Simulating Global Illumination Using Adaptive Meshing

Paul S. Heckbert

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720
USA

June 21, 1991

Simulating Global Illumination Using Adaptive Meshing

by

Paul S. Heckbert

Simulating Global Illumination Using Adaptive Meshing

by

Paul S. Heckbert

Abstract

One of the goals of computer graphics is the simulation of global illumination, the interreflection of light between diffuse and specular surfaces in three-dimensional scenes. Like thermal radiation, as studied in mechanical engineering and physics, global illumination is governed by an integral equation. A distinguishing feature of the integral equations of global illumination is that their solutions have numerous discontinuities, such as shadow edges, caused by occlusion. We show that the most common global illumination algorithms, radiosity and ray tracing, are simple finite element and Monte Carlo methods for solving integral equations, respectively. In the process of re-deriving these techniques, a number of alternative algorithms with higher accuracy are suggested. The principal alternatives explored in this thesis are adaptive meshing techniques that resolve discontinuities.

Radiosity algorithms can be made more accurate by a priori *discontinuity meshing*, placing mesh boundaries on significant discontinuities. Discontinuities are found using an object-space visible surface algorithm from the point of view of object vertices. The accuracy of radiosity simulations can also be improved using linear, quadratic, or higher degree elements instead of constant elements. Algorithms are developed first for two-dimensional *radiosity in flatland* problems, then extended to three dimensions.

Scenes containing diffuse and specular surfaces are most easily simulated using Monte Carlo ray tracing techniques. Traditional ray tracing algorithms trace rays from the eye into the scene. A bidirectional ray tracing algorithm is demonstrated that traces rays from both the lights and the eye, storing the diffuse intensity function in a *radiosity texture* on each diffuse surface in the scene. These textures are adaptively subdivided in order to resolve shadow edges and other discontinuities.

Contents

List of Figures

Acknowledgements

Special thanks are due to Keith Miller and Jim Winget for teaching me about the finite element method. Keith pointed out that discontinuities need special treatment, and Jim suggested the use of objective error metrics for measuring accuracy. In November 1990, when Jim and I began collaborating, we combined work I had done on flatland radiosity, linear elements, discontinuity meshing, and analytic integration with work he had done on higher degree elements, Galerkin methods, and numerical integration for a paper [Heckbert-Winget91]. Many of Jim's ideas have rubbed off here.

Jim Demmel and William Kahan pointed me in the right direction when I was just beginning to understand the numerics. Brian Barsky and Marc Levoy helped me summarize what I had done to a graphics community that expects pretty pictures, not numerical simulations. Providing help and financial support during my four years at Berkeley were Brian Barsky, Carlo Séquin, John Ousterhout, Apple Computer, and AT&T. Thanks also to all of the graphics grad students at Berkeley for their camaraderie. The UC Berkeley libraries have been a wonderful resource; they made it easy to explore the early thermal radiation literature.

For several years, Greg Ward has been a co-adventurer in quest of global illumination algorithms with guaranteed error bounds. We've had many discussions on the pros and cons of radiosity versus ray tracing. Jim Arvo and Keith Miller pointed out the connection between light ray tracing and adjoint integral operators. Some errors in an earlier paper [Heckbert90] were spotted by Nelson Max; they are corrected here. Steve Omohundro pointed me to the density estimation literature.

Henry Moreton, Seth Teller, and Tom Funkhouser generously let me commandeer their SGI workstations. Formatting was eased enormously with Mark Segal's Tex previewer, which provides WYSIWYG typesetting for UNIX. Ziv Gigus bravely led the way through the Latex swamp, and also helped explain the complexities of visibility in three dimensions. Finally, many thanks to Bridget Johnson for providing all three kinds of *mush*!

Please email comments or questions about the thesis to the author at ph@miro.berkeley.edu.

Chapter 1

Introduction

1.1 Global Illumination

The goal of image synthesis research in computer graphics is the development of methods for modeling and rendering three-dimensional scenes. One of the most challenging tasks of image synthesis is the accurate and efficient simulation of *global illumination* effects: the illumination of surfaces in a scene by other surfaces. Early rendering programs treated the visibility (visible surface) and shading tasks independently, employing a *local illumination* model that assumed that the shading of each surface is independent of the shading of every other surface [Foley et al. 90, p. 760]. Local illumination models typically assume that light comes from a finite set of point light sources only. Global illumination models, on the other hand, recognize that visibility and shading are interrelated: the intensity of a surface point is determined by the intensity of all the surfaces visible from that point.

1.2 Visual Effects of Global Illumination

The visual effects of global illumination are so commonplace that we are hardly conscious of them in everyday life, but they are often conspicuous by their absence in computer-generated images. With simple image synthesis algorithms, shadows are typically hard-edged (if they are simulated at all), surfaces may look smooth and plastic, and the scene may have the stark appearance of outdoor lighting. With more sophisticated image synthesis techniques, it is possible to simulate the *penumbras* (soft shadows) from area light sources, the roughness and reflection properties of real materials, and the indirect lighting (interreflection) and color bleeding effects of global illumination. We define *direct lighting* as light that travels in a straight, uninterrupted path from light source to surface, and *indirect lighting* as light that undergoes reflections or transmissions before reaching a surface.

Indirect lighting is significant in indoor scenes. In a room with only downward-pointing lights, for example, the ceiling receives all of its light indirectly. If the floor is painted black then the photons of the light beam that hit it are absorbed, and the ceiling will not be lit; but if the floor is painted bright white or is mirrored then a larger fraction of the photons will be reflected, and they will continue to ricochet around the scene, illuminating the ceiling and everything else. These are facts that every interior designer understands intuitively, but that computer graphics has only begun to simulate in the past ten years. With a typical ray tracer, for instance, indirect illumination is either totally neglected, so

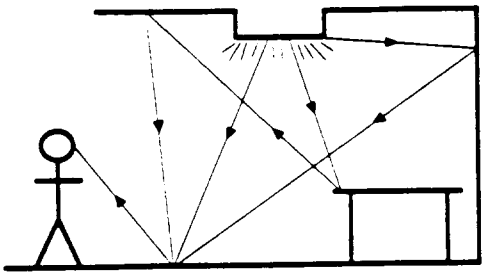

Figure 1.1: *Light reaches surface points by various paths.*

that objects not directly lit are rendered as black, or a constant *ambient term* is included in the illumination model as a first approximation to global illumination [Foley et al. 90. p. 722].

Inaccurate simulation of global illumination on a real, physical scene is like accurate simulation on an unreal, non-physical scene. For example, in figure 1.1, some of the photons that reach our eye from a given point on the floor come directly from the light source, while others travel a path of one or more bounces before reflecting off the floor into our eye.[1] An algorithm that simulates only direct lighting effectively simulates a scene in which surfaces absorb all photons except those that reflect directly toward the viewer. Such a world would be extremely nonphysical, of course, so we should not be surprised that a direct illumination lighting model sometimes leads to unreal-looking images.

1.3 Applications

Global illumination phenomena are relevant to a number of engineering and artistic problems. In architecture and lighting design, a designer must choose colors and materials for walls, floors, and ceiling and the placement and choice of light fixtures and windows in order to achieve a desired light level and light distribution. The effects of direct illumination are simple enough that lighting designers can compute them using tables or hand calculators [IES87a], but interreflection greatly complicates the simulation. In many cases, it is currently necessary to build a scale model, or simply to construct the building and test design choices there – a very expensive proposition. Another related field is thermal radiation in mechanical engineering, in which one simulates the exchange of energy between hot objects that radiate at infrared and visible wavelengths. Although the wavelength range is different, the phenomena and equations of thermal radiation are similar to those in global illumination. In computer graphics, realistic image synthesis has been used for movie special effects, computer art, computer-aided design, flight simulators, and scientific visualization.

[1] Strictly speaking, light has both a particle nature and a wave nature, and photons don't have a well-defined identity, so it is somewhat misleading to suggest that the incident photon and the reflected photon are "the same photon". For our purposes, however, we can imagine that they are, and we often speak of a single photon bouncing around a scene.

1.4 Why Integral Equations?

Global illumination is governed by an integral equation. Why this comes about and what this means are described intuitively here; a rigorous derivation follows in chapter 2.

The light leaving a surface consists of two parts, the emitted light (if this is the surface of a light source) and the reflected or transmitted light. Surfaces differ in their reflective and transmissive behavior: some are *diffuse* or matte, appearing equally bright from all directions, some are *specular* or glossy, showing strong highlights, and some show a mixture of diffuse and specular characteristics. In general, the intensity (roughly speaking, the brightness) of a surface is a function of both position and viewing direction.

Assuming that the geometry and the emissive, reflective and transmissive properties of all surfaces are known, the intensity at each surface point in each direction is determined. The light reflected at a surface point in a given direction could, in general, have come from any direction in the hemisphere visible to that point. Some of this light will come from the designated light sources, and other light will come from reflective surfaces. Transmitted light travels similarly. The intensity at a point is thus related to the intensity of all points visible to it.

The equation describing the interdependency thus has the general form

$$\text{intensity}(p) = \text{emit}(p) + \int_{\text{all surfaces}} \text{intensity}(q)\,\text{scat}(p,q)\,dq$$

where p and q each represent a point and direction, where intensity(p) is the intensity at p, where emit(p) is the intensity of emitted light at p, and scat(p,q) is the fraction of light leaving q that is scattered (reflected or transmitted) to p. Naturally, scat(p,q) = 0 if there is an occluding object between points p and q. The above is called an integral equation because the unknown intensity function appears inside an integral. When the radiation passes through a *participating medium*, a gaseous or liquid medium that either absorbs, scatters, or emits light, the equation becomes *integro-differential*: a combination of both integrals and differentials.

Integral and integro-differential equations cannot be solved analytically, in general, so numerical methods must be used. Algorithms for simulating global illumination can be characterized by the approximations they make to the integral equation.

1.5 Goals of This Work

This research is directed in the long term toward the development of accurate, general, and efficient algorithms for global illumination, where "accurate" could mean either objective or subjective accuracy, depending on the application, "general" means complex scenes containing curved shapes with arbitrary scattering functions and participating media, and "efficient" means low usage of time and memory. Realization of these goals is years off, but the hope is that this thesis can contribute toward those goals in several ways: by leading to a better understanding of the integral equations that govern global illumination, by providing a framework in which existing algorithms can be better understood, and by suggesting new algorithms for approximate solution of these integral equations.

1.6 Organization of the Thesis

The thesis is divided into four main chapters: previous work, radiosity in flatland, radiosity in 3-D, and bidirectional ray tracing.

Chapter 2 discusses previous work: definitions and approximate solution methods for integral equations, a survey of thermal radiation work, including a number of references to early work on the radiosity method, and a survey of past shadow and global illumination algorithms.

Chapters 3 and 4 explore one numerical method for solving the integral equations of global illumination: reduction to a linear system of equations. Chapter 3 focuses on radiosity in flatland, the simulation of global illumination in two-dimensional diffuse scenes. Examination of this simplified problem leads to a better understanding of the integral equations governing global illumination and the properties of the solution functions, and allows easier testing of algorithms. It is shown that existing radiosity algorithms can be viewed as simple finite element methods for solving integral equations.[2] Better approximation methods are discovered and tested. In chapter 4, the algorithms discovered in flatland are generalized to 3-D, and many of the computational geometry problems presented by 3-D visibility testing are discussed.

Chapter 5 explores an alternative, Monte Carlo approach to the solution of integral equations for scenes containing both specular and diffuse surfaces. The approach explored is the storage of diffuse intensity information in radiosity textures on each surface, and the creation and use of these textures using *bidirectional ray tracing* from both the lights and the eye.[3] This algorithm was developed before the research in the remainder of the thesis was done, but with hindsight we briefly explain how bidirectional ray tracing can be regarded as an integral equation approximation method.

The concluding chapter summarizes the accomplishments of this work and discusses ideas for future work.

[2] Some work adapted from a paper jointly written with Jim Winget [Heckbert-Winget91].

[3] Chapter adapted from [Heckbert90], with corrections and additions.

Chapter 2

Previous Work

Global illumination is an interdisciplinary research area. As previously mentioned, global illumination is essentially synonymous with interreflection in the field of lighting design (illuminating engineering), and with thermal radiation in mechanical engineering. There is also overlap with astrophysics, where the thermal radiation in stars is examined, and the field of neutron transport, where the flow of neutrons through matter is studied. There are differences in emphasis, however. In global illumination and lighting design, media are generally assumed to be non-participating (i.e. no fog or other atmospheric effects), but in the fields of thermal radiation, astrophysics, and neutron transport, participating media are quite significant.

Global illumination research involves a number of methods from mathematics: functional analysis for the study of function spaces and operators, integral equations and numerical methods for solving them, finite element methods, numerical integration, linear algebra, and Monte Carlo methods. In computer science, global illumination methods draw heavily on computer graphics and computational geometry techniques for visibility testing and shadow generation.

In this chapter we review five topics: integral equations, thermal radiation, computer vision, shadow algorithms, and global illumination algorithms.

2.1 Integral Equations

This section defines integral equation terminology and discuss analytic and numerical solution methods. There are a number of good books on integral equations for both the beginner [Jerri85] and the mathematically sophisticated [Delves-Mohamed85,Atkinson76, Courant-Hilbert37,Hildebrand65]. Most of this section is adapted from [Delves-Mohamed85, Atkinson76].

An *integral equation* is an equation in which the function to be determined appears inside an integral. The class of integral equation of interest in this work is the *Fredholm integral equation of the second kind*, with general form[1]

$$u(s) = e(s) + \int_a^b dt\, \kappa(s,t)u(t)$$

[1] We use the notation "$\int dx\, f(x)$" instead of "$\int f(x)\, dx$" because it most clearly indicates the variable of integration during multiple integration.

where e and the *kernel* κ are given and u is to be determined. This is a Fredholm integral equation because the limits of integration a and b are constant, and it is a second-kind integral equation because the unknown function $u(s)$ appears outside the integral. Most integral equation properties and solution methods generalize when the domain variables s and t are multi-dimensional [Courant-Hilbert37, p. 152].

The above equation is abbreviated as $u = e + \mathcal{K}u$, where $\mathcal{K}$ denotes the integral operator which when applied to a function u, yields a function

$$(\mathcal{K}u)(s) = \int_a^b dt\, \kappa(s,t)u(t)$$

We write the inner product of two real functions f and g over the domain $[a,b]$ as

$$(f,g) = \int_a^b ds\, f(s)g(s)$$

The *adjoint* of an integral operator $\mathcal{K}$ with kernel $\kappa(s,t)$ is the integral operator $\mathcal{K}^*$ with kernel $\kappa(t,s)$:

$$(\mathcal{K}^*u)(s) = \int_a^b dt\, \kappa(t,s)u(t)$$

It can be shown that $(\mathcal{K}u, v) = (u, \mathcal{K}^*v)$ for any square-integrable functions u and v and any operator $\mathcal{K}$ with square-integrable kernel κ [Delves-Mohamed85]. A function u and kernel κ are defined to be square-integrable when

$$\int_a^b ds\, |u(s)|^2 < \infty$$

and

$$\int_a^b ds \int_a^b dt\, |\kappa(s,t)|^2 < \infty$$

respectively. An integral operator is said to be *self-adjoint* if it equals its adjoint, or equivalently, if its kernel is symmetric:

$$\mathcal{K} = \mathcal{K}^* \qquad \Longleftrightarrow \qquad \kappa(s,t) = \kappa(t,s) \quad \forall s,t$$

Formally, our integral equation $u = e + \mathcal{K}u$ can be rewritten $(\mathcal{I} - \mathcal{K})u = e$, where $\mathcal{I}$ is the identity operator, and the solution can be obtained by inverting the operator $\mathcal{I} - \mathcal{K}$:

$$u = (\mathcal{I} - \mathcal{K})^{-1}e$$

It is typically impractical and unnecessary to invert the integral operator explicitly, however [Alpert90].

2.1.1 Neumann Series Approximation

Approximate solutions to many integral equations can be found iteratively. Starting with some initial guess $u^{(0)}(s)$, subsequent approximations are defined by

$$u^{(i)} = e + \mathcal{K}u^{(i-1)}$$

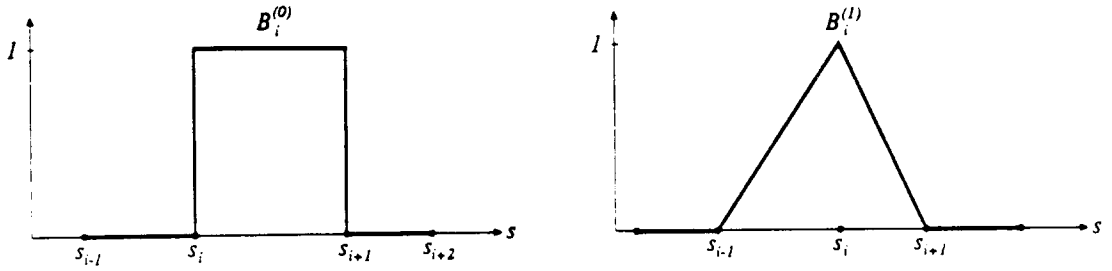

Figure 2.1: *Left: piecewise-constant (box) basis function. Right: piecewise-linear (hat) basis function.*

If we start with $u^{(0)} = e$, then the ith approximant is the truncated series

$$u^{(i)} = e + \mathcal{K}e + \mathcal{K}^2 e + \cdots + \mathcal{K}^i e$$

where $\mathcal{K}^i$ denotes i successive applications of the integral operator $\mathcal{K}$.

The sequence $u^{(i)}$ converges if the norm of the integral operator is less than 1 ($||\mathcal{K}|| < 1$). where the operator norm is defined in terms of a function norm:

$$||\mathcal{K}|| = \max_{u \neq 0} \frac{||\mathcal{K}u||}{||u||} \tag{2.1}$$

The L_p norms are a general class of function norms:

$$||f||_p = \left(\int_a^b dx \, |f(x)|^p \right)^{1/p} \tag{2.2}$$

the most common of which are the L_1 norm, the L_2 norm, and the L_∞ norm.

When $||\mathcal{K}|| < 1$, the exact solution to the integral equation is given by the *Neumann series*

$$u = u^{(\infty)} = \sum_{i=0}^{\infty} \mathcal{K}^i e \tag{2.3}$$

The Neumann series is a generalization of the geometric series for $1/(1-\alpha)$.

2.1.2 Projection Methods

The projection method for the solution of integral equations approximates the exact solution function, which lies in an infinite-dimensional function space, by projecting it to a finite-dimensional function space. This is also the approximation method used in the finite element method [Becker et al. 81,Strang-Fix73]. The exact solution function $u(s)$ is approximated by a linear combination of basis functions:

$$\hat{u}(s) = \sum_{i=1}^{n} u_i B_i(s) \tag{2.4}$$

where u_i are the unknown coefficients and B_i are the chosen basis functions.

Common choices for basis functions are piecewise polynomials of finite support. For a one-dimensional domain $[a,b]$, polynomial elements are defined in terms of a sequence of element endpoints (*nodes*) s_i, where $a = s_0 \le s_1 \le \ldots \le s_n = b$. These points are analogous to knot vectors for splines [Bartels et al. 87]. The box function and hat function are the simplest polynomial elements (figure 2.1):

$$\begin{aligned} \text{box:} \quad & B_i^{(0)}(s) = \begin{cases} 1 & \text{if } s_i \le s < s_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ \text{hat:} \quad & B_i^{(1)}(s) = \begin{cases} \frac{s - s_{i-1}}{s_i - s_{i-1}} & \text{if } s_{i-1} \le s \le s_i \\ \frac{s_{i+1} - s}{s_{i+1} - s_i} & \text{if } s_i \le s \le s_{i+1} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.5}$$

We call these *constant elements* and *linear elements*, respectively. Quadratic and higher degree elements are common as well.

Of course, it is an approximation to assume that a function in this function space will solve the integral equation. We would like to find a solution that minimizes the error $\hat{u} - u$, but since the exact solution u is unknown, this is impossible. Instead, we choose $\hat{u}$ so that $\hat{u} \approx e + \mathcal{K}\hat{u}$ is nearly satisfied; the "best" solution in this function space is defined to be the one that minimizes the *residual*

$$r(s) = \hat{u}(s) - (\mathcal{K}\hat{u})(s) - e(s)$$

The minimum can be defined in several ways. The *collocation* method constrains the residual to be zero at a set of points, while the *Galerkin* method constrains the residual function to be orthogonal to each of the basis functions. If n constraints are chosen, then either of these projection methods reduces the integral equation problem to a system of n equations in n unknowns. Since the integral operator $\mathcal{K}$ is linear, this will be a linear system of equations.

Collocation Method

The collocation method constrains the approximation by requiring that the residual be zero at n collocation points s_i'. For 1-D constant elements, the collocation points are often chosen to be the midpoints of the elements, while for 1-D linear elements, the collocation points are usually the element endpoints. The constraint for each i has the form $r(s_i') = 0$, so

$$\begin{aligned} 0 &= r(s_i') \\ &= \hat{u}(s_i') - (\mathcal{K}\hat{u})(s_i') - e(s_i') \\ &= \sum_{j=1}^{n} u_j \Big(B_j(s_i') - (\mathcal{K}B_j)(s_i') \Big) - e(s_i') \\ &= \sum_{j=1}^{n} u_j (M_{ij} - K_{ij}) - e_i \end{aligned}$$

where

$$\begin{aligned} M_{ij} &= B_j(s_i') \\ K_{ij} &= (\mathcal{K}B_j)(s_i') = \int_a^b dt\, \kappa(s_i', t) B_j(t) \\ e_i &= e(s_i') \end{aligned}$$

Collocation thus results in the linear system of equations:

$$(\mathbf{M} - \mathbf{K})\mathbf{u} = \mathbf{e} \tag{2.6}$$

where $\mathbf{M}$ and $\mathbf{K}$ are $n \times n$ matrices and $\mathbf{u}$ and $\mathbf{e}$ are n-element column vectors. (Boldface denotes matrices and vectors.)

The matrix $\mathbf{M}$ is called the *mass matrix* or *stiffness matrix*. When constant elements are used with collocation points in the interior of each element, or when linear elements are used with a collocation point at each element endpoint, the mass matrix will be the identity: $\mathbf{M} = \mathbf{I}$. For broader basis functions, the more limited the support of the basis functions, the more nearly diagonal the mass matrix will be.

Galerkin Method

The Galerkin method constrains the approximation by requiring that the residual function be orthogonal to each of the n basis functions. Using inner product notation, the Galerkin orthogonality constraint for each i is:

$$\begin{aligned} 0 &= (B_i, r) \\ &= (B_i, u) - (B_i, \mathcal{K}u) - (B_i, e) \\ &= \sum_{j=1}^{n} u_j (B_i, B_j) - \sum_{j=1}^{n} u_j (B_i, \mathcal{K}B_j) - (B_i, e) \\ &= \sum_{j=1}^{n} u_j (M_{ij} - K_{ij}) - e_i \end{aligned}$$

with mass matrix

$$M_{ij} = (B_i, B_j) = \int_a^b ds\, B_i(s) B_j(s)$$

discretized kernel

$$K_{ij} = (B_i, \mathcal{K}B_j) = \int_a^b ds\, B_i(s) \int_a^b dt\, \kappa(s,t) B_j(t)$$

and homogeneous vector

$$e_i = (B_i, e) = \int_a^b ds\, B_i(s) e(s)$$

As with collocation, Galerkin transforms the integral equation into a linear system of equations:

$$(\mathbf{M} - \mathbf{K})\mathbf{u} = \mathbf{e} \tag{2.7}$$

Computing the matrix $\mathbf{K}$ with Galerkin methods requires double integration of the kernel, but with collocation methods, only single integration is required. In either case, these integrals often cannot be done analytically, and must be done numerically. The mass matrix is an identity iff the basis functions are orthonormal.

2.1.3 Rayleigh-Ritz Variational Method

Like the projection methods, the *Rayleigh-Ritz variational method* seeks the minimum-error solution in a reduced function space in the form of (2.4), but it differs in derivation: rather than first restrict the function space to finite dimension and then define the error in the finite space, the Rayleigh-Ritz technique defines an error in an infinite-dimensional (Hilbert) space and then restricts the function space. As we will see, however, the Rayleigh-Ritz method is essentially the same as the Galerkin method.

Historically, the Rayleigh-Ritz method has used global basis functions, while the collocation and Galerkin methods have typically employed local, finite element basis functions. Consequently, the approximation function $\hat{u}$ of the Rayleigh-Ritz method generally does not interpolate or approximate a set of points (s_i', u_i), as it does with collocation or Galerkin methods.

Solution of the integral equation can be rephrased as a variational optimization problem. The solution u to the integral equation $u = e + \mathcal{K}u$, for self-adjoint $\mathcal{K}$, is an extremum of the following quadratic integral form with respect to variations in the function $u(s)$ [Hildebrand65, p. 328]:

$$Q(u) = \frac{1}{2}(u, u) - \frac{1}{2}(\mathcal{K}u, u) - (e, u)$$

In general, it is impossible to find the exact function u that results in an extremum of Q in the space of all continuous functions. The Rayleigh-Ritz method determines an approximate extremum of the infinite-dimensional function space by finding the extremum $\hat{u}$ of $Q(\hat{u})$ in the finite-dimensional function space of equation (2.4) [Hildebrand65, p. 181].

At the extremum, variation of any coefficient u_i has no first order effect, so

$$\begin{aligned}
\frac{\partial Q}{\partial u_i} = 0 &= (B_i, u) - \frac{1}{2}(\mathcal{K}B_i, u) + \frac{1}{2}(\mathcal{K}u, B_i) - (e, B_i) \\
&= \sum_{j=1}^{n} u_j (B_i, B_j) - \frac{1}{2}\sum_{j=1}^{n} u_j \Big((\mathcal{K}B_i, B_j) + (B_i, \mathcal{K}B_j) \Big) - (B_i, e) \\
&= \sum_{j=1}^{n} u_j (M_{ij} - K_{ij}) - e_i
\end{aligned}$$

where

$$\begin{aligned}
M_{ij} &= (B_i, B_j) \\
K_{ij} &= \frac{1}{2}\Big((\mathcal{K}B_i, B_j) + (B_i, \mathcal{K}B_j) \Big) \\
e_i &= (B_i, e)
\end{aligned}$$

As before, the coefficients u_i of the basis functions are calculated by solving the system of equations $(\mathbf{M} - \mathbf{K})\mathbf{u} = \mathbf{e}$.

If the integral operator $\mathcal{K}$ is self-adjoint, then $K_{ij} = (B_i, \mathcal{K}B_j)$, and this method is equivalent to the Galerkin method [Kantorovich-Krylov58, pp. 150-153], [Fletcher84], [Sparrow-Haji-Sheikh65, discussion comments from Hrycak]. The Rayleigh-Ritz variational method can thus be regarded as an alternative derivation of the Galerkin method.

2.1.4 Comparison of Numerical Solution Techniques

Collocation makes use of the kernel of the integral operator only along constant-s lines in st space, whereas Galerkin makes use of the entire $[a, b] \times [a, b]$ square domain of st space.

Thus, from an intuitive standpoint, Galerkin extracts more information from the kernel than collocation does, allowing it to approximate the integral operator more robustly.

2.2 Thermal Radiation

Heat transfer is the branch of mechanical engineering concerned with the propagation of heat. There are three mechanisms for heat transfer: conduction, convection, and radiation. *Conduction* is the transfer of heat to neighboring points due to a temperature gradient, without appreciable displacement of particles, *convection* is the transfer of heat due to the mixing of fluids, and *thermal radiation* is electromagnetic radiation at visible (*light*) or infrared wavelengths that is emitted by a material due to its temperature [McAdams54]. Quantum mechanics explains that radiation is caused by the release of photons from excited molecules and atoms. The excitation of matter when irradiated at infrared wavelengths causes atomic rotations and vibrations that we call *heat*.

Radiation is different in character from conduction and convection in several ways. First, compared to conduction and convection, thermal radiation is relatively insignificant in solid or liquid media and at low temperatures. It is a more significant fraction of the total heat transfer in gases and at high temperatures [Eckert-Drake72]. Conduction and convection are affected primarily by temperature difference and very little by temperature level, whereas radiation increases rapidly with temperature level [Hottel54]. The energy transferred by conduction and convection is proportional to temperature differences to the first power (approximately), while thermal radiation is proportional to temperature to the fourth power [Siegel-Howell81]. Secondly, radiation requires no medium (it can take place in a vacuum), while conduction requires a liquid or solid medium, and convection requires a liquid medium. A third difference is that conduction and convection are local phenomena involving neighboring matter, while thermal radiation results in global effects between distant surfaces. This is also known as action-at-a-distance. The effects of conduction and convection are described by differential equations but the effects of radiation are governed by integral equations [Eckert-Drake72, p. 567]. Radiation in participating media and radiation with conduction and convection are some of the most complex phenomena studied in thermal radiation; they are governed by integro-differential equations.

2.2.1 Heat Transfer Culture

The thermal radiation literature is best understood within the context of the heat transfer culture from which it developed. In the tradition of heat transfer, physical processes are simulated to compute physical quantities such as temperature and intensity in real units. A heat transfer engineer measures intensity in watts per meter squared, for example, while a (casual) computer graphicist might equate intensity with an integer between 0 and 255.

Much of the early thermal radiation work was done in Germany and the United States in the 30's through 50's, before computers were commonplace. Consequently, problems that would today be solved by software tools employing a brute force algorithm were in the past worked out by physical analogy, clever approximations, and analog methods. Analytic techniques were used wherever possible instead of more computationally involved numerical methods. Although integral equations are the most natural way to express the fundamental phenomena of thermal radiation, they have often been avoided in the thermal radiation literature (one of the most popular texts doesn't introduce them until page 250 [Siegel-

Howell81]). This tendency may result because integral equations are not widely taught in most engineering curricula.

Hilbert discussed integral equations in thermal radiation problems early in this century [Hilbert12]. Much of the early work on the solution of integral equations for thermal radiation concerned specific problems, such as the heat transfer through a cylindrical hole in a furnace wall [Buckley27,Buckley28,Buckley34,Hottel-Keller55,Sparrow60]. Simplified problems involving two plates of finite width and infinite length were studied by Sparrow in the thermal radiation literature [Sparrow60], and later by Horn in the computer vision literature [Horn77]. Early derivation of integral equations for general diffuse scenes was made by Poljak in the USSR and by Jensen in Denmark [Poljak35,Jensen48]. Their methods are described in Jakob's book [Jakob57]. After this early work, it seems that there was a period in which the integral equation formulation of the problem was overshadowed by the radiosity formulation. The popularity of the radiosity method and other methods that assumed an entirely diffuse scene was spurred in part by the shortage of empirical data or mathematical models for more complex reflectivity functions.

Within the past two decades, advances in computer hardware and software have revolutionized heat transfer even further. Most simulations are now done by computer using the finite element method, and many of the clever old analytic tricks and analog devices are being supplanted by brute force algorithms. In spite of the popularity of finite element methods, however, many finite element programs allow only the simplest geometric primitives and meshes to be used. In the area of data structures for the description of geometry and topology, many of these large systems are years behind current techniques of computer-aided geometric design. This is due, in part, to the use of outdated programming languages like FORTRAN which lack structured data types.

Thermal radiation research has been strongly driven by applications. A typical early application was the analysis of heat loss through a furnace window [Hottel-Keller55]. In the past thirty years, applications such as solar power, nuclear reactors, nuclear explosives, and space exploration have become more common. A frequent application of thermal radiation in the 1960's was the design of cooling fins to dispose of waste heat from spaceships.

2.2.2 Radiation To and From a Surface Element

Radiation is absorbed and scattered as it propagates through a medium. A medium is called a *participating medium* if the absorption and scattering are significant, and a *non-participating medium* if they are negligible. In global illumination, for example, we regard fog and smoke as participating media, and vacuum or air as non-participating. It is convenient in thermal radiation to model geometry as consisting of volumes of material with homogeneous properties bounded by surfaces. In fact, surfaces are a mathematical idealization of the physical reality, which is a narrow transition region between two materials of different atomic composition.

A good summary of the basic physics of thermal radiation is [Eckert-Drake72], from which most of this section is adapted. An alternative derivation of the physics of reflection and transmission for computer graphics is given in [Shirley91]. That derivation has the advantage of using the standardized terminology of illuminating engineering [IES87b], while the summary here employs the less standardized terminology of thermal radiation. For an excellent bibliography on the literature of *radiometry* (the study of radiation), which has much in common with the field of thermal radiation, see [Horn-Brooks89, bibliography].

The following table summarizes the relevant physical quantities from thermal radiation. their dimension, and their units in the metric system:

SYMBOL	PHYSICAL QUANTITY	DIMENSION	UNITS
ϵ	emissivity	1	1
ρ	reflectivity	1	1
τ	transmissivity	1	1
α	absorptivity	1	1
T	temperature	temperature	°Kelvin
λ	wavelength	distance	m
$d\mathbf{x}$	differential surface area	area	m^2
$d\Theta$	differential solid angle	solid angle	steradian
	energy	energy	joule = $kg\,m^2/s^2$
Φ	heat flux, power	power = energy/time	watt = joule/s
e	emissive power	power/area	$watt/m^2$
u	radiosity	power/area	$watt/m^2$
i	intensity	power/(area × solid angle)	$watt/(m^2 steradian)$

In a general form, radiation is a function of phase, polarization, time t, wavelength λ, position $\mathbf{x} = (x, y, z)$, and direction $\Theta = (\theta, \phi)$ [Nicodemus76,Nicodemus78], where θ is azimuth and ϕ is angle from the surface normal (inclination). In this work we make the assumptions of geometric optics, namely that radiation can be simulated using rays and that radiation is *incoherent*, having all phases. These assumptions preclude the simulation of diffraction and interference phenomena. Unless stated otherwise, we also ignore polarization, and assume that scenes are static, media are non-participating, and that surfaces are *gray*, having wavelength-independent properties within the wavelength band of interest. If emissivity, reflectivity, or other properties vary with wavelength then the spectrum can be broken into several wavelength bands, each of which can be simulated independently.

Radiation is usually measured in the units of *intensity*, which is defined to be the energy passing through a given area in a given direction in a given amount of time. The heat flux through a differential surface element of intensity i and area $d\mathbf{x}$ at position $\mathbf{x}$, in direction Θ and solid angle $d\Theta$, is

$$\Phi = i(\mathbf{x}, \Theta) \cos\phi \, d\mathbf{x} \, d\Theta$$

The cosine term enters because the projected area of $d\mathbf{x}$ in direction Θ is $d\mathbf{x} \cos\phi$.

Emission

A *blackbody* is defined to be a perfect emitter and absorber; the material that emits and absorbs the maximum energy for a given temperature. The *emissive power* of a material is the energy emitted over all wavelengths and directions per unit time and area. By Stefan-Boltzmann's law, the *emissive power* of a blackbody at temperature T into a medium with refractive index n is

$$e_b = n^2 \sigma T^4$$

where σ is the Stefan-Boltzmann constant. Henceforth we assume that the medium has index of refraction $n = 1$, to simplify the formulas. Air's index of refraction is very close to 1. The intensity i_b of a blackbody is direction-independent, so when integrated over the hemisphere of directions, it is easily related to the emissive power:

$$e_b = \int_{\text{hemi}} d\Theta \cos\phi \, i_b = i_b \int_0^{2\pi} d\theta \int_0^{\pi/2} d\phi \sin\phi \cos\phi = \pi i_b$$

since the differential solid angle is: $d\Theta = \sin\phi \, d\theta d\phi$.

The emissive power of a general surface is defined relative to a blackbody:

$$e = \epsilon_h e_b$$

where ϵ_h is the *hemispherical emissivity*, a fraction between 0 and 1 that can vary with position. Emissivity is 1 for a blackbody and 0 for a perfect reflector or transmitter.

The intensity of an emitter is direction-dependent, in general. If the directional intensity distribution of emitted radiation is $i_{\text{emit}}(\Theta)$ then the *directional emissivity* $\epsilon(\Theta)$ is defined as a fraction of the blackbody emission:

$$i_{\text{emit}}(\Theta) = \epsilon(\Theta) i_b \tag{2.8}$$

Similar to the blackbody, the integral of emitted intensity over the hemisphere weighted by projected area gives the emissive power:

$$e = \int_{\text{hemi}} d\Theta \cos\phi \, i_{\text{emit}}(\Theta)$$

As a consequence of these definitions, the hemispherical emissivity and directional emissivity are interrelated by

$$\epsilon_h = \frac{1}{\pi} \int_{\text{hemi}} d\Theta \cos\phi \, \epsilon(\Theta)$$

For a *diffuse emitter*, with direction-independent emissivity, $\epsilon_h = \epsilon(\Theta)$.

Reflection and Transmission

When thermal radiation arrives at a surface, the energy is either reflected back into the original hemisphere, transmitted into the opposite hemisphere, or absorbed (transduced into kinetic energy).

Reflection is fully described by the *bidirectional reflectivity distribution function* (BRDF), which is the fraction of energy incident on a surface point from one direction that is reflected in another direction [Nicodemus et al. 77]. If Φ_i is the incident flux from incoming direction $\Theta_i = (\theta_i, \phi_i)$ with inclination ϕ_i, having solid angle $d\Theta$, and Φ_o is the outgoing flux through the same solid angle in direction $\Theta_o = (\theta_o, \phi_o)$, then the bidirectional reflectivity is defined as:

$$\rho(\Theta_i, \Theta_o) = \frac{\Phi_o}{\Phi_i \cos\phi_i d\Theta}$$

where Θ_i and Θ_o are both directions in the upper hemisphere. Bidirectional transmissivity τ is defined similarly. Note that the bidirectional reflectivity distribution function, at its most general, is a function of four variables: the two dimensions of the input direction and two dimensions of the output direction: $\rho(\Theta_i, \Theta_o) = \rho(\theta_i, \phi_i, \theta_o, \phi_o)$. Helmholtz' reciprocity law says that the BRDF is symmetric: $\rho(\Theta_i, \Theta_o) = \rho(\Theta_o, \Theta_i)$.

Many materials have *isotropic* reflectivity, for which the azimuthal dependence is a function of the angle between incoming and outgoing azimuth only, reducing the dimension of the BRDF to three:

$$\rho(\theta_i, \phi_i, \theta_o, \phi_o) = \rho(\theta_o - \theta_i, \phi_i, \phi_o)$$

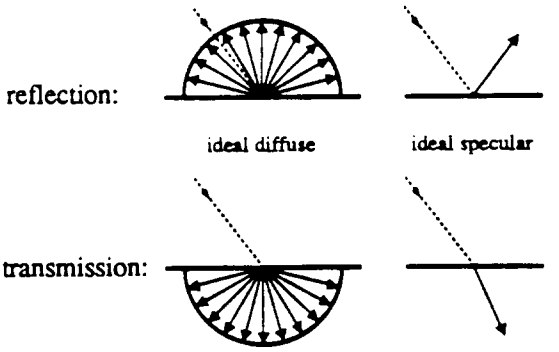

Figure 2.2: *Two idealized classes of reflectivity and transmissivity: ideal diffuse and ideal specular; showing a polar plot of intensity for fixed incoming direction and varying outgoing direction.*

The most important special cases are *ideal diffuse* or *Lambertian* materials, for which the outgoing intensity is direction-independent, and *ideal specular* materials, for which the outgoing intensity is limited to the solid angle of the incident radiation around the mirror direction (figure 2.2). Chalk is a good approximation of an ideal diffuse material, and polished metal is close to an ideal specular material. The BRDF of a diffuse material is constant, while that of a specular material is a delta function:

$$\rho_{\text{spec}}(\theta_i, \phi_i, \theta_o, \phi_o) = \delta(\theta_o - \theta_i - \pi)\delta(\phi_o - \phi_i)$$

Real materials have neither of these idealized reflectivity distributions. Brushed metal, for example, is anisotropic. Attempts have been made to approximate empirical BRDF's mathematically both in the thermal radiation [Torrance-Sparrow67] and computer graphics [Phong75,Blinn77,Cook-Torrance82] communities. Approximations good enough to make very realistic-*looking* images are possible by modeling a rough surface as a piecewise smooth surface with random microscopic bumps. Recent work has employed a more empirical approach, approximating empirical measurements of real BRDF's with spherical harmonics, for instance [He et al. 91].

The fraction of the incident intensity for a given incoming direction that is reflected anywhere can be calculated by integrating the bidirectional reflectivity over the hemisphere of outgoing directions, yielding the *directional hemispherical reflectivity*:

$$\rho_h(\Theta_i) = \int_{\text{hemi}} d\Theta_o \cos\phi_o \, \rho(\Theta_i, \Theta_o)$$

For a diffuse material, ρ is constant, and $\rho_h = \pi\rho$. Directional hemispherical transmissivity τ_h is defined analogously. The fraction of radiation from direction Θ_i that is absorbed is denoted $\alpha(\Theta_i)$. The fractions of the incident intensity with incoming direction Θ_i that are reflected, transmitted, and absorbed are therefore $\rho_h(\Theta_i)$, $\tau_h(\Theta_i)$, and $\alpha(\Theta_i)$, respectively. By conservation of energy, they sum to one at each position and in each direction: $\rho_h(\Theta_i) + \tau_h(\Theta_i) + \alpha(\Theta_i) = 1$. Since radiation intensity is nonnegative, each of these coefficients must be greater than or equal to zero. In practice, zero reflectivity, transmissivity, and absorptivity are never achieved, so $0 < \rho_h, \tau_h, \alpha < 1$. Because of dirt and other factors, it

is uncommon to find reflectivities above .85 [Ward90]. By Kirchhoff's laws, the absorptivity and emissivity for a given wavelength, direction, and polarization state are equal [Planck14]:

$$\alpha(\Theta) = \epsilon(\Theta)$$

2.2.3 Integral Equation for General, Opaque Surfaces

Simulation of radiation in general scenes is one of the fundamental problems of the field of thermal radiation. We follow Özisik's presentation of the mathematics of thermal radiation before discussing approximations [Özisik73]. This is a departure from most of the thermal radiation literature, which typically presents the radiosity method before discussion of integral equations.

Simulation of general scenes containing arbitrary reflectivity distributions and participating media is very difficult, because the intensity distributions for such scenes are functions of five variables, in general: three for position, and two for direction. Without participating media, the dimension is reduced to four, but such functions are still quite difficult to analyze.

Özisik derives the integral equation that defines intensity in a scene with non-participating media and opaque surfaces with general reflectivity [Özisik73]. We now make explicit the position-dependence of the surface properties. For an opaque surface, the outgoing intensity i_o at position $\mathbf{x}$ in direction Θ_o equals the emitted intensity i_{emit}, plus the reflected intensity i_{refl}:

$$i_o(\mathbf{x}, \Theta_o) = i_{\text{emit}}(\mathbf{x}, \Theta_o) + i_{\text{refl}}(\mathbf{x}, \Theta_o)$$

For a surface with temperature distribution $T(\mathbf{x})$, the emitted intensity is determined by equation (2.8):

$$i_{\text{emit}}(\mathbf{x}, \Theta_o) = \frac{\epsilon(\mathbf{x}, \Theta_o)\sigma}{\pi} T^4(\mathbf{x})$$

The reflected intensity in a given outgoing direction is the integral of the bidirectional reflectivity times the incident intensity over all incoming directions:

$$i_{\text{refl}}(\mathbf{x}, \Theta_o) = \int_{\text{hemi}} d\Theta_i \, cos\phi_i \, \rho(\mathbf{x}, \Theta_i, \Theta_o) \, i_i(\mathbf{x}, \Theta_i) \tag{2.9}$$

So the outgoing intensity function is $i_o = i_{\text{emit}} + i_{\text{refl}}$ [Özisik73, eq. 4-1]:

$$i_o(\mathbf{x}, \Theta_o) = \frac{\epsilon(\mathbf{x}, \Theta_o)\sigma T^4(\mathbf{x})}{\pi} + \int_{\text{hemi}} d\Theta_i \, cos\phi_i \, \rho(\mathbf{x}, \Theta_i, \Theta_o) \, i_i(\mathbf{x}, \Theta_i) \tag{2.10}$$

In a non-participating medium, intensity is not attenuated with distance, so if the first surface point hit by the ray from point $\mathbf{x}$ in direction Θ_i is $\mathbf{x}'$, and the azimuth and inclination of this ray are $\Theta'_o = (\theta'_o, \phi'_o)$ in the local coordinate system of point $\mathbf{x}'$, then the incoming intensity at $\mathbf{x}$ and the outgoing intensity at $\mathbf{x}'$, in this common direction are equal (figure 2.3):

$$i_i(\mathbf{x}, \Theta_i) = i_o(\mathbf{x}', \Theta'_o)$$

By changing the variable of integration from incoming direction Θ_i to surface position $\mathbf{x}'$, we can eliminate the incoming intensity function, leaving the outgoing intensity function as the only unknown. To do this we find the solid angle $d\Theta_i$ subtended by a surface element

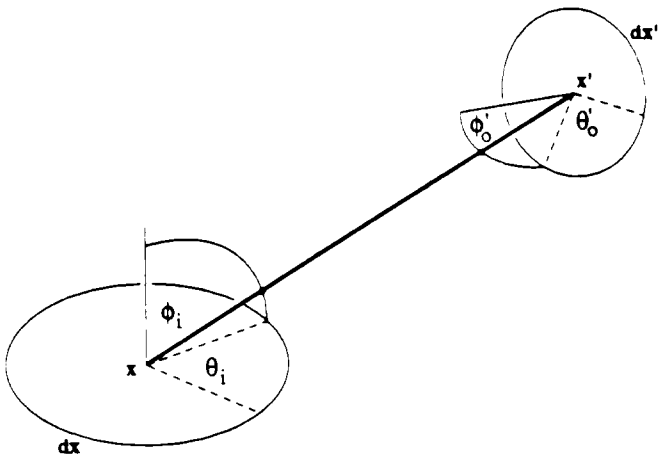

Figure 2.3: *Geometry of thermal radiation leaving point* $\mathbf{x}'$ *and arriving at point* $\mathbf{x}$.

with area $d\mathbf{x}'$ at a distance r and angle ϕ'_o to the normal at $\mathbf{x}'$. The solid angle of $d\mathbf{x}'$ at distance r equals this area projected onto a sphere of radius r, divided by r^2, so:

$$d\Theta_i = \frac{d\mathbf{x}' \cos\phi'_o}{r^2}$$

Therefore, the integral equation governing thermal radiation in a scene with opaque surfaces with general emissivity and reflectivity in a non-participating medium is[2]:

$$i_o(\mathbf{x}, \Theta_o) = \frac{\epsilon(\mathbf{x}, \Theta_o)\sigma}{\pi} T^4(\mathbf{x}) + \int_\Gamma d\mathbf{x}' \frac{\cos\phi_i \cos\phi'_o}{r^2} v\, \rho(\mathbf{x}, \Theta_i, \Theta_o)\, i_o(\mathbf{x}', \Theta'_o) \qquad (2.11)$$

where v is the *visibility function*, which equals 1 if $\mathbf{x}$ and $\mathbf{x}'$ are inter-visible, and 0 if they are occluded from each other's view; and Γ is the set of all surfaces in the scene. The variables Θ_i, Θ'_o, r, and v are functions of $\mathbf{x}$ and $\mathbf{x}'$. This is a Fredholm integral equation of the second kind. The only unknown in this equation is the outgoing intensity i_o.

2.2.4 Integral Equation for Diffuse, Opaque Surfaces

The problem simplifies significantly if we assume that all surfaces are diffuse emitters and reflectors. Then emissivity and reflectivity are functions of position only: $\epsilon(\mathbf{x}, \Theta_o) = \epsilon_h(\mathbf{x})$, $\rho(x, \Theta_i, \Theta_o) = \rho_h(\mathbf{x})/\pi$. With diffuse reflectivity, the intensity of outgoing radiation is independent of the directional distribution of incident radiation: $i_o(\mathbf{x}, \Theta) = i_o(\mathbf{x})$. Put another way, diffuse reflectivity obliterates the history of the incident radiation [Sparrow-Cess78]. In a non-participating medium, intensity is then a function of two-dimensional surface position only. The integral equation governing diffuse, opaque surfaces in a non-participating medium is:

$$i_o(\mathbf{x}) = \frac{\epsilon_h(\mathbf{x})\sigma T^4(\mathbf{x})}{\pi} + \rho_h(\mathbf{x}) \int_\Gamma d\mathbf{x}' \frac{\cos\phi_i \cos\phi'_o}{\pi r^2} v\, i_o(\mathbf{x}')$$

[2] a generalization of Özisik's equations 4-1 and 5-1.

The sum of emitted and reflected radiation over the hemisphere of directions at a point $\mathbf{x}$ is called the *radiosity*[3] $u(\mathbf{x})$. Because surfaces are assumed diffuse, $u(\mathbf{x}) = \pi i_o(\mathbf{x})$, and

$$u(\mathbf{x}) = \epsilon_h(\mathbf{x})\sigma T^4(\mathbf{x}) + \rho_h(\mathbf{x}) \int_\Gamma d\mathbf{x}' \frac{\cos\phi_i \cos\phi_o'}{\pi r^2} v\, u(\mathbf{x}') \tag{2.12}$$

Essentially equivalent integral equations are given in [Sparrow-Haji-Sheikh65], [Hottel-Sarofim67, eq. 3-6a], [Özisik73, eq. 5-1].

A number of methods have been used for solving the integral equations of thermal radiation. The most popular of these is the radiosity method.

2.2.5 Radiosity Method

The *radiosity method* for simulating thermal radiation in scenes of diffuse, opaque surfaces was developed by Hottel, Eckert, and Gebhart [Hottel54,Eckert-Drake59,Gebhart61]. These three formulations are shown to be essentially equivalent by Sparrow [Sparrow63]. The radiosity method has also been called the "zone method" [Hottel-Sarofim67], "zone analysis" [Özisik73], and the "lumped sum method" [Love68].

The radiosity method makes the following assumptions [Sparrow63]:

(1) Media are non-participating.

(2) Emission and reflection are diffuse.

(3) The *zones* into which the surfaces are divided are gray, having wavelength-independent emissivity and reflectivity.

(4) Each zone is *isothermal*, that is, each zone has constant temperature.

(5) The incident intensity (and hence also the outgoing intensity from both emission and reflection) is constant within each zone. This condition is rarely met, in practice.

If the emissivity or reflectivity is not wavelength-independent, then the wavelength band can be subdivided, and if the temperature or intensity of a zone is not constant, then the zone can be subdivided. These remedies will improve the approximation.

Following [Özisik73], we now derive the radiosity method from the integral equation for diffuse interreflection, equation (2.12). We assume the surfaces have been subdivided into n zones Γ_i (or, in finite element parlance, "elements") in this way. By these assumptions, for $\mathbf{x}$ in zone i we can let the temperature be T_i, the hemispherical emissivity can be ϵ_i, the reflectivity can be ρ_i, the radiosity can be u_i, and, the emissive power can be $e_i = \epsilon_i \sigma T_i^4$. With the radiosity function assumed constant within each zone, the integral is no longer a function of radiosity, but only of geometry:

$$u_i = e_i + \rho_i \sum_{j=1}^{n} u_j \int_{\Gamma_j} d\mathbf{x}' \frac{\cos\phi_i \cos\phi_o'}{\pi r^2} v$$

[3] The word "radiosity" was coined by Parry Moon [Moon36]. Moon invented a number of other radiometric terms such as "pharosage" and "helios" that never caught on. Though the word is popular today, at least in computer graphics, "radiosity" was called "an undesirable word" by Hoyt Hottel [Hottel-Sarofim67, p. 74], one of the people who in the 50's invented the discretization technique that later became known as the "radiosity method" [Sparrow63].

Integrating these expressions over zone i, we have

$$A_i u_i = A_i e_i + \rho_i \sum_{j=1}^{n} u_j \int_{\Gamma_i} dx \int_{\Gamma_j} dx' \frac{\cos\phi_i \cos\phi_o'}{\pi r^2} v \tag{2.13}$$

where A_i is the area of zone i. If we define the *form factor* F_{ij} between zones i and j to be the fraction of the radiative power leaving zone i in all directions that strikes zone j directly, then

$$F_{ij} = \frac{1}{A_i} \int_{\Gamma_i} dx \int_{\Gamma_j} dx' \frac{\cos\phi_i \cos\phi_o'}{\pi r^2} v \tag{2.14}$$

Using form factors, equation (2.13) can be rewritten concisely as

$$u_i = e_i + \rho_i \sum_{j=1}^{n} u_j F_{ij}$$

which is a system of n equations in n unknowns u_i. These *radiosity equations* can be rewritten in matrix notation as $\mathbf{Au} = \mathbf{e}$ where $\mathbf{A}$ is an $n \times n$ matrix and $\mathbf{u}$ and $\mathbf{e}$ are n-element column vectors, where

$$A_{ij} = I_{ij} - \rho_i F_{ij}$$

and $\mathbf{I}$ is the identity matrix. The equations for a five-zone problem, for example, are thus:

$$\begin{pmatrix} 1-\rho_1F_{11} & -\rho_1F_{12} & -\rho_1F_{13} & -\rho_1F_{14} & -\rho_1F_{15} \\ -\rho_2F_{21} & 1-\rho_2F_{22} & -\rho_2F_{23} & -\rho_2F_{24} & -\rho_2F_{25} \\ -\rho_3F_{31} & -\rho_3F_{32} & 1-\rho_3F_{33} & -\rho_3F_{34} & -\rho_3F_{35} \\ -\rho_4F_{41} & -\rho_4F_{42} & -\rho_4F_{43} & 1-\rho_4F_{44} & -\rho_4F_{45} \\ -\rho_5F_{51} & -\rho_5F_{52} & -\rho_5F_{53} & -\rho_5F_{54} & 1-\rho_5F_{55} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{pmatrix}$$

Before computers, only a handful of zones could be conveniently handled, but using current hardware and efficient algorithms for solving linear systems, thousands or millions of zones can be accommodated.

It is also possible to "compute" the radiosity solution using analog electrical circuits. A circuit whose nodes correspond to zones, with resistance between nodes inversely proportional to the form factor between zones, and appropriate voltages applied at emissive nodes, will have voltage at each node corresponding to radiosity, and current between nodes proportional to heat flux [Paschkis36,O'Brien55,Oppenheim56,Eckert-Drake72]. This network analog was used extensively for simulations before the advent of computers.

Alternate Derivation

It is also possible to derive the radiosity equations without reference to integral equations, as many early papers and current textbooks in thermal radiation do [Hottel54,Siegel-Howell81], but such a derivation can cloud one's understanding of the properties of the solution functions.

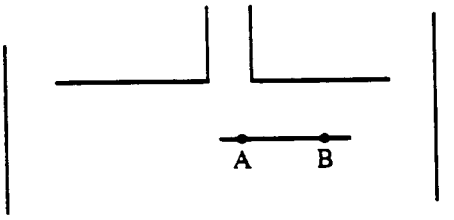

Figure 2.4: *Point A's upper hemisphere is not totally covered, but point B's upper hemisphere is covered.*

Form Factors

The form factor is a purely geometric, dimensionless quantity. It is independent of emissivity, reflectivity, radiosity, and other surface properties. A number of synonyms for "form factor" have been used: geometric configuration factor, view factor, angle factor, and shape factor. We use the term "form factor", which is most common in the computer graphics literature.

A number of form factor properties are evident. If zone i does not see zone j (because of occlusion, or because one is in the back halfspace of the other), then $F_{ij} = 0$. If zone i cannot see itself (i.e. if the object is locally convex) then $F_{ii} = 0$. The following reciprocity relation can be derived from equation (2.14):

$$A_i F_{ij} = A_j F_{ji}$$

In the thermal radiation literature, a 3-D scene is typically called an *enclosure*. An enclosure is a volume bounded by real surfaces of known emissivity, reflectivity, and temperature; and by imaginary surfaces (filling windows and holes, for instance) over which the entering radiation is known [Love68]. When the entire hemisphere above zone i is covered with objects, then the sum of the form factors from each zone to every other zone equals 1: $\sum_{j=1}^{n} F_{ij} = 1$, but when part of the hemisphere is not covered from zone i (when it can see "outer space", as at point A of figure 2.4), the sum of its form factors is less than 1: $\sum_{j=1}^{n} F_{ij} < 1$. Unfortunately, use of the word "enclosure" has led to the misconception by some that the radiosity method is only applicable in a closed space. This is not so. As long as the five conditions above are met, the simulation is valid.

The calculation of form factors has always been the most difficult step in the radiosity method [Love68]. In theory, one could always determine a form factor from its definition in equation (2.14), but in practice, the form factor between two surfaces involves two double-integrations, and these integrals are often intractable analytically. There are several methods for calculating form factors [Love68]:

Published Tables. Tables have been collected of form factors between rectangles, circular disks, spheres, cylinders, and other shapes in a variety of configurations. See, for example, [Siegel-Howell81]).

Form Factor Algebra. There are a number of clever techniques by which one can extend the coverage of a form factor table. For example, the form factor between

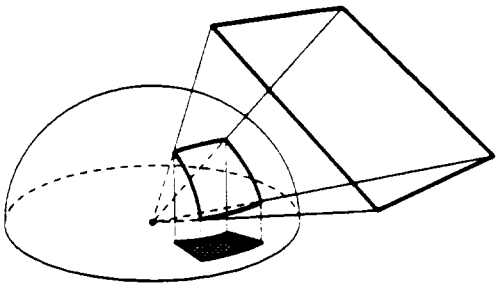

Figure 2.5: *Nusselt's analog for computing form factors.*

zone i and the zone formed by the union of two zones j and k, denoted $F_{i,j\cup k}$ can be found using simple inclusion-exclusion logic: $F_{i,j\cup k} = F_{i,j} + F_{i,k} - F_{i,j\cap k}$. Many more tricks are possible [Özisik73,Siegel-Howell81].

Graphical, Mechanical, and Optical Methods. The form factor from a point i (or infinitesimal zone) to another zone j can be computed by projecting zone j onto a hemisphere centered on point i, then parallel-projecting this region on the sphere down onto the tangent plane at i. The form factor equals the fraction of the circular base of the hemisphere covered by the projection (figure 2.5). This is called the Nusselt analog [Nusselt28]. This geometrical analogy has been used as the basis for several graphical, mechanical, and optical form factor calculation methods [Jakob57,Eckert-Drake72].

Numerical Integration. The integrals of (2.14) can also be approximated by numerical quadrature methods such as the trapezoid rule, Simpson's rule, or Gaussian quadrature [Ralston-Rabinowitz78]. Such methods became much more practical with the advent of computers.

Stokes Theorem. The area integrals of equation (2.14) can be transformed into contour integrals using Stokes' theorem [Moon36]. This often allows analytic formulas to be found more easily.

Symbolic Integration. The method of last resort is symbolic integration. It is feasible only for the simplest geometries.

Zoning

The accuracy of the radiosity method is an important and difficult issue. The fifth assumption of the radiosity method is that radiosity is constant within each zone. This assumption is unrealistic, and it causes great difficulties. As stated by Hottel, however, the method "can be used to make allowance for any degree of complexity of an enclosure and to approach the true solution to any degree of approximation dependent on the number of zones into which a surface is divided. The guiding principle in deciding upon the number of zones necessary is that any reradiation or reflection must come from a zone small enough so that different

parts of its surface do not have a significantly different view of the various other surfaces" [Hottel54]. Many of the adaptive meshing methods discussed in this thesis can be viewed as solutions to the problems caused by the assumption of constant radiosity.

2.2.6 Other Methods

Aside from the radiosity method, the other techniques for solving the integral equations of thermal radiation include iterative methods, kernel approximation methods, global approximation methods, finite element methods, and Monte Carlo methods.

Iterative Method

The second method for solution of the integral equations is the iterative technique that approximates the Neumann series. As discussed in section §2.1.1, an initial approximation $u^{(0)}(\mathbf{x})$ to the solution function is chosen, and this is iterated by applying the integral operator $\mathcal{K}$:

$$u^{(i)} = e + \mathcal{K}u^{(i-1)}$$

The integrals required to compute $u^{(1)}$ are often tractable analytically [Jensen48], but the integrals of the second iteration or beyond must often be calculated numerically [Sparrow et al. 61].

Kernel Approximation Method

If the kernel of an integral equation can be approximated well by n terms of the series

$$\kappa(s,t) = \sum_{i=1}^{n} a_i e^{b_i|s-t|}$$

then the integral equation can be transformed into a differential equation of order $2n$ [Sparrow-Cess78]. This *exponential kernel method* has been used for simulating thermal radiation in a long cylinder.

Global Approximation Method

Global approximation methods employ the Rayleigh-Ritz variational approach described in §2.1.3. They approximate the exact solution with a linear combination of basis functions each of global support. The integral equation for diffuse scenes, equation (2.12) can be solved approximately by that method if we define

$$e(\mathbf{x}) = \epsilon_h(\mathbf{x})\sigma T^4(\mathbf{x})$$

$$\kappa(\mathbf{x},\mathbf{x}') = \rho_h(\mathbf{x})\frac{\cos\phi_i \cos\phi_o'}{\pi r^2} v$$

$$(\mathcal{K}u)(\mathbf{x}) = \int_\Gamma d\mathbf{x}'\, \kappa(\mathbf{x},\mathbf{x}')u(\mathbf{x}')$$

This approach has been applied to thermal radiation problems by Sparrow [Sparrow60, Sparrow-Haji-Sheikh65].

Finite Element Method

Finite element methods approximate functions using a linear combination of basis functions of finite support. Collocation and Galerkin methods are two of many possible discretization techniques that are used by the finite element approach to approximate an integral or differential equation by a linear system of equations. The radiosity method, which was developed before finite element techniques came to maturity, can be viewed as a simple finite element method, a connection explored in chapter 3. More advanced methods for simulating the integral and integro-differential equations of thermal radiation in participating media have been explored [Chung88]. Other work has explored Galerkin techniques [Gwinner87] and isoparametric elements [Keavey88] for thermal radiation.

Monte Carlo Method

The final integral equation solution method discussed here is the Monte Carlo method. Monte Carlo methods can be explained by either physical analogy or by mathematical approximation. The physical analogy, alluded to earlier, is that radiation can be regarded as a multitude of photons ricocheting around a scene. The Monte Carlo method for thermal radiation approximates the statistics of the real scene by simulating the paths of a small fraction (only millions, say) of them. The mathematical derivation is to approximate the integrals of (2.11) with finite sums of samples chosen pseudorandomly. Monte Carlo methods are less accurate than most other methods, but in some integro-differential problems, and problems involving many dimensions (such as specular reflectivity), they are the only solution methods currently available.

Monte Carlo methods for thermal radiation are surveyed in [Siegel-Howell81]; they are applied to specular and transmissive materials in [Rushmeier-Torrance90] and to participating media in [Rushmeier87]. Monte Carlo methods have also been used for simulations in illuminating engineering [Tregenza83].

2.2.7 General Scenes

How complex can the scenes for thermal radiation simulation get? It is not hard to imagine 3-D scenes containing unusual objects that would defy simulation by all existing algorithms, and even defy description using existing thermal radiation models.

For example, imagine a scene consisting of objects with fractal geometry moving at near light speed through a foggy, participating gas with wavelength-dependent emission, absorption, and scattering. Further imagine that the objects are made of a variety of translucent materials with complex anisotropic bidirectional reflectivity and transmissivity distributions. These materials might exhibit all known properties: conduction (diffusion of heat), convection (fluid flow), absorption, scattering, polarization, diffraction, interference, birefringence (double refraction), dispersion (wavelength-dependent refraction), fluorescence (crosstalk between wavelengths), incandescence (emission due to heat), luminescence (emission due to chemical reaction, for example), or phosphorescence (time-delayed emission).

Models exist for simulating most of these effects individually, but they cannot be simulated in combination at present. Future improvements in mathematical models and algorithms should bring these simulations within reach.

2.3 Computer Vision

Computer vision and image synthesis solve problems that are inverses of one another: whereas image synthesis computes intensities from geometry, computer vision computes geometry from intensities.

Global illumination complicates the shape-from-shading problem [Horn-Brooks89]. Because of interreflection, shading is not a function of just surface orientation, or even of just orientation and position, but is dependent on nearby objects in the scene [Koenderink-van Doorn83,Forsyth-Zisserman89]. Forsyth and Zisserman made the simple observation that global illumination is most significant between white surface (ρ_h near 1) and weakest between black surfaces (ρ_h near 0). Also, they noted that radiosity typically rises at reflex corners between diffuse surfaces (where touching surfaces face each other), and the peak is more pronounced as reflectivity increases. Such peaks are visible in the corners of a room.

2.4 Shadow Algorithms

In computer graphics, once geometry of a scene has been specified, a *rendering* algorithm is used to generate a picture. Rendering consists of two tasks: visibility determination (formerly known as "hidden surface elimination") [Sutherland et al. 74] and *shading* (the computation of intensities).

The simplest shading algorithms simulate direct illumination from point light sources without regard for occlusion. The next improvement is the simulation of shadows. We study shadow algorithms because they simulate a first approximation to global illumination, and because shadows have traditionally been difficult to resolve well with radiosity and ray tracing algorithms, which are the algorithms most commonly used to simulate global illumination. With radiosity algorithms, hard shadows often look jagged, and with ray tracing algorithms, soft shadows are typically either nonexistent or noisy. In real life, shadows from point light sources are sharp, and shadows from area light sources are soft; they have both a partial shadow *penumbra* region and a total shadow *umbra* region.

Shadows have been simulated in several ways (see survey [Woo et al. 90]). One of the oldest methods is the use of shadow volumes [Crow77]. As a preprocess to rendering, the volumes of space that are in shadow with respect to each point light source are determined, then during pixel generation (or *scan conversion*), each surface point is tested for inclusion against each of these volumes to see if it is in shadow.

Another approach is ray tracing, in which a ray is traced from each surface point toward each light to test for occlusion [Whitted80]. With a ray tracing algorithm it is possible to simulate penumbras by stochastically tracing rays to a random sampling of points on the light source, but the resulting image tends to be noisy unless an immense number of rays are traced [Cook et al. 84]. An alternative to tracing a number of rays of infinitesimal thickness is to trace a single cone of finite thickness. Such *cone tracing* algorithms have been used to simulate penumbras, but unfortunately the technique is generally limited to polygons and spheres [Amanatides84]. Ray tracing is discussed further in the next section.

A third method involves visible surface determination from the point of view of the light sources. This approach is attractive because of the number and generality of visibility algorithms in computer graphics and computational geometry. One can use either an *image space* approach or an *object space* approach [Sutherland et al. 74]. The former discretely

samples the visible surfaces analogous to a frame buffer, while the latter determines the visible portions continuously.

Image space shadow algorithms generate z-buffer images of the scene from the point of view of each light source, and use them while rendering from the point of view of the viewer's *eye* to test if visible points are in shadow [Williams78,Reeves et al. 87]. These shadow algorithms are more flexible than most, since they are not limited to polygons, but they are difficult to tune. Choosing the resolution for the light images is critical, since aliasing of shadow edges results if the light images are too coarse. Z-buffer shadow algorithms can simulate penumbras by rendering the scene from a number of points on each area light source, but this is extremely slow [Brotman-Badler84].

Object space shadow algorithms generate *surface detail polygons*, modifying the scene description by splitting all polygons into shadowed and unshadowed portions that are shaded appropriately in the final rendering from the eye. The geometry polygons and surface detail polygons can be represented using various data structures, including straightforward concave polygons with holes [Atherton et al. 78] or binary space partitioning trees of convex polygons [Chin-Feiner90].

An object space approach has been used very successfully by Nishita to generate penumbras from convex polygonal light sources [Nishita-Nakamae83]. Nishita's algorithm determines the polygonal umbra and penumbra regions as a pre-process, then shades them appropriately during scan conversion. Even without simulating interreflection, his algorithms produced some extremely realistic-looking images. Campbell has recently extended this technique by doing more of the shading computations in object space [Campbell-Fussell91].

2.5 Global Illumination Algorithms

The fields of global illumination and thermal radiation study many of the same phenomena, but the applications and cultures of the two communities differ.

In global illumination, only visible light is relevant, not infrared. Typically three wavelength samples are chosen, corresponding to the red, green, and blue phosphors of most cathode ray tubes (and, roughly speaking, to the long, medium, and short wavelength cones in the human retina). Four or more wavelength samples can be used for more accurate spectral approximation [Meyer88], but the differences in the results are subtle. Standard RGB samples are used here.

Also, in global illumination we are typically simulating habitable interiors, not furnaces or other incandescing materials, so temperature differences and the attendant conduction and convection phenomena are of no concern. The emission of light sources is not parameterized by temperature, as in the Stefan-Boltzmann law, but is treated as a black box function $i_{\text{emit}}(\mathbf{x}, \Theta_o)$.

2.5.1 Realistic Image Synthesis

Standards have advanced dramatically in the scant thirty-odd years that computer graphics has existed: early researchers were happy to make any almost any picture at all, while recent efforts have pushed the pursuit of perceptual reality, or *realistic image synthesis*, to the point that viewers sometimes cannot distinguish the real from the simulated [Meyer et al. 86].

While much work in realistic image synthesis is devoted toward modeling the shape and motion of real objects, global illumination research is concerned with realistic shading. In an

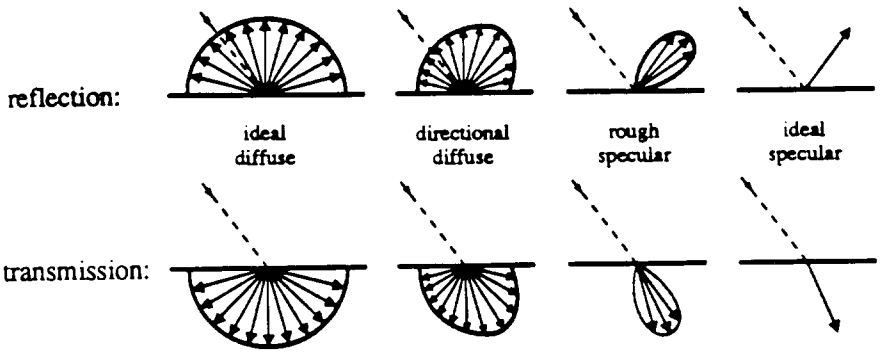

Figure 2.6: *Four classes of reflectivity and transmissivity: ideal diffuse, directional diffuse, rough specular, and ideal specular; showing a polar plot of intensity for fixed incoming direction and varying outgoing direction.*

architectural design application, "realistic" might mean an objectively, physically accurate light intensity in watts per square meter per steradian, but in a computer art application, "realistic" might mean perceptually, subjectively indistinguishable from reality. In this work we favor the former definition of realism, because it is simpler, more quantifiable, and does not depend on as-yet-undiscovered human visual models.

2.5.2 General Reflectivity and Transmissivity

In addition to ideal diffuse and ideal specular reflectivity, discussed in §2.2.2, it is helpful to define two additional classes. We define *directional diffuse* to be reflection that is a smooth but non-constant function of direction [Sillion et al. 91], and *rough specular* to be reflection to a finite number of cones [Heckbert90] (figure 2.6). (These two classes overlap, and their definitions are intentionally vague, in order that they and the two ideal classes will cover all possible BRDF's.) Directional diffuse and rough specular transmissivity are defined analogously.

An ideal diffuse surface has equal intensity from all viewing directions, but a general surface's intensity varies with viewing direction, so we say that ideal diffuse reflection is *view-independent* while general reflection is *view-dependent*. For computer graphics purposes, the simplest materials have a position-invariant, isotropic BRDF consisting of a linear combination of ideal diffuse and ideal specular reflection, but a fully-general BRDF can be position-dependent and simulate textured, anisotropic, directional diffuse or rough specular surfaces.

We define *scattering* to mean either reflection or transmission. The BRDF ρ and BTDF τ can be regarded as two halves of a *bidirectional scattering distribution function* (BSDF) μ:

$$\mu(\Theta_i, \Theta_o) = \rho(\Theta_i, \Theta_o) + \tau(\Theta_i, \Theta_o)$$

2.5.3 The Global Illumination Equation

With this notation, as a generalization of equation (2.9), the scattered intensity is

$$i_{\text{scat}}(\mathbf{x}, \Theta_o) = \int_{\text{sphere}} d\Theta_i \cos\phi_i \, \mu(\mathbf{x}, \Theta_i, \Theta_o) \, i_i(\mathbf{x}, \Theta_i)$$

Note that only one integral over a sphere of directions is needed above, but if the scattering function were split into reflective and transmissive parts, an integral over each of the two hemispheres would be necessary – a more cumbersome notation.

Similar to equation (2.11), global illumination between surfaces with arbitrary scattering functions in a non-participating medium is governed by the integral equation

$$i_o(\mathbf{x}, \Theta_o) = i_{\text{emit}}(\mathbf{x}, \Theta_o) + \int_\Gamma d\mathbf{x}' \frac{\cos\phi_i \cos\phi_o'}{r^2} \, v \, \mu(\mathbf{x}, \Theta_i, \Theta_o) \, i_o(\mathbf{x}', \Theta_o') \qquad (2.15)$$

We call this the *global illumination equation* for non-participating media. The unknown here, as before, is the outgoing intensity i_o. The term i_{emit} here could be due to incandescence, as in thermal radiation, or some other physical mechanism. In global illumination simulations, we typically do not care.

Integral equations similar to this have appeared in the thermal radiation literature [Özisik73], in illuminating engineering [Moon36], in the field of neutron transport in physics, where it is called the *Boltzmann equation* [Lewis-Miller84], in computer vision [Koenderink-van Doorn83], where it has been called the *mutual illumination equation* [Forsyth-Zisserman89], and in computer graphics, where it has been called the *rendering equation* [Kajiya86,Immel et al. 86,Bouville et al. 90].

When the emitted intensity and the reflectivity are diffuse, and surfaces are opaque, we substitute hemispherical quantities: the emissive power $e(\mathbf{x}) = \pi i_{\text{emit}}(\mathbf{x}, \Theta_o)$, the hemispherical emissivity $\rho_h(\mathbf{x}) = \pi\rho(\mathbf{x}, \Theta_i, \Theta_o)$, and the radiosity $u(\mathbf{x}) = \pi i_o(\mathbf{x}, \Theta_o)$. This yields an integral equation similar to equation (2.12):

$$u(\mathbf{x}) = e(\mathbf{x}) + \rho_h(\mathbf{x}) \int_\Gamma d\mathbf{x}' \frac{\cos\phi_i \cos\phi_o'}{\pi r^2} \, v \, u(\mathbf{x}')$$

The kernel of this integral equation in a specular scene is non-smooth and sparse (zero almost everywhere), while the kernel in a diffuse scene with no occlusions is smooth and dense (nonzero almost everywhere). Global illumination algorithms can be characterized by the approximations they make to the global illumination equation. In diffuse scenes, one can exploit the smoothness of the kernel using radiosity algorithms, and in specular scenes, one can exploit the sparseness of the kernel using ray tracing algorithms.

2.5.4 Ray Tracing Algorithms

Classic ray tracing employs a recursive *trace* procedure which, given a ray and a scene, returns the intensity of light arriving at the origin of the ray in the direction opposite to the ray [Whitted80]. The procedure finds the first surface intersected by the ray, and recursively traces rays in the specularly reflected and transmitted directions, if any, to determine the light that comes via those paths (figure 2.7). Also, *shadow rays* are traced toward designated point light sources to test for occlusion of direct lighting. The intensity returned by the procedure is a linear combination of the intensities from these directions. To generate a

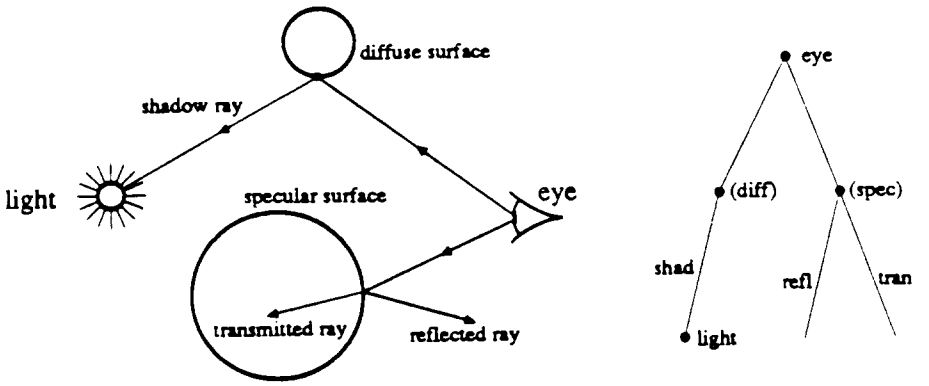

Figure 2.7: *Tree of rays traced by a classic ray tracing algorithm shown both in the scene (left) and schematically (right).*

picture one simply traces rays from the viewer's *eye* through each pixel of an imaginary projection screen using *trace* to calculate pixel intensities. Non-recursive variants of the ray tracing algorithm date back to the 1960's [Appel68,Goldstein-Nagel71]. Good tutorial material on ray tracing can be found in [Glassner89].

One of the principal advantages of ray tracing as a shading and visible surface algorithm is its flexibility with a variety of geometric primitives. The elementary operations on geometric shapes needed by a ray tracer are simple: read object from file, intersect with a ray, determine surface normal, and look up surface properties. Ray tracing software can be structured in an object oriented way so that the core of the algorithm (the *trace* routine) is independent of the geometric primitives. Ray-object intersection algorithms have been developed for polygons, quadrics, cubic patches, implicit surfaces, fractals, blobby models, and other primitives (see [Glassner89] for survey). Other rendering algorithms, by comparison, are often limited to polygons or some other simple geometric primitive.

Intersection testing can be done by brute force testing of each shape in the scene, or it can be optimized using various spatial data structures. Using the right data structures, the number of objects tested for intersection against each ray can be reduced dramatically. Since intersection testing is the dominant cost of brute force ray tracing algorithms on complex scenes, spatial subdivision can provide a significant optimization. The three most commonly-used spatial subdivision schemes are hierarchically nested bounding volumes, uniform grids, and octrees. The choice of subdivision scheme is usually independent of shading algorithm, however, so spatial data structures for optimizing ray tracing are not a concern of this research.

In a classic ray tracing algorithm, rays are recursively traced in the (specularly) reflected and transmitted directions until the ray tree reaches some maximum depth or until the contribution of each ray drops below a threshold. The direction of the specularly reflected ray is simply the mirror direction of the incident ray, and the direction of the specularly transmitted ray is given by Snell's law, which bends the ray according to the index of refraction of the transparent material. Only in scenes with nearly perfect reflectors or transmitters (such as a scene between two parallel mirrors) will the ray tree depth required to make a fairly accurate picture exceed five or so.

Classic ray tracing assumes that the BSDF has an ideal specular component, but no

directional diffuse or rough specular component, and that the incident light that is diffusely scattered is a sum of delta functions in the direction of each light source. This latter assumption implies a local illumination model for diffuse scattering. Light coming from directions other than the light sources is not scattered diffusely in a classic ray tracing algorithm.

Distribution Ray Tracing

A more realistic illumination model would include directional diffuse and rough specular BSDF's and would compute diffuse scattering globally. Exact simulation of these effects requires the integration of incident light over cones of finite solid angle. While ray tracing cannot evaluate such integrals analytically, it can be generalized to numerically approximate such computations using *distribution ray tracing*[4] [Cook et al. 84,Lee et al. 85,Dippe-Wold85,Cook86,Kajiya86,Mitchell87,Ward et al. 88,Arvo-Kirk90]. In distribution ray tracing, rays are distributed, either uniformly or in a Monte Carlo (stochastic) manner, throughout any distributions needing integration. The method was initially developed as an antialiasing method, but it was soon realized that distributing rays over various distributions was useful for the simulation of a number of optical phenomena in addition to antialiasing, including spatial antialiasing, temporal antialiasing, penumbras, rough specular reflection, and depth of field [Cook et al. 84]. Kajiya demonstrated that the method could be extended to diffuse interreflection [Kajiya86]. Many rays must be traced to accurately integrate the broad scattering distributions of rough specular and diffuse surfaces: often hundreds or thousands per surface intersection. The most successful implementation of distribution ray tracing to date is probably Ward's RADIANCE program [Ward et al. 88].

Distribution ray tracing generally employs stochastic (random) samples and Monte Carlo integration. Monte Carlo integration works on a very general class of non-continuous functions, but unfortunately, it converges slowly. The error of Monte Carlo integration with n samples is $O(n^{-1/2})$, which is large when compared to the errors of most other numerical integration methods. On suitably continuous functions, the error of the rectangle rule is $O(n^{-1})$, the trapezoid rule is $O(n^{-2})$, and Simpson's rule is $O(n^{-4})$. These fast integration methods could be used if continuity of the integrand were guaranteed, but this is not possible with most ray tracing algorithms, since they provide only point samples, and no information about the intensities of rays in nearby directions.

Alternatives to ray tracing that are better suited to integration over solid angles are cone tracing [Amanatides84], *beam tracing* [Heckbert-Hanrahan84], and *pencil tracing* [Shinya et al. 87]. The first two algorithms integrate over circular cones and polygonal cones, respectively, but as mentioned earlier they are currently limited to simple geometry. The third variation, pencil tracing, holds more promise as a general technique because it only requires objects to be intersected with rays, not cones or beams. A *pencil*, in this context, is a ray with an associated coordinate system that gives information about the direction of nearby rays. This pencil is updated as a ray reflects or refracts off surfaces to account for convergence and divergence of rays. When the ray on the axis of the pencil hits a surface, the pencil's coordinate system allows the approximation of the region of intersection of the cone with the surface, facilitating integration. *Triangle tracing* is a simple hybrid of beam

[4] We proposed the name "distribution ray tracing" as an alternative to the more common name, "distributed ray tracing", which is confusing because of its parallel hardware connotations [Heckbert90].

tracing and pencil tracing ideas: three rays are traced at the corners of a triangular beam; it facilitates integration over a triangle [Strauss88].

Light Ray Tracing

Rays are traced from the eye in classic ray tracing, but it can also be useful to trace rays from the lights. Rays traced from the eye we call *eye rays* and rays traced from the lights we call *light rays*.[5]

Light ray tracing was originally proposed by Appel [Appel68], who "stored" his radiosities on paper with a plotter. Light ray tracing with polygonal beams was proposed in previous work with Hanrahan [Heckbert-Hanrahan84] where radiosities were stored as surface detail polygons like [Atherton et al. 78]. This approach was modified by Strauss, who deposited light directly in screen pixels when a diffuse surface was hit by a beam, rather than store the radiosities with the surface [Strauss88]. Watt has recently used light beam tracing to simulate refraction at water surfaces [Watt90]. Refraction of light by gems and curved surfaces has also been simulated using pencil tracing [Shinya et al. 89]. Arvo used light ray tracing to compute his radiosity textures [Arvo86], as did Shirley [Shirley90]. Overall, A combination of light ray tracing and eye ray tracing is pursued in chapter 5.

2.5.5 Radiosity Algorithms

As mentioned in §2.2, the term *radiosity* is used in two senses. First, radiosity is the energy flux from emission, reflection, and transmission leaving a diffuse surface, per unit time and area (or roughly speaking, the surface's brightness), and second, radiosity is a thermal radiation approximation method and global illumination algorithm. The meaning should be clear by context. Whereas ray tracing algorithms typically generate a picture from a particular viewpoint, radiosity algorithms compute the intensity function on all surfaces in the scene, after which a picture can be generated.

The radiosity method from the thermal radiation field [Hottel54,Sparrow63] was adapted to generate pictures by Torrance, Goral, and Nishita [Goral et al. 84,Nishita-Nakamae85]. Analogous to thermal radiation methods, the *classic radiosity* algorithm [Cohen-Greenberg85] subdivides each surface into polygons each of which is assumed to have constant emissivity, reflectivity, and radiosity. The algorithm then determines the fraction of energy diffusely radiated from each polygon to every other polygon: the pair's form factor. From the form factors, a large system of equations is constructed whose solution is the radiosities of each polygon. This system is then solved with either Gauss-Seidel iteration [Strang88] or with progressive techniques that compute the matrix a column at a time [Cohen et al. 88]. The system is solved for each of the wavelength bands (typically red, green, and blue), but form factors need be computed only once.

Other than the generation of pictures and the use of real-time hardware for display, the greatest contribution of computer graphics to the radiosity method has probably been the development of techniques for computing form factors in complex scenes. Form factors can be determined analytically for simple geometries [Siegel-Howell81,Baum et al. 89], but for complex geometries, occlusion makes the integrals difficult, so numerical integration methods are used, employing a visibility algorithm to test for occlusion. The most popular

[5] We avoid the terms "forward ray tracing" and "backward ray tracing" because they are ambiguous: some people consider photon motion "forward", while others consider Whitted's rays "forward".

technique is the *hemicube* method [Cohen-Greenberg85]. A hemicube is a half-cube placed at the center (or vertices) of each polygon. From the point of view of the hemicube's center, the scene is rendered onto its five faces using a z-buffer visibility algorithm. From the five resulting, low resolution raster images, one row or column of form factors of the form factor matrix **F** can be approximated. An advantage of the hemicube method over other form-factor computation methods is that much of the work can be done with the fast z-buffer hardware of graphics machines such as SGI's and HP's. Ray tracing has recently been promoted as an alternative to the hemicube method [Wallace et al. 89,Sillion-Puech89]. To the author's knowledge, object space visible surface algorithms such as [Weiler-Atherton77, Fuchs et al. 80] have never been applied to form factor calculation.

The output of the radiosity algorithm is one radiosity value per polygon. Since diffuse scattering is by definition view-independent, these radiosities are valid from any viewpoint. The radiosity computation is followed by a visibility algorithm to generate a picture. [6]

Classic radiosity assumes ideal diffuse scattering, so it does not simulate specular scattering at all. The radiosity method can be generalized to simulate specular scattering by storing not just a single radiosity value with each polygon, but a representation of the direction-dependence of the intensity function. Such a *directional radiosity* algorithm can simulate both diffuse and specular scattering globally. If the direction-dependence is stored as a two-dimensional array [Immel et al. 86,Shao et al. 88,Buckalew-Fussell89] then the memory requirements are excessive, but if smoother basis functions are used, such as spherical harmonics, then the method becomes more practical [Sillion et al. 91].

2.5.6 Hybrid Methods

Ray tracing is best at specular and radiosity is best at diffuse, and the above attempts to generalize ray tracing to diffuse and to generalize radiosity to specular stretch the algorithms beyond the light scattering realms for which each is best suited, making them less accurate and less efficient.

Another class of algorithms is formed by hybridizing the methods, using a two-pass approach that applies a radiosity pass followed by the ray tracing pass. Such algorithms have been explored by Wallace and Sillion.

The first pass of Wallace's algorithm consists of classic radiosity extended to include diffuse-to-diffuse scattering that bounces off planar mirrors [Wallace et al. 87]. He follows this with a classic ray tracing pass (implemented using a z-buffer). Unfortunately, the method is limited to planar surfaces (because of the polygonization involved in the radiosity algorithm) and to perfect planar mirrors.

Sillion's algorithm is like Wallace's but it computes its form factors using ray tracing instead of hemicubes [Sillion-Puech89]. This eliminates the restriction to planar mirrors. The method still suffers from the polygonization inherent in the radiosity step, however.

[6] In this sense, the radiosity algorithm is not a rendering algorithm, but just a shading algorithm. Classic radiosity algorithms simulate diffuse scenes only, so their output is a view-independent *pre-shaded scene*. Consequently it is easy to view these scenes with interactive camera animation (a "walkthrough") using fast graphics hardware. Unfortunately, this capability has led to the misconception that the radiosity algorithm is interactive, while ray tracing is not [Greenberg91]. In fact, when one is using a real-time display program to explore a pre-shaded, static scene, this scene could have been shaded using radiosity, ray tracing, or a number of other algorithms, with no difference in interactivity.

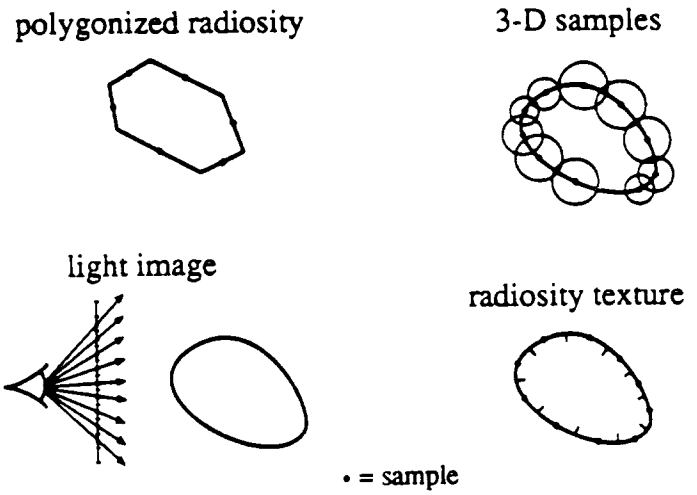

Figure 2.8: *Data structures for representing the radiosity function.*

2.5.7 Data Structures for the Radiosity Function

Sampling artifacts arise in both ray tracing and radiosity algorithms. Many of the sampling problems of ray tracing have been solved by adaptive algorithms [Whitted80,Lee et al. 85,Dippe-Wold85,Mitchell87,Painter-Sloan89], particularly for the simulation of specular scattering. The sampling problems of the radiosity algorithm are less well studied, probably because its sampling process is less explicit and more complex than that of ray tracing.

We examine four data structures for representing the radiosity function $u(\mathbf{x})$: light images, polygons, samples in 3-D, and textures (figure 2.8).

The computation of form factors and other integrals is a critical operation in radiosity algorithms, regardless of which data structure is used. Several algorithms have been used to compute these integrals: analytic methods, hemicubes at the receiver (called *gathering*), hemicubes at the sender (called *shooting*), and ray tracing from the eye or from the light.

Light Images

The simplest data structure, the *light image*, simulates only shadows, the first order effects of global illumination. Light images are pictures of the scene from the point of view of each light source. As discussed earlier, they can be generated using a z-buffer shadow algorithm. The choice of resolution of the light images is problematical.

Polygonized Radiosity

Polygonized radiosity is the representation of the radiosity function by polygons of constant radiosity. A simple example mentioned earlier is the surface detail polygons used by object space shadow algorithms. They are a first-approximation to the interreflection simulated by radiosity algorithms.

The most common method for computing polygonized radiosity is the classic radiosity algorithm. A major problem with this algorithm is that surfaces are polygonized before

radiosities are computed. Difficulties result if this polygonization is either too coarse or too fine.

Sharp shadow edges caused by small light sources can be undersampled if the polygonization is too coarse, resulting in blurring or aliasing of the true radiosity function. Cohen developed the "substructuring" technique in response to this problem [Cohen et al. 86]. It makes an initial pass computing radiosities at low resolution, then splits polygons that appear to be in high-variance regions and recomputes radiosities.[7] Substructuring helps, but it has not yet been fully automated, as the subdivision stopping criterion appears to be a polygon size selected in some ad hoc manner. The limitations of the method are further demonstrated by the absence to date of radiosity pictures in published work exhibiting sharp shadow edges.

The other extreme of radiosity problems is oversampling of radiosities due to polygonization that is too fine for the hemicube. The resulting quantization can be cured by adaptive subdivision of the hemicube or of the light rays [Wallace et al. 89,Baum et al. 89].

We conclude that polygonization criteria remain a difficult problem for the radiosity method.

It is interesting to note the similarities between radiosity algorithms and the Atherton-Weiler algorithm. Conceptually, the original radiosity method gathers light to each polygon by rendering the scene from the point of view of each receiver, but the progressive radiosity algorithm shoots light by rendering the scene from the point of view of each sender (a light source). A progressive radiosity algorithm is thus much like repeated application of the Atherton-Weiler shadow algorithm. Campbell has recently implemented a radiosity algorithm that employs this approach [Campbell-Fussell90].

Samples in 3-D

Radiosities can be computed using brute force distribution ray tracing [Kajiya86], but the method is inefficient because it samples the slowly-varying radiosity function densely. To exploit the coherence of radiosity values, Ward sampled the diffuse component sparsely, and saved this information in a world space octree [Ward et al. 88], leading to great time savings. A slight problem with this algorithm is that it can easily miss light reflected by specular surfaces onto diffuse surfaces, since it shoots rays from the eye toward the lights, and not vice-versa.

Radiosity Texture

The fourth data structure for radiosities is the *radiosity texture*. Instead of polygonizing each surface and storing one radiosity value per polygon, radiosity samples are stored in a texture on every diffuse surface in the scene [Arvo86]. A texture is a function that is mapped onto a surface and used as a shading parameter in some way [Blinn-Newell76,Heckbert86]. Arvo called his textures "illumination maps"; in this thesis we call them radiosity textures. He computed them by tracing rays from the light sources. An adaptive variant of these textures is explored in chapter 5.

The polygonized radiosity concept is very similar to the radiosity texture concept, except the former is wedded to the idea of the polygons representing both geometry and shading,

[7] Cohen calls the large polygons "patches" and their sub-polygons "elements". We avoid the former term here, since it is sometimes assumed to be a synonym for "parametric surface".

while the radiosity texture data structure more explicitly segregates the data, associating all shading information with the texture, and all geometric information with the underlying surface.

2.5.8 Uniform vs. Adaptive Meshing

Polygonized radiosity and radiosity texture data structures both employ a mesh. Meshes can be segregated into two types, *uniform* or *adaptive*. Simple radiosity systems typically employ uniform meshes, subdividing rectangular polygons into a uniform grid of rectangular elements, for example. Adaptive meshing can be either *a priori*, in which the mesh is chosen before the solution is found, or *a posteriori*, in which a mesh is chosen based on a previous solution, a new solution is found, and the cycle is repeated as necessary [Rheinboldt-Mesztenyi80,Bank et al. 83].

A priori methods attempt to predict where additional subdivision is needed beyond a uniform mesh. Campbell's grid generation scheme predicts the location of shadow edges by approximating the light sources by one or more points and projecting all silhouette edges onto other surfaces in the scene [Campbell-Fussell90]. This technique is a valuable step toward better meshing, but it will usually find lines down the center of a penumbra, rather than the boundaries of the penumbra, where the discontinuities occur. Subdividing near a discontinuity improves the potential accuracy, but does not improve it as much as subdivision directly on the discontinuity.

A posteriori methods for global illumination have been examined more fully for ray tracing algorithms than for radiosity algorithms. In ray tracing algorithms one typically does not work with a mesh of elements chosen before solution, but with a set of samples that are accumulated during the course of the algorithm. Ward's algorithm, for example, samples most densely at corners and other regions where surfaces are in proximity [Ward et al. 88].

Another class of a posteriori methods are the *multigrid* techniques. Multigrid methods are fast solution techniques for finite difference and finite element approximations of differential or integral equations [Paddon-Holstein85]. They have not yet been applied to global illumination problems in a general way, to the author's knowledge. Multigrid methods solve a problem on a succession of scales, propagating low spatial frequency, slowly-varying components from coarse grids to fine grids, and propagating high-frequency, rapidly-varying components from fine grids back to coarse grids. For many classes of partial differential equations, they provide the fastest known solution methods.

Cohen's substructuring technique can be regarded as a simple form of multigrid method [Cohen et al. 86]. Substructuring typically solves the problem on only two grids, one of low resolution and one of high resolution, while multigrid methods solve the problem on a pyramid of grids.

Chapter 3

Radiosity in Flatland

In a three-dimensional world, it is difficult to visualize the global illumination equation and to test algorithms for its solution because of the high dimensionality of the functions involved. In full generality, intensity is a function of three-dimensional position $\mathbf{x}$, two-dimensional direction (θ, ϕ), wavelength, time, phase, and polarization, for a total of nine variables. The kernel κ of the integral equation for such a problem would have even more dimensions. Clearly it is difficult to understand such complex functions.

To simplify the problem, we temporarily restrict our attention to *radiosity in flatland*: a two-dimensional world consisting of opaque objects with diffuse emissivity and reflectivity. A flatland world is equivalent to a three-dimensional world where all objects have infinite extent along one direction. For now, we will restrict ourselves further to a static scene with line segments and closed polygonal shapes, diffuse light sources, no wavelength-dependence (i.e. grayscale), and no participating media. Clearly, this world could be generalized to allow specular reflective, and diffuse and specular transmissive materials, participating media, and curved surfaces, but we do not explore that here.

In this flatland world the global illumination problem reduces to the determination of the radiosity (a scalar) at each point on the edges in the scene. A flatland scene is shown in figure 3.1. Instead of shading two-dimensional surfaces and computing two-dimensional integrals, as we do in 3-D graphics, in flatland graphics we shade one-dimensional edges and compute one-dimensional integrals. Relative to three-dimensional worlds, in flatland one finds that analytic results are easier to come by, algorithms are easier to debug, brute force techniques such as Monte Carlo integration converge faster, and it is possible to compute approximate solutions so accurate that they can be regarded as exact. This facilitates the use of quantitative error metrics for the objective comparison of algorithms.

A second, less serious, reason to study radiosity in flatland is that Abbott's classic book [Abbott84] discussed the customs of that world's inhabitants, the flatlanders, but neglected their shading.

3.1 Integral Equation for Radiosity in Flatland

The dimensions of physical quantities in flatland differ from those in 3-D. In flatland, radiosity and emissive power have dimensions of power/distance, and intensity has units of power/(distance $\times$ angle).

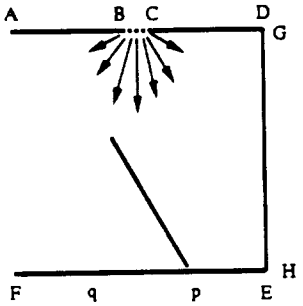

Figure 3.1: *Flatland test scene. All edges are reflective except dashed edge BC at top. which is a light source, and angled edge, which is black. The angled obstacle causes a sharp shadow edge at p and a gradual penumbra at q.*

Figure 3.2: *Radiosity as a function of arc length along the non-black edges of test scene. Note the sharp shadow edge at p and the gradual one at q.*

Figure 3.3: *Visibility geometry for edge points with parameter values s and s'.*

Suppose the scene consists of m edges, and the length of edge i is L_i. Note that free-floating line segments must be shaded on both sides, while edges of polygons only need to be shaded on the outside. Line segments can be regarded as 2-sided polygons. Each edge i is parameterized by arc length s_i in the range 0 to L_i, and the radiosity along each edge is given by $u_i(s_i)$. For convenience, we concatenate the domains of these functions in arbitrary order to create a single function $u(s)$ parameterized by s, which runs from 0 to $L = \sum_i L_i$. Note that this concatenation introduces explicit discontinuities at edge endpoints. In the following, when we refer to "the point s", what we mean is the point with parameter value s. The radiosity function can now be plotted as a piecewise-continuous function, as shown in figure 3.2.

A diffuse emitter with intensity i_{emit} has an emissive power of $2i_{\text{emit}}$, as is found by integrating $\cos\phi$ over the upper semicircle:

$$e = i_{\text{emit}} \int_{-\pi/2}^{\pi/2} d\phi \cos\phi = 2i_{\text{emit}}$$

Similarly, by analogy to hemispherical reflectivity in 3-D, the *semicircular reflectivity* of a diffuse material in flatland is $\rho_h = 2\rho$. The radiosity at each point is $u(s) = 2i_o(s)$.

For simplicity, we assume that each edge has constant reflectivity ρ_h and constant emissive power e. Edges are reflectors if $\rho_h > 0$, and light sources if $e > 0$, and occasionally both. As in 3-D, reflectivity is a unitless quantity between 0 (black) and 1 (perfect white).

The formulas for the integral equation of global illumination in flatland can be derived as in §2.2. We use the following variables (figure 3.3): position is parameterized by arc length variables s and s', outgoing angles at s and s' are ϕ_o and ϕ'_o, respectively, the incoming angle at s is ϕ_i, the distance between points s and s' is r, and the visibility flag between points s and s' is v. In 3-D, the solid angle subtended by an area at distance r is proportional to $1/r^2$, but in flatland, the angle subtended by a differential element of length ds' is proportional to $1/r$:

$$d\phi_i = \frac{ds' \cos\phi'_o}{r}$$

Therefore, like equation (2.15), the global illumination equation for flatland scenes with arbitrary surface scattering is:

$$i_o(s, \phi_o) = i_{\text{emit}}(s, \phi_o) + \int_0^L ds' \frac{\cos\phi_i \cos\phi'_o}{r} v\, \mu(s, \phi_i, \phi_o)\, i_o(s', \phi'_o)$$

and the integral equation of diffuse, opaque, flatland scenes is:

$$u(s) = e(s) + \rho_h(s) \int_0^L ds' \frac{\cos\phi_i \cos\phi'_o}{2r} v\, u(s') \tag{3.1}$$

As before, we abbreviate this as $u = e + \mathcal{K}u$, where the integral operator is:

$$(\mathcal{K}u)(s) = \int_0^L ds'\, \kappa(s,s')u(s')$$

and kernel

$$\kappa(s,s') = \rho_h(s)\frac{\cos\phi_i \cos\phi_o'}{2r}\, v$$

The variables ϕ_i, ϕ_o', v, and r are functions of s and s'. Note that the integral operator performs only a one-dimensional integration in flatland, in contrast to the two-dimensional integrations of 3-D.

If the unit normal at each point is $N(s)$ and the vector from s to s' is R, then $\cos\phi = R\cdot N(s)/|R|$, $\cos\phi' = -R\cdot N(s')/|R|$, and the kernel can also be written as

$$\kappa(s,s') = -\rho_h(s)\frac{(R(s,s')\cdot N(s))(R(s,s')\cdot N(s'))}{2|R(s,s')|^3}\, v(s,s') \tag{3.2}$$

The above kernel is not symmetric, but it can be symmetrized in the following way. If $\rho_h(s) \neq 0$ everywhere, then we can divide equation (3.1) by $\sqrt{\rho_h(s)}$:

$$\frac{u(s)}{\sqrt{\rho_h(s)}} = \frac{e(s)}{\sqrt{\rho_h(s)}} + \int_0^L ds'\rho_h(s)\frac{\sqrt{\rho_h(s')}}{\sqrt{\rho_h(s)}}\frac{\cos\phi_i\cos\phi_o'}{2r}\, v\, \frac{u(s')}{\sqrt{\rho_h(s')}}$$

yielding the integral equation $\tilde{u} = \tilde{e} + \tilde{\mathcal{K}}\tilde{u}$ where

$$\tilde{u}(s) = u(s)/\sqrt{\rho_h(s)}$$
$$\tilde{e}(s) = e(s)/\sqrt{\rho_h(s)}$$
$$(\tilde{\mathcal{K}}u)(s) = \int_0^L ds'\, \tilde{\kappa}(s,s')u(s')$$

with symmetric kernel

$$\tilde{\kappa}(s,s') = \sqrt{\rho_h(s)\rho_h(s')}\frac{\cos\phi_i\cos\phi_o'}{2r}\, v$$

In some scenes, the reflectivity ρ_h will be zero on certain edges, and changes are necessary to make the above symmetrization method work. In such cases, the domain $[0, L]$ can be separated into nonreflecting $\rho_h = 0$ and reflecting $\rho_h > 0$ portions, and the two portions treated differently. In the nonreflecting domain the radiosity is simply the emissive power: $u(s) = e(s)$. The above symmetrization technique can then be applied to the reflective domain with a modified emissive power function that includes the direct illumination effect of the nonreflective domain.

There is no cookbook solution method for integral equations; most cannot be solved analytically. Exact solutions to equation (3.1) are known only in the simplest geometries.

3.1.1 Simple Corner Scene

Consider the case of two edges of unit length at right angles (figure 3.4). If the emissive power, reflectivity, and radiosity of the edges are e_i, ρ_i, and $u_i(s_i)$ for $i = 1, 2$, then the

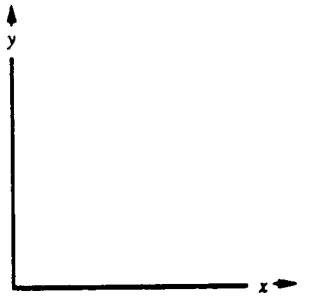

Figure 3.4: *Two unit edges forming a corner.*

solution is determined by the system of integral equations:

$$u_1(x) = e_1 + \frac{\rho_1}{2}\int_0^1 dy\, u_2(y)\frac{xy}{(x^2+y^2)^{3/2}}$$
$$u_2(y) = e_2 + \frac{\rho_2}{2}\int_0^1 dx\, u_1(x)\frac{xy}{(x^2+y^2)^{3/2}}$$

This can be solved analytically in the case where one of the reflectivities is zero, but no solution is known in the general case [Sparrow et al. 61,Horn77]. This suggests that analytic solution of any global illumination problem involving diffuse interreflection is impossible.

3.1.2 Neumann Series

The Neumann series for the global illumination integral equation (2.3)

$$u = e + \mathcal{K}e + \mathcal{K}^2e + \mathcal{K}^3e + \cdots$$

has a simple physical interpretation: the ith term $(\mathcal{K}^ie)(s)$ is the light that reaches the point s after exactly i 'hops' [Kajiya86], where a hop is an unoccluded straight-line path between surfaces. The approximant $u^{(i)}(s)$ is the light that reaches point s in i hops or fewer. Early illumination models (what Kajiya called the 'Utah approximation') simulated only direct illumination $u^{(1)} = e + \mathcal{K}e$; global illumination attempts to compute $u^{(\infty)}$. Unfortunately, it seems impossible to perform the multiple integrals $\mathcal{K}^ie$ analytically even for radiosity in flatland.

For the simple two-edge corner scene, the author has been unable to find a closed form beyond $u^{(1)}$. The integrals for $u_1^{(2)}(x)$ and $u_2^{(2)}(y)$ did not yield to hand integration or the symbolic algebra systems MACSYMA, MAPLE, or Mathematica.

The Neumann series for global illumination converges for physically realizable scenes. In flatland, power is the integral of radiosity with respect to arc length, and radiosity is nonnegative, so power is given by the L_1 norm: power $= \int ds\, u(s) = \int ds |u(s)| = ||u||_1$. By conservation of energy, the reflectivity $\rho_h < 1$ everywhere, so the total power of the light in a scene decreases after each reflection, since some of the power is absorbed at each bounce. Therefore, $||\mathcal{K}e||_1 < ||e||_1$, and the operator norm $||\mathcal{K}||_1 < 1$ by equation (2.1), so the Neumann series converges.

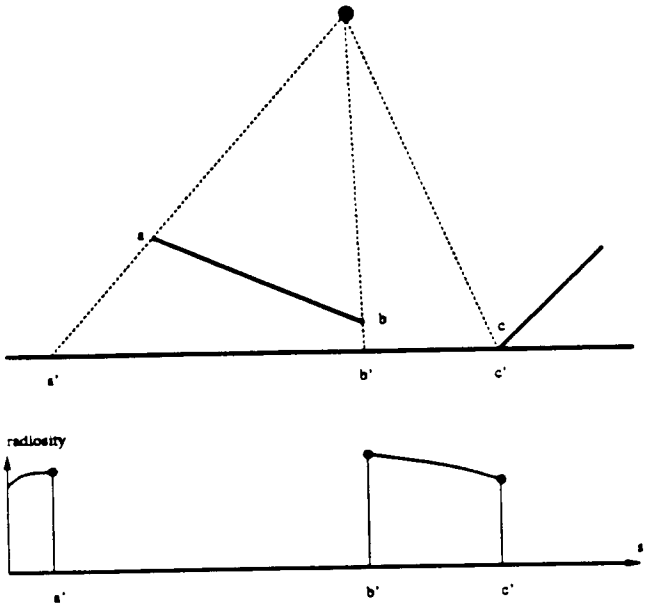

Figure 3.5: *D^0 discontinuities caused by point light source at top illuminating an occluded edge at bottom. D^0 discontinuities occur at a', b', and c'.*

3.1.3 Kernel Properties

In flatland, the kernel is a bivariate function. Since we have abutted the domains of the edges in the scene, the kernel's domain consists of rectangular blocks corresponding to pairs (e, f) of edges. The kernel is discontinuous at the boundaries of these blocks, and also along occlusion curves that trace out hyperbolas in st space (see figure 3.11). Note also that the kernel is singular at reflex corners in the scene (where touching surfaces face each other), because $\kappa \to \infty$ as $r \to 0$.

3.2 Discontinuities

We can derive many of the qualitative properties of the exact solution function $u(s)$ from the properties of the kernel and the geometry of the scene, even without solution algorithms.

We call a discontinuity in the kth derivative of a function a D^k discontinuity. A function thus has a D^k discontinuity at a point if it is C^{k-1} there but not C^k. Discontinuities in the radiosity can result from discontinuities in emissivity, reflectivity, normal vector, or visibility. Creases in a surface and polyhedron edges are examples of normal vector discontinuities. A D^k discontinuity in the normal of a curve causes a D^k discontinuity in the radiosity at that point.

In figure 3.5 we see how shadows due to a point light source can cause D^0 discontinuities in the value of the radiosity. Linear light sources cast hard shadows (causing D^0 discontinuities) when objects touch, and soft shadows (causing D^1 discontinuities) when objects do not touch (figure 3.6). The pattern generalizes to higher degree discontinuities according to the following law:

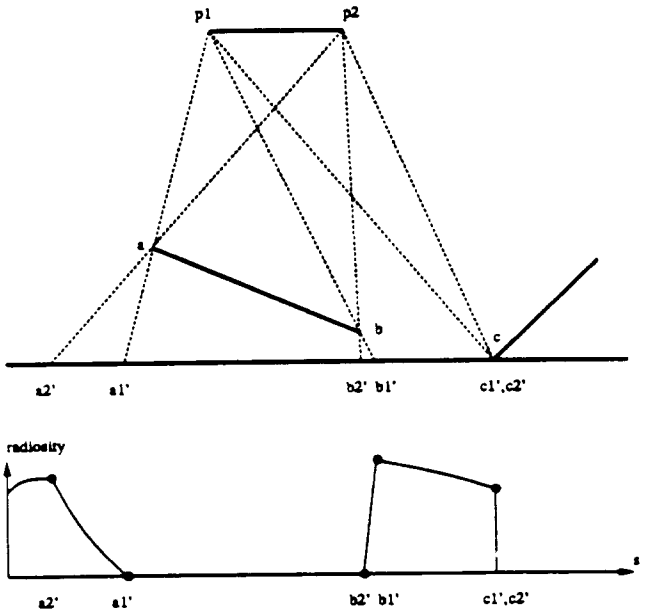

Figure 3.6: *D^1 and D^0 discontinuities caused by linear light source at top illuminating an occluded edge at bottom. D^1 discontinuities delimit the penumbras at a2', a1', b2', and b1'. D^0 discontinuity at the hard shadow edge c1'.*

Discontinuity Propagation Law: **If there is a D^k discontinuity at point s' on one edge, then it will cause D^k discontinuities at all touching points visible to it, and D^{k+1} discontinuities at the projection of all of the silhouette points from its point of view.** A touching point is a point at which two edges touch or intersect non-tangentially. If there are no occlusions in a scene then there are no discontinuities due to changes in visibility.

Earlier we noted that shadows from point and linear light sources cause D^0 and D^1 discontinuities, respectively. Using the Neumann series we can show how higher degree discontinuities arise after additional hops of light.

Theorem: There can be an infinite number of discontinuities of various degrees in the radiosity function.

Proof: This is proven with an example. Consider the scene in figure 3.7 that consists of three reflective and emissive edges in a triangle and $m-3$ black edges on a line in their interior. Let q_i denote the number of D^i discontinuities in $u^{(i)}$. If each triangle edge has a different emissive power, then $q_0 = 3$. After one hop, each corner creates one D^1 shadow edge from each end of the interior edges, so $q_1 = 3 \times 2(m-3)$. After subsequent hops, each D^i discontinuity creates $2(m-3)$ D^{i+1} shadow edges, but one of these is coincident with its 'grandparent' D^{i-1} discontinuity, and in general the remaining $2m-7$ shadow edges will not be coincident with lower degree discontinuities, so $q_i = 6(m-3)(2m-7)^{i-1}$ for $i \geq 1$. The exact solution $u^{(\infty)}$ will have all of these discontinuities. The number of discontinuities of all degrees is thus infinite in this scene. ∎

The possibility of so many discontinuities suggests that no analytic solution to the inte-

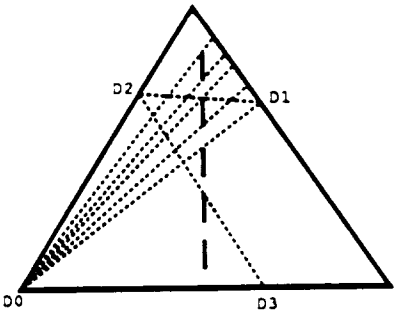

Figure 3.7: *Propagation of discontinuities. Solid lines show edges in scene; dotted lines show rays of light leading to discontinuities.*

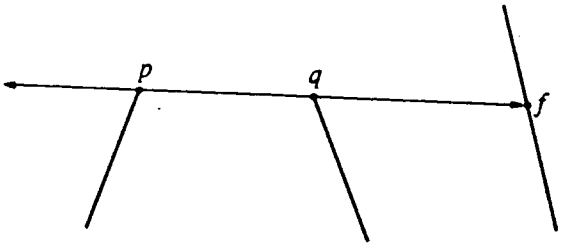

Figure 3.8: *f is a critical point caused by the critical line through endpoints p and q.*

gral equations for global illumination is possible in the presence of occlusions. We will therefore turn to numerical methods. Even with numerical approximations, however, accurate solution requires attention to the discontinuities. These can be predicted by determining visibility in the scene.

3.3 Visibility in Flatland

Visibility among opaque shapes in the plane has been studied extensively in computational geometry [Edelsbrunner et al. 83,O'Rourke87]. Consider a scene of m non-intersecting opaque line segments (edges) whose edges touch only at their endpoints. We call an edge point which is collinear with two edge endpoints visible from it a D^1 *critical point* and the line along which they lie is a *critical line* (figure 3.8). (We call it a D^1 critical point because there will likely be a D^1 discontinuity at this point. Because most of the critical points we discuss are of this type, we usually call them simply "critical points".)

The critical lines subdivide the scene into a *visibility partition* (figure 3.9), each cell of which is the maximal connected region of points in the plane with views of identical topology, each edge of which lies on a critical line, and each vertex of which is an edge endpoint or a critical point. Every pair of inter-visible edge endpoints p and q causes a critical line, and each critical line can cause two critical points at most, one if the ray from p in the direction $p - q$ hits an edge, and another if the ray from q in the direction $q - p$ hits an edge. Figure 3.8 shows the case where the second ray hits an edge, but the first ray

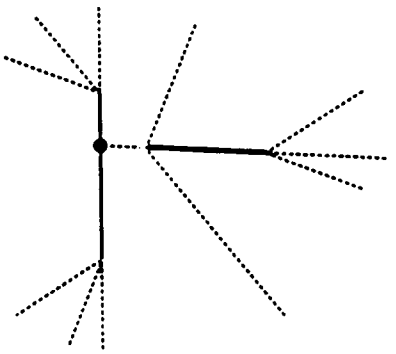

Figure 3.9: *Visibility partition for a scene of two edges. This scene has only one critical point (the dot).*

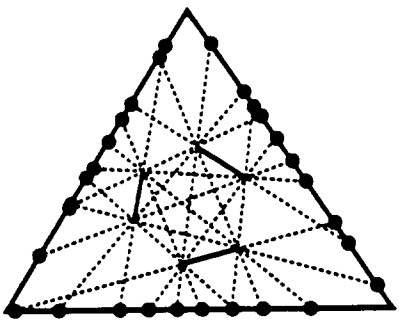

Figure 3.10: *Scene of a "Stonehenge circle" of $m-3$ separated edges in a circle inside a triangle (m=6 here). The scene has $2m-6$ edge endpoints in the interior. Critical lines through pairs of interior endpoints cause $2\binom{2m-6}{2}$ critical points, and critical lines through a triangle vertex and an interior point cause $3(2m-6)$ more critical points (not shown), for a total of $4m^2-20m+24$ critical points.*

does not. The contribution of each critical line to the visibility partition is thus either two rays, a ray and a line segment, or two line segments. The dual of this partition is called the *aspect graph* in the field of computer vision [Koenderink-van Doorn79,Gigus-Malik90].

The visibility partition is a subset of the cells of the planar arrangement of all $\binom{2m}{2}$ lines through the edge endpoints. The number of such cells is $O(m^4)$, but fortunately, to locate possible discontinuities we need not find the entire visibility graph. We are only interested in visibility from the edges, so we need only the *edge visibility partition*: the intervals of topologically isomorphic visibility along the edges of the scene.

Earlier, in figure 3.7, we gave an example of a scene in which the number of discontinuities grew exponentially with degree. We can now prove an upper bound on the number of discontinuities of each degree. Because we have assumed that edges do not intersect or touch in their interior, all D^0 discontinuities in such a scene will lie at edge endpoints, so $q_0 \leq 2m$. Assuming no point light sources, all D^1 discontinuities will occur at critical

points. Since each critical point is determined by a pair of edge endpoints, there are no more than $2\binom{2m}{2}$ critical points. The number of D^1 discontinuities for any scene is thus $q_1 \leq 4m^2$. This upper bound is achievable, asymptotically: figure 3.10 shows a scene with about $4m^2$ critical points. O'Rourke has given a similar scene with about m^2 critical points [O'Rourke87]. Discontinuities of degree $i > 1$ occur at edge points that are collinear with a D^{i-1} discontinuity and an edge endpoint both visible to it. There are $2m$ endpoints, so $q_i \leq 2mq_{i-1}$ for $i > 1$, and the upper bound on the number of discontinuities of degree i is $q_i \leq (2m)^{i+1}$. In practice, the number is usually far smaller than this upper bound, since parallel edges and occlusion reduce the number of critical points.

3.4 Numerical Solution Methods

Since no general analytic solution techniques for solving integral equations are known, we turn to numerical approximations. The integral equation can be approximated with either collocation or Galerkin techniques, as discussed in §2.1.

Classic radiosity algorithms use the collocation constraint with constant elements. If collocation points are placed at the center of the element intervals, at $s_i^* = (s_i + s_{i+1})/2$, and we let $K_{i,j} = \int_{s_j}^{s_{j+1}} ds' \kappa(s_i^*, s')$ then the collocation constraint is $u_i = e_i + \sum_j K_{i,j} u_j$. With the integral operator now reduced to a finite matrix, the integral equation is approximated by the linear system of equations $\mathbf{u} = \mathbf{e} + \mathbf{K}\mathbf{u}$, or $(\mathbf{I} - \mathbf{K})\mathbf{u} = \mathbf{e}$. Classic radiosity algorithms use the collocation approximation since they typically evaluate visibility only from the centers of polygons. When the hemicube algorithm is used to compute form factors, they make an additional approximation by numerically integrating the kernel.

For polynomial basis functions and flatland radiosity, the form factors can be computed analytically. The derivation is given in appendix A. The result is that each form factor $K_{i,j}$ can be computed analytically with the calculation of several square roots, a logarithm, and a number of arithmetic operations.

3.5 Solving Radiosity Systems

In order to find the best techniques for solving the system of equations it is necessary to understand the properties of radiosity matrices.

3.5.1 Radiosity Matrix Properties

There is a vast literature on the solution of the systems of linear equations [Golub-Van Loan89], but the fastest, most accurate methods for solving radiosity problems will likely be those that exploit the special properties of radiosity matrices. If collocation or Galerkin techniques are used, then the linear system has the form $\mathbf{A}\mathbf{u} = \mathbf{e}$, where $\mathbf{A} = \mathbf{M} - \mathbf{K}$ (equation (2.6) or (2.7)). For point collocation techniques with constant or linear elements, $\mathbf{M} = \mathbf{I}$. The radiosity matrix $\mathbf{A}$ for a flatland scene is shown in figure 3.11. Since the radiosity matrix is derived by integrating the kernel κ, many of the kernel's properties are reflected in the matrix.

When higher degree elements or Galerkin techniques are used, radiosity matrices become more complex and more difficult to characterize, but when constant or linear elements are used, with point collocation, the matrices are relatively easy to characterize.

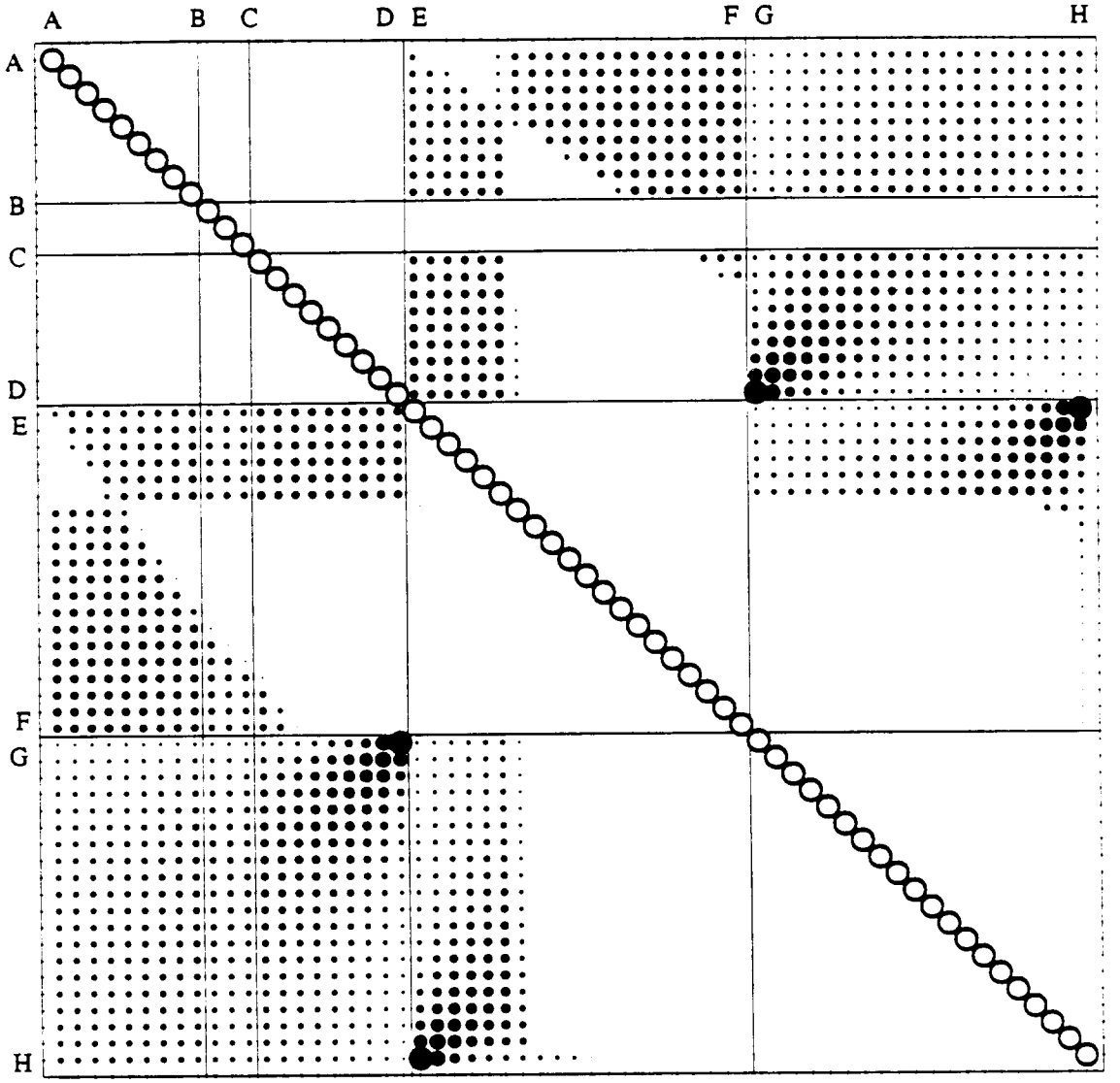

Figure 3.11: *Nonzero elements of the matrix $I - K$ for our test scene, with $n = 60$ equations. Diagonal elements (open circles) are all 1; off-diagonal elements are shown with area proportional to magnitude. White areas are zeros. Large dots indicate large form factors at reflex corners.*

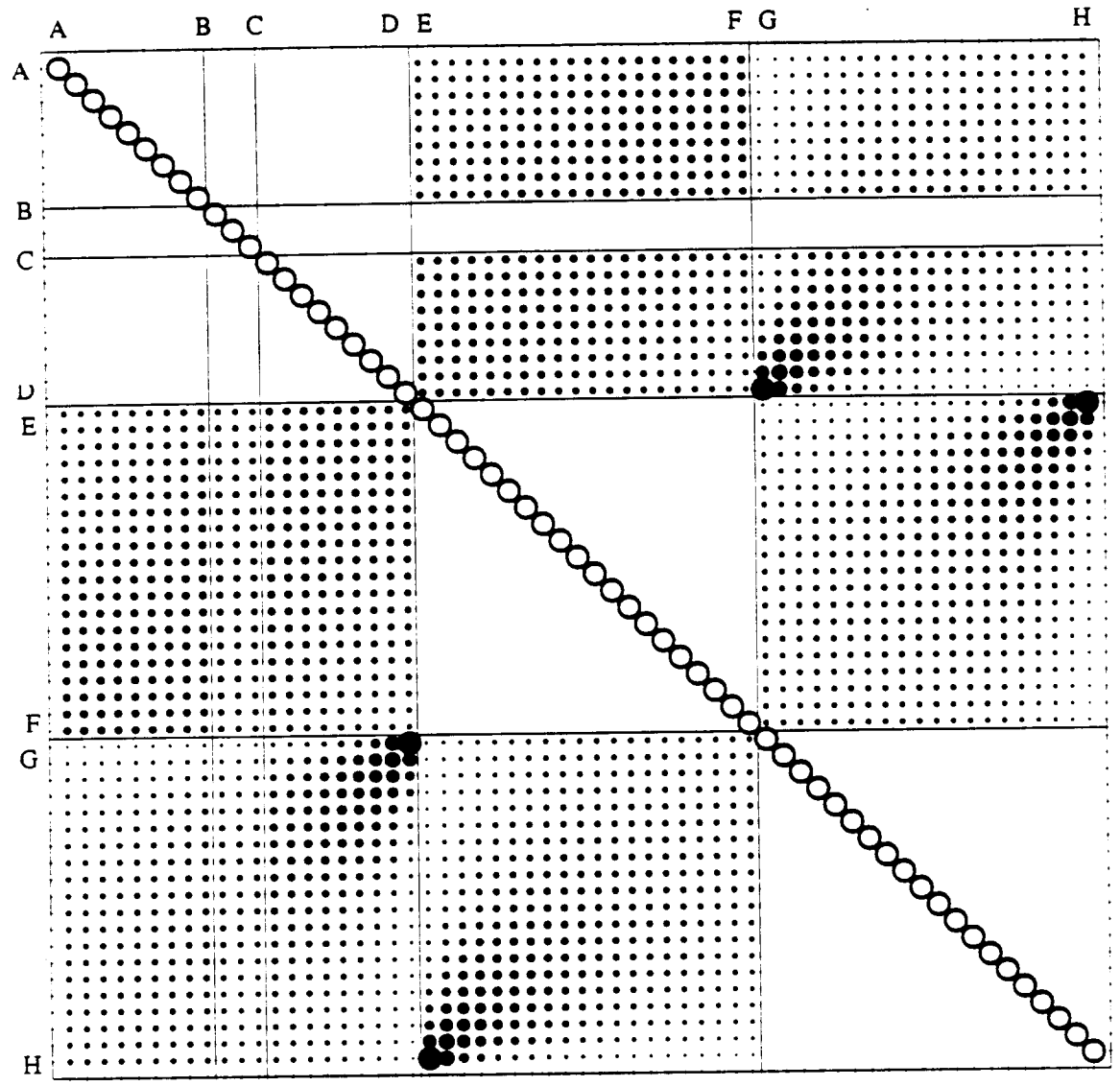

Figure 3.12: *Nonzero elements of the matrix $I - K$ for test scene with diagonal occluding edge removed. Occlusion causes sparseness. Blocks on diagonal contain zeros because no two points on the same face can "see" each other, and rows B-C are zero because edge B-C is nonreflective ($\rho_h = 0$).*

The matrices **A** encountered in radiosity problems are usually large, moderately sparse, diagonally dominant, non-symmetric positive definite, with largest magnitude eigenvalue less than 1. They are moderately sparse for most scenes of interest since each surface can "see" only a small fraction of the other surfaces. In a scene with no occlusion, such as the interior of a sphere, each surface point can see every other surface point, so the kernel and radiosity matrix for a reflective, occlusion-free scene are totally dense (nonzero everywhere).

If a systematic element ordering scheme is used, then the radiosity matrix will have block character, where blocks containing nonzeros correspond to surfaces that are inter-visible. A block that is entirely nonzero comes from a pair of surfaces with no intervening occluders, while a block that is only partially nonzero comes from a pair of surfaces that are partially occluded and partially inter-visible. The boundaries of the regions of zeros trace out hyperbolas in the matrix, just as in the kernel, if each edge is subdivided into elements of equal length. Except for the block boundaries and occlusion discontinuities, the kernel and matrix values vary smoothly.

Any physically valid scene will have a diagonally dominant radiosity matrix **A**. Physical validity constrains the range of reflectivity ($0 < \rho_k < 1$), which implies that integrals of the kernel are bounded and that the largest eigenvalue is less than one. This implies that the Neumann series (2.3) converges.

If the scene can be partitioned into two subscenes A and B such that no edge of A can see an edge of B, then the simulation can be split into two independent subproblems. This would occur in an architectural model, for example, if the door between two rooms A and B were closed, preventing light from traveling between them. When the scene can be partitioned in this way, the matrix **K** will be block diagonal with two blocks.

We define the graph of an $n \times n$ matrix **K**, $G(\mathbf{K})$, to be the directed graph with nodes $1 \ldots n$ and an edge from node i to node j iff $K_{ij} \neq 0$. The graph of a scene that can be partitioned is disconnected; that is, the graph has two nodes such that there is no directed path between them. When a scene cannot be partitioned, this graph is connected, the transitive closure of this graph is a complete graph (with edges between all pairs of nodes), and the matrix $(\mathbf{I} - \mathbf{K})^{-1}$ has no nonzeros.

3.5.2 Linear System Solving

Systems of equations can be solved by either direct methods or iterative methods [Strang86, Golub-Van Loan89]. Direct methods such as Gaussian elimination do not exploit the sparseness of matrices as well as some other methods, so they are most suited to small or low-sparsity systems.

One class of iterative methods, exemplified by the conjugate gradient method, generate a sequence of approximate solutions that are guaranteed to converge after n iterations for $n \times n$ systems (assuming exact arithmetic), and usually find accurate solutions far sooner. The conjugate gradient method is only suited to symmetric positive definite systems (all eigenvalues positive), but many radiosity problems can be preconditioned into this form by the substitutions $\mathbf{K} = \mathbf{P}\mathbf{K}^*$ and $\mathbf{u} = \mathbf{P}\mathbf{u}^*$, where $\mathbf{P} = \mathrm{diag}(\rho_1, \rho_2, \cdots, \rho_n)$. The matrix $\mathbf{K}^*$ is symmetric, so these substitutions transform the non-symmetric system $(\mathbf{I}-\mathbf{K})\mathbf{u} = \mathbf{e}$ into the symmetric system $(\mathbf{P} - \mathbf{P}\mathbf{K}^*\mathbf{P})\mathbf{u}^* = \mathbf{e}$. This approach is an example of the *preconditioned conjugate gradient method.*

Most iterative methods do not guarantee convergence in a finite number of iterations, but nonetheless they can be very fast and accurate in practice. Some of these methods

place conditions on the eigenvalues of the matrix, however. Iterative methods are often the fastest algorithms for extremely sparse matrices.

The matrices that are encountered in radiosity problems have well-behaved eigenvalues, but they are not very sparse; not nearly as sparse as those for many multidimensional problems. (In general, differential equations leads to much sparser systems than integral equations, because the numerical approximation to a derivative involves only local samples, while approximation of an integral requires samples over the entire domain.)

The simplest iterative algorithm is Jacobi's method, which, when applied to problems of the form $(\mathbf{I} - \mathbf{K})\mathbf{u} = \mathbf{e}$, computes the sequence of approximants

$$\mathbf{u}^{(i)} = \mathbf{e} + \mathbf{K}\mathbf{u}^{(i-1)}$$

for some initial guess $\mathbf{u}^{(0)}$. When applied to radiosity matrices of this form, and the initial guess $\mathbf{u}^{(0)} = \mathbf{e}$ is used, Jacobi's method is a discrete approximation to the Neumann series. As in the Neumann series, the approximant $\mathbf{u}^{(i)}$ for a radiosity problem consists of the light reaching each point in i hops or fewer. The Gauss-Seidel iterative method is a simple variant of Jacobi's method, which, for a large class of matrices, converges twice as fast as Jacobi. A simple extrapolating variation of Gauss-Seidel called *successive overrelaxation* accelerates convergence further.

3.5.3 Radiosity-Specific Techniques

An advantage of many iterative techniques is that the only use they make of the matrix is the computation of matrix-vector products. Consequently, the matrix need not be stored explicitly, but can be computed on the fly, row-by-row (or column-by-column), as the matrix product is computed.

The progressive radiosity approach is a radiosity-specific solution method of this type [Cohen et al. 88]. It computes selected columns of the radiosity matrix, shooting light from emitters and bright reflectors in decreasing order of total power, to iterate toward convergence. It is a form of Jacobi iteration operating on a column-wise expansion of the matrix $\mathbf{K}$. Unfortunately, no rigorous theoretical analysis of progressive radiosity has been done to date.

Hanrahan's radiosity solution technique [Hanrahan-Salzman90,Hanrahan et al. 91] is another novel approach which can be analyzed by analogy to algorithms for solving linear systems. Instead of using a fixed mesh resulting in a square matrix of form factors, shooting surfaces are subdivided adaptively relative to their distance to each receiving surface. This method avoids the matrix formulation altogether, instead sampling the kernel adaptively in a quadtree-like pattern.

3.6 Meshing

Early research in radiosity focused on the computation of form factors and efficient solution of the system of equations, but the issue of meshing or discretization of surfaces was little discussed; until recently it has remained a black art and a manual process for the most part [Baum et al. 91].

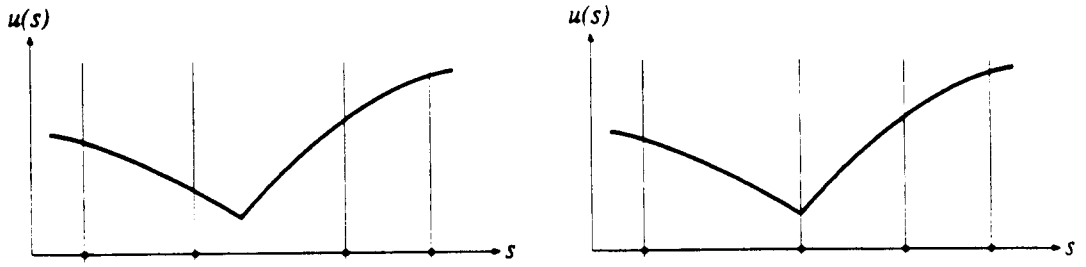

Figure 3.13: *Left: Mesh does not resolve the D^1 discontinuity (dots are element nodes). Right: Mesh resolves discontinuity.*

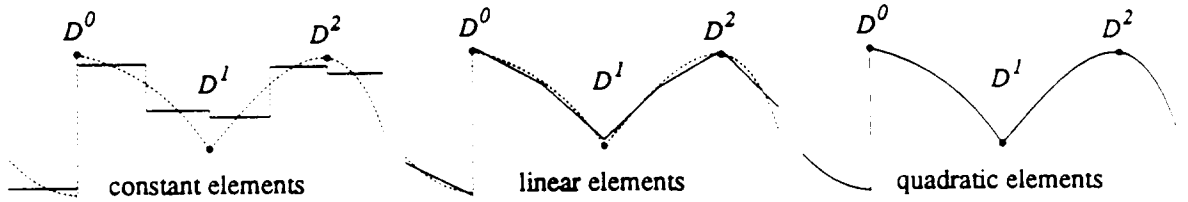

Figure 3.14: *Discontinuities of various degrees resolved and not resolved using constant, linear, and quadratic elements. Exact function shown dashed; approximations shown solid.*

3.6.1 Discontinuity Meshing

Discontinuity meshing is an approach to meshing that attempts to accurately resolve the most significant discontinuities in the solution by optimal positioning of element boundaries. (For the 1-D elements we use in flatland, the boundaries are the nodes.) In general, an approximation using polynomial elements will have discontinuities at element boundaries only.

When discontinuities in the true solution fall on element boundaries, the mesh is said to *resolve* the discontinuity (see figure 3.13). Elements of degree p can resolve discontinuities of degree D^0 through D^p, depending on the continuity constraints that are imposed between neighboring elements. To achieve the highest accuracy approximation, all discontinuities that can be resolved should be resolved. When using constant elements, discontinuity meshing can resolve all D^0 discontinuities. When using linear elements, a D^0 discontinuity can be resolved only if the boundaries coinciding with that discontinuity are unconstrained (allowing the elements on either side to have independent values), and a D^1 discontinuity can be resolved whether the boundary is constrained to be C^0 or is unconstrained (C^{-1}). The rules generalize to higher degree elements. The more constrained the mesh, the fewer the number of degrees of freedom to compute, but the stiffer (more nearly singular) the resulting system of equations.

In general, to resolve a D^p discontinuity, elements of degree p or higher must be used, and an element boundary must coincide with the discontinuity. It is not fruitful to resolve

discontinuities of degree greater than the element degree because the errors so caused are swamped by the discontinuities introduced at element boundaries (figure 3.14).

3.6.2 Discontinuity Meshing Algorithm

In flatland, discontinuity meshing can be done quite easily. As before, assume a scene consisting of m opaque line segments. Since the lowest degree discontinuities are generally the most significant, we first resolve D^0 discontinuities, then D^1 discontinuities, and so on up to the degree of the elements being used.

The most important discontinuities to resolve are D^0. D^0 discontinuities come from intersecting edges or from the projection of silhouette points from point light sources. Intersecting edges can be found in $O(m \log m)$ time, [Preparata-Shamos85]. These intersection points are marked as D^0 critical points (points at which a D^0 discontinuity could occur).

Next, D^1 discontinuities are found (unless constant elements are used, in which case we can stop here). D^1 discontinuities occur at D^1 critical points where there is a remote change in visibility, i.e. where an edge endpoint becomes occluded. As proven earlier, there are $O(m^2)$ D^1 critical points in a scene. A critical point will not cause a discontinuity if it is on a black edge ($\rho_k = 0$).

All D^1 critical points can be found as follows:

```
for each node p
    for each node q
        if no edge intersects line segment pq then {
            e = trace_ray(p, p-q)
            f = trace_ray(q, q-p)
            if e<>NULL then critical_point(e)
            if f<>NULL then critical_point(f)
        }
```

The routine `trace_ray(p, d)` traces a ray from point `p` in direction `d` and returns the first edge point hit, if any. The above algorithm, implemented straightforwardly, has $O(m^3)$ time cost. Alternatively, visibility can be determined by applying an $O(m \log m)$ radial sweepline perspective visibility algorithm [Edelsbrunner et al. 83] at both endpoints of each edge, yielding an $O(m^2 \log m)$ algorithm. Even faster algorithms that reduce the overall cost to $O(m^2)$ or less are also known [Edelsbrunner-Guibas89,Ghosh-Mount87].

Higher degree discontinuities can be located in a similar fashion. To find all D^k discontinuities, one loops over all D^{k-1} discontinuities, tracing rays from that point through all edge endpoints which are silhouettes with respect to that point. The point on the first edge hit by such a ray is called a D^k critical point.

3.7 Implementation

A program has been implemented to simulate radiosity in flatland which allows a choice of element degree and meshing technique. The program simulates diffuse interreflection among non-intersecting, simple polygons, with colored, diffuse reflecting and/or emitting edges. To simulate colors, samples at red, green, and blue wavelengths are used, and the wavelength bands are assumed independent, so the system of linear equations can be solved independently for each band. The program solves for radiosities using a collocation formulation with analytic form factors (no numerical integration) and either constant or

linear elements, as detailed in appendix A. Either uniform or discontinuity meshing can be used. Critical points on the m edges are found with the $O(m^3)$ ray tracing algorithm described above. Form factors are calculated with an object space visible edge algorithm using an $O(m^2)$ radial sweepline technique. For a scene discretized into n elements, the total cost of computing the matrices is $O(nm^2 + \alpha n^2)$, where α is the fraction of nonzero elements in the sparse matrix. The systems of equations are solved with successive overrelaxation using a default overrelaxation parameter of $\omega = 1.4$. The sparse matrix data structure uses $6\alpha n^2$ bytes for each of the three (R, G, B) components. For the scenes tested, matrix density α typically ranged between 10% and 40%.

Three display views are supported, an interactive flatland view, which is a top view of the scene, a graph of the red, green, and blue solution curves as a function of arc length, and a schematic of the radiosity matrix. The program runs on a Silicon Graphics workstation, and is able to re-solve and redisplay a scene consisting of 100 elements in about one second, while scenes containing 1000 elements require several minutes. An intuitive understanding of the occlusion discontinuities in the matrix is easily built up by interactively moving, creating, and deleting objects in the flatland view.

3.8 Results

Test runs have been performed to compare the speed and accuracy of six different solution methods resulting from a combination of one of the two meshing techniques, uniform meshing or discontinuity meshing, and one of three element types.

The three element types used are piecewise constant elements (box basis), piecewise linear elements (hat basis), and what we call Gouraud elements. Gouraud elements are piecewise constant elements for formulation and solution of the problem, followed by linear interpolation between collocation points for the 'display' step. In other words, the Gouraud approximation is computed as a post-process to the piecewise-constant approximation by interpolation and extrapolation between neighboring elements. We call them Gouraud elements by analogy with the computer graphics shading technique [Foley et al. 90]. Gouraud elements are the most commonly used approximation technique in existing radiosity algorithms.

To measure error objectively, we use the following error norm. If the exact radiosity function is u and the approximation is $\hat{u}$, then the error is defined to be

$$\text{error} = \frac{||\hat{u} - u||_2}{||u||_2}$$

where we use the L_2 norm defined in equation(2.2).

Since analytic solutions are not known for radiosity problems involving interreflection, the "exact" solution u is taken to be the approximate solution resulting from an extremely fine mesh. Both uniform and discontinuity meshing converge very close to this solution, verifying that it is a valid reference.

Figures 3.15-3.17 show solution of the radiosity for the scene in figure 3.1 with three of the six solution methods, and figures 3.18 and 3.19 show the convergence rate and the time-accuracy tradeoff.

As we might expect, constant and Gouraud elements show nearly the same (low) accuracy, convergence with decreasing element size is faster with linear elements than with

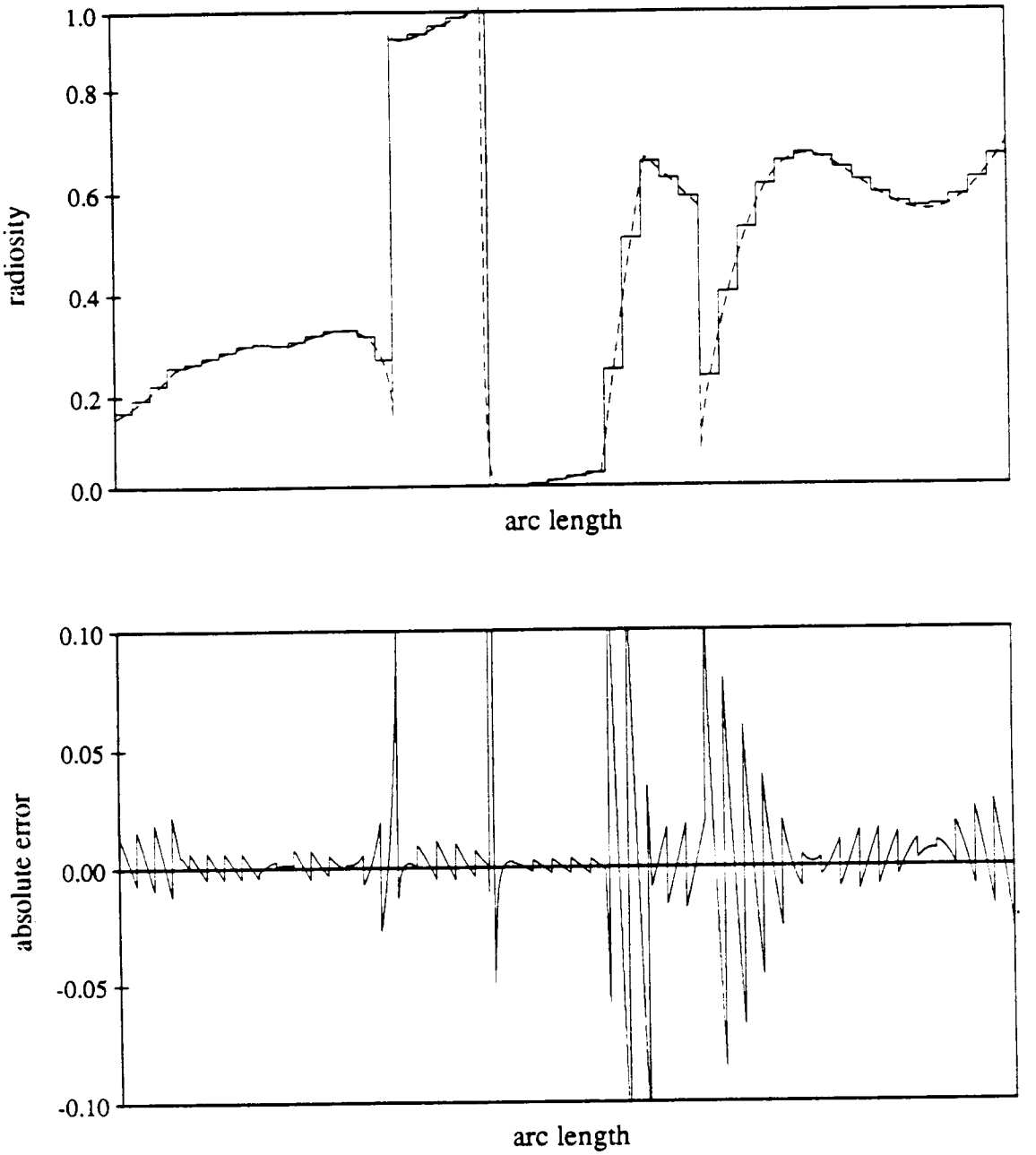

Figure 3.15: *Solid: approximate solution using uniform mesh and constant elements. Dashed: exact solution. Error is high everywhere.*

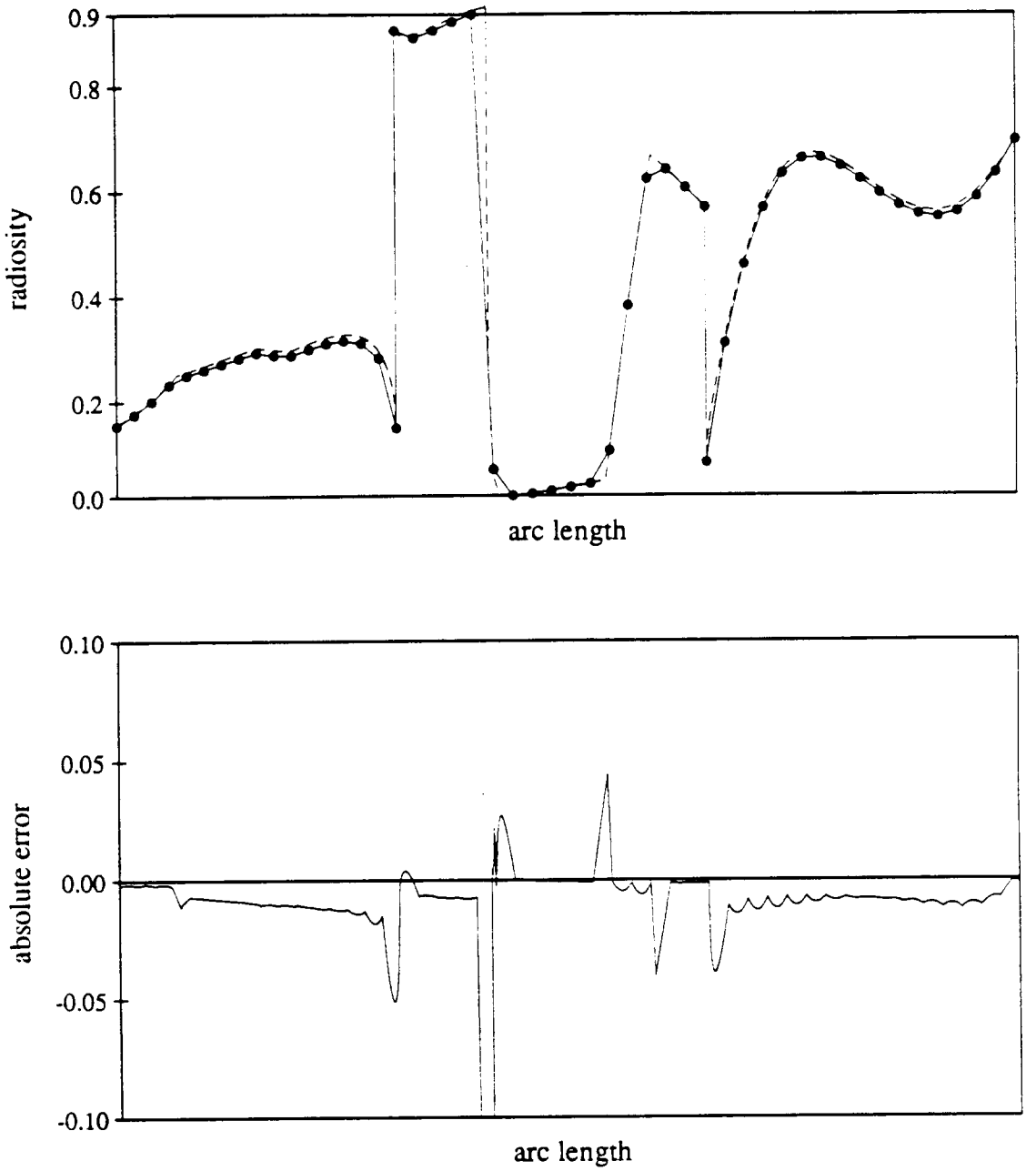

Figure 3.16: *Solid: approximate solution using uniform mesh and linear elements: dots are mesh nodes. Dashed: exact solution. Error is high at D^1 discontinuities because mesh does not resolve them. Light loss at discontinuities causes solution to be too dark at right.*

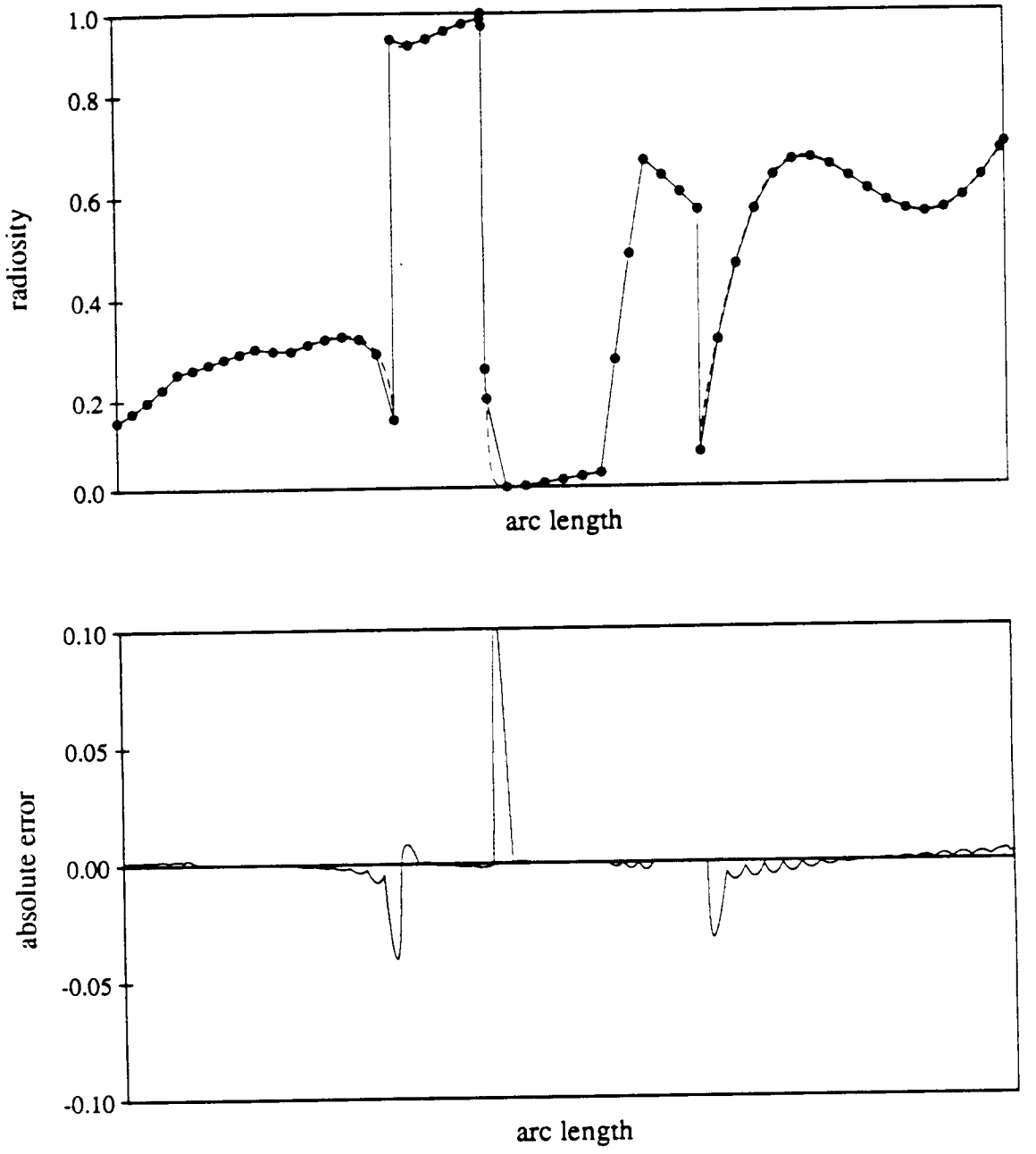

Figure 3.17: *Solid: approximate solution using discontinuity mesh and linear elements; dots are mesh nodes. Dashed: exact solution. Error is much reduced (the largest remaining errors seem to be caused by reflex corners).*

constant or Gouraud elements, and convergence rate is highly dependent on discontinuities. Without discontinuity meshing, solution with any of the three element types converges slowly, but with discontinuity meshing, convergence can be relatively fast. Discontinuity meshing with linear elements gives results over 10 times more accurate on fine meshes, in these experiments. Figure 3.19 shows that the use of discontinuity meshing and linear elements are cost-effective; for this experiment they give the best accuracy for a given amount of total CPU time, and the fastest results at a given accuracy.

Consider a specific example. When the test scene of figure 3.1 was simulated with discontinuity meshing with $n = 91$ equations, it required 73K bytes of memory and 1.3 seconds of CPU time. To achieve the same accuracy with uniform meshing required $n = 775$ equations, 4.5M bytes of memory, and 74 seconds. Discontinuity meshing, for this test case, gave results of the same quality as uniform meshing using about 1/60th the time and 1/60th the memory.

For a simple scene, about 80% of the CPU time is spent computing the radiosity matrix, 5% on discontinuity meshing, and 15% on system solving. This balance could change depending on the relative number of discontinuities and elements, and system sparseness.

3.9 Conclusions

These experiments show that the use of discontinuity meshing and linear elements can improve the accuracy of radiosity solutions dramatically, relative to conventional approximations employing constant elements or constant elements that are linearly-interpolated as a postprocess (Gouraud elements) and simple, uniform meshes. We have not proven this theoretically for all scenes, but we have demonstrated it on rather simple scenes in flatland.

The accuracy could theoretically be increased even further by doing higher degree discontinuity meshing (resolving D^2, D^3, etc) or by using quadratic and higher degree elements. Galerkin techniques should be explored; they should give more accurate results than collocation. Unfortunately, it appears impossible to compute the double integrals required for flatland Galerkin analytically, so Gaussian quadrature or similar numerical methods must be employed [Heckbert-Winget91]. We would like to find the most cost-effective combination of these techniques. We expect that this combination will have well-balanced errors, in the sense that the errors made at each stage (basis, mesh, constraint, integration, solution) will be roughly equal.

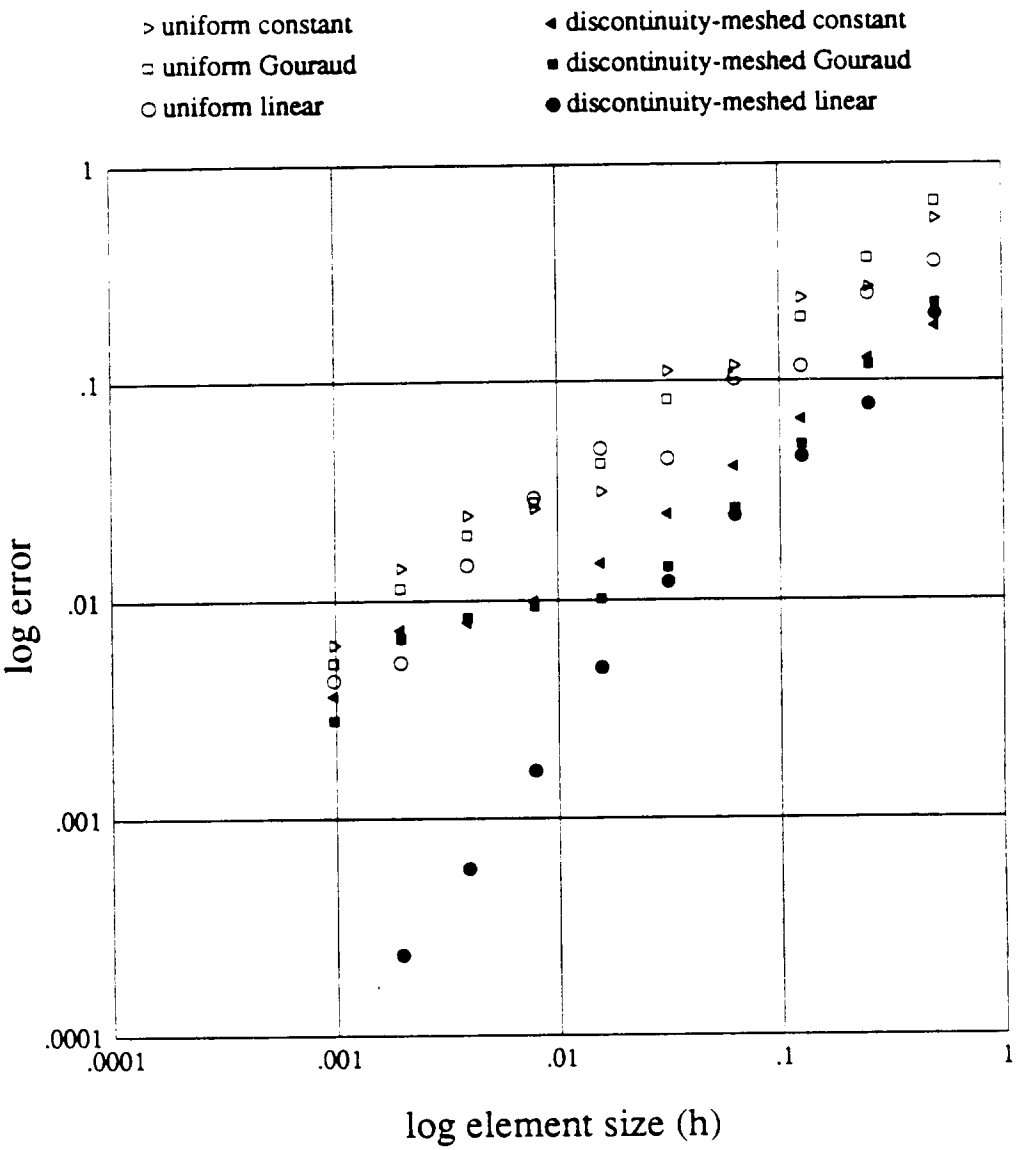

Figure 3.18: *Element size vs. error for six different solution methods. Note that discontinuity meshing with linear elements is far more accurate, and that Gouraud elements are little better than constant elements.*

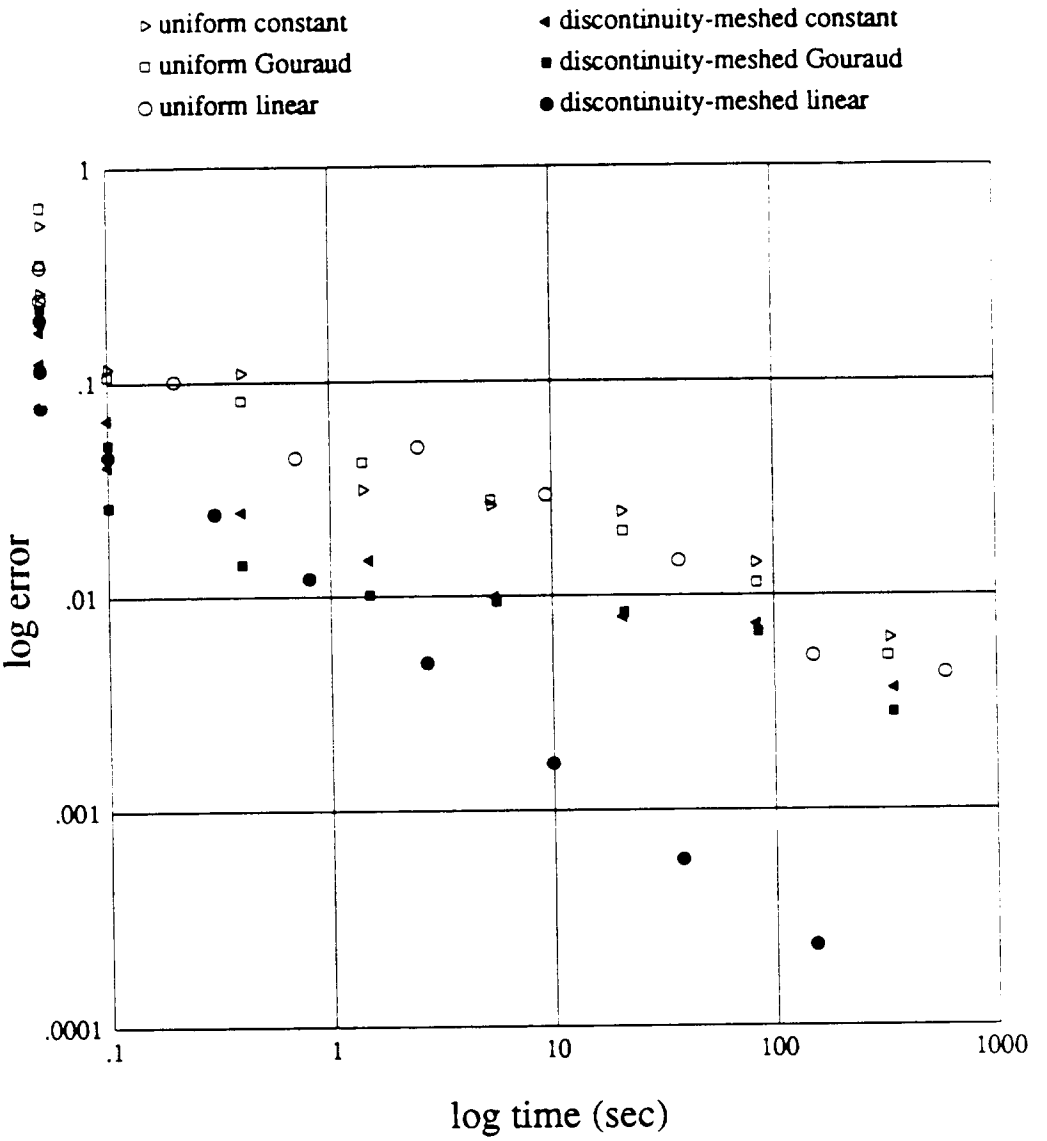

Figure 3.19: *Total CPU time (meshing, matrix computation, solution, and I/O) vs. error for six different solution methods. In this experiment, discontinuity meshing with linear elements is the most cost effective of the techniques tested except for very fast, crude simulations.*

Chapter 4

Radiosity in Three Dimensions

The general character of kernels, visibility, and discontinuities in 3-D radiosity problems are similar to those of 2-D. As before, we restrict discussion to non-participating media.

Shading a 3-D scene is more complex than shading a 2-D scene in several respects. First, whereas the intensity function for a flatland scene with arbitrary scattering has a two-dimensional domain (arc length and direction) and the intensity function for diffuse scattering has a one-dimensional domain (arc length), in 3-D the intensity function for arbitrary scattering has four dimensions (two for surface parameters and two for direction) and the intensity function for diffuse scattering has two dimensions (the surface parameters). Also, the geometry and topology of surfaces and the 2-D meshes on these surfaces are significantly more complex in 3-D than the analogous curves and 1-D meshes in 2-D.

We assume a scene consisting of m opaque, diffuse polygons in 3-D space. The topological objects in this scene are the polygonal faces and their edges and vertices.

4.1 Visibility in 3-D

Visibility and the discontinuities caused by changes in visibility are more complex in 3-D than in 2-D. In 2-D, changes in visibility (also known as *visual events*) occur along critical lines, and potential discontinuities occur at critical points where these lines intersect edges of the shapes in the scene. In 3-D, visual events occur along *critical surfaces*, and potential discontinuities occur along *critical curves* where these surfaces intersect the faces of the scene.

For a scene of polygons, there are two types of visual events: edge-vertex events (or EV events) and edge-edge-edge events (or EEE events) [Gigus-Malik90].

EV events result from an inter-visible edge e and vertex v. The critical surface for this event is a subset of the plane containing the edge and the vertex. More precisely, it is the set of points collinear with vertex v and a point p of edge e, that are not between v and p. We call this surface a *wedge* (figure 4.1). The critical curves caused by an EV event are line segments; they are the points of the intersection of the wedge with the faces of the scene that are visible to v and p.

EEE events result from three inter-visible, skew edges (figure 4.2). The critical surface is the set of points simultaneously collinear with one point from each edge but not between any two of these three points. This surface is a subset of a quadric [Gigus-Malik90]. The critical curves caused by an EEE event are conics; they are the face points on the critical

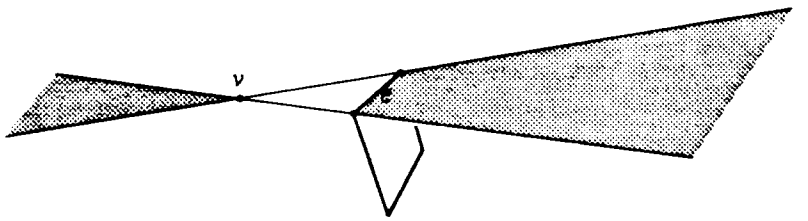

Figure 4.1: *The wedge defined by vertex v and edge e.*

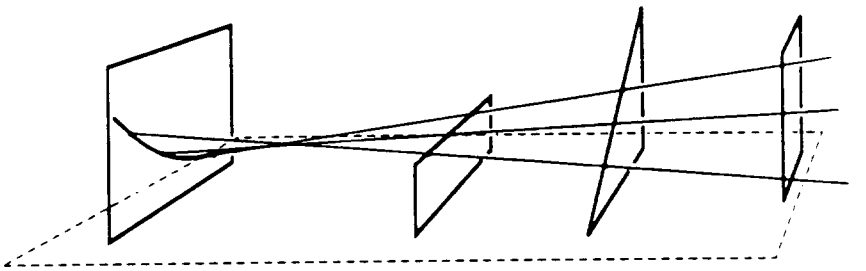

Figure 4.2: *EEE event caused by three skew edges at right creates a quadric critical surface and a conic critical curve on face at left.*

surface that are visible to the three edge points on the line. The EEE critical surface is a ruled surface (i.e. each point of the surface lies on a line coincident with the surface). In general, this quadric is a hyperboloid of one sheet [Hilbert-Cohn-Vossen32, p. 14].

4.2 Discontinuities in 3-D

As in flatland, touching or intersecting surfaces usually yield D^0 discontinuities along their common curve. The most significant discontinuities resulting from non-touching edges in flatland are D^1, but in 3-D, they are generally D^2.

4.2.1 EV and EEE Discontinuities

Many of the characteristics of EV discontinuities in 3-D are exhibited in a scene with emitter, occluder, and receiver polygons in parallel planes (figure 4.3). When the emitter, occluder, and receiver are not in parallel planes, the penumbra mesh is topologically equivalent, but geometrically distorted.

The degree of a discontinuity can be determined by considering the variation in direct illumination intensity with infinitesimal displacements away from the critical curve. For such infinitesimal displacements, the discontinuity degree is equal to the degree of the change in the visible (unoccluded) area of the emitter from a moving viewpoint (figure 4.4).

Considering an EV event involving a vertex from an emitter and an edge from an occluder, no edge of the emitter will be parallel to an edge of the occluder, in general, so infinitesimal variations in position on the receiving plane will result in quadratic variation

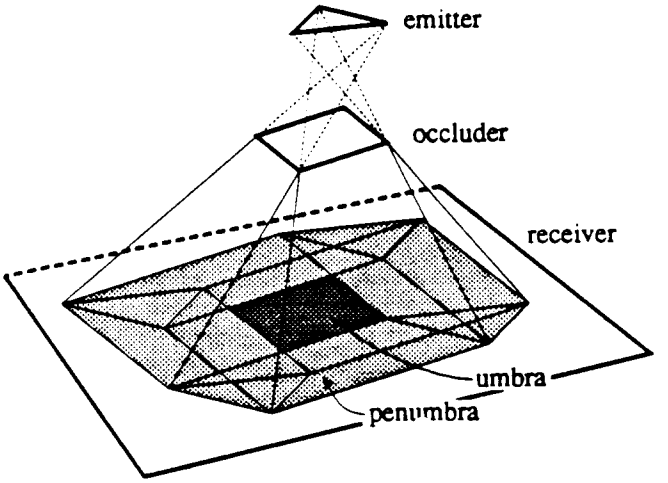

Figure 4.3: *Discontinuities caused by a triangular emitter, rectangular occluder, and receiver in parallel planes (after [Nishita-Nakamae83]).*

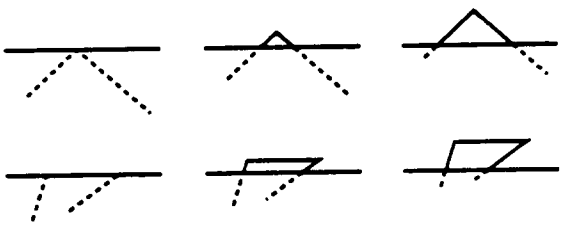

Figure 4.4: *View from receiver near an EV event. Top: As emitter emerges from behind edge of an occluder, its visible area generally grows quadratically, generating a D^2 discontinuity. Bottom: When emitter has edge parallel to occluder, its visible area grows linearly, generating a D^1 discontinuity.*

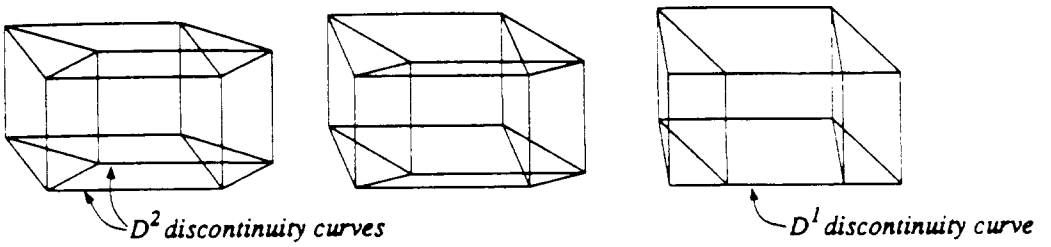

Figure 4.5: *Discontinuities from a triangular emitter and rectangular occluder, for three different rotations of the emitter. When edges of emitter and occluder become parallel, two D^2 discontinuity curves coincide, yielding a D^1 discontinuity.*

in intensity, yielding a D^2 discontinuity. In the special case of parallel emitter and occluder edges, the intensity will vary linearly in the neighborhood of the critical curve, yielding a D^1 discontinuity.

The EV discontinuities caused by a triangular emitter and rectangular occluder are shown in figure 4.5. When rotation of the occluder in its plane causes two edges of emitter and occluder to become parallel, two D^2 discontinuity curves coincide, yielding a D^1 discontinuity.[1]

The discontinuities along EEE critical curves are generally D^2. Our understanding of EEE critical curves is currently rather poor, however.

4.2.2 Number of Discontinuities

In a "general" scene, most discontinuity curves are D^2. It is only the coincidence of two D^2 discontinuities which causes D^1 discontinuities.

The number of EV critical surfaces is $O(m^2)$ worst case, and the number of EEE critical surfaces is $O(m^3)$ worst case, for a scene of m faces. These give the upper bounds of $O(m^3)$ EV critical curves and $O(m^4)$ EEE critical curves, respectively, since each critical surface can intersect at most m faces. These may not be least upper bounds. They are pessimistic for practical scenes, which often have parallel edges leading to many coincident critical surfaces and a far lower number of critical curves. Further work is needed on the complexity of critical curves in practical scenes.

4.3 Discontinuity Meshing in 3-D

Mesh boundaries must follow discontinuity curves in order to resolve a discontinuity in 3-D (figure 4.6). As in flatland, it is D^0 discontinuities resulting from touching or intersecting surfaces and point light sources that most urgently need to be resolved [Baum et al. 91].

In most scenes, the majority of critical curves will have very small discontinuities, that is, they will have degree two or higher and/or very small coefficients, so failure to resolve them

[1] This can be observed empirically by holding a stiff piece of paper horizontally below a rectangular lamp fixture and a few inches above a white tabletop. Rotate the piece of paper in its plane, and observe the increased crispness of the shadows at parallel orientations. The intensity function has D^1 discontinuities at parallel orientations, but D^2 otherwise.

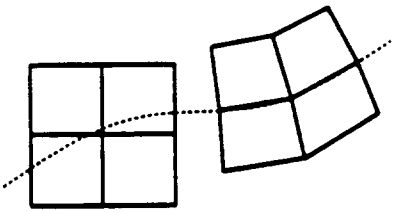

Figure 4.6: *Discontinuity curve and mesh in surface parameter space. Left: Mesh does not resolve the discontinuity (dashed curve). Right: Mesh approximately resolves discontinuity.*

will not lead to large errors. In fact, unless all other sources of error in the simulation are very small (including choice of basis function, numerical integration, and systems solving), the error caused by failure to resolve small discontinuities will be negligible.

The most practical approach for discontinuity meshing in 3-D is thus to resolve only the most "significant" discontinuities. One possible definition of "significance" would employ the degree and coefficient of the discontinuity. Such a measure would be a function of the infinitesimal neighborhood only. A different, perhaps better, definition of significance would consider all changes in a finite neighborhood. Two nearly coincident discontinuities can cause the same total change as two exactly coincident discontinuities. Reasonable candidates for finite significance metrics are the first derivative (gradient) and second derivative of the direct illumination intensity over a small region.

Without further analysis or experiments to determine the sources of error in a 3-D radiosity simulation, it is difficult to say with certainty which discontinuities should be resolved. In general, however, we recommend the following. When constant elements are used, only strong (closely spaced or large coefficient) D^0 discontinuities need be resolved. When linear elements are used, only D^0 and strong D^1 discontinuities need be resolved. And when quadratic elements are used, only D^0, D^1 and strong D^2 discontinuities need be resolved. Experiments are needed to verify these recommendations, however. Increasing the degree of the elements increases the potential accuracy, but it also increases the number of discontinuities that can be (and need to be) resolved, in a simulation method with well-balanced errors.

Radiosity functions in 3-D are smoother than those in 2-D, in general. Since linear elements have been shown to be quite successful in 2-D, and 3-D radiosity solutions are smoother than those of 2-D, the use of quadratic elements with C^1 continuity between elements appears promising for 3-D.

Figure 4.3 suggests an algorithm for determining all EV critical line segments. To generate the mesh of critical line segments, we can use an object space visibility algorithm [Atherton et al. 78,Nishita-Nakamae83,Campbell-Fussell90,Chin-Feiner90]. Half of the critical segments can be generated by projecting the occluders from each of the vertices of the emitter forward onto each of the other faces in the scene. The other half of the critical segments are generated by projecting the emitter from each of the vertices of the occluders backward onto each of the faces in the scene. The first half of these segments are approximated by Campbell's algorithm, but not the second half [Campbell-Fussell90].

The generation of the conic critical curves from EEE events is more complex, and less easily related to visibility algorithms.

Chapter 5

Bidirectional Ray Tracing

An alternative approach for the simulation of global illumination is Monte Carlo techniques. When using Monte Carlo techniques for visibility operations, the fundamental operation is the tracing of a ray: determining the intersections of a ray with the geometric shapes in the scene. The use of Monte Carlo techniques facilitates the simulation of specular scattering (recall the scattering classes of figure 2.6).

We present a rendering method designed to provide general simulation of global illumination for realistic image synthesis. This method simulates a broader class of scenes than the preceding radiosity algorithms; it is not limited to polygonal, diffuse surfaces as most radiosity techniques are.

Separating surface scattering into diffuse plus specular, we compute the specular component on the fly, as in ray tracing, and store the diffuse component (the radiosity) for later-reuse, similar to a radiosity algorithm. Radiosities are stored in *adaptive radiosity textures* (*rexes*) that record the pattern of light and shadow on every diffuse surface in the scene. They adaptively subdivide themselves to the appropriate level of detail for the picture being made, resolving sharp shadow edges automatically.

These techniques are more general than those of the previous chapters because they allow curved surfaces, transmission, and more general light scattering, but they are not so general that the algorithm becomes extremely inefficient. We make the following assumptions:

(1) Only surfaces are relevant. The scattering or absorption of volumes can be ignored.

(2) Curved surfaces are important. The world is not polygonal.

(3) Shadows, penumbras, texture, diffuse interreflection, specular reflection, and refraction are all important.

(4) We can ignore the phenomena of fluorescence (light wavelength crosstalk), polarization, and diffraction.

(5) The bidirectional surface scattering function μ can be expressed as a linear combination of diffuse and specular reflectivity and transmissivity functions:

$$\begin{aligned}\mu(\mathbf{x}, \Theta_i, \Theta_o) =& k_{dr}(\mathbf{x})\rho_d(\Theta_i, \Theta_o) + k_{sr}(\mathbf{x})\rho_s(\Theta_i, \Theta_o) \\ &+ k_{dt}(\mathbf{x})\tau_d(\Theta_i, \Theta_o) + k_{st}(\mathbf{x})\tau_s(\Theta_i, \Theta_o)\end{aligned}$$

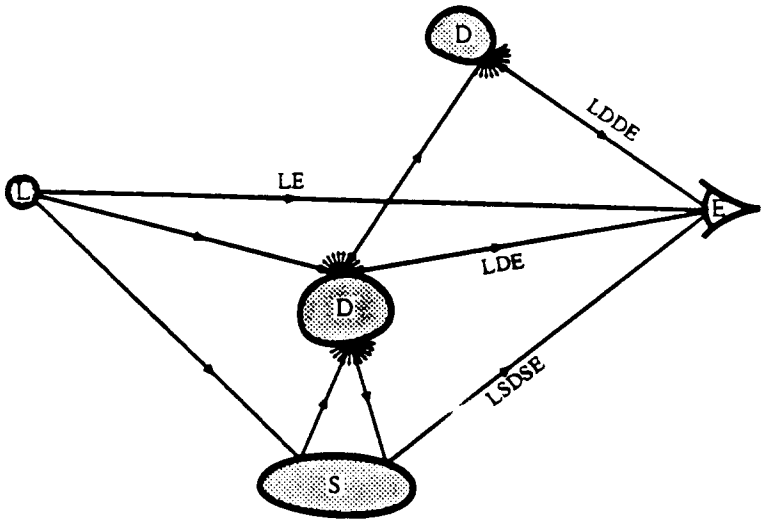

Figure 5.1: *Selected photon paths from light (L) to eye (E) by way of diffuse (D) and specular (S) surfaces. For simplicity, the surfaces shown are entirely diffuse or entirely specular; normally each surface would be a mixture.*

The four diffuse and specular reflectivity and transmissivity functions above have the specific directional dependency given in §2.2.2, but are independent of surface position, while the coefficients k_{ij} are direction-independent but position-dependent.

(6) Specular surfaces are not rough; all specular scattering is ideal.

5.1 Approach

Our approach is a hybrid of radiosity and ray tracing ideas. Rather than patch together these two algorithms, however, we seek a simple, coherent, hybrid algorithm. To provide the greatest generality of shape primitives and optical effects, we choose ray tracing as the visibility algorithm. Because ray tracing is weak at simulating global diffuse scattering, the principal task before us is therefore to determine an efficient method for calculating radiosities using ray tracing.

To exploit the view-independence and coherence of radiosity, we store radiosity with each diffuse surface, using an *adaptive radiosity texture*, or *rex*. A rex records the pattern of light and shadow and color bleeding on a surface. We store radiosity as a texture, rather than as a polygonization, in order to decouple the data structures for geometry and shading, and to facilitate adaptive subdivision of radiosity information. We store it with the surface, rather than in a global octree [Ward et al. 88], or in a light image, based on the intuition that radiosities are intrinsic properties of a surface. We expect that the memory required for rexes will not be excessive, since dense sampling of radiosity will be necessary only where it has a high gradient, such as at shadow edges.

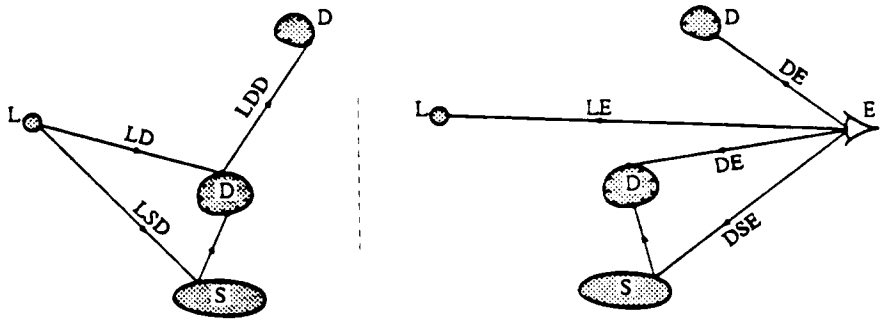

Figure 5.2: *Left: first level light ray tracing propagates photons from the light to the first diffuse surface on a path (e.g. LD and LSD); higher levels of progressive light ray tracing simulate indirect diffuse scattering (e.g. LDD). Right: eye ray tracing shoots rays from the eye, extracting radiosities from diffuse surfaces (e.g. it traces LE, DE, and DSE in reverse).*

Next we need a general technique for computing the rexes. The paths by which photons travel through a scene can motivate our algorithm (figure 5.1). If the scattering function $\mu(\Theta_i, \Theta_o)$ can be split into two parts, the diffuse scattering μ_d and the specular scattering μ_s, then the kernel of the global illumination equation can be split as $\kappa = \kappa_d + \kappa_s$, and by linearity, the integral operator can be split as well: $\mathcal{K} = \mathcal{D} + \mathcal{S}$. The Neumann series for the intensity solution function can then be expanded in terms of the diffuse and specular scattering operators $\mathcal{D}$ and $\mathcal{S}$:

$$\begin{aligned} u &= e + \mathcal{K}e + \mathcal{K}^2e + \mathcal{K}^3e + \cdots \\ &= \Big(\mathcal{I} + (\mathcal{D}+\mathcal{S}) + (\mathcal{D}+\mathcal{S})^2 + \cdots\Big)e \\ &= \Big(\mathcal{I} + (\mathcal{D}+\mathcal{S}) + (\mathcal{D}\mathcal{D} + \mathcal{D}\mathcal{S} + \mathcal{S}\mathcal{D} + \mathcal{S}\mathcal{S}) + \cdots\Big)e \end{aligned}$$

Formally, this shows that light can travel by a path that includes any number of diffuse or specular scattering bounces, in any order. This infinite series accounts for all light paths. Note that integral operators are not commutative, in general. Sillion employs a similar derivation to motivate his ray tracing algorithm [Sillion-Puech89].

By analogy to the integral operator notation above, we denote photon paths using a regular expression notation[1]. We regard each scatter along a photon's path from light (L) to eye (E) as either diffuse (D) or specular (S) or a blend of the two. Each path can therefore be labeled with some string in the set given by the regular expression $L(D|S)^*E$. The simplest paths are thus LE; LDE, LSE; $LDDE$, $LDSE$, $LSDE$, $LSSE$; ... just as in the Neumann series above. Classic ray tracing simulates only $LDS^*E \mid LS^*E$ paths, while classic radiosity simulates only LD^*E. Eye ray tracing has difficulty finding paths such as LS^+DE because it doesn't know where to look for specularly reflected light when integrating the hemisphere. Such paths are easily simulated by light ray tracing, however.

[1] A *regular expression* is a particular notation for a set of strings. The regular expression X^+ denotes a string of one or more X characters: either X, or XX, or XXX, or $XXXX$, etc. The regular expression X^* denotes zero or more X characters. Parentheses denote grouping and a vertical bar (|) denotes 'or'.

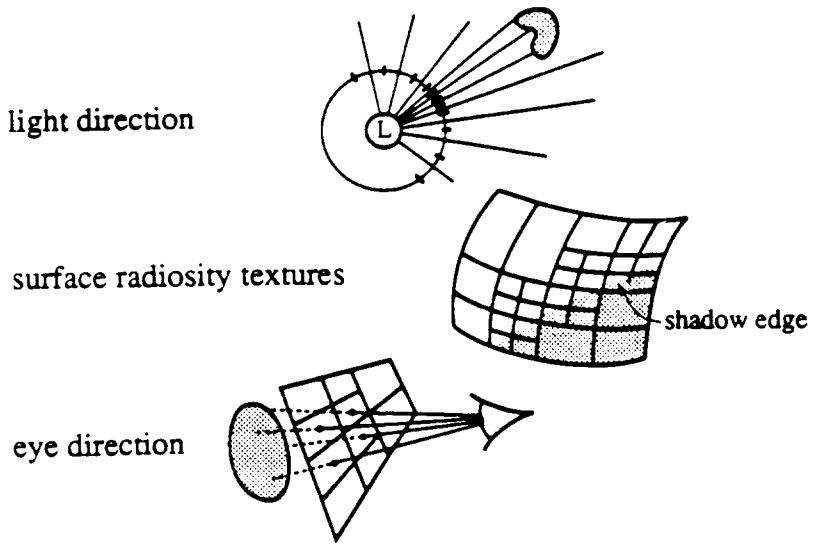

Figure 5.3: *Three sampling processes in the bidirectional ray tracing algorithm.*

Just as eye ray tracing is a Monte Carlo approximation of the integral operator $\mathcal{K}$, light ray tracing approximates the adjoint integral operator $\mathcal{K}^*$.

We digress for a moment to discuss units. Light rays carry power and eye rays carry intensity. Each light ray carries a fraction of the total power emitted by the light.

We can simulate paths of the form LS^*D by shooting light rays (photons) into the scene, depositing each photon's power into the rex of the first diffuse surface encountered (figure 5.2, left). Such a light ray tracing pass will compute a first approximation to the radiosities. This can be followed by an eye ray tracing pass in which we trace DS^*E paths in a backward direction, extracting intensity from the rex of the first diffuse surface encountered (figure 5.2, right). The net effect of these two passes will be the simulation of all LS^*DS^*E paths. The rays of the two passes "meet in the middle" to exchange information. To simulate diffuse interreflection, we shoot progressively from bright surfaces [Cohen et al. 88] during the light ray tracing pass, thereby accounting for all paths: $\{L(S^*D)^*S^*E\} = \{L(D|S)^*E\}$. We call these two passes the *light pass* and *eye pass*. Such *bidirectional ray tracing* using adaptive radiosity textures can thus simulate all photon paths, in principle.

Our bidirectional ray tracing algorithm is thus a hybrid. From radiosity we borrow the idea of saving and reusing the diffuse component, which is view-independent, and from ray tracing we borrow the idea of discarding and recomputing the specular component, which is view-dependent.

5.2 Sampling

There are three separate multidimensional sampling processes involved in this approach: sampling of directions from the light, sampling of directions from the eye (screen sampling), and sampling of radiosity on each diffuse surface (figure 5.3). All sampling is adaptive.

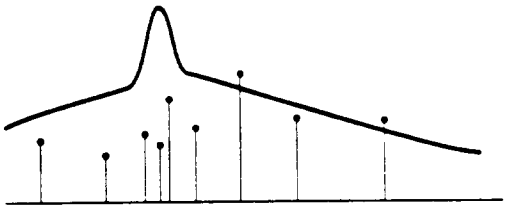

Figure 5.4: *Photons incident on a rex (shown as spikes with height proportional to power) are samples from the true, piecewise-continuous radiosity function (curve), which is the probability density of a continuous random variable. We try to estimate the function from the samples.*

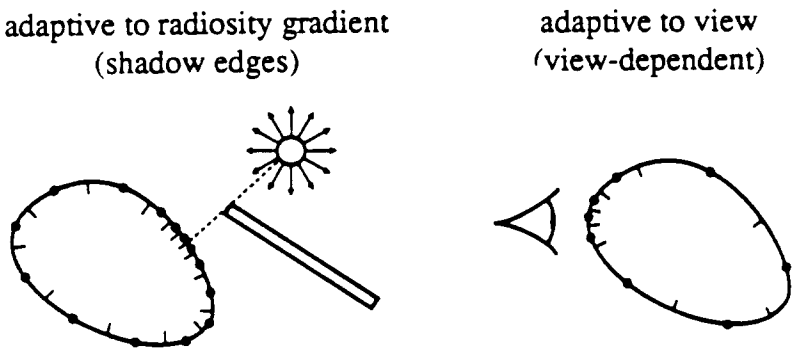

Figure 5.5: *Radiosity textures should be adaptive to both the gradient of the radiosity function and to the view.*

5.2.1 Adaptive Radiosity Textures (Rexes)

Rexes are textures indexed by surface parameters u and v, as in standard texture mapping [Blinn-Newell76,Heckbert86]. We associate a rex with every diffuse or partially-diffuse surface. By using a texture and retaining the initial geometry, instead of polygonizing, we avoid the polygonized silhouettes of curved surfaces common in radiosity pictures that do not employ adaptive meshing.

In the bidirectional ray tracing algorithm, the rexes collect power from incident photons during the light pass, and this information is used to estimate the true radiosity function during the eye pass (figure 5.4). Our rexes thus serve much like *density estimators* that estimate the probability density of a random variable from a set of samples of that random variable [Silverman86]. Density can be estimated using either histogram methods, which subdivide the domain into buckets; or kernel estimators, which store every sample and reconstruct the density as a sum of weighted kernels (similar to a spline).

The resolution of a rex should be related to its screen size (figure 5.5). Ideally, we want to resolve shadow edges sharply in the final picture, which means that rexes should store details as fine as the preimage of a screen pixel. On the other hand, resolution of

details smaller than this is unnecessary, since subpixel detail is beyond the Nyquist limit of screen sampling. Cohen's substructuring technique is adaptive, but its criteria appear to be independent of screen space, so it cannot adapt and optimize the radiosity samples for a particular view.

To provide the light pass with information about rex resolution we precede the light pass with a *size pass* in which we trace rays from the eye, labeling each diffuse surface with the minimum rex feature size.

5.2.2 Adaptive Light Sampling

Adaptive sampling of light rays is desirable for several reasons. Sharp resolution of shadow edges requires rays only where the light source sees a silhouette. Also, it is only necessary to trace light paths that hit surfaces visible (directly or indirectly) to the eye. Thirdly, omnidirectional lights disperse photons in a sphere of directions, but when such lights are far from the visible scene, as is the sun, the light ray directions that affect the final picture subtend a small solid angle. Finally, stratified sampling should be used for directional lights to efficiently simulate the directional dependence of their emissivity. Thus, to avoid tracing irrelevant rays, we sample the sphere of directions adaptively [Sillion-Puech89,Wallace et al. 89].

For area light sources, we use stratified sampling to distribute the ray origins across the surface with a density proportional to the local radiosity. Stratified sampling should also be used to shoot more light rays near the normal, while the flux from a small element is proportional to the cosine of the angle with the normal. If the surface has both a standard texture and a rex mapped onto it, then the rex should be modulated by this standard texture before shooting. With area light sources, the distribution to be integrated is thus four-dimensional: two dimensions for surface parameters u and v, and two dimensions for ray direction. For best results, a 4-D data structure such as a k-d tree should be used to record and adapt the set of light rays used.

5.2.3 Adaptive Eye Sampling

Eye rays (screen pixels) are sampled adaptively as well. Techniques for adaptive screen sampling have been covered well by others [Warnock69,Whitted80,Mitchell87,Painter-Sloan89].

5.3 Three Pass Algorithm

Our bidirectional ray tracing algorithm thus has three passes. We discuss these passes here in a general way; the details of a particular implementation are discussed in §5.4. The passes are:

size pass – record screen size information in each rex

light pass – progressively trace rays from lights and bright surfaces, depositing photons on diffuse surfaces to construct radiosity textures

eye pass – trace rays from eye, extracting light from diffuse surfaces to make a picture

Specular reflection and transmission bounces are followed on all three passes. Distribution ray tracing can be used in all passes to simulate the broad distributions of rough specular reflections and other effects.

Size Pass

As previously described, the size pass traces rays from the eye, recording information in the rexes about the mapping between surface parameter space and screen space. This information is used by each rex during the light pass to terminate its adaptive subdivision.

Light Pass

Indirect diffuse scatter is simulated during the light pass by regarding bright diffuse surfaces as light sources as in progressive radiosity [Cohen et al. 88], and shooting light rays from them. The rex records the shot and unshot power.

The adaptive algorithm for light ray tracing must ensure that: (a) a minimum level of light sampling is achieved; (b) more rays are devoted near silhouettes, shadows, and high curvature areas; (c) sharp radiosity gradients are resolved to screen pixel size; and (d) light rays and rexes are subdivided cooperatively. Because rexes subdivide as the solution evolves, we consider them an a posteriori mesh.

Eye Pass

The eye pass is like a standard ray tracing algorithm except that the diffuse intensity is extracted out of the rex, instead of from a shadow ray. The radiosity of a surface patch is its power divided by its world-space surface area.

After the three passes are run, one could move the eye point and re-run the eye pass to generate other views of the scene, but the results would be inferior to those made by recomputing the rexes adapted to the new viewpoint.

Observations

Because light rays are concentrated on visible portions of the scene and radiosity is resolved adaptive to each surface's projection in screen space, the radiosity calculation performed in the light pass is view-dependent. But this is as it should be: although the exact radiosity values are view-independent, the radiosity sample locations needed to make a picture are not. When computing moving-camera animation, one could prime the rexes by running the size pass for selected key frames to achieve more view-independent sampling.

5.4 Implementation

The current implementation realizes many, but not all, of the ideas proposed above. It performs bidirectional ray tracing using adaptive sampling for light, eye, and rex. It has no size pass, just a light pass and an eye pass. The program can render scenes consisting of constructive solid geometry (CSG) combinations of spheres and polyhedra. Specular scattering is assumed ideal, and diffuse transmission is not simulated. The light pass shoots

photons from omnidirectional point light sources, and does not implement progressive radiosity. The implementation thus simulates only LS^*DS^*E paths at present. We trace ray trees, not just ray paths [Kajiya86].

Data Structures

Quadtrees were used for each of the 2-D sampling processes [Samet90]: one for the outgoing directions of each light, one for the parameter space of each radiosity texture, and one for the eye (figure 5.3).

The light and eye quadtrees are quite similar; their records are shown below in pseudocode. Each node contains pointers to its child nodes (if not a leaf) and to its parent node. Light space is parameterized by (r, s), where r is longitude and s is latitude, and eye space (screen space) is parameterized by (x, y). Each node represents a square region of the parameter space whose corner is given by (r_0, s_0) or (x_0, y_0) and whose size is proportional to 2^{-level}.

The light quadtree sends one light ray per node at a location uniformly distributed over the square. Also stored in each light quadtree node is the ID of the surface hit by the light ray, if any, and the surface parameters (u, v) at the intersection point. This information is used to determine the distance in parameter space between rex hits.

Eye quadtrees are simpler. Each node has pointers to the intensities at its corners. These are shared with neighbors and children. Eye ray tracing is adaptive but not stochastic.

A rex quadtree node represents a square region of (u, v) parameter space on a diffuse surface. (Note in figure 5.3, center, that quadtree-style subdivision can never resolve discontinuities as directly as a priori discontinuity meshing, discussed earlier. It is a simpler data structure, however.) Leaves in the rex quadtree act as histogram buckets, accumulating the number of photons and their power. We often use the terms "rex node" and "bucket" interchangeably. Rex nodes also record the world space surface area of their surface patch.

```
light_node: type =                  {LIGHT QUADTREE NODE}
   record
      leaf: boolean;                {is this a leaf?}
      mark: boolean;                {should node be split?}
      level: int;                   {level in tree (root=0)}
      parent: ^light_node;          {parent node, if any}
      nw, ne, se, sw: ^light_node;  {four children, if not a leaf}
      r0, s0: real;                 {params of corner of square}
      r, s: real;                   {dir. params of ray (lon,lat)}
      surfno: int;                  {id of surface hit, if any}
      u, v: real;                   {surf params of surface hit}
   end;

eye_node: type =                    {EYE QUADTREE NODE}
   record
      leaf: boolean;                {is this a leaf?}
      mark: boolean;                {should node be split?}
      level: int;                   {level in tree (root=0)}
      parent: ^eye_node;            {parent node, if any}
      nw, ne, se, sw: ^eye_node;    {four children, if not a leaf}
      x0, y0: real;                 {coords of corner of square}
      inw, ine, ise, isw: ^color;   {intensity samples at corners}
   end;
```

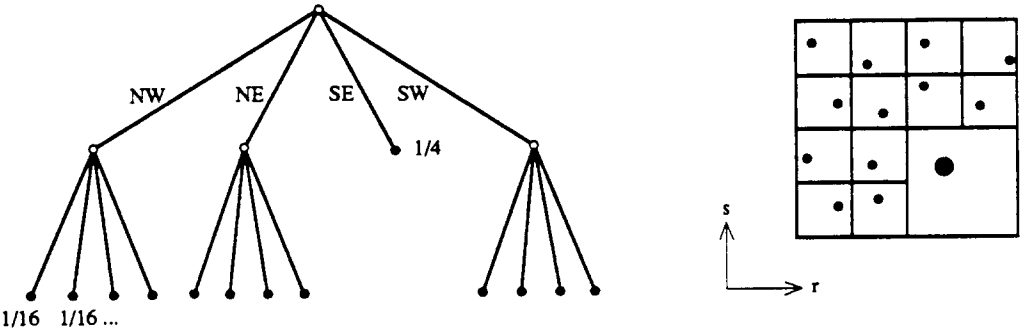

Figure 5.6: *Light quadtree shown schematically (left) and in light direction parameter space (right). When a light quadtree node is split, its power is redistributed to its four sub-nodes, which each send a ray in a direction (r,s) jittered within their parameter square. The fractional power of each light ray is shown next to the leaf node that sends it.*

```
rex_node: type =                    {REX QUADTREE NODE}
   record
      leaf: boolean;                {is this a leaf?}
      mark: boolean;                {should node be split?}
      level: int;                   {level in tree (root=0)}
      parent: ^rex_node;            {parent node, if any}
      nw, ne, se, sw: ^rex_node;    {four children, if not a leaf}
      u0, v0: real;                 {surf params of square corner}
      area: real;                   {surface area of this bucket}
      count: int;                   {#photons in bucket, if leaf}
      power: color;                 {accumulated power of bucket}
   end;
```

The implementation uses the following algorithm.

5.4.1 Light Pass

First, rex quadtrees are initialized to a chosen starting level (level 3, say, for 8x8 subdivision), and the counts and powers of all leaves are zeroed.

For each light, light ray tracing proceeds in breadth first order within the light quadtree, at level 0 tracing a single ray carrying the total power of the light, at level 1 tracing up to 4 rays, at level 2 tracing up to 16 rays, etc (figure 5.6). At each level, we adaptively subdivide both the light quadtree and the rex quadtrees. Changing the rex quadtrees in the midst of light ray shooting raises what we call the histogram redistribution problem, however: if a histogram bucket is split during collection, it is necessary to redistribute the parent's power variable among the children. Since no record is kept of each photon's location, only totals, there is no way to do this reliably without a priori knowledge, so we clear the rex at the beginning of each level and reshoot. This is an expensive solution, but effective.

Processing a given level k of light rays involves three steps: (1) rex subdivision to split rex buckets containing a high density of photons, (2) light marking to mark light quadtree nodes where more light rays should be sent, and (3) light subdivision to split marked light nodes.

Rex subdivision consists of a sweep through every rex quadtree in the scene, splitting all rex buckets whose photon count exceeds a chosen limit. Buckets with a high count arise because light sources shoot a high density of light rays near silhouette curves and other changes in visibility. Since changes in visibility are a necessary condition for a shadow boundary, we subdivide the rex in such areas. All counts and powers in the rexes are zeroed at the end of this sweep.

Light marking traverses the light quadtree, marking all level k nodes that meet the following subdivision criteria:

(1) Always subdivide until a minimum level is reached.

(2) Never subdivide beyond a maximum level (if a size pass were implemented, it would determine this maximum level locally).

Otherwise, look at the light quadtree neighbors above, below, left, and right, and subdivide if the following is true:

(3) The ray hit a diffuse surface, and one of the four neighbors of the light node hit a different surface or was beyond a threshold distance in (u, v) parameter space from the center ray's intersection point.

To help prevent small feature neglect, we also mark for subdivision all level $k-1$ leaves that neighbor on level k leaves that are marked for subdivision. This last rule guarantees a *restricted quadtree* [Von Herzen-Barr87] where each leaf node's neighbors are at a level within plus or minus one of the center node's.

Light subdivision traverses the light quadtree splitting the marked nodes. Subdividing a node splits a ray of power p into four rays of power $p/4$ (figure 5.6). When a light node is created (during initialization or subdivision) we select a point at random within its square (r, s) domain to achieve jittered sampling [Cook86] and trace a ray in that direction. Marked nodes thus shoot four new rays, while unmarked nodes re-shoot their rays. During light ray tracing we follow specular bounces, splitting the ray tree and subdividing the power according to the reflectivity/transmissivity coefficients k_{ij}, and deposit the power on any diffuse surfaces that are hit. When a diffuse surface is hit, we determine the (u, v) of the intersection point, and descend the surface's rex quadtree to find the rex node containing that point. The power of that node is incremented by the power of the ray.[2]

5.4.2 Eye Pass

The eye pass is a fairly standard adaptive supersampling ray tracing algorithm: nodes are split when the intensity difference between the four corners exceeds some threshold. To generate a picture, nodes larger than a pixel perform bilinear interpolation to fill in the pixels they cover, while nodes smaller than a pixel are averaged together to compute a pixel. The picture is stored in floating point format initially, then scaled and clamped to the range [0,255] in each of the channels: red, green, and blue.

5.5 Results

[2] The original implementation incremented the power of the node by the power of the ray times the cosine of the incident angle [Heckbert90], but Nelson Max pointed out that this was nonphysical.

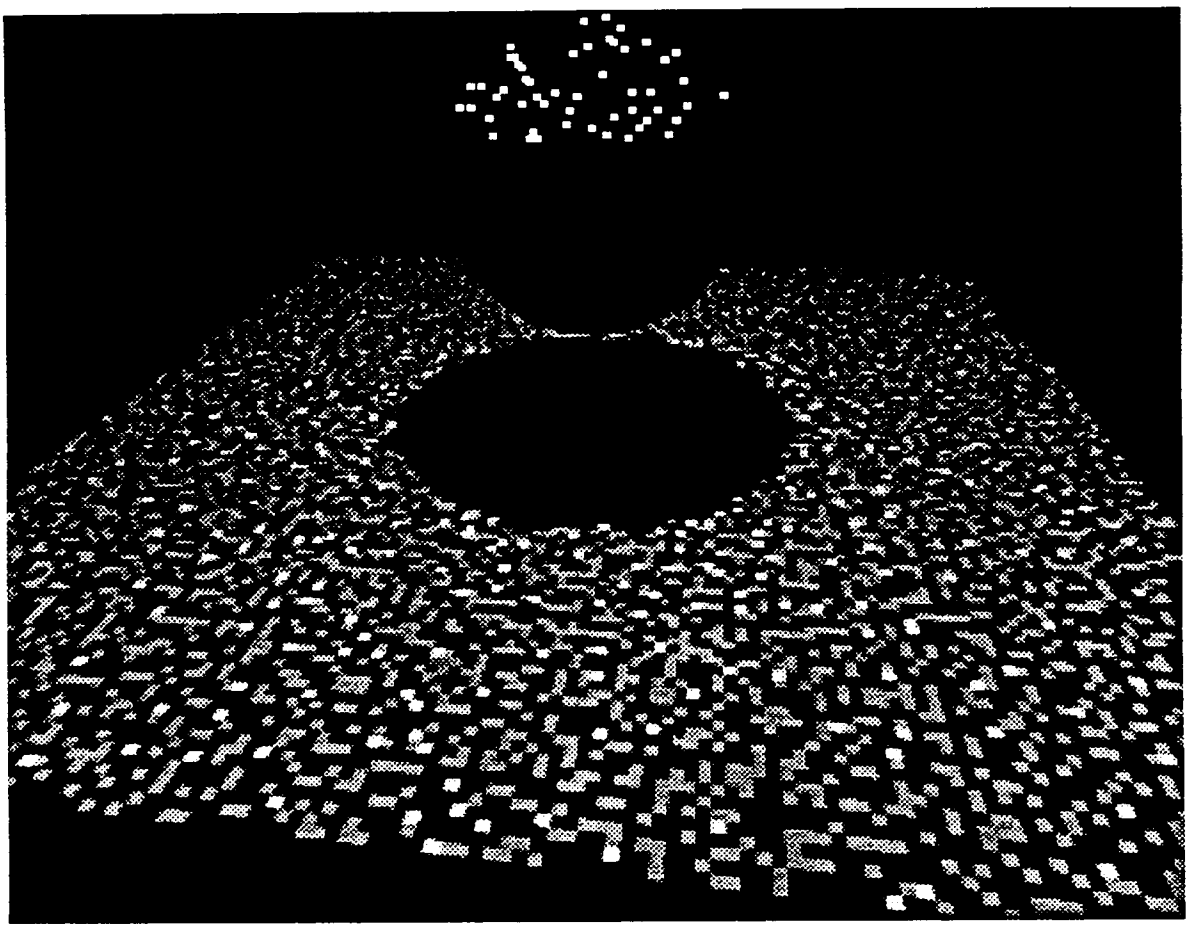

Figure 5.7: *Noisy appearance results when too few light rays are received in each rex bucket (too few light rays or too fine a rex). Scene consists of a diffuse sphere above a diffuse floor both illuminated by an overhead light source.*

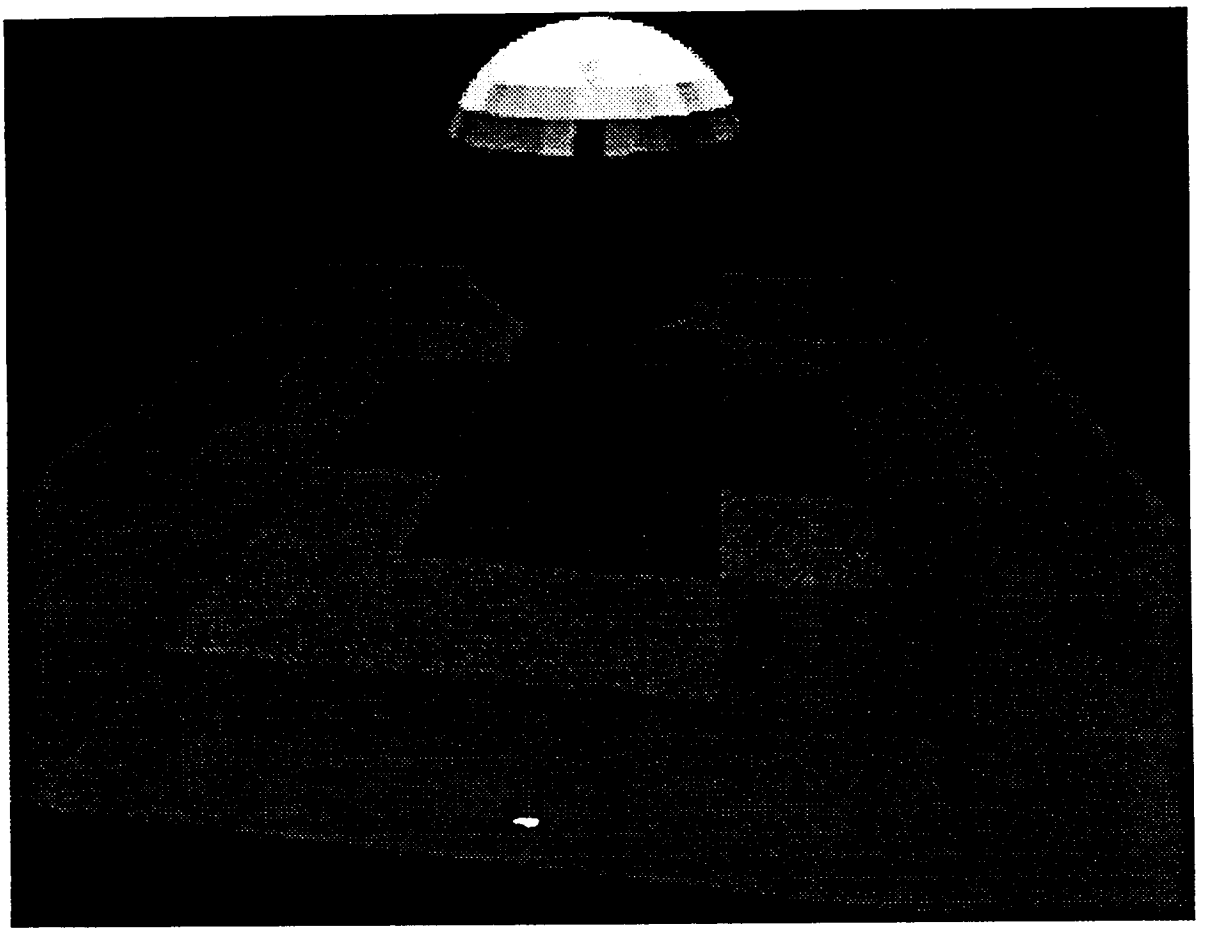

Figure 5.8: *Blocky or blurry appearance results when rex buckets are much larger than a screen pixel (too coarse a rex).*

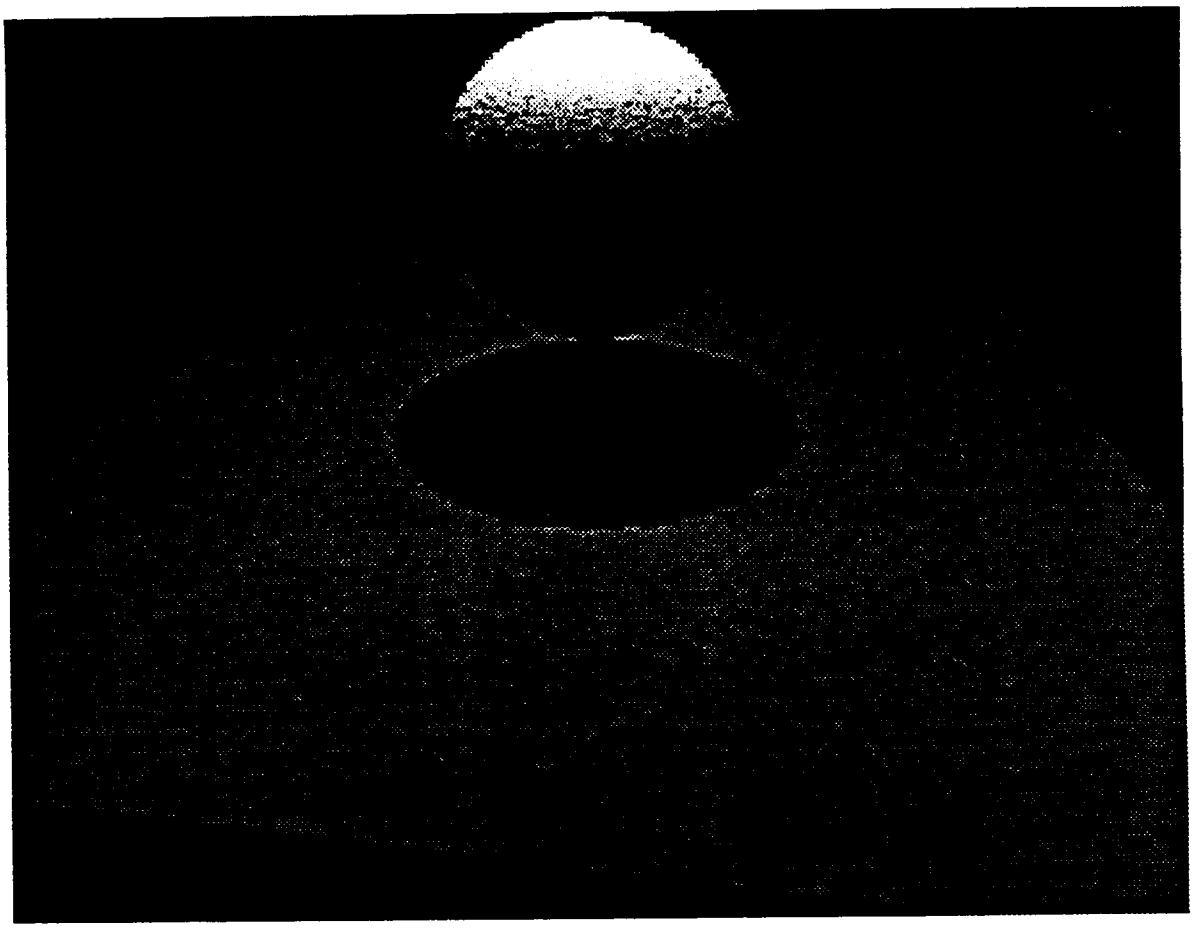

Figure 5.9: *Proper balance of light sampling and rex sampling reduces both noise and blockiness.*

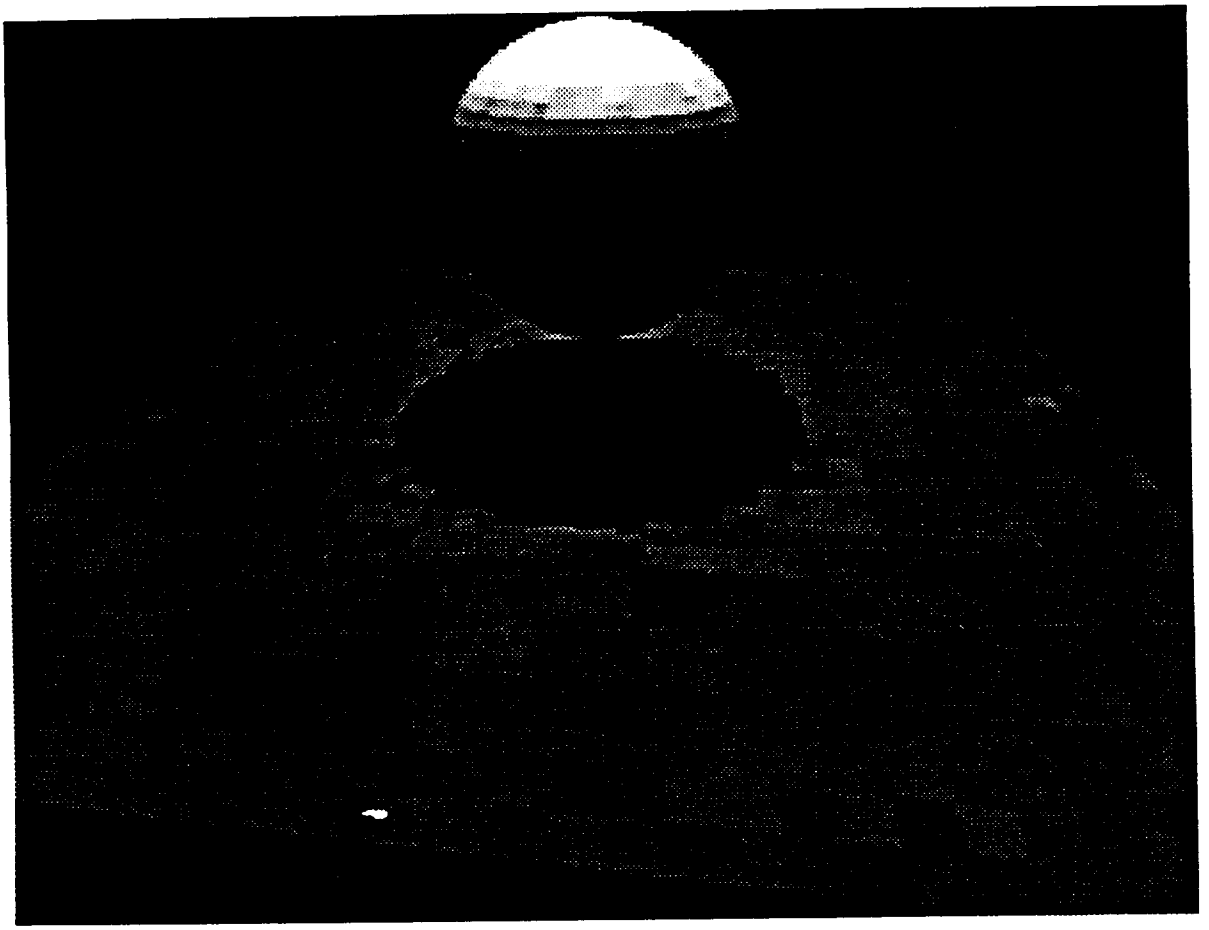

Figure 5.10: *Rex with adaptation: the rex of the floor is initially a single bucket, but it splits adaptively near the edges of the square and near the shadow edge.*

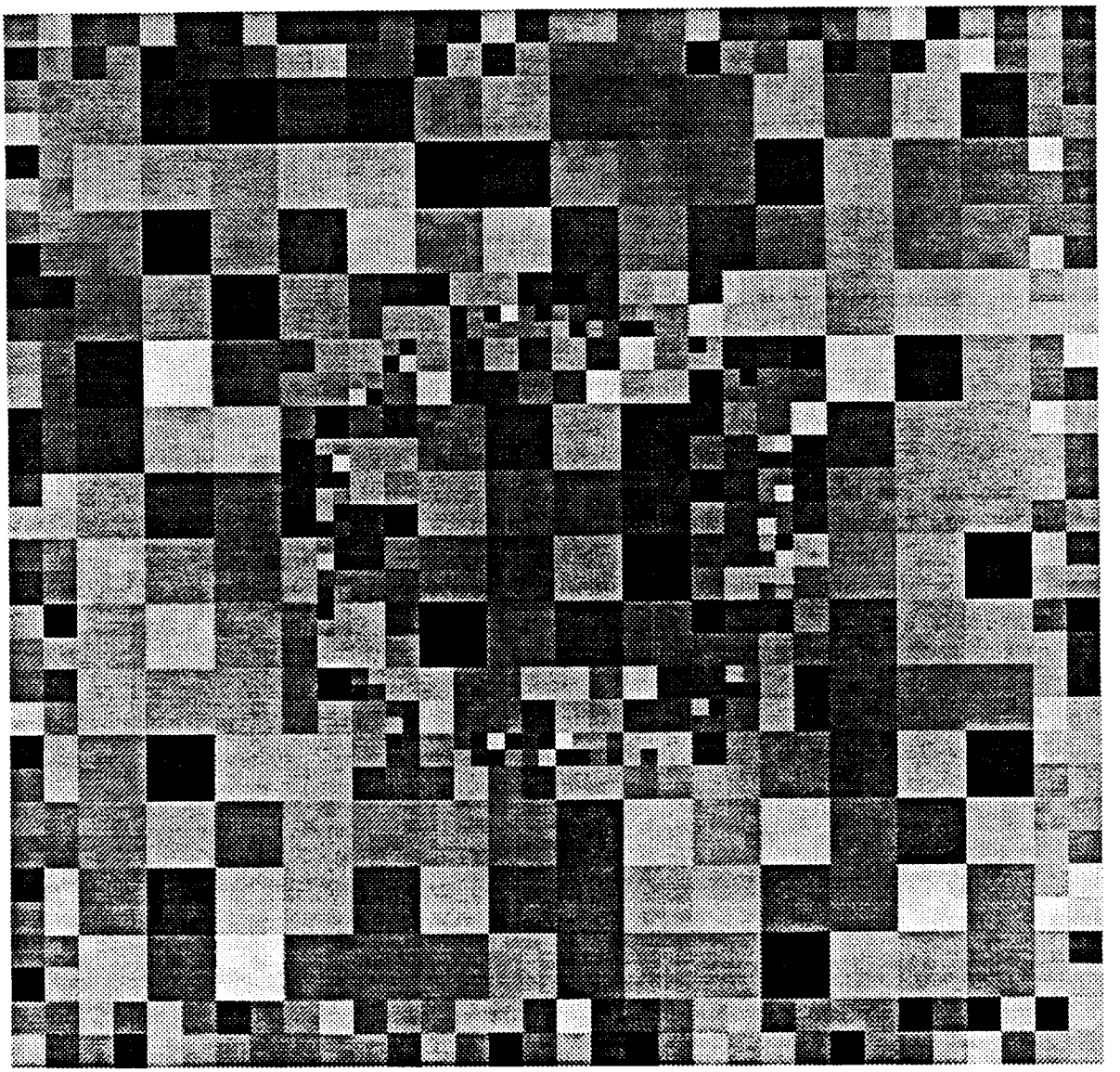

Figure 5.11: *Rex quadtree in (u, v) space of previous figure's floor. Each leaf node's square is colored randomly. Note the subdivision near the shadow edge and the quadtree restriction.*

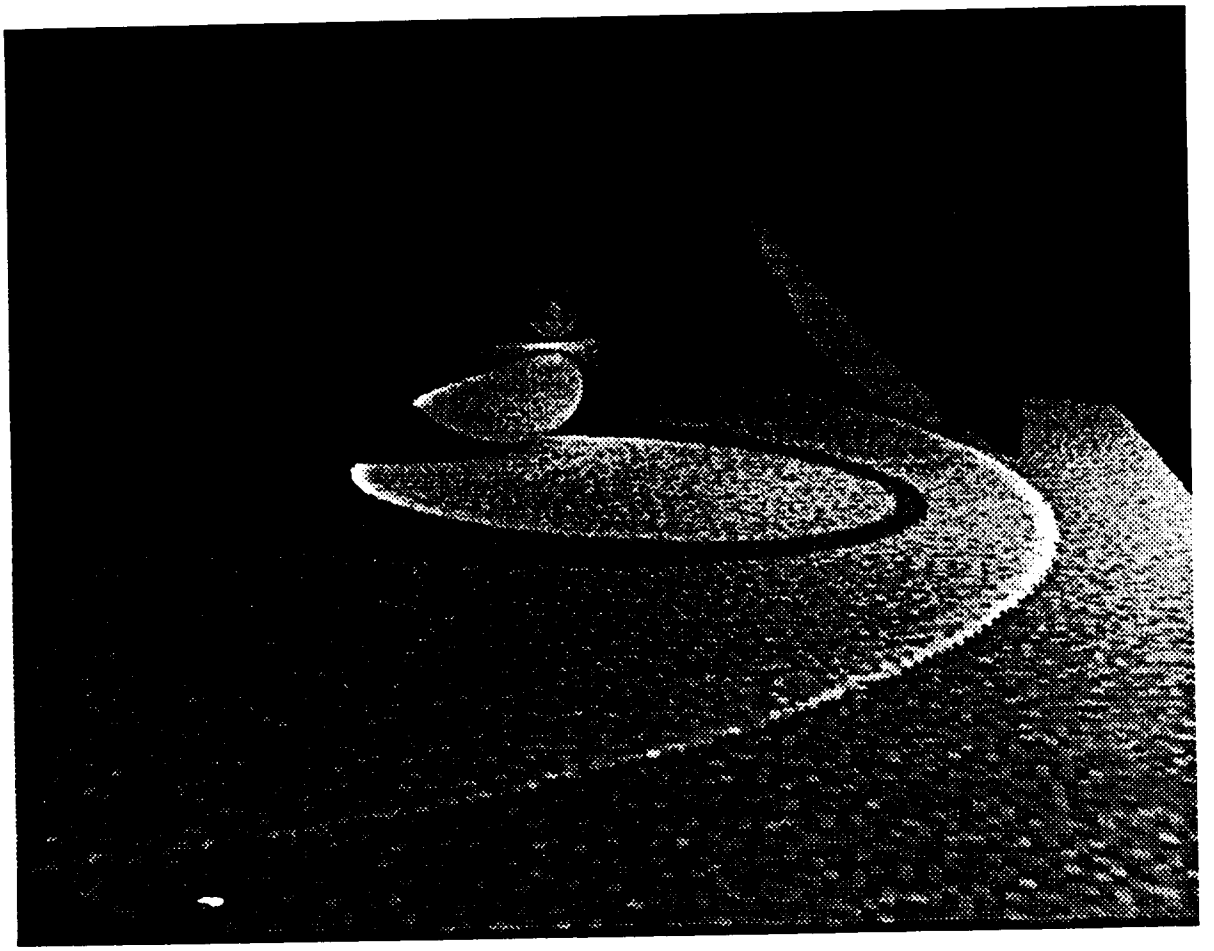

Figure 5.12: *Light focusing and reflection from a lens and chrome ball. Scene is a glass lens formed by CSG intersection of two spheres, a chrome ball, and a diffuse floor, illuminated by a light source off screen to the right. Note focusing of light through lens onto floor at center (an $LSSD$ path), reflection of refracted light off ball onto floor (an $LSSSD$ path involving both transmission and reflection), the reflection of light off lens onto floor forming a parabolic arc (an LSD path), and the reflection of the lens in the ball (an $LSSDSSE$ path, in full).*

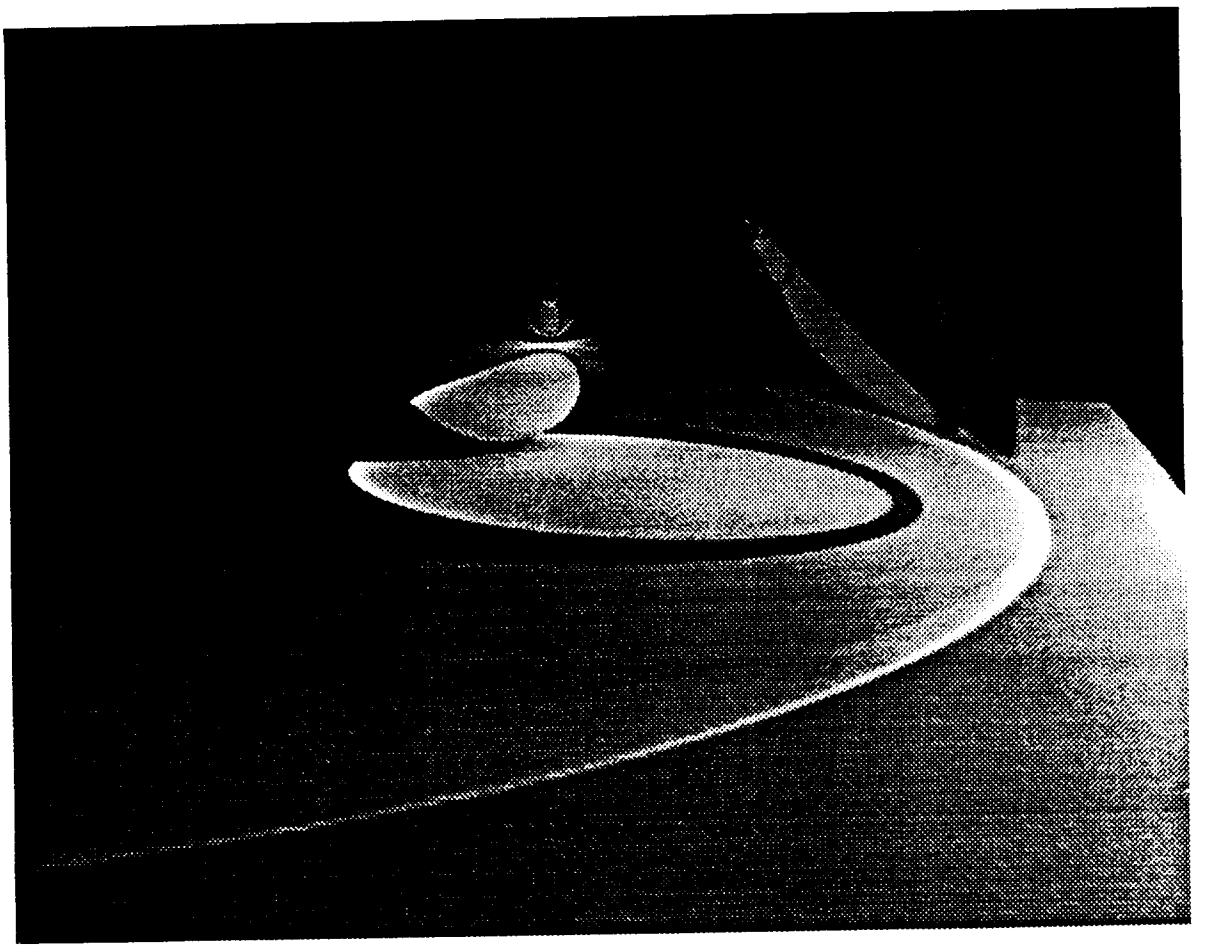

Figure 5.13: *Lens scene with more light and eye rays. Six times as many light rays and 14 times as many eye rays were used as for the previous figure.*

Figures 5.7-5.13 were generated with this program. Figures 5.7, 5.8, and 5.9 show the importance of coordinating the light ray sampling process with the rex resolution. Sending too few light rays results in a noisy radiosity estimate from the rex, and too coarse a rex results in blocky appearance. When the rex buckets are approximately screen pixel size and the light ray density deposits several photons per bucket (at least 10, say), the results are satisfactory. We estimate the radiosity using a function that is constant within each bucket; this simple estimator accounts for the blockiness of the images. If Gouraud interpolation were used, as in most radiosity programs, we could trade off blockiness for blurriness.

Figure 5.10 shows adaptive subdivision of a rex quadtree, splitting more densely near shadow edges (the current splitting criteria cause unnecessary splitting near the border of the square). Its rex quadtree is shown in figure 5.11.

Figures 5.12 and 5.13 show off some of the effects that are simulated by this algorithm.

The first of these was computed using a straightforward breadth-first traversal of the light quadtree, and required about 20 megabytes. With simple breadth-first traversal of light and eye quadtrees, the memory requirements are proportional to the number of light samples plus the number of eye samples. Consequently, it becomes impractical to generate high resolution, low noise pictures except on machines with hundreds of megabytes of memory. Simple breadth-first traversal yielded a memory-limited algorithm.

To allow more light and eye rays to be sent, scanline traversal of the light and eye quadtrees was implemented. Rather than store the entire light or eye quadtree simultaneously, only a moving strip is kept in memory at one time. For light quadtrees, a "scanline" is a line of constant s (latitude) in rs space, while for eye quadtrees, a scanline is a line of constant y in xy space. By improving the locality of reference of the algorithm in this way, the memory requirements were reduced to be roughly proportional to the sum of the square roots of the number of light rays and the number of eye rays. Figure 5.13 used only about 6 megabytes, even though it shot far more light rays and eye rays than figure 5.12.

Statistics for these images are listed below, including the number of light rays, the percentage of light rays striking an object, the resolution of the rex, the resolution of the final picture, the number of eye rays, and the CPU time. All images were computed on a MIPS R2000 processor, except figure 5.13, which was generated on a Sparcstation. CPU times are stated for the computer on which the computation was done. Ray trees were traced to a depth of 5.

#LRAYS	%HIT	REX	EYE	#ERAYS	TIME	FIG
87,400	10%	128^2	256^2	246,000	1.0 min.	fig. 5.7
87,400	10%	8^2	256^2	139,000	0.6 min.	fig. 5.8
822,000	68%	128^2	256^2	146,000	3.5 min.	fig. 5.9
331,000	20%	vbl	256^2	139,000	1.3 min.	fig. 5.10
1,080,000	61%	256^2	512^2	797,000	6.4 min.	fig. 5.12
6,460,000	82%	512^2	1024^2	11,100,000	205 min.	fig. 5.13

5.6 Conclusions

The bidirectional ray tracing algorithm outlined here appears to be a general, potentially accurate approach for global illumination of scenes consisting of diffuse and pure specular surfaces. It can be accurate because it can account for all possible light paths; and it is general because it supports both the radiosity and ray tracing realms: shapes both planar and

curved, materials both diffuse and specular, and lights both large and small. Distribution ray tracing can be used to simulate effects not directly supported by the algorithm.

The adaptive radiosity texture (rex) is a new data structure that has several advantages over previous radiosity storage schemes. It can adaptively subdivide itself to resolve sharp shadow edges to screen pixel size, thereby eliminating visible artifacts of radiosity sampling. Its subdivision can be automatic, requiring no ad hoc user-selected parameters.

The current implementation is only a partial exploration of the approach presented at the beginning of this chapter, and many problems remain. A brief list follows: Good adaptive sampling of area light sources appears to require a 4-D data structure. Better methods are needed to determine the number of light rays. The redistribution problems of histograms caused us to send each light ray multiple times. To avoid this problem we could store all (or selected) photon locations using kernel estimators [Silverman86]. Excessive memory is currently devoted to the light quadtree, since one node is stored per light ray. Perhaps the quadtree could be subdivided in more-or-less (r,s) scanline order, and the memory recycled (quadtree restriction appears to complicate this, however). Adaptive subdivision algorithms that compare the ray trees of neighboring rays do not mix easily with path tracing and distribution ray tracing, because the latter obscure coherence. Last but not least, the interdependence of light ray subdivision and rex subdivision is precarious.

In spite of these challenges, we are hopeful. The approach of bidirectional ray tracing using adaptive radiosity textures appears to contain the mechanisms needed to simulate global illumination in a general way.

Chapter 6

Conclusions

Phenomena equivalent to global illumination have been studied in many fields. The phenomena of thermal radiation in mechanical engineering, interreflection in lighting design, mutual illumination in computer vision, and global illumination in computer graphics have different applications, their literature is usually independent, and they operate in various bands of the electromagnetic spectrum, but they are nearly identical physically and mathematically. In scenes with non-participating media, these phenomena are governed by an integral equation.

Integral equations provide a concise statement of the physics of global illumination. Global illumination can be studied without reference to integral equations, but such an approach is awkward. Studying global illumination without integral equations is like studying classical dynamics without differential equations.

The kernel of the integral equation of global illumination reflects the geometry and visibility between surface points. Diffuse scenes with no occlusion have smooth, dense kernels whereas specular scenes have non-smooth, sparse kernels.

With occlusions, however, the kernel has discontinuities of value, and these cause discontinuities of first, second, and higher derivatives in the intensity solution function. The most significant of these discontinuities are perceived as shadows. A scene with occlusions can have an infinite number of discontinuities of various degrees. Many other interdependency problems in engineering and science such as the n-body problem [Greengard-Rokhlin87] do not have such discontinuities. One of the principal contributions of this thesis has been a better understanding of discontinuities: their causes, their importance, algorithms for finding them, and adaptive mesh data structures for representing them.

This thesis has explored two general approaches for numerical solution of the integral equation of global illumination: finite element methods, simple application of which gives rise to radiosity algorithms, and Monte Carlo methods which employ ray tracing.

In diffuse 3-D scenes, intensity is direction-independent, so it is a function of two-dimensional surface position only. The low dimensionality of the intensity function allows it to be stored in memory, and the integral equation can easily be approximated by a system of linear equations.

In scenes containing specular surfaces, intensity is a function of four dimensions, in general: two for surface position and two for direction. With so many unknowns, it is impractical to store a representation of the intensity function for the entire scene. Instead, it is computed on the fly using Monte Carlo techniques, with ray tracing as the visibility

algorithm.

6.1 Finite Element/Radiosity Algorithms

The radiosity method was initially developed in the field of thermal radiation in the 50's. The field of computer graphics has adapted this method to visible wavelengths, employing new techniques for numerical integration of visibility in complex scenes (the use of hemicubes and ray tracing for form factor calculation, for example).

Computer graphics has not learned all that it could from thermal radiation, however. Because few computer graphicists have delved into the thermal radiation literature, there has generally been a poor understanding of the derivation and assumptions of the radiosity method in the graphics community. In particular, the use of uniform meshes and constant-intensity elements has not been sufficiently questioned. By going back to primary sources in thermal radiation [Hottel54,Sparrow63], we have learned more about the limitations and approximations of the radiosity method.

This thesis has attempted to put the radiosity method in perspective by showing that it is one of several approximation methods that can be derived from the integral equations of thermal radiation and global illumination.

6.1.1 Two Dimensions

A fairly thorough study of radiosity in a diffuse, two-dimensional "flatland" world was made in order to better understand these integral equations and their solution functions. The solution functions and the kernel are easy to visualize in flatland.

Radiosity solution functions are seen to have discontinuities caused by touching surfaces, intersections, creases, or occlusion. The kernel of the integral equation is singular at reflex corners, where touching edges face each other. These discontinuities and singularities suggest the need for adaptive meshing.

It was demonstrated that the classic radiosity method constitutes one of the simplest finite element methods for simulating global illumination in a diffuse scene. Classic radiosity typically makes use of constant elements, a uniform mesh, collocation methods for discretization, a hemicube for numerical integration, and Gauss-Seidel or progressive iterative methods for solving linear systems. It was shown that the use of constant elements during problem formulation and solution followed by Gouraud interpolation for display gives results that are objectively no more accurate than the use of constant elements throughout (though they may look better).

We have explored several improvements on these approaches, demonstrating that both higher degree elements and a priori adaptive meshes can improve the accuracy of 2-D radiosity algorithms considerably.

The most prominent discontinuities in the solution can be predicted using visibility methods from computational geometry. With this discontinuity meshing, mesh boundaries are placed on significant discontinuities. Using linear elements and discontinuity meshing, one experiment in flatland achieved a 60-fold reduction in CPU time and memory use relative to standard methods.

In retrospect, study of flatland radiosity was a very fertile research topic, facilitating understanding, visualization, analysis, and experimentation.

6.1.2 Three Dimensions

Extensions of these algorithms to 3-D have been suggested, but not yet implemented. Discontinuity meshing is more complex in 3-D than in 2-D. In 3-D, the number of discontinuities is combinatorially larger, the mesh topology is more complex, the degree of most discontinuities is higher, and discontinuity curves can be both linear and conic. Most of the algorithms needed to implement a 3-D radiosity program with discontinuity meshing and higher order elements exist, however. The computation of some discontinuity curves can be done using a 3-D visible surface algorithm, analogous to repeated application of a shadow algorithm from the point of view of each secondary light source. Others have shown that discontinuities caused by touching or intersecting surfaces can be resolved in complex scenes [Baum et al. 91], and that shadow discontinuities can be computed for scenes of moderate complexity [Nishita-Nakamae83,Campbell-Fussell90]. These are important first steps toward algorithms that deal with discontinuities in a robust way.

6.1.3 Future Experiments

A number of experiments are needed to determine the best combination of element degree (linear, quadratic, etc.), discontinuity significance criterion (to ignore negligible discontinuities), reflex corner meshing rules, discretization method (collocation or Galerkin), integration method (analytic, hemicube, Gaussian, or other), and linear system solution method (successive overrelaxation, conjugate gradient, multigrid, progressive, etc.).

Ideally, we would like algorithms with a user-selected error tolerance, so that users willing to tolerate 10% error get fast results, but users insisting on .1% error must wait longer. To achieve this goal, further error analysis is needed in order to bound the error of each approximation stage, including basis function, mesh, numerical integration, and systems solving. The eigenvalues, eigenfunctions, and eigenvectors of the continuous and discrete kernels should be studied.

There are other experiments and generalizations to be explored: Further experiments are needed to compare the efficiency and accuracy of the a priori adaptive mesh methods discussed here with a posteriori adaptive mesh methods [Cohen et al. 86]. Combinations of these ideas with recent techniques that adaptively sample the kernel to exploit its smoothness [Hanrahan et al. 91] and data structures that compactly represent directional diffuse intensity functions [Sillion et al. 91] also appear promising.

It might be fruitful to perform many of these experiments in flatland before generalizing to 3-D. For example, one could generalize the simple flatland world discussed here to curved surfaces, arbitrary scattering, and participating media, and make a very interesting and thorough study of flatland global illumination.

Exploring all of these ideas will be exciting research.

6.2 Monte Carlo/Ray Tracing Algorithms

The Monte Carlo approach to the simulation of global illumination has been explored as well. Monte Carlo approaches are best suited to entirely specular scenes, but they are general enough to be used on scenes with arbitrary scattering functions.

Distribution ray tracing can accommodate arbitrary scattering, but to simulate diffuse interreflection they often must trace hundreds or thousands of rays in order to integrate the

incident intensity from all directions. This is very expensive. Fortunately, intensity can be split into a diffuse part and a specular part, and the diffuse component (the radiosity) is seen to be very smooth. The diffuse component can thus be sampled sparsely across each surface [Ward et al. 88].

Rather than recompute the diffuse component repeatedly, we store it for later re-use in a radiosity texture on each surface in the scene. For efficiency, this radiosity texture should have resolution adaptive to intensity gradient, so that shadows can be resolved sharply but memory is not wasted on smooth areas, and adaptive to screen resolution, so that there is a roughly equal number of radiosity samples contributing to each pixel. Computing any more samples than this would be a waste of time. Traditional polygonized radiosity methods, however, employ a fixed, pre-ordained, non-adaptive, view-independent mesh. In contrast, we have developed an a posteriori adaptive radiosity texture, implemented using a quadtree, that is adaptive to intensity gradient and screen size, and hence view-dependent.

We have explored an algorithm which is a hybrid of ray tracing algorithms for visibility testing and data structures for storing radiosity. Rays are traced both from the lights and the eye in order to simulate light propagation by all possible paths. Rays from the lights act like photons, bouncing at specular surfaces, but stopping at diffuse surfaces to deposit their power in a radiosity texture. Direct illumination can be simulated by shooting millions of such light rays from the light sources. Diffuse interreflection can then be simulated by shooting light rays from the most brightly lit diffuse surfaces, analogous to a progressive radiosity algorithm [Cohen et al. 88]. Finally, a picture can be generated by tracing rays from the eye, determining the diffuse component from the radiosity textures.

This algorithm has been implemented and demonstrated on scenes consisting of planar and curved surfaces with scattering characteristics ranging from diffuse reflection to specular transmission. The focusing of light through transparent, refracting objects has been simulated, an effect difficult to achieve with most ray tracers. The computation times were reasonable but memory requirements were high.

The general concepts of the bidirectional ray tracing algorithm and the adaptive radiosity texture appear quite promising, but the adaptive subdivision criteria discussed here in §5.4 need more work. Further study is needed to understand the complex interaction between the light ray sampling grid and the radiosity texture sampling grids. Or perhaps a simpler, more brute force approach can be found that requires fewer cooperative sampling processes. Overall, the future prospects for the bidirectional ray tracing approach do not seem as promising as those of finite element methods for radiosity. On the bright side, Shirley has made some beautiful pictures with similar techniques [Shirley90,Shirley91]. Pencil tracing offers another alternative that might allow some of the sampling issues encountered here to be overcome [Shinya et al. 89]. A hybrid technique that builds on some of the work here has yielded some very nice pictures [Chen et al. 91].

6.3 Finale

New techniques for global illumination can be discovered if we in the computer graphics community step back from the somewhat ad hoc radiosity and ray tracing algorithms with which we are familiar and identify the abstract problems that we are trying to solve. Following the example set by Sutherland, Sproull, and Schumacker in their taxonomy of visibility algorithms [Sutherland et al. 74], and by Kajiya with his rendering equation [Kajiya86],

we have attempted to interrelate algorithms by re-deriving them from their mathematical essence. By regarding global illumination as an integral equation problem, we have discovered a variety of approximation methods, some of which are identical to existing algorithms, and many of which are new.

Bibliography

[Abbott84] Edwin A. Abbott. *Flatland: A Romance of Many Dimensions.* Dover, New York, 1884.

[Alpert90] Bradley Alpert. Sparse representation of smooth linear operators. Technical report, CS Dept, Yale U, Aug 1990. DCS/RR-814.

[Amanatides84] John Amanatides. Ray tracing with cones. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):129–135, July 1984.

[Appel68] Arthur Appel. Some techniques for shading machine renderings of solids. *AFIPS 1968 Spring Joint Computer Conf.*, 32:37–45, 1968.

[Arvo86] James Arvo. Backward ray tracing. In *SIGGRAPH '86 Developments in Ray Tracing seminar notes*, Aug. 1986.

[Arvo-Kirk90] James Arvo and David Kirk. Particle transport and image synthesis. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):63–66, August 1990.

[Atherton et al. 78] Peter R. Atherton, Kevin Weiler, and Donald P. Greenberg. Polygon shadow generation. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):275–281, Aug. 1978.

[Atkinson76] Kendall E. Atkinson. *A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1976.

[Bank et al. 83] Randolph E. Bank. Andrew H. Sherman, and Alan Weiser. Refinement algorithms and data structures for regular local mesh refinement. In R. Stepleman et al, editor, *Scientific Computing, IMACS Trans. on Scientific Computation*, volume 1. North-Holland, 1983.

[Bartels et al. 87] Richard Bartels, John Beatty, and Brian Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, San Mateo, CA, 1987.

[Baum et al. 89] Daniel R. Baum, Holly E. Rushmeier, and James M. Winget. Improving radiosity solutions through the use of analytically determined form factors. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):325–334, July 1989.

[Baum et al. 91] Daniel R. Baum, Stephen Mann, Kevin P. Smith, and James M. Winget. Making radiosity usable: Automatic preprocessing and meshing techniques for the generation of accurate radiosity solutions. *Computer Graphics (SIGGRAPH '91 Proceedings)*, July 1991. To appear.

[Becker et al. 81] Eric B. Becker, Graham F. Cary, and J. Tinsley Oden. *Finite Elements: An Introduction*, volume 1. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[Blinn77] James F. Blinn. Models of light reflection for computer synthesized pictures. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2):192–198, Summer 1977.

[Blinn-Newell76] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *CACM*, 19(10):542–547, Oct. 1976.

[Bouville et al. 90] Christian Bouville, Kadi Bouatouch, Pierre Tellier, and Xavier Pueyo. A theoretical analysis of global illumination models. In *Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, pages 53–66, Rennes, France, June 1990.

[Brotman-Badler84] Lynne Shapiro Brotman and Norman I. Badler. Generating soft shadows with a depth buffer algorithm. *IEEE Computer Graphics and Applications*, 4(10):5–24, Oct. 1984.

[Buckalew-Fussell89] Chris Buckalew and Donald Fussell. Illumination networks: Fast realistic rendering with general reflectance functions. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):89–98, July 1989.

[Buckley27] H. Buckley. *Philosophical Magazine*, 4:753, 1927.

[Buckley28] H. Buckley. *Philosophical Magazine*, 6:447, 1928.

[Buckley34] H. Buckley. *Philosophical Magazine*, 17:576, 1934.

[Campbell-Fussell90] A. T. Campbell, III and Donald S. Fussell. Adaptive mesh generation for global diffuse illumination. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):155–164, Aug. 1990.

[Campbell-Fussell91] A. T. Campbell, III and Donald S. Fussell. Analytic illumination with polygonal light sources. Technical report, CS Dept, U of Texas at Austin, April 1991. TR-91-15.

[Chen et al. 91] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. A progressive multi-pass method for global illumination. *Computer Graphics (SIGGRAPH '91 Proceedings)*, July 1991. To appear.

[Chin-Feiner90] Norman Chin and Steven Feiner. Near real-time shadow generation using BSP trees. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):99–106, Aug. 1990.

[Chung88] T. J. Chung. Integral and integro-differential systems. In Minkowycz et al, editor, *Handbook of Numerical Heat Transfer*, pages 579–624. Wiley, 1988.

[Cohen et al. 86] Michael F. Cohen, Donald P. Greenberg, David S. Immel, and Philip J. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, pages 26–35, Mar. 1986.

[Cohen et al. 88] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):75–84, Aug. 1988.

[Cohen-Greenberg85] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: A radiosity solution for complex environments. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):31–40, July 1985.

[Cook et al. 84] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):137–145, July 1984.

[Cook86] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. on Graphics*, 5(1):51–72, Jan. 1986.

[Cook-Torrance82] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Trans. on Graphics*, 1(1):7–24, Jan. 1982.

[Courant-Hilbert37] Richard Courant and David Hilbert. *Methods of Mathematical Physics*, volume 1. John Wiley & Sons, New York, 1937.

[Crow77] Franklin C. Crow. Shadow algorithms for computer graphics. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2), Summer 1977.

[Delves-Mohamed85] L. M. Delves and J. L. Mohamed. *Computational methods for integral equations*. Cambridge University Press, Cambridge, U.K., 1985.

[Dippe-Wold85] Mark A. Z. Dippe and Erling Henry Wold. Antialiasing through stochastic sampling. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):69–78, July 1985.

[Eckert-Drake59] Ernst R. G. Eckert and Robert M. Drake, Jr. *Heat and Mass Transfer*. McGraw-Hill, New York, 1959.

[Eckert-Drake72] Ernst R. G. Eckert and Robert M. Drake, Jr. *Analysis of Heat and Mass Transfer*. McGraw-Hill, New York, 1972.

[Edelsbrunner et al. 83] Herbert Edelsbrunner, Mark H. Overmars, and Derick Wood. Graphics in flatland: A case study. *Advances in Computing Research*, 1:35–59, 1983.

[Edelsbrunner-Guibas89] Herbert Edelsbrunner and Leonidas Guibas. Topologically sweeping an arrangement. *J. Comp. Sys. Sc.*, 38:165–194, 1989.

[Fletcher84] C. A. J. Fletcher. *Computational Galerkin Methods*. Springer-Verlag, New York, 1984.

[Foley et al. 90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice, 2nd ed.* Addison-Wesley, Reading MA, 1990.

[Forsyth-Zisserman89] David Forsyth and Andrew Zisserman. Mutual illumination. In *Proc. Computer Vision and Pattern Recognition (CVPR '89)*, pages 466–473. IEEE Computer Society Press, June 1989.

[Fuchs et al. 80] Henry Fuchs, Zvi M. Kedem, and Bruce F. Naylor. On visible surface generation by a priori tree structures. *Computer Graphics (SIGGRAPH '80 Proceedings)*, 14(3):124–133, July 1980.

[Gebhart61] Benjamin Gebhart. *Heat Transfer*. McGraw-Hill, New York, 1961.

[Ghosh-Mount87] Ghosh and Mount. An output sensitive algorithm for computing visibility graphs. In *FOCS Proceedings*, 1987.

[Gigus-Malik90] Ziv Gigus and Jitendra Malik. Computing the aspect graph for line drawings of polyhedral objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(2):113–122, Feb. 1990.

[Glassner89] Andrew Glassner, editor. *Introduction to Ray Tracing*. Academic Press, 1989.

[Goldstein-Nagel71] Robert A. Goldstein and Roger Nagel. 3-D visual simulation. *Simulation*, 16(1):25–31, Jan. 1971.

[Golub-Van Loan89] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, 1989.

[Goral et al. 84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):213–222, July 1984.

[Greenberg91] Donald P. Greenberg. Computers and architecture. *Scientific American*, pages 104–109, Feb. 1991.

[Greengard-Rokhlin87] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Computational Physics*, 73:325–348, 1987.

[Gwinner87] J. Gwinner. On the Galerkin approximation of nonsmooth boundary integral equations arising in radiative heat transfer. In *Boundary Elements IX*, volume 3: Fluid Flow and Potential Applications, Southampton, UK, 1987. Computational Mechanics Pub.

[Hanrahan et al. 91] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, July 1991. To appear.

[Hanrahan-Salzman90] Pat Hanrahan and David Salzman. A rapid hierarchical radiosity algorithm for unoccluded environments. In *Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, pages 151–171, Rennes, France, June 1990.

[He et al. 91] Xiao D. He, Kenneth E. Torrance, François X. Sillion, and Donald P. Greenberg. A comprehensive physical model for light reflection. *Computer Graphics (SIGGRAPH '91 Proceedings)*, July 1991. To appear.

[Heckbert86] Paul S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, Nov. 1986.

[Heckbert90] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):145–154, Aug. 1990.

[Heckbert-Hanrahan84] Paul S. Heckbert and Pat Hanrahan. Beam tracing polygonal objects. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):119–127, July 1984.

[Heckbert-Winget91] Paul S. Heckbert and James M. Winget. Finite element methods for global illumination. Technical report, CS Dept, UC Berkeley, June 1991. To appear.

[Hilbert12] David Hilbert. Begründung der elementaren strahlungstheorie. *Physikalische Zeitschrift*, 13:1056, 1912.

[Hilbert-Cohn-Vossen32] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination.* Chelsea Publishing, New York, 1932.

[Hildebrand65] Francis B. Hildebrand. *Methods of Applied Mathematics, 2nd ed.* Prentice-Hall, Englewood Cliffs, NJ, 1965.

[Horn77] Berthold K. P. Horn. Understanding image intensities. *Artificial Intelligence*, 8:201–231, 1977.

[Horn-Brooks89] Berthold K. P. Horn and Michael J. Brooks, editors. *Shape from Shading.* MIT Press, Cambridge MA, 1989.

[Hottel54] Hoyt C. Hottel. Radiant heat transmission, 3rd ed. In W. H. McAdams, editor, *Heat Transmission*, New York, 1954. McGraw-Hill.

[Hottel-Keller55] Hoyt C. Hottel and J. D. Keller. Effect of reradiation on heat transmission in furnaces and through openings. *Trans. Amer. Soc. of Mech. Eng.*, IS-55-6:39–49, 1955.

[Hottel-Sarofim67] Hoyt C. Hottel and Adel F. Sarofim. *Radiative Transfer.* McGraw-Hill, New York, 1967.

[IES87a] *IES Lighting Handbook: Reference Volume.* Illuminating Engineering Society, New York, 1987.

[IES87b] *Nomenclature and Definitions for Illuminating Engineering.* Illuminating Engineering Society, New York, 1987.

[Immel et al. 86] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):133–142, Aug. 1986.

[Jakob57] Max Jakob. *Heat Transfer*, volume 2. John Wiley & Sons, New York, 1957.

[Jensen48] H. Højgaard Jensen. Some notes on heat-transfer by radiation. *Matematisk-Fysiske Meddelelser*, 24(8):1–26, 1948.

[Jerri85] Abdul J. Jerri. *Introduction to Integral Equations with Applications.* Dekker, New York, 1985.

[Kajiya86] James T. Kajiya. The rendering equation. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):143–150, Aug. 1986.

[Kantorovich-Krylov58] L. V. Kantorovich and V. I. Krylov. *Approximate Methods of Higher Analysis.* Interscience Publishers, New York, 1958.

[Keavey88] M. A. Keavey. An isoparametric boundary solution for thermal radiation. *Communications in Applied Numerical Methods*, 4:639–646, 1988.

[Koenderink-van Doorn79] J. J. Koenderink and A. J. van Doorn. The internal representation of solid shape with respect to vision. *Biol. Cybern.*, 32:211–216, 1979.

[Koenderink-van Doorn83] J. J. Koenderink and A. J. van Doorn. Geometrical modes as a general method to treat diffuse interreflections in radiometry. *J. Opt. Soc. Am.*, 73(6):843–850, June 1983.

[Lee et al. 85] Mark E. Lee, Richard A. Redner, and Samuel P. Uselton. Statistically optimized sampling for distributed ray tracing. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):61–67, July 1985.

[Lewis-Miller84] E. E. Lewis and W. F. Miller. *Computational Methods of Neutron Transport.* John Wiley & Sons, New York, 1984.

[Love68] Tom J. Love. *Radiative Heat Transfer.* Charles E. Merrill Publishing, Columbus, OH, 1968.

[McAdams54] W. H. McAdams. *Heat Transmission, 3rd ed.* McGraw-Hill, New York, 1954.

[Meyer et al. 86] Gary W. Meyer, Holly E. Rushmeier, Michael F. Cohen, Donald P. Greenberg, and Kenneth E. Torrance. An experimental evaluation of computer graphics imagery. *ACM Trans. on Graphics*, 5(1):30–50, Jan. 1986.

[Meyer88] Gary W. Meyer. Wavelength selection for synthetic image generation. *Computer Vision, Graphics and Image Processing*, 41:57–59, 1988.

[Mitchell87] Don P. Mitchell. Generating antialiased images at low sampling densities. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):65–72, July 1987.

[Moon36] Parry Moon. *The Scientific Basis of Illuminating Engineering.* McGraw-Hill, New York, 1936.

[Nicodemus et al. 77] Fred E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. *Geometrical Considerations and Nomenclature for Reflectance.* Oct. 1977. NBS Monograph 160.

[Nicodemus76] Fred E. Nicodemus, editor. *Self-Study Manual on Optical Radiation Measurements: Part 1–Concepts, Chapters 1-3.* Mar. 1976. NBS Technical Note 910-1.

[Nicodemus78] Fred E. Nicodemus, editor. *Self-Study Manual on Optical Radiation Measurements: Part 1–Concepts, Chapters 4 and 5.* Feb. 1978. NBS Technical Note 910-2.

[Nishita-Nakamae83] Tomoyuki Nishita and Eihachiro Nakamae. Half-tone representation of 3-D objects illuminated by area sources or polyhedron sources. In *COMPSAC '83, Proc. IEEE 7th Intl. Comp. Soft. and Applications Conf.*, pages 237–242, Nov. 1983.

[Nishita-Nakamae85] Tomoyuki Nishita and Eihachiro Nakamae. Continuous tone representation of 3-D objects taking account of shadows and interreflection. *Computer Graphics (SIGGRAPH '85 Proceedings)*, 19(3):23–30, July 1985.

[Nusselt28] Wilhelm Nusselt. Graphische bestmmung des winkelverhaltnisses bei der wärmestrahlung. *Zeitschrift des Vereines Deutscher Ingenieure*, 72:673, 1928.

[O'Brien55] Philip F. O'Brien. Interreflections in rooms by a network method. *J. Opt. Soc. Amer.*, 45(6):419–424, June 1955.

[Oppenheim56] A. K. Oppenheim. *Trans. Amer. Soc. of Mech. Eng.*, 78:725, 1956.

[O'Rourke87] Joseph O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[Özisik73] Necati M. Özisik. *Radiative Transfer and Interactions with Conduction and Convection.* Wiley, New York, 1973.

[Paddon-Holstein85] D. J. Paddon and H. Holstein, editors. *Multigrid Methods for Integral and Differential Equations.* Clarendon Press, Oxford, 1985.

[Painter-Sloan89] James Painter and Kenneth Sloan. Antialiased ray tracing by adaptive progressive refinement. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):281–288, July 1989.

[Paschkis36] V. Paschkis. *Electrotech. Maschinenbau*, 54:617, 1936.

[Phong75] Bui-Tuong Phong. Illumination for computer-generated pictures. *CACM*, 18(6), June 1975.

[Planck14] Max Planck. *The Theory of Heat Radiation.* Blakiston's Son & Co. Philadelphia, 1914.

[Poljak35] G. Poljak. Analysis of heat interchange by radiation between diffuse surfaces. *Technical Physics of the USSR*, 1(5,6):555–590, 1935. (in German).

[Preparata-Shamos85] Franco Preparata and Michael I. Shamos. *Computational Geometry.* Springer Verlag, 1985.

[Ralston-Rabinowitz78] Anthony Ralston and Philip Rabinowitz. *A First Course in Numerical Analysis, 2nd ed.* McGraw-Hill, New York, 1978.

[Reeves et al. 87] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):283–291, July 1987.

[Rheinboldt-Mesztenyi80] Werner Rheinboldt and Charles K. Mesztenyi. On a data structure for adaptive finite element mesh refinements. *ACM Trans. on Math. Software*, 6(2):166–187, June 1980.

[Rushmeier87] Holly E. Rushmeier. The zonal method for calculating light intensities in the presence of a participating medium. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):293–302, July 1987.

[Rushmeier-Torrance90] Holly E. Rushmeier and Kenneth E. Torrance. Extending the radiosity method to include specularly reflecting and translucent materials. *ACM Trans. on Graphics*, 9(1):1–27, Jan. 1990.

[Samet90] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.

[Shao et al. 88] Min-Zhi Shao, Qun-Sheng Peng, and You-Dong Liang. A new radiosity approach by procedural refinements for realistic image synthesis. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):93–101, Aug. 1988.

[Shinya et al. 87] Mikio Shinya, Tokiichiro Takahashi, and Seiichiro Naito. Principles and applications of pencil tracing. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):45–54, July 1987.

[Shinya et al. 89] Mikio Shinya, Takafumi Saito, and Tokiichiro Takahashi. Rendering techniques for transparent objects. In *Proc. Graphics Interface '89.* pages 173–182, 1989.

[Shirley90] Peter Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90*, pages 205–212, May 1990.

[Shirley91] Peter Shirley. *Physically Based Lighting Calculations for Computer Graphics.* PhD thesis, Dept. of CS, U. of Illinois, Urbana-Champaign. 1991.

[Siegel-Howell81] Robert Siegel and John R. Howell. *Thermal Radiation Heat Transfer.* Hemisphere Publishing Corp., Washington, DC, 1981.

[Sillion et al. 91] François X. Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. A global illumination solution for general reflectance distributions. *Computer Graphics (SIGGRAPH '91 Proceedings)*, July 1991. To appear.

[Sillion-Puech89] François Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):335–344, July 1989.

[Silverman86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London, 1986.

[Sparrow et al. 61] Ephraim M. Sparrow, J. L. Gregg, J. V. Szel, and P. Manos. Analysis, results, and interpretation for radiation between some simply-arranged gray surfaces. *ASME J. of Heat Transfer*, pages 207–214, May 1961.

[Sparrow60] Ephraim M. Sparrow. Application of variational methods to radiation heat-transfer calculations. *ASME J. of Heat Transfer*, pages 375–380. Nov. 1960.

[Sparrow63] Ephraim M. Sparrow. On the calculation of radiant interchange between surfaces. In Warren Ibele, editor, *Modern Developments in Heat Transfer*, New York, 1963. Academic Press.

[Sparrow-Cess78] Ephraim M. Sparrow and R. D. Cess. *Radiation Heat Transfer.* Hemisphere, 1978.

[Sparrow-Haji-Sheikh65] Ephraim M. Sparrow and A. Haji-Sheikh. A generalized variational method for calculating radiant interchange between surfaces. *ASME J. of Heat Transfer*, pages 103–109, Feb. 1965.

[Strang86] Gilbert Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Box 157, Wellesley MA 02181, 1986.

[Strang88] Gilbert Strang. *Linear Algebra and Its Applications, 3rd ed.* Harcourt Brace Jovanovich, San Diego, 1988.

[Strang-Fix73] Gilbert Strang and George J. Fix. *An Analysis of the Finite Element Method.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[Strauss88] Paul S. Strauss. *BAGS: The Brown Animation Generation System.* PhD thesis, Dept. of CS, Brown U, May 1988. Tech. Report CS-88-2.

[Sutherland et al. 74] Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker. A characterization of ten hidden-surface algorithms. *Computing Surveys.* 6(1):1, Mar. 1974.

[Torrance-Sparrow67] Kenneth E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. *J. Opt. Soc. Amer.*, 57(9):1105–1114, Sept. 1967.

[Tregenza83] P. R. Tregenza. The Monte Carlo method in lighting calculations. *Lighting Research & Technology*, 15(4):163–170, 1983.

[Von Herzen-Barr87] Brian Von Herzen and Alan H. Barr. Accurate triangulations of deformed, intersecting surfaces. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):103–110, July 1987.

[Wallace et al. 87] John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):311–320, July 1987.

[Wallace et al. 89] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):315–324, July 1989.

[Ward et al. 88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):85–92, Aug. 1988.

[Ward90] Greg Ward. Private communication, 1990.

[Warnock69] John E. Warnock. A hidden surface algorithm for computer generated halftone pictures. Technical report, CS Dept, U. of Utah, June 1969. TR 4-15.

[Watt90] Mark Watt. Light-water interaction using backward beam tracing. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):377–385, Aug. 1990.

[Weiler-Atherton77] Kevin Weiler and Peter R. Atherton. Hidden surface removal using polygon area sorting. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2):214–222, Summer 1977.

[Whitted80] Turner Whitted. An improved illumination model for shaded display. *CACM*, 23(6):343–349, June 1980.

[Williams78] Lance Williams. Casting curved shadows on curved surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):270–274, Aug. 1978.

[Woo et al. 90] Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, Nov. 1990.

Appendix A

Analytic Form Factors for Flatland Radiosity

For polynomial basis functions and flatland scenes discretized using collocation, the form factors can be computed analytically. We derive these formulas for the case of linear basis functions (hats) below. This appendix employs a slight change of notation over previous sections (see figure A.1).

A.1 Kernel

Heretofore we have expressed the kernel $\kappa(s,t)$ as a function of the variables θ, ϕ, and r, which are themselves functions of arc length s and t:

$$\kappa(s,t) = \rho_h(s)\frac{\cos\phi(s,t)\,\cos\theta(s,t)}{2r(s,t)}\,v(s,t)$$

For linear elements it is straightforward to substitute for these intermediate variables and express the kernel directly as a function of s and t. This must be done to discretize the kernel into a matrix.

When computing the form factor between two edges, we use local parameter values σ and τ with range $[0,1]$ on edges e and f, respectively (figure A.2). The variable σ is linearly related to s, and τ is linear in t. Let point $P(\sigma)$ lie on edge e and point $Q(\tau)$ lie on edge f, where $P(\sigma) = P_0 + \sigma\Delta P$, $\Delta P = P_1 - P_0$, and $Q(\tau) = Q_0 + \tau\Delta Q$, $\Delta Q = Q_1 - Q_0$, and let R be the vector between them: $R(\sigma,\tau) = Q(\tau) - P(\sigma)$. Also let the lengths of the edges be

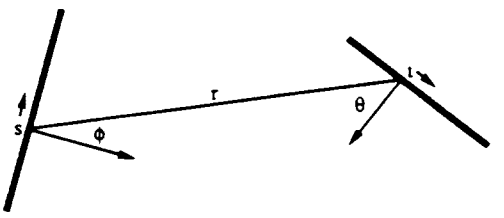

Figure A.1: *Visibility geometry for edge points with parameter values s and t.*

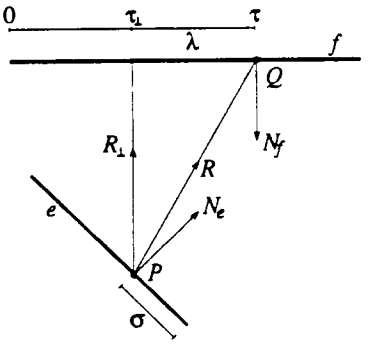

Figure A.2: *Geometry for form factor calculation between edges e and f. Vector R is resolved into parallel and perpendicular components.*

$l_e = |\Delta P|$ and $l_f = |\Delta Q|$, and their unit normals be N_e and N_f. We index edges by e and f and element nodes by i and j. There are typically many elements per edge.

In the collocation method, we fix σ at each collocation point $P = P(\sigma)$ and integrate over τ. For fixed σ, if R is decomposed into components parallel and perpendicular to ΔQ. then:

$$R = R_\perp - \lambda \Delta Q \qquad \text{where} \qquad \begin{aligned} \lambda &= \tau + \tau_\perp \\ R_\perp &= Q_0 - P + \tau_\perp \Delta Q \\ \tau_\perp &= \frac{(P - Q_0)\cdot \Delta Q}{l_f^2} \end{aligned}$$

The change of variable from τ to λ is made in order to complete the square in the formula for the distance r; it is easy to verify that $R_\perp \perp \Delta Q$, and therefore, the distance between points P and Q is

$$r = |R| = \sqrt{|R_\perp|^2 + \lambda^2 l_f^2}$$

The vector $R_\perp$ is the vector from the collocation point P to the nearest point on the line containing edge f, and the parameter λ is proportional to the distance of a point $Q(\tau)$ on edge f from this nearest point.

Because we are fixing σ, $\cos\theta$ is constant, and $R \cdot N_f = -|R_\perp|$. Recalling equation (3.2), the kernel is thus

$$\begin{aligned} \kappa(s,t) &= -\rho_h(s) \frac{(R\cdot N_e)(R\cdot N_f)}{2|R|^3} V(s,t) \\ &= \rho_h(s) \frac{(R\cdot N_e)|R_\perp|}{2|R|^3} V(s,t) \\ &= \frac{1}{2}\rho_h(s)|R_\perp| \Big((R_\perp + \lambda \Delta Q)\cdot N_e \Big) \Big(|R_\perp|^2 + \lambda^2 l_f^2 \Big)^{-3/2} V(s,t) \end{aligned}$$

and, as mentioned earlier, λ is a linear function of t. With the above substitutions, the kernel is now in a form that can be multiplied by basis functions and integrated in order to compute form factors. But first, basis functions must be chosen.

Figure A.3: *Left and right half-hat basis functions.*

A.2 Form Factors

When linear elements are used, continuity (of value, but not of slope) is assumed at most element nodes, so most basis functions will be two-sided hat functions (equation (2.5)). At edge endpoints or other locations where a step discontinuity is expected, however, half-hat basis functions will be used to decouple the value of the function on either side of a node. in case $u(s_-) \neq u(s_+)$ (figure A.3). In either case, the basis function has the piecewise-linear form

$$B_i(s) = \begin{cases} \gamma_{10}^{(i)} + \gamma_{11}^{(i)} s & \text{if } s_{i-1} \le s \le s_i \\ \gamma_{20}^{(i)} + \gamma_{21}^{(i)} s & \text{if } s_i \le s \le s_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

For a standard two-sided hat,

$$\begin{pmatrix} \gamma_{10}^{(i)} & \gamma_{11}^{(i)} \\ \gamma_{20}^{(i)} & \gamma_{21}^{(i)} \end{pmatrix} = \begin{pmatrix} -s_{i-1}/(s_i - s_{i-1}) & 1/(s_i - s_{i-1}) \\ s_{i+1}/(s_{i+1} - s_i) & -1/(s_{i+1} - s_i) \end{pmatrix}$$

For a left half-hat, the left side of the basis function is as defined above, but the right side is zeroed with $\gamma_{20} = \gamma_{21} = 0$, and for a right half-hat, the right side of the basis function is as above, but the left side is zeroed with $\gamma_{10} = \gamma_{11} = 0$. Note that these basis functions are what would result in the limit as $s_{i+1} \to s_i$ or $s_{i-1} \to s_i$, respectively. In finite element lingo, this is known as a double node, while in spline lingo, this is called a knot of multiplicity two.

For linear elements, the collocation method requires that the residual be zero at each element node s_i. A system of linear equations $(\mathbf{M} - \mathbf{K})\mathbf{u} = \mathbf{e}$ results, where $\mathbf{M} = \mathbf{I}$. $e_i = e(s_i')$, and

$$\begin{aligned} K_{ij} &= \int_0^L dt\, B_j(t) \kappa(s_i', t) \\ &= \int_{s_{i-1}}^{s_i} dt\, (\gamma_{10}^{(i)} + \gamma_{11}^{(i)} t)\, \kappa(s_i', t) + \int_{s_i}^{s_{i+1}} dt\, (\gamma_{20}^{(i)} + \gamma_{21}^{(i)} t)\, \kappa(s_i', t) \\ &= I(s_i', \gamma_{10}^{(i)}, \gamma_{11}^{(i)}, s_{i-1}, s_i) + I(s_i', \gamma_{20}^{(i)}, \gamma_{21}^{(i)}, s_i, s_{i+1}) \end{aligned}$$

where

$$I(s', \gamma_0, \gamma_1, s_0, s_1) = \int_{s_0}^{s_1} dt\, (\gamma_0 + \gamma_1 t)\, \kappa(s', t)$$

One such integral is made for each side of the hat basis function. We abbreviate this function I for short. If edge f extends from $t = s_a$ to $t = s_a + l_f$ then $t = s_a + \tau l_f = s_a + (\tau_\perp + \lambda) l_f$,

so we can change the variable of integration from t to λ:

$$I = \int_{\lambda_0}^{\lambda_1} d\lambda\, l_f\Big(\gamma_0 + \gamma_1[s_a + (\tau_\perp + \lambda)l_f]\Big)\kappa(s', t(\lambda))$$
$$= \frac{1}{2}\rho_h(s')l_f|R_\perp| \int_{\lambda_0}^{\lambda_1} d\lambda\,(\beta_0 + \beta_1\lambda)\Big((R_\perp + \lambda\Delta Q)\cdot N_e\Big)\Big(|R_\perp|^2 + \lambda^2 l_f^2\Big)^{-3/2} V(s', t(\lambda))$$

where

$$\lambda_0 = (s_0 - s_a)/l_f - \tau_\perp$$
$$\lambda_1 = (s_1 - s_a)/l_f - \tau_\perp$$
$$\beta_0 = \gamma_0 + \gamma_1(s_a + \tau_\perp l_f)$$
$$\beta_1 = \gamma_1 l_f$$

If the entire interval $tin[s_0, s_1]$ is visible to point s', then $V(s', t) = 1$, and

$$I = \frac{1}{2}\rho_h(s')l_f|R_\perp| \int_{\lambda_0}^{\lambda_1} d\lambda\,(\beta_0 + \beta_1\lambda)(R_\perp\cdot N_e + \lambda\Delta Q\cdot N_e)l_f^{-3}\left(\frac{|R_\perp|^2}{l_f^2} + \lambda^2\right)^{-3/2}$$
$$= C\int_{\lambda_0}^{\lambda_1} d\lambda\,(\alpha_0 + \alpha_1\lambda + \alpha_2\lambda^2)(a + \lambda^2)^{-3/2}$$

where

$$C = \frac{1}{2}\rho_h(s')|R_\perp|/l_f^2$$
$$\alpha_0 = \beta_0 R_\perp\cdot N_e$$
$$\alpha_1 = \beta_0\Delta Q\cdot N_e + \beta_1 R_\perp\cdot N_e$$
$$\alpha_2 = \beta_1\Delta Q\cdot N_e$$
$$a = |R_\perp|^2/l_f^2$$

With a little calculus, we can evaluate the integrals above in closed form:

$$\int dx\,(x^2 + a)^{-3/2} = \frac{1}{a}x(x^2 + a)^{-1/2}$$
$$\int dx\,x(x^2 + a)^{-3/2} = -(x^2 + a)^{-1/2}$$
$$\int dx\,x^2(x^2 + a)^{-3/2} = \sinh^{-1}\left(\frac{x}{\sqrt{a}}\right) - x(x^2 + a)^{-1/2}$$
$$= \log(x + \sqrt{x^2 + a}) - x(x^2 + a)^{-1/2} - \frac{1}{2}\log a$$

for any $a \neq 0$. To compute the form factors for quadratic or higher degree basis functions, we also need the recurrence relation:

$$(m-1)\int dx\,x^{m+1}(x^2 + a)^{-3/2} = x^m(x^2 + a)^{-1/2} - am\int dx\,x^{m-1}(x^2 + a)^{-3/2}$$

If the collocation point $P(\sigma)$, the "viewpoint" for form factor calculation, is not coincident with edge f, then its distance to edge f is positive ($|R_\perp| > 0$) and $a > 0$, so the above formulas apply. We treat this case first, and then the coincident case ($a = 0$).

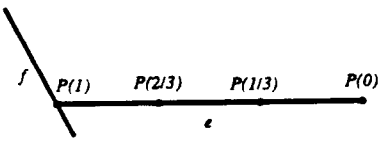

Figure A.4: *Reflex corner between edges e and f causes one of the collocation points at an edge endpoint (in this case, $P(1)$) to fall on edge f.*

A.2.1 Case 1: Viewpoint Not on Edge f

When $a > 0$, using the above indefinite integrals, we find

$$\begin{aligned} I(s', \gamma_0, \gamma_1, s_0, s_1) &= C \left[\left\{ \left(\frac{\alpha_0}{a} - \alpha_2 \right) \lambda - \alpha_1 \right\} (\lambda^2 + a)^{-1/2} + \alpha_2 \sinh^{-1} \left(\frac{\lambda}{\sqrt{a}} \right) \right]_{\lambda_0}^{\lambda_1} \\ &= C \left\{ \left(\frac{\alpha_0}{a} - \alpha_2 \right) \left(\frac{\lambda_1}{\mu_1} - \frac{\lambda_0}{\mu_0} \right) - \alpha_1 \left(\frac{1}{\mu_1} - \frac{1}{\mu_0} \right) + \alpha_2 \log \left(\frac{\lambda_1 + \mu_1}{\lambda_0 + \mu_0} \right) \right\} \end{aligned} \tag{A.1}$$

where $\mu_0 = \sqrt{\lambda_0^2 + a}$ and $\mu_1 = \sqrt{\lambda_1^2 + a}$.

The above formula for the function I is evaluated for each side of the hat basis function to compute a single form factor K_{ij}. Where half-hat basis functions are used, one of the two integrals I will have basis coefficients $\gamma_0 = \gamma_1 = 0$ and will therefore be zero. When box basis functions are used, $\gamma_1 = 0$ and therefore $\alpha_2 = 0$, so no logarithm is required.

A.2.2 Case 2: Viewpoint on edge f

At a reflex corner with coincident edge endpoints, element nodes from two edges will coincide. If linear elements are used and collocation points are chosen to be the element nodes, then at a reflex corner between edges e and f, one of the element nodes of edge e will lie on edge f, and vice-versa (figure A.4). The form factors for these collocation points will have $|R_\perp| = 0$ and hence $a = C = 0$ in the above formulas.

The above formulas are indeterminate in this case, so the true value must be found in other ways. The physically motivated solution to this problem is to perturb the collocation point somewhat, to move it off edge f by a tiny amount so that $a \neq 0$. This can be done numerically or symbolically.

The symbolic solution is made by taking the limit of the above formula (A.1) as $a \to 0^+$. There are four sub-cases:

$$I(s', \gamma_0, \gamma_1, s_0, s_1) = \begin{cases} 0 & \text{if } \lambda_0 < 0 \text{ and } \lambda_1 < 0, \text{ or } \lambda_0 > 0 \text{ and } \lambda_1 > 0 \\ -\frac{1}{2}\beta_0 N_e \cdot N_f - \frac{\alpha_1}{2l_f} & \text{if } \lambda_0 > 0 \text{ and } \lambda_1 = 0 \\ -\frac{1}{2}\beta_0 N_e \cdot N_f + \frac{\alpha_1}{2l_f} & \text{if } \lambda_0 = 0 \text{ and } \lambda_1 > 0 \\ -\frac{1}{2}\beta_0 N_e \cdot N_f & \text{if } \lambda_0 < 0 \text{ and } \lambda_1 > 0 \end{cases}$$