Discontinuity Meshing for Radiosity Image Synthesis

Filippo Tampieri
Ph.D Thesis

93-1346
May 1993

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

DISCONTINUITY MESHING

FOR

RADIOSITY IMAGE SYNTHESIS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Filippo Tampieri

May 1993

DISCONTINUITY MESHING

FOR

RADIOSITY IMAGE SYNTHESIS

Filippo Tampieri, Ph.D.

Cornell University 1993

The simulation of global illumination is one of the most fundamental problems in computer graphics, with applications in a wide variety of areas. This problem studies the light energy transfer between reflective surfaces in an environment. Initially derived from the field of thermal engineering, radiosity has emerged over the past several years as one of the most promising solution methods.

Despite having produced some of the most realistic-looking computer generated images to date, radiosity methods have not yet met with widespread acceptance. The main obstacle has been their need for very careful and time consuming user intervention, without which, current techniques are prone to generating a wide range of annoying visual artifacts. These artifacts are generally due to poor surface meshing, resulting in insufficient sampling density and ineffective sample placement.

This thesis investigates the roots of this problem by taking a step back from the traditional finite element formulation of radiosity and examining the more general integral equation formulation. An analysis of the radiance functions described by this equation shows how umbra and penumbra boundaries as well as other sharp changes in illumination actually correspond to discontinuities in the radiance function and its derivatives. The results of this analysis have led to the concept of *discontinuity meshing*, whereby accurate approximations to the radiance functions are computed by explicitly representing their discontinuities as boundaries in the mesh.

This concept has been applied to the design of a discontinuity meshing algorithm for polyhedral environments. The algorithm is embedded in a progressive refinement radiosity system and uses piecewise quadratic interpolation to reconstruct a smooth radiance function while preserving discontinuities where appropriate.

The radiosity solutions produced by the new algorithm are compared against a photograph of a physical environment, an analytical solution, and a conventional, yet state-of-the-art, radiosity system, and its performance on architectural models of medium complexity is measured. The results are remarkably accurate both numerically and visually. The new discontinuity meshing algorithm drastically reduces, and in many cases eliminates, many of the annoying artifacts typical of conventional radiosity meshes, producing images of previously unattained quality. Moreover, the meshing is completely automatic and produces solutions that are highly view-independent.

Biographical Sketch

The author was born in Novara, Italy, on December 1st, 1962. He was first exposed to Computer Graphics while studying Computer Science at the Universitá degli Studi di Milano, where he spent his senior year implementing the obligatory ray tracer, the distinctive mark of any respectable computer graphnic.

During that time, he grew very fond of his creature and, worried that the world of industry might require him to abandon it, he decided to embrace the profession of eternal student. In 1985, he entered a competition for a Fulbright scholarship which eventually lead him to enter the Program of Computer Graphics at Cornell University in August 1986.

While at Cornell, the author did research on global illumination algorithms for parallel computer architectures. During this investigation, he was exposed to many new and wild ideas and, despite his resistance, by the time he received his Master's, he was showing the first symptoms of the radiosity virus.

In an attempt to fight his illness, in September 1988, the author returned to Italy, where he worked as a computer graphics consultant on geometric modeling and more ray tracing.

The radiosity virus, though, proved too resilient to overcome and, by Summer 1989, the author was back at Cornell as a PhD candidate in search for a cure.

There, he spent his last years researching an antidote that could save future fellow graphnics from spending their life making pictures of the dreadful "Cornell box."

Dedicato a Molly,

ai miei genitori

e a mio fratello Andrea

Acknowledgements

First of all, I would like to thank my parents for all their love, encouragement, and support; they never faltered despite the many years, the long distance, and my many "surprise announcements."

I am deeply grateful to Professor Don Greenberg. His enthusiasm and energy have always been contagious; he has been a caring friend and his continuous trust and support have helped through many hard times. As an advisor, he has provided many suggestions that have greatly improved the quality of this thesis. As the director of the Program of Computer Graphics, he has provided the terrific environment in which my work, and that of many other fellow students, was carried out.

My appreciation goes to Professor Roy Hall for his help reviewing this thesis and for the many discussions, personal and professional, that we have had over the past few years.

Thanks also go to Professor Charles Van Loan for serving on my committee.

I would also like to acknowledge and thank Professor Daniele Marini for introducing me to computer graphics, many years ago. Had he not gone through the trouble of getting me onto a graphics workstation, I think my past few years would have been very different.

Next, I would like to thank my research partner, officemate, and friend, Dani Lischinski, without whom this work would not have been possible. We thought we were going to take a month and extend radiosity to curved surfaces, and ended up two years later with beautiful images of the flattest models in the lab. Working together has been instructive, productive, and especially fun. Most of the ideas I present in this thesis are actually our ideas, born and matured out of our endless discussions.

My gratitude also goes to all the past and present students and staff at the lab who contributed in many ways to make my time here a truly enriching experience.

Many thanks to Ellen French and Fran Brown for protecting me from bureaucracy and feeding me on many occasions. To Jonathan Corson-Rikert for keeping the lab running so well in Don's absence. To Hurf Sheldon for keeping the system running so smoothly and for his patience and prompt help the many times I needed assistance. To Xiao Dong He, sailor Sheldon, my back-to-back officemate, for his easy-going attitude. To Ben Trumbore, DeathBed expert, for being always ready to help. To Brian Smits for shedding light on the mysteries of color and spectral curves. To Chris Schoeneman and Rick Pasetto for writing graphics libraries that people want to use; to Rick for the colored padding noise and to Chris for Widget Lisp. To Dan Kartch for the many wonderful cakes that brought me back to the lab so many Mondays. To Harold Zatz for volunteering to be our TeX guru. To Patrick Heynen for helping me find my way on the NeXT machines.

I would also like to thank Greg Spencer for modeling the simple scene used to demonstrate the visual artifacts due to uniform meshing shown in Figures 3.1 and 3.2; Ted Himlan for helping Dani and I set up the simple physical scene shown in Figure 6.1; Matt Hyatt for modeling the Victorian Room, the White Room, the Green Room, and the Pink Room used in Chapter 6; and Charles D'Autremont for modeling the MackIntosh Room used in the same chapter.

Cornell has been a wonderful place to learn. It has been great for my academic education, but has also been the exploratory playground for many other interests. I would like to thank sifu Maurice Haltom and all the instructors and students at the Agape School of Movement Studies for teaching me Chinese martial arts and trying to teach me African Dance. Thanks to David Pearson for introducing me to the rock climbing experience; to Karl Bohringer for the many hours spent clinging to "the wall," and to Dan Tillemans and the other instructors for taking me to climb real rock. Thanks to Jen Whiting, Iris, and Ken for pushing me through the "birth canal" in J4. Thanks to Sara for teaching the "foreigners office" how to sail catamarans. And thanks to Maya for squeezing my big nose.

I would also like to take the opportunity to thank the Italian friends who kept in touch during these many years away from home. Many thanks and lots of love to Andrea, my brother, for growing closer with distance and time, for writing the funniest letters I have ever read, and for keeping at it despite my infrequent replies. Thanks to Paolo Bucci and Roberto Mameli, the other two fish out of water, for sharing our moods and experiences. Thanks to the "cucadores" for sharing their Saturday night whenever I was in town. And thanks to Stefano and Laura Piccardi, Luigi Pergolini, Manuela Bandini, Lucilla Consalez, and Enrico and Gabriella Molinari for never forgetting their friend across the Ocean.

Lastly, I would like to thank my friend, Molly Pedersen, for her love, friendship, understanding, and support. She has been a wonderful companion and an inexhaustible source of energy, helping me grow, taking me away from the office on the rare sunny days, and showing me how to tell a dog's smile.

This work was supported by the National Science Foundation grant "Interactive Input and Display Techniques" (CCR-8617880) and by the NSF/DARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219). I would also like to acknowledge the generous equipment grant from Hewlett Packard Corporation on whose workstations this research was conducted.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Realistic image synthesis is one of the oldest and most fundamental problems of computer graphics. The initial attempts at creating convincing renderings of reality were based on somewhat empirical techniques [GOUR71,PHON75], but their impact was so profound that those algorithms are now an essential component of any modern graphics workstation. As the field matured, however, many researchers have recognized the need for a more rigorous approach, whereby image synthesis is not just the art of making "pretty pictures," but is instead a faithful simulation of physical laws.

Recently, much research in computer graphics has focused on global illumination, the complex phenomena describing the interaction of light energy between reflective surfaces in an environment. Over the past decade, two different methods have emerged: Radiosity and Monte Carlo methods.

Initially derived from the field of thermal engineering [SPAR63], radiosity reduces the global illumination problem to a system of linear equations employing techniques similar to those used in finite element analysis [GORA84,COHE85, NISH85a].

Monte Carlo methods, instead, use the integral equation formulation of the global illumination problem, and try to solve it by point sampling the integrand using ray tracing techniques [KAJI86,WARD88].

Both methods have generated some of the most realistic images to date and both are very computationally intensive. However, there is a fundamental difference between the two: radiosity approaches compute solutions in object space, while Monte Carlo methods yield solutions in image space. As a consequence, the same radiosity solution can be used to render images from many different viewer's positions, while Monte Carlo methods must recompute the entire simulation every time the viewer's position is changed. We classify algorithms according to these properties into *view-independent* and *view-dependent* methods respectively.

The ability to generate interactive "walk-throughs" from view-independent, physically accurate, global illumination simulations, makes radiosity methods attractive for a wide variety of applications, particularly enclosed spaces such as in architectural design and lighting analyses.

Despite all the areas that could benefit from radiosity, this method has not yet met with widespread acceptance. This is due for the most part to the lack of satisfactory meshing algorithms. The meshing techniques used in current radiosity systems are prone to generating a wide range of annoying visual artifacts that can only be eliminated at the cost of tedious and time consuming manual adjustments to the mesh, often made surface by surface and repeated on a trial-and-error basis until a proper mesh density is found.

Clearly, new meshing techniques are needed to generate high quality solutions with the degree of reliability and automation necessary for practical applications. This thesis investigates a new approach to meshing, *discontinuity meshing*, that is completely automatic and drastically reduces the visual artifacts that afflict conventional radiosity algorithms. This approach promises to play a significant role in bringing radiosity methods closer to the demands of useful applications.

An analysis of the characteristics of radiance functions provides the insight and rationale behind the concept of discontinuity meshing. The analysis shows how umbra and penumbra boundaries, as well as other sharp changes in illumination actually correspond to discontinuities in the radiance function and its derivatives. The shape, location, and order of these discontinuities is determined by the interaction of the light sources and obstacles in the environment. Discontinuity meshing explicitly represents radiance discontinuities as boundaries in the mesh. This is in contrast to standard techniques which generate mesh elements by geometric subdivision alone. When one of these discontinuities crosses a mesh element, a large error may occur because the simple interpolation schemes used to reconstruct the radiance function cannot reproduce sharp illumination changes within an element.

This idea has been applied to the design of a discontinuity meshing algorithm for polyhedral environments. The algorithm is embedded in a progressive refinement radiosity system for ideal diffuse reflectance distributions and uses piecewise quadratic interpolation to reconstruct a smooth radiance function while preserving discontinuities where appropriate.

The actual implementation has generated some remarkably accurate solutions for environments of moderate complexity, showing significant improvements in both the numerical and the visual quality of the simulations. Although the system is not yet suitable for very complex models, it does prove the validity of the concepts and ideas, and our results suggest that discontinuity meshing may become an important component of future radiosity systems.

A complete description of the algorithm and experimental results and comparisons is presented in the following sections.

Chapter 2 introduces the necessary terminology, discusses the global illumination problem with a formulation that is cast in terms of radiance functions, and investigates the characteristics of the latter ones.

Chapter 3 reviews previous work, discussing favorable properties and limitations of the meshing, direct energy transfer, and radiance reconstruction algorithms used in current radiosity systems.

Chapter 4 discusses the design and implementation of a discontinuity meshing algorithm capable of capturing the illumination cast by a single area light source of constant emission onto the surfaces of a polyhedral environment.

Chapter 5 presents a modified progressive refinement radiosity system that uses the new discontinuity meshing algorithm, extending it to correctly compute light transfers from secondary light sources.

Chapter 6 reports on a series of experiments designed to test the performance of the new system and assess the validity of the ideas presented in the thesis.

Finally, Chapter 7 discusses advantages and limitations of the proposed methods and provides guidelines for future research.

Chapter 2

Radiance Functions

This chapter discusses the global illumination problem with a formulation that is cast in terms of radiance functions. According to this formulation, each surface in a scene has an associated radiance function that describes the illumination at any point on the surface.

The characteristics of radiance functions are then investigated. This study provides the tools for the analysis of past and current radiosity algorithms presented in the next chapter as well as the insight into the problem that led to the new solution algorithms presented later in this thesis.

2.1 Definition of Terms and Symbols

This section introduces the symbols and definitions that will be used throughout the rest of this thesis. Most of the units, symbols, and defining equation of the fundamental radiometric quantities are taken from the *IES Lighting Handbook* [IES91] as given by the Illuminating Engineering Society.

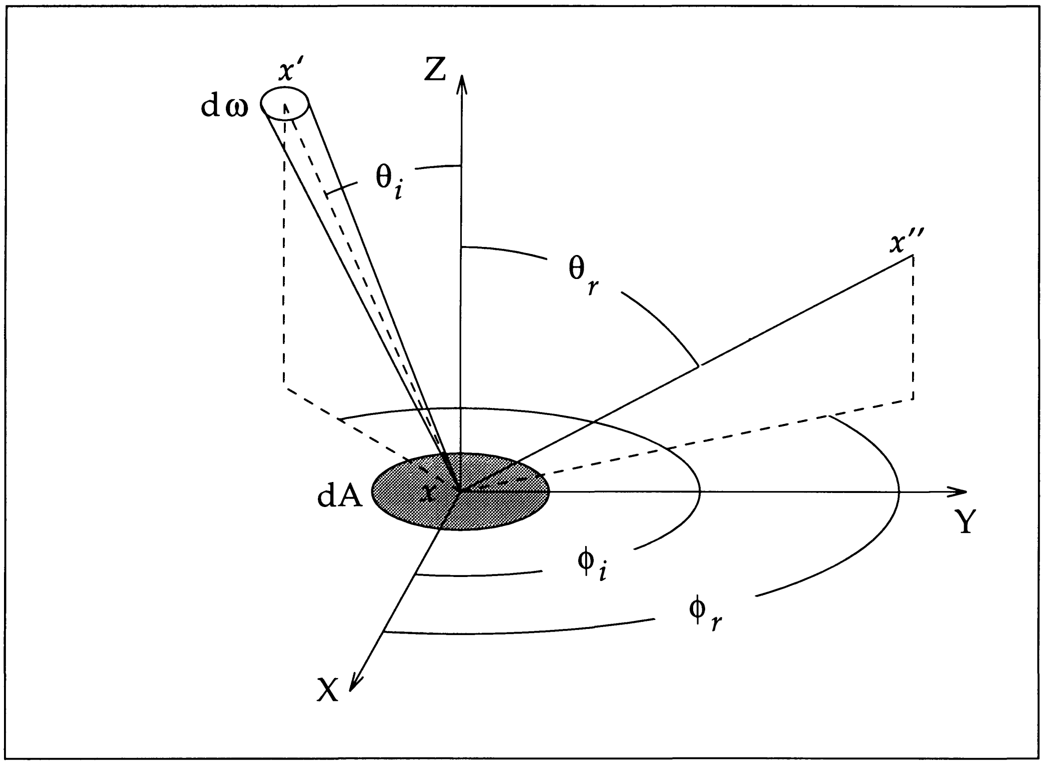


Figure 2.1: Geometry for BRDF's

Bidirectional reflectance-distribution function (BRDF): the ratio of the radiance reflected in direction (θ_r, ϕ_r) to the radiant flux density incident from direction (θ_i, ϕ_i) through a unit solid angle (see Figure 2.1). The BRDF describes the reflecting properties of a surface element placed at x and will be denoted by:

$$f_r(x; \theta_i, \phi_i; \theta_r, \phi_r)$$

or by

$$f_r(x', x, x'').$$

The latter notation will be used to indicate radiant energy arriving at point x from the direction of x' and leaving x in the direction of x''.

Irradiance (E): the density of radiant flux incident on a surface ($d\Phi/dA$). It is denoted by E and measured in watts per square meter (Wm^{-2}).

Radiance (L): the radiant flux per unit solid angle per unit projected area. More precisely, the radiance leaving a differential surface area in direction (θ, ϕ) (relative to the surface normal) is equal to $d^2\Phi/(d\omega dA\cos\theta)$. It is denoted by L and is measured in watts per steradian per square meter ($Wsr^{-1}m^{-2}$).

Radiant energy (Q): energy traveling in the form of electromagnetic waves. It is denoted by Q and is measured in joules (J).

Radiant exitance (M): the density of radiant flux leaving a surface ($d\Phi/dA$). It is denoted by M and measured in watts per square meter (Wm^{-2}).

Radiant flux (Φ): the radiant energy flowing through an area per unit time (dQ/dt). It is denoted by Φ and is measured in watt (W).

Radiant flux density ($d\Phi/dA$): the radiant energy flowing through a unit area per unit time ($d\Phi/dA$). It is referred to as *radiant exitance* when the flow is leaving the surface and as *irradiance* when the flow is incident on the surface.

Radiant intensity (I): the radiant flux emanating from a point light source through a unit solid angle around a given direction ($d\Phi/d\omega$). It is denoted by I and measured in watts per steradian (Wsr^{-1}).

Radiosity (B): the density of radiant flux leaving a Lambertian reflector. This is the name commonly used for radiant flux density in the computer graphics literature. It is denoted by B and measured in watts per square meter (Wm^{-2}).

Reflectivity (ρ): the fraction of incident radiant energy that is reflected (as opposed to absorbed or transmitted.) It is denoted by $\rho(x)$. This quantity is commonly used to specify the reflecting properties of a Lambertian surface and is related to a constant BRDF by:

$$\begin{aligned}\rho(x) &= \int_{\Omega} f_r(x)\cos\theta_r d\omega \\ &= (\int_{\Omega}\cos\theta_r d\omega) f_r(x) = \pi f_r(x)\end{aligned}$$

All of the above quantities are actually wavelength dependent. Wherever needed, this dependency will be made explicit by specifying the wavelength λ.

2.2 The Rendering Equation

The transfer of radiant energy is governed by the laws of physics. The complexities of the interaction of light and surfaces in an environment were abstracted by Kajiya [KAJI86] in 1986 in a compact formulation of the global illumination problem, generally known as the *rendering equation*. This is a Fredholm integral equation of the second kind [HECK91a] and is presented here in a slightly modified form [SHIR90b]:

$$L(x,x'',\lambda) = L^e(x,x'',\lambda) + \int_S f_r(x',x,x'',\lambda) L(x',x,\lambda)\frac{\cos\theta_i\cos\theta_j}{r^2}v(x,x')dA(x') \quad (2.1)$$

where

$L(x,x'',\lambda)$ is the *radiance* at point x in the direction of point x'' at wavelength λ.

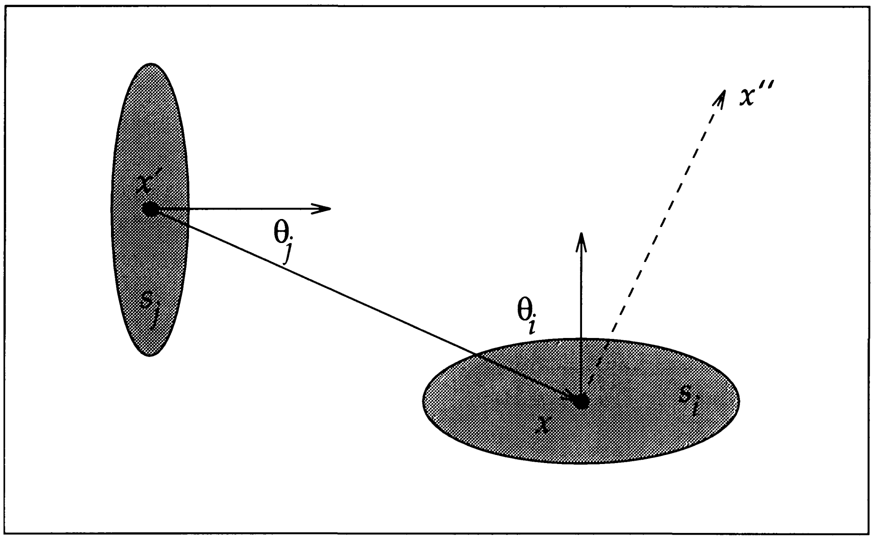


Figure 2.2: Geometry for Equations (2.1) and (2.2).

$L^e(x, x'', \lambda)$ is the radiant exitance per unit solid angle emitted from x towards x'' at wavelength λ.

S is the union of all the surfaces in the environment.

$v(x, x')$ is a visibility term; it is 1 if x and x' are visible to each other, and 0 otherwise;

$dA(x')$ is a differential surface area centered at x'.

r is the distance between x and x';

θ_i and θ_j are the angles between the surface normals at x and x' and the line connecting the two points (see Figure 2.2).

The rendering equation as given above is well suited for view-dependent global illumination algorithms like ray tracing and Monte Carlo techniques. In fact, these algorithms typically discretize the integration by tracing rays from x in a number of random directions, chosen perhaps according to the distribution of f_r. Instead, view-independent algorithms, such as radiosity, typically work by computing the energy transfer between pairs of surfaces (or surface elements) in the environment. For this latter class of algorithms, it is therefore more convenient to break the domain of integration S into the surfaces $s_1, s_2, \ldots, s_n$ of the scene, to get:

$$L_i(x, x'', \lambda) = \qquad (2.2)$$
$$L_i^e(x, x'', \lambda) + \sum_{s_j \in S} \int_{x' \in s_j} f_{ri}(x', x, x'', \lambda) L_j(x', x, \lambda) \frac{\cos\theta_i \cos\theta_j}{r^2} v(x, x') dA(x')$$

According to this formulation, each surface s_i in the environment has an associated *radiance function* L_i. Solving for the set of L_i's, $i \in [1..n]$, yields a view-independent solution to the global illumination problem.

If all the surfaces are ideal diffuse (Lambertian) reflectors, the general rendering equation can be simplified to:

$$L_i(x, \lambda) = L_i^e(x, \lambda) + f_{ri}(x, \lambda) \sum_{s_j \in S} \int_{x' \in s_j} L_j(x', \lambda) \frac{\cos\theta_i \cos\theta_j}{r^2} v(x, x') dA(x') \qquad (2.3)$$

This equation will be referred to as the *diffuse rendering equation* and most of the remainder of this thesis will be dedicated to its solution.

2.3 The Radiosity Equation

The computer graphics literature has traditionally formulated the diffuse global illumination problem in terms of radiosity [GORA84]. In this section we will show how the traditional radiosity equation can be derived from the diffuse rendering equation given above. In order to simplify the equations, the dependency on the wavelength λ will be omitted.

By definition, the radiosity, $B_i(x)$, is the radiant flux density leaving surface s_i at point x. This quantity can be computed from the radiance, $L_i(x)$ by integrating the latter over a hemisphere centered above x. For an ideal diffuse (Lambertian) reflector, the radiance is equal in all direction, thus:

$$\begin{aligned} B_i(x) &= \int_\Omega L_i(x) \cos\theta d\omega \\ &= (\int_\Omega \cos\theta d\omega) L_i(x) = \pi L_i(x). \end{aligned}$$

The traditional radiosity formulation requires that the environment be discretized into patches of constant radiosity and constant reflectivity. For simplicity, we will assume that these patches correspond to our surfaces s_i's. Then, the area averaged radiosity B_i and reflectivity ρ_i for a finite area surface s_i of area A_i are given by

$$B_i = \frac{1}{A_i} \int_{x \in s_i} \pi L_i(x) dA(x)$$

and

$$\rho_i = \frac{1}{A_i} \int_{x \in s_i} \pi f_{ri}(x) dA(x).$$

We can now derive the traditional radiosity equation by integrating each side of Equation (2.3) over the area of surface s_i:

$$\begin{aligned} B_i A_i &= \int_{x \in s_i} \pi L_i(x) dA(x) \\ &= \int_{x \in s_i} \pi L_i^e(x) dA(x) + \\ &\quad \int_{x \in s_i} \pi f_{ri}(x) \sum_{s_j \in S} \int_{x' \in s_j} L_j(x') \frac{\cos\theta_i \cos\theta_j}{r^2} v(x, x') dA(x') dA(x) \\ &= \int_{x \in s_i} \pi L_i^e(x) dA(x) + \\ &\quad \rho_i \sum_{s_j \in S} \int_{x \in s_i} \int_{x' \in s_j} L_j(x') \frac{\cos\theta_i \cos\theta_j}{r^2} v(x, x') dA(x') dA(x) \end{aligned}$$

$$
\begin{aligned}
&= B_i^e A_i + \rho_i \sum_{s_j \in S} B_j \int_{x \in s_i} \int_{x' \in s_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} v(x, x') dA(x') dA(x) \\
&= B_i^e A_i + \rho_i \sum_{s_j \in S} B_j A_i F_{ij} \qquad (2.4)
\end{aligned}
$$

where B_i^e is the emitted radiosity and F_{ij} is the form factor from surface s_i to surface s_j. Dividing each side of the equation by A_i yields the traditional *radiosity equation:*

$$
B_i = B_i^e + \rho_i \sum_{s_j \in S} B_j F_{ij} \qquad (2.5)
$$

2.4 Characteristics of Radiance Functions

In trying to evaluate existing global illumination algorithms or to design effective new ones it is useful to examine actual radiance functions. Determining which characteristics of radiance functions are mostly responsible for the "realistic look" of actual objects and surfaces can help concentrate research efforts on subproblems whose solution will yield the highest payoffs. Furthermore, understanding the mechanisms responsible for the most crucial properties of radiance functions can provide important clues to the design of effective simulation techniques.

This section is concerned with the characteristics of the radiance functions related to the diffuse rendering equation (see Equation (2.3)). In the real world, these functions exhibit an abundance of subtle and complex illumination details that are hard to simulate, even for the simplest environments. Figure 2.3 shows a photograph of a cardboard triangle suspended over a matte white floor illuminated by a square diffuse light source. The most obvious characteristic of the radiance function across the floor is the contrast between the smoothness of the unoccluded areas and the sharpness of the boundaries of the triangular shadow. The shadow itself is divided into a darker region *(umbra)* that is completely occluded from the light source and a surrounding area *(penumbra)* that provides a transition to the lighter regions outside the shadow. Umbras and penumbras are marked by boundaries of different sharpness; for example, the variation in radiance at the lower and outer boundary of the penumbra is so sharp that it gives rise to a bright Mach band, while that towards the upper left of the picture is much less noticeable.

An analysis of radiance functions will show that these illumination details correspond to discontinuities in the radiance function and/or its derivatives (see Figure 2.4). An analysis and classification of these discontinuities first appeared in Heckbert's thesis [HECK91a]. Following Heckbert's definition:

A function L has a D^k discontinuity at x, if it is C^{k-1} but not C^k there.

The analysis that follows will be limited to planar polygonal non-interpenetrating surfaces. This class of surfaces was chosen for a number of reasons:

1. The analysis is simpler than in the case of more general geometries.
2. The results of the analysis are immediately applicable to the design and understanding of radiosity algorithms.
3. The vast majority of current radiosity systems are limited to this class of surfaces and use polyhedral approximations to handle curved surfaces.

The diffuse rendering equation described by Equation (2.3) is a linear system of equations. Each equation describes the radiance function over a surface s_i as the sum of simpler radiance functions, each one representing the contribution of

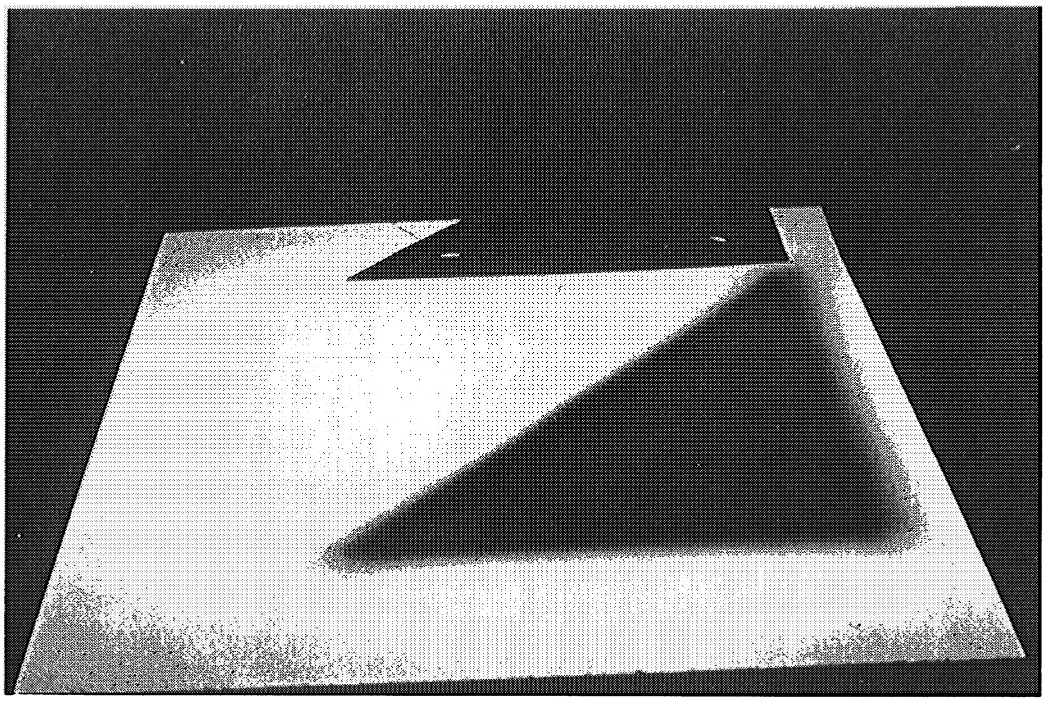


Figure 2.3: Photograph of a simple scene. The light is a small square area source.

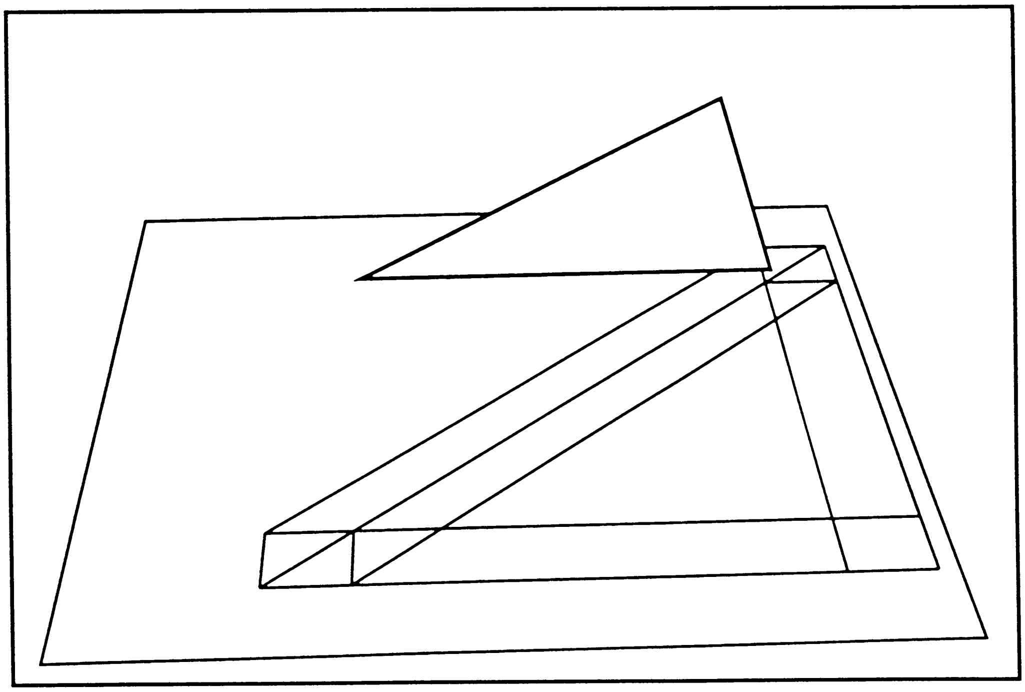


Figure 2.4: Discontinuity lines in a simple scene

a single surface in the environment. More formally:

$$L_i(x) = L_i^e(x) + \sum_{s_j \in S} L_{ij}(x) \tag{2.6}$$

where

$$L_{ij}(x) = f_{ri}(x) \int_{x' \in s_j} L_j(x') \frac{\cos\theta_i \cos\theta_j}{r^2} v(x, x') dA(x') \tag{2.7}$$

denotes the contribution of surface s_j to surface s_i. Due to the principle of superposition, it is then sufficient to study the properties of a single L_{ij} to give a characterization of the radiance function L_i.

The first part of the analysis will assume that surfaces s_i and s_j do not touch each other and that the radiance function L_i of the source s_j is smooth, i.e. C^∞. These constraints will be lifted at end of the discussion.

Under these assumptions, the integrand in Equation (2.7) is a smooth function. Therefore, if the visibility of the source s_j is constant or varies smoothly, the radiance function L_{ij} will be smooth as well. Abrupt changes in visibility, however, will introduce discontinuities of various orders into L_{ij}. The rest of this section will introduce the possible discontinuities and explain their causes.

2.4.1 D^0 Discontinuities

By definition, D^0 discontinuities are discontinuities in the function itself (value discontinuities). They are introduced by edges or vertices of occluders (or the light source itself) lying on the receiver s_i. Thus, they can occur either along line segments, or in a pointwise fashion. An example is shown in Figure 2.5.

D^0 discontinuities also occur along shadow boundaries cast by point light sources. In this case no penumbra regions are present, and the transition from the umbra into the unoccluded area is a discontinuous one. For conciseness, only finite area light sources will be discussed. The analysis of point light sources is only simpler.

In an environment with m edges, no interpenetrating polygons, and no point light sources, there can be $O(m^2)$ D^0 discontinuities, as it is possible for each of $O(m)$ edges to lie on $O(m)$ faces.

2.4.2 D^1 and D^2 Discontinuities

Discontinuities in the first and second derivatives of the function are referred to as D^1 and D^2 discontinuities, respectively. They are caused by the interaction of a light source with the occluding objects intervening between the source and the receiver, causing abrupt changes in the visibility of the former one. These changes (also known as *visual events*) have been studied in the computational geometry and computer vision literature [GIGU90,GIGU91]. There, it is shown that these visibility changes occur along surfaces defined by the edges and the vertices of the objects in the scene (including light sources.) The visual events in a polyhedral environment can be classified into two types: edge-vertex (EV) events and edge-edge-edge (EEE) events [GIGU90].

EV events occur on a subset of the plane defined by an edge and a vertex fully or partially visible to each other. The set of points on that plane from which the vertex can be seen coinciding with the edge is called the *critical surface*. Visibility changes on a receiver occur along the curves of intersection between the receiving plane and the critical surfaces. For polygonal environments, these *critical curves* are line segments in the case of an EV event. They correspond to either D^1 or D^2 discontinuities, as demonstrated in Figures 2.6 and 2.7. D^1 discontinuities are perceived as Mach bands on the receiver. Strong D^2 discontinuities are

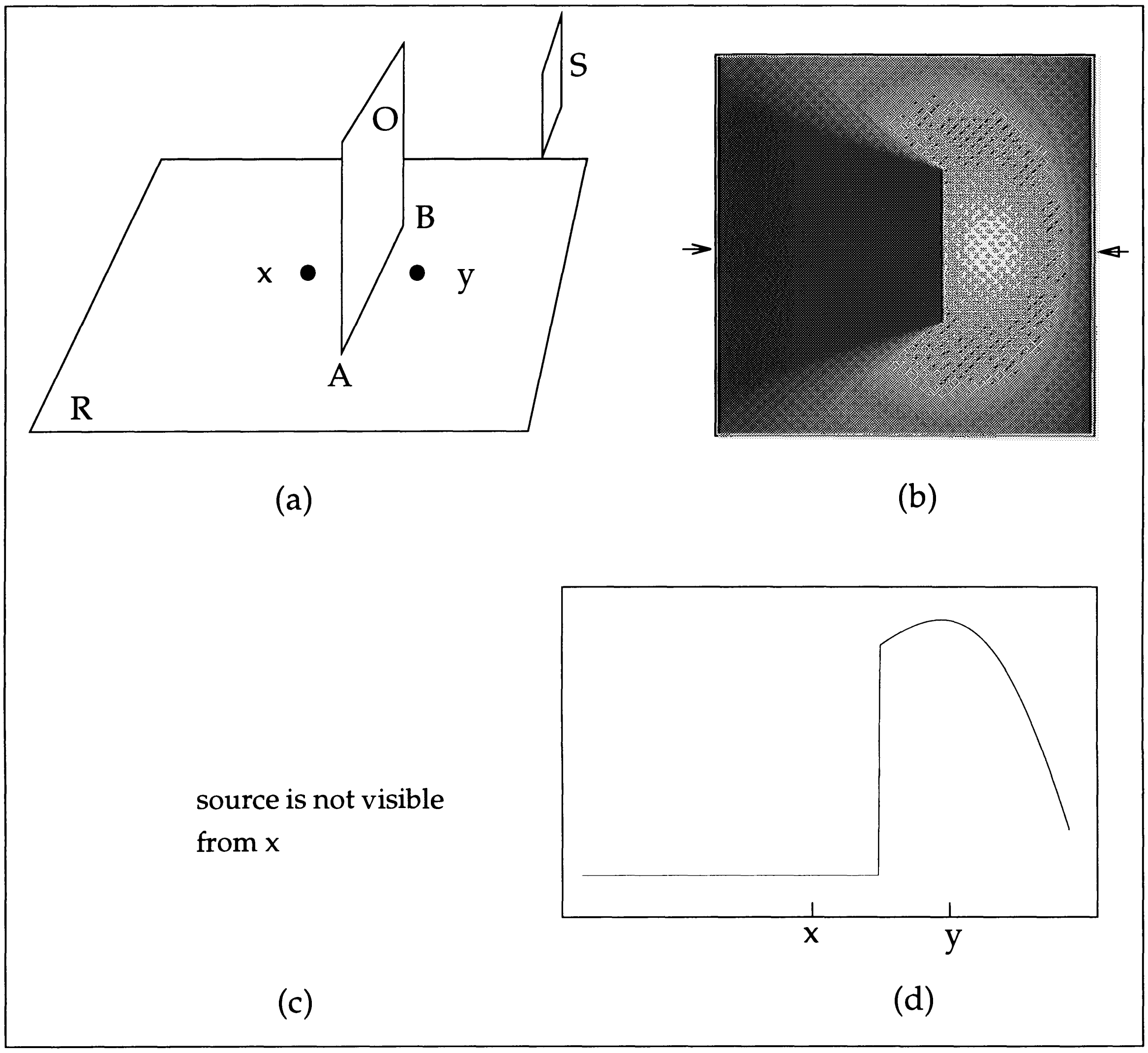


Figure 2.5: An example of a D^0 discontinuity. (a) Edge AB of the occluder O lies on the receiver R. (b) The radiance function over R. (d) The radiance function along the line through x and y.
Points on R immediately to the left of AB cannot see the source S and the radiance there is zero. However, immediately to the right of AB, the entire source is visible, and the radiance there is non-zero. Thus, the radiance function is discontinuous along AB. The points A and B are points of singularity in the radiance function.

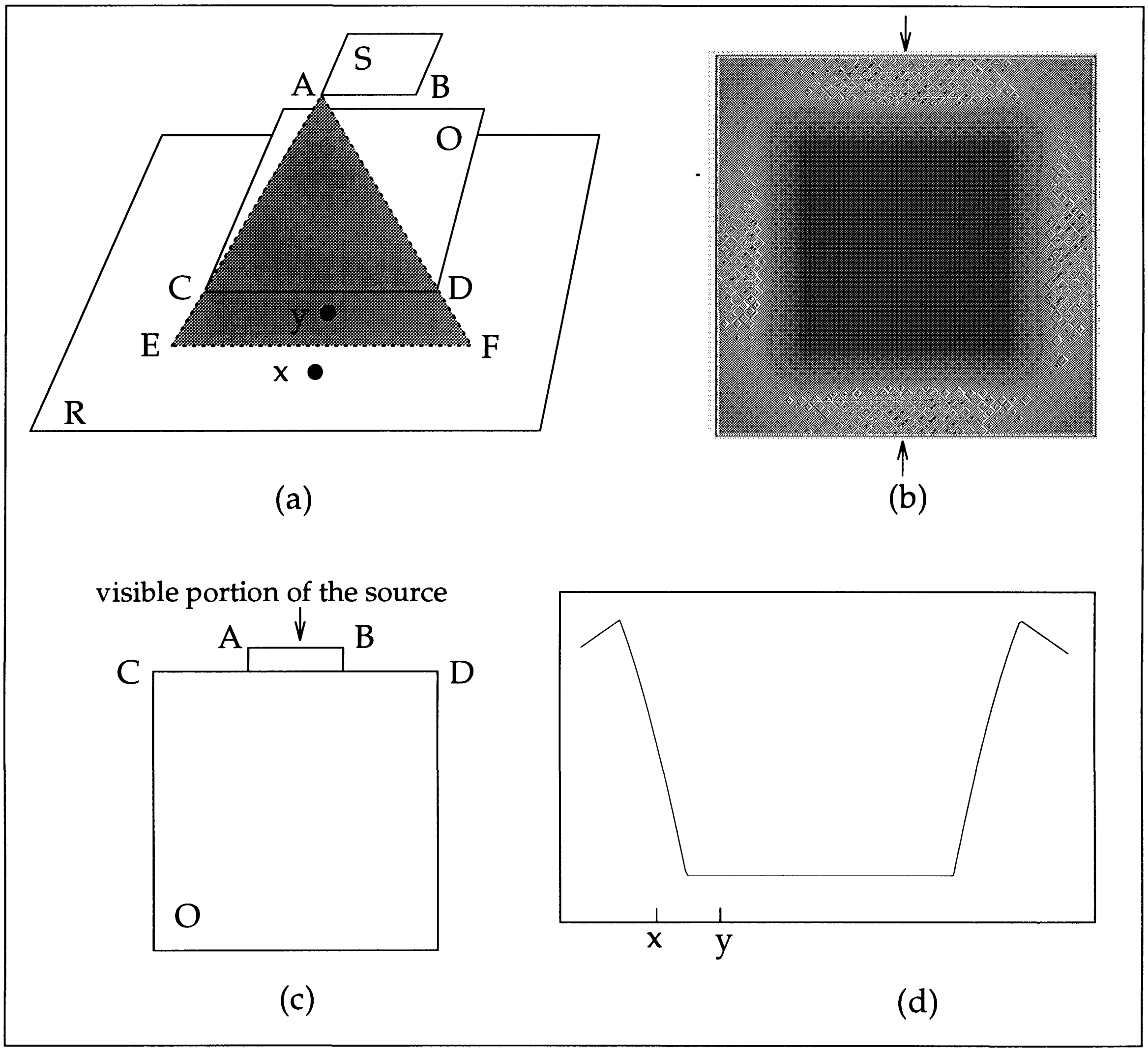


Figure 2.6: A D^1 discontinuity caused by an edge-vertex (EV) event. (a) Edge AB of the light source S is coplanar to edge CD of the occluding polygon O. The critical surface (shaded area) defined by A and CD intersects the receiving plane along EF. (b) The radiance function over R. (c) The occluder O and the light source S as seen from x. (d) The radiance function along the line through x and y. From point y on R, none of the source is visible, hence the radiance there is zero. As we move from y towards x, part of the source adjacent to AB becomes revealed. The visible area grows linearly in the displacement from EF towards x. Thus, along EF the radiance function has a D^1 discontinuity.

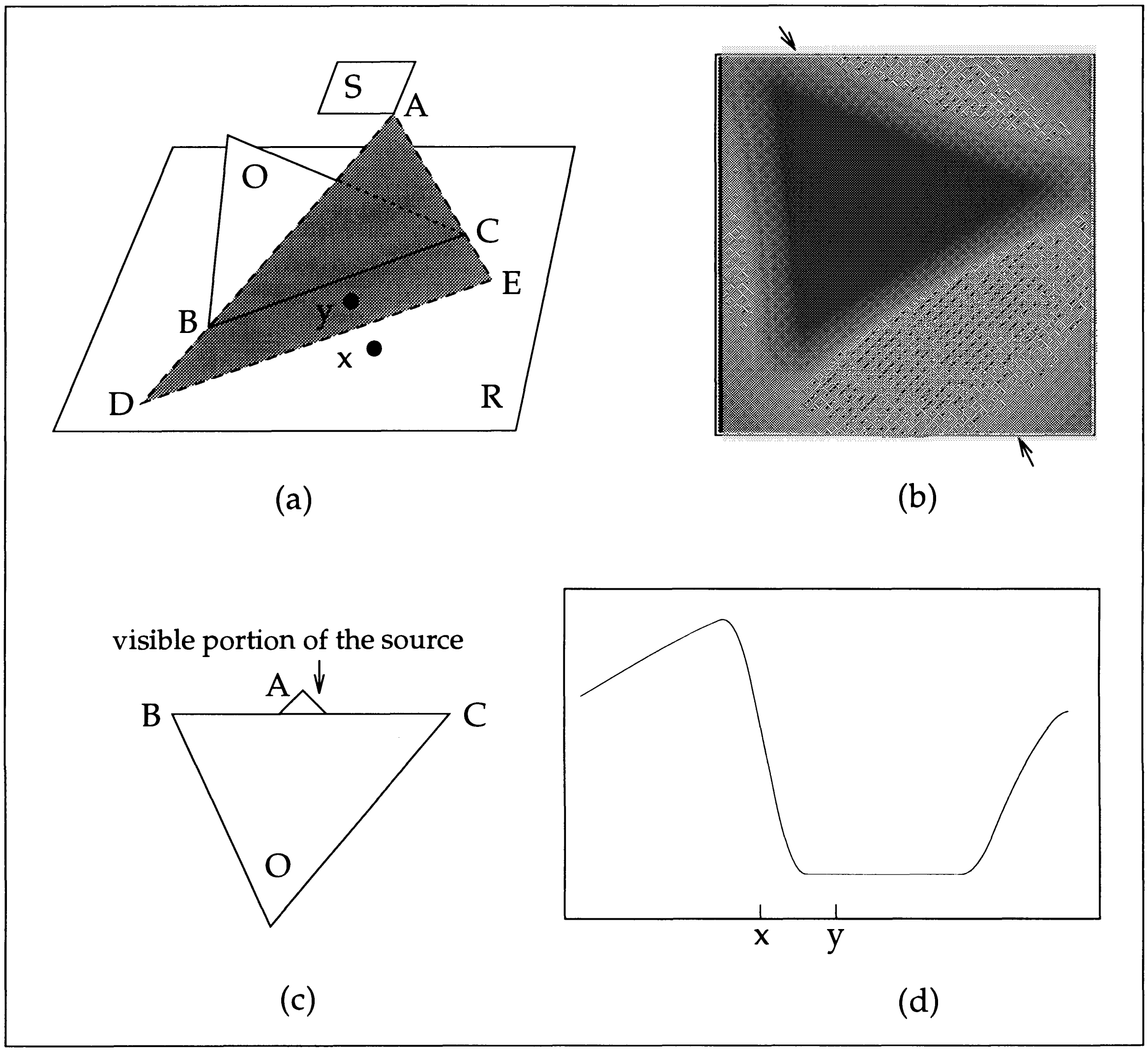


Figure 2.7: A D^2 discontinuity caused by an edge-vertex (EV) event. (a) The critical surface (shaded area) defined by vertex A and edge BC intersects the receiving plane along DE. (b) The radiance function over R. (c) The occluder O and the light source S as seen from x. (d) The radiance function along the line through x and y.

From point y on R none of the source is visible, hence the radiance there is zero. As we move from y towards x, part of the source adjacent to vertex A becomes revealed. The visible area grows quadratically in the displacement from DE towards x. Thus, along DE the radiance function has a D^2 discontinuity.

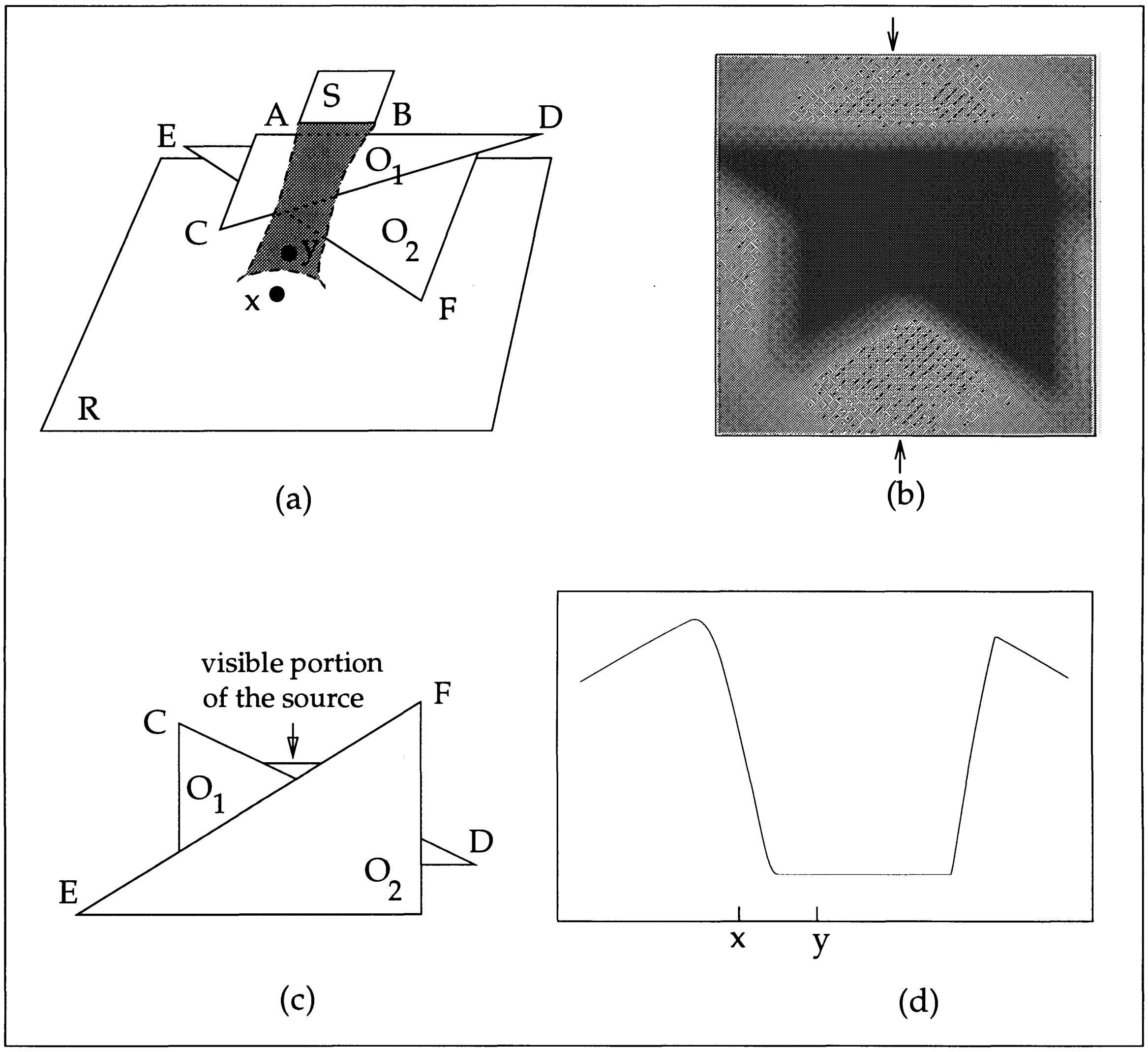


Figure 2.8: An edge-edge-edge (EEE) event. (a) The quadric critical surface (shaded area) defined by the three edges *AB*, *CD*, and *EF* intersects the receiver *R*, resulting in a conic critical curve (shown dotted). (b) The radiance function over *R*. (c) The occluders O_1 and O_2 and the light source *S* as seen from *x*. (d) The radiance function along the line through *x* and *y*.
As we move from *y* towards *x*, part of the source becomes revealed. A displacement from the critical curve towards *x* results in quadratic growth in the visible source area, hence the discontinuity along that curve is D^2. This example illustrates that when several occluding obstacles are involved, the boundaries between umbra and penumbra regions on a receiver may be curved.

noticeable as well.

EEE events occur on a subset of a ruled quadric surface defined by the family of lines that go through three skew edges, each visible to the others. The critical surface is the set of all the points from which the three participating edges can be seen intersecting in one point. The critical curves are conics and generally correspond to D^2 discontinuities as demonstrated in Figure 2.8.

In the examples shown in the figures all the visual events involve either a vertex or an edge of the source. However, visual events can be defined by edges and vertices belonging to any object in the environment. Thus, in an environment with m edges, there can be $O(m^2)$ EV critical surfaces, and $O(m^3)$ EEE critical surfaces. These can give rise to $O(m^3)$ and $O(m^4)$ critical curves, respectively. In general, the event will cause a discontinuity in L_{ij} if it is defined by edges and vertices located between the receiver i and the source j and if it is visible from both. Figure 2.9 shows the critical curves resulting on the receiver for the examples depicted in Figures 2.5, 2.6, 2.7, and 2.8.

2.4.3 Propagation of Discontinuities

The preceding discussion assumed that the radiance function L_j was smooth. This is generally true for primary light sources, but, as the analysis above just showed, secondary light sources can have radiance functions exhibiting discontinuities of various orders. The effect of these discontinuities was summarized by Heckbert [HECK91a] as the "Discontinuity Propagation Law", which states that D^k discontinuities on the source can result in D^{k+1} and D^{k+2} discontinuities on the receivers. In general, discontinuities of higher orders are less noticeable than low order ones.

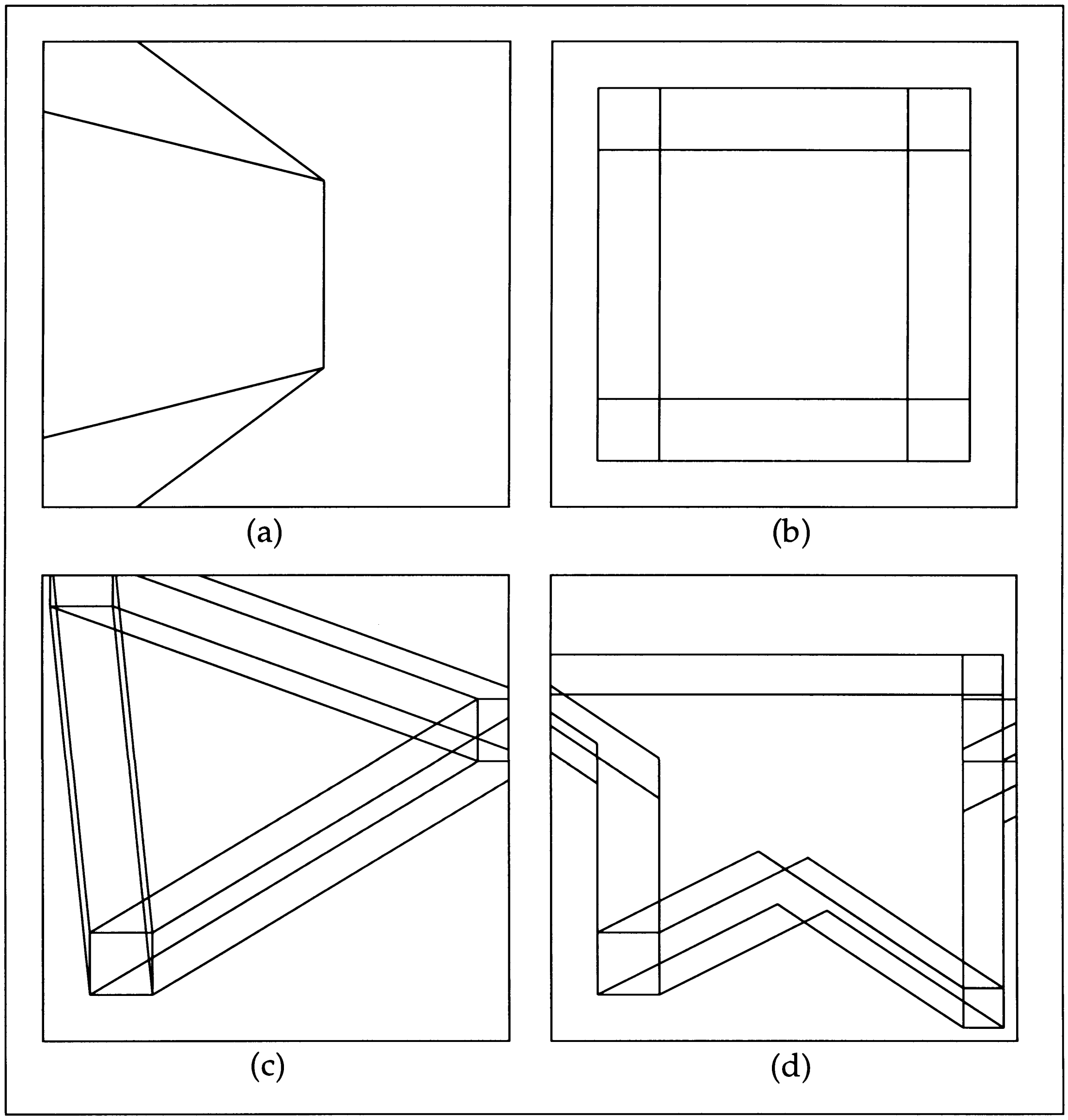


Figure 2.9: Critical curves on receiver R for the simple scenes shown in: (a) Figure 2.5, (b) Figure 2.6, (c) Figure 2.7, and (d) Figure 2.8

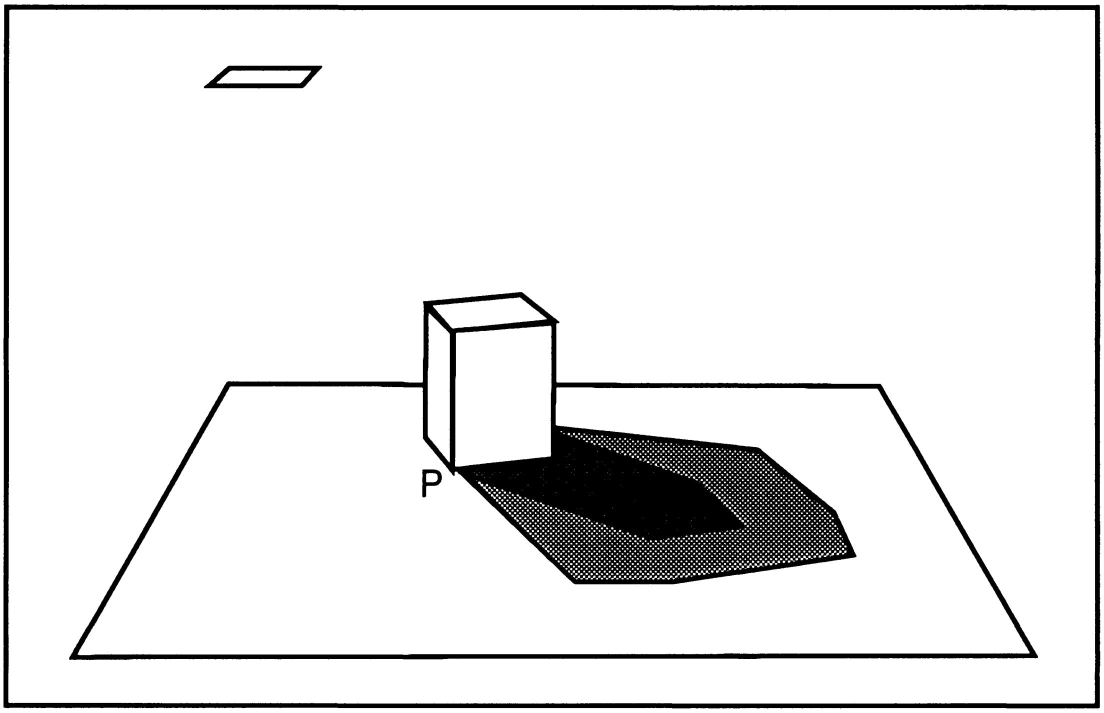


Figure 2.10: Example of function singularity

2.4.4 Singularities

Radiance functions can exhibit *singularities*. These may rise at the endpoints of D^0 discontinuity lines where the radiance gradient becomes infinite.

Figure 2.10 shows an example of a singularity. The dashed lines identify the umbra and penumbra boundaries of the shadow cast by the cube onto the floor. A penumbra region lies between the two boundaries and provides a continuous transition between the unoccluded and totally occluded regions of the floor. As we move in towards the cube, though, the width of the penumbra gets smaller and smaller and the gradient of the radiance function becomes larger and larger. In the limit, under the corner of the cube, the gradient becomes infinite.

This corresponds to a natural phenomena we have all experienced in the environment around us: shadow boundaries are sharper close to the object casting the shadow and get fuzzier moving away from it. Furthermore, the quality of the shadow in the immediate vicinity of the cube helps us determine whether or not the cube is actually in contact with the floor. Thus, correctly reproducing these singularities increases the realism of our simulations.

Chapter 3

Previous Work

3.1 Radiosity

The radiosity approach is a physically-based method that attempts to solve the global illumination problem according to the principles of transfer and conservation of energy; it was derived from techniques used in thermal engineering for the determination of radiative heat exchange in enclosures [SIEG81] and was first introduced to the computer graphics community by Goral *et al.* [GORA84].

Since its first appearance in 1984, radiosity methods have been extended to occluded environments [COHE85], textured surfaces [COHE86], specular surfaces [IMME86,WALL87,SILL89], light sources of arbitrary directional distribution [WALL89,DORS91], arbitrary bidirectional reflectance distribution functions [SILL91], dynamic environments [BAUM86,GEOR90,CHEN90b], curved surfaces [ZATZ92], and complex environments [COHE88b,HANR91b,SMIT92].

3.1.1 Diffuse Radiosity

Most radiosity algorithms are restricted to ideal diffuse reflectors and attempt to provide a *view-independent* solution to the global illumination problem. Thus, "walk-thoughs" of static diffuse environments can be often generated at interactive rates from a single radiosity solution.

Radiosity algorithms use a discrete formulation of the diffuse global illumination problem. The traditional radiosity equation and its derivation were given in Section 2.3. When the environment is subdivided into a set of small surface areas, or *elements*, of constant radiosity and constant reflectivity, the radiant energy transfer between surfaces is governed by a simple system of linear equations. This can be expressed in matrix notation as:

$$(I - M)b = e \tag{3.1}$$

where

$M \in \mathbb{R}^{n \times n}, M = [M_{ij}] = [\rho_i F_{ij}]$, is the form factor matrix;

$b \in \mathbb{R}^n, b = [B_i]$, is a vector of unknown element radiosities;

$e \in \mathbb{R}^n, b = [B_i^e]$, is a vector of emitted element radiosities.

The matrix $I - M$ is a so-called "M-matrix," i.e. all the diagonal elements are positive and all the off-diagonal elements are negative. Furthermore, for physical reflectivities ($\rho_i \in [0, 1]$), the matrix is also strongly diagonally dominant. Many nice properties are known about these matrices [VARG62], including nonsingularity and the convergence of Jacobi and Gauss-Seidel iterations.

3.1.2 Full Matrix Solution

The early radiosity algorithms were organized in three major steps:

1. Compute the matrix M of form factors.

2. Solve for $b = (I - M)^{-1}e$.

3. Display the results.

The first radiosity implementation in computer graphics [GORA84] used Gaussian elimination to invert the matrix $I - M$. Later, Cohen *et al.* [COHE85] greatly reduced the cost of Step 2 by using Gauss-Seidel iteration to solve for the unknown radiosities. Unfortunately, though, solving the system of linear equations is not the most expensive step in the process. Computing the form factors between all pairs of elements in an environment, even when employing a fast hardware z-buffer for the hemicube form factor calculation algorithm [COHE85], is very computationally intensive. Furthermore, the cost of storing the form factor matrix, even when employing compression techniques to exploit the sparsity of the matrix, is prohibitive for anything but very simple scenes.

Precomputing the entire form factor matrix is not strictly necessary. The form factors could be computed one row at a time during the Gauss-Seidel iteration. This strategy, though, would require recomputing each row of form factors at each iteration and thus greatly increase the computation time.

3.1.3 Progressive Refinement Radiosity

The shortcomings of the full matrix approach were overcome by the introduction of progressive refinement radiosity [COHE88b]. This algorithm is based on a refinement operator that uses a single column of the matrix to update the radiosities of all elements in the environment with the contribution from a chosen source.

The difference between conventional radiosity and progressive radiosity is best explained in terms of *gathering* and *shooting*. With Gauss-Seidel iteration, each pass over row i of the form factor matrix results in computing a new estimate of radiosity B_i based on the current estimates for all the other radiosities in b. In physical terms, the estimate B_i is computed by "gathering" in the radiant energy contribution from the rest of the environment. With progressive refinement radiosity, each pass over column j of the form factor matrix results in computing a new estimate of all the radiosities in b based on the current contribution from element j. In effect, element j "shoots" its energy out into the environment.

Since each refinement operation effects the radiosities of the entire environment, useful intermediate results can be displayed even at the very beginning of the solution process. The initial rate of convergence can be improved greatly by shooting at each iteration from the element with the highest remaining energy contribution rather than visiting each column in a strictly sequential order.

This algorithm has the same asymptotic convergence rate as Jacobi and Gauss-Seidel iterations. However, by intelligently observing that the global illumination in a scene is mostly determined by the dominant effect of primary and a few secondary sources, progressive radiosity comes close to the final solution in a fraction of the time that it would take for convergence. In practice, the iteration process is seldom carried to convergence and only a small fraction of the form factor matrix is ever computed.

3.2 Meshing

Most radiosity algorithms to date attempt to capture the variations in illumination across a surface in a scene by discretizing the surface into pieces small enough that the radiance function across each piece can be closely approximated by a simple polynomial [GORA84]. This process is referred to as *meshing*.

Meshing is perhaps the most problematic module of any radiosity system and is responsible for many of the visual artifacts typical of radiosity images. The next section will use uniform meshing as a way of example to illustrate these problems. The following sections will discuss different meshing algorithms and their ability to eliminate or reduce such problems.

3.2.1 Uniform Meshing

The first radiosity algorithms introduced by Goral *et al.* [GORA84], Cohen and Greenberg [COHE85], and Nishita and Nakamae [NISH85a] were based on uniform meshing. Each surface in the environment was meshed in parametric space into a set of rectangular elements by splitting each surface along the lines of a regular grid. Radiance samples were then taken at the center of each element.

This simple technique is easy to implement, but, as many researchers pointed out [HAIN91b,BAUM91,WALL92], it results in a large variety of visual artifacts:

Missing shadows. If the shadow cast by an obstacle in the scene onto a receiving surface falls within sample points, it will be entirely missed (see the legs of the table closest to the back wall in Figure 3.1.)

Blocky shadows. If a shadow area is undersampled, its shape will be distorted to resemble the underlying mesh structure. Also, due to the interpolation methods used for reconstruction, penumbra areas will usually expand to the sample points immediately outside the true penumbra area (see, for

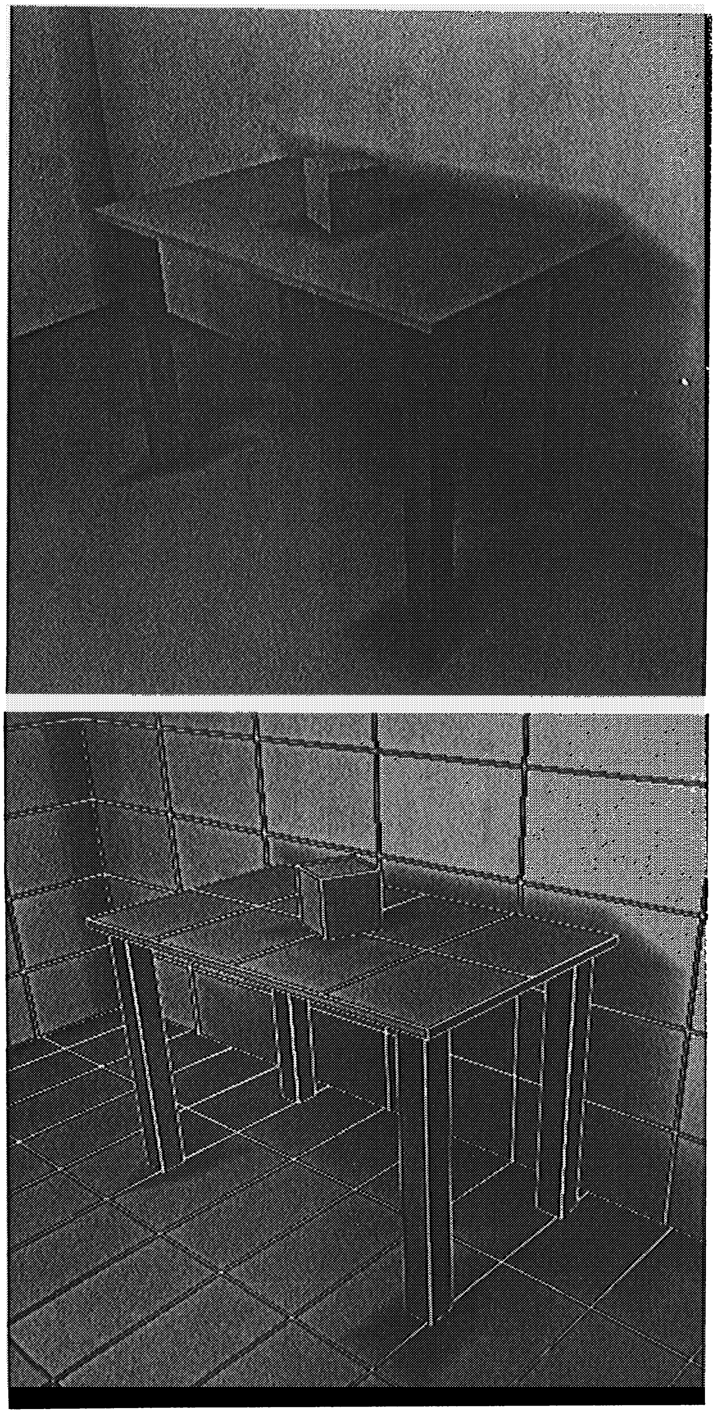


Figure 3.1: Visual artifacts due to uniform meshing. Missing shadows (the legs of the table closest to the back wall.) Blocky shadows (the shadow of the table on the floor and back wall.) Jagged shadow boundaries (the shadow of the leftmost leg of the table.) Shadow leaks (the shadow under the cube and behind the table.) Floating objects (the legs of the table closest to the back wall.) Mach banding (the region of the floor under the table.)

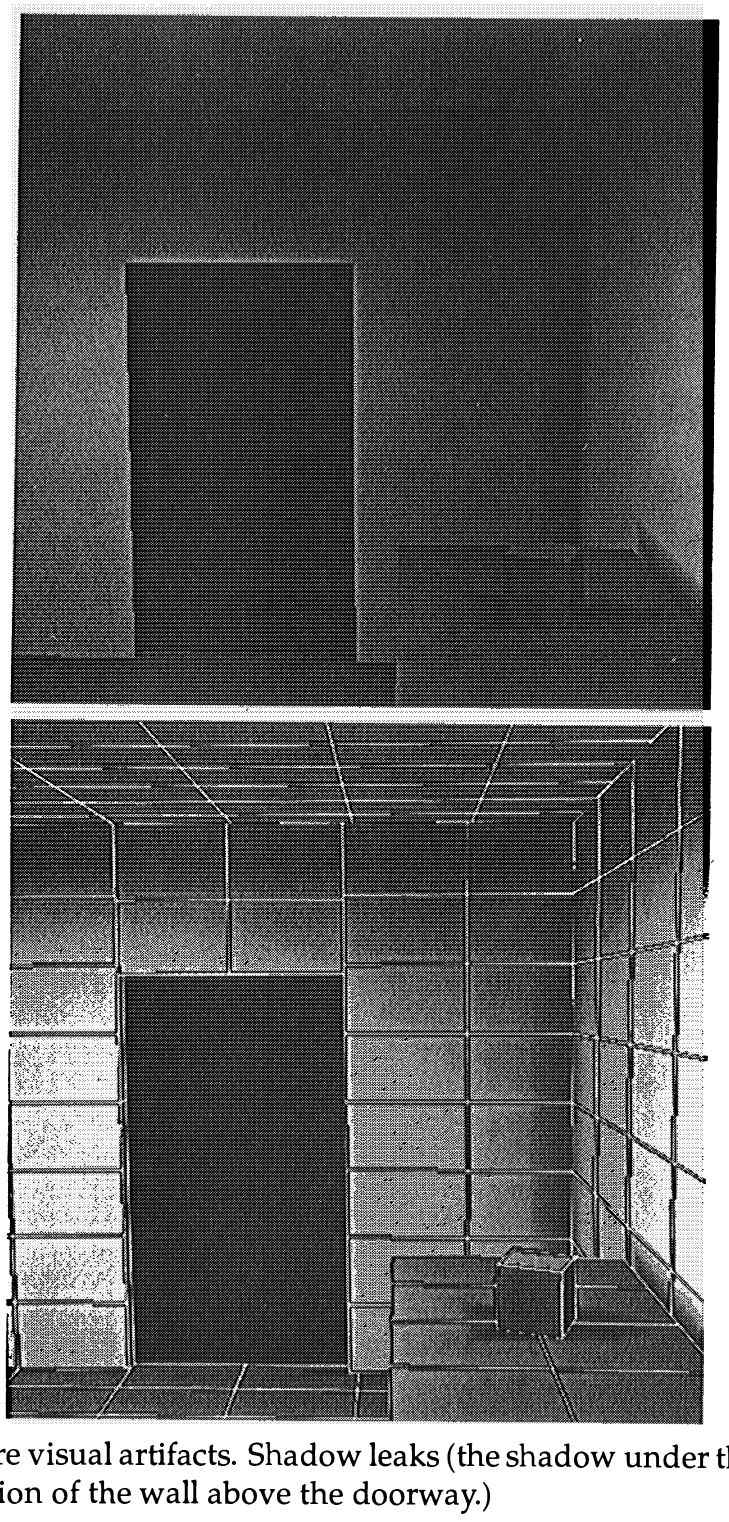


Figure 3.2: More visual artifacts. Shadow leaks (the shadow under the doorway.) Seams (the region of the wall above the doorway.)

example, the shadow of the table on the floor and back wall in Figure 3.1.) Reducing the element size will reduce, but not eliminate, these problems.

Jagged shadow boundaries. Unless the shadow boundaries are parallel to one of the two axis of the mesh, shadow boundaries will appear jagged (see the shadow of the leftmost leg of the table in Figure 3.1.) This aliasing can be eliminated, for a given view, by ensuring that the projected area of each mesh element be smaller that that of a screen pixel.

Shadow/light leaks. If a sharp variation in the illumination falls between sample points, the reconstruction process will smooth it and stretch it, distributing the variation uniformly along the area between the two samples. This can result in dark shadow regions "leaking" into bright illuminated regions and vice versa (see the shadow leaks under the cube and behind the table in Figure 3.1 as well as that under the doorway in Figure 3.2.)

Floating objects. Missing shadow and shadow and light leaks can cause objects to look as if they were floating above the floor (see the legs of the table closest to the back wall in Figure 3.1.)

Mach banding. Most radiosity algorithms to date use linear interpolation to reconstruct the radiance function over a surface. This method introduces unwanted discontinuities in the illumination gradient along the mesh element boundaries. These discontinuities are perceived by our visual system as *Mach bands* (see the Mach bands on the region of the floor under the table in Figure 3.1.) Increasing the mesh resolution usually helps reduce these artifacts. However, better reconstruction methods could be used, e.g.

higher order interpolation, that would greatly reduce Mach banding without needing to increase the sampling rate.

Seams. A complex surface can be modeled as a set of polygons. Each polygon is then independently meshed into elements. If the polygon meshes are not aligned or have different granularities, radiance discontinuities may result along the boundaries between neighboring polygons (see the region of the wall above the doorway in Figure 3.2.) The problem of misaligned meshes has been thoroughly examined by Haines [HAIN91b], and Baum *et al.* [BAUM91], The interested reader is invited to refer to their work for an overview of the possible solutions.

Missing and incorrect shadows, jagged shadow boundaries, shadow/light leaks, and floating objects are all due to undersampling of the radiance function. These artifacts occur in areas where the illumination gradient is highest and thus undersampling is most noticeable.

Increasing the sampling rate by reducing the mesh element size will reduce the artifacts at the cost of increased storage and computing time. Since regular sampling is used, increasing the sampling rate to avoid undersampling shadow boundaries can also lead to oversampling in regions that are uniformly illuminated and where the radiance varies slowly, thus resulting in unnecessary computations.

Deciding what the best mesh resolution is for each surface in a scene is generally generally left to the user. Generating high quality radiosity solutions takes many tedious and time consuming iterations and requires the user to have an unnecessarily high understanding of the workings of meshing and radiosity algorithms. Despite these manual adjustments, the resulting radiosity solutions are generally accurate for only a small range of views and thus, are not truly view-independent. For example, zooming in on a shadow is likely to reveal a jagged boundary.

3.2.2 Adaptive Subdivision

Cohen *et al.* [COHE86] introduced an adaptive subdivision scheme for refining the mesh resolution in regions where the radiance function displays high intensity gradients. Their method meshes the environment into a two-level hierarchy: the first level consists of a coarse subdivision of the environment into patches, the second level further subdivides patches into smaller elements. First, all the patch-to-patch form factors are computed using the hemi-cube method [COHE85] and a system of linear equations is solved to yield the patch radiosities. Then element-to-patch form factors are computed and each element radiosity is calculated as a simple linear combination of the patch radiosities. If the radiosity gradient over an element exceeds a user-specified threshold, the element is subdivided into a set of new smaller elements. The radiosities at these new elements are then calculated exactly as done previously for the other elements. Notice that since the subdivision of an element does not affect the geometry of the environment, the radiosities of all the other elements do not change. This subdivision process can be applied repeatedly to refine the quality of the solution.

This substructuring scheme represents an important step forward over the full-matrix solution method based on uniform meshing [GORA84,COHE85]. Substructuring makes it possible to discretize the environment into many more elements than could ever be handled by the previous scheme, both because of the cost of computing all the element-to-element form factors and because of the enormous memory required to store the full element-to-element matrix.

Provided the size of the initial patches is not too coarse, this adaptive meshing scheme automatically refines the mesh around umbra and penumbra boundaries without unnecessarily refining uniformly illuminated areas. This approach reduces both the need for user intervention and the time and memory costs required to produce an accurate radiosity solution. Most of the problems related to uniform meshing are somewhat reduced, but none are truly eliminated.

The two-level hierarchical organization of the mesh and the adaptive subdivision scheme have also been used within the context of progressive refinement radiosity [COHE88b,WALL92].

3.2.3 Hierarchical Meshing

Hanrahan *et al.* [HANR90,HANR91b] generalized the substructuring approach to a full hierarchical meshing scheme. The surface mesh is stored as a quadtree and each level of the quadtree represents a different level of subdivision. The distinction between source and receiver is blurred and energy transfers can take place between nodes at any level in the respective hierarchical meshes of a source-receiver pair (see example in Figure 3.3.)

If energy is transferred to an internal node, this energy is "pushed down" to the leaves of the structure by recursively distributing the energy to the node's children in proportions determined by their relative areas. The energy stored at the leaves is then "pulled up" and the energy of each interior node is computed as an area weighted average of its children's energies.

Transferring energy between large patches (nodes close to the root of their re-

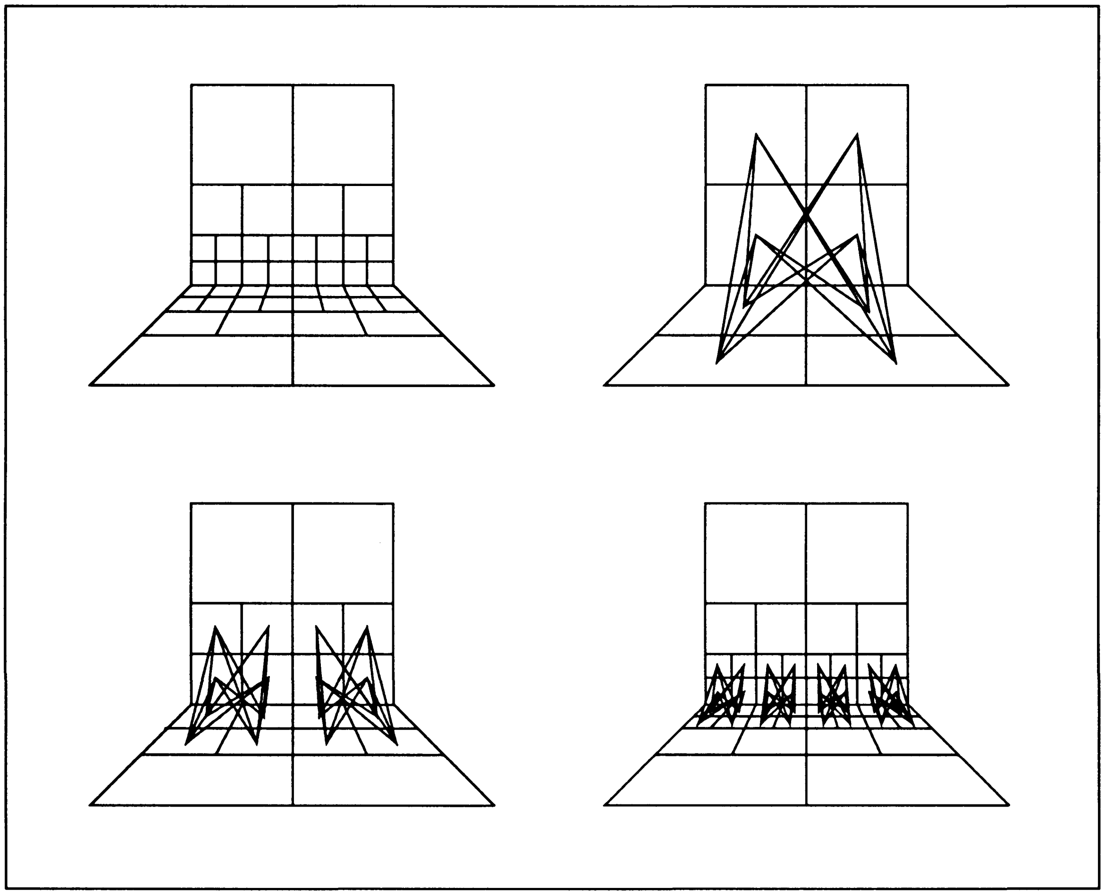


Figure 3.3: Hierarchical meshing on a pair of perpendicular polygons. The line segments show the interactions between pairs of elements at different levels in the quadtree hierarchies (after Hanrahan [HANR91b].)

spective hierarchies) reduces the number of source-receiver interactions needed to produce a radiosity solution; transferring energy between small patches reduces errors associated with the computation of form factors and visibility terms.

Hanrahan *et al.* compute an upper bound on the error incurred in transferring energy between two nodes. Energy between a source-receiver pair is then transferred by subdividing source and receiver until all the interactions fall within the error bound.

With this scheme, the number of interactions for an environment of m patches and n elements (leaf nodes) is reduced from $O(mn)$ to only $O(n)$. Furthermore, this meshing scheme is completely automatic.

On the other hand, however, requiring that all energy transfers occur in quantities smaller than a given threshold limits the efficiency of the algorithm. Also, the errors incurred gathering energy from different patches can compound, but no error bound for the final solution was shown. Furthermore, although the radiosity solution is quantitatively accurate, because of the point sampling nature of the visibility calculations and of the constant elements used to subdivide the input surfaces, the images computed with this technique can still exhibit all the visual artifacts produced by its predecessors.

3.2.4 Discontinuity Meshing

All of the meshing techniques discussed above mesh the input surfaces according to local geometric considerations only. Therefore, shadow boundaries and other sharp variations in the illumination will generally fall within the interior of elements; since reconstruction is usually done by interpolating the radiance values at the element vertices, these illumination details will be distorted, smoothed, or even entirely missed.

These problem can be eliminated by explicitly representing discontinuities in the radiance function and its derivatives as boundaries in the mesh. *Discontinuity meshing* methods implement this idea and can greatly improve the quality of radiosity images, even though these methods only capture a subset of the discontinuities described in the Section 2.4.

Capturing value discontinuities

Baum *et al.* [BAUM91] preprocess the environment to split surfaces along lines of intersection or contact. Thus, the corresponding D^0 discontinuities are marked by edges in the mesh. The radiance samples stored with these edges are the values corresponding to the illuminated side of the edge. The problem of storing both the occluded and unoccluded radiance values is avoided by making all D^0 edges lie at the perimeter of the mesh. If a surface A lies on another surface B, the entire area covered by A is dropped from the mesh of B (see example in Figure 3.4.)

This approach has several advantages over previous meshing techniques:

- The quality of radiosity images is significantly improved by eliminating floating objects and the most obvious shadow and light leaks.
- The need for user intervention is reduced since value discontinuities are captured automatically, independently of the initial size of the mesh elements.
- Time and storage costs are reduced since adaptive subdivision is no longer needed to narrow in on value discontinuities.

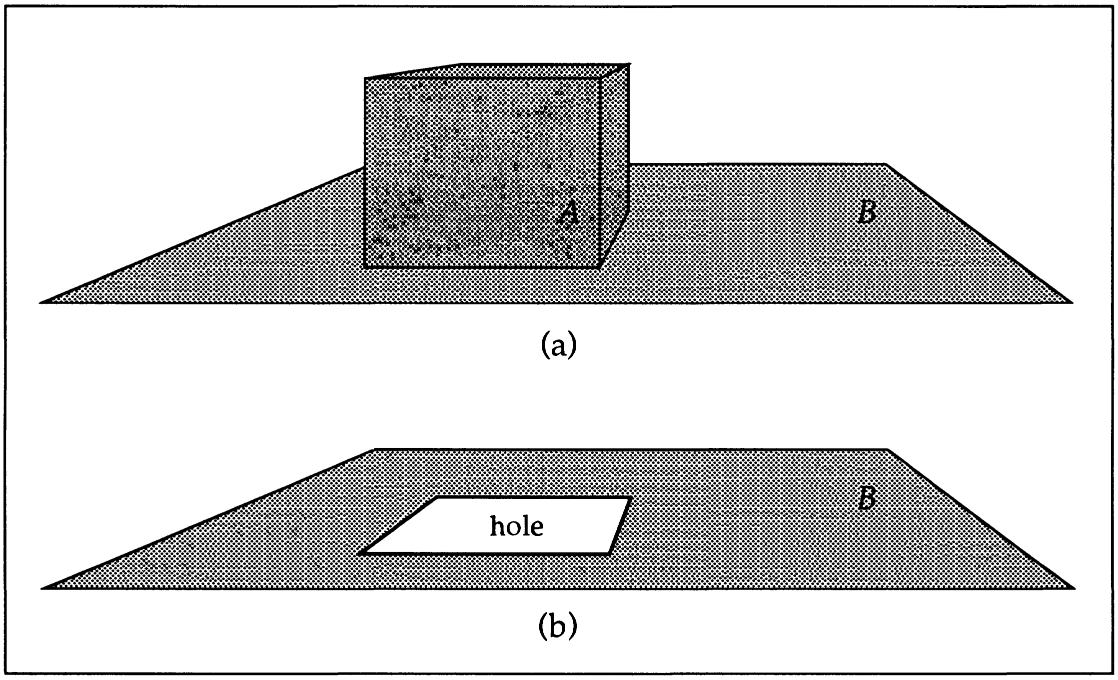


Figure 3.4: Computing D^0 discontinuities prior to meshing. (a) The initial environment has object A resting on surface B. (b) The area covered by object A is removed from surface B; the resulting D^0 edges are shown as thicker lines (object A is not shown.)

This meshing algorithm, however, is still liable to miss small illumination details and generate shadows with imprecise coverage and poor definition.

Capturing umbra and penumbra boundaries

Campbell and Fussell [CAMP90] suggested meshing the environment by projecting each object onto each surface in the environment from the light source (see example in Figure 3.5a.) For point light sources, shadow volumes were used to mesh the environment. Finite area light sources were discretized into smaller parts and each part was approximated by a point light source. This method works well for small light sources, but as the number of point sources needed for an adequate approximation grows, subdivision within regions of penumbra becomes too fine.

Recently, Campbell and Fussell [CAMP91a,CAMP91b,CAMP91c] have proposed an alternative method for polygonal environments whereby umbra and penumbra volumes are used to classify each surface into regions of umbra, penumbra, and total visibility (see example in Figure 3.5b.) Each region is further split into elements, and the illumination at the vertices of each element is computed analytically. Numerical optimization techniques are then used to determine the correct element density inside unoccluded and penumbra regions.

The surface mesh is stored as a two-dimensional BSP tree [FUCH80] with the mesh elements as the leaves of the tree (see examples in Figure 3.5.) Since vertices are not shared among elements, illumination calculations may be recomputed several times at each location; on the other hand, because of this organization, D^0 edges do not require any special treatment.

Umbra and penumbra volumes were previously used by Nishita and Naka-

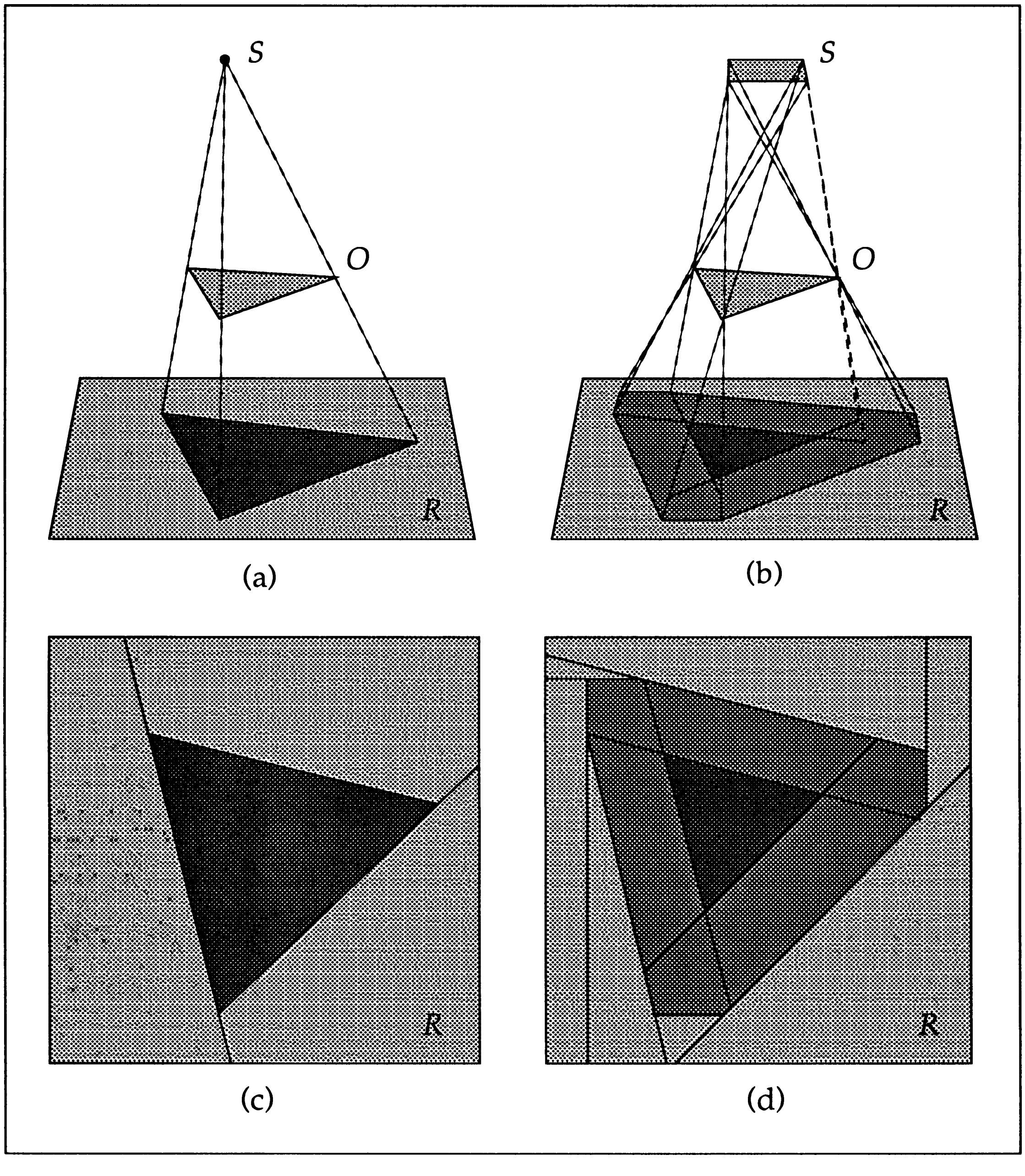


Figure 3.5: Shadow volumes (a) and umbra and penumbra volumes (b) constructed from source S and object O are used to classify surface R into regions of umbra, penumbra, and total visibility. The resulting meshes (c-d) are stored as two-dimensional BSP trees.

mae [NISH83,NISH85a] to classify regions in a scene according to light source visibility. However, this information was not used to mesh the environment in object-space, but rather to speed up the illumination calculations within a screen-space, scanline-based renderer (see discussion in Section 3.3.2.) In fact, Campbell's approach is, to the author's knowledge, the first object-space algorithm to accurately compute shadows cast by area light sources. Many of the visual artifacts typical of previous radiosity algorithms are eliminated without any manual intervention. All the shadows are captured, no matter how small, umbra and penumbra regions have the correct shape and sharp straight boundaries, and floating objects and shadow and light leaks are eliminated.

However, despite the impressive improvements in image quality, a number of problems still remain:

- Umbra and penumbra volumes only capture a subset of the EV events responsible for the discontinuities on a receiving surface. More precisely, only umbra and penumbra boundaries are computed; discontinuities falling within a penumbra area are ignored.

- As Teller pointed out [TELL92], shadow and discontinuity meshing algorithms based on umbra and penumbra volumes only take into consideration visual events originating from the interaction of a light source and a single polygonal occluder. Since multiple interactions are ignored, some umbra areas may be incorrectly classified as penumbras.

- For the same reason, EEE events are ignored. The discontinuity curves generated by these events are conics and can not be correctly represented as edges of a polygonal mesh.

- Mach bands can still appear in the resulting images since linear interpolation is used for reconstruction.
- The problem of radiance singularities is still unaddressed.

Discontinuity meshing in 2D

The idea of including discontinuities in radiance functions and their derivatives as boundaries in the mesh was developed independently by Heckbert [HECK91a, HECK92c] and by Lischinski and the author at Cornell Program of Computer Graphics [LISC91]. In both cases, the ideas were implemented and tested in 2D, showing promising results.

Discontinuity meshing in 3D

Recently, both Heckbert [HECK92a] and Lischinski, Tampieri, and Greenberg [LISC92] extended their ideas to 3D polygonal environments.

Heckbert [HECK92a] constructs a discontinuity mesh that contains all the discontinuity lines arising from EV events in which primary light sources directly participate. First, the critical lines are computed on all the surfaces in the environment, and then Delaunay triangulation and mesh relaxation are used to produce the final discontinuity mesh on each surface.

Lischinski *et al.* [LISC92] construct a different discontinuity mesh for each light source. Each mesh contains the discontinuity lines arising from EV events in which that light source directly participates. A two-dimensional BSP-tree coupled with a topological data structure is used to construct and represent the mesh. Adaptive subdivision is used to increase the element density where necessary. The contributions due to different light sources are merged into a separate data structure which accumulates the total illumination over each surface. A detailed presentation of this method can be found in the next two chapters.

Both algorithms still ignore EEE events and only consider VE (or EV) events involving a light source. As opposed to Campbell's algorithm [CAMP91c], these discontinuity meshing methods correctly capture discontinuities falling within penumbra regions.

Furthermore, in the Cornell work higher order interpolation schemes are used to reconstruct the radiance function and correctly resolve the different orders of discontinuity along mesh edges. This technique reproduces the correct degree of sharpness of various illumination details, effectively resolves radiance singularities, and eliminates the annoying Mach banding typical of linear interpolation schemes.

3.3 Direct Energy Transfer

The ability to compute the light energy contribution of one mesh element to another is one of the most important building blocks of any radiosity algorithm. As was shown in Equation (2.7), this computation requires evaluating a complex integral. Analytical solutions have only been offered for very simple cases, such as that of a source of constant emission, and in the absence of any occluding obstacles.

In order to simplify the problem, traditional radiosity algorithms have discretized the diffuse rendering equation into a simple system of linear equations (see Equation (2.5)). Then, the contribution of element s_j to the illumination of element s_i becomes:

$$B_i = \rho_i B_j F_{ij} \tag{3.2}$$

where

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} v(x, x') \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_j dA_i \tag{3.3}$$

This formulation relies on the assumption that the radiance must be constant across each element in the scene. If this assumption is met, the problem of computing the energy transfer between two elements is reduced to that of computing a form factor, which, in the case of Lambertian reflectors, is a purely geometrical term.

3.3.1 Numerical Form Factors

The computation of form factors is complicated by the presence of a visibility term under the integral. Considerable research efforts have been devoted to the investigation of accurate and efficient methods for computing form factors.

Goral *et al.* [GORA84], who introduced radiosity to the computer graphics community, transformed the area integrals to contour integrals and evaluated them numerically by discretizing the contours into a set of short line segments. This technique was computationally expensive and only simple unoccluded environments were demonstrated.

Cohen and Greenberg [COHE85] were the first to present an efficient algorithm for computing form factors in the presence of occlusions. Their hemicube algorithm was a numerical approximation of the geometric analog for the form factor integral developed by Nusselt [SIEG81]. The inner integral of Equation (3.3) was approximated by a weighted sum of the hemicube pixels covered by the

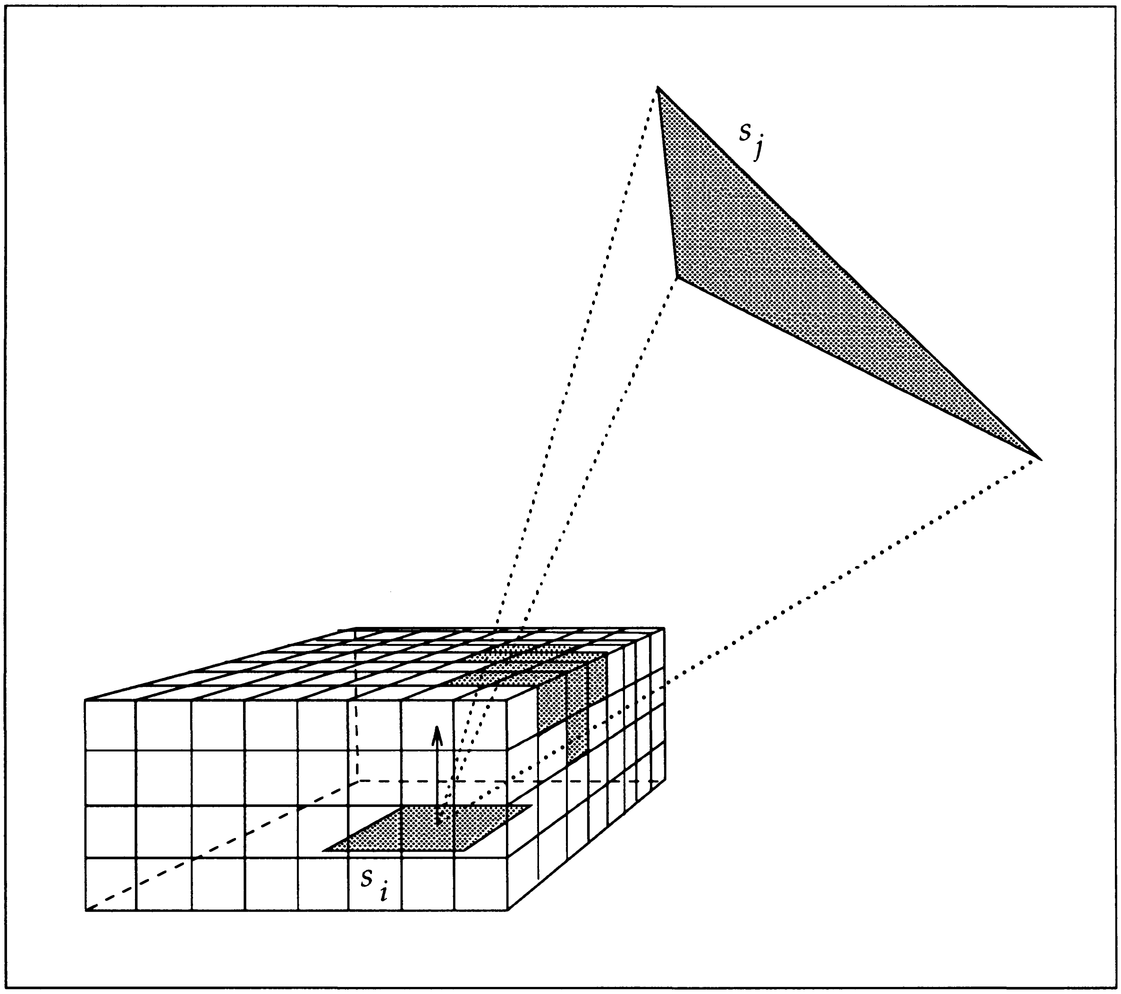


Figure 3.6: The hemicube algorithm. The form factor from surface s_j to the center of surface s_i is approximated by the weighted sum of the hemicube pixels covered by s_j's projection.

projection of element s_j relative to the center of element s_i (see Figure 3.6.) The outer integral was simply dropped by assuming that the distance between the two elements is large compared to their size and that the visibility of s_j is independent on the location on s_i (constant visibility.) The hemicube algorithm can be implemented using a hardware z-buffer and is thus could be very efficient.

Unfortunately, the introduction of progressive refinement radiosity [COHE88b] and its use on larger, more complex, environments, uncovered many problems and inaccuracies connected to the use of the hemicube. As Baum *et al.* [BAUM89] pointed out, in order to accurately compute form factors with the hemicube algorithm, a number of assumptions must be met. The distance between elements must be large compared to their area, the visibility of one element must be the same as seen from any point on the other element, and the area of the projection of one element onto the hemicube must be correctly represented by an integral number of hemicube pixels. The full matrix solution projected patches onto hemicubes centered over mesh elements. Since the patches were large compared to elements and the environments being solved were generally very simple, the hemicube computations were relatively accurate. In progressive refinement radiosity, however, the role of patches and elements was reversed. The Hemicube was placed on large patches, often violating the constant visibility assumption, and small elements were projected onto the hemicube, causing significant aliasing problems and resulting in the patch-to-element form factors being severely over/under-estimated. Furthermore, progressive radiosity made it possible to solve more complex models than could be handled by the full matrix approach, thus making the hemicube artifacts even more apparent.

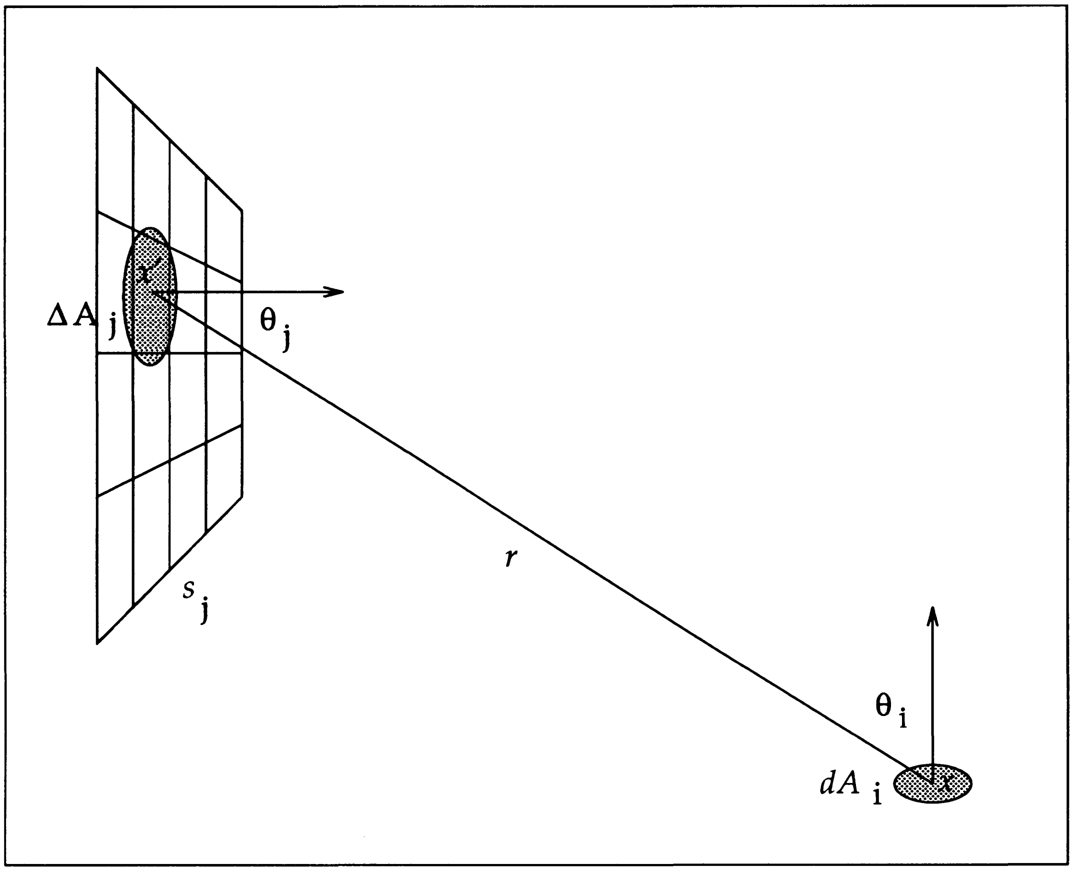


Figure 3.7: Ray-traced form factors. Surface s_j is subdivided into smaller regions; the contribution from each region ΔA_j to the form factor dF_{A_j,dA_i} is approximated by a disc-to-point form factor.

When elements are close to each other or when one of them is a primary light source, Baum *et al.* [BAUM89] proposed replacing the hemicube with an analytical formula for the computation of the inner integral [HOTT67] coupled with a numerical evaluation of the outer contour integral. This method works well for unoccluded form factors. Partial visibility is resolved by recursively subdividing the elements and adding up the new smaller form factors.

A different solution was offered by Wallace *et al.* [WALL89]. Rather than using the traditional finite element formulation of radiosity [GORA84], the authors proposed a point sampling approach. Thus, instead of computing the energy transfer between pairs of elements, the authors computed the radiance contributed by element s_j to a differential area on a receiving surface s_i:

$$B_i(x)dA_i(x) = \rho_i B_j A_j dF_{A_j,dA_i} \tag{3.4}$$

where

$$dF_{A_j,dA_i} = \int_{A_j} v(x,x') \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_j(x') \tag{3.5}$$

The integral was approximated by the formula for a disc-to-point form factor and adaptive subdivision of element s_j was used to reduce the error (see Figure 3.7.) Visibility was estimated using ray tracing.

The ray-traced form factor method is efficient and flexible. While the hemicube method requires that a complete row or column of form factors be computed at once, with the same resolution, the other method allows individual control on the accuracy and computational cost of each form factor.

3.3.2 Analytical Form Factors

All of the methods presented above, though to different degrees, are subject to aliasing due to the point sampling nature of the visibility computation. Nishita and Nakamae [NISH85a] proposed an analytical, exact method of computing the contribution from a source of constant emission to a point. The algorithm was used within the context of a scanline rendering system and was applied to points spaced at intervals of a few pixels across each scanline. The algorithm started by building umbra and penumbra volumes for each source-obstacle pair so that each point on a receiving surface could be easily classified as being in the umbra, penumbra, or unoccluded region with respect to the light source. The

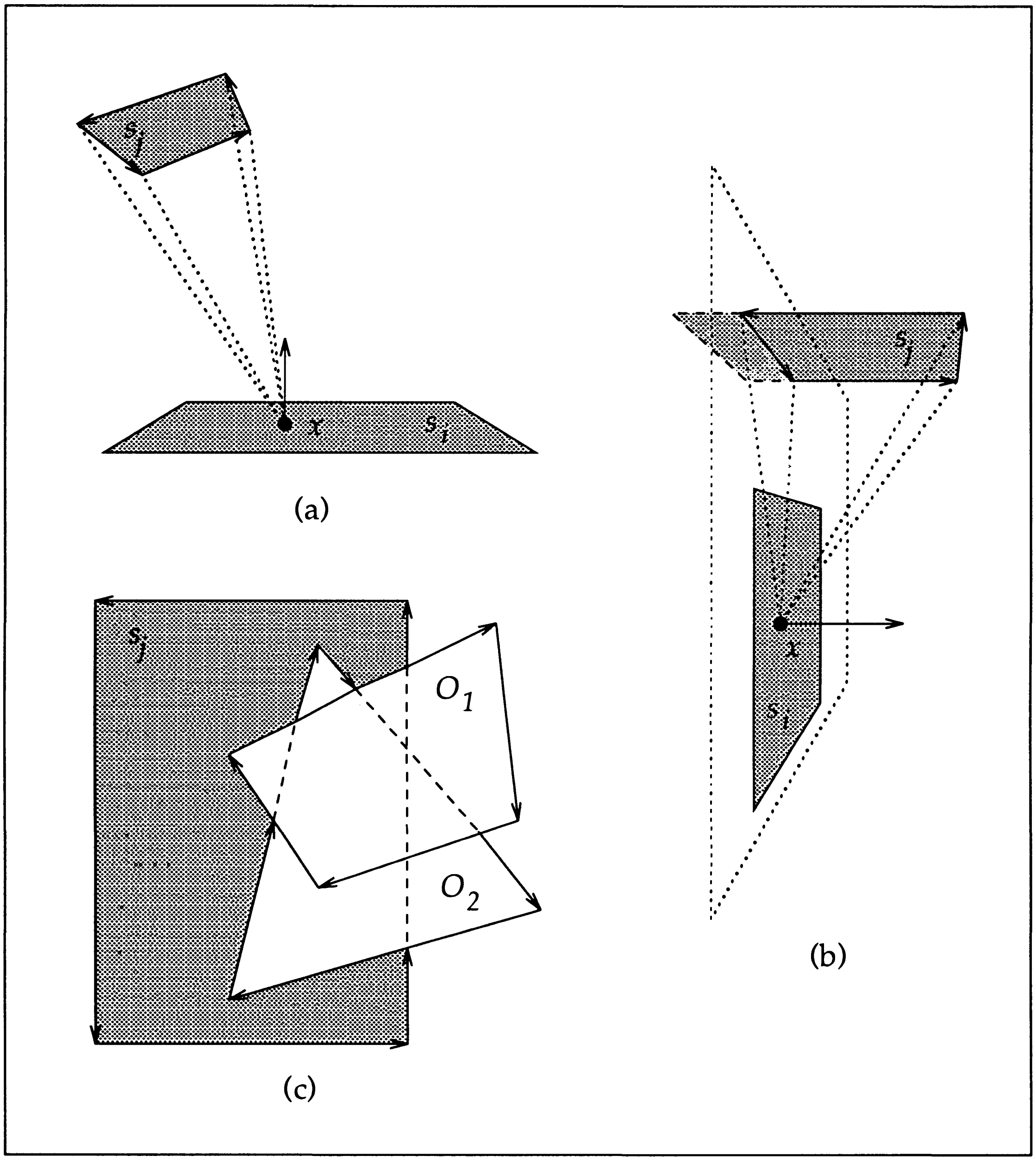


Figure 3.8: Analytical form factors. (a) Contour integration along the path indicated by the arrows is used to compute the contribution of source s_j to point x. (b) Only the part of the source lying in front of the receiver is considered in computing the energy transfer. (c) Obstacles intervening between the source and receiver affect the integration path.

illumination at an unoccluded point was then computed analytically using the contour integral method [HOTT67] (see Figure 3.8a.) If the light source polygon was intersected by the supporting plane of the receiving point, the illumination was computed by integrating along the contour of the part of the light source lying in the front of the receiving plane (see Figure 3.8b.) If the receiving point was in the penumbra area, the illumination was computed by integrating over segments of the contours of the source and obstacles in a way that amounted to integrating across the portions of the source that were visible from the receiving point (see Figure 3.8c.) Computing the illumination at points lying in the penumbra was expensive but yielded the correct result. No approximations were made.

3.3.3 Non-Constant Exitance

In the preceding algorithms, the expression relating the energy contribution of one element to another element or differential area is based on the assumption that the exitance of the source be constant. This assumption is often violated, especially due to the subdivision of the input surfaces into patches and elements first introduced by Cohen *et al.* [COHE86] and now adopted by most radiosity systems. While elements are often small enough to satisfy the constant radiosity assumptions, patches, being clusters of (possibly many) elements, often violate this premise. Furthermore, since patches, rather than elements, are often used as a source, a number of problems may result.

These problems were discussed by Tampieri and Lischinski [TAMP91], who also presented a simple solution, based on ray-traced form factors [WALL89] (see Figure 3.7.) The source patch was recursively subdivided into smaller regions and ray-traced form factors were used. Rather than assuming constant emission across the entire patch, though, the underlying element mesh was interrogated to compute separate radiance values for each region. This method has all the efficiency and flexibility of ray-traced form factors, while at the same time providing an accurate treatment of arbitrary radiance distributions.

The scheme of Hanrahan *et al.* [HANR90,HANR91b] is also immune to the problems connected to violations of the constant radiosity assumptions. By using the hierarchical mesh and subdividing interacting pairs of elements until the energy transferred between the two falls below a given threshold, a source of arbitrary distribution can be subdivided until each piece can be safely treated as a constant emitter.

3.4 Radiance Function Reconstruction

Radiosity algorithms typically compute radiance values only at selected sample locations on the surfaces of the scene. Reconstruction techniques are then used to provide radiance values at any desired location.

Radiance functions must be reconstructed for display and, in the case of progressive refinement radiosity, used for selecting the patch with the highest unshot radiant flux. The former task requires that the reconstructed radiance functions reproduce as many of the characteristics discussed in Section 2.4 as possible. The latter task only requires a rough approximation since only the integral of the radiance over a surface is needed and choosing the wrong patch would not affect the accuracy of the solution anyway.

The traditional radiosity formulation presented in Equation (2.5) discretizes the continuous domain of the diffuse global illumination problem into a finite set of elements using constant shape functions. Higher order shape functions, though, should be used at reconstruction time because they yield more realistic pictures.

3.4.1 Constant Elements

When using constant shape functions, the mesh elements are displayed as flat shaded polygons. From a subjective, or visual, standpoint, this piecewise constant approximation is usually very poor because the value discontinuities introduced by the reconstruction process are very easily perceived by our visual system as sudden "jumps" in the illumination of a surface. These artifacts reveal the artificial discretization of the surface into elements.

3.4.2 Gouraud Shading

In order to improve the visual appearance of radiosity solutions, Goral *et al.* [GORA84] suggested using Gouraud shading [GOUR71]. First, vertex radiosities are computed from the element radiosities, then linear interpolation in screen space is used to compute the radiosities corresponding to the locations within mesh elements that map onto pixel centers.

This technique yields significantly better results that simple flat shading, but is not without problems. The reconstructed radiance functions are now continuous everywhere across a surface, but exhibit gradient discontinuities across element boundaries, which are perceived as Mach bands [RATL72]. Furthermore, as Duff pointed out [DUFF79], when interpolating in screen space, the shading across a surface may appear to change with the view due to both the non-linear mapping of distances effected by the viewing projection and the axis-dependence of the Gouraud shading algorithm for polygons other than triangles.

Despite these drawbacks, this technique is widely adopted in current radiosity systems because Gouraud shading is now available in hardware on most graphics workstations.

3.4.3 Bilinear Interpolation

Cohen and Greenberg [COHE85] reconstructed the radiance function using bilinear interpolation in object space. Unlike Gouraud shading, this method yields a unique value at every point on an element, independent of the view and orientation of the element. Bilinear interpolation, though, like Gouraud shading, introduces unwanted first derivative (slope) discontinuities across element boundaries.

These techniques require radiance values at the element vertices, but, in fact, when they were first proposed [GORA84,COHE85], radiosity systems computed radiance values at the center of the elements. Cohen and Greenberg [COHE85] computed vertex radiance values as a simple average of the radiances of the surrounding elements. This technique works reasonably well for regular rectangular meshes, but for triangular or arbitrary quadrilateral meshes its not at all clear what choice of weights should be used in computing the averages.

Furthermore, reconstructing the radiance function from the radiance samples now takes two interpolation stages, each one introducing further errors and approximations into the solution. A simple solution to this problem was offered by Wallace *et al.* [WALL89], who proposed computing the radiance values directly at the element vertices, thus reducing the reconstruction process to a simple interpolation step.

3.4.4 Higher Order Interpolation

Higher order interpolation schemes have been proposed as an alternative to linear interpolation. Reichert [REIC92] described a C^1 quadratic interpolation scheme, based on work done by Cendes and Wong [CEND87], that imposes first derivative continuity across the mesh edges and thus greatly reduces visual artifacts. Unfortunately, this technique also has the effect of washing out sharp illumination details such as umbra and penumbra boundaries.

All of the interpolation methods discussed above yield a reconstructed radiance function that is infinitely differentiable inside each individual mesh element. Thus, no shadow boundary crossing an element can be correctly reconstructed. This results in the jagged shadow boundaries, expanded or exaggerated penumbrae, floating objects, and shadow and light leaks, so typical of radiosity images.

As first demonstrated by Campbell and Fussell [CAMP90], these problems can be greatly reduced and sharp umbra and penumbra boundaries can be convincingly rendered if the surfaces in the scene are split into mesh elements along these boundary lines.

Chapter 4

Single Source Discontinuity Meshing

The previous chapter showed that the conventional meshing algorithms used by most radiosity systems to date are the cause of many inaccuracies in radiosity solutions. In particular, umbra and penumbra boundaries and other sharp variations in the illumination across a surface are poorly reproduced if not entirely missed.

The analysis of radiance functions presented in Section 2.4 showed that these illumination changes correspond to discontinuities in the radiance function and its derivatives and occur along critical curves determined by the geometric interaction of the objects and light sources in the environment.

Conventional radiosity algorithms ignore these discontinuity curves. If a critical curve crosses a mesh element, the reconstructed radiance will be inaccurate. In fact, the piecewise polynomial interpolation methods used to reconstruct the radiance function over a surface mesh are unable to reproduce a discontinuity that crosses a mesh element.

The ability to predict the location of critical curves and the order of the corresponding discontinuities can be used to design a new algorithm less prone to the problems that afflict conventional radiosity. The new approach explicitly represents the discontinuities in the radiance function and its derivatives as boundaries in the mesh and uses a piecewise polynomial interpolation scheme to correctly reproduce these discontinuities in the reconstructed radiance function.

This concept, known as *discontinuity meshing*, was first introduced by the author and Lischinski at the Cornell Program of Computer Graphics [LISC91] and, independently, by Heckbert [HECK91a,HECK92c].

In the succeeding two chapters a radiosity algorithm based on discontinuity meshing and capable of producing accurate solutions to the diffuse global illumination problem for three-dimensional polygonal environments is discussed. The presentation is organized in two main parts: the rest of this chapter discusses a discontinuity meshing algorithm that computes the direct illumination due to a single primary light source of finite area and constant emission. The next chapter extends the basic algorithm to secondary light sources of arbitrary emission and uses the result as a refinement operator within a progressive-style radiosity algorithm to produce a diffuse radiosity solution.

4.1 Single Source Discontinuity Meshing—Algorithm Overview

The direct illumination problem that we want to solve is stated more rigorously as follows:

> Given a set $S = \{s_1, s_2, \ldots, s_n\}$ of convex polygonal surfaces, their diffuse BRDF's $f_{r1}, f_{r2}, \ldots, f_{rn}$, and a convex polygonal light source s_0 of constant emission L_0, compute the set of radiance functions $L_{10}, L_{20}, \ldots, L_{n0}$

foreach visual event v_k involving s_0 **do**
    trace v_k through S and insert resulting critical curves in the meshes
end for
for $s_i \in S$ **do**
    triangulate mesh for s_i
    foreach element t_k in mesh for s_i **do**
        compute L_{i0} at sample locations within t_k
        adaptively subdivide t_k as needed
    end for
end for

Figure 4.1: Pseudocode for the single source discontinuity meshing algorithm

such that

$$L_{i0}(x) = f_{ri} L_0 \int_{x' \in s_0} \frac{\cos\theta_i \cos\theta_0}{r^2} v(x, x') dA(x') \tag{4.1}$$

for all $i = 1, 2, \ldots, n$.

Of course, solving Equation 4.1 for every point x on surface s_i is not feasible. Instead, we use a sampling approach whereby the radiance function L_i0 is evaluated at a few selected locations and then reconstructed using a suitable interpolant to yield an approximation $\tilde{L}_{i0}$.

In computer graphics, this kind of approach has roots that go back to the earliest shading algorithms such as Gouraud shading [GOUR71]. The originality of the algorithm described here, though, rests in its ability to predict the location and order of the radiance discontinuities, and use these results to guide the choice of sample locations and reconstruction interpolant.

Pseudocode for the single source discontinuity meshing algorithm is given in Figure 4.1. The algorithm starts by tracing all the visual events involving source s_0 through the environment S. Whenever a visual event results in a critical curve on a surface s_i, one or more edges are created in the discontinuity mesh for s_i. These edges store the location of the critical curve as well as the order of discontinuity associated with it.

Once all the visual events involving source s_0 are processed, each surface in the environment has been discretized into a discontinuity mesh where each important discontinuity in the radiance function and its derivatives is explicitly represented by one or more edges.

At this point, the mesh is triangulated and the direct illumination due to source s_0 is computed at a set of selected locations within each mesh element. Once sample values are computed over an element, an error estimate is computed for that element, and, if necessary, the element is adaptively subdivided to yield a more accurate approximation to L_{i0}. Since no discontinuity is allowed to cross an element, though, the radiance function inside each element is well behaved and can in most cases be captured by a relatively small number of samples.

Finally, a piecewise quadratic interpolant, $\tilde{L}_{i0}$, is used to reconstruct an approximation to the actual radiance function L_{i0}. The interpolation scheme provides smooth behavior inside mesh elements while still preserving the appropriate order of discontinuity across each mesh edge.

The following sections describe each of the steps outlined above in greater detail and present the data structures and techniques used for an efficient implementation of the algorithm.

4.2 Locating the Discontinuities

In Section 2.4, we showed that the direct illumination due to a single area light source can cause value (zero) and first and second derivative discontinuities in the radiance functions of the receiving surfaces.

This section describes the techniques used to locate these discontinuities in a polygonal environment with no interpenetrating surfaces.

4.2.1 Locating D^0 Discontinuities

Value, or D^0, discontinuities are responsible for the highest illumination contrast. Failure to capture these discontinuities by global illumination simulations has resulted in the shadow/light leaks and the floating objects that are evident in so many radiosity images (see Figures 3.1 and 3.2.)

D^0 discontinuities are due to touching surfaces and point light sources. Point light sources can be handled similarly to area light sources and their implementation will not be discussed here.

When surfaces touch, one surface may act as an occluder preventing light from the source from reaching part of the other surface, or receiver. If the occluder lies on the receiver, i.e. the two surfaces are coplanar, then the D^0 discontinuities will occur along the perimeter of the occluder; otherwise, the occluder may have either an edge or a vertex lying on the receiver, resulting in a D^0 discontinuity occurring along a line segment or at a single point respectively.

The location of these discontinuities is then independent of the position of the light sources and can be computed in a separate preprocessing step.

```
class Surface has
    Plane plane
    Polygon polygon
    list of Segment D⁰edges
    list of Point D⁰vertices
    method ComputeD⁰(in Segment)
end class

class Node has
    BoundingBox box
    list of Node children
    list of Surface surfaces
    method ComputeD⁰(in Segment)
end class
```

Figure 4.2: Pseudocode for the computation of D^0 discontinuities (Part 1)

Implementation

Given the set $S = \{s_1, s_2, \ldots, s_n\}$ of input surfaces, we could easily find all D^0 discontinuities by checking all possible pairs of surfaces, $(s_i, s_j), i \neq j$. Assuming that the number of edges per polygonal surface is bounded, this strategy would result in $O(n^2)$ time complexity.

A more efficient algorithm can be designed by organizing the input surfaces into a hierarchy of bounding volumes, a popular scheme used by many ray tracing systems [WHIT80].

Pseudocode for the algorithm is given in Figures 4.2 and 4.3. An object of type *Surface* is used to represent each surface s_i in the environment; the object provides access to the polygonal representation of the surface as well as its plane equation; also, the object is used to store a list of the D^0 discontinuities falling on its associated surface.

The environment is organized into a hierarchy of bounding volumes. A popular technique for constructing such a hierarchy was described by Goldsmith and Salmon [GOLD87]. Class *Node* is used to represent the nodes of the hierarchy. Each node has its bounding box, a list of its children, and, if it is a leaf node, a list of the surfaces it contains. The hierarchy of bounding volumes is accessed through the special node *root*.

The algorithm checks each edge in the environment against the hierarchy of bounding volumes, creating D^0 segments and points as it progresses.

Given an edge e, the algorithm traverses the hierarchy of bounding volumes visiting only the nodes whose bounding boxes contain at least part of the edge. If a leaf node is reached, then all the surfaces it contains are checked against e.

When an edge e is checked against a surface s, two cases may arise. If the edge lies in the supporting plane of the surface, we compute the part, if any, of e contained within the boundaries of s and insert it into the list of D^0 discontinuity segments falling on the surface. If, instead, the edge intersects the surface, we insert the point of intersection into the list of D^0 discontinuity points falling on the surface.

The performance of this algorithm is strictly dependent on the quality of the hierarchy of bounding boxes used. In the degenerate case, the hierarchy reduces to a linear list, giving a worst time complexity of $O(n^2)$. In general, though, for a relatively balanced hierarchy, we can expect a $O(n \log n)$ performance.

```
foreach edge e in the environment do
    root.ComputeD⁰(e)
end for

method Node::ComputeD⁰(in Segment e) is
    if e intersects box then
        foreach Surface s in surfaces do
            s.ComputeD⁰(e)
        end for
        foreach Node nd in children do
            nd.ComputeD⁰(e)
        end for
    end if
end method

method Surface::ComputeD⁰(in Segment e) is
    if e lies on plane then
        Segment e′ ← intersection of e with polygon
        if e′ ≠ NIL then
            add e′ to D⁰edges
        end if
    else
        Point p ← intersection of e with plane
        if p lies inside or on the boundary of polygon then
            add p to D⁰vertices
        end if
    end if
end method
```

Figure 4.3: Pseudocode for the computation of D^0 discontinuities (Part 2)

4.2.2 Locating D^1 and D^2 Discontinuities

Discontinuities in the first and second derivative of the radiance function, referred to as D^1 and D^2 discontinuities respectively, occur along the umbra and penumbra boundaries cast by finite area light sources as well as along other critical curves inside penumbra regions.

Conventional radiosity algorithms ignore these discontinuities (see Chapter 3) and yield images with jagged shadow boundaries and missing or distorted shadows such as those presented in Figures 3.1 and 3.2. The ability to determine the location of these discontinuities can significantly improve the accuracy of radiosity simulations.

Choosing which discontinuities to represent

The D^1 and D^2 discontinuities are generated by the visual events discussed in Section 2.4. There are two different types of visual events: EV and EEE events. In a polygonal environment with m edges, these events can result respectively in $O(m^3)$ and $O(m^4)$ critical curves. Trying to compute all the D^1 and D^2 discontinuities would result in time and storage costs so high as to limit the system to anything but the simplest environments.

Not all visual events, though, result in discontinuities of the same visual importance. A careful selection may result in an algorithm of lower complexity capable of accepting most simulated environments.

As a first simplification, we decided to ignore EEE events. These events can result in discontinuities along curved segments. Capturing these discontinuities would require a discontinuity mesh capable of representing curved edges and non-polygonal elements. Also, a reconstruction method capable of accepting curved boundary elements would be needed. Fortunately, the boundary between penumbra and unoccluded regions is marked by EV events only; thus, ignoring EEE events does not affect the main shape of the shadows.

As a further simplification, we considered only the EV events to which the light source contributes either the participating edge or participating vertex. Other EV events are only significant if the critical surfaces they define intersect the light source. These events are relatively few, but considering them would add an order of magnitude to the cost complexity of the algorithm.

The number of remaining EV events is only $O(m)$ and the corresponding critical curves are straight line segments for polygonal environments. In particular, these segments include all the boundaries separating totally unoccluded regions from partially occluded ones, as well as additional important discontinuities inside the penumbrae.

The set of EV events that we considered is not as limited as it may appear. In fact, it is a superset of the visual events considered by other shadowing and discontinuity meshing algorithms [CAMP91c,CHIN92,NISH83].

Classifying EV visual events

The order of the discontinuity associated with a critical curve is determined by the rate of change in the amount of light source area that is visible from the receiver when moving across the critical curve (see Figures 2.6 and 2.7.) If the rate of change is linear, the discontinuity is D^1; otherwise, it is D^2. If we consider the visual event and critical surface that generated the critical curve, we will notice that the above rate of change is the same everywhere across the critical surface;

this means that all critical curves corresponding to a given visual event share the same order of discontinuity.

Determining the order of discontinuity associated with an EV visual event is simple. Let V and E be respectively the vertex and edge defining the visual event. If v is the endpoint of an edge E' that is coplanar with E, the event should be classified as D^1; otherwise, it should be classified as D^2.

It is convenient to distinguish between the EV events involving a source vertex and the ones involving a source edge. In what follows, we will refer to the former kind as *VE events* and to the latter as *EV events*.

Computing critical curves for VE visual events

A VE visual event captures the interaction of a vertex V of the light source and an edge E of an occluding surface. The critical surface corresponding to the event is a planar wedge W of infinite extent with its apex at vertex V and bounded by rays through the endpoints of the edge E (see Figure 4.4(a).)

As shown in Section 2.4, discontinuities in the derivatives of the radiance function happen along critical curves corresponding to the intersection of a critical surface with the surfaces in the environment. The discontinuities are caused by abrupt changes in the visibility of the light source. Not all intersections, however, are critical curves. Only the intersections that are visible from the apex V of W, i.e. the critical surface, are actual critical curves. If fact, if an intersection segment is not visible from V, then no change in the visibility of the light source, and therefore no discontinuity, may occur along the segment.

Computing the set of critical curves corresponding to a given VE event is then similar to a visible surface determination problem in 2D (see Figure 4.4(b).) The

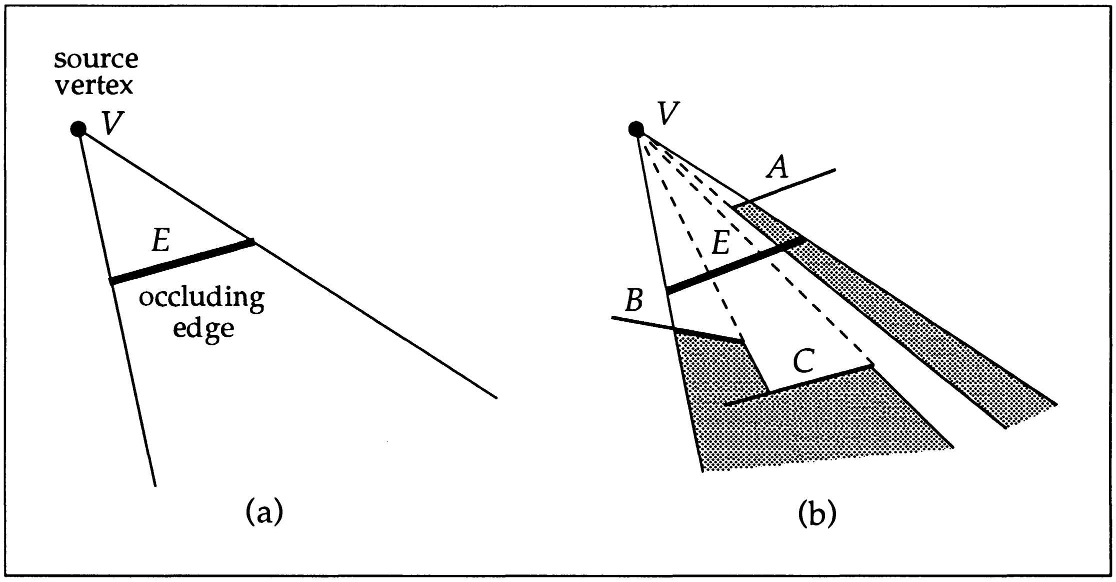


Figure 4.4: (a) A wedge corresponding to a VE event. (b) Interaction of the wedge with the surfaces in the environment. The wedge is clipped against surfaces A, B, and C, but only surfaces B and C gain a new discontinuity boundary. Discontinuities are not added to surfaces intersecting the wedge in the region enclosed between V and E, since the visual event is not visible from such surfaces. The dark areas show the clipped portion of the wedge.

surfaces in the environment are sorted in order of increasing distance from V. If a surface is intersected by W along a visible segment and it faces V, then the segment of intersection is inserted in the surface mesh as a discontinuity edge. If, however, the surface is closer to V than the occluding edge E is, then the surface is not effected by the visual event and no critical curves should be generated.

Computing critical curves for EV visual events

An EV visual event captures the interaction of an edge E of the light source and a vertex V of an occluding surface. The critical surface corresponding to this event is slightly more complex than that of a VE event; the surface is bounded by edge E and by semi-infinite rays originating at the endpoints of the edge and passing through vertex V (see Figure 4.5(a).)

Computing the set of critical curves corresponding to a given EV event is done similarly to the tracing of VE events except that the surfaces in the environment are sorted in order of increasing distance away from E rather than V (see Figure 4.5(b).) If a surface is intersected by the event along a segment that is visible from E and it faces V, the the segment is inserted in the surface mesh as a discontinuity edge. If, however, the intersected surface lies between E and V, then no discontinuity may result.

Implementation

The pseudocode given in Figure 4.6 shows how the selected visual events discussed above are generated and classified with the appropriate order of discontinuity.

Three sets of visual events are generated. First, VE events are generated from

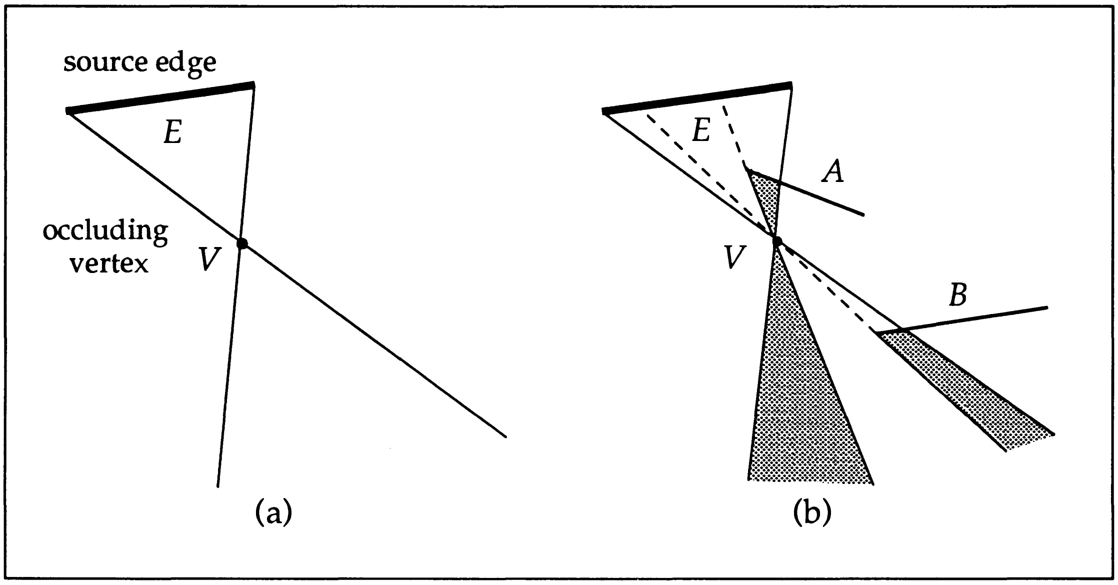


Figure 4.5: (a) A wedge corresponding to an EV event. (b) Interaction of the wedge with the surfaces in the environment. The wedge is clipped against surfaces A and B, but only surface B gains a new discontinuity boundary. Discontinuity boundaries are not added to surfaces intersecting the wedge in the region enclosed between E and V, since the visual event is not visible from such surfaces. The dark areas show the clipped portion of the wedge.

```
class VEevent has
    Point V
    Segment E
    DiscOrder order
    method Set(in Point, in Segment, in DiscOrder)
end class

class EVevent has
    same as above
end class

Point C ← centroid of source s0
foreach edge E of source s0 do
    VEevent ve.Set(C, E, D^2)
    model.TraceVE(ve)
end for
foreach vertex V of source s0 do
    foreach edge E in set of potentially occluding edges do
        if s0 has an edge E' incident on V and coplanar with E then
            VEevent ve.Set(V, E, D^1)
        else
            VEevent ve.Set(V, E, D^2)
        end if
        model.TraceVE(ve)
    end for
end for
foreach edge E of source s0 do
    foreach vertex V in set of potentially occluding vertices do
        if V is the endpoint of an edge E' that is coplanar with E then
            EVevent ev.Set(E, V, D^1)
        else
            EVevent ev.Set(E, V, D^2)
        end if
        model.TraceEV(ev)
    end for
end for
```

Figure 4.6: Pseudocode for the generation and classification of visual events

the centroid of the light source through each of its edges. These events are used to locate the discontinuity in the second derivative of the radiance function on those surfaces in the environment that are intersected by the supporting plane of the light source. In fact, the contribution of the light source is maximum in the direction normal to the source, decreases away from the normal, reaches zero for points coplanar to the source, and is of course zero for any point behind the source.

Second, we trace VE events from each of the vertices of the light source. For each vertex V, there are as many possible visual events as there are edges in the environment. Only a subset of these edges, however, has an impact on our simulation. The set of potentially occluding edges contains only edges that have at least one endpoint in front of the light source. Edges that are shared between two or more polygons are counted only once. If the endpoints of an edge lie on opposite sides of the light source, the edge is clipped and only the part lying in front of the source is considered.

If the environment is made of closed non-interpenetrating polyhedra, this set can be made smaller by including only edges incident to at least one polygon facing the apex V. In fact, a visual event whose participating edge E' did not belong to any polygon facing V would only see polygons that lie between V and E', and would therefore be unable to generate any critical curves (see example in Figure 4.7.)

If the environment contained only convex polyhedra, the set of potentially occluding edges could be made even smaller by including only those edges that appear as silhouette edges when seen from V. The detection of silhouette edges,

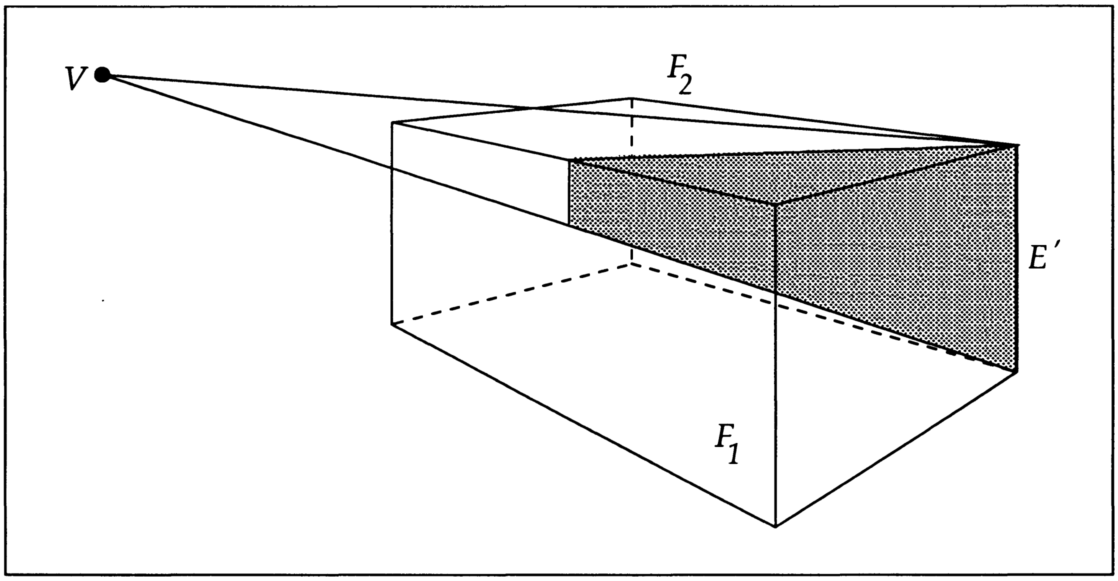


Figure 4.7: Visual event defined by an edge E' whose incident faces are all facing away from V. In a polyhedral environment, this event is inconsequential since it is completely arrested by the faces, F_1 and F_2, intervening between V and E'.

however, was not implemented.

Determining the order of discontinuity carried by these VE events is simple. We take the two edges of the light source, E' and E'', that are incident on V; if either edge lies in the plane defined by V and E, then the visual event will create D^1 discontinuities; otherwise, it will create D^2 discontinuities.

Finally, we trace EV events from each of the edges of the light source. Similarly to VE events, we compute a set of potentially occluding vertices by including only those vertices in the environment that lie in front of the light source and belong to a polygon facing the light source. Vertices sharing the same location are inserted only once.

In order to determine the order of discontinuity of an EV event, each occluding vertex V stores a list of its incident edges. If any of these edges is coplanar with the edge of the light source generating the EV event, then the event is classified as D^1; otherwise, the event is D^2.

As mentioned before, computing the critical curves associated to a VE visual event is essentially a visible surface determination algorithm. We order to implement this process efficiently, we organize the surfaces in the environment in a BSP-tree [FUCH80]. This data structure, in fact, provides a simple and fast way of sorting its surfaces by distance with respect to any point in space.

The pseudocode given in Figures 4.8 to 4.10 shows how to compute the critical curves (or segments) generated by a VE event *ve*.

The environment is stored as a BSP-tree. Each node stores a plane equation *plane* and a list of the surfaces lying on that plane, as well as subtrees to the part of the environment in front of the plane ($child_{\text{out}}$) and that behind the plane ($child_{\text{in}}$).

Given a VE event *ve*, the BSP-tree is traversed in front-to-back order along the wedge defined by the participating vertex and edge of *ve*, starting from its apex *ve.V* (see Figure 4.9.) If the wedge faces the plane of a BSP-tree node, then the wedge is clipped against the surfaces associated with that node (see Figure 4.4.) Only the unclipped parts of the wedge are traced further. The traversal can stop as soon as all portions of the wedge have been clipped. In order to keep track of which parts of a VE event remain to be traced, the data structure for the event stores a list of active intervals. At first, this list contains a single interval, representing the entire wedge. When tracing the event through the environment, however, portions of the wedge may be clipped and old intervals may be dropped from the list of active intervals or replaced by new smaller ones. Should the list of active intervals become empty, the recursive traversal of the BSP-tree

class *VEevent* **has**
    . . .
    list of *Interval intervals*
end class

class *Surface* **has**
    Plane plane
    Polygon polygon
    . . .
    method *ClipVE*(**inout** *VEevent*)
end class

class *BSPnode* **has**
    Plane plane
    list of *Surface surfaces*
    BSPnode $child_{out}$, $child_{in}$
    method *TraceVE*(**inout** *VEevent*)
    method *ClipVE*(**inout** *VEevent*)
end class

Figure 4.8: Pseudocode for the location of the discontinuities generated by a VE event (Part 1)

```
method BSPnode::TraceVE(inout VEevent ve) is
    if ve.intervals = NIL then
        return
    end if
    if ve.V is in front of plane then
        child_out.TraceVE(ve)
        if ve faces plane then
            ClipVE(ve)
            child_in.TraceVE(ve)
        end if
    else if ve.V is behind plane then
        child_in.TraceVE(ve)
        if ve faces plane then
            ClipVE(ve)
            child_out.TraceVE(ve)
        end if
    else
        if ve.E lies on plane or either endpoint of ve.E is in front of plane then
            child_out.TraceVE(ve)
        end if
        if either endpoint of ve.E is behind plane then
            child_in.TraceVE(ve)
        end if
    end if
end method

method BSPnode::ClipVE(inout VEevent ve) is
    foreach Surface s in surfaces do
        s.ClipVE(ve)
        if ve.intervals = NIL then
            break
        end if
    end for
end method
```

Figure 4.9: Pseudocode for the location of the discontinuities generated by a VE event (Part 2)

```
method Surface::ClipVE(inout VEevent ve) is
    list of Interval intervals' ← NIL
    foreach Interval I in ve.intervals do
        clip I against polygon to get I1, I2, I3
        if I1 ≠ NIL then
            intervals' ← intervals' + {I1}
        end if
        if plane faces ve.V and I2 ≠ NIL and I2 is not between ve.V and ve.E then
            add I2 to surface's discontinuity mesh
        end if
        if I3 ≠ NIL then
            intervals' ← intervals' + {I3}
        end if
    end for
    ve.intervals ← intervals'
end method
```

Figure 4.10: Pseudocode for the location of the discontinuities generated by a VE event (Part 3)

may be safely interrupted.

The pseudocode given in Figure 4.10 shows in more detail how a VE event *ve* can be clipped against a surface in the environment. Each active interval I of *ve* is examined in turn. First, I is intersected with the surface polygon. Since the polygons in a BSP-tree are guaranteed to be non-concave, I can be split at the most into three subintervals. I_1 and I_3, if non-empty, fall outside the polygon and are therefore the new active intervals; I_2, if it exists, lies inside the polygon and is a candidate for insertion in the discontinuity mesh of the surface. As each interval I is examined, a new list of active segments, *intervals'*, is created which will replace the old list in *ve.intervals* at the end of the process.

A number of speedups were used to obtain an efficient implementation. For

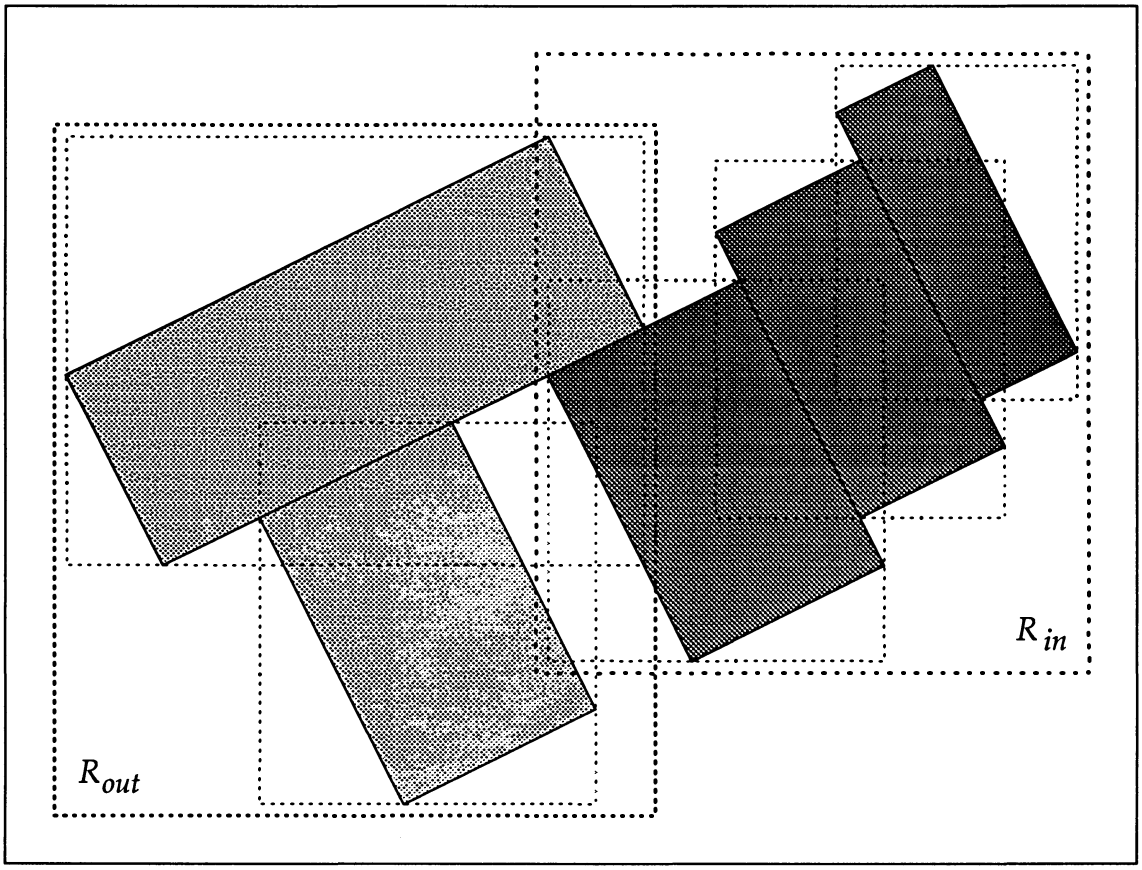


Figure 4.11: Coplanar surfaces in a BSP-tree node are grouped in two sets, according to their orientation (light surfaces face outwards, dark surfaces face inwards.) Each set is further organized in a two-level hierarchy of bounding rectangles.

clarity, however, these details were omitted from the pseudocode. One of these efficiency schemes is nonetheless worth mentioning.

The coplanar surfaces stored with a BSP-tree node are grouped into two sets: one containing all the surfaces that face towards the outside of the supporting plane of the node and one containing all the surfaces that face towards its inside (think of the plane as a halfspace.) Each set of surfaces is further organized in a two-level hierarchy of bounding rectangles (see Figure 4.11.) At the lower level, each surface is bounded by its own rectangle; at the higher level, a single rectangle bounds all the surfaces. All of these rectangles are aligned to the axis of a two-dimensional coordinate system local to the plane of the BSP-tree node.

Before any of the active intervals of a VE event are examined, the entire wedge representing the event is intersected with the plane of the BSP-tree node. If the intersection segment falls completely outside a higher level bounding rectangle, then its contents are skipped. Otherwise, the intersection segment is further checked against the lower level bounding rectangles before the active intervals of the visual event can be clipped against the surfaces.

This simple scheme is very effective due to the following reasons:

1. Large groups of coplanar surfaces may be distant from the path of the visual event and can be culled efficiently using the higher level bounding rectangles.

2. Polygons in the BSP-tree may have many vertices and be therefore costly to intersect. Using the lower level bounding rectangles can avoid needless computations.

3. Visual events can be fragmented into many intervals. Checking the event first in its entirety may quickly cull out many needless intersection tests.

The implementation of EV events is not too different from that of VE events. An EV event *ev* can be looked at as two VE wedges sharing the same apex V (see Figure 4.5.) First, the BSP-tree is traversed in front-to-back order from V in the direction of E; then, once E is reached, a new front-to-back traversal is started from V in the direction opposite to E. The list of active intervals of *ev* is modified by the surfaces intersected during the first traversal, but no critical curves are generated. The same list is then used as the initial list of active intervals for the second traversal. This time, any visible intersections with polygons facing V will result in new discontinuity segments being added to the appropriate discontinuity meshes.

4.3 Constructing the Discontinuity Mesh

The critical curves computed in the previous section are used to construct a discontinuity mesh capable of capturing all the important discontinuities in the radiance function and its derivatives. Each surface maintains its own mesh. This discontinuity mesh is represented by a data structure called a *Discontinuity Mesh Tree*, or *DM-tree*.

A DM-tree consists of two parts: a two-dimensional BSP-tree and a data structure of topologically interconnected faces (elements), edges, and vertices. Each inner node of the tree contains the line equation of a discontinuity segment. Each leaf contains a pointer to a face bounded by the intersection of the halfspaces encountered along the path down from the root of the tree. This data structure is constructed incrementally: each discontinuity segment is filtered down the tree, eventually splitting only the intersected leaves. The faces attached to these leaves are also split, and the newly added edges are labeled with the appropriate order of discontinuity.

The topological mesh is arranged similarly to a winged-edge data structure so that no duplication of edges and vertices is necessary and each element can easily determine its neighbors. In this way the contribution of the source can be computed once on each vertex and shared between the incident elements. Since edges are shared between neighboring elements, no T-vertices are introduced when elements are split.

Two-dimensional BSP-trees were also used to represent radiosity meshes by Campbell and Fussell [CAMP90]. Note however, that the elements stored at the leaves of their *element BSP-tree* were not topologically connected. This results in redundant illumination computations, as well as in a need for a separate pass to eliminate T-vertices [CAMP91c].

Figure 4.12 shows an example of how a DM-tree is constructed incrementally from a sequence of visual events. The simple configuration given in (a) shows the discontinuity lines on a receiving surface R, generated by the interaction of vertex V of source S with obstacle O. Initially, (b) the DM-tree of the receiver consists of a single node pointing to a single face F. When the first edge of the obstacle is projected on the receiver, (c) the line equation a of the resulting discontinuity line is used to split face F into faces F_1 and F_2, and the DM-tree is updated so that its root now stores line equation a and points to the two new faces on either side of a. Notice that two collinear edges have been created along line a: one corresponds to the actual discontinuity segment, while the other is due to our splitting scheme and is not marked as a discontinuity. Next, the second VE event is processed (d). A new discontinuity segment is generated and the DM-tree is searched from the root to determine whether any face is crossed by this segment. Face F_2 is found and is therefore split into faces F_{21} and F_{22}. Finally, the third VE event is processed to yield the DM-tree depicted in (e).

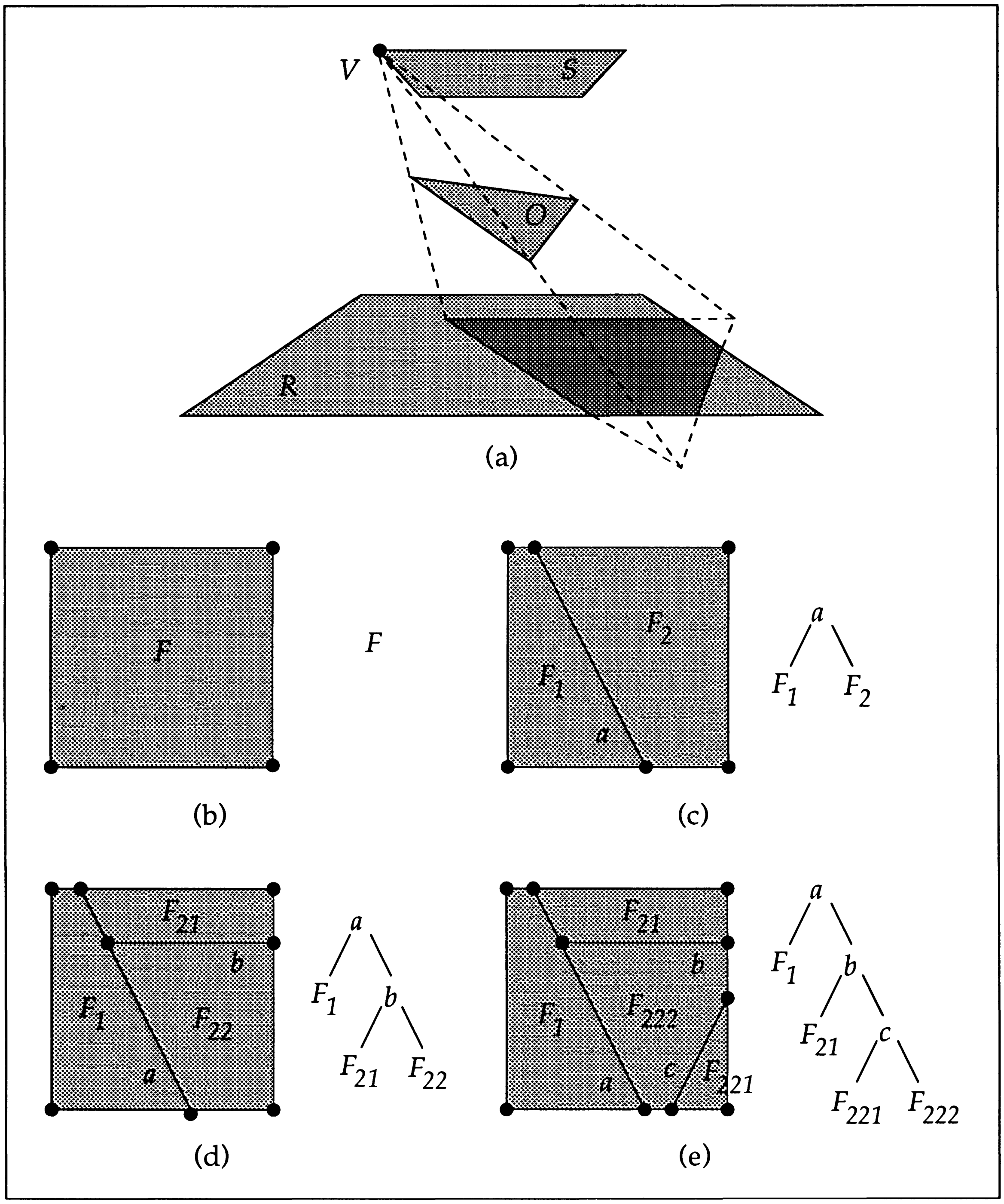


Figure 4.12: Incremental construction of a DM-tree

4.3.1 Implementation

The pseudocode given in Figure 4.13 shows the data structures used to implement a DM-tree.

The hierarchical portion of the DM-tree is made of *DMnode* nodes. If the node is a leaf, then *face* points to the face in the topological mesh corresponding to that node. Otherwise, the node partitions the surface into two halfspaces; variable *line* stores the equation of the dividing line, and $child_{\text{out}}$ and $child_{\text{in}}$ point to the subtrees of mesh elements that are respectively to the outside and inside of *line*.

The topological part of the mesh is represented by a winged-edge data structure. Each face points to a doubly linked ring of edges. Each edge has pointers to its two endpoints, neighboring edges, and incident faces (possibly NIL.) The boundary of *face*[*i*] can be traversed either clockwise or counter-clockwise by following the chain of *prev*[*i*] or *next*[*i*] pointers respectively. Each vertex has its coordinates and a pointer back to one to its incident edges. The order of discontinuity of an edge or a vertex is stored in variable *order*.

To start with, each surface has a discontinuity mesh consisting of a single node and a single face. This mesh is then refined incrementally by inserting the critical curves and their associated order of discontinuity. The D^0 discontinuity segments computed in Section 4.2.1 are inserted first. Then, D^1 and D^2 discontinuity segments are computed as shown in Section 4.2.2 and inserted in the mesh on the fly.

The pseudocode given in Figure 4.14 shows how a discontinuity segment is inserted in a discontinuity mesh by filtering it down starting at the root of the

```
class Surface has
    ...
    DMnode mesh
    method InsertSegment(in Segment, in DiscOrder)
end class

class DMnode has
    pointer to Face face
    Line line
    DMnode child_out, child_in
    method InsertSegment(in Segment, in DiscOrder)
end class

class Face has
    pointer to Edge boundary
    method AdjustBoundary(in Segment, in DiscOrder)
    method Split(in Segment, in DiscOrder, out Face, out Face)
end class

class Edge has
    DiscOrder order
    pointer to Vertex endpoint[2]
    pointer to Edge next[2], prev[2]
    pointer to Face face[2]
end class

class Vertex has
    Point coord
    DiscOrder order
    pointer to Edge edge
end class
```

Figure 4.13: Pseudocode for the DM-tree data structures

```
method Surface::InsertSegment(in Segment seg, in DiscOrder ord) is
    mesh.InsertSegment(seg, ord)
end method

method DMnode::InsertSegment(in Segment seg, in DiscOrder ord) is
    if this is an interior node then
        if seg crosses line then
            split seg across line to yield seg_out and seg_in
            child_out.InsertSegment(seg_out, ord)
            child_in.InsertSegment(seg_in, ord)
        else if seg lies on the outside of line then
            child_out.InsertSegment(seg, ord)
        else
            child_in.InsertSegment(seg, ord)
        end if
    else
        if seg falls on the boundary of face then
            face.AdjustBoundary(seg, ord)
        else
            line ← supporting line of seg
            child_out ← create new leaf node
            child_in ← create new leaf node
            face.Split(seg, ord, child_out.face, child_in.face)
            dispose of face and mark this node as non-leaf
        end if
    end if
end method
```

Figure 4.14: Pseudocode for the insertion of a critical curve into the discontinuity mesh of a surface

DM-tree. Notice that the input segment is generated by the algorithms given in the previous section and is guaranteed to lie completely inside (possibly on the boundary) the surface being meshed.

If the node being visited is an interior node, we compare the discontinuity segment against the line equation stored at that node. If the segment crosses the line, we split it across the line and attempt to insert each of the resulting pieces independently by recursively searching the $child_{\text{out}}$ and $child_{\text{out}}$ subtrees respectively. If the segment falls completely on one side of the dividing line, then we continue the search on that side only. If the segment lies on the dividing line, then we continue the search down the hierarchy by arbitrarily choosing to follow the $child_{\text{in}}$ subtree.

When a discontinuity segment reaches a leaf node, it must be inserted into the topological part of the mesh. The discontinuity segment must either fall on the boundary of the face associated with the leaf node, or be completely contained within the face. The pseudocode given in Figure 4.15 describes these two cases in detail.

In the former case, the endpoints of the segment are added to the boundary of the face possibly splitting one or two edges. The edges covered by the segment are then reclassified according to the order of discontinuity of the segment. Notice that if two discontinuities overlap, the one with the lower order, i.e. the sharper one, will prevail.

In the latter case, the face is split along the supporting line of the discontinuity segment resulting in a new edge E (see parts a and b of Figure 4.16.) The edge is initially classified as D^{∞}. Then, E is split in up to three parts at the endpoints of

method *Face::AdjustBoundary*(**in** *Segment seg*, **in** *DiscOrder ord*) **is**
  foreach endpoint V of *seg* **do**
    if V lies on an edge E of *boundary* **then**
      split E and create a new vertex corresponding to V
    end if
  end for
  foreach *Edge* E covered by *seg* **do**
    if $ord < E.order$ **then**
      $E.order \leftarrow ord$
    end if
  end for
end method

method *Face::Split*(**in** *Segment seg*, **in** *DiscOrder ord*, **out** *Face* F_{out}, **out** *Face* F_{in}) **is**
  split face along *seg* to create new faces F_{out} and F_{in} joined by edge E
  $E.order \leftarrow D^{\infty}$
  clip E against *seg* to get E_1, E_2, E_3 (E_1 and E_3 may be NIL)
  $E_2.order \leftarrow ord$
end method

Figure 4.15: Pseudocode for the insertion of a discontinuity segments into the leaves of a DM-tree

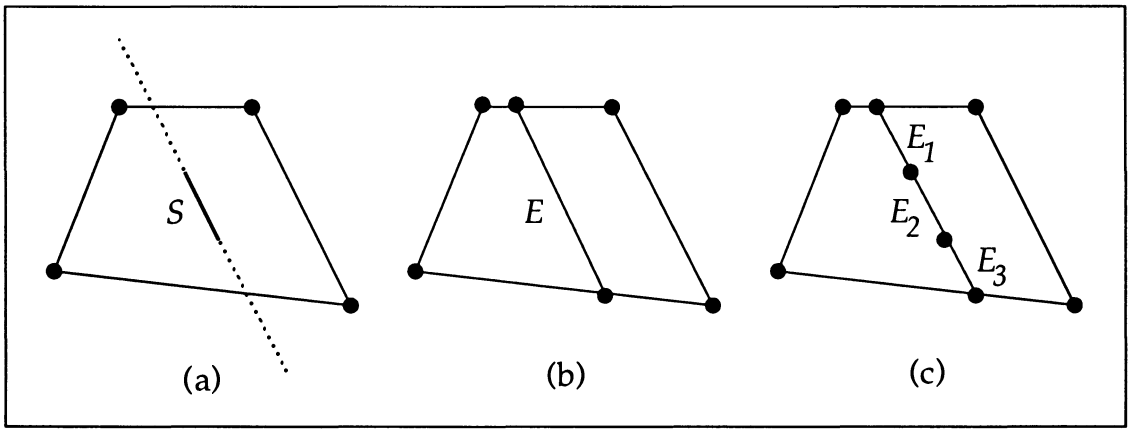


Figure 4.16: Splitting a face to capture a discontinuity segment. (a) The initial face and discontinuity segment S. (b) The face is split along the supporting line of S and a new edge E is created. (c) E is split into three parts, with edge E_2 corresponding to the discontinuity segment S.

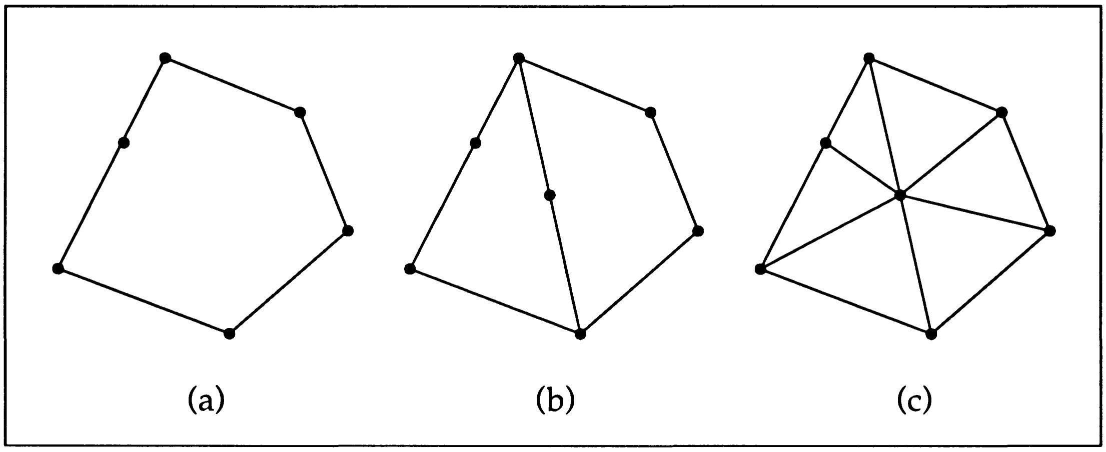


Figure 4.17: Triangulating a mesh element. (a) The initial element. (b) The new vertex is added halfway between the two most distant vertices. (c) New edges are created between each vertex on the element boundary and the new vertex.

the segment and the part that is covered by the segment is reclassified with the order of discontinuity of the segment (Figure 4.16c.)

4.4 Sampling and Reconstruction on the Receiver

Once the discontinuity mesh is built, sample locations must be chosen at which to compute the radiance contribution of the current light source. The contributed radiance function is then reconstructed from the samples.

The discontinuity mesh built using the techniques described in the previous sections partitions a surface into regions within which the radiance function is continuous. Because of the binary space partitioning scheme used, the resulting mesh elements are convex polygons, but may have many vertices. Reconstructing a smooth radiance function across such general elements while meeting the boundary conditions is difficult. Therefore, each region is triangulated first. The triangulation algorithm avoids creating new vertices on the boundaries of the mesh elements so that each element can be triangulated independently. The algorithm is illustrated in Figure 4.17 by way of a simple example. We start with a convex element with possibly collinear vertices (a). First, we choose the pair of vertices V_i and V_j that are the farthest apart and create a new vertex halfway between V_i and V_j (b). And then, we triangulate the element by creating new edges between each vertex on the element boundary and the new vertex (c).

Triangular elements of different orders can now be used to represent the radiance function. Constant and linear elements are only able to resolve D^0 and D^1 discontinuities respectively and generate annoying visual artifacts such as blockiness and Mach bands. Higher order elements can correctly resolve D^0, D^1, and D^2 boundary discontinuities, and can more accurately approximate the radiance function inside each element.

We use quadratic elements with each mesh triangle defined by six radiance values: one at each vertex and one at the midpoint of each edge (see Figure 4.19.) Rather than actually computing six values for each triangular element, radiance values are shared between neighboring elements across non-D^0 edges. Correct treatment of D^0 edges and vertices requires some attention. The radiance function is not uniquely defined along these edges and vertices and the radiance values in their vicinity depend on the side from which the edge is approached. The correct values for a given element are the limit values of the radiance function at the edge midpoints and vertices when moving towards them from inside the element. In practice, a good approximation to these limit values can be computed by slightly displacing the sample locations corresponding to the vertices and the midpoints of the edges inward, so that they lie strictly inside the element.

A special problem is posed by elements which are incident on a D^0 vertex V,

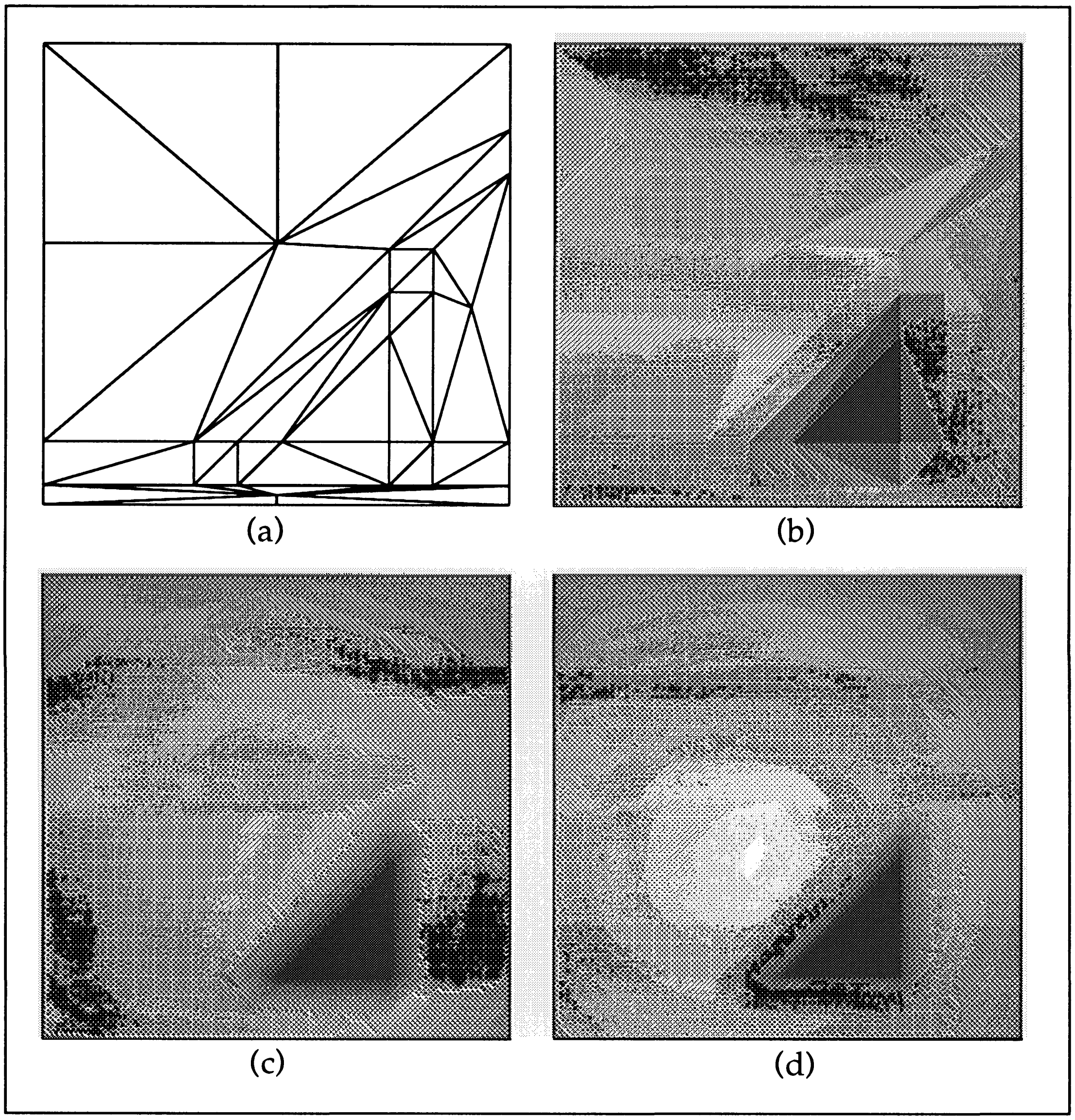


Figure 4.18: Radiance function reconstruction by piecewise polynomial interpolation. (a) The underlying discontinuity mesh. (b) Constant elements. (c) Linear elements. (d) Quadratic elements.

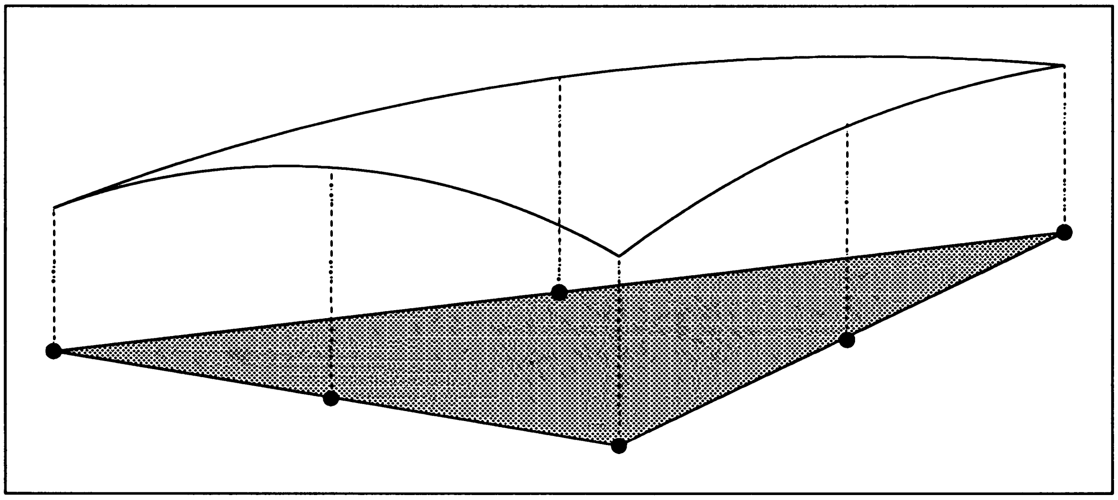


Figure 4.19: Quadratic interpolation over a triangular mesh element

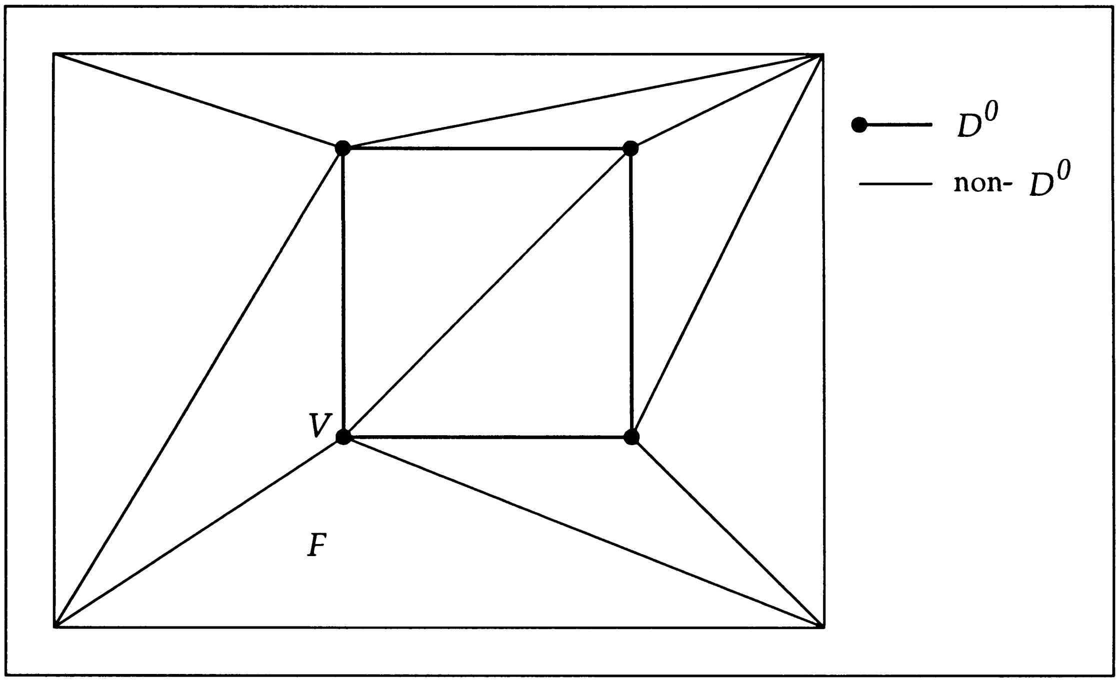


Figure 4.20: A simple discontinuity mesh with D^0 edges and D^0 vertices marked by thicker lines. Face F, among others, is incident on a D^0 vertex, V, but has no D^0 edges. The radiance function over F has a singularity at V.

but do not contain D^0 edges (e.g. element F incident on V in Figure 4.20.) There is a need to correctly represent the singularity that the radiance function may have at V. Assigning a single radiance value to V may cause D^0 discontinuities in the interpolated function along the edges incident on it. To solve this problem we fit a degenerate bi-quadratic patch over the triangular domain of the element. The degenerate patch has three control points corresponding to V and thus a range of values can be represented simultaneously there, while maintaining continuity along the incident edges.

Our implementation correctly resolves D^0 and D^1 discontinuities. The actual radiance function is C^1 continuous across D^2 boundaries. Currently, we do not impose C^1 continuity across these boundaries. It is possible to do so using the piecewise cubic interpolation scheme described by Salesin *et al.* [SALE92].

Certain operations, e.g. ray tracing, require the ability to quickly inquire for the radiance value at a given location x on a surface s_i. The pseudocode given in Figure 4.21 shows how this radiance value can be efficiently evaluated using the DM-tree data structure representing the discontinuity mesh for s_i.

First, the DM-tree is descended recursively, starting at the root, until the leaf node, or element, containing point x is located. Then, point x is transformed to the (u, v) parametric coordinates of this element. Finally, the quadratic patch representing the radiance function over the element is evaluated at (u, v) and the desired radiance value $L_i(x)$ is stored in variable L.

Assuming that the DM-tree is relatively balanced, this operation takes $O(\log n)$, where n is the number of elements in the discontinuity mesh of a single surface, s_i.

method *DMnode::EvaluateRadiance*(**in** *Point x*, *Radiance L*) is
  if this is a leaf **then**
    Real u, $v \leftarrow$ parametric coordinates of x in this face
    $L \leftarrow$ radiance value of quadratic patch at (u, v)
  else
    if x is on the out side of *line* **then**
      $child_{\text{out}}.EvaluateRadiance(x, L)$
    else
      $child_{\text{in}}.EvaluateRadiance(x, L)$
    end if
  end if
end method

Figure 4.21: Pseudocode for the evaluation of the radiance function at a point x on a surface

4.5 Computing the Source Contribution to a Point

The radiance contributed by a source s_0 of constant emission L_0 to a point x is given by Equation (4.1).

If no occluding surfaces intervene between x and s_0, the visibility term v is everywhere 1, and the area integral can be transformed into a countour integral using Stoke's theorem. This, in turn, can be evaluated analytically using the formula provided by Hottel and Sarofin [HOTT67] (see Figure 4.22a):

$$F_{dA_i(x) \rightarrow A_0} = \frac{1}{2\pi} \sum_{j=0}^{n-1} \frac{\cos^{-1}(R_j \cdot R_{j\oplus 1})}{\|R_j \times R_{j\oplus 1}\|} (R_j \times R_{j\oplus 1}) \cdot N_i \tag{4.2}$$

where

$F_{dA_i(x) \rightarrow A_0}$ is the form factor notation for the integral in Equation (4.1);

$\oplus$ represents addition modulo n;

n is the number of vertices of the polygonal source s_0;

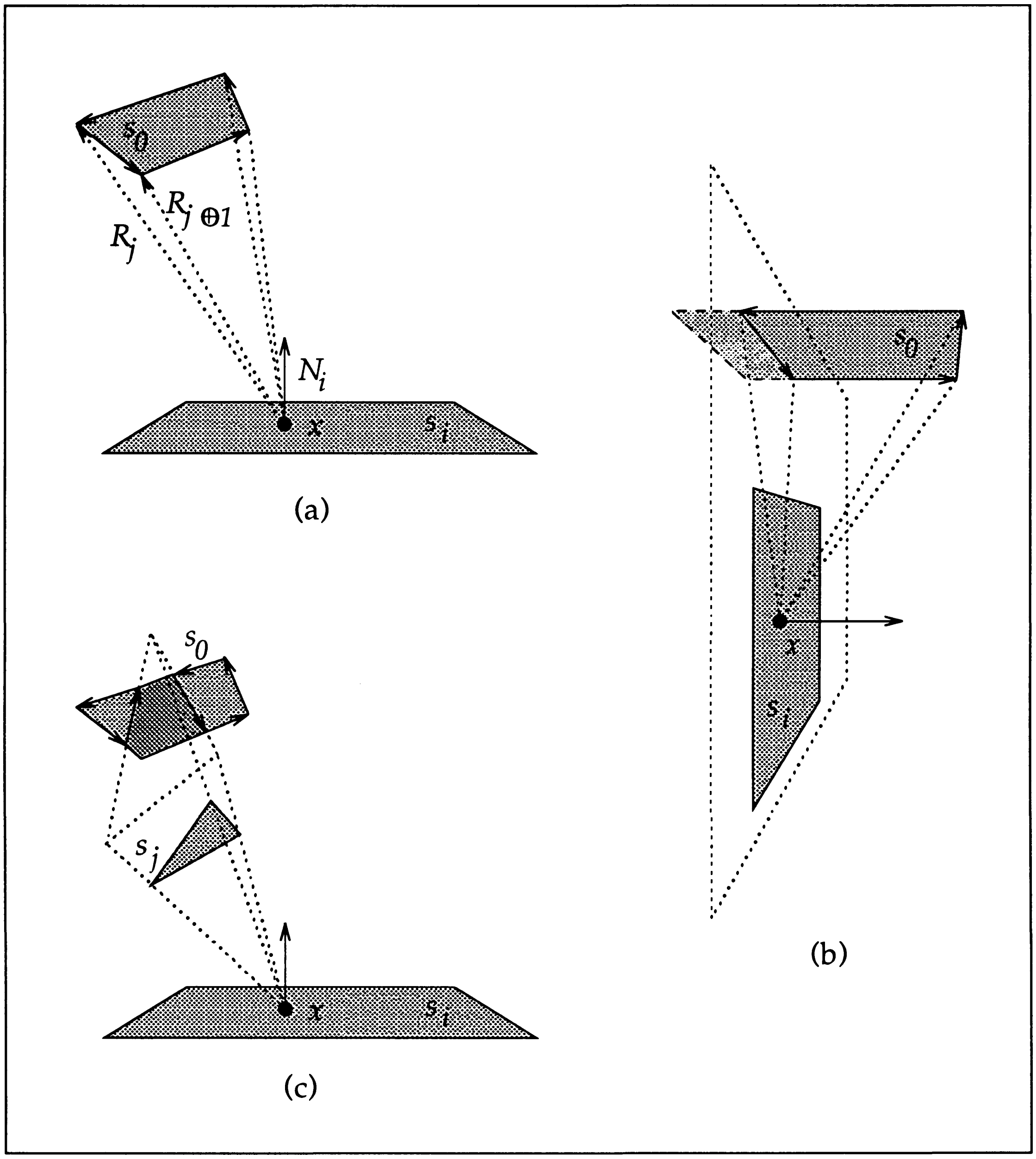


Figure 4.22: Analytical form factors. (a) Hottel and Sarofin's method for computing unoccluded form factors. (b) Only the part of the source lying in front of the receiver is considered in computing the energy transfer. (c) Only the visible parts of the source contribute to the form factor.

R_j is the unit vector from point x on surface s_i to the ith vertex of source s_0; and

N_i is the unit normal to surface s_i at point x.

As Nishita and Nakamae first pointed out [NISH83], this formula requires that the light source s_0 be completely in front of the receiving surface s_i. If, instead, the supporting plane of surface s_i intersects the source s_0, then the source must be split and the integration must be carried out on that part of s_0 that is in front of s_i only (see Figure 4.22b.)

The contribution of source s_0 to a point x can be computed analytically in the presence of occlusions, too. In fact, it is sufficient to determine which parts of s_0 are visible from x, and integrate the contribution of each part separately. To compute these visible regions, we project all the potential obstacles from the point of view of x onto the supporting plane of the source s_0 and discard any parts of the source that are covered by any of the projected obstacles (see Figure 4.22c.)

Point sampling algorithms for testing the visibility of the light source [COHE85, HANR91b,WALL89] can also be used, but they are not nearly as accurate, and are prone to aliasing.

4.5.1 Implementation

Figure 4.23 gives the pseudocode for the computation of the form factor from a point x on a surface s_i to a polygonal light source s_0.

Each occluding surface in the environment is projected onto the supporting plane of the source s_0 and its projection is intersected with the source to determine which parts of s_0 are visible from the receiving point x.

Considering all the surfaces in the environment as potential occluders would

```
compute shadow shaft for x and s_0
foreach Surface s_j in the environment do
    if s_j is not culled by the shadow shaft then
        project s_j onto supporting plane of s_0
        intersect projection with BSP-tree for s_0
    end if
end for
Real ff ← 0
foreach visible leaf polygon p in BSP tree for s_0 do
    ff ← ff + form factor F_{dA_i(x)→A_p}
end for
```

Figure 4.23: Pseudocode for the computation of form factors in the presence of occlusions

be terribly inefficient. Instead, we use a shaft culling technique similar to that proposed by Haines and Wallace [HAIN91c] to drastically reduce the number of potential occluders that must be projected onto the source s_0.

Our shadow shaft is a pyramid with the source s_0 as its base and the receiving point x as its apex. More formally, the shaft is defined as the intersection of halfspaces: one halfspace for each of the edges of the source s_0, defined as the plane passing through the receiving point x and the edge itself, and one halfspace for the supporting plane of the source itself.

Checking a surface s_j against our shaft is quick and easy: if all the vertices of s_j lie outside any of the halfspace defining the shaft, then s_j can be safely culled out of consideration as its projection onto the supporting plane of s_0 could never intersect the source.

In order to keep track of which parts of the source are visible from x we use a two-dimensional BSP-tree. A boolean variable is associated with each of the leaves of the BSP-tree to indicate whether or not the region of the source corresponding to that leaf is visible or occluded. Each time the projection of a new surface s_j is intersected with s_0, the BSP-tree is updated to reflect the change in the visible portions of the source. The intersection algorithm uses the techniques presented by Naylor *et al.* [NAYL90], who show how polyhedral set operations can be implemented by merging BSP-trees.

Once the visible parts of the source are determined, the form factor from x to s_0, $F_{dA_i(x)\rightarrow A_0}$, is computed as the sum of the unoccluded form factors from x and each of the visible parts of the source. Each of these form factors, in turn, is computed using the analytical formula reported in Equation (4.2).

4.6 Adaptive Subdivision

Discontinuity meshing guarantees that no mesh elements will be crossed by strong discontinuities, thus making simple polynomial functions suitable to approximate the radiance distribution across a receiving surface. However, there are still instances in which the radiance across a mesh element varies too quickly for the simple quadratic interpolation scheme presented in Section 4.4 to provide a sufficient approximation. Large or badly shaped mesh elements and the close proximity of a strong light source are often responsible for this problem. In these cases, adaptive mesh subdivision can be used to effectively reduce the errors.

In order to design a mesh refinement algorithm, we must decide when and how to subdivide a mesh element.

4.6.1 When to Subdivide

Intuitively, a mesh element should be subdivided whenever its approximation to the correct radiance function falls below the desired accuracy according to a predetermined error metric.

Defining the best error metric to use is difficult. Each metric quantifies a particular kind of error. In the following, we will present three different error metrics and discuss their effect on the accuracy of the solution as well as their ease of implementation and computational cost.

2-norm error metric

The 2-norm error over a surface element e is defined as:

$$\|L - \tilde{L}\|_2 = (\int_e (L(x) - \tilde{L}(x))^2 dA(x))^{\frac{1}{2}} \tag{4.3}$$

where

$L(x)$ is the actual radiance at point x on element e;

$\tilde{L}(x)$ is the estimated radiance at point x on element e as computed from the quadratic interpolant defined over e;

$dA(x)$ is a differential surface area centered at x.

This metric measures the overall error over the entire surface e. Using the 2-norm error metric to control adaptive subdivision has the effect of improving the quantitative accuracy of the solution. However the distribution of the error across the surface is not taken into account; for example, the same error could be localized to a small subregion or be equally distributed across the whole surface element. Visually, the former case would cause much more noticeable artifacts than the latter; thus, the 2-norm error metric is not very well suited for qualitative analysis. Furthermore, the error measured by this metric is proportional to the area of the surface element and can therefore lead to unnecessary subdivision of large elements.

Infinity-norm error metric

The infinity-norm error over a surface element e is defined as the maximum absolute error over the element:

$$\|L - \tilde{L}\|_{\infty} = \max_e |L(x) - \tilde{L}(x)| \,. \tag{4.4}$$

Using the ∞-norm error metric to trigger the refinement process has the effect of improving the accuracy of the solution qualitatively, i.e. visually, as well as quantitatively. Also, unlike the 2-norm error metric, this metric is independent of the element size and thus allows the simulation to capitalize on large elements whenever possible. This independence, though, coupled with numerical approximations, could lead to infinite subdivision. This problem can be easily solved by preventing subdivision whenever the element area falls below a prespecified minimum.

Computing the exact error according to either the 2-norm or the ∞-norm error metric is not feasible since it would require knowledge of the actual radiance at infinitely many points across a surface. An approximation, though, can be computed using a finite number of samples. The more the samples, the better the approximation; however, the cost of estimating the error should, of course, be less than that of computing the radiance samples necessary to define the interpolant used to approximate the radiance function over an element.

On the average, excluding boundary conditions, samples taken on a mesh vertex will typically be shared among about six elements and those taken on a mesh edge will be shared between two elements; thus, using a piecewise quadratic interpolant to reconstruct the radiance function will take about two samples per element ($3 \times \frac{1}{6} + 3 \times \frac{1}{2}$). This suggests that a single radiance sample be used to estimate the error and decide whether or not to trigger adaptive subdivision. We take this sample at the centroid of the element. Although the error estimate thus computed is not always reliable and can cause the adaptive subdivision to stop prematurely; in practice, this strategy has yielded good results.

Smooth-boundary error metric

The piecewise quadratic interpolation scheme presented in Section 4.4 does not impose C^1 continuity across elements boundaries. This may lead to unwanted Mach bands along mesh edges classified as D^2 or higher. Adaptive subdivision can help eliminate these artifacts.

In fact, we can measure the angle of the cusp formed by the reconstructed radiance function along the edge between two elements, and subdivide if this angle is too far from flat. More formally, let *angle*(f, e) be the function that takes as input a triangular face f and an edge e incident on f, and whose behavior is described by the following pseudocode:

$x \leftarrow$ *midpoint*(e)
$g \leftarrow$ *neighbor*(f, e)
if *order*$(e) \geq D^2$ **and** $g \neq$ NIL **then**
    angle $\leftarrow$ *arccos*(*tangent*$(f, e, x) \cdot$ *tangent*(g, e, x))
else
    angle $\leftarrow \pi$

end if
return *angle*

where

midpoint(e) is the midpoint of edge e;

neighbor(f, e) is the neighbor of face f across from edge e;

tangent(f, e, x) is the tangent vector to the reconstructed radiance function for element f, at point x, and in the direction perpendicular to edge e.

Then, we define the *smooth-boundary error* of a triangular face f as:

$$\max_E \, | \, angle(f, e_i) - \pi \, | \tag{4.5}$$

where $E = \{e_i\}$ is the set of edges incident on f.

This error metric has the effect of forcing subdivision in areas where the piecewise quadratic interpolant fails to reconstruct a smooth approximation to the radiance function.

4.6.2 How to Subdivide

The mesh subdivision strategy we have adopted uses the 2-triangles mesh refinement algorithm presented by Rivara [RIVA87].

The meshes produced by her method have a number of desirable properties that make them particularly suitable to the task of capturing radiance functions:

1. No T-vertices are introduced by the refinement process. This property is important since T-vertices could introduce value discontinuities in the reconstructed radiance function.

2. The transition between large and small triangles is smooth; i.e. it happens gradually over intermediate-sized triangles rather than abruptly. The resulting distribution of sample points is less prone to generating visual artifacts at reconstruction time.

3. The refined mesh tends to be made of well shaped triangles. More precisely, the algorithm yields triangles whose longest edge is shorter than twice the length of either of the other edges [RIVA87]. This property is important because the binary space partitioning scheme used to construct the discontinuity mesh is liable to produce elements with bad aspect ratios. Well shaped elements, in fact, can reduce errors in the radiance approximation [BAUM91].

2-triangles mesh refinement algorithm

Rivara's 2-triangles mesh refinement algorithm is based on simple bisection operations and is illustrated in Figure 4.24. The algorithm starts by bisecting an element s by the midpoint of its longest edge e (a). This operation creates a new vertex v. If element s has a neighbor t across edge e, then v is a T-vertex and the triangulation is no longer considered valid. In order to restore validity, element t is also bisected along the midpoint of its longest edge e' (b). If e and e' are the same edge, then we are done; otherwise, v and v', the midpoint of edge e', are connected to form new triangles (c). If v' is, in turn, a T-vertex, the process is iterated until all mesh elements are triangles once again.

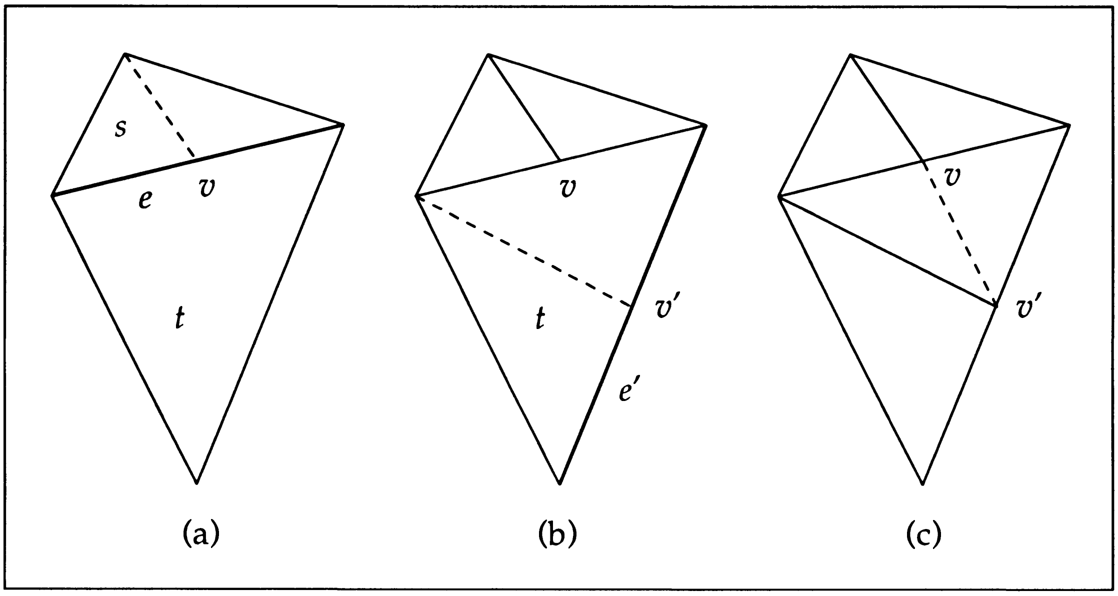


Figure 4.24: Rivara's 2-triangles mesh refinement algorithm

Implementation details

In our implementation, whenever a triangular element is bisected, the corresponding dm-tree node is refined as well. In this way, point location within a mesh can be carried out efficiently using the same binary search described in Section 4.4. Also, maintaining a one-to-one correspondence between the mesh elements and the leaf nodes of the dm-tree, enables us to implement adaptive subdivision without any change to the existing data structures.

The new edges created by the refinement process are marked as D^∞. This information can be used at radiance reconstruction time to maintain the highest possible order of continuity across these edges. In the current implementation, however, the reconstructed radiance function is only guaranteed to be C^0 across these edges.

Every time a new triangular element is created during the refinement process, new radiance samples must be taken at its vertices and edge midpoints. Most of the sample locations, however, will fall on the vertices and edge midpoints of the elements of the original mesh. Recomputation of the same values is avoided by means of a simple flagging scheme.

This adaptive refinement algorithm has proved to work very well. In particular, even a small amount of subdivision can improve the quality of the discontinuity mesh considerably.

Chapter 5

Discontinuity Meshing Radiosity

In the previous chapter, we introduced the concept of discontinuity meshing. We applied this concept to the design of an algorithm capable of accurately capturing the characteristics of the radiance functions over a set of polygonal surfaces resulting from the direct illumination of a finite area light source of constant emission.

Despite its accuracy, this algorithm, as it is, is not very useful. Real environments typically contain multiple light sources and often a significant portion of their illumination is due to interreflections between surfaces. In this chapter, we show how to develop a radiosity algorithm that uses the single source discontinuity meshing algorithm as a building block to compute a global illumination solution for diffuse polygonal environments.

5.1 Discontinuity Meshing Radiosity—Algorithm Overview

The global illumination problem was discussed in Section 2.2 and its formulation for diffuse environments was given in Equation (2.3). Following the conventions established before, we restate our problem as follows:

Given a set $S = \{s_1, s_2, \ldots, s_n\}$ of convex polygonal surfaces, their diffuse BRDF's $f_{r1}, f_{r2}, \ldots, f_{rn}$, and constant emissions $L_1^e, L_2^e, \ldots, L_n^e$, compute the set of radiance functions $L_1, L_2, \ldots, L_n$ such that

$$L_i(x) = L_i^e + f_{ri} \sum_{s_j \in S} \int_{x' \in s_j} L_j(x') \frac{\cos\theta_i \cos\theta_j}{r^2} v(x, x') dA(x') \qquad (5.1)$$

for all $i = 1, 2, \ldots, n$.

As with the single source discontinuity meshing algorithm presented in the previous chapter, we will use a meshing approach to sample the radiance functions at a finite set of locations and reconstruct an approximation $\tilde{L}_i$ to the actual solution.

The new discontinuity meshing radiosity algorithm is based on progressive refinement radiosity [COHE88b]. This approach essentially reduces the problem of solving the diffuse rendering equation into that of computing the direct illumination from a single light source. Since we have already developed an algorithm that solves this last problem, the progressive refinement approach is only a natural choice.

Pseudocode for the discontinuity meshing radiosity algorithm is given in Figure 5.1. The cumulative effect of multiple light sources is maintained in $\tilde{L}_i$. Initially, the radiance $\tilde{L}_i$ of a surface s_i is simply equal to the emission L_i^e (possibly zero) of the surface. Then, the approximation $\tilde{L}_i$ is refined by adding, at each iteration, the direct illumination $\tilde{L}_{ij}$ contributed by a new light source s_j, whether primary or secondary.

In order to implement this approach, two major steps are necessary. First, the single source discontinuity meshing algorithm must be extended to handle secondary sources whose emission is typically not a constant. Second, a method

foreach *Surfaces*$_i$ **in** S **do**
    $\tilde{L}_i \leftarrow L_i^e$
    $\Delta\tilde{L}_i \leftarrow L_i^e$
end for
repeat until convergence
    find surface s_j with maximum unshot power
    compute $\tilde{L}_{ij}$ for all s_i in S
    foreach *Surfaces*$_i$ **in** S **do**
        $\tilde{L}_i \leftarrow \tilde{L}_i + \tilde{L}_{ij}$
        $\Delta\tilde{L}_i \leftarrow \Delta\tilde{L}_i + \tilde{L}_{ij}$
    end for
    $\Delta\tilde{L}_j \leftarrow 0$
end repeat

Figure 5.1: Pseudocode for the discontinuity meshing radiosity algorithm

of combining radiance functions implicitly defined by sample locations on different discontinuity meshes must be devised.

In the following sections, we describe these two steps in detail and conclude with a discussion on the algorithmic efficiency.

5.2 Locating the Discontinuities

As shown in Section 2.4, the direct illumination due to an area light source of constant emission results in radiance functions that may exhibit D^0, D^1, and D^2 discontinuities. The ability to predict the order and location of these discontinuities was used in the previous chapter to design an algorithm capable of accurately capturing the direct illumination due to such a source on a set of polygonal surfaces.

When the indirect illumination is also considered, additional discontinuities are created. According to the Discontinuity Propagation Law presented in Section 2.4.3, in fact, discontinuities of order k on a secondary light source are propagated into the environment, resulting in discontinuities of order $k+1$ and $k+2$. Thus, the total number of discontinuity curves in the environment grows rapidly with the number of interreflections considered. In the limit, a full global illumination solution would exhibit an infinite number of discontinuities.

Trying to locate all these discontinuities would be impractical even for a finite number of interreflections. Fortunately, though, as the order of a discontinuity increases, its effect on both the quantitative and qualitative accuracy of our simulations diminishes.[1] Following this simple observation, it is possible to drastically cut down on the number of discontinuities considered without noticeably affecting the quality of the simulations.

5.2.1 Implementation

In our implementation, we consider D^0, D^1, and D^2 discontinuities only. Capturing higher order discontinuities would add an order of magnitude to the time and storage costs of the simulation and would require modifying our reconstruction scheme. The piecewise quadratic interpolation technique presented in Section 4.4, in fact, is capable of resolving D^0, D^1, and D^2 discontinuities only.

In order to propagate discontinuities on a secondary source that might result in D^1 or D^2 discontinuities on a receiver, we consider visual events in which the participating edge is either a D^0 or D^1 discontinuity segment on the source.

The pseudocode given in Figure 5.2 shows how to generate and classify the

[1]The order of a discontinuity is actually only one of several factors whose interplay determines the visual importance of a discontinuity. Developing a metric of visual importance involves the study of complicated perceptual issues and was not attempted in this research. A brief discussion of the problem, however, is offered in the concluding chapter.

```
foreach discontinuity segment E on source s_j do
    if order of E = D^0 then
        foreach vertex V in set of potentially occluding vertices do
            if V is the endpoint of an edge E' that is coplanar with E then
                EVevent ev.Set(E, V, D^1)
            else
                EVevent ev.Set(E, V, D^2)
            end if
            model.TraceEV(ev)
        end for
    else if order of E = D^1 then
        foreach vertex V in set of potentially occluding vertices do
            if V is the endpoint of an edge E' that is coplanar with E then
                EVevent ev.Set(E, V, D^2)
                model.TraceEV(ev)
            end if
        end for
    end if
end for
```

Figure 5.2: Pseudocode for the generation and classification of the visual events associated with the discontinuity curves on secondary light sources

visual events associated with the discontinuity curves of a secondary light source s_j. Additional visual events are generated by the edges and vertices of the light source as it was done for primary light sources (see pseudocode in Figure 4.6.)

We look at each discontinuity segment on source s_j, in turn. D^0 discontinuities interact with the set of potentially occluding vertices (see discussion in Section 4.2.2) to generate D^1 and D^2 EV visual events which are then traced through the environment. D^1 discontinuities, instead, can potentially generate both D^2 and D^3 EV visual events, but only D^2 events are actually created and traced. D^2 discontinuity segments on the source, finally, can only propagate to D^3 and higher order discontinuities and are therefore ignored.

5.3 Computing the Source Contribution to a Point

In a global illumination simulation, any surface in the environment may act as a secondary light source by reflecting part of the incident light back into the environment. These sources, as opposed to the primary light sources discussed in the previous chapter, typically have non-uniform emittance.

The radiance contributed by a source s_j to a point x on a receiving surface s_i is given by:

$$L_{ij}(x) = f_{ri} \int_{x' \in s_j} L_j(x') \frac{\cos\theta_i \cos\theta_j}{r^2} v(x, x') dA(x') \tag{5.2}$$

As was shown in Section 4.5, the visibility term $v(x, x')$ can be taken out from under the integral by determining which parts of s_j are visible from point x. However, the presence of the arbitrary $L_j(x')$ term still precludes the possibility for an analytical solution.

Our algorithm uses numerical integration techniques to compute the contribution from each of the visible parts of the light source. These regions are convex polygons of arbitrarily many sides. Rather than trying to integrate L_j over these complicated domains, we first triangulate each region and compute the radiance contribution as:

$$L_{ij}(x) = f_{ri} \sum_{t \in T} \frac{1}{A_t} \int_{x' \in t} L_j(x') dA(x') \int_{x' \in t} \frac{\cos\theta_i \cos\theta_j}{r^2} dA(x') \tag{5.3}$$

where

T is the set of visible triangles for s_j; and

A_t is the area of triangle t.

Notice that the radiance contribution of each individual triangle is computed as the product of the area average radiance of the triangle and its form factor. Splitting the contribution integral in this fashion is only accurate if the radiance across a triangle is relatively constant. In order to meet this condition, our algorithm adaptively subdivides the triangle into four similar triangles until the relative error between two successive estimates falls below a specified threshold.

The $\int_{x' \in t} L_i(x')dA(x')$ term in the equation above is the radiant flux of source s_j over the area of triangle t. Assuming that the behavior of L_i over t can be well approximated by a quadratic polynomial, it is possible to evaluate the integral analytically using the quadrature formula for quadratic triangular elements given by Brebbia and Dominguez [BREB89], page 269:

$$\int_{x' \in t} L_i(x')dA(x') = \frac{1}{3}(L_i(x_1) + L_i(x_2) + L_i(x_3)) \tag{5.4}$$

where x_1, x_2, and x_3 are the midpoints of the edges of t.

The form factor integral is also evaluated analytically using Equation (4.2) as was shown in Section 4.5.

5.4 Combining Radiance Functions

Conventional radiosity systems use a single mesh throughout the entire simulation. The mesh can be adaptively subdivided in an attempt to capture fine illumination details, but basically the same set of sample locations is used to compute the contribution from all the light sources, both primary and secondary, as well as to store the accumulated radiance values.

Such a scheme would be impractical for discontinuity meshing, since every surface in the environment may become a source, participating in a different set of visual events. Accounting for all these visual events would result in a very large number of critical curves and very complex meshes. Computing the radiance contribution from every source to such a mesh would be very costly.

Our solution, instead, consists of building a separate discontinuity mesh for each source s_j using only the critical curves resulting from visual events involving this source. In this way, the number of points at which the contribution L_{ij} from a source s_j to a receiving surface s_i is computed is substantially reduced.

The cumulative radiance function L_i is also stored as a discontinuity mesh since, of course, any discontinuity in a L_{ij} will also be a discontinuity in L_i. Our progressive radiosity system refines $\tilde{L}_i$ incrementally by adding in at each iteration the latest contribution $\tilde{L}_{ij}$:

$$\tilde{L}_i^{\text{new}} \leftarrow \tilde{L}_i^{\text{old}} + \tilde{L}_{ij} \tag{5.5}$$

In this way, only the storage used to represent the current $\tilde{L}_i$ needs to be maintained, while that required by the $\tilde{L}_{ij}$ is released at the end of each iteration.

Since for each source s_j the discontinuity mesh of the interpolant $\tilde{L}_{ij}$ is typically substantially different from the meshes used for previous sources, the problem of merging contributions is not trivial.

In order to compute $\tilde{L}_i^{\text{new}}$ we start by creating a new discontinuity mesh containing all the significant discontinuity boundaries from $\tilde{L}_i^{\text{old}}$ and $\tilde{L}_{ij}$. These boundaries can be identified by inspecting the corresponding discontinuity meshes and can be inserted into the new discontinuity mesh using the algorithm outlined in Figure 4.14. The new mesh is then triangulated and quadratic elements are computed as described in subsections 4.4 and 4.6. Note however, that the value at any point p is given by $\tilde{L}_i^{\text{old}}(p) + \tilde{L}_{ij}(p)$. Thus, rather than integrating over the source at each sample location of the new mesh, we only need to evaluate $\tilde{L}_i^{\text{old}}(p)$ and $\tilde{L}_{ij}(p)$. Each value is computed by locating the element containing p in the appropriate mesh and evaluating the quadratic interpolant. The key steps of the algorithm are depicted by way of a simple example in Figure 5.3.

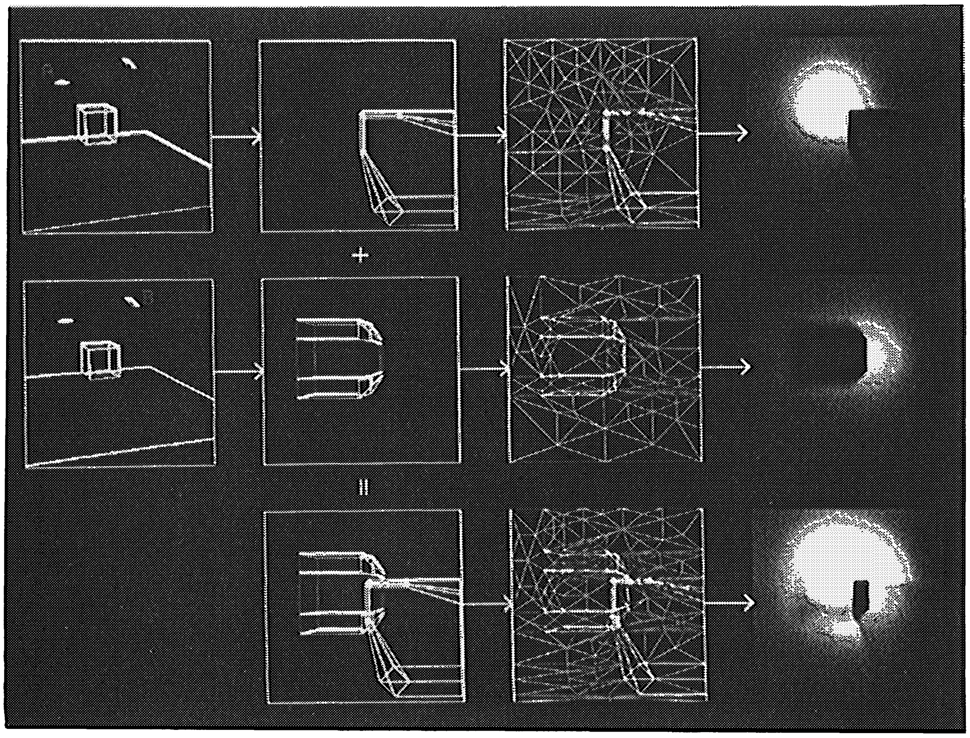
Since there is no need to process visual events nor to compute the contribution of a source at each location, constructing the new interpolant $\tilde{L}_i^{\text{new}}$ is typically much faster than computing the contribution $\tilde{L}_{ij}$. As will be shown in Chapter 6, in fact, the accumulation process takes up only a small percentage of the total computation time and thus does not compromise the advantages gained by using a separate discontinuity mesh for each source.

Our scheme for combining radiance functions is essentially a resampling method, and therefore a certain loss of accuracy can be expected. Notice, though, that since the important critical curves are reproduced in the new discontinuity mesh, the resampling occurs inside regions where the radiance function can be considered smooth, and thus the additional error is small. Furthermore, since the time spent combining radiance functions constitutes a very small part of the total simulation time, we can afford to increase the level of adaptive subdivision used by the combination algorithm so as to provide the desired level of accuracy.

5.5 Efficiency Considerations

The discontinuity meshing radiosity algorithm presented in this chapter is very accurate, but also very computationally intensive.

Despite the restrictions on the kind of visual events that we chose to trace through the environment (see Sections 4.2.2 and 5.2,) the process of locating the



(a) (b) (c) (d)

Figure 5.3: Combining the illumination from two light sources in the presence of a simple obstacle (the cube floating above the floor). Columns (b), (c), and (d) show a plan view of the floor. Row 1. (a) Surface A is selected as the current source. (b) Discontinuity boundaries due to source A are computed. D^1 and D^2 discontinuities are drawn respectively in red and yellow. (c) A triangular mesh which resolves the discontinuity boundaries is produced. (d) The source contribution $L_{\text{floor},A}(x)$ is computed at sample locations within the mesh to yield the interpolant $\tilde{L}_{\text{floor},A}$. Row 2. The process is repeated with surface B selected as the source. Row 3. (b) The discontinuity boundaries due to A and B are combined. (c) A new triangular mesh with the combined discontinuity boundaries is produced. (d) Interpolants $\tilde{L}_{\text{floor},A}$ and $\tilde{L}_{\text{floor},B}$ are evaluated at sample locations within the new mesh and the sample values are added to yield the combined contribution of A and B.

radiance function discontinuities takes a large portion of the execution time. The visual events generated by secondary light sources are mainly responsible for this cost. The number of visual events associated with a secondary source, in fact, may be very large due to both the possible large size of the surface compared to that of typical primary sources and to the possible large number of D^0 and D^1 discontinuity segments falling on it (for example, the floor of a furnished room which contains many shadows.)

Computing the radiance contribution of the light source to a point is also a computationally intensive process. Determining the visible portions of a source for each sample point we want to take, in fact, can be much more expensive than using a more conventional point sampling approach such as the hemicube [COHE85] or ray-traced form factors [WALL89].

Another possible source of inefficiency can be found in our solution to the problem of combining radiance functions. Any time that two radiance functions are merged, all their discontinuities are inserted in the resulting discontinuity mesh. Since the set of discontinuities generated on a receiver is typically different for every source, an attempt to compute an accurate global illumination solution for even a medium size environment might result in storage problems.

Many of the above inefficiencies can be alleviated by noticing that many computations are carried out to a much higher degree of precision than is required to achieve a final radiosity solution of a given accuracy. The same observation led Pat Hanrahan *et al.* [HANR91b] to the design of a rapid hierarchical radiosity algorithm, which is an order of complexity faster than its predecessors.

In the following sections, we use this observation to improve the efficiency of the discontinuity meshing radiosity algorithm.

5.5.1 Computation of Radiance Discontinuities

In Chapter 4 we showed how incorporating lines of discontinuity in the radiance function (including its first and second derivatives) as edges in a surface mesh may result in an illumination algorithm capable of eliminating shadow leaks, jagged shadow boundaries, and other artifacts that afflict conventional radiosity systems.

Some of these discontinuity edges, however, may sometimes be omitted from a surface mesh without resulting in any visible artifacts. For example, if a light source s_j contributes but a very small percentage to the overall illumination of a receiving surface s_i, any discontinuity lines s_j may cause on s_i are likely to be unnoticeable.

Limiting the range of visual events

The radiance contribution of a source s_j to a surface s_i decreases quadratically with the distance between s_j and s_i. When the two surfaces are sufficiently distant, then, computing discontinuity lines becomes unnecessary. In our discontinuity meshing radiosity algorithm, discontinuity lines are computed by tracing visual events from a source through the environment, visiting the receiving surfaces in sorted order, away from the source. This computation may be shortened by terminating the traversal as soon as we reach a surface s_i such that L_{ij} falls below a specified threshold. The precise value of L_{ij} is not critical for termination criteria as long as we can compute an upper bound for it. Such an upper bound can be computed much faster than L_{ij} itself by simply assuming total visibility between the source and receiver.

Dropping weak discontinuities when combining radiance functions

In Section 5.4, we described an algorithm for combining radiance functions that takes two piecewise quadratic interpolants, $\tilde{L}_i^{\text{old}}$ and $\tilde{L}_{ij}$, and yields a new interpolant, $\tilde{L}_i^{\text{new}}$, representing their combined effect. Each of the initial interpolants is defined over its own discontinuity mesh. The discontinuity mesh for $\tilde{L}_i^{\text{new}}$ is constructed starting with a single DM-tree node and adding in all the discontinuity segments from both $\tilde{L}_i^{\text{old}}$ and $\tilde{L}_{ij}$.

As we observed in the introduction to this section, this algorithm can result in an excessive growth of the DM-tree data structures as the progressive refinement radiosity progresses. This growth can be controlled, often without significant losses in accuracy, by noticing that not all discontinuities are equally noticeable. Some of the discontinuities of $\tilde{L}_{ij}$, in fact, may become invisible once they are combined with $\tilde{L}_i^{\text{old}}$, and vice versa. Developing a metric capable of assessing the visual impact of a discontinuity in the illumination across a surface, however, involves complicated perceptual issues that were beyond the scope of the research described in this thesis. A brief discussion of the problem is attempted in the concluding chapter.

Although we recognized the need for perceptual metrics, only a simple, purely quantitative, approach was actually implemented. Rather than trying to measure the relative strength of a discontinuity segment c, we measure its relative contribution to the radiance of the receiving surface; assuming c is coming from a radiance function $\tilde{L}_{ij}$, we compute:

$$\max_{x \in c} \frac{\tilde{L}_{ij}(x)}{\tilde{L}_i^{\mathrm{old}}(x) + \tilde{L}_{ij}(x)} \tag{5.6}$$

and insert the segment in the discontinuity mesh for $\tilde{L}_i^{\mathrm{new}}$ only if its strength exceeds a specified threshold ϵ_L. Only a few samples taken along the discontinuity segment c are needed to obtain a good estimate of its relative contribution. By controlling the magnitude of ϵ_L we can control the growth of our data structures and are therefore able to apply our radiosity algorithm to more complex environments.

5.5.2 Computation of the Contribution of a Source to a Point

In Section 5.3, we described an algorithm for computing the contribution from a source s_j of arbitrary emission to a point x on a receiving surface s_i in the presence of occlusions.

Despite the efficient implementation using shaft culling (see Section 4.5), a significant part of the computation time is still spent projecting obstacles onto the supporting plane of the source and intersecting them with s_j to compute the parts of the light source that are visible from each sample point x.

When the contribution $L_{ij}(x)$ is very small, its calculation by means of the above algorithm is carried out to unnecessary accuracy. Thus, whenever the contribution is estimated to fall below a specified threshold, we can evaluate $L_{ij}(x)$ by means of a faster ray-traced form factor algorithm [WALL89] without a significant loss in the overall accuracy of our solution L_i.

Once again, a precise estimate of the contribution $L_{ij}(x)$ is not necessary as long as we can compute an upper bound for it. This upper bound can be computed very quickly by numerically integrating $L_{ij}(x)$ over the entire source s_j using the algorithm presented in Section 5.3, but omitting the visibility calculations.

When the contribution from s_j to a receiving surface s_i is computed using different algorithms for nearby sample locations, annoying visual artifacts may result. These artifacts are due to a sudden change in the accuracy of the contribution estimate when switching from the analytical projection-based method to the ray-traced form factor algorithm or vice versa. This problem can be avoided by requiring that the same method be used for all the sample points x_k on the surface s_i necessary to capture the contribution L_{ij} of the source s_j. In other words, a single method is used to compute the energy transfer within a given source-receiver pair.

If each input polygon is treated as an independent surface, radiance discontinuities may occur at the boundary between two contiguous surfaces, resulting in the visual artifacts mentioned above. Our implementation avoids this problem by preprocessing the input scene so that each surface is actually a connected set of coplanar polygons of homogeneous surface reflectance characteristics. Furthermore, polygons within the same surface are topologically connected into a single surface discontinuity mesh. This way, mesh conformity is maintained across neighboring polygons even during adaptive subdivision of the mesh.

Chapter 6

Results

6.1 Physical vs. Simulated World

This section presents a qualitative comparison between a photograph of a simple physical environment and a computer generated rendering of the same scene based on discontinuity meshing radiosity.

The test environment consists of three surfaces only: a receiver, a light source, and an occluding surface. The receiver is a square piece of wood covered with several coats of matte white paint. The source is a lamp centered above the receiver and casting its light through a square-shaped diffusing panel. The obstacle is a triangle cut out of black cardboard and suspended over the receiver using fish line.

The receiver was surrounded on three sides (one side was missing to allow for photographs) by matte black panels so as to reduce interreflections and allow us to compare the effect of direct illumination only.

Figure 6.1 shows the reference photograph of this simple scene. Figure 6.2 shows a rendering of the same view, computed from a view-independent so-

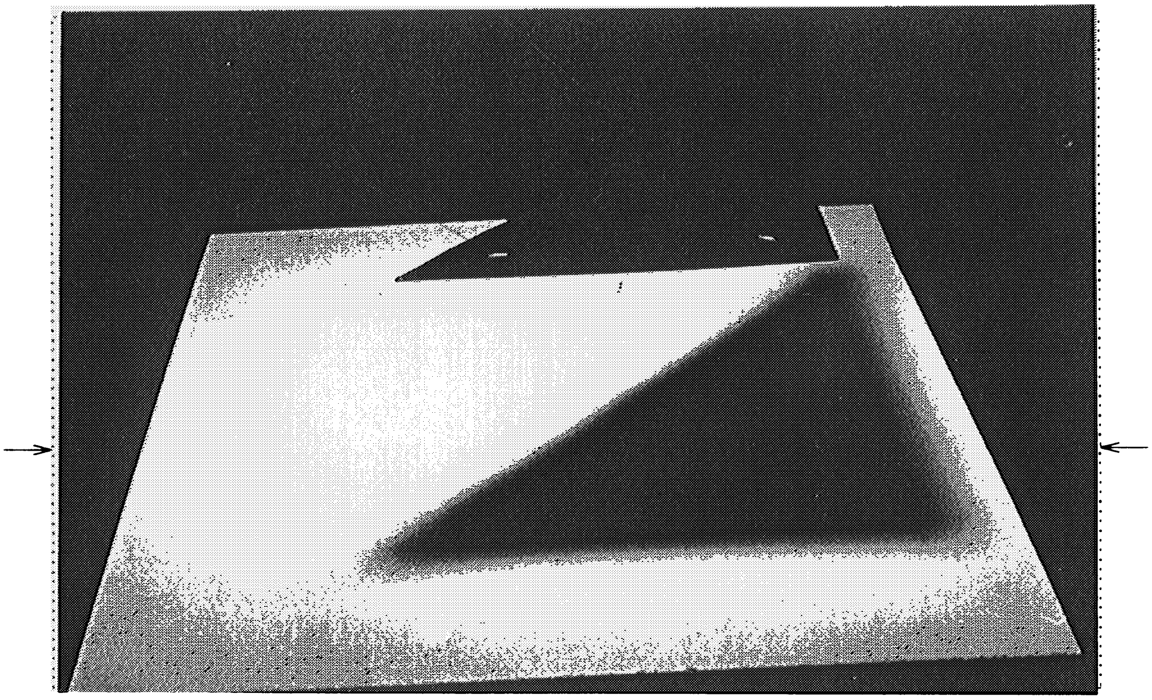


Figure 6.1: Photograph of a simple scene

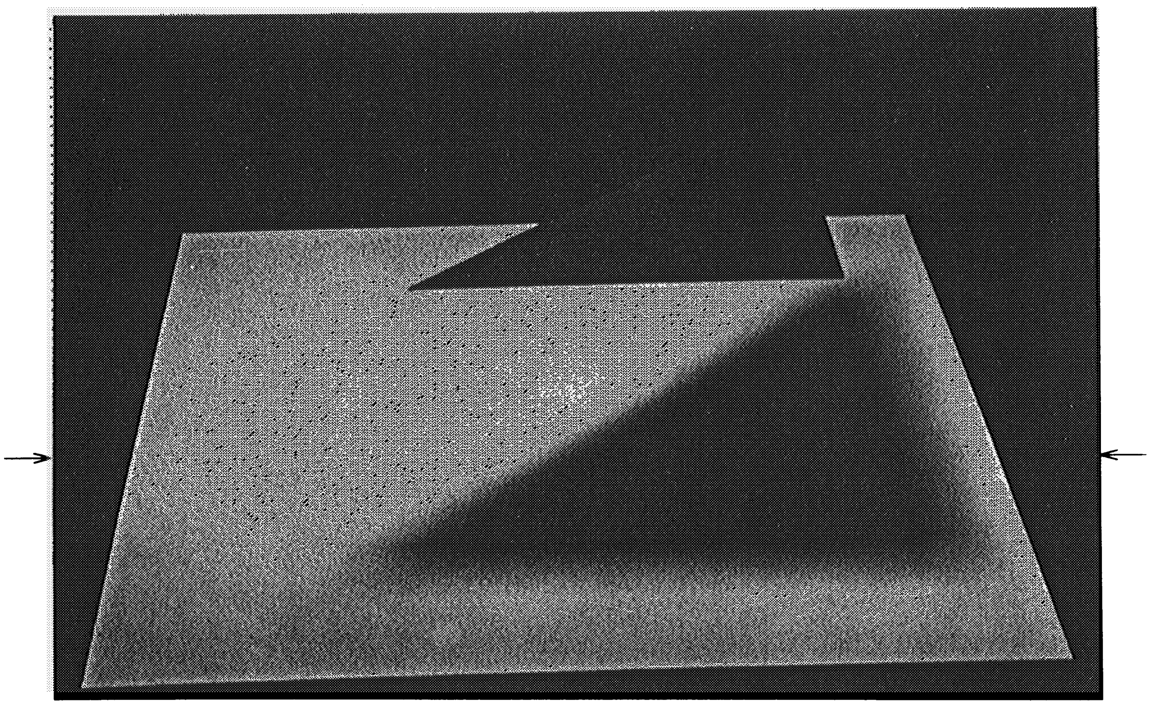


Figure 6.2: Computer simulation of a simple scene

lution generated using the single source discontinuity meshing algorithm described in Chapter 4. The two images are remarkably similar. Even subtle illumination details are captured. Note, for example, the dark bands extending from the corners of the umbra and into the penumbra region as well as the bright horizontal Mach band along the outer edge of the penumbra near the bottom of the picture.

The behavior of the illumination across the receiving surface can be perhaps better understood by looking at a cross section of the radiance function. Figure 6.3 shows a cross section taken along a horizontal scanline passing through the middle of the triangular umbra of Figures 6.1 and 6.2. Only an overall qualitative comparison is meant here since the views in the original images are aligned only very approximately. The physical values reflect noise introduced by the photographic and scanning processes. The scanline shown in the figure traverses both the umbra and penumbra regions cast on the receiver. The plot of the simulated radiance clearly shows how shadow boundaries correspond to sharp changes in an otherwise smooth radiance function.

These sharp changes in the illumination occur along discontinuity boundaries. Figure 6.4a shows their location in a top view of the receiving surface. Figure 6.4b, instead, shows how these lines can be incorporated into a discontinuity mesh to yield the solution displayed in Figure 6.2.

6.2 Single Source Discontinuity Meshing vs. Analytical

This section presents a comparison between a discontinuity meshing solution and an analytical solution of a simple environment.

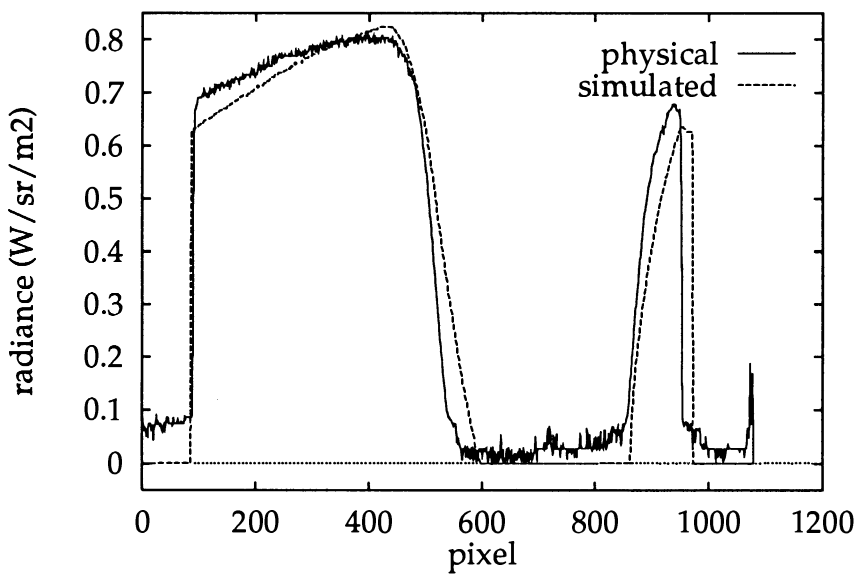


Figure 6.3: Plot of the radiance values across a scanline of the images shown in Figures 6.1 and 6.2. The scanline crosses the middle of the triangular umbra in the original images. Only an overall qualitative comparison is meant here since the views in the original images are aligned only very approximately. The physical values reflect noise introduced by the photographic and scanning processes.

Unfortunately no closed-form solution to the radiosity equation is known for arbitrary environments. It is possible, however, to analytically compute the contribution of one or more polygonal light sources of constant emission to any point in an arbitrary polygonal environment (see Section 4.5.)

Therefore, we chose to compute our reference solution in a view-dependent fashion, by sampling the illumination at a finite set of sample points arranged in a regular grid on a projection plane, and compare it to a rendered view of a discontinuity meshing solution computed using direct illumination only.

Two views were selected for the comparisons presented in this section. The first, shown in Figure 6.5, is a view of a simple environment illuminated by a single square light source placed at the center of the ceiling. The second, shown in Figure 6.6, is a top view of the floor of the same simple environment.

Because of the view-dependent method, it was necessary to compute a different reference solution for each of the two figures shown in the upper right corners (b). The discontinuity meshing solution, instead, was computed only once and then rendered twice, to obtain the two different views shown in the upper left corners (a).

The resulting mesh for the discontinuity meshing solution is shown in image (c) of the two figures. Image (d), instead, shows a pseudocolored image of the absolute error between discontinuity meshing and reference radiance values, computed on a pixel-by-pixel basis. Finally, image (e) shows a histogram of the pseudocolored error image. The histogram is colored to show the mapping between error values and colors.

The pseudocolored images are very helpful for a qualitative analysis of the error. A look at the mesh and error images shown side-by-side in Figure 6.5, reveals a correlation between the magnitude of the error and the underlying mesh structure. In general, in fact, the error is small along mesh edges and is larger in the middle of mesh elements. Furthermore, inspection of the mesh and error images in Figure 6.6 shows that the largest errors, corresponding to the red regions, occur near singular points. In fact, the largest error found occurs around a singularity where the lower left corner (as seen from above) of the tall box touches the floor (see Figure 6.6d.) The magnitude of this error is 0.099 $Wsr^{-1}m^{-2}$, while the brightest radiance value in the reference images was 4.72 $Wsr^{-1}m^{-2}$.

Figure 6.8a shows a plot of the radiance function along the scanline passing through such largest error. The match between the discontinuity meshing and analytical solutions is so close that it was necessary to zoom in in order to reveal any discrepancies (Figure 6.8b.)

In order to give a quantitative measure of the error, we used 2-norms [GOLU89]. Given an image A of size n-by-m, we define its 2-norm, $||A||_2$, as:

$$||A||_2 = \left(\frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ij})^2 \right)^{\frac{1}{2}} \tag{6.1}$$

We can then define the absolute error E and the relative error e between two images A and B respectively as:

$$E = ||A - B||_2 = \left(\frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ij} - B_{ij})^2 \right)^{\frac{1}{2}} \tag{6.2}$$

and

$$e = \frac{||A - B||_2}{||B||_2} = \left(\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ij} - B_{ij})^2}{\sum_{i=1}^{n} \sum_{j=1}^{m} (B_{ij})^2} \right)^{\frac{1}{2}} \tag{6.3}$$

This definition of absolute and relative errors is useful because it gives a measure of the average error over the entire images while at the same time giving particular weight to large localized errors. In contrast, computing the overall error as a simple unweighted average of the individual pixel errors would have had the effect of washing out localized, but possibly large, errors. Note also that the A_{ij}'s and B_{ij}'s are actual radiance values, i.e. values which have not yet been mapped to colors for display on a CRT device.

Table 6.1 reports the absolute error E and relative error e for both the full view shown in Figure 6.5 and the top view of the floor shown in Figure 6.6. The 2-norm of the reference image, $\|B\|_2$, is also given for reference. As it can be seen, the discontinuity meshing algorithm produced a solution accurate to within about half of one percent error.

6.3 Accumulating Radiance Functions by Resampling vs. Exact Addition

This section examines the error introduced by the resampling scheme used to combine radiance function due to different sources.

The accumulation algorithm was described in detail in Section 5.4. In order to add two radiance functions their underlying discontinuity meshes are inspected and a new discontinuity mesh is created. The new mesh contains as edges the discontinuity segments of each of the two input meshes, but can otherwise be very different from them. Radiance values for each of the new mesh elements are then determined by sampling the two input radiance functions at corresponding locations. This mechanism essentially introduces an additional resampling step in our radiosity algorithm and is likely to introduce additional errors in our radiance approximations. Because the discontinuity segments are reproduced in the combined mesh and because the radiance function generally varies slowly

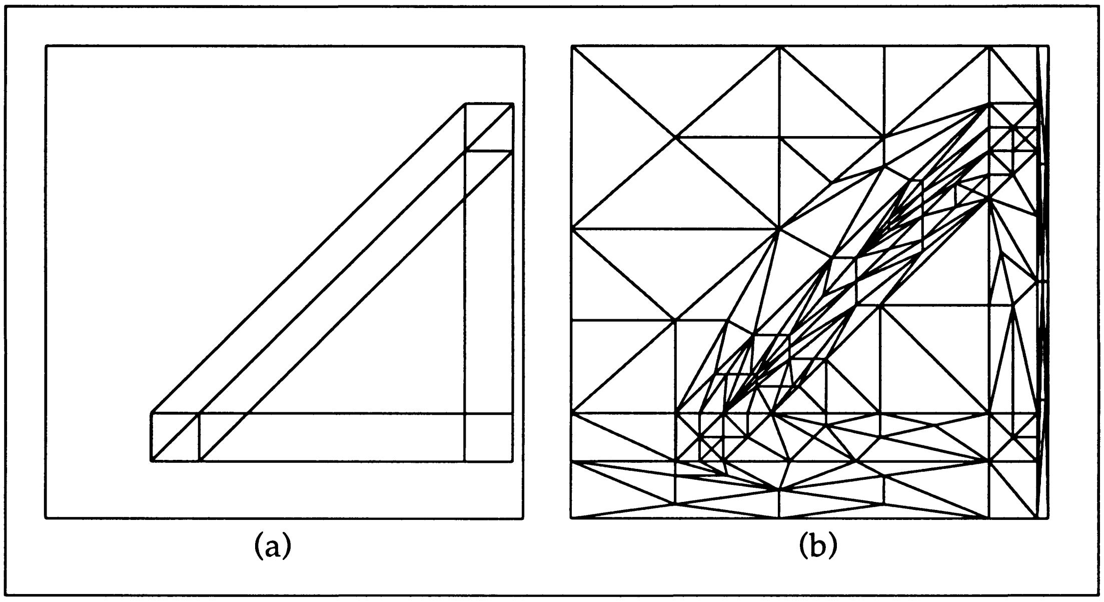


Figure 6.4: Top view of the floor of the simple scene. (a) The critical curves resulting from the interaction of the light source and the triangular obstacle. (b) The resulting discontinuity mesh.

	$\|B\|_2$ $(Wsr^{-1}m^{-2})$	E $(Wsr^{-1}m^{-2})$	e
full view	0.54676	0.00326	0.00597
single surface	0.47820	0.00215	0.00450

Table 6.1: Discontinuity meshing vs. analytical solution. Error measurements for the comparisons shown in Figures 6.5 and 6.6.

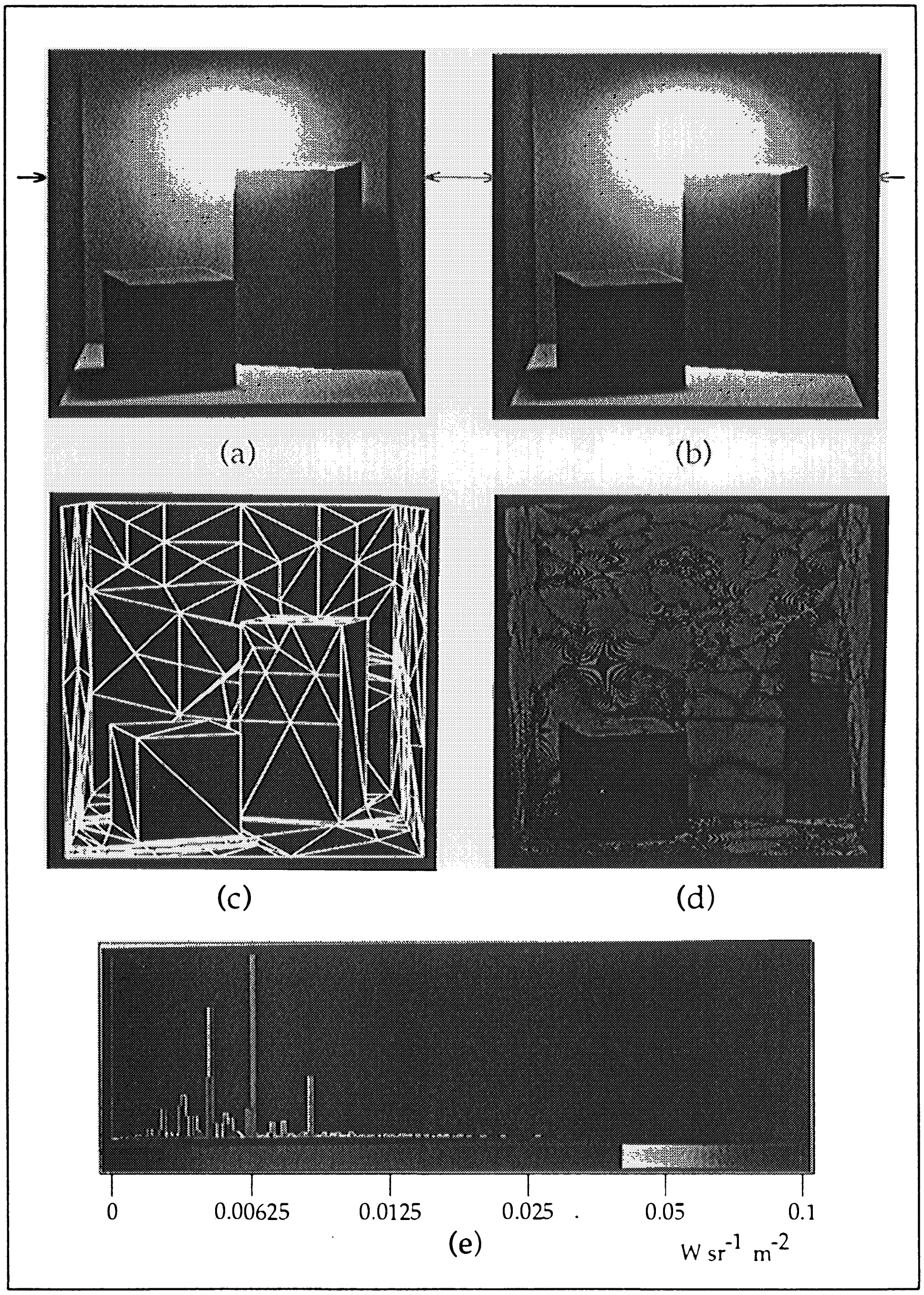


Figure 6.5: Full view. Discontinuity meshing vs. analytical solution. (a) The discontinuity meshing solution. (b) The pixel-by-pixel analytical solution. (c) The mesh underlying the discontinuity meshing solution. (d) Pseudocolored image of the absolute error between discontinuity meshing and analytical solutions. (e) Histogram of the absolute error image. For better resolution, error values are mapped to colors on a logarithmic scale. The maximum and average brightness of the reference image are respectively 4.7219 and 0.4015 $Wsr^{-1}m^{-2}$

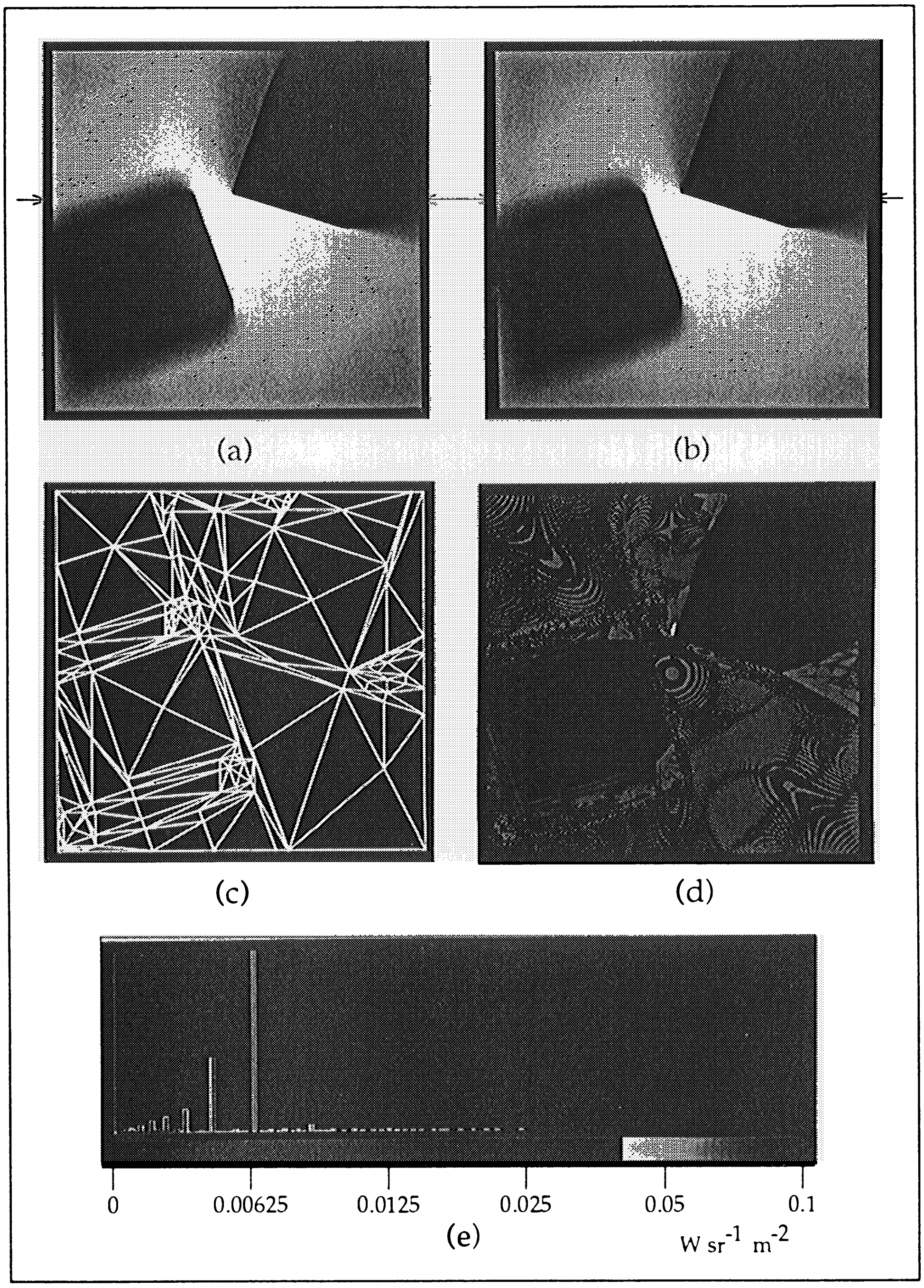
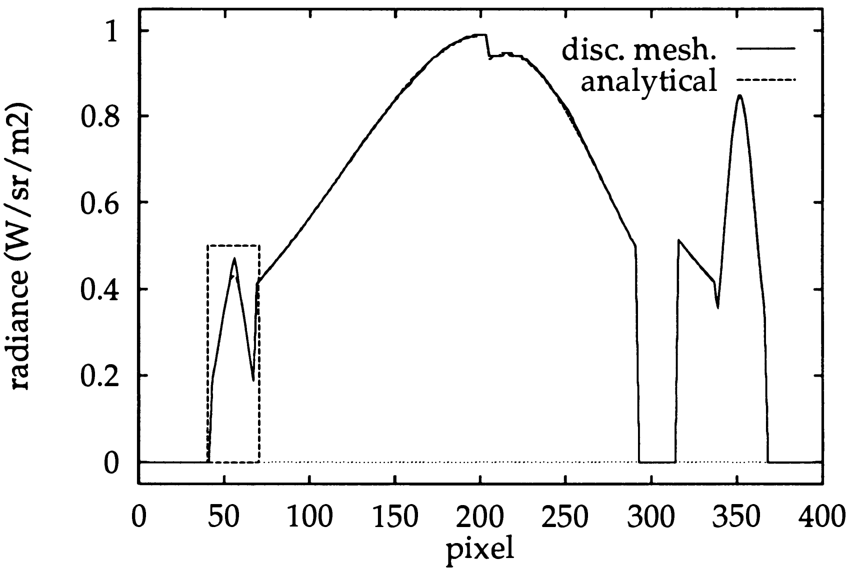
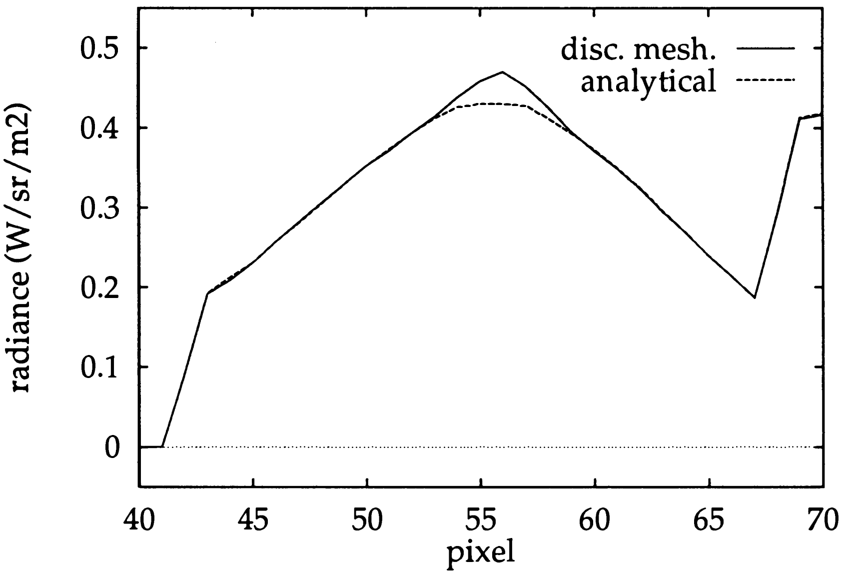


Figure 6.6: Floor surface only. Discontinuity meshing vs. analytical solution. (a) The discontinuity meshing solution. (b) The pixel-by-pixel analytical solution. (c) The mesh underlying the discontinuity meshing solution. (d) Pseudocolored image of the absolute error between discontinuity meshing and analytical solutions. (e) Histogram of the absolute error image. For better resolution, error values are mapped to colors on a logarithmic scale. The maximum and average brightness of the reference image are respectively 0.8809 and 0.3620 $Wsr^{-1}m^{-2}$

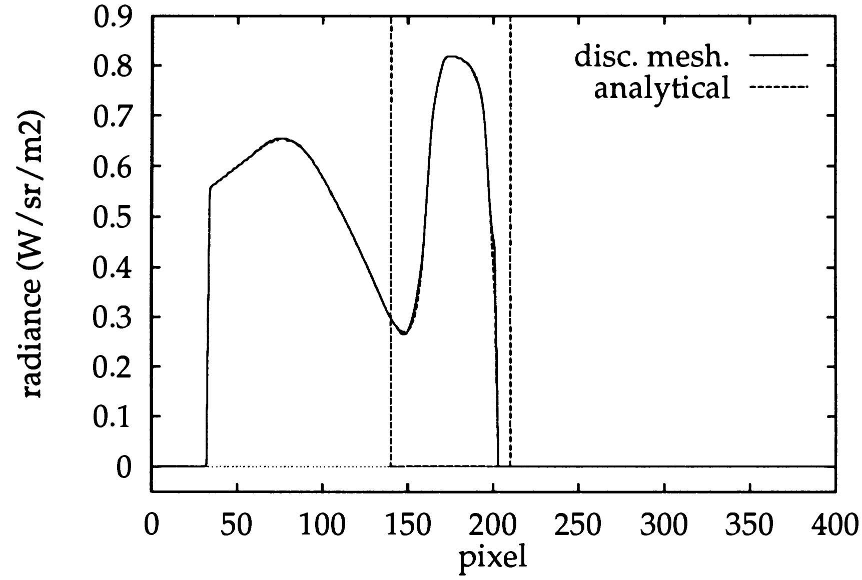


(a)

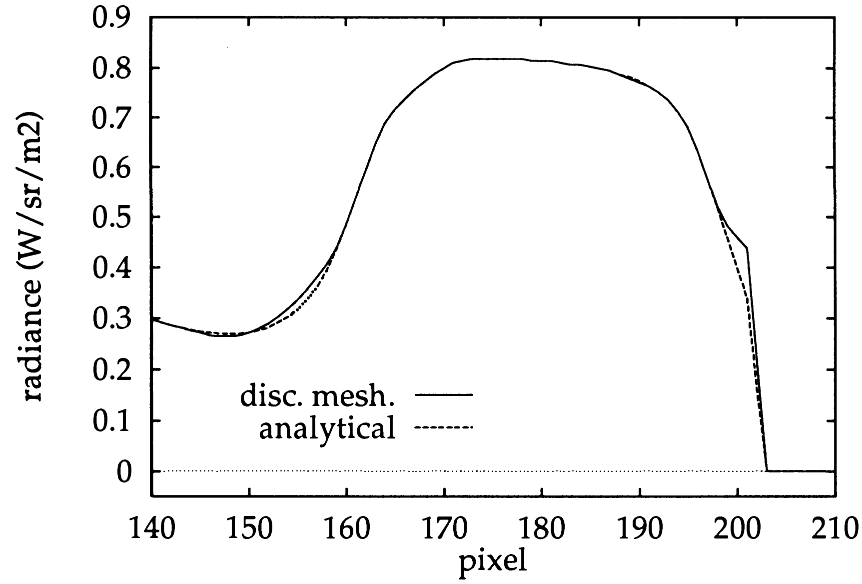


(b)

Figure 6.7: Plot of the radiance function corresponding to the discontinuity meshing and analytical solutions along the scanline containing the largest error for the images in Figure 6.5. (a) The entire scanline. (b) A blow-up of the boxed region, showing the largest error.



(a)



(b)

Figure 6.8: Plot of the radiance function corresponding to the discontinuity meshing and analytical solutions along the scanline containing the largest error for the images in Figure 6.6. (a) The entire scanline. (b) A blow-up of the boxed region, showing a radiance singularity at about pixel 200.

within each of our mesh elements, we would expect our accumulation algorithm to only introduce minor errors.

In order to test the accuracy of the accumulation algorithm, we compare it against a simple, but generally impractical, reference method that combines radiance functions in screen space to completely eliminate any resampling errors. The reference method stores the contribution of each shooting source in a different discontinuity mesh and keeps them all in memory, never trying to combine them into a single mesh. To evaluate the radiance at a given point x on a surface, all these meshes are interrogated and the returned radiance values are summed to yield the accumulated result. To render a radiosity solution for a given view, we repeat this process for each pixel in our image.

Clearly, this method is impractical. The amount of memory required to store all these meshes is prohibitive and the time needed to evaluate the radiance function increases linearly with the number of shooting sources processed. However, this method does not introduce any resampling errors and can therefore be used to produce reference solutions to test the radiance accumulation algorithm described in Section 5.4.

Figure 6.9 shows a comparison of the two methods for a simple environment. Images (a) and (b) show renderings computed respectively from the discontinuity meshing radiosity and reference solutions after twenty progressive refinement iterations.

The resulting mesh for the discontinuity meshing solution is shown in image (c). Image (d), instead, shows a pseudocolored image of the absolute difference between the discontinuity meshing and reference solutions, computed on a

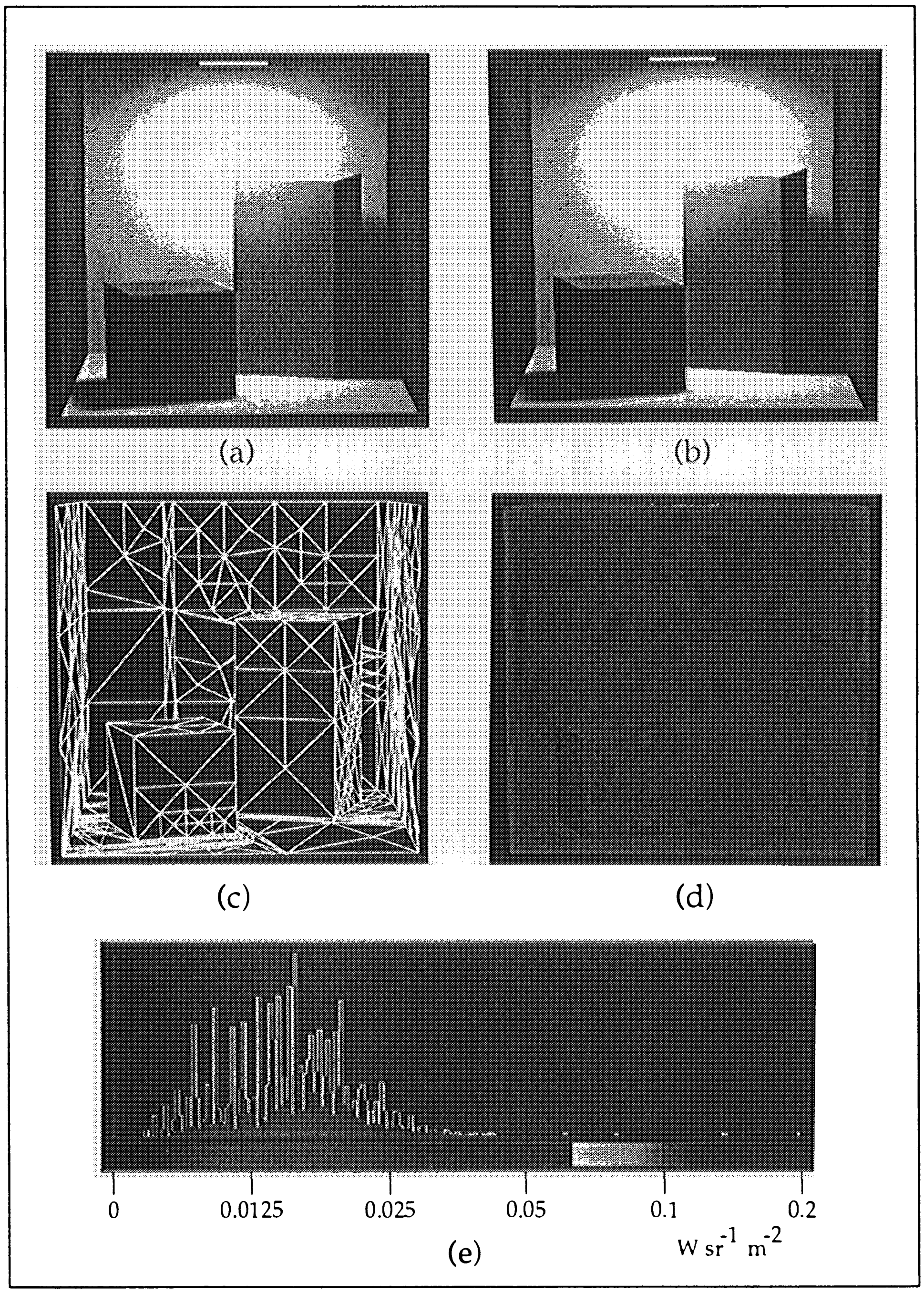


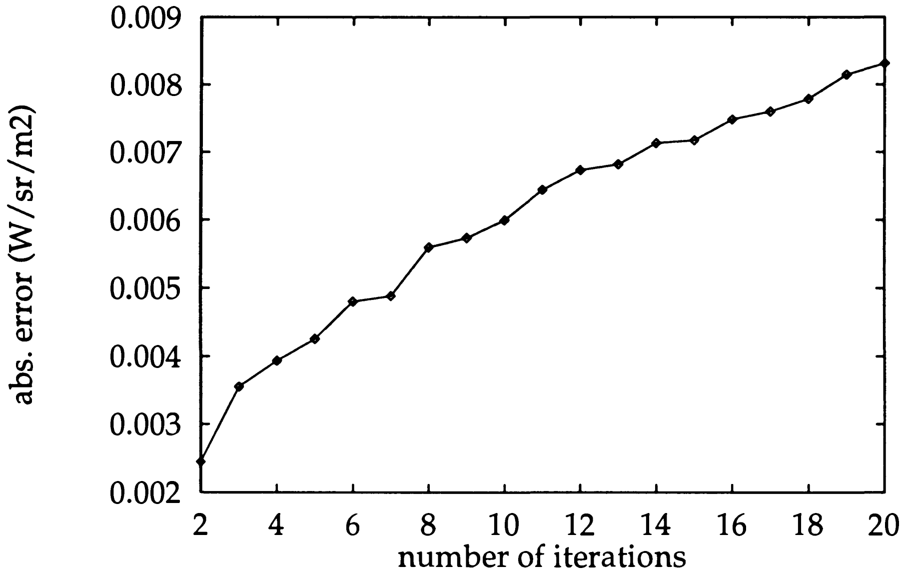
Figure 6.9: Accumulating radiance functions by resampling vs. exact addition. (a) Solution computed by the discontinuity meshing radiosity algorithm. (b) Reference solution computed by adding in each contribution at each pixel. (c) The final mesh for the discontinuity meshing radiosity solution. (d) Pseudocolored image of the absolute difference between discontinuity meshing radiosity and reference solutions. (e) Histogram of the absolute difference image. For better resolution, difference values are mapped to colors on a logarithmic scale. The maximum and average brightness of the reference image are respectively 50.420 and 0.658 $Wsr^{-1}m^{-2}$

pixel-by-pixel basis. Finally, image (e) shows a histogram of the pseudocolored difference image. The histogram is colored to show the mapping between difference values and colors. The largest difference found was 0.2295 $Wsr^{-1}m^{-2}$, while the brightest radiance value in the reference image was 50.42 $Wsr^{-1}m^{-2}$.

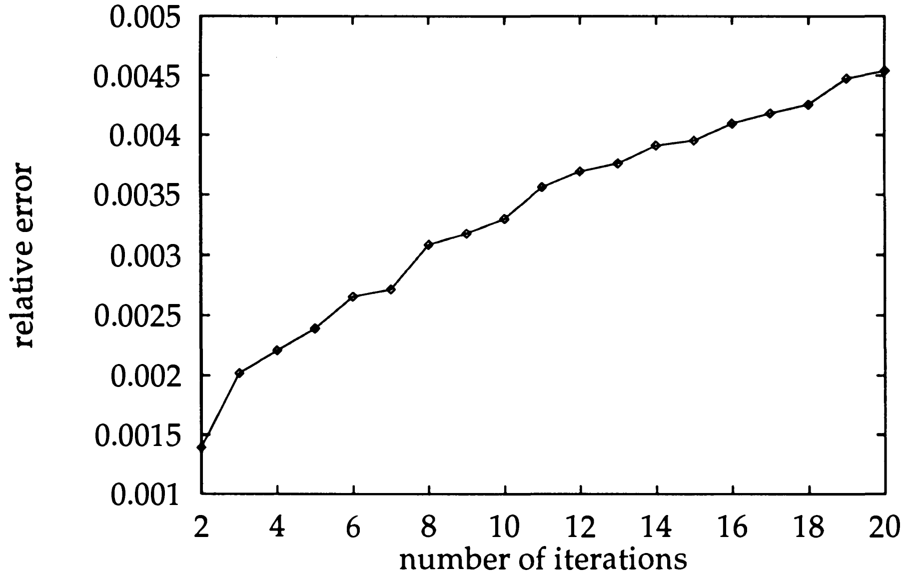
The overall absolute and relative differences for the view shown were computed using Equations (6.2) and (6.3) to yield values of 0.00831 $Wsr^{-1}m^{-2}$ and 0.00454 respectively. For reference, the 2-norm of the reference image was 1.83035 $Wsr^{-1}m^{-2}$. Thus, after twenty iterations the relative difference is less then half of one percent.

Figure 6.10 gives graphs showing the absolute error E and relative error e introduced by the combination method as a function of progressive refinement iterations. These errors were computed by comparing the radiosity solutions generated by the two combination methods for the same number of iterations. As it could be expected, both the absolute and relative errors grow with the number of iterations; however, both their magnitude and the rate of growth are very small.

The error introduced by the combination method, of course, is just one component of the overall error resulting from a progressive refinement radiosity approach. This latter error still decreases rapidly with the number of progressive refinement iterations. Figure 6.11 gives graphs showing the absolute and relative errors resulting from the combined effect of the progressive refinement approach and radiance accumulation algorithm. These errors were computed by comparing the intermediate radiosity solutions generated by the discontinuity meshing radiosity algorithm after each iteration with the final reference solution shown



(a)



(b)

Figure 6.10: Error introduced by the radiance accumulation algorithm as a function of progressive refinement iterations. The absolute error E (a) and the relative error e (b) were computed from a sequence of discontinuity meshing radiosity and reference images (the final pair is shown in Figure 6.9) using Equations (6.2) and (6.3).

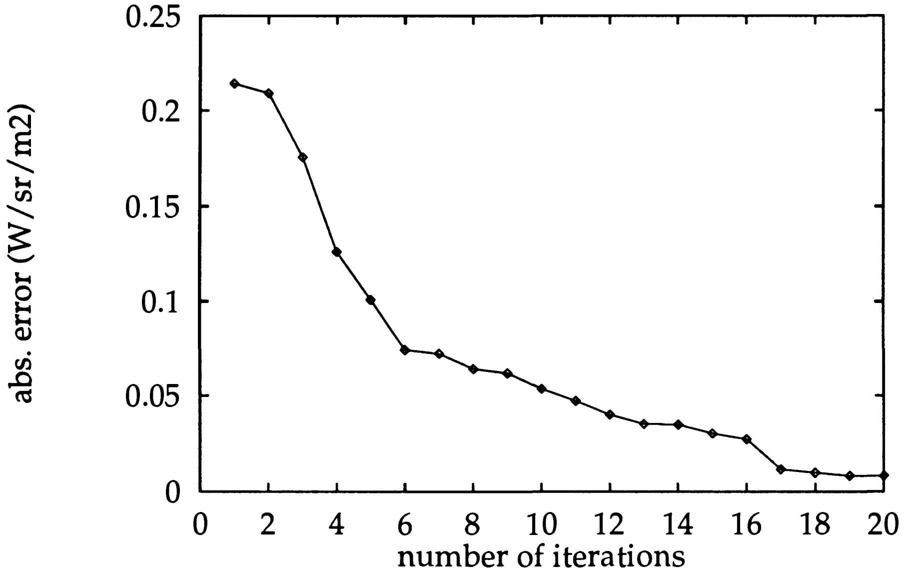
in Figure 6.9b.

Figure 6.12 shows the discontinuity meshes for the first six iterations of our simulation. Each discontinuity mesh corresponds to a different source. The radiance contribution corresponding to each source is also shown in the figure.

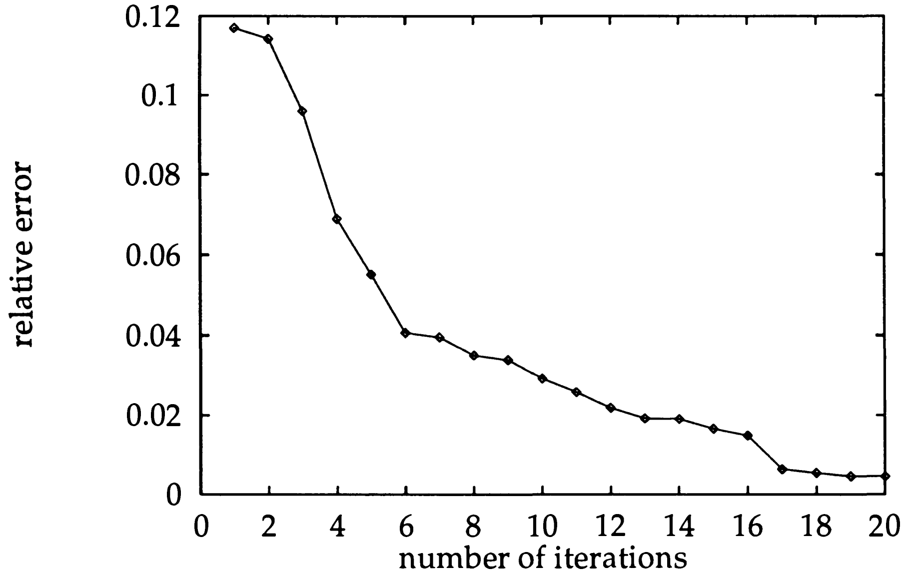
In order to make the contributions visible, a different mapping of radiance values to RGB displayed colors was used than that shown in Figure 6.9. However, the same mapping was used consistently for all of the six images shown.

The sources can be easily inferred from the pictures. Going down the rows, left to right: the primary source in the middle of the ceiling (not shown), the back wall, the floor, the left wall, the right wall, and the top of the right box. Each mesh is significantly different from the others. Each one contains a different set of discontinuity segments. Also, adaptive mesh subdivision was triggered in different areas for each of the discontinuity meshes; in each of the pictures the mesh is denser in those areas were the rate of change of the radiance contribution is higher.

Could combining such different meshes result in an excessive growth of the accumulated mesh? Figure 6.13 shows that this is not generally the case. The graph plots the number of elements in the accumulated mesh as a function of progressive refinement iterations. For reference, the size of the discontinuity mesh capturing the contribution of the current shooting source is also shown for each iteration. As can be observed from the graph, the size of the accumulated mesh grows rapidly during the first iterations, but then it tends to level out and reach an approximate steady state.



(a)



(b)

Figure 6.11: Combined effect of the progressive refinement approach and the radiance accumulation algorithm as a function of progressive refinement iterations. The absolute error E (a) and the relative error e (b) were computed from a sequence of discontinuity meshing radiosity images (the last image is shown in Figure 6.9a) and a reference solution (shown in Figure 6.9b) using Equations (6.2) and (6.3).

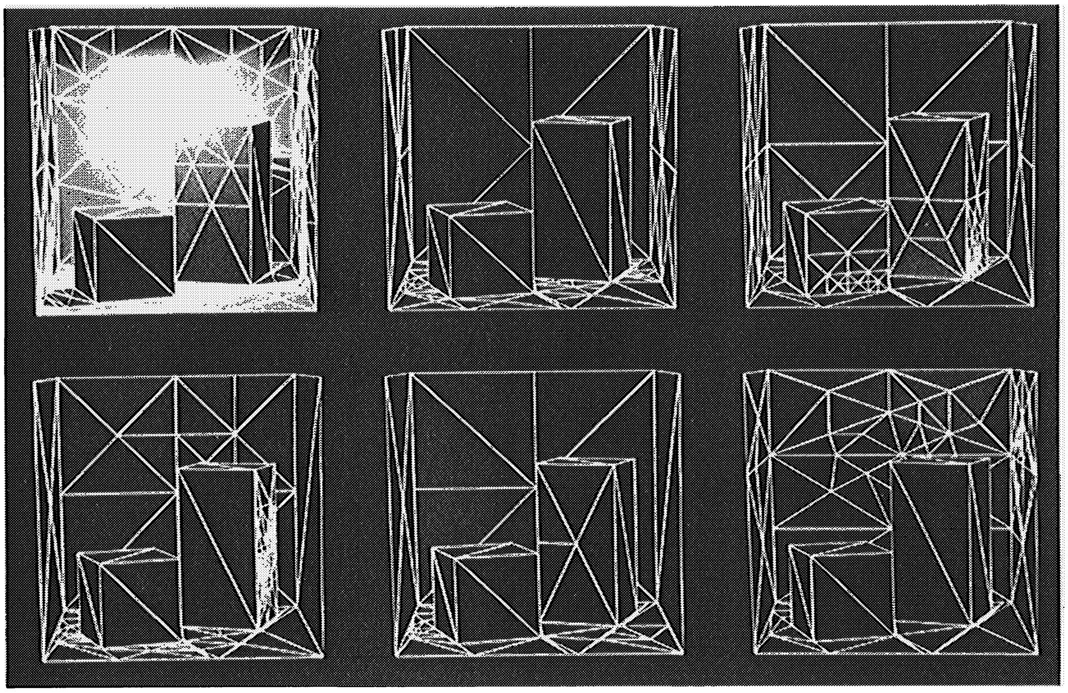


Figure 6.12: Discontinuity meshes and individual contributions for the first six iterations of the discontinuity meshing radiosity solution shown in Figure 6.9. Going down the rows, left to right, the sources are: the primary source in the middle of the ceiling (not shown), the back wall, the floor, the left wall, the right wall, and the top of the right box.

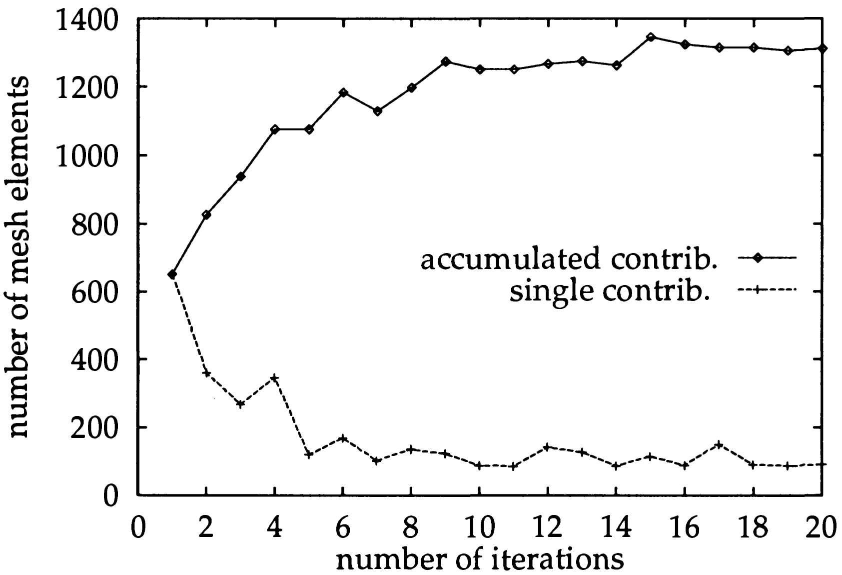


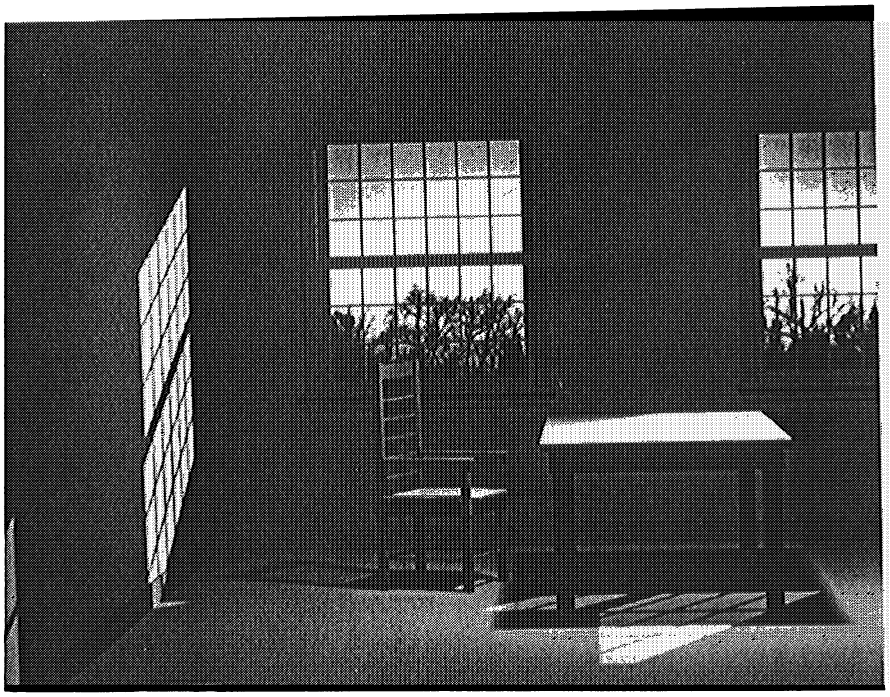
Figure 6.13: Growth in mesh size as a function of progressive refinement iterations for the discontinuity meshing solution shown in Figure 6.9. One curve shows the size of the mesh representing a single source contribution; the other shows the size of the mesh representing the accumulated effect of multiple contributions.

6.4 Discontinuity Meshing Radiosity vs. Conventional Radiosity

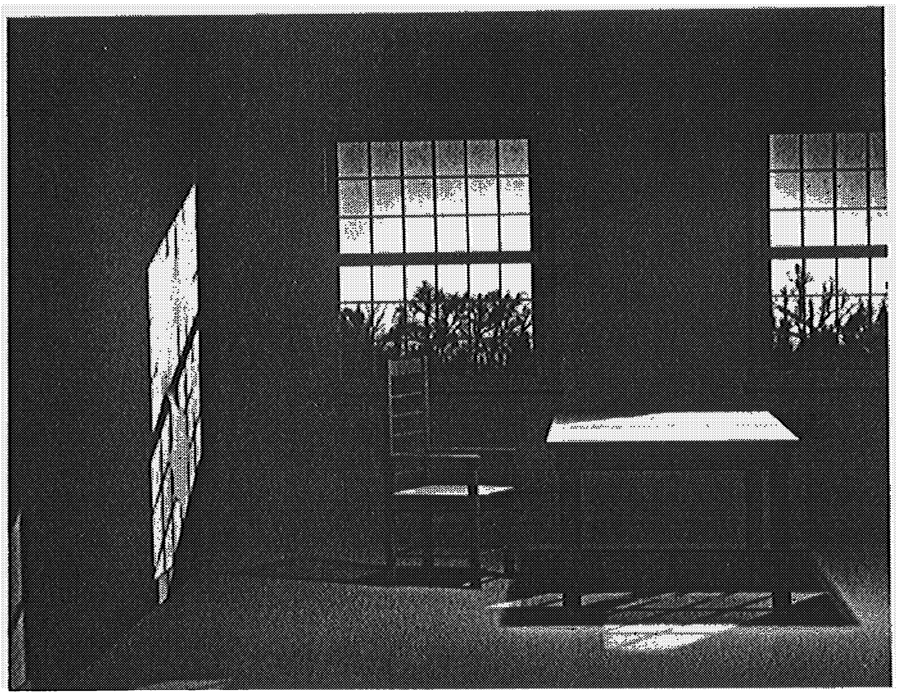
This section presents a qualitative comparison between discontinuity meshing radiosity (DMR) and conventional progressive refinement radiosity (PRR).

A rendering system based on this latter algorithm was implemented during the past several years at the Program of Computer Graphics [TRUM91]. It includes ray-traced form factors [WALL89] and adaptive mesh subdivision. Users have control over parameters such as initial patch size, minimum element size and the tolerance used for adaptive subdivision. Meshing can be controlled both globally and on a surface-by-surface basis.

The model used for the comparison is a simple reproduction of a Victorian-style room consisting of 1382 polygons. The illumination originates from two light sources: the sun, approximated by a polygon placed at a considerable distance, and an area light source at the center of the room's ceiling. Both systems were allowed to perform two iterations, thus the only illumination present in the solutions is the direct illumination from the two light sources. Allowing secondary sources to shoot would have complicated the comparison significantly and would have required software changes to the PRR system. In fact, secondary sources correspond to input polygons in DMR and to smaller mesh patches in PRR; thus, if a secondary source A were processed in DMR, we would have to make sure that all the patches corresponding to surface A in PRR were processed as well. The solution produced by DMR was computed without any need for user intervention. Only an error threshold for adaptive subdivision needed to be specified. The solution resulted in 35,540 elements (more statistics are given in Tables 6.2 and 6.3.) Then, a solution with a similar number of elements (37,846)

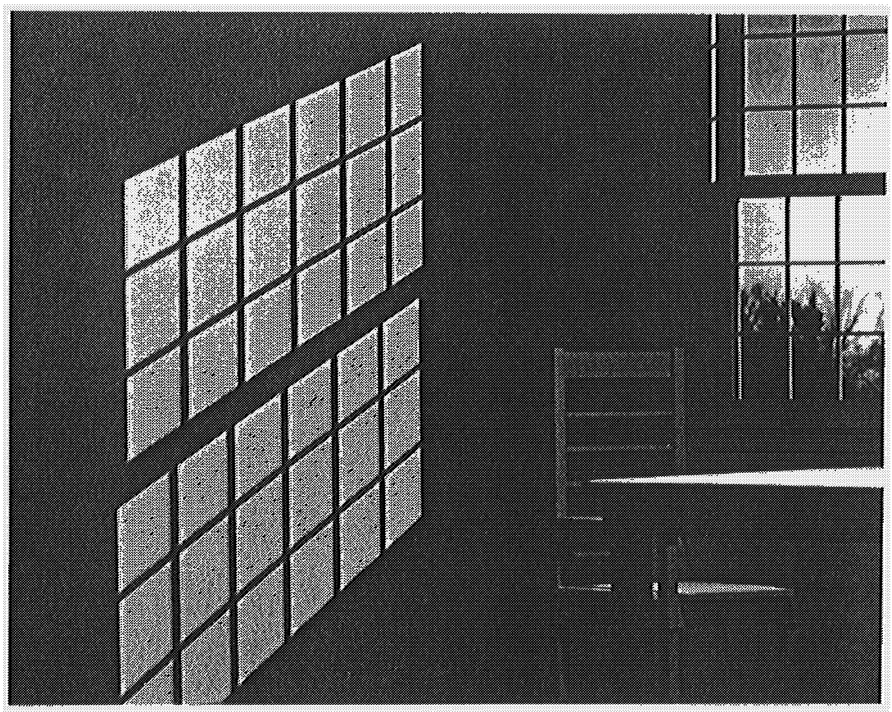


(a)

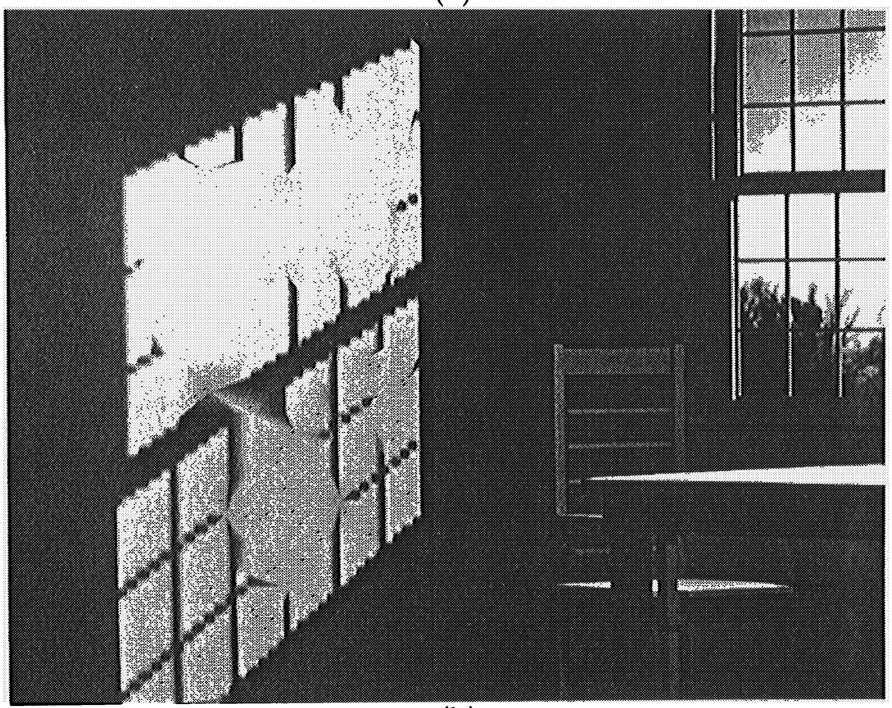


(b)

Figure 6.14: Victorian Room. A ray-traced view of a discontinuity meshing radiosity solution (a) and the corresponding conventional progressive refinement radiosity solution (b).

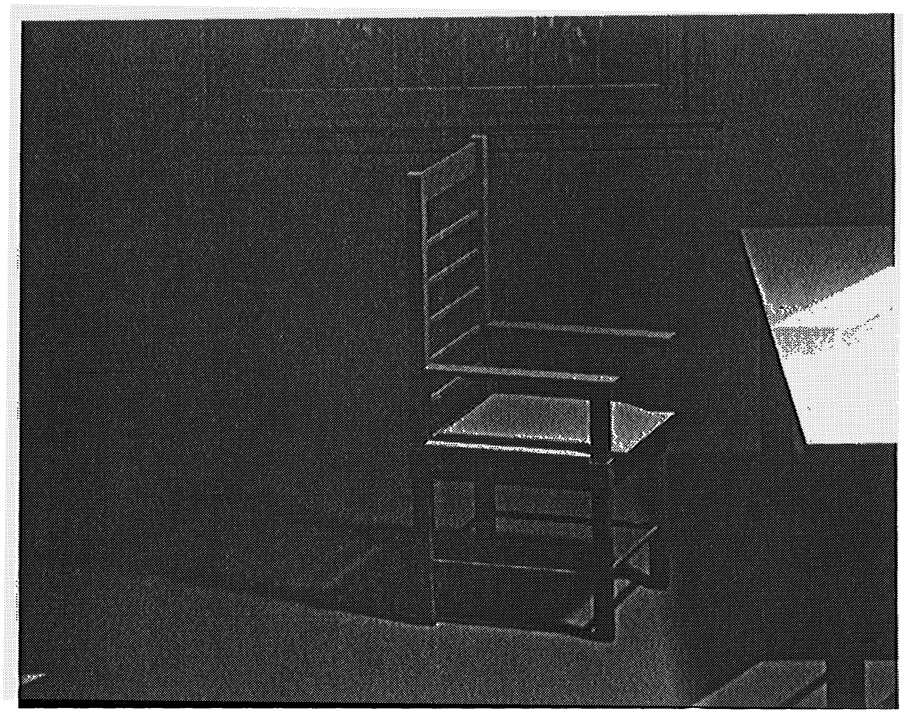


(a)

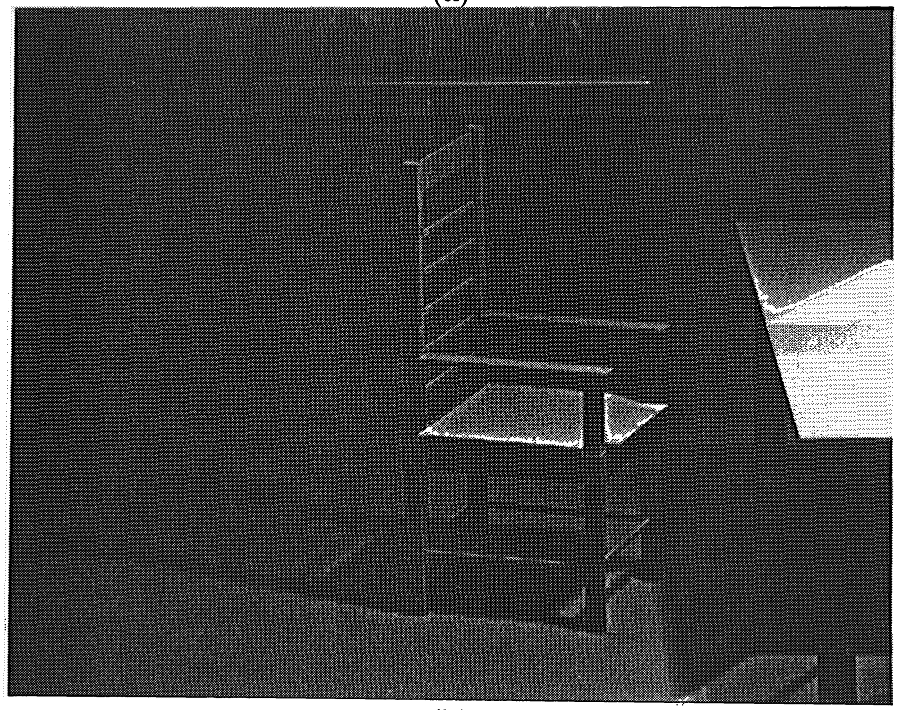


(b)

Figure 6.15: A closer look at the wall. A ray-traced view of a discontinuity meshing radiosity solution (a) and the corresponding conventional progressive refinement radiosity solution (b).



(a)



(b)

Figure 6.16: A closer look at the chair. A ray-traced view of a discontinuity meshing radiosity solution (a) and the corresponding conventional progressive refinement radiosity solution (b).

was produced by the PRR system. This solution was obtained through a process of trial and error: we experimented with various initial meshing densities and minimum element sizes to obtain reasonable quality. To make a fair comparison no surface-by-surface tweaking was performed; only global meshing parameters were used. For accuracy of the illumination computations, 64 rays per form factor were used by the PRR system.

Three images, each with a different view of the model, were rendered from each of the two solutions. The rendering algorithm used was ray-tracing (eye rays only) with adaptive supersampling for anti-aliasing.

A general view of the room is shown in the images in Figure 6.14. Note how DMR captures the thin shadows from the window on the left wall, many of which are entirely missed by PRR. The PRR solution also exhibits some shadow leaks from under the window frames.

The next view (Figure 6.15) is a closer look at the left wall of the room. Note the high quality of the shadows in the image produced from the DMR solution and the realistic softening of the shadows on the wall as we move away from the window. On the other hand, the artifacts in the PRR solution are now even more noticeable. The captured shadows are severely aliased: some appear jagged; others are too wide and blurry.

Finally, Figure 6.16 features a closer view of the chair. Note the high quality of the shadow cast by the chair on the floor in image (a). The shadow is sharp and crisp near the legs of the chair, and it becomes softer farther away. In image (b) this shadow exhibits artifacts that were not visible in the general view of the room. Also note that the shadow cast by the armrest on the seat is captured by

Victorian Room	
Input polygons	1,382
Patches	1,382
BSP tree nodes	3,412
BSP tree polygons	5,406
D^0 edges	2,702
VE events	12,636
EV events	10,532
D^1 discontinuities	1,349
D^2 discontinuities	10,278
Final elements	35,540

Table 6.2: Statistics for the discontinuity meshing solution of the Victorian Room model

Victorian Room		
Task	time	%
Build discont. mesh	11 : 52	36.81
Illumination comp.	17 : 10	53.26
Combining rad. funcs.	02 : 08	6.62
Miscellaneous	01 : 04	3.31
Total	32 : 14	100.00

Table 6.3: Timings for the discontinuity meshing solution of the Victorian Room model. All timings are given in minutes and seconds on an HP720 workstation.

DMR, but is entirely missing in the PRR solution. The fine self shadowing on the chair and the window frame is correctly captured by the new method, yet almost completely absent from the PRR solution. Another important difference is visible in the penumbra cast by the table top on the floor. The intensity changes smoothly in (a), while in (b) it appears to change in several steps. This is due to the use of ray-traced form factors in the PRR system, which computes the visibility of the source from each point of the floor by point sampling, in contrast to the accurate hidden surface removal algorithm employed by the new method.

The sets of figures clearly demonstrate the superiority of the new algorithm in computing solutions of high visual quality, independent of the viewer position.

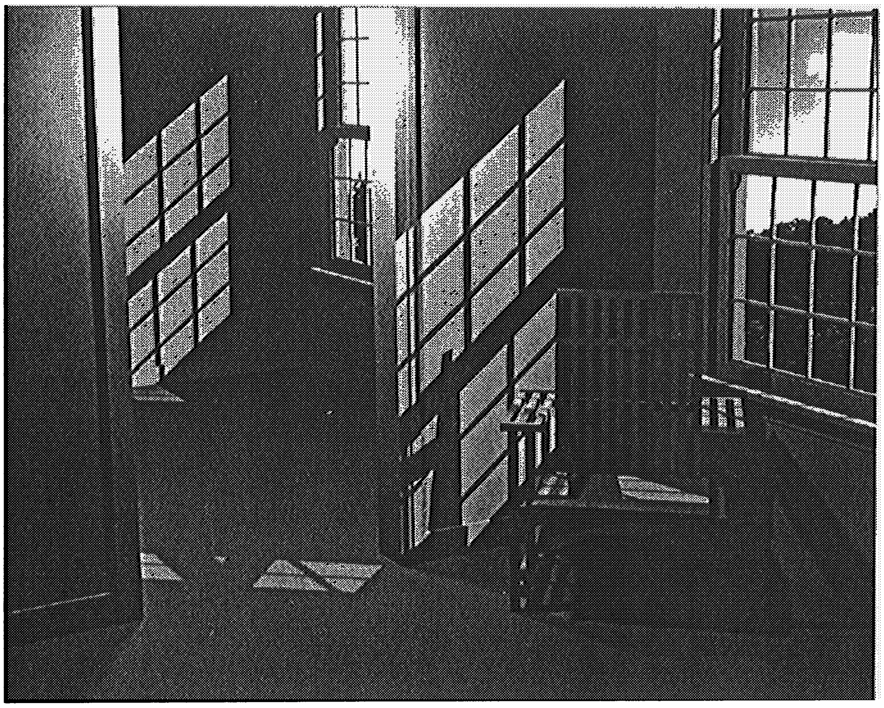
6.5 Statistics

This section examines the performance of the discontinuity meshing radiosity algorithm presented in Chapter 5 when applied to environments of "reasonable" complexity.

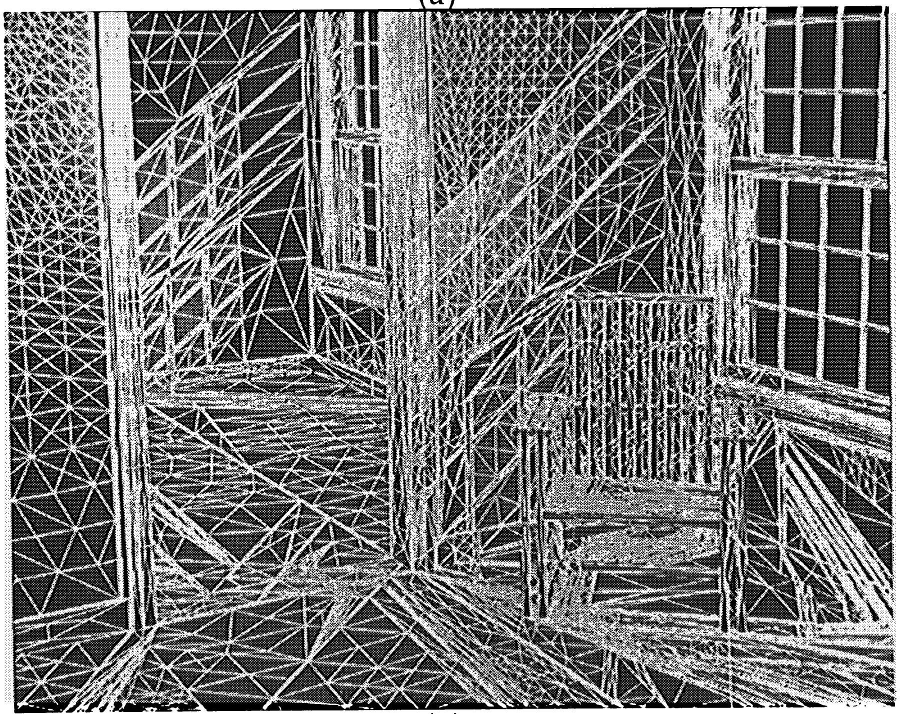
Four different architectural interiors were modeled and fed to the discontinuity meshing radiosity system to generate diffuse global illumination solutions. Ray-traced renderings of these solutions are shown in Figures 6.17, 6.18, 6.20, and 6.21 along with their corresponding discontinuity meshes.

Table 6.4 gives some statistics generated during the radiosity simulations for the four models. Reported are the size of the input, the size of the BSP tree used to sort the surfaces in the environment, the number of visual events generated, the number of resulting discontinuity segments, and the size of final mesh.

Two important observations can be made from the results shown in the ta-



(a)



(b)

Figure 6.17: White Room. (a) A ray-traced view of a discontinuity meshing radiosity solution. (b) The corresponding discontinuity mesh.

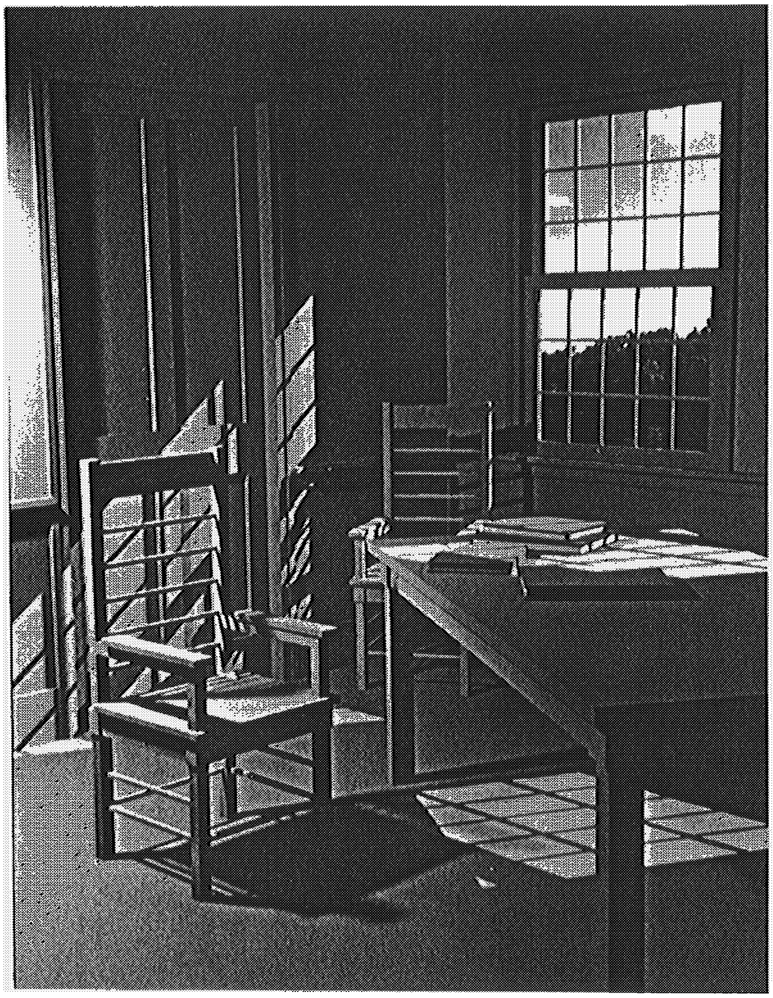


Figure 6.18: Green Room. A ray-traced view of a discontinuity meshing radiosity solution.

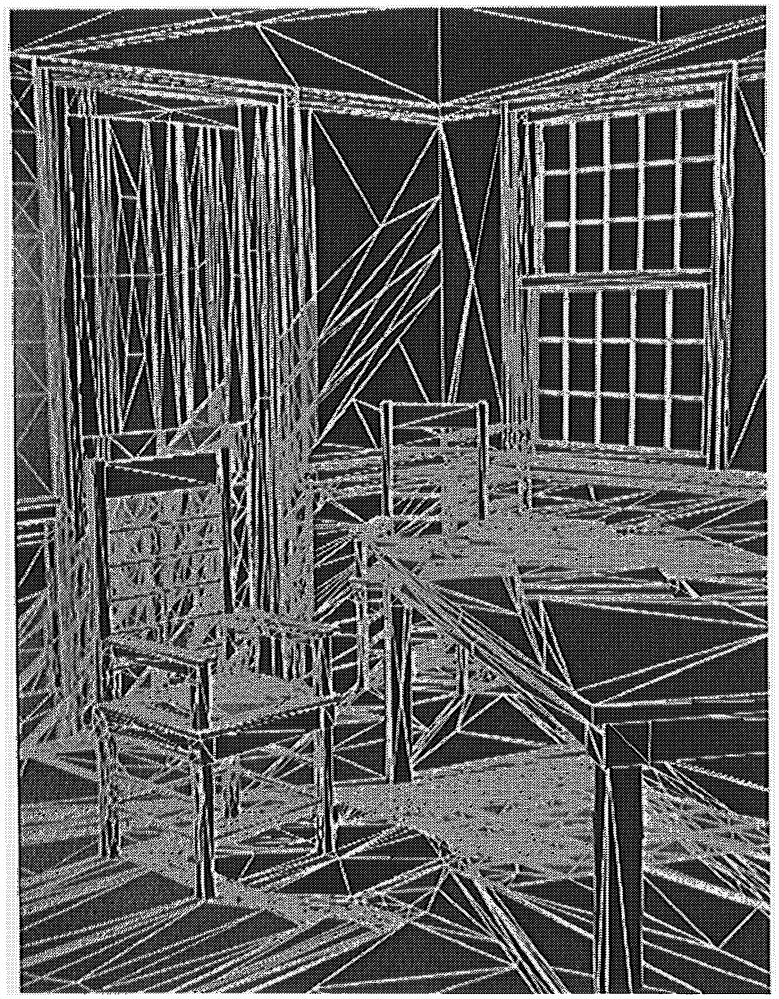
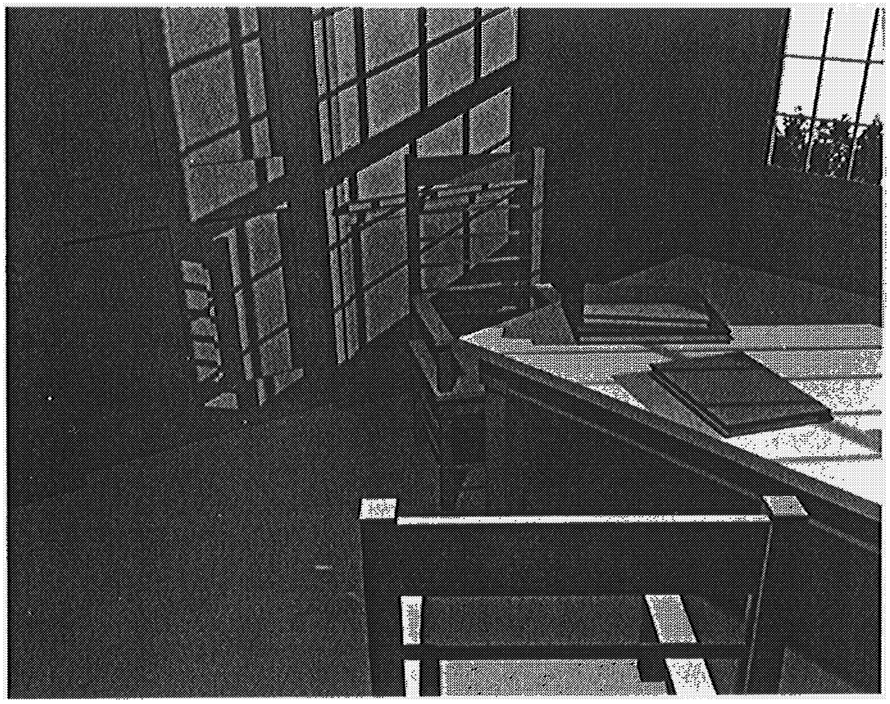
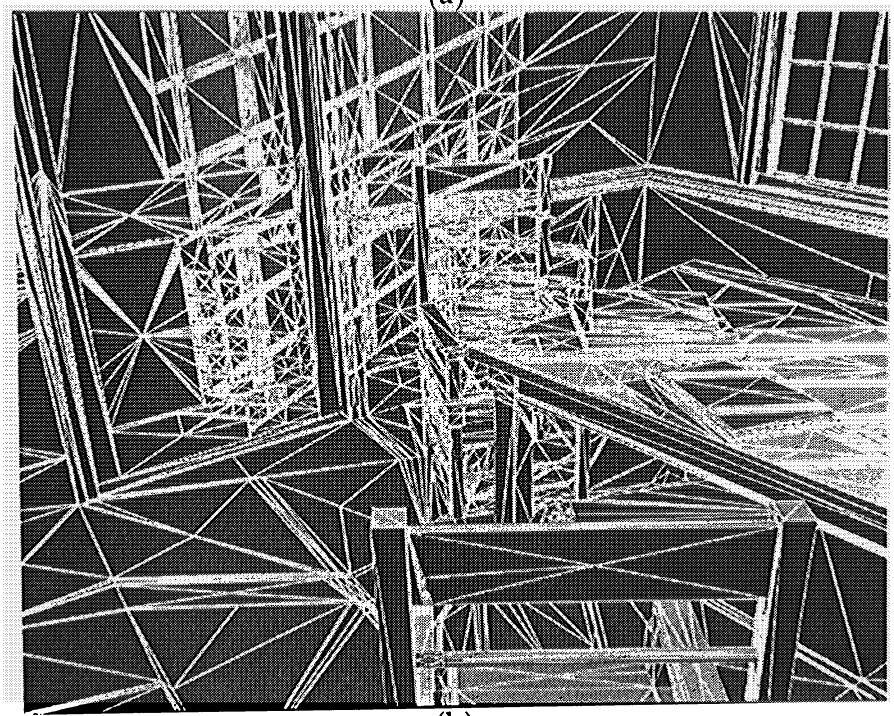


Figure 6.19: Green Room. The final discontinuity mesh.

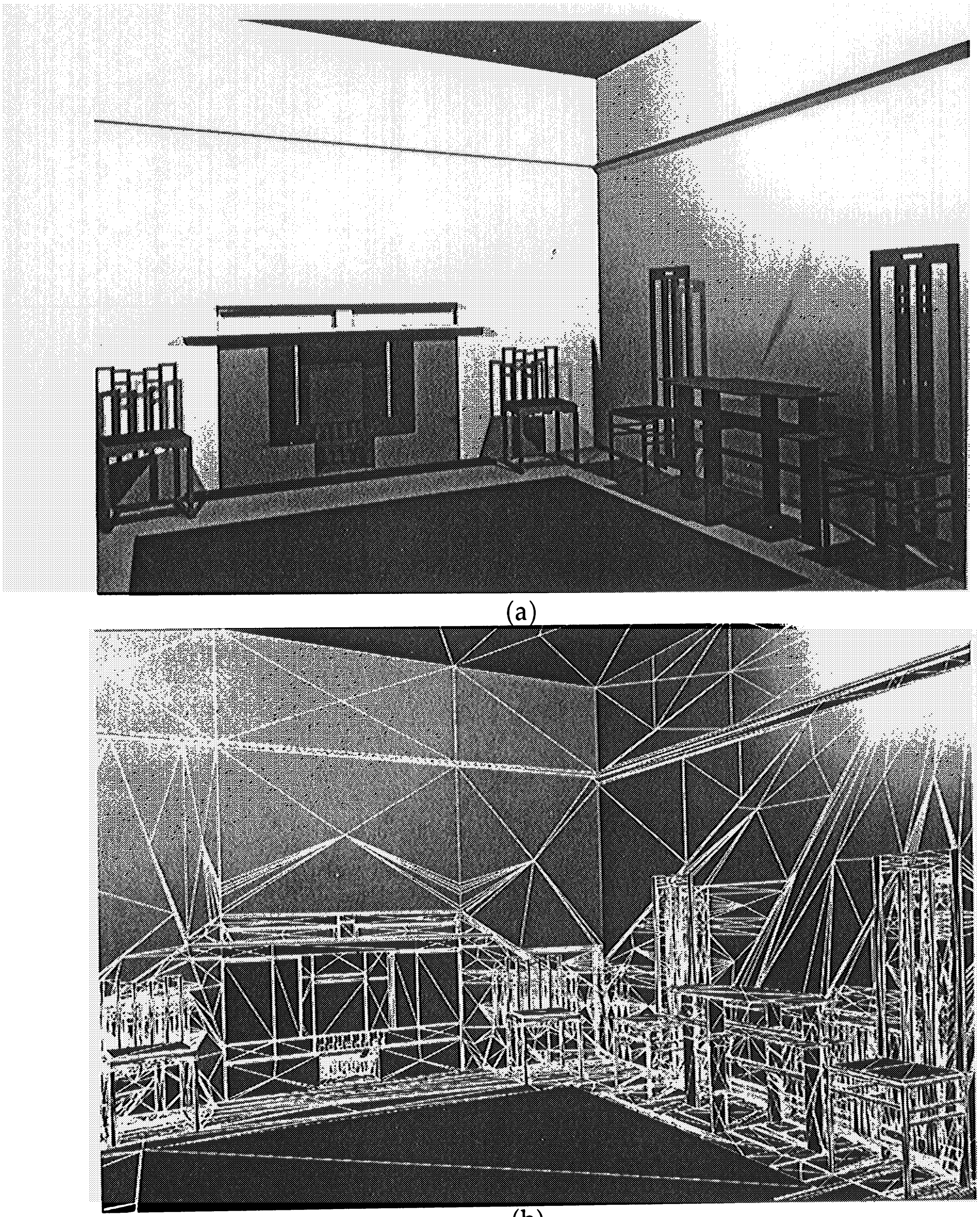


(a)



(b)

Figure 6.20: Pink Room. (a) A ray-traced view of a discontinuity meshing radiosity solution. (b) The corresponding discontinuity mesh.



(a)

(b)

Figure 6.21: Mackintosh Room. (a) A ray-traced view of a discontinuity meshing radiosity solution. (b) The corresponding discontinuity mesh.

ble. First, the ratio of BSP tree nodes and polygons to input polygons is very high. In fact, the algorithm used to build the BSP tree is very simplistic. Any improvements capable of reducing these ratios would immediately improve the performance of the entire discontinuity meshing radiosity system since it would reduce the number of BSP polygons that must be intersected with visual events when the latter are traced through the environment.

Second, although in principle there could be $O(n)$ discontinuity segments for each visual event, in practice there seem to be many, many fewer. This is due to the fact that in any realistic environment many patches are not visible to each other.

Table 6.5 gives timings for the four test environments. The discontinuity meshing radiosity algorithm is broken down into its constituent modules and computation times are reported for each module.

Most of the time is taken up by the illumination computations, i.e. by computing the light energy contribution from a source to a point on a receiving surface. This is due to the compute-intensive visibility calculations used at that step. The cost of the illumination computations could be reduced using the techniques discussed in Section 5.5.2 at the cost of some accuracy. The results reported in Table 6.5, however, were computed using the more expensive approach.

Building the discontinuity mesh for each of the shooting sources amounts to approximately one quarter to one third of the total computation time. The higher percentage taken in the case of the Mackintosh Room environment is due to the particularly large ratio of BSP nodes and BSP polygons to input polygons (see Table 6.4.) A better, more balanced, BSP tree could substantially reduce the

	Models			
	White Room	Green Room	Pink Room	Mackintosh
Input polygons	1,311	1,688	1,688	1,154
Patches	1,311	1,688	1,688	1,154
BSP tree nodes	1,760	3,885	5,004	8,197
BSP tree polygons	3,103	5,522	7,303	9,500
D^0 edges	2,561	3,092	3,092	1,879
VE events	16,834	15,238	15,840	9,992
EV events	13,952	13,456	13,604	8,816
D^1 discontinuities	6,707	1,482	0	517
D^2 discontinuities	7,211	14,528	8,952	22,166
Final elements	69,139	55,348	59,947	48,187

Table 6.4: Statistics for the discontinuity meshing solution of the White Room, Green Room, Pink Room, and Mackintosh Room models

	Models			
	White Room		Green Room	
Task	time	%	time	%
Build discont. mesh	07 : 20	24.33	13 : 50	32.46
Illumination comp.	16 : 09	53.60	24 : 30	57.49
Combining rad. funcs.	05 : 20	17.70	02 : 50	6.65
Miscellaneous	01 : 19	4.37	01 : 27	3.40
Total	30 : 08	100.00	42 : 37	100.00

	Models			
	Pink Room		Mackintosh	
Task	time	%	time	%
Build discont. mesh	10 : 43	29.55	14 : 48	46.74
Illumination comp.	22 : 26	61.86	12 : 50	40.53
Combining rad. funcs.	01 : 49	5.01	02 : 54	9.16
Miscellaneous	01 : 18	3.58	01 : 08	3.57
Total	36 : 16	100.00	31 : 40	100.00

Table 6.5: Timings for the discontinuity meshing solution of the White Room, Green Room, Pink Room, and Mackintosh Room models. All timings are given in minutes and seconds on an HP720 workstation.

computation time for this module.

Finally, the algorithm used to combine the individual radiance contribution into a single, accumulated, radiance function takes a smaller, but yet non-negligible portion of the total computation time.

Chapter 7

Summary and Conclusion

7.1 Summary and Main Contributions

In the ten years since their introduction to the computer graphics community [GORA84], radiosity methods have produced some of the most realistic-looking computer generated images to date. Furthermore, radiosity is not merely an image synthesis technique: it is a physically-based simulation of light energy transfer that can be used to predict the value of actual physically measurable quantities.

The ability to simulate global illumination, as well as the possibility of generating interactive "walk-throughs" from the view-independent solutions, makes radiosity methods attractive for a wide variety of applications, ranging from architectural design and lighting analysis to remote sensing and flight simulators.

Despite all the areas that could benefit from radiosity, this method has not yet met with widespread acceptance. This is not surprising since with current radiosity systems users have to go through a very tedious and time consuming iterative process in order to generate a surface mesh capable of producing a global illumination solution to the desired level of detail. Without very careful user intervention, these meshes are prone to generating a wide range of annoying visual artifacts. In fact, as many researchers have pointed out, poor meshes are actually responsible for most of the visual artifacts appearing in radiosity images.

Clearly, new automatic meshing techniques are needed to generate high quality radiosity solutions with the degree of reliability necessary for practical applications. This thesis has investigated the roots of the problems with current meshing techniques and proposed a new approach to meshing that is completely automatic and that, used within a progressive refinement radiosity system, has generated radiosity solutions of previously unattained quality.

In order to gain new insight into the problem of generating highly accurate radiosity solutions and to better understand the flaws and limitations of the meshing techniques used in conventional radiosity systems, we have taken a step back from the traditional finite element formulation of radiosity [GORA84] (see Equation (2.5).) Lifting the assumption that radiosity is constant across each element results in the more general diffuse rendering equation (Equation (2.3)), a formulation of radiosity as a system of integral equations in the style of the rendering equation first proposed by Kajiya [KAJI86]. Each equation describes the radiance function over a surface s_i as the sum of simpler radiance functions, each one representing the contribution of a single surface in the environment (see Equation (2.7).)

An analysis of the radiance functions described by this last equation has shown how umbra and penumbra boundaries as well as other sharp changes in illumination within the penumbra regions actually correspond to discontinuities in the radiance function and its derivatives. The shape, location, and order of these discontinuities is determined by the interaction of the light sources and obstacles in the environment. The results of this analysis have led to the concept of discontinuity meshing, whereby shadow boundaries and other sharp variations in the illumination across a surface are captured by explicitly representing the corresponding discontinuities as boundaries in the mesh.

Based on this concept, we have designed and implemented a discontinuity meshing algorithm capable of producing accurate solutions to the radiosity problem for three-dimensional polyhedral environments illuminated by area light sources. The algorithm stores a radiance function as a set of sample points organized in a discontinuity meshing tree. This data structure allows discontinuity segments to be inserted into the mesh incrementally while maintaining mesh conformity and provides fast access to the radiance values. Piecewise quadratic interpolation is used in conjunction with discontinuity meshing to reconstruct a smooth radiance function while preserving discontinuities where appropriate.

Finally, in Chapter 6, we examined the accuracy of the new algorithm by comparing it against a photograph of a physical environment, an analytical solution, and a conventional, yet state-of-the-art, radiosity system, and measured its performance on architectural models of medium complexity.

The results have been extremely promising. The new discontinuity meshing algorithm improves upon previous techniques in many respects. The major advantages are described below.

Mesh generation

As opposed to conventional radiosity meshing algorithms that require specifying an initial sampling density, the new discontinuity meshing algorithm is completely automatic. The resulting mesh is guaranteed to capture all the important shadow boundaries as well as other sharp illumination details resulting from discontinuities in the penumbra areas.

Identifying the location of these trouble areas allows the discontinuity meshing algorithm to distribute sample points very effectively. In fact, a small set of sample points suffices to capture fine illumination details that would have required a much larger set with more conventional radiosity methods. Furthermore, while the visual quality of shadow boundaries in conventional radiosity solutions is tightly dependent on the projected size of the mesh elements onto the image plane, discontinuity meshing solutions exhibit a very high degree of view-independence.

While conventional radiosity systems maintain a single mesh (possibly refining it by adaptive subdivision) throughout the entire simulation, out new algorithm creates a fresh discontinuity mesh at each progressive refinement iteration. Within the progressive refinement radiosity framework, this strategy has shown to require less radiance samples for a given accuracy than the more traditional approach. The possible loss of accuracy introduced by the resampling technique used to accumulate radiance functions has proven to be negligible. As we reported in Section 6.2, in fact, the errors introduced by this scheme are very small (less than half of one percent in our test.)

Radiance reconstruction

The simple piecewise quadratic interpolation scheme used to reconstruct the radiance function has proved to be superior to the more popular bilinear interpolation and Gouraud shading techniques. For a given visual accuracy, the quadratic formulation requires less elements than the bilinear approach and in most cases eliminates unwanted Mach banding without incurring the computational costs and complexity of more sophisticated methods [SALE92].

General radiance distributions

Removing the assumption that elements must have constant radiosity from the radiosity formulation allows for the correct treatment of light sources of non-uniform radiance distribution (generally the case for secondary sources.) This eliminates any restriction on patch size resulting in more accurate and reliable physical simulations, better efficiency, and reduced need for manual intervention.

7.2 Further Research

Hopefully, the algorithm presented in this thesis and the results generated should convince the reader of the power of the discontinuity meshing approach to radiosity image synthesis. While the concept and ideas are fundamentally correct, however, some implementation issues deserve more attention.

7.2.1 Robustness

Tracing visual events through the environment and building the discontinuity mesh requires performing many intersection and clipping operations. Since all the operations are carried out in finite precision, numerical accuracy becomes an issue. Different visual events, for example, may sometimes result in nearly coincident discontinuity segments; whether these segments are classified as parallel, coincident, or intersecting, may make the difference between an accurate simulation and one that contains streaks of light and other annoying visual artifacts.

Small errors in the arithmetic computations not only result in numerical errors, but can propagate to cause inconsistencies in our data structures. The topological data structures used to represent the discontinuity meshes are particularly vulnerable to this kind of errors. Special care has to be taken to maintain consistency at all times if frustrating program crashes are to be avoided.

In our implementation, we alleviated these problems significantly by verifying and enforcing consistency after numerically sensitive calculations and by using "representatives" and Epsilon-Geometry [SALE91].

However, despite these precautions, the current implementation of the discontinuity meshing algorithm is not completely reliable. Occasionally, small inconsistencies occur that result in visual artifacts in the displayed radiosity solution. Clearly, a more robust solution is needed before the algorithm can be used in practical applications.

7.2.2 Data Structures

The discontinuity meshing radiosity system we have implemented makes extensive use of BSP trees. A 3D BSP tree is used to sort the input polygons so that visual events can be traced through the environment very efficiently. A 2D BSP tree is used as part of the DM tree data structure to represent the discontinuity mesh on a surface, support incremental updates of the mesh, and allow for efficient point location.

These were just two of several possible implementation choices and their selection is not an inherent characteristic of discontinuity meshing algorithms. In retrospect, we would recommend the choice of a 3D BSP tree for tracing visual events, but would suggest investigating data structures other than the 2D BSP tree for the surface mesh. Use of a 2D BSP tree in this context can often lead to elements with very bad aspect ratios. Our adaptive subdivision scheme can improve these aspect ratios when applied to these elements. However, the refinement step is driven by energy considerations rather than aspect ratios, and offers no guarantee that all badly shaped elements will be corrected.

The quality of the discontinuity meshes is also affected by the order in which discontinuity segments are inserted in the 2D BSP tree. Rather than inserting these segments on the fly as described in Section 4.3, it might be better to collect them all and then choose an insertion order based on some heuristics, e.g. trying to split a node so as yields almost equal areas [LISC93].

The discontinuity mesh, however, could be represented by an altogether different data structure. Heckbert [HECK92a] uses constrained Delauney triangulation in his 3D implementation of discontinuity meshing. This triangulation technique is also used by Lischinski, Tampieri, and Greenberg [LISC93] to yield remarkably well-shaped discontinuity meshes.

7.2.3 Performance and Flexibility

The new discontinuity meshing algorithm is remarkably accurate when computing the direct illumination from a small number of primary light sources. Computing the illumination due to secondary light sources, however, has proven to be particularly expensive. The secondary sources are typically larger than primary light sources and their radiance functions may have several discontinuities; as a consequence, many more visual events are associated with a secondary light source than are used for a primary light source. This can result in finer meshes and can require higher numbers of radiance samples than are needed for primary light sources. Clearly, these computations are carried out to an unnecessary accuracy since secondary sources are typically much weaker than primary light sources. Furthermore, a significant effort is spent propagating the discontinuities that cross secondary light sources; since this process increases the order of discontinuity, however, their effect on the receiving surfaces is often hardly noticeable.

The techniques discussed in Section 5.5 were developed to speed up the computation of secondary light source contributions at the cost of some accuracy. Although these techniques have proved useful in computing radiosity solutions of moderately complex environments, the benefits accrued by trading off accuracy for speed are limited. Many energy transfers are still computed with unnecessarily high accuracy, thus preventing use of the system with more complex models.

The flexibility of the discontinuity meshing algorithm is further limited by the progressive refinement radiosity framework in which it was implemented. Combining discontinuity meshing with hierarchical radiosity [LISC93] has recently shown how discontinuity meshing techniques can be successfully applied to the solution of radiosity problems of reasonable complexity, yielding results of unprecedented quality while at the same time improving the performance of the original hierarchical radiosity approach [HANR91b].

7.2.4 Solution Accuracy and Perceptual Issues

As mentioned in the preceding section, some of the computations associated with the discontinuity meshing algorithm are carried out to unnecessarily high accuracy. Ideally, if the goal of the radiosity simulation is to produce photorealistic results, no computing time and memory should be spent refining a solution unless doing so resulted in visible improvements in the generated images.

Predicting whether a certain operation, e.g. propagating a discontinuity segment or subdividing a mesh element, will make a visual difference is very difficult since it would necessarily involve perceptual issues that are hard to quantify. Our ability to perceive a radiance discontinuity as a Mach band, for example, depends not only on the order of the discontinuity, but also on the brightness of the radiance function in the neighborhood of the discontinuity, its rate of change, its color, and the position of the observer.

Developing a perceptual metric capable of assessing the visual impact of a given illumination detail, e.g. a shadow or a radiance discontinuity, however, could potentially result in very significant computational savings. In fact, any rendering algorithm that allows for trade-offs between computational resources and solution accuracy would greatly benefit from such a metric.

7.2.5 Extensions

The discontinuity meshing radiosity algorithm presented in this thesis is limited to ideal diffuse reflectance distributions and polyhedral environments. It is only natural, then, to think of ways to extending it to general reflectance distributions and curved geometries.

General reflectance distributions

The conventional progressive refinement radiosity algorithm was initially extended to integrate specular and diffuse reflection [WALL87,SILL89] and later modified to accept general reflectance distributions [SILL91]. Augmenting these algorithms with discontinuity meshing can be done using the same techniques presented in this thesis since the location and order of the radiance discontinuities are generally not affected by the shape of the reflectance distributions. Specular spikes in the bidirectional reflectance distribution, though, may result in discontinuities on a receiving surface that would have to be accounted for using different techniques.

Caustics, i.e. specular to diffuse energy transfers, also need special attention. In order to simulate the transfer of light from a light source to a receiving surface through an intermediate specularly reflective surface, radiosity algorithms use the concept of "virtual source." A virtual source is created by reflecting the original source about the mirror surface, so that the indirect transfer from the original source to the receiving surface can be computed as the direct transfer from the original source, through the mirror, to the receiving surface. The resulting radiance contribution may exhibit discontinuities similar to the more conventional diffuse to diffuse transfers. These discontinuities are computed by tracing visual events involving the virtual source and treating the mirror surface as a hole, or portal [TELL92], into the environment.

If the reflecting surface is not a perfect mirror, however, the boundaries of the caustic on the receiving surface will be blurry. In this case, computing the location of the resulting discontinuities using analytical methods may not be possible; in fact, it is not even clear whether the caustic causes discontinuities and, if it does, what their order should be.

Curved geometries

Extending discontinuity meshing algorithms to curved geometries appears to be a difficult problem. Although the concept of discontinuity meshing is not limited to planar geometries, the algorithms we used to trace visual events and locate critical curves, and the data structures we used to represent discontinuity meshes are only applicable to polyhedral environments.

Designing a discontinuity meshing algorithm for curved surfaces requires solutions to non-trivial problems. Curved surfaces interact in very complex ways to generate visual events. Computing critical curves from these visual events requires intersecting curved surfaces. Representing such general critical curves as edges in a discontinuity mesh requires describing mesh elements with curved boundaries, and even if parametric representations were used, these boundaries would be curved in the parametric space of the surface. Furthermore, reconstructing a radiance function from such discontinuity meshes requires satisfying continuity constraints across curved element boundaries.

Many of these problems could be addressed by discretizing the curves into line segments. How to decide on the appropriate level of subdivision, though, is not clear. One of the achievements of the discontinuity meshing algorithm, in fact, has been to increase the degree of view-independence of radiosity solutions. Approximating a curve with straight line segments would certainly reduce the view-independence of the solutions. Future useful research could investigate what families of curved surfaces would allow for analytical solutions to some of these problems. Some work in this direction has already been done in the field of computer vision to determine intervisibility between curved objects [KRIE89, KRIE90,SRIP89].

7.3 Conclusion

This thesis has proposed a new concept, discontinuity meshing, as a way of overcoming the limitations of current meshing techniques for radiosity. The rationale, and mathematical justification, for this idea was provided by an analysis of the characteristics of the radiance functions.

A discontinuity meshing radiosity algorithm was presented and results were shown indicating that discontinuity meshing can generate remarkably accurate solutions while at the same time drastically reducing the need for user intervention.

Although much work remains to be done in order to design a robust and efficient discontinuity meshing algorithm, we believe that discontinuity meshing represents an important step towards making radiosity practical for a wide variety of applications; the concept and ideas are sound and the benefits they provide suggest that they should become an integral part of future radiosity systems.

Bibliography

[AIRE90] Airey, John M., John H. Rohlf, and Frederick P. Brooks Jr. "Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments," Proceedings of the 1990 Symposium on Interactive 3D Graphics (Snowbird, Utah, March 25-28, 1990), in *Computer Graphics*, 24(2), March 1990, pages 41–50.

[AMAN84] Amantides, J. "Ray Tracing with Cones," Proceedings of SIGGRAPH'84 (Minneapolis, Minnesota, July 23–27, 1984), in *Computer Graphics*, 18(3), July 1984, pages 129–135.

[ARVO86] Arvo, James. "Backward Ray Tracing," SIGGRAPH'86 Developments in Ray Tracing Course Notes, August 1986.

[ARVO87] Arvo, James and David Kirk. "Fast Ray Tracing by Ray Classification," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 55–64.

[ARVO90] Arvo, James and David Kirk. "Particle Transport and Image Synthesis," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 63–66.

[ARVO91] Arvo, James, editor. *Graphics Gems II*, Academic Press, Inc., Boston, Massachusetts, 1991.

[BAUM86] Baum, Daniel R., John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. "The Back-Buffer Algorithm: An Extension of the Radiosity Method to Dynamic Environments," *The Visual Computer*, 2(5), September 1986, pages 298–308.

[BAUM89] Baum, Daniel R., Holly E. Rushmeier, and James M. Winget. "Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors," Proceedings of SIGGRAPH'89 (Boston, Massachusetts, July 31–August 4, 1989), in *Computer Graphics*, 23(3), July 1989, pages 325–334.

[BAUM91] Baum, Daniel R., Stephen Mann, Kevin P. Smith, and James M. Winget. "Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions," Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 28–August 2, 1991), in *Computer Graphics*, 25(4), July 1991, pages 51–60.

[BERG86] Bergman, Larry, Henry Fuchs, Eric Grant, and Susan Spach. "Image Rendering by Adaptive Refinement," Proceedings of SIGGRAPH'86 (Dallas, Texas, August 18–22, 1986), in *Computer Graphics*, 20(4), August 1986, pages 29–38.

[BLIN77] Blinn, Jim F. "Models of Light Reflection for Computer Synthesized Pictures," Proceedings of SIGGRAPH'77 (San Jose, California, July 20–22, 1977), in *Computer Graphics*, 11(3), July 1977, pages 192–198.

[BLIN88] Blinn, James F. "Jim Blinn's Corner: Me and My (Fake) Shadow," *IEEE Computer Graphics and Applications*, 8(1), January 1988, pages 82–86.

[BOUV90] Bouville, Christian, Kadi Bouatouch, Pierre Tellier, and Xavier Pueyo. "Theoretical Analysis of Global Illumination Models," in Proceedings of Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics (Rennes, France, June 11–12, 1990), June 1990, pages 53–66.

[BREB89] Brebbia, C. A. and J. Dominguez. *Boundary Elements: An Introductory Course*, Computational Mechanics Publications, Southampton Boston, Massachusetts, 1989.

[BUCK89] Buckalew, Chris and Donald Fussell. "Illumination Networks: Fast Realistic Rendering with General Reflectance Functions," Proceedings of SIGGRAPH'89 (Boston, Massachusetts, July 31–August 4, 1989), in *Computer Graphics*, 23(3), July 1989, pages 89–98.

[CAMP90] Campbell, A. T., III and Donald S. Fussell. "Adaptive Mesh Generation for Global Diffuse Illumination," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 155–164.

[CAMP91a] Campbell, A. T., III and Donald S. Fussell. *An Analytic Approach to Illumination with Area Light Sources*, Technical Report TR-91-25, Department of Computer Sciences, University of Texas, Austin, Texas, August 1991.

[CAMP91b] Campbell, A. T., III and Donald S. Fussell. *Analytic Illumination with Polygonal Light Sources*, Technical Report TR-91-15, Department of Computer Sciences, University of Texas, Austin, Texas, April 1991.

[CAMP91c] Campbell, III, A. T. *Modeling Global Diffuse Illumination for Image Synthesis*, PhD dissertation, University of Texas at Austin, Austin, Texas, December 1991.

[CEND87] Cendes, Zoltan J. and Steven H. Wong. "C^1 Quadratic Interpolation Over Arbitrary Point Sets," *IEEE Computer Graphics and Applications*, 7(11), November 1987, pages 8–16.

[CHAT87] Chattopadhyay, Subdeb and Akira Fujimoto. "Bi-directional Ray Tracing," in Proceedings of CG International'87, 1987, pages 335–343.

[CHEN90a] Chen, Hong and En-Hua Wu. "An Adapted Solution of Progressive Radiosity and Ray-Tracing Methods for Non-diffuse Environments," in Proceedings of CG International'90: Computer Graphics Around the World, 1990, pages 477–490.

[CHEN90b] Chen, Shenchang Eric. "Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 135–144.

[CHEN91] Chen, Shenchang Eric, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. "A Progressive Multi-Pass Method for Global Illumination," Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 28–August 2, 1991), in *Computer Graphics*, 25(4), July 1991, pages 165–174.

[CHIN89] Chin, Norman and Steven Feiner. "Near Real-Time Shadow Generation Using BSP Trees," Proceedings of SIGGRAPH'89 (Boston, Massachusetts, July 31–August 4, 1989), in *Computer Graphics*, 23(3), July 1989, pages 99–106.

[CHIN92] Chin, Norman and Steven Feiner. "Fast Object-Precision Shadow Generation for Area Light Sources Using BSP Trees," in Proceedings of 1992 Symposium on Interactive 3D Graphics (Cambridge, Massachusetts, March 29–April 1, 1992), March 1992.

[COHE85] Cohen, Michael F. and Donald P. Greenberg. "The Hemi-Cube: A Radiosity Solution for Complex Environments," Proceedings of SIGGRAPH'85 (San Francisco, California, July 22–26, 1985), in *Computer Graphics*, 19(3), July 1985, pages 31–40.

[COHE86] Cohen, Michael F., Donald P. Greenberg, and David S. Immel. "An Efficient Radiosity Approach for Realistic Image Synthesis," *IEEE Computer Graphics and Applications*, 6(2), March 1986, pages 26–35.

[COHE88a] Cohen, Michael F. "A Consumer's and Developer's Guide to Radiosity," SIGGRAPH'88 A Consumer's and Developer's Guide to Image Synthesis Course Notes, August 1988.

[COHE88b] Cohen, Michael F., Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. "A Progressive Refinement Approach to Fast Radiosity Image Generation," Proceedings of SIGGRAPH'88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics*, 22(4), August 1988, pages 75–84.

[COOK81] Cook, Robert L. and Kenneth E. Torrance. "A Reflectance Model for Computer Graphics," Proceedings of SIGGRAPH'81 (Dallas, Texas, August 3–7, 1981), in *Computer Graphics*, 15(3), August 1981, pages 307–316.

[COOK82] Cook, Robert L. "A Reflectance Model for Computer Graphics," *ACM Transactions on Graphics*, 1(1), January 1982, pages 7–24.

[COOK84a] Cook, Rob L. "Shade Trees," Proceedings of SIGGRAPH'84 (Minneapolis, Minnesota, July 23–27, 1984), in *Computer Graphics*, 18(3), July 1984, pages 223–231.

[COOK84b] Cook, Robert L. "Distributed Ray Tracing," Proceedings of SIGGRAPH'84 (Minneapolis, Minnesota, July 23–27, 1984), in *Computer Graphics*, 18(3), July 1984, pages 137–146.

[COOK86a] Cook, Robert L. "Practical Aspects of Distributed Ray Tracing," SIGGRAPH'86 Developments in Ray Tracing Course Notes, August 1986.

[COOK86b] Cook, Robert L. "Stochastic Sampling in Computer Graphics," *ACM Transactions on Graphics*, 5(1), January 1986, pages 51–72.

[CORT82] Cortey, Noel E. "Interpolation of Arbitrary Spaced Points by Closed Surfaces," *computer & graphics*, 6(1), 1982, pages 351–364.

[CROW77] Crow, Franklin C. "Shadow Algorithms for Computer Graphics," Proceedings of SIGGRAPH'77 (San Jose, California, July 20–22, 1977), in *Computer Graphics*, 11(2), July 1977, pages 242–248.

[CROW81] Crow, Franklin C. "A Comparison of Antialiasing Techniques," *IEEE Computer Graphics and Applications*, 1(1), January 1981, pages 40–48.

[CROW84] Crow, Franklin C. "Summed-Area Tables for Texture Mapping," Proceedings of SIGGRAPH'84 (Minneapolis, Minnesota, July 23–27, 1984), in *Computer Graphics*, 18(3), July 1984, pages 207–212.

[DIPP85] Dippé, Mark A. Z. and Erling Henry Wold. "Antialiasing Through Stochastic Sampling," Proceedings of SIGGRAPH'85 (San Francisco, California, July 22–26, 1985), in *Computer Graphics*, 19(3), July 1985, pages 69–78.

[DL79] Di Laura, David L. "On a New Technique for Interreflected Component Calculations," *Journal of the Illumination Engineering Society*, 1979, pages 53–59.

[DL81] Di Laura, David L. "On Radiative Transfer Calculations in Unempty Rooms," *Journal of the Illumination Engineering Society*, 1981, pages 120–126.

[DL82] Di Laura, David L. "On the Simplification of Radiative Transfer Calculations," *Journal of the Illumination Engineering Society*, 1982, pages 12–16.

[DORS91] Dorsey, Julie O'B., François X. Sillion, and Donald P. Greenberg. "Design and Simulation of Opera Lighting and Projection Effects," Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 28–August 2, 1991), in *Computer Graphics*, 25(4), July 1991, pages 41–50.

[DUFF79] Duff, Tom. "Smoothly Shaded Renderings of Polygonal Objects on Raster Displays," Proceedings of SIGGRAPH'79 (Chicago, Illinois, August 8–10, 1979), in *Computer Graphics*, 13(2), August 1979, pages 270–275.

[FARI90] Farin, Gerald. *Curves and Surfaces for Computer Aided Geometric Design, Computer Science and Scientific Computing*, Academic Press, Inc., San Diego, California, second edition, 1990.

[FOLE90] Foley, James D., Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics, Principles and Practice,* Addison-Wesley Publishing Company, Reading, Massachusetts, second edition, 1990.

[FOUR88] Fournier, Alain and Eugene Fiume. "Constant-Time Filtering with Space-Variant Kernels," Proceedings of SIGGRAPH'88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics*, 22(4), August 1988, pages 229–238.

[FRAN82] Franke, Richard. "Scattered Data Interpolation: Tests of Some Methods," *Mathematics of Computation*, 38(157), January 1982.

[FUCH80] Fuchs, Henry, Zvi M. Kedem, and Bruce Naylor. "On Visible Surface Generation by a Priori Tree Structures," Proceedings of SIGGRAPH'80 (Seattle, Washington, July 14–18, 1980), in *Computer Graphics*, 14(3), June 1980, pages 175–181.

[FUJI86] Fujimoto, Akira, Tanaka Takayuki, and Iwata Kansei. "ARTS: Accelerated Ray-Tracing System," *IEEE Computer Graphics and Applications*, 6(4), April 1986, pages 16–26.

[GEOR90] George, David W., Francois X. Sillion, and Donald P. Greenberg. "Radiosity Redistribution for Dynamic Environments," *IEEE Computer Graphics and Applications*, 10(4), July 1990, pages 26–35.

[GIGU88] Gigus, Ziv, John Canny, and Raimund Seidel. "Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects," in Proceedings of the Second International Conference on Computer Vision, December 1988, pages 30–39.

[GIGU90] Gigus, Ziv and Jitendra Malik. "Computing the Aspect Graph for Line Drawings of Polyhedral Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2), February 1990, pages 113–122.

[GIGU91] Gigus, Ziv, John Canny, and Raimund Seidel. "Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6), June 1991, pages 542–551.

[GLAS84] Glassner, Andrew S. "Space Subdivision for Fast Ray Tracing," *IEEE Computer Graphics and Applications*, 4(10), October 1984, pages 15–22.

[GLAS86] Glassner, Andrew. "Adaptive Precision in Texture Mapping," Proceedings of SIGGRAPH'86 (Dallas, Texas, August 18–22, 1986), in *Computer Graphics*, 20(4), August 1986, pages 297–306.

[GLAS89] Glassner, Andrew S., editor. *An Introduction to Ray Tracing*, Academic Press, Inc., San Diego, California, 1989.

[GLAS90] Glassner, Andrew S., editor. *Graphics Gems*, Academic Press, Inc., Boston, Massachusetts, 1990.

[GOLD87] Goldsmith, Jeffrey and John Salmon. "Automatic Creation of Object Hierarchies for Ray Tracing," *IEEE Computer Graphics and Applications*, 7(5), May 1987, pages 14–20.

[GOLU89] Golub, Gene H. and Charles F. Van Loan. *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland, second edition, 1989.

[GORA84] Goral, Cindy M., Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. "Modeling the Interaction of Light Between Diffuse Surfaces," Proceedings of SIGGRAPH'84 (Minneapolis, Minnesota, July 23–27, 1984), in *Computer Graphics*, 18(3), July 1984, pages 213–222.

[GORD91] Gordon, Dan and Shuhong Chen. "Front-to-Back Display of BSP Trees," *IEEE Computer Graphics and Applications*, 11(5), September 1991, pages 79–85.

[GOUR71] Gouraud, H. "Continuous Shading of Curved Surfaces," *IEEE Transactions on Computers*, C-20(6), June 1971, pages 623–628.

[GREE86] Greenberg, Donald P., Michael F. Cohen, and Kenneth E. Torrance. "Radiosity: A Method for Computing Global Illumination," *The Visual Computer*, 2(5), September 1986, pages 291–297.

[GREE89] Greenberg, Donald P., Michael Cohen, Roy Hall, Holly Rushmeier, and John Wallace. "SIGGRAPH'89 Radiosity Course Notes," August 1989.

[GREE90] Greenberg, Donald P., Michael Cohen, David George, Holly Rushmeier, John Wallace, and Greg Ward. "SIGGRAPH'90 Radiosity Course Notes," August 1990.

[GREE91] Greenberg, Donald P., Michael Cohen, Holly Rushmeier, François Sillion, and John Wallace. "SIGGRAPH'91 Radiosity Course Notes," August 1991.

[HAIN86] Haines, Eric A. and Donald P. Greenberg. "The Light buffer: a Shadow Testing Accelerator," *IEEE Computer Graphics and Applications*, 6(9), September 1986, pages 6–16.

[HAIN91a] Haines, Eric A. "Beams O' Light: Confessions of a Hacker," SIGGRAPH'91 Frontiers in Rendering Course Notes, July 1991.

[HAIN91b] Haines, Eric A. "Ronchamp: A Case Study for Radiosity," SIGGRAPH'91 Frontiers in Rendering Course Notes, July 1991.

[HAIN91c] Haines, Eric A. and John R. Wallace. "Shaft Culling for Efficient Ray-Traced Radiosity," in Proceedings of the Second Eurographics Workshop on Rendering (Barcelona, Spain, May 13–15, 1991), May 1991.

[HALL83] Hall, Roy A. and Donald P. Greenberg. "A Testbed for Realistic Image Synthesis," *IEEE Computer Graphics and Applications*, 3(8), November 1983, pages 10–20.

[HALL89] Hall, Roy. *Illumination and Color in Computer Generated Imagery*, Springer-Verlag, New York, New York, 1989.

[HANR90] Hanrahan, Pat and David Salzman. "A Rapid Hierarchical Radiosity Algorithm for Unoccluded Environments," in Proceedings of Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics (Rennes, France, June 11–12, 1990), June 1990, pages 151–172.

[HANR91a] Hanrahan, Pat. "Rapid Hierarchical Global Illumination Algorithms," SIGGRAPH'91 Frontiers in Rendering Course Notes, July 1991.

[HANR91b] Hanrahan, Pat, David Salzman, and Larry Aupperle. "A Rapid Hierarchical Radiosity Algorithm," Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 28–August 2, 1991), in *Computer Graphics*, 25(4), July 1991, pages 197–206.

[HE91] He, Xiao D., Kenneth E. Torrance, François X Sillion, and Donald P. Greenberg. "A Comprehensive Physical Model for Light Reflection," Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 28–August 2, 1991), in *Computer Graphics*, 25(4), July 1991, pages 175–186.

[HECK84] Heckbert, Paul S. and Pat Hanrahan. "Beam Tracing Polygonal Objects," Proceedings of SIGGRAPH'84 (Minneapolis, Minnesota, July 23–27, 1984), in *Computer Graphics*, 18(3), July 1984, pages 119–128.

[HECK86a] Heckbert, Paul S. "Filtering by Repeated Integration," Proceedings of SIGGRAPH'86 (Dallas, Texas, August 18–22, 1986), in *Computer Graphics*, 20(4), August 1986, pages 315–322.

[HECK86b] Heckbert, Paul S. "Survey of Texture Mapping," *IEEE Computer Graphics and Applications*, 6(11), November 1986, pages 56–67.

[HECK90] Heckbert, Paul S. "Adaptive Radiosity Textures for Bidirectional Ray Tracing," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 145–154.

[HECK91a] Heckbert, Paul S. *Simulating Global Illumination Using Adaptive Meshing*, PhD dissertation, Department of EECS, UC Berkeley, California, June 1991.

[HECK91b] Heckbert, Paul S. and James M. Winget. *Finite Element Methods for Global Illumination*, Technical Report UCB/CSD 91/643, CS Division, University of California at Berkeley, Berkeley, California, July 1991.

[HECK92a] Heckbert, Paul S. "Discontinuity Meshing for Radiosity," in Proceedings of the Third Eurographics Workshop on Rendering (Bristol, UK, May 18–20, 1992), May 1992, pages 203–216.

[HECK92b] Heckbert, Paul S. "Finite Element Methods for Radiosity," SIGGRAPH'92 Global Illumination Course Notes, July 1992.

[HECK92c] Heckbert, Paul S. "Radiosity in Flatland," in Proceedings of Eurographics'92, September 1992.

[HORN75] Hornbeck, Robert W. *Numerical Methods*, Quantum Publishers, New York, New York, 1975.

[HOTT67] Hottel, Hoyt C. and Adel F. Sarofim. *Radiative Transfer*, McGraw-Hill, New York, New York, 1967.

[HOU78] Hou, Hsieh S. and Harry C. Andrews. "Cubic Splines for Image Interpolation and Digital Filtering," *IEEE Transactions on Acoustic, Speech, and Signal Processing*, ASSP-26(6), December 1978, pages 508–517.

[HOWE82] Howell, John R. *A Catalog of Radiation Configuration Factors*, McGraw-Hill, New York, New York, 1982.

[IES91] IES. *IES Lighting Handbook*, Illuminating Engineering Society of North America, New York, New York, 1991.

[IMME86] Immel, David S., Michael F. Cohen, and Donald P. Greenberg. "A Radiosity Method for Non-Diffuse Environments," Proceedings of

SIGGRAPH'86 (Dallas, Texas, August 18–22, 1986), in *Computer Graphics*, 20(4), August 1986, pages 133–142.

[KAJI86] Kajiya, James T. "The Rendering Equation," Proceedings of SIGGRAPH'86 (Dallas, Texas, August 18–22, 1986), in *Computer Graphics*, 20(4), August 1986, pages 143–150.

[KAY86] Kay, Timothy L. and James T. Kajiya. "Ray Tracing Complex Scenes," Proceedings of SIGGRAPH'86 (Dallas, Texas, August 18–22, 1986), in *Computer Graphics*, 20(4), August 1986, pages 269–278.

[KEYS81] Keys, Robert G. "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Transactions on Acoustic, Speech, and Signal Processing*, ASSP-29(6), December 1981, pages 1153–1160.

[KIRK91] Kirk, David and James Arvo. "Unbiased Sampling Techniques for Image Synthesis," Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 28–August 2, 1991), in *Computer Graphics*, 25(4), July 1991, pages 153–156.

[KOK90] Kok, Arjan J. F., Celal Yilmaz, and Laurens H. J. Bierens. "A Two-Pass Radiosity Method for Bézier Patches," in Proceedings of Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics (Rennes, France, June 11–12, 1990), June 1990, pages 117–126.

[KOK91] Kok, Arjan J. F. and Frederik Jansen. "Source Selection for the Direct Lighting Computation in Global Illumination," in Proceedings of the Second Eurographics Workshop on Rendering (Barcelona, Spain, May 13–15, 1991), May 1991.

[KRIE89] Kriegman, D. J. and J. Ponce. "Computing Exact Aspect Graphs of Curved Objects: Solids of Revolution," in Proceedings of the IEEE Workshop on the Interpretation of 3D Scenes, IEEE Society Press, New York, New York, November 1989, pages 116–121.

[KRIE90] Kriegman, D. J. and J. Ponce. "Computing Exact Aspect Graphs of Curved Objects: Parametric Surfaces," in Proceedings of the Eighth National Conference on Artificial Intelligence, AAAI Press/MIT Press, Cambridge, Massachusetts, June 1990, pages 1074–1079.

[LEE85] Lee, Mark E., Richard A. Redner, and Samuel P. Uselton. "Statistically Optimized Sampling for Distributed Ray Tracing," Proceedings of SIGGRAPH'85 (San Francisco, California, July 22–26, 1985), in *Computer Graphics*, 19(3), July 1985, pages 61–68.

[LEE90] Lee, Mark E. and Richard A. Redner. "A Note on the Use of Non-linear Filtering in Computer Graphics," *IEEE Computer Graphics and Applications*, 10(3), May 1990, pages 23–29.

[LISC90] Lischinski, Daniel and Jakob Gonczarowski. "Improved Techniques for Ray Tracing Parametric Surfaces," *The Visual Computer*, 6(3), 1990, pages 134–152.

[LISC91] Lischinski, Dani, Filippo Tampieri, and Donald P. Greenberg. *Improving Sampling and Reconstruction Techniques for Radiosity*, Technical Report 91-1202, Department of Computer Science, Cornell University, Ithaca, New York, August 1991.

[LISC92] Lischinski, Dani, Filippo Tampieri, and Donald P. Greenberg. "Discontinuity Meshing for Accurate Radiosity," *IEEE Computer Graphics and Applications*, 12(6), November 1992, pages 25–39.

[LISC93] Lischinski, Dani, Filippo Tampieri, and Donald P. Greenberg. "Combining Hierarchical Radiosity and Discontinuity Meshing," Proceedings of SIGGRAPH'93 (Anaheim, California, August 1–6, 1993), in *Computer Graphics*, 27(4), August 1993.

[LS90] Le Saec, Bertrand and Christophe Schlick. "A Progressive Ray Tracing Based Radiosity with General Reflectance Functions," in Proceedings of Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics (Rennes, France, June 11–12, 1990), June 1990, pages 103–116.

[MAX90] Max, Nelson. "Cone-Spheres," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 59–62.

[MAXW86] Maxwell, Gregory M., Michael J. Bailey, and Victor W. Goldschmidt. "Calculations of the Radiation Configuration Factor Using Ray Casting," *Computer-Aided Design*, 18(7), September 1986, pages 371–379.

[MIST87] Mistrick, R. G. and D. L. Di Laura. "A New Finite Orthogonal Transform Applied to Radiative Transfer Calculations," *Journal of the Illumination Engineering Society*, 1987, pages 115–128.

[MITC87] Mitchell, Don P. "Generating Antialiased Images at Low Sampling Densities," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 65–72.

[MITC88] Mitchell, Don P. and Arun N. Netravali. "Reconstruction Filters in Computer Graphics," Proceedings of SIGGRAPH'88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics*, 22(4), August 1988, pages 221–228.

[MITC91] Mitchell, Don P. "Spectrally Optimal Sampling for Distributed Ray Tracing," Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 28–August 2, 1991), in *Computer Graphics*, 25(4), July 1991, pages 157–164.

[NAIM87] Naiman, Avi and Alain Fournier. "Rectangular Convolution for Fast Filtering of Characters," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 233–242.

[NAKA90] Nakamae, Eihachiro, Kazufumi Kaneda, Takashi Okamoto, and Tomoyuki Nishita. "A Lighting Model Aiming at Drive Simulators," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 395–404.

[NAYL90] Naylor, Bruce, John Amanatides, and William Thibault. "Merging BSP Trees Yields Polyhedral Set Operations," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 115–124.

[NEUM90] Neumann, Laszlo and Attila Neumann. "Efficient Radiosity Methods for Non-Separable Reflectance Models," in Proceedings of Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics (Rennes, France, June 11–12, 1990), June 1990, pages 83–102.

[NEUM91] Neumann, Laszlo and Csaba Kelemen. "Solution of Interreflection Problem for Very Complex Environments by Transillumination Method," in Proceedings of the Second Eurographics Workshop on Rendering (Barcelona, Spain, May 13–15, 1991), May 1991.

[NEWM79] Newman, William M. and Robert F. Sproull. *Principles of Interactive Computer Graphics*, McGraw-Hill, Tokyo, Japan, 1979.

[NISH83] Nishita, Tomoyuki and Eihachiro Nakamae. "Half-Tone Representation of 3-D Objects Illuminated by Area Sources or Polyhedron Sources," in Proceedings of the IEEE Computer Society's International Computer Software and Applications Conference (COMPSAC83) (Chicago, Illinois, November 1983), November 1983, pages 237–241.

[NISH85a] Nishita, Tomoyuki and Eihachiro Nakamae. "Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflections," Proceedings of SIGGRAPH'85 (San Francisco, California, July 22–26, 1985), in *Computer Graphics*, 19(3), July 1985, pages 23–30.

[NISH85b] Nishita, Tomoyuki, Isao Okamura, and Eihachiro Nakamae. "Shading Models for Point and Linear Light Sources," *ACM Transactions on Graphics*, 4(2), April 1985, pages 124–146.

[NISH86] Nishita, Tomoyuki and Eihachiro Nakamae. "Continuos Tone Representation of Three-Dimensional Objects Illuminated by Sky Light," Proceedings of SIGGRAPH'86 (Dallas, Texas, August 18–22, 1986), in *Computer Graphics*, 20(4), August 1986, pages 125–132.

[NISH87] Nishita, Tomoyuki, Yasuhiro Miyawaki, and Eihachiro Nakamae. "A Shading Model for Atmospheric Scattering Considering Luminous Intensity Distribution of Light Sources," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 303–310.

[PAIN89] Painter, James and Kenneth Sloan. "Antialiased Ray Tracing by Adaptive Progressive Refinement," Proceedings of SIGGRAPH'89 (Boston, Massachusetts, July 31–August 4, 1989), in *Computer Graphics*, 23(3), July 1989, pages 281–288.

[PARK83] Park, Stephen K. and Robert A. Schowengerdt. "Image Reconstruction by Parametric Cubic Convolution," *Computer Vision, Graphics, and Image Processing*, 23(3), September 1983, pages 258–272.

[PEAC85] Peachey, Darwyn R. "Solid Texturing of Complex Surfaces," Proceedings of SIGGRAPH'85 (San Francisco, California, July 22–26, 1985), in *Computer Graphics*, 19(3), July 1985, pages 279–288.

[PERL85] Perlin, Ken. "An Image Synthesizer," Proceedings of SIGGRAPH'85 (San Francisco, California, July 22–26, 1985), in *Computer Graphics*, 19(3), July 1985, pages 287–296.

[PHON75] Phong, Bui-Tuong. "Illumination for Computer-Generated Pictures," *Communications of the ACM*, 18(6), June 1975, pages 311–317.

[PRES90] Press, WilliamH., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*, Cambridge University Press, Cambridge, Massachusetts, 1990.

[RATL72] Ratliff, Floyd. "Contour and Contrast," *Scientific American*, June 1972, pages 90–101.

[REIC92] Reichert, Mark C. *A Two-Pass Radiosity Method Driven by Lights and Viewer Position*, Master's thesis, Program of Computer Graphics, Cornell University, Ithaca, New York, January 1992.

[RIVA86] Rivara, Maria-Cecilia. "Adaptive Finite Element Refinement and Fully Irregular and Conforming Triangulations," in Babuska, I., J. Gago, E. R. de Oliveira, and O. C. Zienkiewicz, editors, *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, John Wiley and Sons, 1986, pages 359–370.

[RIVA87] Rivara, Maria-Cecilia. "Numerical Generation of Nested Series of General Triangular Grids," in Chui, C. K., L. L. Schumaker, and F. I. Utreras, editors, *Topics in Multivariate Approximation*, Academic Press, Inc., Orlando, Florida, 1987, pages 193–206.

[RUSH87] Rushmeier, Holly E. and Kenneth E. Torrance. "The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 293–302.

[RUSH90a] Rushmeier, Holly E., Daniel R. Baum, and David E. Hall. "Accelerating the Hemi-Cube Algorithm for Calculating Radiation Form Factors," in 5th AIAA/ASME Thermophysics and Heat Transfer Conference (Seattle, Washington, June 1990), 1990.

[RUSH90b] Rushmeier, Holly E. and Kenneth E. Torrance. "Extending the Radiosity Method to Include Specularly Reflecting and Translucent Materials," *ACM Transactions on Graphics*, 9(1), January 1990, pages 1–27.

[SABL87] Sablonniére, Paul. "Composite Finite Elements of Class C^2," in Chui, C. K., L. L. Schumaker, and F. I. Utreras, editors, *Topics in Multivariate Approximation*, Academic Press, Inc., Orlando, Florida, 1987, pages 207–218.

[SALE89] Salesin, David and Jorge Stolfi. "The ZZ-Buffer: A Simple and Efficient Rendering Algorithm with Reliable Antialising," in Proceedings of the PIXIM'89 Conference (Hermes Editions, Paris), September 1989, pages 451–466.

[SALE91] Salesin, David. *Epsilon-Geometry: Building Robust Algorithms from Imprecise Computations*, PhD dissertation, Department of Computer Science, Stanford University, Palo Alto, California, 1991.

[SALE92] Salesin, David, Dani Lischinski, and Tony DeRose. "Reconstructing Illumination Functions with Selected Discontinuities," in Proceedings of the Third Eurographics Workshop on Rendering (Bristol, UK, May 18–20, 1992), May 1992, pages 99–112.

[SAME90a] Samet, Hanan. *Applications of Spatial Data Structures*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1990.

[SAME90b] Samet, Hanan. *The Design and Analysis of Spatial Data Structures*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1990.

[SCHM86] Schmitt, Francis J. M., Brian A. Barsky, and Wen-Hui Du. "An Adaptive Subdivision Method for Surface-Fitting from Sampled Data," Proceedings of SIGGRAPH'86 (Dallas, Texas, August 18–22, 1986), in *Computer Graphics*, 20(4), August 1986, pages 179–188.

[SCHU87] Schumaker, Larry L. "Triangulation Methods," in Chui, C. K., L. L. Schumaker, and F. I. Utreras, editors, *Topics in Multivariate Approximation*, Academic Press, Inc., Orlando, Florida, 1987, pages 219–232.

[SEDE84] Sederberg, Thomas and David C. Anderson. "Ray Tracing of Steiner Patches," Proceedings of SIGGRAPH'84 (Minneapolis, Minnesota, July 23–27, 1984), in *Computer Graphics*, 18(3), July 1984, pages 159–164.

[SEDE90] Sedeberg, Thomas W. "Techniques for Cubic Algebraic Surfaces," *IEEE Computer Graphics and Applications*, 10(4), July 1990, pages 14–25.

[SEGA90] Segal, Mark. "Using Tolerances to Guarantee Valid Polyhedral Modeling Results," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 105–114.

[SEQU89] Sequin, Carlo H. and Eliot K. Smyrl. "Parametrized Ray-Tracing," Proceedings of SIGGRAPH'89 (Boston, Massachusetts, July 31–August 4, 1989), in *Computer Graphics*, 23(3), July 1989, pages 307–314.

[SHAO88] Shao, Min-Zhi, Qun-Sheng Peng, and You-Dong Liang. "A New Radiosity Approach by Procedural Refinements for Realistic Image Synthesis," Proceedings of SIGGRAPH'88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics*, 22(4), August 1988, pages 93–102.

[SHIN87] Shinya, Mikio, Tokiichiro Takahashi, and Seiichiro Naito. "Principles and Applications of Pencil Tracing," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 45–54.

[SHIR90a] Shirley, Peter. *Physically Based Lighting Calculations for Computer Graphics*, PhD dissertation, Department of Computer Science, University of Illinois, Urbana-Champaign, Illinois, 1990.

[SHIR90b] Shirley, Peter. "Physically Based Lighting Calculations for Computer Graphics: a Modern Perspective," in Proceedings of Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics (Rennes, France, June 11–12, 1990), June 1990, pages 67–82.

[SHIR90c] Shirley, Peter. "A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes," in Proceedings of Graphics Interface'90 (Halifax, Nova Scotia, May 1990), May 1990, pages 205–212.

[SHIR91] Shirley, Peter. "Time Complexity of Monte Carlo Radiosity," in Proceedings of Eurographics'91 (Vienna, Austria, September 1–6, 1991), September 1991, pages 459–465.

[SIEG81] Siegel, Robert and John R. Howell. *Thermal Radiation Heat Transfer*, Hemisphere Publishing Corp., Washington D.C., 1981.

[SILL89] Sillion, François and Claude Puech. "A General Two-Pass Method Integrating Specular and Diffuse Reflection," Proceedings of SIGGRAPH'89 (Boston, Massachusetts, July 31–August 4, 1989), in *Computer Graphics*, 23(3), July 1989, pages 335–344.

[SILL91] Sillion, François X, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. "A Global Illumination Solution for General Reflectance Distributions," Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 28–August 2, 1991), in *Computer Graphics*, 25(4), July 1991, pages 187–196.

[SMIT87] Smith, Alvy Ray. "Planar 2-Pass Texture Mapping and Warping," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 263–272.

[SMIT92] Smits, Brian E., James R. Arvo, and David H. Salesin. "An Importance-Driven Radiosity Algorithm," Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 26–31, 1992), in *Computer Graphics*, 26(4), July 1992, pages 273–282.

[SPAR63] Sparrow, Ephraim M. "On the Calculation of Radiant Interchange between Surfaces," in Ibele, Warren E., editor, *Modern Developments in Heat Transfer*, Academic Press, New York, New York, 1963.

[SPAR78] Sparrow, E. M. and R. D. Cess. *Radiation Heat Transfer*, Hemisphere Publishing Corp., Washington D.C., 1978.

[SRIP89] Sripradisvarakul, Thawach and Ramesh Jain. "Generating Aspect Graphs for Curved Objects," in Proceedings of the IEEE Workshop on the Interpretation of 3D Scenes (Austin, Texas, November 1989), IEEE Society Press, New York, New York, November 1989, pages 109–115.

[STEW88] Stewman, John and Kevin Bowyer. "Creating the Perspective Projection Aspect Graph of Polyhedral Objects," in Proceedings of the Second International Conference on Computer Vision, December 1988, pages 494–500.

[STRO86] Stroustrup, Bjarne. *The C++ Programming Language, Addison-Wesley Series in Computer Science*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[SUTH74] Sutherland, Ivan E., Robert F. Sproull, and R. A. Schumacker. "A Characterization of Ten Hidden-Surface Algorithms," *ACM Computing Surveys*, 6(1), March 1974.

[TAKA90] Takagi, Atsushi, Hitoshi Takaoka, Tetsuya Oshima, and Yoshinori Ogata. "Accurate Rendering Technique Based on Colorimetric Conception," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 263–272.

[TAMM85] Tamminen, Markku and F. W. Jansen. "An Integrity Filter for Recursive Subdivision Meshes," *computer & graphics*, 9(4), 1985, pages 351–364.

[TAMP91] Tampieri, Filippo and Daniel Lischinski. "The Constant Radiosity Assumption Syndrome," in Proceedings of the Second Eurographics Workshop on Rendering (Barcelona, Spain, May 13–15, 1991), May 1991.

[TANA91] Tanaka, Toshimitsu and Tokiichiro Takahashi. "Shading with Area Light Sources," in Proceedings of Eurographics'91 (Vienna, Austria, September 1–6, 1991), September 1991, pages 235–246.

[TARJ83] Tarjan, Robert Endre. *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1983.

[TELL92] Teller, Seth J. "Computing the Antipenumbra of an Area Light Source," Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 26–31, 1992), in *Computer Graphics*, 26(4), July 1992, pages 139–148.

[THIB87] Thibault, William C. and Bruce F. Naylor. "Set Operations on Polyhedra Using Binary Space Partitioning Trees," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 153–162.

[TRUM91] Trumbore, Ben, Wayne Lytle, and Donald P. Greenberg. "A Testbed for Image Synthesis," in Proceedings of Eurographics'91 (Vienna, Austria, September 1–6, 1991), September 1991, pages 467–480.

[VARG62] Varga, Richard S. *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

[vH87] von Herzen, Brian and Alan H. Barr. "Accurate Triangulation of Deformed, Intersecting Surfaces," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 103–110.

[vH89] von Herzen, Brian. *Applications of Surface Networks to Sampling Problems in Computer Graphics*, PhD dissertation, California Institute of Technology, Pasadena, California, 1989.

[vL91] van Liere, R. "Divide and Conquer Radiosity," in Proceedings of the Second Eurographics Workshop on Rendering (Barcelona, Spain, May 13–15, 1991), May 1991.

[WALL87] Wallace, John R., Michael F. Cohen, and Donald P. Greenberg. "A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods," Proceedings of SIGGRAPH'87 (Anaheim, California, July 27–31, 1987), in *Computer Graphics*, 21(4), July 1987, pages 311–320.

[WALL89] Wallace, John R., Kells A. Elmquist, and Eric A. Haines. "A Ray Tracing Algorithm for Progressive Radiosity," Proceedings of SIGGRAPH'89 (Boston, Massachusetts, July 31–August 4, 1989), in *Computer Graphics*, 23(3), July 1989, pages 315–324.

[WALL90a] Wallace, John. "Trends in Radiosity for Image Synthesis," in Proceedings of Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics (Rennes, France, June 11–12, 1990), June 1990, pages 1–14.

[WALL90b] Wallace, John R. "Radiosity and Ray Tracing: A comparison of Shading Strategies," SIGGRAPH'90 Advanced Topics in Ray Tracing Course Notes, August 1990.

[WALL92] Wallace, John R. "Introduction to Meshing Algorithms for Radiosity," SIGGRAPH'92 Radiosity Course Notes, July 1992.

[WANG91a] Wang, Changyaw. "Direct Lighting Calculation by Monte Carlo Integration," in Proceedings of the Second Eurographics Workshop on Rendering (Barcelona, Spain, May 13–15, 1991), May 1991.

[WANG91b] Wanger, Leonard R. *Perceiving Spatial Relationships in Computer Generated Images*, Master's thesis, Program of Computer Graphics, Cornell University, Ithaca, New York, May 1991.

[WARD88] Ward, Gregory J., Francis M. Rubinstein, and Robert D. Clear. "A Ray Tracing Solution for Diffuse Interreflection," Proceedings of SIGGRAPH'88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics*, 22(4), August 1988, pages 85–92.

[WATT90] Watt, Mark. "Light-Water Interaction using Backward Beam Tracing," Proceedings of SIGGRAPH'90 (Dallas, Texas, August 6–10, 1990), in *Computer Graphics*, 24(4), August 1990, pages 377–386.

[WEGH84] Weghorst, H., G. Hooper, and Donald P. Greenberg. "Improved Computational Methods for Ray Tracing," *ACM Transactions on Graphics*, 3(1), January 1984, pages 52–69.

[WHIT80] Whitted, Turner. "An Improved Illumination Model for Shaded Display," *Communications of the ACM*, 23(6), June 1980, pages 343–349.

[WOLB89] Wolberg, George and Terrance E. Boult. "Separable Image Warping with Spatial Lookup Tables," Proceedings of SIGGRAPH'89 (Boston, Massachusetts, July 31–August 4, 1989), in *Computer Graphics*, 23(3), July 1989, pages 369–378.

[WOLB90] Wolberg, George. *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, California, 1990.

[XU89] Xu, Hau, Qun-Shang Peng, and You-Dong Liang. "Accelerated Radiosity Method for Complex Environments," in Proceedings of Eurographics'89 (September 1989), September 1989, pages 51–61.

[YEN56] Yen, J. L. "On Nonuniform Sampling of Bandwidth-Limited Signals," *IRE Transactions on Circuit Theory*, 3, December 1956, pages 251–257.

[ZATZ92] Zatz, Harold Robert Feldman. *Galerkin Radiosity: A Higher Order Solution Method for Global Illumination*, Master's thesis, Program of Computer Graphics, Cornell University, Ithaca, New York, August 1992.