An Adaptive Discretization Method for Progressive Radiosity

Paul Lalonde
Department of Computer Science *
6356 Agricultural Road - Room 333
University of British Columbia
Vancouver, B.C. Canada V6T 1Z2

Abstract

The solutions of the radiosity method are highly dependent on the discretization used. All methods used to generate these discretizations have to date depended upon the scene being formed of polygonal faces. However, these are often not the most efficient representations of the objects. The meshing process usually only takes geometry into account, making shadow edges awkward to deal with. In addition, there are a number of restrictions that the radiosity method requires of the model that most available modellers do not enforce.

The method presented here allows non-polygonal objects to be used as input to a progressive radiosity method. The environment is sampled by ray casting, removing the need for a polygonal representation to be provided. The method allows the generation of a discretization that is sensitive to lighting changes, not only to geometric constraints. One effect of this is that higher order discontinuities in surface lighting are detected and the discretization can be focused in these areas without user intervention.

Résumé

Les solutions des méthodes de radiosité sont très dépendantes de la discrétisation utilisée. Toutes les méthodes utilisées à date pour créer ces discrétisations dépendent de la représentation de la scène par des polygones, ce qui n'est que rarement la représentation la plus efficace. De plus, le processus pour générer la grille porte souvent seulement attention à la géométrie, ce qui cause des difficultés pour trouver la limite des ombrages. Il y a aussi un nombre de contraintes que la méthode de radiosité impose sur les modèles que la plupart des modélisateurs ne garantissent pas.

La méthode présentée permet d'utiliser des objets non-polygonaux avec une méthode de radiosité progressive. La scène est échantillonnée par lancer de rayons, ce qui enlève la contrainte des scènes polygonales. La méthode crée une discrétisation dépendante des changements d'illumination en plus de la géométrie. Une conséquence de ceci est que les discontinuités d'illumination d'ordre supérieure d'une surface peuvent être détectées et l'effort de discrétisation peut être concentré dans ces régions.

*This work was done in the Department of Computing and Information Science at Queen's University

Keywords: Radiosity, automatic meshing, progressive radiosity, automatic discretization, triangulation

1 Introduction

Interreflection between surfaces plays an important role in indoor lighting. Without indirect lighting almost everything in a room would be completely black. But in reality we can look under tables and still find enough light to see a dropped pencil, this light coming from reflections off other surfaces in the environment. The reason that light-tinted paints seem to make a room larger is that light colours reflect more energy, causing brighter interreflections and so more light to see by. Unsurprisingly, scenes rendered without accounting for these interreflection effects look unnatural.

One way to solve the interreflection problem is the radiosity method. Adapted from the thermal engineering literature [10, 11], the radiosity algorithm works by calculating the diffuse energy transfers between the surfaces of the scene. To do this the surfaces are discretized into a mesh, the energy transfer being solved over this mesh. Each element of the mesh is assumed to have constant illumination and reflectance over its area.

1.1 The Radiosity Algorithm

Considering only diffuse reflection makes the amount of reflected light dependent only on the angle of incidence of the light, as described by Lambert's law, $I_r = I_i \rho \cos \theta_i$. The radiosity method works by building a system of equations relating the radiosity of each surface element, measured as energy per unit area per unit time, in a scene to every other element. Light sources emit light, also measured as energy per unit area per unit time. Both can be thought of as the brightness or intensity of light. The radiosity method also assumes that all light-surface interactions are Lambertian in nature. These interactions can be written down as

$$B_i dA_i = E_i dA_i + \rho_i \int_A B_j F_{ji} dA_j \qquad (1)$$

where the terms are

B_i the radiosity of differential area A_i;

E_i the emission of dA_i;

ρ_i the reflectivity of dA_i;

F_{ji} the form factor, relating the fraction of energy leaving dA_j arriving at dA_i

The solution of Equation 1 for all dA_i's gives the radiosity at every point. Since it is usually not possible to evaluate this expression directly it is usually discretized and solved numerically.

1.1.1 Full Matrix Radiosity

Assuming that each element A_i has a constant radiosity and reflectance over its surface gives the system

$$B_i A_i = E_i A_i + \rho_i \sum_j B_j F_{ji} A_j \qquad (2)$$

This can be simplified by noting that reversing the indices of two elements does not change the relative fractions of energy received by one from the other, that is to say $F_{ij} A_i = F_{ji} A_j$, which implies that $F_{ij} = F_{ji} A_j / A_i$. We can then divide Equation 2 through by A_i to obtain

$$B_i = E_i + \rho_i \sum_j B_j F_{ij} \qquad (3)$$

This can then solved for B as a large system of linear equations, $(I - \rho F)B = E$ using a Gauss Seidel [4] iterative method [5].

1.1.2 Progressive Radiosity

The full matrix radiosity method requires $O(n^2)$ space and $O(n^2)$ time to compute, where n is the number of elements in the discretization, because the matrix to solve is composed from a form factor for each pair of elements, and the Gauss-Seidel method works in $O(n^2)$ time. An alternative method has been developed [3]. Their progressive method casts light from each element, removing the requirement to store $O(n^2)$ form factors to solve the system of equations, by computing the form factors as they are needed.

The method works by keeping track of how much radiant energy an element has received that it has not re-radiated. The element with the most unradiated radiosity is selected and its energy cast to every other element in the scene. The element with next greatest radiosity is then selected to cast its unradiated radiosity, and the process repeats until the unshot radiosity the system drops below some tolerance. This method has the advantage of generating a passable image very quickly while converging to solution after more iterations. Also, there is no need to store the matrix of form factors; rather, they are calculated every time they are needed. This makes a complete solution less efficient to calculate, as the form factor calculations are expensive and must be repeated, but the method yields useful images earlier in the process, at interactive rates.

One limitation to these radiosity techniques is that the scene must be discretized to solve the equations. This can be problematic because many objects are not easily or efficiently described by polygons, so their discretizations use much more storage space than their parametric definition. A second limitation caused by the need for the discretization is that generating quality images requires a great deal of human interaction. The models generated by most modellers are inadequate as meshes for the radiosity method since many of the restrictions on the mesh that are required for a good radiosity solution are not enforced by the modeller. Thus such a user must be familiar with the workings of the radiosity method in order to manually adjust the model. This makes the radiosity method much less useful to the professionals who should be benefiting from it most, such as architects and designers.

This paper presents a method by which a mesh can be generated automatically, without user intervention. The method also allows non-polygonal objects to be used in the model, without having to discretize them explicitly into polygons before beginning the rendering process.

2 Previous Work

Campbell and Fussell approached the problem of subdividing the scene from a purely geometric standpoint[8]. They used BSP (Binary Space Partitioning) trees to detect projections of edges in the scene, thus finding lines upon which discontinuities in shading may occur. The method relies on the BSP representation of the scene, limiting environments to polygons.

Baum *et al.* developed a system in which the polygons in an environment could be subdivided if the shading at the vertices of the polygons is too different [1]. Upon detection of polygons with excessively high shading gradients these would be subdivided and new vertices interpolated. Their method also detects cases in which the input geometry violates the model requirements imposed by the radiosity method and corrects the model. Again, their method is dependent on a polygonal representation of the environment.

2.1 Failings

The methods presented depend upon a polygonal representation of the scene geometry. Each object must be decomposable into a set of polygonal faces. However, many objects exist that are not easily convertible to polygonal representations. It would be useful to be able to use any implicit surface without having to provide means of subdividing it.

Many objects are also very expensive to represent as polygons, for instance, those objects that are modeled as swept curves. Using each facet of such an object in the radiosity solution is expensive, since the storage and time costs of the radiosity method are $O(n^2)$ with respect to the number of elements. There are cases in which complex objects are hidden deep in shadow, contributing little to the scene. But these still require expensive form factor calculations to each facet, when only a few coarse patches could have provided adequate accuracy.

It would be useful to store objects in a simple form that is easy to manipulate and is independent of the discretization used by the radiosity method. The discretization should be generated without regard to the kind of geometric primitives that model the scene, using instead a metric based on lighting and illumination gradients over the scene, which in turn are dependent on the geometry.

3 Adaptive Illumination-Based Subdivision

The method presented here generates a discretization based on the difference in lighting, orientation and curvature of surfaces. Careful choice of the feature space allows objects to be discretized as necessary in order to obtain good shading definition. The geometry is sampled, but is not the only basis for the discretization. Since ray tracing is used to sample the scene the true geometry can be used to test for occlusions in form factor calculations. Likewise, the resampling step that generates the image for display can be performed using the true geometry, while gathering lighting information from the discretized environment. This removes silhouetting artifacts caused by curved objects being approximated by a series of polygons from the generated image. The mesh is formed using visibility information from each vertex, causing the mesh to be based on the whole objects rather than on the artifacts generated in the modelling process, such as patch boundaries within a polygonal representation of a swept surface.

3.1 Criteria for Discretization

Before selecting a discretization method it is important to understand what is required from the resulting discretization mesh. As mentioned above the mesh should not be entirely dependent on the geometry of the objects being discretized. The discretization should allow elements to span multiple objects, generating element boundaries where discontinuities occur in the illumination on surfaces and not exclusively in the provided representation of the geometry. By doing this a smaller mesh can be generated that better covers the scene. It would also be beneficial if the discretization allowed storage of radiosities at vertices rather than with the elements, to facilitate resampling [12]. The method should make it easy to build the discretization adaptively, refining it as more information about lighting and shading discontinuities becomes available.

Also, the method used to generate the discretization should place a higher priority on discretizing those parts of the scene that will generate the greatest increase in accuracy for work expended. If the discontinuities visible from every emitting and re-emitting element could be detected and included in the discretization at each step of the progressive radiosity solution, a very accurate result could be obtained. The discretization could then be built in such a way that the objects receiving light from the most important emitters would be discretized where the most change occurred [6]. The high cost of applying the radiosity method makes this particularly important if near-interactive run-times are required.

3.2 Discretization by Point Sampling

Many of these goals can be addressed by building a triangulation of the environment from sample points that are generated by ray casting. A triangulation was selected as the discretization because it is simple to work with: all its elements are convex polygons, and all neighbours sharing an edge of a triangle are connected to the triangle at two of the three vertices of the triangle (making adjacency checks easy while keeping the data structures simple). These properties are more difficult to maintain with quadrilaterals or other higher-order polygon meshes. Using a progressive-radiosity algorithm it is possible to provide a discretization that is sensitive to the elements emitting and reflecting light [1]. Since errors in the early stages of the progressive method propagate more extensively than those in later stages, due to the monotonic nature of the progressive method, it is important that the early stages be as accurate as possible. Biasing the discretization to be more responsive to the discontinuities, as well as continuous changes caused by stronger emitters, helps to reduce errors in the early passes of the progressive method.

The discretization process presented herein works in tight consort with the progressive radiosity algorithm. At each progressive radiosity step many rays are cast from the current emitter's vertices to identify discontinuities. The intersections of these rays with both the environment and the growing triangulation are recorded, along with the normals of intersection and the surface properties. These are then tested to determine if their addition to the triangulation would be beneficial to the solution obtained.

By considering all points of intersection along a ray rather than just the first point it is possible to detect and discretize areas that fall in shadow. This is important to the accurate generation of elements along shadow boundaries. If this is not done then no triangulation vertices will be created where shadows fall, leading to poorly defined shadows as no elements will span the shadow boundaries. By comparing the intersections with the real environment to those with the discretization it is also possible to detect features such as bumps, holes, and discontinuities that are inadequately represented by an existing element of the triangulation. The following sections will address the details of generating this discretization.

3.2.1 The Progressive Refinement Algorithm

The modified progressive radiosity algorithm works as follows; the steps are explained in detail in the next few subsections. The triangulation starts empty.

```
Procedure Radiosity
  Discretize light sources into triangles
  Refine from the eye, as described in Section 3.2.6
  $T \leftarrow$ brightest triangle
  While ( the solution has not converged )
    For each vertex $V$ of the triangulation $\notin T$
        Calculate new vertex radiosities
      $V.\Delta B \leftarrow V.\rho * T.B * F_{V-T}$
      $V.B \leftarrow V.B + V.\Delta B$
    End for
        Refine the discretization from current element
    Refine($T$)
    $T.\Delta B \leftarrow 0$
    For each triangle $R$ of the triangulation
          Calculate triangle radiosities
      $R.\Delta B \leftarrow$Average of R's vertices's $\Delta$ radiosities
      $R.B \leftarrow R.B + R.\Delta B$
      If ( Vertex $\Delta B$'s are too different ) then split($R$)
      If ( $R.\Delta B > T.\Delta B$ ) then $T \leftarrow R$
    End for
  End while
```

The principal changes are in the addition of the eye pass and the refinement step before calculating triangle radiosities. The refinement step assures that a sufficient discretization is formed relative to the current emitter, T. Once the method has converged to a solution, a resampling step from the view point is applied to complete the coverage of the triangulation seen from the view point and to generate the image.

3.2.2 Maintaining the Triangulation

As the triangulation develops, the mesh generation algorithm attempts to enforce five restrictive rules:

1. No edge in the triangulation will be longer than some δ.
2. No triangle will be smaller in area than some Δ.
3. Any vertex v' of the triangulation must be associated with a point v in the scene, so that the triangulation retains some resemblance to the actual scene.
4. The normals associated with the point v associated with any vertex v' of a triangle must be within some tolerance α_n of the normals of the points associated with the other vertices of that triangle.
5. During any iteration of the progressive radiosity method, the delta radiosity (ΔB) associated with any vertex of a triangle must be no more than some factor α_s from the ΔB's for any other vertex of that triangle.

When restrictions conflict, the lower numbered ones hold first. For instance, if application of Rule 5 indicates that a new triangle should be created, but its area would be smaller than Δ, then Rule 2 takes precedence and Rule 5 is not enforced.

This set of restrictions gives certain guarantees about the solutions. The maximum edge length guarantees that no excessively large elements are generated. It also allows the algorithm to search for neighbouring vertices in a limited neighbourhood rather than searching the entire discretization, significantly decreasing the running time of the implementation.

The minimum triangle size guarantees that no elements will be generated that are excessively small, leading to overly expensive solutions. This can happen in areas with a high shading gradient and at discontinuities. Making sure that each vertex of the triangulation is associated with a point in the scene guarantees that the surface information at each point of the triangulation properly represents the underlying scene. The last two requirements guarantee that triangles formed on shading discontinuities and on areas with high shading gradients will be split into more triangles, allowing these discontinuities to be properly represented in the discretization. In practice the delta radiosities are thresholded so that triangles dimly lit by an emitter are not needlessly split.

The triangulation is built incrementally, adding non-intersecting edges into the mesh and connecting them into three-cycles to make triangles. While building the triangulation a possible edge list is maintained. This list records all edges that are in the triangulation as well as any edge connecting two vertices that are not at the time part of a three-cycle to be connected into triangles. This list is used both to verify that any new edges do not cross any previously built edges, and to search for three-cycles of edges to be added to the triangulation.

To insert a point into the triangulation, a new vertex v is generated to correspond with the scene intersection generated by ray casting; the surface properties and normal of the vertex are those of the corresponding point in the scene. Each vertex w within a radius of δ is recorded, and each segment $\overline{vw}$ is tested against the segments of the possible edge list originating from a vertex within $\frac{\sqrt{5}}{2}\delta$ for intersection. Any edge that can intersect an edge of length δ with vertex v must have an end point within this radius. If no intersection is found, the edge $\overline{vw}$ is added to the possible edge list. Once all such edges are added, they are searched for three-cycles which include v and any such cycle is added as a triangle in the triangulation. To increase the speed of the searches the vertices are stored in a regular grid data structure so that only a few voxels need to be searched to find end points of edges within a given radius.

3.2.3 The Refinement Step

The refinement step works by casting rays from each corner of the current emitting element into both the scene and the discretization. All points of intersection are recorded and tested for inclusion into the triangulation. The refinement step is given by the following algorithm.

```
Procedure Refine($T$)
  For each vertex $V$ of $T$
    While ( more refinement is needed )
          Generate a new ray
      $R \leftarrow$ new-ray($V$)
          Get all intersections with scene
      $I_s \leftarrow R \cap Scene$
      $I_d \leftarrow R \cap Discretization$
      Merge lists $I_s$ and $I_d$, pairwise by distance
          Possibly change the triangulation
      For each pair $I_{si}, I_{di}$
        If ( $I_{si}$ is a dummy intersection ) then
          remove-from-triangulation($T$)
        If ( $I_{di}$ is a dummy intersection ) then
          extend-triangulation($I_{di}$)
        If ( both $I_{si}$ and $I_{di}$ are real ) then
          refine-triangulation($T$,$I_{si}$,$I_{di}$)
      End for
    End while
  End for
```

First a ray is generated; then it is intersected with both the scene and the discretization; and then the list of intersections is traversed to see if any of the points of intersection need to be added to the discretization. The distribution of the sample rays generated will be covered in Section 4. For the time being it is sufficient to note that the rays must somehow cover the hemisphere visible from the current emitter. Modifying the distribution affects the efficiency and accuracy of the solutions generated by the presented method.

Once generated, the intersection lists I_s (scene intersections) and I_d (discretization intersections) must be paired by distance. If a scene intersection and a discretization intersection are sufficiently close to one another relative to the distance from the origin of the intersecting ray, then they are paired. This simple distance heuristic can be wrong in situations where surfaces are very close to one another, choosing the wrong triangulation intersection to associate with the geometry intersection. (Consider, for example, Figure 1.) Experimental results have, however, shown this heuristic to be adequate.

Given this list of intersections it is then possible to start modifying the triangulation. Each pair of intersections (I_{si}, I_{di}) must be tested for one of three cases. If there is an intersection I_{s_i} with the scene and there is no corresponding intersection with the triangulation I_{d_i}, then the triangulation must be extended

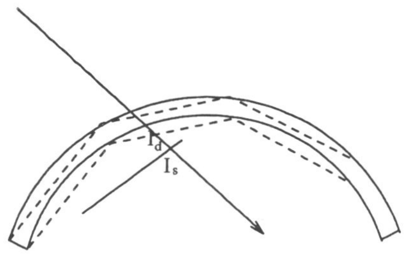

Figure 1: I_s may be paired with the discretization intersection I_d because the line segment has no discretization built on it, and I_d is close enough to be accepted by the distance heuristic.

to include the new point. If there is a triangulation intersection but no intersection with the scene, then the triangle intersected does not adequately represent the scene at that point and the triangle must be destroyed. If both exist, then the points must be tested to see if the triangulation needs to be refined or if it is adequate in its current state.

3.2.4 Removing Triangles

In the case where there is a discretization intersection without a scene intersection, the triangle intersected must be removed from the triangulation. This is caused by the scene being inadequately represented by the triangulation at that point, such as in cases where inadequate sampling caused a triangle to be built over a hole or around a steep peak in either the geometry or the illumination. This can be done by simply removing the triangle from the discretization. The edges should remain in the possible edge list since they may be useful for adding future triangles and will not hinder future progress.

3.2.5 Triangle Refinement

After each refinement step every triangle is examined to determine if the vertex delta radiosities due to the current emitter are within a tolerance α_s from their neighbours; this tolerance is also weighted by the importance of the change in radiosity at the element due to the current emitter relative to the element's radiosity. If they are, and the triangle is not already smaller than the user supplied limit Δ, then the triangle needs to be split, by adding a new vertex in the triangle, correcting the triangulation in that neighbourhood and interpolating the accumulated radiosity at that vertex from its neighbours. Care must be taken at this stage that the new vertex inserted into the triangulation corresponds to a point in the scene. This can be done by casting a ray toward the center of the triangle to be subdivided and using the point of intersection as the new vertex of the triangulation.

3.2.6 The Eye Pass

Although the method presented herein could be used to generate a solution independent of the viewpoint, the method can benefit greatly from a refinement step from the view point. This eye pass identifies geometric discontinuities that are visible from the view point, and generates a more pleasing solution with fewer gaps left to be interpolated in the resampling step by assuring a better coverage of the image plane with triangles. Without this step the discretization visible from the view point will contain many gaps that require expensive and inaccurate interpolation during the resampling step.

This refinement step can be done at a fairly fine level – the number of added triangles is small compared to the final number of triangles, and the cost of one refinement step is small compared to the cost of the complete solution.

There is one major difference between this refinement pass and all the others: there is no light being emitted from the eye, so all the tests on vertex and triangle delta radiosities cannot be used. This limits the information available to the triangulation algorithm to local geometric information. Other than this, the eye pass proceeds identically to the other refinement passes.

3.3 Form Factor Calculation

Form factors are needed each time a new delta radiosity is calculated, which occurs at several stages. The Form Factors are calculated from the differential areas at the vertices of the triangulation, dA_j, to the current emitting element, A_i, which may be adaptively subdivided. Occlusion is tested by casting rays through the scene, testing against the true geometry rather than the triangulated environment. The form factor from a vertex to a polygon can be calculated using the following equation, given the geometry of Figure 2.

$$F_{dA_j - A_i} = \frac{1}{2\pi} \sum_{g \in G_i} N_j \cdot \Gamma_g \qquad (4)$$

where

1. G_i is the set of edges of surface i,
2. N_j is the normal of differential area j, and
3. Γ_g is a vector with length equal to the angle γ expressed in radians, and perpendicular to the plane of R_g and R_{g+1}.

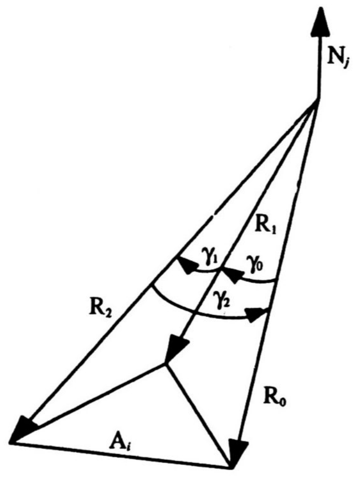

Figure 2: Geometry of analytic form factor

A detailed derivation can be found in [7]. Baum *et al.* make use of this form [2], but neglect to observe that the vector Γ_g is given by the cross product of the normalized vectors R_g and R_{g+1}, which are already available from the occlusion test calculations. Occlusion is tested by casting rays to the emitter corners from the receiving vertex, and if needed the emitter can be subdvided to increase the accuracy of the solution, as was done by Wallace *et al.* in [12].

3.3.1 Resampling

Generating an image from the radiosities stored in the discretization is done by casting rays at the scene, much the same way as ray tracing works. However, rather than casting shadow rays, a ray is cast into the discretization and the intersections paired, as in section 3.2.3. If there is a scene intersection and no discretization intersection then the triangulation is extended, and a radiosity value extrapolated from the vertices of the triangles to which the point was connected, weighted by the length of the connecting edge and its angular coverage. This value is then used to shade the pixel. Though this interpolation is innacurate very few are actually needed if the eye pass provided a good coverage of the scene visible from the eye. If there is a discretization intersection and no scene intersection, then the intersection must be discarded. If both exist, then the radiosity value is interpolated from the triangle hit. This value is returned to shade the pixel.

4 Sample Generation

The method used to generate the sample rays has an important impact on the efficiency of the algorithm and quality of the results. A small number of rays should provide an adequate discretization, while adding more rays should further refine it. More work should be spent discretizing the environment along shading discontinuities caused by the most important emitters and reflectors. It is also beneficial that more work be spent on parts of the scene that receive the most light from the sources and bright re-emitters, since these are evaluated early in the solution process, and they propagate more error through the solution than elements that cast their energy later in the process, as these elements are dimmer and so contribute less to the global lighting.

4.1 Uniform Distribution

A regular distribution of rays through the hemisphere above an element (called the visible hemisphere) does not satisfy these needs. Such a method requires that a fixed number of rays be cast, and lends itself to various aliasing problems, as described in [2]. A further failing is that the same amount of work is spent discretizing regions near the horizon, where less light will be received. Because these surfaces receive less light, errors made at the horizon will not be as important those made where more light is cast.

4.2 Importance-Based Distributions

Two of these problems can be overcome using a distribution proposed in [9]. The method distributes "jittered" samples on a unit circle centered at the source vertex. Jittered samples are generated by taking regularly distributed points on a grid and perturbing their positions by as much as half the interval between points. These points are then projected to a unit hemisphere above the source and rays are generated passing through the source vertex and the projected grid point. The sample density over the visible hemisphere is proportional to the cosine of the angle from the tangent to the surface to the sample point (which is the same relationship described by Lambert's law) thereby generating more rays where more of the emitter's light will be distributed. The density of the rays at different elevations on the hemisphere is proportional to the magnitude of the form factor to an element normal to the ray at the same elevation. Thus more samples are cast where more energy will be distributed, leading to more accurate solutions. The jittering also provides a means of anti-aliasing,

$\delta \backslash \alpha_s$	1.0	1.5	5.0
0.1	9056	8292	6237
0.15		8157	4468
0.2		7256	3438
0.3		7851	2787

Table 1: Number of Discretization Elements for various values of α_s and δ at iteration 25

converting aliasing effects to noise.

There is one deficiency inherent in the sampling method: small objects can easily be missed, causing some shadows to be missed. This can be partially addressed by forcing a minimum number of samples to be cast from each vertex. These initial samples can then be uniformly distributed or jittered, to try to obtain a good first cut of the discretization. A better solution is to generate the rays adaptively, but a treatment of adaptive sampling is beyond the scope of this paper.

5 Results and Discussion

The method described in Section 3 was implemented as a preprocessing pass to a ray tracer, RayShade[1].

A sample environment consisting of a room 1 meter on a side, with a sphere of 10 cm diameter resting upon a cylinder of the same diameter, 30 cm tall. A 0.01 m^2 light source was placed in the ceiling. This scene was rendered with different values for the parameters δ, the maximal edge length in the triangulation, and α_s, the maximum allowable difference in intensity at each vertex of a triangle. Figures 3 (a)-(b) show the scene rendered for different values of δ. A longer edge length in the triangulation generates a number of long, thin polygons, causing noise in the images. Figure 3 (c) shows the triangulation generated after 800 iterations with $\delta = 0.1$ and $\alpha_s = 2.0$. Table 1 gives the number of triangles generated at various values of α_s and δ.

5.1 Failings

The restriction allowing only point sampling of the environment is artificial and causes some difficulties. Starting without a mesh causes the early stages of the solution to be very inaccurate, often requiring that the energy transfer process be restarted after a few iterations because accumulated errors are too large. The extrapolation of lighting values to vertices outside the existing triangulation is simply too error prone. Restarting the energy transfer process with the built triangulation addresses this partially. The images in figures 3 and 4 were computed without this fix to better illustrate the problem. A better solution may be to use available geometric information to generate a coarse discretization to be further refined, perhaps using heuristics similar to those described in [1]. However, doing this removes part of the flexibility of the presented method by requiring a discretization of the objects *a priori*.

Another failing of the current implementation is that the number of elements generated is not efficiently bounded; it has proven necessary to limit the number of elements generated in order to keep the method fast. Although the number of elements generated in each successive iteration tends to become smaller, no practical metric has been developed that will detect when further discretization is futile. The current implementation simply tries to find regions to discretize where none exists, thus wasting computing time.

[1]RayShade is a public domain raytracer written by Craig Kolb. It is available on several internet ftp sites.

6 Conclusion

6.1 Summary

The discretization method presented allows the use of non-polygonal geometries with the progressive radiosity method, without requiring an *a priori* subdivision of the scene into polygons. The method uses point sampling to identify discontinuities in shading, and areas with a large illumination gradients caused by important sources in the scene. The elements used in the discretization are linear interpolants rather than constant elements, yielding more pleasing penumbrae at shadow edges rather than the sharp discontinuities associated with other radiosity algorithms. The discretization also allows the true geometry to be used in all occlusion tests and in the resampling step, yielding true object outlines rather than polygonal approximations.

6.2 Future Work

The results generated by the presented method are highly dependent on the method used to generate the sample rays that are used to probe the environment. More research needs to be done on the effects of different sampling methodologies on the discretization obtained. In particular some form of adaptive sampling appears to be very important. Making more

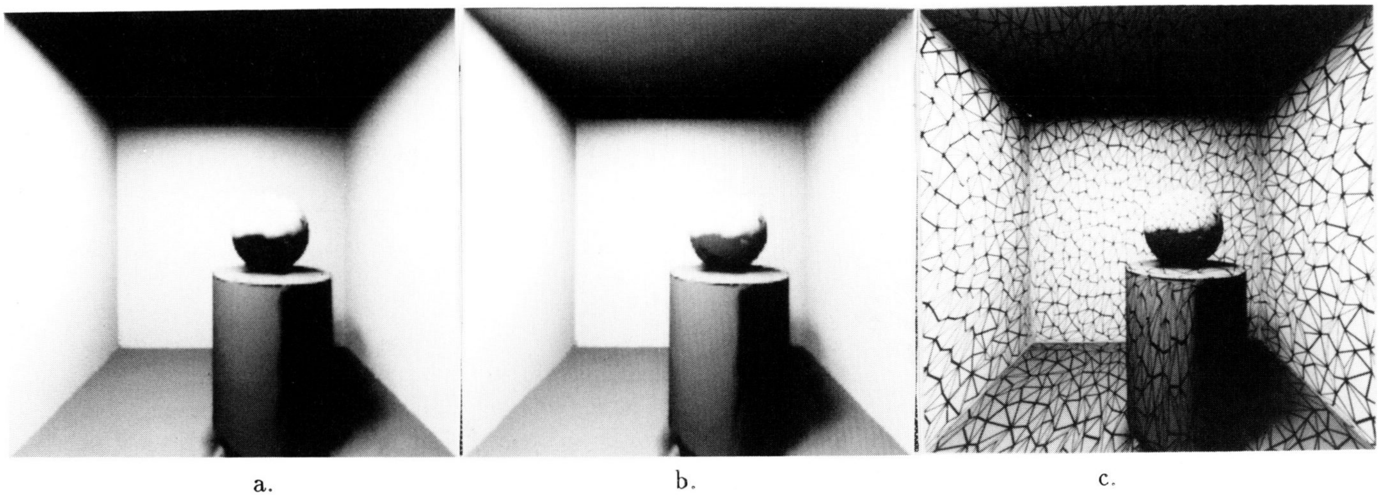

Figure 3: Two renderings of a simple scene ($\alpha_s = 2.0$, $\delta = 0.1$) at (a) 50 iterations, (b) 800 iterations, and (c) the generated mesh. Note the accumulation of errors arround the base of the pedestal and the equator of the sphere.

efficient use of rays cast will yield both shorter run times and more efficient discretizations.

Another useful path of investigation would be alternate triangulations. It should be possible to maintain a triangulation with a larger area to perimeter ratio than the current implementation does. Such a triangulation would help reduce the error caused by the long thin triangles currently generated when the δ parameter is too large.

6.3 Acknowledgements

Thanks to David Rappaport and Randy Ellis for supervising the Master's thesis from which this work is chiefly adapted. Funding for this research was provided the Ontario Graduate Scholarship Program and the Natural Sciences and Engineering Research Council.

References

[1] 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.

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

[3] 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, August 1988.

[4] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1983.

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

[6] Paul Heckbert. *Simulating Global Illumination Using Adaptive Meshing.* PhD thesis, University of California, Berkeley, Computer Science Division, June 1991.

[7] Hoyt C. Hottel and Adel F. Sarofim. *Radiative Transfer*. McGraw–Hill, 1967.

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

[9] Thomas Malley. A shading method for computer generated images. Master's thesis, The University of Utah, 1988.

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

[11] Ephraim M. Sparrow and Robert D. Cess. *Radiation Heat Transfer*. Hemisphere Publishing Corp., 1978.

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