

Exact Illumination in Polygonal Environments using Vertex Tracing

Michael M. Stark and Richard F. Riesenfeld

University of Utah
Salt Lake City, UT
mstark@cs.utah.edu

Abstract. Methods for exact computation of irradiance and form factors associated with polygonal objects have ultimately relied on a formula for a differential area to polygon form factor attributed to Lambert. This paper presents an alternative, an analytical expression based on vertex behavior rather than the edges of the polygon. Using this formulation, irradiance values in a scene consisting of partially occluded uniformly emitting polygons can be computed exactly by examining only the set of apparent vertices visible from the point of evaluation without explicit reconstruction of polygon contours. This leads to a fast, low-overhead algorithm for exact illumination computation that involves no explicit polygon clipping and is applicable to direct lighting and to radiosity gathering across surfaces or at isolated points.

1 Introduction and Previous Work

Fast and accurate computation of shadows continues to be one of the more perennial problems in computer graphics. Related problems include form-factor computation, visibility, and image reconstruction. In polygonal scenes, these problems ultimately amount to integration over visible portions of polygons. In this paper we consider the computation of the irradiance due to a collection of partially occluded uniformly emitting polygons. Numerous methods have been used to perform or approximate the integration, such as Monte Carlo integration, ray casting or other structured sampling approaches. Soler and Sillion [18] used the FFT to approximate the convolution integral for an occluded polygon. The hemi-cube and related algorithms can be used in the situation where there are many emitting polygons. Exact evaluation methods generally involve clipping the emitting polygon against all the intervening occluders then applying Lambert's formula. The backprojection method by Drettakis and Fiume [8] first partitions the receiver into regions of topologically equivalent visibility, then the scene is efficiently backprojected onto the emitting polygon. The method of Hart *et al.* [12] exploits scanline coherence to reduce the number of polygons involved in the clipping.

Methods such as the backprojection algorithm are fast, but have significant overhead. Furthermore, in radiosity environments where all the objects are emitters, the methods would have to be applied to each polygon in the scene. Aspect graphs or visibility maps can be used to compute the irradiance in a complex scene, but the entire graph must be constructed before evaluation can take place.

This paper presents an alternative to Lambert's formula, developed by projecting the emitting polygons onto an image plane. The resulting summation is reformulated in terms of the projected vertices and the slope of the "incoming" and "outgoing" edges. We then show how this formula can be used to compute the irradiance due to all the polygons in the scene by examining only the apparent vertices, *i.e.*, the vertices of

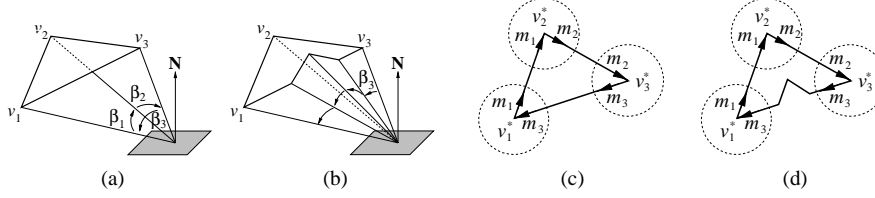


Fig. 1. (a) The geometry for Lambert's formula. (b) The angles β_i depend on the edges, so if part of the polygon is clipped by an occluder, the terms associated with the vertices of the affected edges have different values. (c) Using Green's theorem in the image plane produces a formula in terms of the local behavior at the vertices. (d) The contributions of the existing vertices are not affected if a bite is taken out of the polygon.

the visibility map, without actually computing the entire structure or performing any polygon clipping.

2 Irradiance from Diffusely Emitting Polygons

We recall that the irradiance from a uniformly emitting surface \mathcal{S} , which is not self-occluding as viewed from a point r on a receiver can be computed [2] from the surface integral

$$I(r) = M \int_{\mathcal{S}} \frac{\cos \theta_0 \cos \theta}{d^2} dS, \quad (1)$$

where d is the distance from r to a point on \mathcal{S} , θ_0 and θ are the angles made by the ray joining r and the point with the receiver normal at r and the surface normal at the point, respectively. The constant M is an emission constant of \mathcal{S} .

If the surface is a planar polygon P with vertices v_1, \dots, v_n , the irradiance may be computed from a formula attributed to Lambert:

$$L(r) = \sum_{i=1}^n \beta_i \cos \alpha_i, \quad (2)$$

where β_i is the angle subtended by v_i, v_{i+1} from r , and α_i is the angle between the plane containing v_i, v_{i+1} , and r , and the normal to the receiver at r (e.g., [6]). The drawback of (2) is that it depends on the angles between adjacent vertices, and thus requires the complete contour of the polygon to be known. In effect, Lambert's formula is a summation over the edges of the polygon rather than the vertices. Figure 1(a) illustrates the geometry.

Our objective is to construct a formula in terms of the vertices of the polygon P and the local behavior of the incident edges at each vertex. To do this, we project the polygon P through r onto an *image plane*, which is the plane parallel to the surface at r and one unit above (in the direction of the outward normal \mathbf{n} at r) as shown in Figure 2. This projection does not change the irradiance at r [3].

Lambert's formula shows that the irradiance is invariant under rotation about the normal \mathbf{n} , so the orientation of the x and y -axes in the image plane is not important. If \mathbf{u} is an arbitrary unit vector perpendicular to \mathbf{n} , and $\mathbf{v} = \mathbf{n} \times \mathbf{u}$, the projection of a vertex

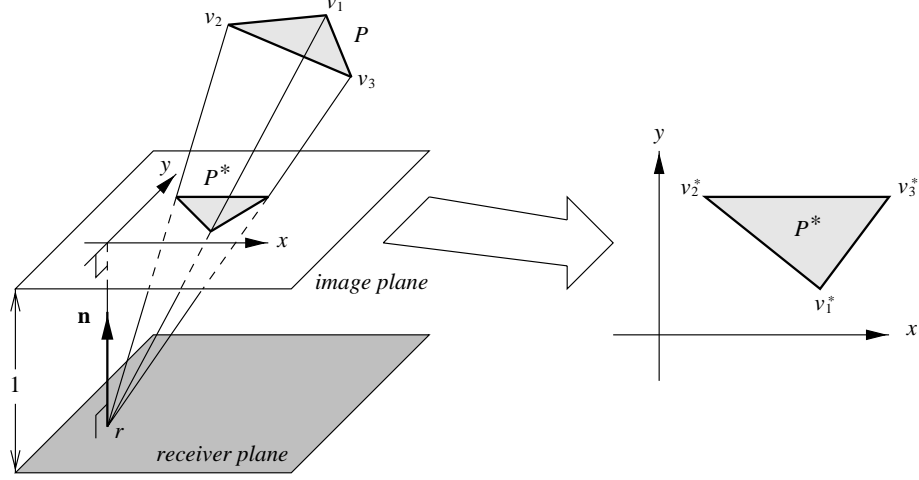


Fig. 2. To apply Green's theorem, the polygon P is projected onto an image plane parallel to the receiver plane, one unit above. The origin of the coordinate system of the image plane lies directly above the point of evaluation r on the receiver. The projection induces a reversal of orientation for front-facing polygons.

v of P may be computed, for example, using the homogeneous transformation

$$v^* = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{n} & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} \mathbf{I} & -r \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} v \\ 1 \end{bmatrix}. \quad (3)$$

(In the case of a polygonal receiver, \mathbf{u} can be a normalized edge; for a curved receiver, \mathbf{u} could be the direction of one of the curvilinear coordinates.) In what follows, we shall assume P has been projected onto the image plane forming a new planar polygon P^* having vertices v_1^*, \dots, v_n^* . Each vertex v_i^* of P^* will be treated as a two-dimensional point (x_i, y_i) in image plane coordinates.

2.1 Integration

For the projected polygon P^* , the integral of (1) has a particularly simple form; it reduces to the ordinary plane double integral (omitting the emission constant)

$$I(r) = \iint_{P^*} \frac{1}{(1 + x^2 + y^2)^2} dx dy. \quad (4)$$

This double integral may be reduced to a contour integral on the boundary of P^* using Green's theorem:

$$\oint_{\partial P^*} F_1 dx + F_2 dy = \iint_{P^*} \left(\frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} \right) dx dy. \quad (5)$$

The usual convention is counter-clockwise vertex ordering with respect to the outward normal. For a "front-facing" polygon, the angle between the outward normal and

the receiver surface normal is negative, so the projected polygon P^* will have a clockwise vertex ordering on the image plane, which means a negatively-oriented boundary contour and the sign of the left-hand side of (5) must be reversed.

Taking $F_2(x, y) \equiv 0$ and $F_1(x, y)$ an anti-derivative of the integrand in (4) with respect to y we obtain from Green's theorem

$$\iint_{P^*} \frac{1}{(1+x^2+y^2)^2} dx dy = \oint_{\partial P^*} F_1(x, y) dx = \sum_{i=1}^n \int_{E_i^*} F_1(x, y) dx.$$

The line integral over each edge can be evaluated by parameterizing the edge with the line equation $y = m_i x + b_i$ and integrating over the domain of the edge $E_i^* = v_i^* v_{i+1}^*$

$$\int_{E_i^*} F_1(x, y) dx = \int_{x_i}^{x_{i+1}} F_1(x, m_i x + b_i) dx = \Omega(x_{i+1}, m_i, b_i) - \Omega(x_i, m_i, b_i)$$

(vertical edges consequently drop out of the summation). Here Ω is

$$\Omega(x, m, b) = \int \left(\int \frac{1}{(1+x^2+y^2)^2} dy \right) \Big|_{y=mx+b} dx,$$

$m_i = (y_{i+1} - y_i)/(x_{i+1} - x_i)$ is the slope of the segment joining v_i^* and v_{i+1}^* , and b_i is the y -intercept of that line.

The irradiance integral may therefore be written as

$$\begin{aligned} I &= \sum_{i=1}^n \Omega(x_{i+1}, m_i, b_i) - \Omega(x_i, m_i, b_i) \\ &= \Omega(x_2, m_1, b_1) - \Omega(x_1, m_1, b_1) + \dots + \Omega(x_1, m_n, b_n) - \Omega(x_n, m_n, b_n) \\ &= \sum_{i=1}^n \Omega(x_i, m_{i-1}, b_{i-1}) - \Omega(x_i, m_i, b_i) \end{aligned}$$

As $b_i = y_i - m_i x_i$ and $b_{i-1} = y_i - m_{i-1} x_i$ the intercept term can be eliminated by introducing a new function $F(x, y, m) = \Omega(x, m, y - mx)$, and the final form of the solution thereby obtained is

$$I = M \sum_{i=1}^n F(x_i, y_i, m_{i-1}) - F(x_i, y_i, m_i). \quad (6)$$

The function F is

$$F(x, y, m) = \frac{Ax}{2} \arctan(Ay) + \frac{C(y - mx)}{2} \arctan[C(x + my)] \quad (7)$$

where

$$A = \frac{1}{\sqrt{1+x^2}}, \quad C = \frac{1}{\sqrt{1+m^2+(y-mx)^2}}. \quad (8)$$

Equations (6), (7) and (8) provide a formula analogous to Lambert's formula for the irradiance due to a uniformly emitting polygon. The first term in (7) is independent of m , and therefore appears to cancel in the summand of (6) so it is tempting to omit it from F . But recall that terms of F with undefined m are omitted outright, so in the case where only one of m_i and m_{i-1} is undefined, there is nothing to cancel the first term. The terms do cancel if neither incident edge is vertical.

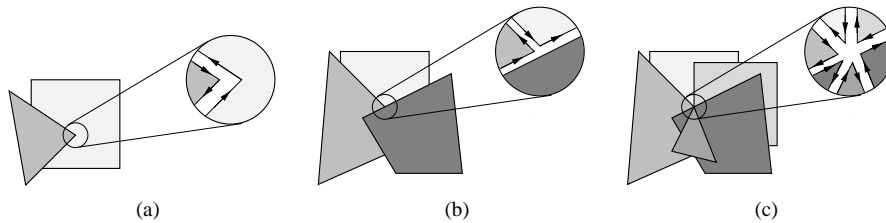


Fig. 3. Common cases of vertex behavior: (a) an intrinsic vertex against a background polygon, (b) an apparent vertex caused by the intersection of two edges against a background polygon, (c) a particularly unfortunate conjunction of three edges and one polygon vertex.

2.2 Remarks

There are several notable points about the result. Most importantly, the formula is a summation over a function of the vertices and the incoming and outgoing slopes m_{i-1} and m_i , respectively, and consequently may be evaluated in any order. In the case of an extraneous vertex, which has the same incoming and outgoing slope, the two F terms cancel and there is no contribution to the sum. Although the formula for F looks complicated, it is fairly easy to evaluate. Both the square root and arctangent functions have desirable computational behavior; note the radicand is bounded above 1.

The formula is valid only for a polygon which lies strictly above the plane of the receiver. As with Lambert's formula, the polygon must be clipped against the receiver plane, but unlike Lambert's formula, the projected polygon must be bounded on the image plane. (Otherwise the foregoing computation would have to be evaluated in the real projective plane.) One solution to this is to clip some small height above the receiver plane, another is to clip against a large bounding square on the image plane. The incurred error, as well as other vertex-based formulations, are discussed in [19].

3 Vertex Tracing and Angular Spans in Polygonal Environments

Equation (6) provides a method independent of vertex order for computing the irradiance due to a polygonal source. In a scene consisting of uniformly emitting polygons and perhaps other opaque occluding polygons, the scene projected onto the image plane consists of a collection of apparent polygons. The cumulative irradiance may therefore be computed at a point by examining only the apparent vertices of these projected polygons. The irradiance contribution at each vertex from (6) is summed over all the projected vertices to compute the total irradiance.

The projection of the scene onto the image plane is equivalent to the construction of the visibility map [16, ?]. That is, once the visibility map is constructed, our formula may be directly applied to compute the irradiance. However, the visibility map by definition includes the complete contour information of the projected polygons, and this defeats the purpose of the vertex formulation.

In this section, we propose a naive algorithm exploiting equation (6) by determining the apparent vertices of the projected scene using path tracing. The method is easily adapted to work with any number of emitting polygons in the scene, and is thus equally applicable to the problem of computing shadows from a single area light source as well as radiosity reconstruction, where every polygon in the scene is assumed to emit. Optimization methods and implementation details are discussed in the next section.

3.1 Visible vertices

Following Arvo [2], there are two types of vertices visible from a point r : *intrinsic* vertices, which are vertices of the original scene polygons, and *apparent* vertices, which are formed by the apparent intersection of two edges. Figure 3 illustrates these types of vertices as they appear from r against a “background” polygon. In Figure 3(a), an intrinsic vertex appears in front of a background polygon. There are two contributions to the irradiance sum in this case, one from the intrinsic vertex, and one from the projected intrinsic vertex onto the background polygon. If the emission constants of the foreground and background polygons are M and M_B , respectively, the contribution to the sum is

$$\begin{aligned} & M [F(x, y, m_{\text{in}}) - F(x, y, m_{\text{out}})] - M_B [F(x, y, m_{\text{in}}) - F(x, y, m_{\text{out}})] \\ = & (M - M_B)F(x, y, m_{\text{in}}) - (M - M_B)F(x, y, m_{\text{out}}). \end{aligned}$$

Figure 3(b) shows an apparent vertex, also against a background polygon. The computation of the irradiance contribution is similar, except there is an extra edge, and there is no contribution from the front-most polygon because the incoming and outgoing slopes are the same.

Figure 3(a) and (b) illustrate what are by far the most common situations for visible vertices. However, it is possible that vertices (intrinsic or apparent) may appear to coincide as viewed from the point of evaluation. We will use the term *conjunctive* vertex for this situation, in homage to ancient astronomers. Examples of conjunctive vertices include the apparent intersection of three edges, two intrinsic vertices, or an intrinsic vertex and an apparent vertex [9, 14]. Figure 3(c) shows an example of a conjunctive vertex containing three apparent vertices and one intrinsic vertex. Despite the complexity of the interaction, the local behavior is still sufficient to compute the irradiance contribution. Our method seamlessly handles conjunctive vertices of arbitrary complexity.

3.2 Angular Spans

The zoomed insets of Figure 3 demonstrate how the local behavior at each vertex (intrinsic, apparent, or conjunctive) can be represented using circular sectors, or angular spans. An angular span is a circular sector with emission and depth information. The angular spans for a vertex are naturally represented as a doubly-linked circular list, having nodes of the form

```
struct span {
    double     $\theta_1$  // smaller boundary
    double     $z$  // depth (set to  $\infty$  for the background)
    spectrum   $M$  // emission constant (can be zero)
}
```

Each angular span actually has two boundaries, θ_1 and θ_2 ; the second boundary is the θ_1 field of the next span in the list. (Our implementation uses tandem arrays to store the span list.) Angular spans are similar to linear spans used in scanline rendering (e.g., [23]) except that the opposite ends of a linear scan line do not “wrap around”. Angular spans crossing the branch cut at π radians have to be handled properly. The algorithm described below depends on a fast implementation of an angular span insertion algorithm, where the spans may be inserted in random depth order.

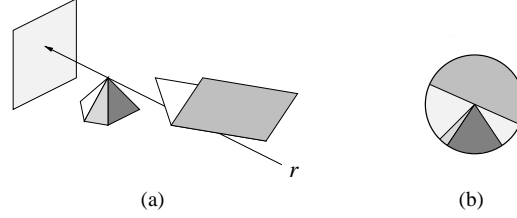


Fig. 4. (a) Tracing a conjunctive vertex, and (b) the resulting angular spans.

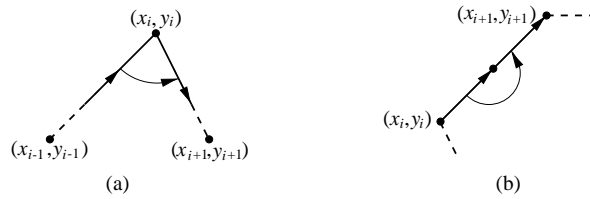


Fig. 5. (a) The span for an intrinsic vertex, (b) the span for an incident edge.

3.3 Naive Vertex Tracing

To compute the irradiance, all the visible vertices in the scene must be examined, which means each intrinsic vertex and all the apparent vertices from apparent edge intersections must be found and tested for visibility. Naively, the apparent vertices can be found by testing each pair of edges in the scene. Visibility is determined by tracing the ray from the point of evaluation r through the projected (intrinsic or apparent) vertex and collecting all vertices, edges and polygons incident on the ray. This way, all the participating polygons in a conjunctive vertex are found during the visibility test. A span list for the vertex is constructed by incrementally adding a span for each object incident on the ray (Figure 4). The depth value z for each span comes from the distance along the ray from r to the point of intersection. When the span list is completed, the contribution for the vertex, consisting of the contributions from all the spans in the list, is added to a master summation of the irradiance. The process is repeated for each vertex in the scene, and upon completion the master sum will equal the total irradiance.

There are two types of spans which will need to be added incrementally: vertex spans and edge spans. For a span at vertex (x_i, y_i) , $\theta_1 = \arctan(y_{i-1} - y_i, x_{i-1} - x_i)$ and $\theta_2 = \arctan(y_{i+1} - y_i, x_{i+1} - x_i)$. For an edge span, $\theta_1 = \arctan(y_i - y_{i+1}, x_i - x_{i+1})$ and $\theta_2 = \theta_1 + \pi$, where the points are as in Figure 5. In addition, when a ray hits the interior of a polygon a “full” span, with $\theta_1 = -\pi$, $\theta_2 = \pi$, is added at the depth of intersection. Once the angular span list has been fully constructed, the contribution of the vertex is computed using the formula

$$\sum_{s \in \text{span list}} (s.M - \text{prev}(s).M)F(x, y, \tan s.\theta_1). \quad (9)$$

Here $\text{prev}(s)$ denotes the predecessor of the span s in the span list. Note that a full angular span by itself has no contribution.

Algorithm 1 General vertex tracing

```
 $\Sigma \leftarrow 0$ 
for each visible vertex  $v$  on an unvisited ray do
  reset the span list

  for each polygon  $P$  which intersects the ray (cone)  $r\vec{v}$  do
    if  $r\vec{v}$  intersects the interior of  $P$  then
      Add a full angular span for  $P$  at the depth  $z_P$  of the intersection

    else if  $r\vec{v}$  intersects vertex  $i$  of  $P$  then
      add a span for vertex  $i$  of  $P$  (as in Figure 5(a)) at the depth of the intersection
      mark the ray through  $(x, y)$  as visited

    else if  $r\vec{v}$  intersects edge  $i$  of  $P$  then
      add a span for edge  $i$  of  $P$  (as in Figure 5(b)) at the depth of the intersection
      mark the ray through  $(x, y)$  as visited
    end if
  end for

  for each span  $s$  do
     $\Sigma \leftarrow \Sigma + (s.M - \text{prev}(s).M)F(x, y, \tan s.\theta_1)$ 
  end for
end for
```

3.4 Conjunctive Vertices and Bookkeeping

Although the angular span method properly handles conjunctive vertices it creates a new problem: the contributions of a conjunctive vertex could be included more than once. For example, if two intrinsic vertices lie on the same ray, the contribution for the resulting conjunctive vertex will be added when the first vertex is traced, then again when the second vertex is traced. Floating-point imprecision complicates this and is discussed in the next section.

In polyhedral environments, where all the objects are closed solids bounded by outward-facing polygons, only silhouette edges and vertices need be examined in the inner loop of the algorithm. However, all vertices and edges are shared, so much more bookkeeping of visited vertices is required unless the environment has more structure than a simple list of polygons. If a winged-edge data structure is used, for example, the vertices and edges are separate data entities, so conjunctive vertices occur only when distinct *shared* intrinsic vertices and/or apparent vertices appear to coincide.

In less structured polygonal environments the back faces of two-sided polygons may be visible and the resulting angular spans have the opposite direction with respect to the contours. In the cases of shared edges and vertices, all the incident polygons will have the same depth. Extra information (such as the normal to the face) is required to assure an invisible polygon does not incorrectly contribute a span. The latter is also an issue for non-convex vertices of closed polygons.

4 Implementation and Efficiency Issues

The naive vertex tracing algorithm does not have good asymptotic behavior. Assuming there are no efficiency structures for ray tracing the scene polygons, the running time could be as large as cubic in the number of scene polygons N (assuming a small

upper bound on the number of vertices per polygon) due to the N^2 comparisons of edges to find apparent vertices, and a trace time of $O(N)$ for each vertex. Both can be significantly improved.

Most graphics systems with large numbers of polygons have some efficiency structure already built in. Ray tracing can be certainly made sub-linear, and in the proper environments can have a logarithmic expected running time. If this is the case, the bottleneck will be the N^2 comparison of the scene edges to find apparent vertices. One solution is to use a bounding volume hierarchy: if two volumes do not appear to intersect, then none of their contents can appear to intersect either. Our implementation use a bounding-spheres hierarchy, as the test for apparent intersection is very simple and fast. In our implementation we put a bounding sphere hierarchy on both the faces and the edges of the objects in the scene to accelerate ray tracing and the apparent intersection tests.

4.1 Single-Source Environments

In direct lighting, where there is only one emitting source, several improvements are possible. First, there are really only three distinct depths, for the background, source, and blocker which can be discretely represented. More significantly, only source vertices and vertices which otherwise appear inside the source need be traced. Performance can be improved by *shaft culling* [11] the scene against the source and only tracing unculled polygons. Additionally the entire scene need not be clipped on the viewing plane—only the source polygon need be clipped.

4.2 Radiosity and Subdivided Environments

In radiosity systems the intrinsic scene polygons are subdivided into many smaller child polygons, often to the extent that the original polygons become vastly outnumbered. This can be exploited in a number of ways by the vertex tracing algorithm. First of all, the ray tracing phase is faster because only the parent polygons need to be ray-traced for visibility. Second, only edges coincident with silhouette edges of the original polygons can appear to intersect, so the number of tests for edge intersections is reduced. Finally, the vast majority of intrinsic vertices are vertices shared by sub-polygons. There is no background polygon in this case, so the angular spans will simply be the edges of the incident sub-polygons and these are straightforward to evaluate.

5 Floating-Point Imprecision and Cone Tracing

The discussion up to now has been entirely mathematical. In an implementation we are forced to contend with the anomalies of finite precision floating-point arithmetic. True conjunctive vertices generally do not occur. Instead the incident edges and vertices will tend to intersect each other at nearby points, or miss each other outright, resulting in erroneous angular spans. Also, nearly parallel edges can result in extraneous apparent intersections.

We solve the conjunctive vertex problem using a variant of *cone tracing* [1]. Rather than tracing a ray through a vertex, a thin cone is traced instead. All faces, edges and vertices which intersect the cone are “snapped” to the axis, forming an approximate conjunctive vertex (Figure 6). Cone tracing is of course more expensive than ordinary ray tracing. However, a second advantage of using bounding spheres is that the cone-sphere intersection test is fast, so tracing a small cone through the interior nodes of

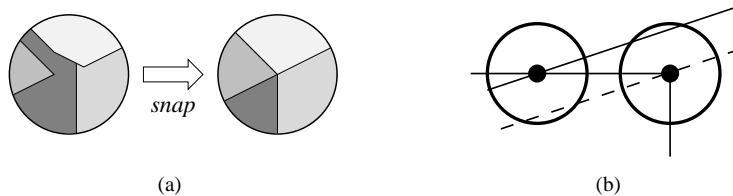


Fig. 6. (a) The view of a nearly conjunctive vertex down the axis of the cone; the vertices and edges are “snapped” to the axis. (b) Snapping asymmetry: the line is snapped to the intrinsic vertex cone, but the vertex is not snapped to the apparent vertex cone.

the hierarchy tree is not significantly more expensive than tracing a ray. Nonetheless, testing cone-polygon intersection is a good deal slower than testing ray-polygon intersection. On the whole, our implementation is slowed roughly by a factor of three from tracing cones rather than rays.

Cone tracing does not completely solve the problem of conjunctive vertices. Extra bookkeeping is required to prevent a conjunctive vertex from being counted more than once. Our implementation stores a flag along with each intrinsic vertex that is set when it is visited, or snapped to another vertex. For vertex-edge and edge-edge pairs we use a hash table which stores pairs of pointers. Each pair of objects, including all the edges incident on the intrinsic vertices, are stored in the hash table. The table and vertex flags must then be consulted before tracing a new vertex. The situation becomes even more complicated if a single edge is snapped at more than one point.

Notice that the tables need only be consulted if a conjunctive vertex has been found, and they are found automatically because the cone for each vertex must be traced through the scene anyway. Our implementation starts by assuming there are no conjunctive vertices. If an “unexpected” vertex or edge is found during a simple cone trace, then the evaluation is handed off to a slower more rigorous version which handles the conjunctive vertices properly. Pixels with conjunctive vertices number from a few to a few hundred in a typical scene, depending on the cone nape angle.

This brings up the issue of what nape angle to use. If the angle is too large, there will be too many conjunctive vertices, and larger snaps incur larger approximation error. If the angle is too small, numerical underflow problems surface. Depending on the scene, a nape angle of between 10^{-6} and 10^{-7} (radians) seems to work well.

6 Results

So far we have used the vertex tracing algorithm primarily as a “plug-in” to a ray tracer, as a function to compute lighting. The 512×512 pixel images in Figure ?? were all created using this method, with one sample per pixel. The running times include both setup and rendering time, including top-level ray tracing. All the images were rendered on a single-processor SGI Mips 195MHz R10K workstation with 512 Mbytes of RAM.

We have found that direct lighting in scenes involving a few hundred reasonably well distributed polygons can generally be rendered in under 30 seconds. The bottleneck occurs when there is a lot of shadow interaction, and this happens when the objects appear in front of each other or the source subtends a large solid angle.

Radiosity reconstruction using the algorithm is significantly slower, but faster than gathering from each patch as a single source. One advantage of the algorithm is that the

coarse “solution”, found in this case by repeated gathering, can be found very quickly as it involves gathering at a relatively small number of isolated points on the surfaces.

It is interesting to note that the aliasing artifacts on the scene geometry, due to the lack of super-sampling, do not appear on the shadow edges; those are already “soft” due to the laws of physics.

7 Conclusion

In this paper we have presented an alternative to Lambert’s formula for the exact evaluation of irradiance due to uniformly emitting polygons. The expression is formulated in terms of the local behavior of the edges at the vertices projected onto an image plane. We described an algorithm exploiting this formulation, based on vertex tracing using angular spans at the vertices, applicable to direct lighting and radiosity gathering. The algorithm is relatively simple, incurs a low overhead, and is likely to fit into existing radiosity systems and their efficiency structures. The details of the efficiency of the algorithm have been discussed only loosely, partially because the performance (and implementation, for that matter) depends greatly on the structure available on the geometry of environment.

The algorithm in its purest form is for computing irradiance at isolated points. Performance could certainly be improved by exploiting coherence, or applying some of the many efficient visibility computation schemes in the literature. But there are advantages to having an algorithm tuned for diverse sampling. First, often this is all one needs. Radiosity solutions can be improved by computing exact values at certain points where the geometry becomes messy. Also, numerical derivative and integration methods often rely on exact function values at certain points. Finally, the independence of the algorithm makes it naturally suited to parallelization.

We do not expect this algorithm to immediately replace existing methods for shadow calculation. However, we hope the simplicity and relatively low overhead of the method will make it attractive in circumstances where other methods become cumbersome, such as in situations where it is undesirable to compute the entire visibility mesh or the entire visibility map.

Acknowledgments

This work was supported in part by DARPA (F33615-96-C-5621) and the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219). Comments from Brian Smits, Bill Martin, Elaine Cohen and the reviewers were helpful in producing this paper and are gratefully acknowledged.

References

1. John Amanatides. Ray tracing with cones. In Hank Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 129–135, July 1984.
2. James Arvo. The irradiance Jacobian for partially occluded polyhedral sources. In *Siggraph '94*, pages 343–350, July 1994.
3. James Arvo. *Analytic Methods for Simulated Light Transport*. PhD thesis, Yale University, 1995.
4. P. Atherton, K. Weiler, and D. Greenberg. Polygon shadow generation. volume 12, pages 275–281, August 1978.

5. Michael F. Cohen and Donald P. Greenberg. The hemi-cube: a radiosity solution for complex environments. *Computer Graphics*, 19(3):31–40, July 1985. ACM Siggraph '85 Conference Proceedings.
6. Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Cambridge, MA, 1993.
7. Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 137–45, jul 1984.
8. George Drettakis and Eugene Fiume. A fast shadow algorithm for area light sources using backprojection. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 223–230. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
9. G. Drettakis. *Structured Sampling and Reconstruction of Illumination for Image Synthesis*. PhD thesis, University of Toronto, 1994.
10. Frédo Durand, George Drettakis, and Claude Puech. The visibility skeleton: A powerful and efficient multi-purpose global visibility tool. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 89–100. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
11. Eric Haines and John Wallace. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering*, 1991.
12. David Hart, Philip Dutré, and Donald P. Greenberg. Direct illumination with lazy visibility evaluation. *Proceedings of SIGGRAPH 99*, pages 147–154, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.
13. Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992.
14. Daniel Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39, November 1992.
15. T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, pages 23–30, July 1985.
16. H. Plantinga and C.R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, 1990.
17. François Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, 1994.
18. Cyril Soler and François X. Sillion. Fast Calculation of Soft Shadow Textures Using Convolution. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 321–332. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
19. Michael M. Stark. Vertex-based formulations of irradiance from polygonal sources. Technical Report UUCS-00-012, Department of Computer Science, University of Utah, May 2000.
20. Michael M. Stark, Elaine Cohen, Tom Lyche, and Richard F. Riesenfeld. Computing exact shadow irradiance using splines. *Proceedings of SIGGRAPH 99*, pages 155–164, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.
21. Seth Teller and Pat Hanrahan. Global visibility algorithms for illumination computations. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 239–246, 1993.
22. John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 315–324, July 1989.
23. Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques*. ACM Press, 1992.
24. Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, November 1990.

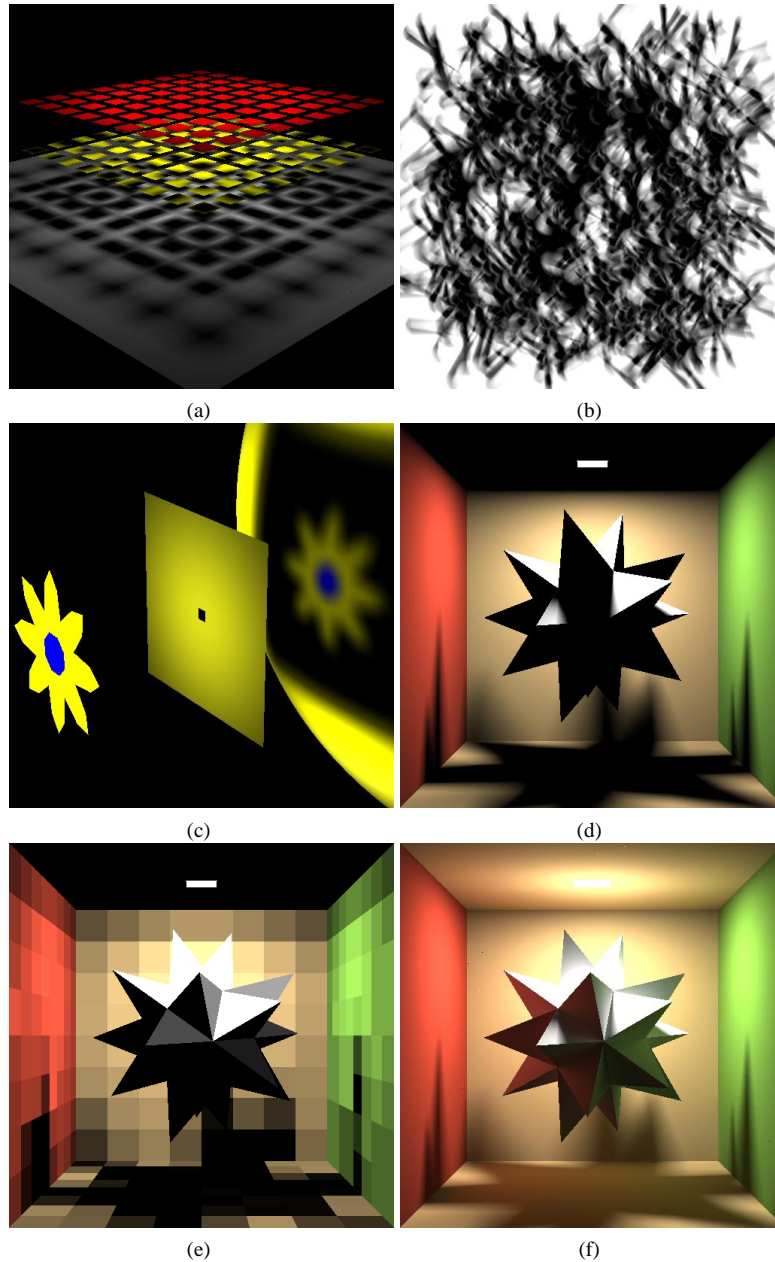


Fig. 7. 512×512 single-sample ray-traced images rendered using the algorithm. (a) Evenly arranged occluding objects (32s) producing interference patterns. (b) A particularly difficult scene (shadow image only) of 1000 thin triangles under a large crescent-shaped source (280s). Notice the shape of the source creeping in. (c) a multi-polygon source (98 seconds) and a more pronounced “pinhole” effect. (d) Direct lighting of a self-shadowing object (21s). (e) A coarse one-bounce approximation (321 patches, 1s), and (f) the indirect lighting reconstruction (48m). Image (f) was rendered ignoring conjunctive vertices, yet there are only a few erroneous pixels.