Discontinuity Meshing for Radiosity*

Paul S. Heckbert

Department of Technical Mathematics & Informatics
Delft University of Technology
Julianalaan 132
2628 BL Delft
Netherlands

email: ph@duticg.tudelft.nl

16 April 1992

Abstract

The radiosity method is the most popular algorithm for simulating interreflection of light between diffuse surfaces. Most existing radiosity algorithms employ simple meshes and piecewise constant approximations, thereby constraining the radiosity function to be constant across each polygonal element. Much more accurate simulations are possible if linear, quadratic, or higher degree approximations are used. In order to realize the potential accuracy of higher-degree approximations, however, it is necessary for the radiosity mesh to resolve discontinuities such as shadow edges in the radiosity function. A *discontinuity meshing* algorithm is presented that places mesh boundaries directly along discontinuities. Such algorithms offer the potential of faster, more accurate simulations. Results are shown for three-dimensional scenes.

Keywords: global illumination, diffuse interreflection, adaptive mesh, shadow.

1 Introduction

One of the most challenging tasks of image synthesis in computer graphics is the accurate and efficient simulation of *global illumination* effects: the illumination of surfaces in a scene by other surfaces. Early rendering algorithms used a *local illumination* model, assuming that surfaces could be shaded independently, typically using a finite set of point light sources. Global illumination models, on the other hand, recognize the true interdependency of the shading problem: the radiance of a surface point is determined by the radiance of all the surfaces visible from that point.

Global illumination simulation is relevant to a number of scientific and engineering fields: lighting design in architecture, shape-from-shading in computer vision, neutron transport in physics, and thermal radiation in mechanical engineering.

The visual artifacts of global illumination are familiar in the everyday world, even if they are sometimes subtle: we see *penumbras* or soft shadows from area light sources, color bleeding between surfaces, and indirect lighting.

*From *Third Eurographics Workshop on Rendering*, Bristol, UK, May 1992, pp. 203-216.

In this work, we restrict our attention to *diffuse* materials: those with a matte surface that appears equally bright from all directions. More general global illumination algorithms simulate *specular*, or glossy, materials and *participating media* such as fog and other translucent media. Because of our assumptions, light radiance is a function of wavelength and the two surface parameters only.

1.1 Previous Work

Existing techniques for simulating global illumination generally fall into two classes: ray tracing methods and radiosity methods. In general, ray tracing is best at simulating specular surfaces and radiosity is best at simulating diffuse surfaces.

Ray tracing simulates light transport by tracing the paths of photons through the scene in one of two directions: either from the lights into the scene or from the eye into the scene. With the addition of Monte Carlo techniques, it is possible to simulate diffuse interreflection [Kajiya86]. This can be done efficiently by caching the slowly-varying diffuse component of radiance [Ward et al. 88].

1.1.1 Radiosity Algorithms

Radiosity methods have their roots in the simulation of thermal radiation in mechanical engineering [Sparrow63], and were extended and optimized for the simulation of complex scenes in computer graphics [Cohen-Greenberg85,Cohen et al. 88]. The radiosity method subdivides the scene into *elements* (subsurfaces) and creates a system of equations whose solution is the *radiosity* (the sum of emitted and reflected radiance, integrated over a hemisphere) of each element.

Early radiosity papers devoted little attention to *meshing*, the subdivision of surfaces into elements. They often performed meshing in a very simple manner: using a grid of congruent rectangular or triangular elements and approximating the true radiosity function with a function that is constant over each element. We call such an approximation a uniform mesh with constant elements. Though Gouraud shading is typically used for display, linear variation in radiosity is not taken into account during form factor calculation, with a few exceptions [Max-Allison92,Heckbert-Winget91]. The true radiosity function is not constant across each element, of course [Tampieri-Lischinski91]. Meshes with constant elements result in jaggy shadow boundaries and other artifacts.

Better meshes can be created by user intervention, *a posteriori* methods, and *a priori* methods.

A posteriori meshing constructs a first approximation using a coarse, uniform mesh, and then refines the mesh where the gradient appears to be large. Refinement can be done using quadtree subdivision [Cohen et al. 86]. Such methods help significantly, but they can miss small features (shadows of small objects), and they exhibit jagged shadow edges unless subdivision is extremely fine.

A priori meshing employs object-space techniques to predict, before solution, where shadow edges and other discontinuities will occur. A mesh is then constructed accordingly, and a solution found. Campbell's a priori meshing split the scene with planes through light source points and edges of occluding polygons, thereby approximating some shadow edges [Campbell-Fussell90]. A later improvement by Campbell found the discontinuities along the boundaries of the penumbra [Campbell91]. Baum split surfaces where they intersected or touched to resolve the most severe discontinuities [Baum et al. 91]. Rigorous meshing of 2-D scenes has also been explored [Heckbert91,Heckbert92,Lischinski et al. 91].

Another meshing improvement is to vary the mesh resolution according to the range of the interaction: a fine mesh is needed for nearby elements but a coarse mesh suffices for distant elements [Hanrahan et al. 91]. Hanrahan found, surprisingly, that many radiosity algorithms waste much of their time computing tiny form factors. He found that closer attention to the numerics of the simulation can sometimes result in great speedups.

1.1.2 Shadow Algorithms

Many of the techniques employed by a priori meshing were pioneered in object space shadow algorithms. The simplest shadow algorithms simulate only point light sources, computing the shadowed and unshadowed polygons using "cookie cutter" algorithms [Atherton et al. 78], or binary space partitioning (BSP) trees [Chin-Feiner90]. Nishita went further, computing penumbra boundaries from area light sources [Nishita-Nakamae83], thereby producing some amazingly realistic pictures without simulating interreflection. Chin has recently done the same using BSP trees [Chin-Feiner92].

1.2 Motivation

Current radiosity algorithms work well for some scenes: the progressive radiosity algorithm [Cohen et al. 88] is relatively fast and its artifacts are often imperceptible. Sometimes, however, existing algorithms are very slow, they exhibit objectionable artifacts, and they are inaccurate.

The computational requirements for radiosity on complex scenes can be prohibitive: a scene of about 5,000 polygons (broken into 1,000,000 elements) requires about 6 CPU-days on a fast (6 megaflop) computer using some of the best existing software [Baum et al. 91]. By comparison, Ward's ray tracing algorithm [Ward et al. 88] renders the same scene in about 2 days on a comparable machine. The relative errors of the two methods are not known, so this comparison is approximate. The fact that a diffuse-specific radiosity algorithm appears to be slower than a more general ray tracing algorithm suggests that faster radiosity algorithms must exist, however.

The artifacts of radiosity algorithms are sometimes annoying: small or distant light sources such as the Sun cause blocky shadow edges; and animation with moving objects suffers from severe popping as meshes change.

The final motivation for research on improved radiosity algorithms is the increasing importance of "physically-based" modeling and rendering: for many applications, pretty pictures are less important than accurate simulations. For simulations in mechanical engineering, for example, the primary output of a radiosity program is not a picture but a set of numbers or functions.

Hanrahan's results suggest that current radiosity methods may be much more brute force than necessary. He achieved great speedups for some scenes by improving the form factor computation algorithm; perhaps similar speedups are possible by better adaptive meshing.

1.3 Lessons from Two Dimensions

A thorough study of radiosity in a two-dimensional "flatland" world was made in order to better understand radiosity functions and the radiosity methods that approximate them [Heckbert91,Heckbert92].

Radiosity functions were shown to have discontinuities of value (degree 0 discontinuities) at touching surfaces, intersections, creases, and along shadow edges from point light sources;

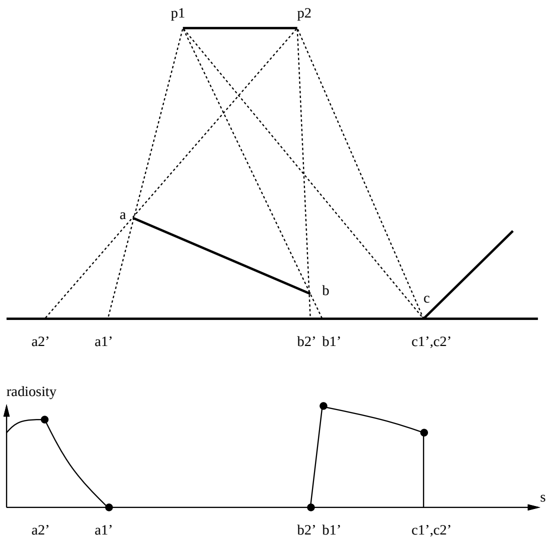

Figure 1: *Degree 1 and degree 0 discontinuities caused by linear light source at top (edge p1-p2) illuminating an occluded edge at bottom (edge a-b). Degree 1 discontinuities delimit the penumbras at a2', a1', b2', and b1'. Degree 0 discontinuity at the hard shadow edge c1'.*

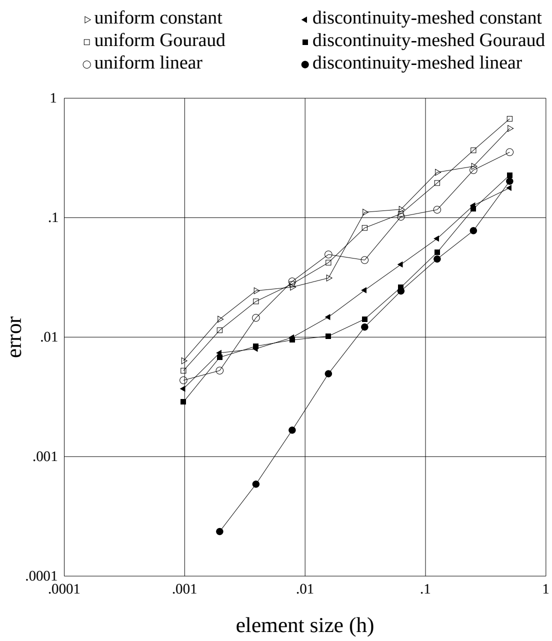

Figure 2: *Log-log plot of element size vs. error for six different solution methods. Note that discontinuity meshing with linear elements is far more accurate than the other methods, and that Gouraud elements are little better than constant elements.*

and first derivative (degree 1) discontinuities along shadow edges from linear light sources (figure 1). Radiosity algorithms employing uniform meshes were found to be inaccurate because they failed to *resolve* these discontinuities; their meshes prevented the discontinuities from being represented by the approximation.

An a priori adaptive meshing technique called *discontinuity meshing* was introduced that predicts the locations of discontinuities and places mesh boundaries at those points. Figure 2 shows a plot of error versus element size in a simple test scene for six different solution algorithms, formed by taking combinations of one of the two meshing techniques, uniform meshing or discontinuity meshing, and one of three element types.

The three element types used were constant elements (box basis), linear elements (hat basis), and what we call Gouraud elements. Gouraud elements are constant elements for form factor calculation, followed by linear interpolation for the 'display' step. We call them Gouraud elements by analogy with the computer graphics shading technique. Gouraud elements are the most commonly used approximation technique in existing radiosity algorithms. The error estimate here is an objective one: the square root of the integral of the squared difference between the approximation and a reference solution (obtained using an extremely fine mesh). Qualitatively similar plots result for other scenes in which occlusion occurs.

The results show that constant and Gouraud elements have nearly the same (low) accuracy. Convergence with decreasing element size is faster with linear elements than with constant or Gouraud elements, and convergence rate is highly dependent on discontinuities. Without discontinuity meshing, solution with any of the three element types converges slowly, but with discontinuity meshing, convergence can be relatively fast. Discontinuity meshing with linear elements gives results over 10 times more accurate on fine meshes, in these experiments.

The use of constant elements during problem formulation and solution followed by Gouraud interpolation for display gives results that are objectively no more accurate than the use of constant elements throughout (though they may look better).

Timing tests showed that, for these scenes, discontinuity meshing and linear elements are cost-effective; they gave the best accuracy for a given amount of total CPU time, and the fastest results at a given accuracy. In one test run, discontinuity meshing with linear elements gave results of the same quality as uniform meshing using about 1/60th the total time and 1/60th the memory.

The principal lesson of the 2-D research is that **discontinuity meshing and higher degree elements can increase the accuracy and/or speed of radiosity algorithms, but the techniques should be used together**. Neither technique, by itself, gives significantly better results than standard methods.

The use of higher degree elements in 3-D is straightforward, if no discontinuity meshing is done. In fact, it is possible to compute the form factors for linear element radiosity using a variant of the hemicube algorithm [Max-Allison92]. Since discontinuity meshing seems much more challenging than the use of higher degree elements, we devote the bulk of our attention to discontinuity meshing.

2 Discontinuity Meshing in 3-D

Discontinuity meshing is an approach to meshing that attempts to accurately resolve the most significant discontinuities in the solution by optimal positioning of element boundaries. For the two-dimensional elements of 3-D radiosity, element boundaries are curves.

2.1 Visibility

Discontinuities in the radiosity solution are caused by changes in visibility called *events*. Changes in visibility occur along *critical surfaces*, and potential discontinuities occur along *critical curves* where these surfaces intersect the faces (polygons) of the scene. For a scene of polygons, there are two types of events: vertex-edge (VE) events and edge-edge-edge (EEE) events [Gigus-Malik90].

VE events result from an inter-visible vertex v and edge e. The critical surface for this event is a subset of the plane containing the vertex and the edge. More precisely, it is the set of points collinear with vertex v and a point p of edge e, that are not between v and p. We call this surface a *wedge* (figure 3). The critical curves caused by a VE event are line segments; they are the points of the intersection of the wedge with the faces of the scene that are visible to v and p.

EEE events result from three inter-visible, skew edges (figure 4). The critical surface is the set of points simultaneously collinear with one point from each edge but not between any two of these three points. This surface is a subset of a quadric [Gigus-Malik90]. The critical curves caused by an EEE event are conic segments; they are the face points on the critical surface that are visible to the three edge points on the line.

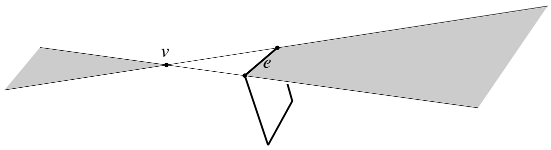

Figure 3: *The wedge defined by vertex v and edge e.*

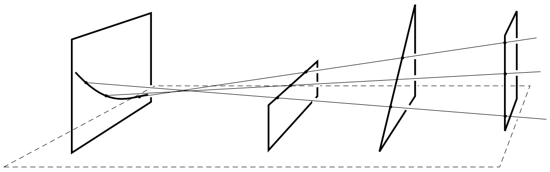

Figure 4: *EEE event caused by three skew edges at right creates a quadric critical surface and a conic critical curve on face at left.*

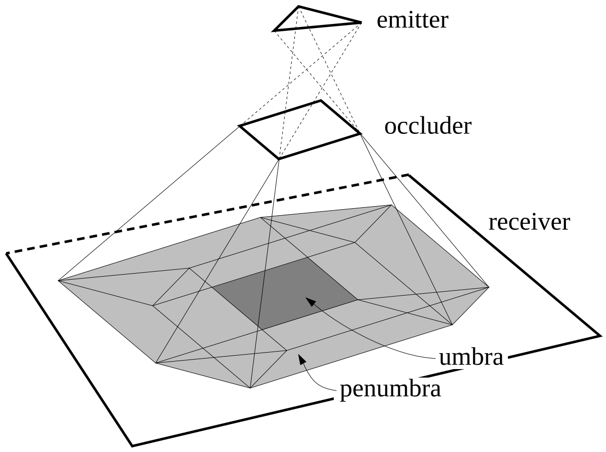

Figure 5: *Discontinuities caused by a triangular emitter, rectangular occluder, and receiver in parallel planes (after [Nishita-Nakamae83]).*

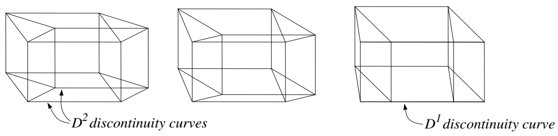

Figure 6: *Discontinuities from a triangular emitter and rectangular occluder, for three different rotations of the emitter. When edges of emitter and occluder become coplanar, two degree 2 discontinuity curves coincide, yielding a degree 1 discontinuity.*

2.2 Discontinuities

Touching or intersecting surfaces usually yield degree 0 discontinuities along their common curve. Shadow edges from point, linear, and area light sources are in general of degree 0, 1, and 2, respectively, but when two discontinuities coincide, the degree of the discontinuity can decrease [Heckbert91]. The lowest degree discontinuities generally cause the greatest error, if not resolved by an approximation.

Many of the characteristics of VE discontinuities are shown in figure 5. A top view of these discontinuities is shown in figure 6. When rotation of the occluder in its plane causes an edge of the emitter and an edge of the occluder to become coplanar, two degree 2 discontinuity curves coincide, yielding a degree 1 discontinuity.[1]

[1] This can be observed empirically by holding a stiff piece of paper horizontally below a rectangular lamp fixture and a few inches above a white tabletop. Rotate the piece of paper in its plane, and observe the

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

The vast majority of discontinuities are imperceptible. The most significant discontinuities are what we perceive as shadows; the least significant ones result from changes in visibility with respect to non-emitting surfaces. Except in special circumstances (e.g. moonshadows), shadows from secondary light sources are negligible.

Earlier adaptive meshing schemes have resolved some of the discontinuities in radiosity solutions. Baum's system resolved the degree 0 discontinuities from touching or intersecting surfaces, but not degree 0 discontinuities from point light sources, or degree 1 or 2 discontinuities. As mentioned previously, the meshes in [Campbell-Fussell90] approximately resolved some discontinuities. Campbell's more recent algorithm, [Campbell91], resolves the discontinuities along the boundaries of the penumbra, but not the discontinuities within the penumbra.

2.3 Discontinuity Meshing Algorithm

Discontinuity meshing can be done in a three phase process: first, critical curves are found and stored with the faces on which they lie, then a mesh that follows the critical lines is created for each face, and finally, basis functions are chosen for the elements. Critical curves are accumulated in a linked list of line segments kept with each face in the scene. The degree of the expected discontinuity is stored with each critical curve. to aid the choice of basis functions later.

There are three classes of critical curves to be generated: those due to touching or intersecting surfaces, VE critical lines, and EEE critical curves. We discuss VE lines in this paper. EEE curves are more difficult to handle [Teller92]. Baum discussed touching and intersecting surfaces in [Baum et al. 91].

Figure 5 suggests an algorithm for determining all VE critical line segments. Half of the critical segments can be generated by projecting the occluders from each of the vertices of the emitter onto each of the other faces in the scene. The other half of the critical segments are generated by projecting the emitter from each of the vertices of the occluders onto each of the faces in the scene. (When we say "emitter" here, it could mean a secondary light source). In [Campbell-Fussell90], the first half of these segments are approximated, but not the second half.

For each triplet of emitter, occluder, and receiver leading to a discontinuity that is deemed to be significant, we find the VE wedge defined by a vertex of the emitter and an edge of the occluder, and the VE wedge from a vertex of the occluder and an edge of the emitter, determine the unoccluded portions of these wedges (figure 7), and compute the critical line segments of their intersection with the receiver.

To process each wedge, the line segments of intersection between the wedge plane and each polygon are found, and then a 2-D object-space visible edge algorithm is used to determine the edges visible from the point of view of the vertex. The algorithm used here is a 2-D variant of the Weiler-Atherton visible surface algorithm [Atherton et al. 78] that

increased crispness of the shadows when edges of the paper are parallel to edges of the light source. The radiance function has degree 1 discontinuities at parallel orientations, but degree 2 otherwise.

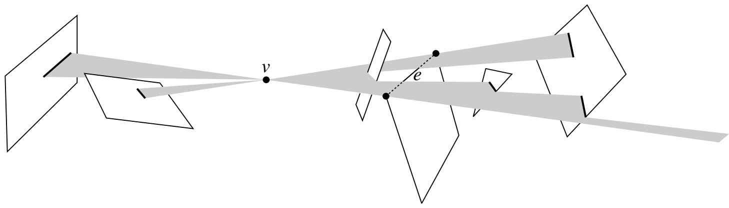

Figure 7: *Critical line segments due to vertex v and edge e shown bold.*

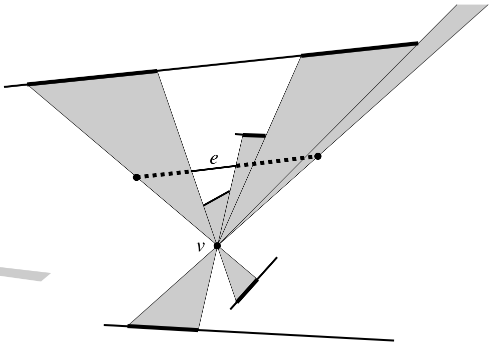

Figure 8: *Visible edge determination in the plane of the wedge. The visible portions of the line segments of intersection between the wedge and the faces in scene are the critical line segments.*

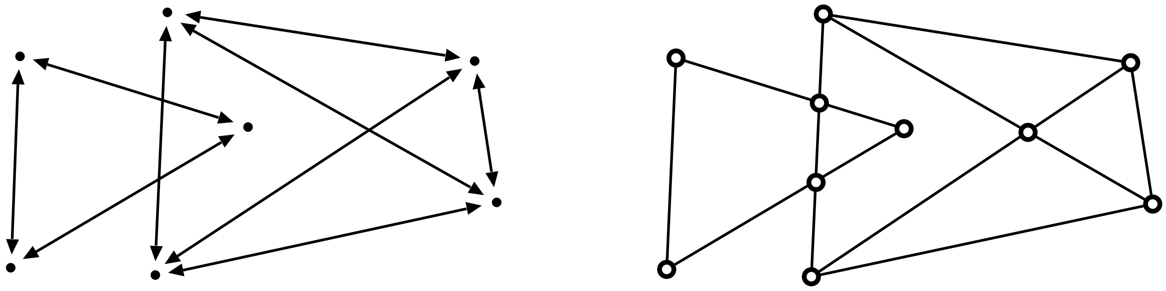

Figure 9: *Left: node and link data structure for critical lines. Right: topological data structure for the planar subdivision.*

maintains a linked list of unoccluded intervals, from which the intersection line segments are subtracted in front to back order (figure 8).

Once all of the critical line segments have been computed, a mesh for each face must be generated that subdivides along the critical line segments. These meshes are significantly more complex than the uniform grids or quadtrees employed in early radiosity algorithms. Therefore, a data structure supporting arbitrary planar subdivisions is recommended. The data structure should allow holes, concave polygons, faces with any number of sides, vertices of arbitrary degree, ordering of edges and faces around a vertex, and fast access to adjacent objects. An extended winged edge data structure was used here [Baumgart74]. BSP trees could also be used to represent the arrangement of critical lines on each face, but they split polygons unnecessarily, constraining the space of meshes, and lead to poorer quality meshes. Campbell employed BSP trees in his work [Campbell-Fussell90,Campbell91], and went to great lengths to minimize excessive subdivision.

The mesh is created in a three step process. First, a node-and-link data structure is created representing the endpoints of the critical lines and their adjacency graph (figure 9, left).

Next, the topology of the arrangement of critical line segments is computed by finding all

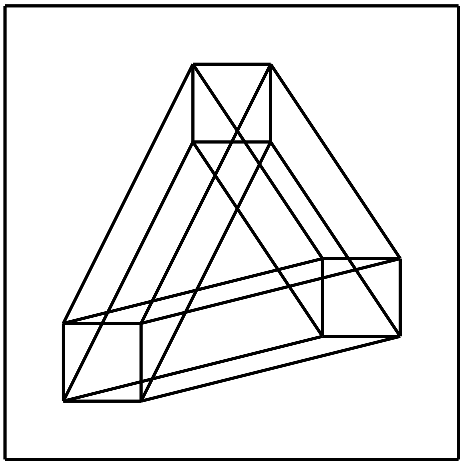

Figure 10: *Critical lines of test scene.*

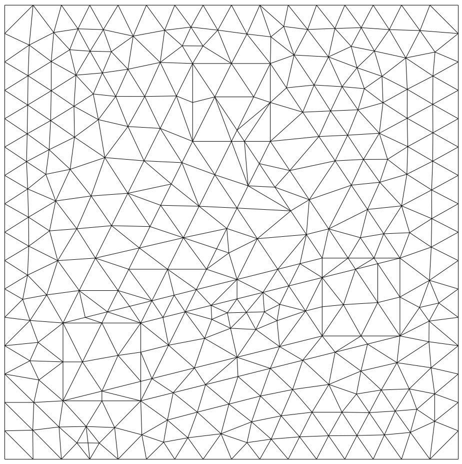

Figure 11: *Triangle mesh resulting from discontinuity meshing. Note that it follows all of the critical lines.*

intersections between critical lines, and subdividing the polygon into regions of homogeneous visibility bounded by critical lines or edges of the original polygon (figure 9, right). This is done with a 2-D sweepline algorithm similar to a polygon clipping algorithm [Weiler80]. This step creates a winged edge data structure. As the sweepline passes across each vertex or intersection point, the order of the emanating edges is found by a radial sort of the geometry. This ordering determines the local topology, which is used to update the winged edge data structure.

At this point, the topology and geometry of the arrangement of critical line segments is known, but the subdivision may be unsuitable for finite element radiosity purposes: faces of this subdivision can be concave, with holes. The final step subdivides these regions according to a user-selected maximum element size (figures 10 and 11). Delaunay triangulation and mesh relaxation were used here [Boender92], but it appears that there is considerable flexibility in this last step.

Once the mesh is constructed, basis functions are chosen for each element. If constant elements are used, then each element has one degree of freedom, and all elements are independent. If linear elements are used, then most vertices will contribute one degree of freedom, but vertices along critical lines can contribute two or more degrees of freedom depending on the discontinuity. Discontinuities in a finite element mesh are analogous to continuity constraints in a parametric curve or surface: basis functions should be selected so that the approximation function is C^{d-1} across a degree d critical curve. Each degree of freedom in the finite element mesh corresponds to a radiosity coefficient in the system of equations. To discuss the few remaining steps, we must review the generalization of radiosity methods to arbitrary basis functions.

2.4 Finite Element Methods for Radiosity

There are a variety of finite element methods for solving the integral equation governing radiosity [Heckbert-Winget91].

Figure 12 shows some of the available options for radiosity simulation using a finite

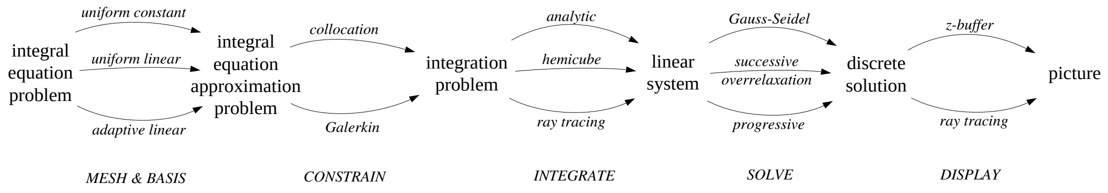

Figure 12: *Steps of finite element simulation of radiosity.*

element approach: one much choose a mesh and basis functions, constraint and integration methods to reduce the problem to a linear system of equations, a system solution method, and a display algorithm. For general references on finite element methods and integral equations, see the texts [Becker et al. 81] and [Delves-Mohamed85], respectively.

Radiosity obeys the integral equation [Heckbert-Winget91,Kajiya86]:

$$b(\mathbf{x}) = e(\mathbf{x}) + \int_\Gamma d\mathbf{x}'\, \kappa(\mathbf{x},\mathbf{x}')b(\mathbf{x}')$$

where $b(\mathbf{x})$ denotes the radiosity at 3-D point $\mathbf{x}$, $e(\mathbf{x})$ is radiant emitted flux density, and Γ is the two-dimensional domain of all surfaces in the scene.

The *kernel* of the integral equation is

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

where $\mathbf{x}'$ is another surface point, $\rho_h(\mathbf{x})$ is the hemispherical reflectance, θ is the angle at $\mathbf{x}$, θ' is the angle at $\mathbf{x}'$, r is the distance between $\mathbf{x}$ and $\mathbf{x}'$, and v, the visibility function, is 1 if points $\mathbf{x}$ and $\mathbf{x}'$ are inter-visible; 0 if occluded. θ, θ', v, and r are functions of $\mathbf{x}$ and $\mathbf{x}'$.

To solve this integral equation numerically, the exact solution $b(\mathbf{x})$ is approximated by a linear combination of basis functions:

$$\hat{b}(\mathbf{x}) = \sum_{i=1}^{n} b_i W_i(\mathbf{x})$$

where b_i are the unknown coefficients and W_i are the basis functions.

The collocation method for approximation, which most current radiosity algorithms employ, yields the following system of equations:

$$(\mathbf{M} - \mathbf{K})\mathbf{b} = \mathbf{e}$$

where $\mathbf{M}$ and $\mathbf{K}$ are $n \times n$ matrices and $\mathbf{b}$ and $\mathbf{e}$ are n-element column vectors:

$$\begin{aligned} M_{ij} &= W_j(\mathbf{x}_i) \\ K_{ij} &= \int_\Gamma d\mathbf{x}'\, \kappa(\mathbf{x}_i,\mathbf{x}')W_j(\mathbf{x}') \\ e_i &= e(\mathbf{x}_i) \end{aligned}$$

where $\mathbf{x}_i$, the collocation points, are typically the centers or vertices of the elements. The "form factors" here are the same as those of standard radiosity algorithms, except for the non-constant basis functions.

The most difficult factor in these integrations is the visibility function v, but this can be evaluated using a hemicube [Cohen-Greenberg85] or ray tracing. Hemicube algorithms can be generalized to evaluate linear basis functions using Gouraud shading [Max-Allison92].

A variety of system solution methods can be used, as indicated in figure 12, including generic methods such as Gauss-Seidel iteration, its faster variant successive overrelaxation, or specialized methods such as progressive radiosity. Display can be done using z-buffer algorithms or ray tracing, similar to integration.

2.5 Results

To test these ideas, a simple matrix radiosity program using ray traced form factors was modified. A matrix radiosity algorithm was used, rather than a progressive radiosity algorithm, so that the errors of progressive radiosity would not mask the errors introduced by the mesh. The linear system was solved by successive overrelaxation, a faster variant of Gauss-Seidel iteration [Golub-Van Loan89]. The program was written in C. The original small and simplistic uniform meshing module of about 150 lines of code was replaced by a much larger, more sophisticated discontinuity meshing module of about 8000 lines. (Baum's a posteriori adaptive mesh code was of similar size.) The modifications to the rest of the program were minor. Only constant and Gouraud elements have been implemented to date.

Figure 13 shows the shadow of a touching occluder simulated with uniform meshing and discontinuity meshing. The latter resolves the discontinuities much more accurately. These pictures were made with Gouraud elements, which probably accounts for the rough approximation of the shadows on the right. Much more accurate results could be expected with linear, quadratic, or higher degree elements in combination with discontinuity meshing.

Figure 14 (top row), shows the shadow of a triangle from a square light source. Note that the uniform mesh solution is quite blocky, while the discontinuity mesh solution is much smoother, exhibiting major discontinuities only where they really exist. Figure 11 shows the arrangement of critical lines and the mesh used for this approximation. Relative to the uniform mesh solution, discontinuity meshing used one fourth the number of elements (it used 112) and half the CPU time. If a posteriori quadtree adaptive meshing to this level were used, the number of elements and CPU time could be reduced, but the blocky artifacts would still remain.

Note that the elements now follow the shadow edges, and the approximation resolves the discontinuities in the true solution function more accurately. Because the mesh follows the discontinuities precisely, the error of the solution in the vicinity of the discontinuity is much reduced.

Figure 14 (bottom row), shows shadows from a distant light source (which is for all practical purposes a point light source). Uniform meshing results in an extremely blocky approximation, even with Gouraud shading, but discontinuity meshing resolves the degree 0 shadow edge exactly. Uniform meshing used 1600 elements, while discontinuity meshing used only 60. Consequently, for this simulation, uniform meshing solves a 1600×1600 system of equations, while discontinuity meshing solves a 60×60 system.

3 Conclusions and Ideas for Further Study

This work is an important step toward more rigorous treatment of radiance discontinuities in 3-D scenes. The algorithms described here should help to minimize the errors of radiosity solutions due to poor meshing. Such errors have heretofore been great.

We have shown that it is possible to find discontinuities in 3-D scenes and create a mesh that resolves the discontinuities. Such meshes offer the potential for faster and/or more accurate radiosity simulations. Although discontinuity meshing and higher degree elements are more complex than more brute force methods, they can pay off because they allow a much coarser mesh to be used. The approach followed here allows very accurate solutions to be explored. Alternative approaches using progressive radiosity and BSP trees appear to be easier to implement [Campbell91,Chin-Feiner92], but the accuracy of their results has not yet been demonstrated.

Although there are many discontinuities in most scenes, most of them are insignificant, so it is likely that discontinuity meshing can be practical for complex scenes. Experiments are needed to discover good measures of discontinuity significance. The application of faster computational geometry algorithms could help as well.

Further experiments are needed to test the accuracy and speed of linear, quadratic, and higher degree elements. EEE critical curves should be treated. Finally, a thorough series of experiments should be performed to determine the error due to each phase of the radiosity algorithm.

4 Acknowledgements

Jim Winget and Keith Miller contributed many ideas to my global illumination work at UC Berkeley. In Delft, Erik Jansen has been host to my postdoctoral fellowship, Frits Post helped me get access to a computer during Christmas week, and Edwin Boender wrote the triangulation code.

References

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

[Baum et al. 91] Daniel R. Baum, Stephen Mann, Kevin P. Smith, and James M. Winget. Making radiosity usable: Automatic preprocessing and meshing techniques for the generation of accurate radiosity solutions. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):51–60, July 1991.

[Baumgart74] Bruce G. Baumgart. *Geometric Modeling for Computer Vision*. PhD thesis, CS Dept, Stanford U., Oct. 1974. AIM-249, STAN-CS-74-463.

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

[Boender92] Edwin Boender. *Finite Element Mesh Generation from CSG Models*. PhD thesis, Dept. of Technical Math. & Informatics, Delft U. of Tech., Netherlands, Sept. 1992.

[Campbell91] A. T. Campbell, III. *Modeling Global Diffuse Illumination for Image Synthesis*. PhD thesis, CS Dept, University of Texas at Austin, Dec. 1991. Tech. Report TR-91-39.

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

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

[Chin-Feiner92] Norman Chin and Steven Feiner. Fast object-precision shadow generation for area light sources using BSP trees. In *1992 Symp. on Interactive 3D Graphics*, Mar. 1992.

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

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

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

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

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

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

[Hanrahan et al. 91] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):197–206, July 1991.

[Heckbert91] Paul S. Heckbert. *Simulating Global Illumination Using Adaptive Meshing.* PhD thesis, CS Division, UC Berkeley, June 1991. Tech. Report UCB/CSD 91/636.

[Heckbert92] Paul S. Heckbert. Radiosity in flatland. *Computer Graphics Forum*, 11(3):181–192, 464, Sept. 1992.

[Heckbert-Winget91] Paul S. Heckbert and James M. Winget. Finite element methods for global illumination. Technical report, CS Division, UC Berkeley, July 1991. UCB/CSD 91/643.

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

[Lischinski et al. 91] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Improving sampling and reconstruction techniques for radiosity. Technical report, CS Dept., Cornell U., Aug. 1991. TR 91-1202.

[Max-Allison92] Nelson L. Max and Michael J. Allison. Linear radiosity approximation using vertex-to-vertex form factors. In David Kirk, editor, *Graphics Gems III*, pages 318–323. Academic Press, 1992.

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

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

[Tampieri-Lischinski91] Filippo Tampieri and Dani Lischinski. The constant radiosity assumption syndrome. In *Second Eurographics Workshop on Rendering*, Barcelona, Spain, May 1991.

[Teller92] Seth J. Teller. Computing the antipenumbra of an area light source. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):139–148, July 1992.

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

[Weiler80] Kevin Weiler. Polygon comparison using a graph representation. *Computer Graphics (SIGGRAPH '80 Proceedings)*, 14(3):10–18, July 1980.

Figure 13: *Shadow of a nearly touching occluder. Left: uniform mesh solution, right: discontinuity mesh solution. Uniform meshing leads to a "light leak" under the triangle, while discontinuity meshing does not. Both solutions use Gouraud elements.*

Figure 14: *Four pictures on left: shadow of a triangle. Top row: shadow of a triangle from a nearby square light source, bottom row: shadow of a triangle from a distant light source. Left column: uniform mesh solution, right column: discontinuity mesh solution. Picture on right: Demonstration of diffuse interreflection. Critical lines shown in red, sight lines connecting vertices in blue.*