and $\{\vec{E}^{*}\,\} = \vec{E}$. Second, the choice is conservative if the definition is symmetric, i.e. the definition on the face between Ω_i and Ω_j equals the definition on the face defined by the elements in reverse order. This can only be true if $[\![\phi^{*}_{\kappa}\,]\!] = [\![\vec{E}^{*}\,]\!] = 0$. Though this conservative definition seems to repeat part of the definition of consistency, note that conservativity it must be true for all functions, not just for smooth functions, as required by consistency. Thus, using

[1]i.e. It is continuous and has a complete set of continuous derivatives.

the conservative property, the basic primal form can be simplified to

$$\int_{\Omega} \kappa_d \vec{\nabla}_{\mathsf{p}}\, \phi \cdot \vec{\nabla}_{\mathsf{p}}\, \theta \; d\mathbf{x} + \int_{\Omega} \sigma_a \phi\theta \; d\mathbf{x} - \int_{\partial\Omega} \big(\vec{E}^* \cdot \vec{n}\big)\theta \; d\mathbf{x}$$

$$+ \int_{\mathcal{F}_{\mathcal{I}}} [\![\vec{\nabla}_{\mathsf{p}}\, \theta]\!]\{\phi^*_\kappa - \kappa_d\phi\} - \{\vec{\nabla}_{\mathsf{p}}\, \theta\} \cdot [\![\kappa_d\phi]\!] - \{\vec{E}^*\} \cdot [\![\theta]\!] \; d\mathbf{x} = \int_{\Omega} Q^0\theta \; d\mathbf{x} \quad \text{(C.24)}$$

Next the IP primal form is created by defining ϕ^*_κ and $\vec{E}^*$ as

$$\phi^*_\kappa = \{\kappa_d\phi\} \quad \text{(C.25)}$$

$$\vec{E}^* = \{\kappa_d \vec{\nabla}_{\mathsf{p}}\, \phi\} - \eta[\![\kappa_d\phi]\!] \quad \text{(C.26)}$$

The coefficient η is a user specified constant that determines the relative effect of the discontinuous penalty. However, the solution to the IP primal form is well defined for any positive value of η [ABCM02]. The effect of the choice of η is described in Section 6.2.4. Substituting Equations (C.25) and (C.26) into Equation (C.24) yields

$$\int_{\Omega} \kappa_d \vec{\nabla}_{\mathsf{p}}\, \phi \cdot \vec{\nabla}_{\mathsf{p}}\, \theta \; d\mathbf{x} + \int_{\Omega} \sigma_a \phi\theta \; d\mathbf{x} - \int_{\partial\Omega} \big(\kappa_d \vec{\nabla}_{\mathsf{p}}\, \phi \cdot \vec{n}\big)\theta \; d\mathbf{x}$$

$$- \int_{\mathcal{F}_{\mathcal{I}}} \{\vec{\nabla}_{\mathsf{p}}\, \theta\} \cdot [\![\kappa_d\phi]\!] + \{\kappa_d \vec{\nabla}_{\mathsf{p}}\, \phi\} \cdot [\![\theta]\!] - \eta[\![\kappa_d\phi]\!] \cdot [\![\theta]\!] \; d\mathbf{x} = \int_{\Omega} Q^0\theta \; d\mathbf{x} \quad \text{(C.27)}$$

Finally use the diffusive source boundary condition (Equation (5.5)) to eliminiate the boundary term in Equation (C.27) and derive the IP primal form.

Interior Penalty Primal Form. *Find ϕ such that for all $\theta \in \mathbb{H}^{\mathrm{dg}}_\phi$*

$$\int_{\Omega} \kappa_d \vec{\nabla}_{\mathsf{p}}\, \phi \cdot \vec{\nabla}_{\mathsf{p}}\, \theta \; d\mathbf{x} + \int_{\Omega} \sigma_a \phi\theta \; d\mathbf{x} + \frac{1}{2A} \int_{\partial\Omega} \phi\theta \; d\mathbf{x}$$

$$- \int_{\mathcal{F}_{\mathcal{I}}} \{\vec{\nabla}_{\mathsf{p}}\, \theta\} \cdot [\![\kappa_d\phi]\!] + \{\kappa_d \vec{\nabla}_{\mathsf{p}}\, \phi\} \cdot [\![\theta]\!] - \eta[\![\kappa_d\phi]\!] \cdot [\![\theta]\!] \; d\mathbf{x}$$

$$= \int_{\Omega} Q^0\theta \; d\mathbf{x} + \frac{2}{AF_{dt}} \int_{\partial\Omega} \Gamma_s\theta \; d\mathbf{x} \quad \text{(C.28)}$$

C.5 Step 5: Discontinuous Galerkin Matrix Equation

The final step in this derivation creates the final matrix equation using the IP primal form. This process is identical to the derivation in Section 5.3.2 except that Equation (5.14) has been replaced with Equation (C.28). First $\mathbb{H}_\phi^{\mathrm{dg}}$ is assumed to have a finite basis.

$$\mathcal{B}(\mathbf{x}) = \big\{\beta_0(\mathbf{x}), \beta_1(\mathbf{x}), \ldots, \beta_{n-1}(\mathbf{x})\big\}$$

Second, ϕ is assumed to lie in that basis, i.e. $\phi = \sum_{i=0}^{n-1} a_i \beta_i$. The result is the basic matrix form: *find* $\{a_i\}$ *such that for all* β_j

$$\sum_{\beta_i \in \mathbb{H}_\phi^{\mathrm{dg}}} a_i \Bigg[\int\limits_\Omega \kappa_d \vec{\nabla}_{\mathsf{p}}\, \beta_i \cdot \vec{\nabla}_{\mathsf{p}}\, \beta_j \; d\mathbf{x} + \int\limits_\Omega \sigma_a \beta_i \beta_j \; d\mathbf{x}$$

$$+ \frac{1}{2A} \int\limits_{\partial\Omega} \beta_i \beta_j \; d\mathbf{x} - \int\limits_{\mathcal{F}_\mathcal{I}} \{\vec{\nabla}_{\mathsf{p}}\, \beta_j\} \cdot [\![\kappa_d \beta_i]\!] + \{\kappa_d \vec{\nabla}_{\mathsf{p}}\, \beta_i\} \cdot [\![\beta_j]\!] - \eta [\![\kappa_d \beta_i]\!] \cdot [\![\beta_j]\!] \; d\mathbf{x} \Bigg]$$

$$= \int\limits_\Omega Q^0 \beta_j \; d\mathbf{x} + \frac{2}{AF_{dt}} \int\limits_{\partial\Omega} \Gamma_s \beta_j \; d\mathbf{x} \quad \text{(C.29)}$$

Next, since the solution can be discontinuous, it may be convenient to represent the material coefficients κ_d and σ_a discontinuously. In this case, the two jump terms containing κ_d in Equation (C.29) are still not well defined. Using the harmonic mean κ_d^* of the local values in each element, κ_d^i and κ_d^j, as a element boundary approximation, solves this problem.

$$\kappa_d^* = \frac{2\kappa_d^i \kappa_d^k}{\kappa_d^i + \kappa_d^k} \quad \text{(C.30)}$$

Finally, since by construction (see Section 6.1), the basis functions have support only with a single element, the jump and average of the basis functions and their gradients have a simple form (see Equations (C.14) and (C.15)).

$$[\![\beta_i]\!] = \beta_i \vec{n}_i \quad \text{(C.31)}$$

$$\{\vec{\nabla}_{\mathsf{p}}\, \beta_i\} = \tfrac{1}{2} \vec{\nabla}_{\mathsf{p}}\, \beta_i \quad \text{(C.32)}$$

Using Equations (C.30), (C.31) and (C.32), the face term in Equation (C.29) can be simplified

$$\int_{\mathcal{F}_{\mathcal{I}}} \tfrac{1}{2}\kappa_d^*\big(\vec{\nabla}_{\mathsf{p}}\,\beta_j \cdot \beta_i\vec{n}_i + \vec{\nabla}_{\mathsf{p}}\,\beta_i \cdot \beta_j\vec{n}_j\big) - \eta\kappa_d^*\beta_i\vec{n}_i \cdot \beta_j\vec{n}_j \; d\mathbf{x} \tag{C.33}$$

Here $\vec{n}_i$ and $\vec{n}_j$ are the outwards pointing normals of the faces corresponding from the elements that contain β_i and β_j respectively.[2] Substituting Equation (C.33) into Equation (C.29) and rearranging terms yields the final discontinuous Galerkin matrix equation (reprinted here from Equation (6.1)).

Discontinuous Galerkin Matrix Equation.

$$(\mathsf{D} + \mathsf{M} + \mathsf{S} + \mathsf{E})\vec{\mathsf{a}} = \vec{\mathsf{q}} + \vec{\mathsf{g}}$$

where

$$\mathsf{D}_{ij} = \int_{\Omega} \kappa_d \vec{\nabla}_{\mathsf{p}}\,\beta_i \cdot \vec{\nabla}_{\mathsf{p}}\,\beta_j \; d\mathbf{x}; \qquad \mathsf{M}_{ij} = \int_{\Omega} \sigma_a \beta_i \beta_j \; d\mathbf{x}; \qquad \mathsf{S}_{ij} = \frac{1}{2A}\int_{\partial\Omega} \beta_i\beta_j \; d\mathbf{x};$$

$$\mathsf{E}_{ij} = -\int_{\mathcal{F}_{\mathcal{I}}} \tfrac{1}{2}\kappa_d^*\big(\vec{\nabla}_{\mathsf{p}}\,\beta_j \cdot \beta_i\vec{n}_i + \vec{\nabla}_{\mathsf{p}}\,\beta_i \cdot \beta_j\vec{n}_j - \eta\beta_i\vec{n}_i \cdot \beta_j\vec{n}_j\big) \; d\mathbf{x};$$

$$\vec{\mathsf{q}}_i = \int_{\Omega} Q^0 \beta_i \; d\mathbf{x}; \qquad \vec{\mathsf{g}}_i = \frac{2}{AF_{dt}}\int_{\partial\Omega} \Gamma_s \beta_i \; d\mathbf{x}$$

[2]It is not true that $\vec{n}_i = -\vec{n}_j$. Sometimes β_i and β_j lie in the same element and $\vec{n}_i = \vec{n}_j$

C.6 Derivation of Identities

Derivation of Equation (C.16).

$$\sum_{\Omega_i \in \Omega} \left[\int_{\partial \Omega_i} \alpha \vec{a} \cdot \vec{n} \; d\mathbf{x} \right] = \int_{\partial \Omega} \alpha \vec{a} \cdot \vec{n} \; d\mathbf{x} + \sum_{f_{ij} \in \mathcal{F}_{\mathcal{I}}} \left[\int_{f_{ij}} \alpha_i \vec{a}_i \cdot \vec{n}_i + \alpha_j \vec{a}_j \cdot \vec{n}_j \; d\mathbf{x} \right]$$

$$\begin{aligned}
\alpha_i \vec{a}_i \cdot \vec{n}_i + \alpha_j \vec{a}_j \cdot \vec{n}_j &= \tfrac{1}{2}\alpha_i \vec{a}_i \cdot \vec{n}_i + \tfrac{1}{2}\alpha_i \vec{a}_i \cdot \vec{n}_i + \tfrac{1}{2}\alpha_j \vec{a}_j \cdot \vec{n}_j + \tfrac{1}{2}\alpha_j \vec{a}_j \cdot \vec{n}_j + \\
&\quad \tfrac{1}{2}\alpha_i \vec{a}_j \cdot \vec{n}_i - \tfrac{1}{2}\alpha_i \vec{a}_j \cdot \vec{n}_i + \tfrac{1}{2}\alpha_j \vec{a}_i \cdot \vec{n}_i - \tfrac{1}{2}\alpha_j \vec{a}_i \cdot \vec{n}_i \\
&= \tfrac{1}{2}\alpha_i \vec{a}_i \cdot \vec{n}_i + \tfrac{1}{2}\alpha_i \vec{a}_i \cdot \vec{n}_i + \tfrac{1}{2}\alpha_j \vec{a}_j \cdot \vec{n}_j + \tfrac{1}{2}\alpha_j \vec{a}_j \cdot \vec{n}_j + \\
&\quad \tfrac{1}{2}\alpha_i \vec{a}_j \cdot \vec{n}_i + \tfrac{1}{2}\alpha_i \vec{a}_j \cdot \vec{n}_j + \tfrac{1}{2}\alpha_j \vec{a}_i \cdot \vec{n}_i + \tfrac{1}{2}\alpha_j \vec{a}_i \cdot \vec{n}_j \\
&= \tfrac{1}{2}\big(\alpha_i \vec{a}_i \cdot \vec{n}_i + \alpha_i \vec{a}_j \cdot \vec{n}_i + \alpha_j \vec{a}_i \cdot \vec{n}_j + \alpha_j \vec{a}_j \cdot \vec{n}_j\big) + \\
&\quad \tfrac{1}{2}\big(\alpha_i \vec{a}_i \cdot \vec{n}_i + \alpha_i \vec{a}_j \cdot \vec{n}_j + \alpha_j \vec{a}_i \cdot \vec{n}_i + \alpha_j \vec{a}_j \cdot \vec{n}_j\big) \\
&= \tfrac{1}{2}\big(\vec{a}_i \cdot \vec{n}_i + \vec{a}_j \cdot \vec{n}_j\big)\big(\alpha_i + \alpha_i\big) + \\
&\quad \tfrac{1}{2}\big(\vec{a}_i + \vec{a}_i\big)\big(\alpha_i \vec{n}_i + \alpha_j \vec{n}_j\big) \\
&= [\![\vec{a}]\!]\{\alpha\} + \{\vec{a}\} \cdot [\![\alpha]\!]
\end{aligned}$$

$$\sum_{\Omega_i \in \Omega} \left[\int_{\partial \Omega_i} \alpha \vec{a} \cdot \vec{n} \; d\mathbf{x} \right] = \int_{\partial \Omega} \alpha \vec{a} \cdot \vec{n} \; d\mathbf{x} + \int_{\mathcal{F}_{\mathcal{I}}} [\![\vec{a}]\!]\{\alpha\} + \{\vec{a}\} \cdot [\![\alpha]\!] \; d\mathbf{x}$$

Derivation of Equation (C.19).

$$\begin{aligned}
-\int\limits_{\Omega} \alpha \vec{\nabla}_{\mathsf{p}} \cdot \vec{a}\, d\mathbf{x} &= \sum_{\Omega_i \in \Omega} \left[\int\limits_{\Omega_i} \alpha \vec{\nabla} \cdot \vec{a}\, d\mathbf{x} \right] && \textit{Equation (C.11)} \\
&= \sum_{\Omega_i \in \Omega} \left[\int\limits_{\Omega_i} \vec{\nabla}\alpha \cdot \vec{a}\, d\mathbf{x} - \int\limits_{\partial\Omega_i} \alpha \vec{a} \cdot \vec{n}\, d\mathbf{x} \right] && \textit{Equation (5.12)} \\
&= \int\limits_{\Omega} \vec{\nabla}_{\mathsf{p}}\, \alpha \cdot \vec{a}\, d\mathbf{x} - \sum_{\Omega_i \in \Omega} \left[\int\limits_{\partial\Omega_i} \alpha \vec{a} \cdot \vec{n}\, d\mathbf{x} \right] && \textit{Equation (C.11)} \\
&= \int\limits_{\Omega} \vec{\nabla}_{\mathsf{p}}\, \alpha \cdot \vec{a}\, d\mathbf{x} - \int\limits_{\partial\Omega} \alpha \vec{a} \cdot \vec{n}\, d\mathbf{x} \\
&\qquad - \int\limits_{\mathcal{F}_{\mathcal{I}}} [\![\vec{a}]\!]\{\alpha\} + \{\vec{a}\} \cdot [\![\alpha]\!]\, d\mathbf{x} && \textit{Equation (C.16)}
\end{aligned}$$

BIBLIOGRAPHY

[ABCM02] Arnold D. N., Brezzi F., Cockburn B., Marini L. D.: Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM Journal of Numerical Analysis 39*, 5 (2002), 1749–1779.

[BKH07] Bangerth W., Kayser-Herold O.: *Data Structures and Requirements for hp Finite Element Software*. Tech. Rep. ISC-07-04-MATH, Institute for Scientific Computation, Texas A&M University, 2007.

[Bla72] Blackwell H. R.: Luminance difference thresholds. *Handbook of Sensory Physiology VII*, 4 (1972.), 78101.

[CHH03] Carr N. A., Hall J. D., Hart J. C.: Gpu algorithms for radiosity and subsurface scattering. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2003), Eurographics, Eurographics Association, pp. 51–59.

[CTW*04] Chen Y., Tong X., Wang J., Lin S., Guo B., Shum H.-Y.: Shell texture functions. *ACM Transactions on Graphics 23*, 3 (Aug 2004), 343–353.

[Deb02] Debevec P.: Image-based lighting. *IEEE Comput. Graph. Appl. 22*, 2 (2002), 26–34.

[DEJ*99] Dorsey J., Edelman A., Jensen H. W., Legakis J., Pedersen H. K.: Modeling and rendering of weathered stone. In *Proceedings of SIGGRAPH 99* (1999), ACM, ACM Press / ACM SIGGRAPH, pp. 225–234.

[DJ05] Donner C., Jensen H. W.: Light diffusion in multi-layered translucent materials. *ACM Transactions on Graphics 24*, 3 (Aug 2005), 1032–1039.

[DJ07] Donner C., Jensen H. W.: Rendering translucent materials using photon diffusion. In *Eurographics Symposium on Rendering 2007* (2007), Eurographics, pp. 243–251.

[DS03] Dachsbacher C., Stamminger M.: Translucent shadow maps. In *EGRW '03: Proceedings of the 14th Eurographics workshop on*

Rendering (2003), Eurographics, Eurographics Association, pp. 197–201.

[dSRGKZB83] de S. R. Gago J. P., Kelly D. W., Zienkiewicz O. C., Babuška I.: A posteriori error analysis and adaptive processes in the finite element method: Part II — Adaptive mesh refinement. *Int. J. Num. Meth. Engrg. 19* (1983), 1621–1656.

[DWd*08] Donner C., Weyrich T., d'Eon E., Ramamoorthi R., Rusinkiewicz S.: A layered, heterogeneous reflectance model for acquiring and rendering human skin. *ACM Transactions on Graphics 27*, 5 (2008), 1–12.

[Eva98] Evans L. C.: *Partial Differential Equations*. American Mathematical Society, 1998.

[GFB83] Groenhuis R. A. J., Ferwerda H. A., Bosch J. J. T.: Scattering and absorption of turbid materials determined from reflection measurements 1: Theory. *Appl. Opt. 22*, 16 (1983), 2456–2462.

[GHA05] Gibson A. P., Hebden J. C., Arridge S. R.: Recent advances in diffuse optical imaging. *Physics in Medicine and Biology 50*, 4 (2005), R1–R43.

[GLL*04] Goesele M., Lensch H. P. A., Lang J., Fuchs C., Seidel H.-P.: Disco: acquisition of translucent objects. *ACM Transactions on Graphics 23*, 3 (Aug 2004), 835–844.

[HBV03] Hao X., Baby T., Varshney A.: Interactive subsurface scattering for translucent meshes. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics* (2003), ACM, ACM, pp. 75–82.

[HK93] Hanrahan P., Krueger W.: Reflection from layered surfaces due to subsurface scattering. In *Proceedings of SIGGRAPH 93* (1993), ACM, ACM Press / ACM SIGGRAPH, pp. 165–174.

[HMBR05] Haber T., Mertens T., Bekaert P., Reeth F. V.: A computational approach to simulate subsurface light diffusion in arbitrarily shaped objects. In *GI '05: Proceedings of Graphics*

Interface 2005 (2005), Canadian Human-Computer Communications Society, Canadian Human-Computer Communications Society, pp. 79–86.

[HST*94] Haskell R. C., Svaasand L. O., Tsay T.-T., Feng T.-C., McAdams M. S., Tromberg B. J.: Boundary conditions for the diffusion equation in radiative transfer. *J. Opt. Soc. Am. A 11*, 10 (1994), 2727–2741.

[HV04] Hao X., Varshney A.: Real-time rendering of translucent meshes. *ACM Transactions on Graphics 23*, 2 (2004), 120–142.

[Ish78] Ishimaru A.: *Wave Propagaion and Scattering in Random Media*. Academic Press, 1978.

[JB02] Jensen H. W., Buhler J.: A rapid hierarchical rendering technique for translucent materials. In *Proceedings of SIGGRAPH 2002* (2002), ACM, ACM Press / ACM SIGGRAPH, pp. 576–581.

[JC98] Jensen H. W., Christensen P. H.: Efficient simulation of light transport in scences with participating media using photon maps. In *Proceedings of SIGGRAPH 98* (1998), ACM, ACM Press / ACM SIGGRAPH, pp. 311–320.

[JLD99] Jensen H. W., Legakis J., Dorsey J.: Rendering of wet materials. In *Rendering Techniques '99* (1999), Rendering Techniques, pp. 273–282.

[JMLH01] Jensen H. W., Marschner S. R., Levoy M., Hanrahan P.: A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001* (2001), ACM, ACM Press / ACM SIGGRAPH, pp. 511–518.

[Kaj86] Kajiya J. T.: The rendering equation. In *Computer Graphics (Proceedings of SIGGRAPH 86)* (1986), ACM, pp. 143–150.

[KK05] Kirby R. M., Karniadakis G. E.: Selecting the numerical flux in discontinuous galerkin methods for diffusion problems. *Journal of Scientific Computing 22* (Jun 2005), 385–411.

[Kol01] Kolehmainen V.: *Novel approaches to image reconstruction indiffusion tomography*. PhD thesis, Kuopio University, 2001.

[KPSC06] Kirk B., Peterson J. W., Stogner R. H., Carey G. F.: `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers 22*, 3–4 (2006), 237–254.

[LGB*02] Lensch H. P. A., Goesele M., Bekaert P., Kautz J., Magnor M. A., Lang J., Seidel H.-P.: Interactive rendering of translucent objects. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications* (2002), IEEE, IEEE Computer Society, p. 214.

[LHKK79] Lawson C. L., Hanson R. J., Kincaid D. R., Krogh F. T.: Basic linear algebra subprograms for fortran usage. *ACM Trans. Math. Softw. 5*, 3 (1979), 308–323.

[LPT05] Li H., Pellacini F., Torrance K. E.: A hybrid monte carlo method for accurate and efficient subsurface scattering. In *Rendering Techniques 2005: 16th Eurographics Workshop on Rendering* (Jun 2005), Eurographics, pp. 283–290.

[Mit87] Mitchell D. P.: Generating antialiased images at low sampling densities. In *Computer Graphics (Proceedings of SIGGRAPH 87)* (1987), ACM, pp. 65–72.

[Mit91] Mitchell D. P.: Spectrally optimal sampling for distribution ray tracing. In *Computer Graphics (Proceedings of SIGGRAPH 91)* (1991), ACM, pp. 157–164.

[MKB*03a] Mertens T., Kautz J., Bekaert P., Reeth F. V., Seidel H.-P.: Efficient rendering of local subsurface scattering. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (2003), IEEE, IEEE Computer Society, p. 51.

[MKB*03b] Mertens T., Kautz J., Bekaert P., Seidelz H.-P., Reeth F. V.: Interactive rendering of translucent deformable objects. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (2003), Eurographics, Eurographics Association, pp. 130–140.

[MPBM03] Matusik W., Pfister H., Brand M., McMillan L.: A data-driven reflectance model. *ACM Transactions on Graphics 22*, 3 (Jul 2003), 759–769.

[NRH*77] Nicodemus F., Richmond J., Hsia J., Ginsberg I., Limperis T.: *Geometrical considerations and nomenclature for reflectance.* National Bureau of Standards (US), Oct 1977.

[PH00] Pharr M., Hanrahan P.: Monte carlo evaluation of non-linear scattering equations for subsurface reflection. In *Proceedings of SIGGRAPH 2000* (2000), ACM, ACM Press / ACM SIGGRAPH, pp. 75–84.

[PKK00] Pauly M., Kollig T., Keller A.: Metropolis light transport for participating media. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (2000), Eurographics, Springer-Verlag, pp. 11–22.

[PvBM*06] Peers P., vom Berge K., Matusik W., Ramamoorthi R., Lawrence J., Rusinkiewicz S., Dutré P.: A compact factored representation of heterogeneous subsurface scattering. *ACM Transactions on Graphics 25*, 3 (Aug 2006), 746–753.

[SAHD95] Schweiger M., Arridge S. R., Hiraoka M., Delpy D. T.: The finite element method for the propagation of light in scattering media: Boundary and source conditions. *Medical Physics 22*, 11 (1995), 1779–1792.

[SG05] Si H., Gaertner K.: Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable* (Sep 2005), Meshing Roundtable, pp. 147–163.

[SKS02] Sloan P.-P., Kautz J., Snyder J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of SIGGRAPH 2002* (2002), ACM Press / ACM SIGGRAPH, pp. 527–536.

[SLS05] Sloan P.-P., Luna B., Snyder J.: Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics 24*, 3 (Aug 2005), 1216–1224.

[Sta95] Stam J.: Multiple Scattering as a Diffusion Process. In *Eurographics Rendering Workshop 1995* (1995), Eurographics, pp. 41–50.

[THCM04] Tarini M., Hormann K., Cignoni P., Montani C.: Polycubemaps. *ACM Transactions on Graphics 23*, 3 (Aug 2004), 853–860.

[Tur92] TURK G.: Re-tiling polygonal surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 92)* (1992), ACM, pp. 55–64.

[TWL*05] TONG X., WANG J., LIN S., GUO B., SHUM H.-Y.: Modeling and rendering of quasi-homogeneous materials. *ACM Transactions on Graphics 24*, 3 (Aug 2005), 1054–1061.

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. *ACM Transactions on Graphics 25*, 3 (Aug 2006), 1081–1088.

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics 24*, 3 (Aug 2005), 1098–1107.

[WTL05] WANG R., TRAN J., LUEBKE D.: All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM Transactions on Graphics 24*, 3 (Aug 2005), 1202–1207.

[WZT*08] WANG J., ZHAO S., TONG X., LIN S., LIN Z., DONG Y., GUO B., SHUM H.-Y.: Modeling and rendering of heterogeneous translucent materials using the diffusion equation. *ACM Transactions on Graphics 27*, 1 (2008), 9.1–9.18.