

Exact Radiosity Reconstruction and Shadow Computation 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 introduces an alternative to Lambert's formula, an analytical expression which independent of the vertex order of the polygon. In 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, where no vertex ordering is required. The method is particularly applicable to radiosity reconstruction, in which all the scene polygons are diffuse emitters, and also in environments where efficiency structures have already been established.

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, all of these problems eventually 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 [21] used FFT hardware 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 [9] first partitions the receiver into regions of topologically equivalent visibility, then the scene is efficiently backprojected onto the emitting polygon.

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

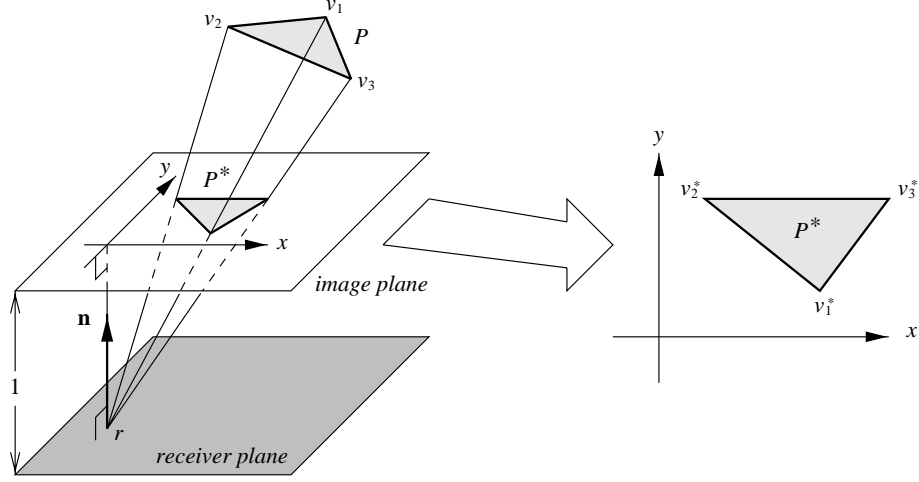


Fig. 1. To apply Green's theorem, the polygon P is projected onto an image plane parallel to the receiver plane, one unit above. The origin of the coordinate system of the image plane lies directly above the point of evaluation r on the receiver. The projection induces a reversal of orientation for front-facing polygons.

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:

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

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. [7]). 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 2(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 1. This projection does not change the irradiance at r [1].

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

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

(In the case of a polygonal receiver, \mathbf{u} may be obtained taken as 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 with coordinates (x_i, y_i) .

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

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

To evaluate the integral we will use Green's theorem, which reduces the double integral to a contour integral on the boundary of P^*

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

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, 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) 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 therefore 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

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

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 thereby arrive at the final form of the solution

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

The function F is evaluated as

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

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, so there is nothing to cancel the first term. The terms do cancel if neither incident edge is vertical.

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. 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 vertices must be clipped some small height above the plane. (Otherwise the foregoing computation would have to be evaluated in the real projective plane.) 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. Finally, 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 never less than 1.

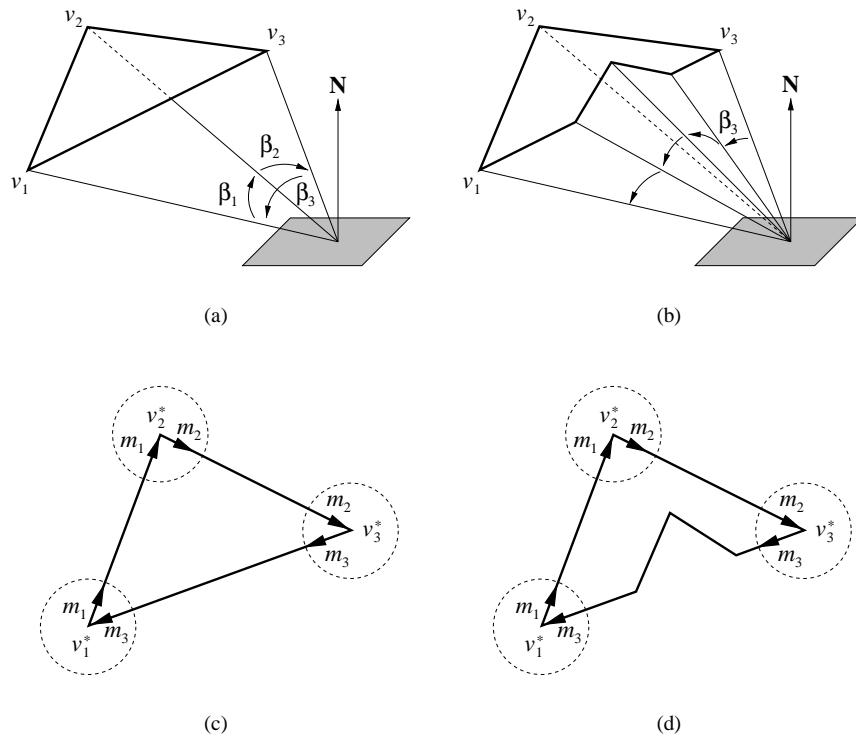


Fig. 2. (a) The geometry for Lambert's formula. (b) The angles β_i depend on the edges, so if part of the polygon is clipped by an occluder, the terms associated with the vertices of the affected edges have different values. (c) Using Green's theorem in the image plane produces a formula in terms of 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.

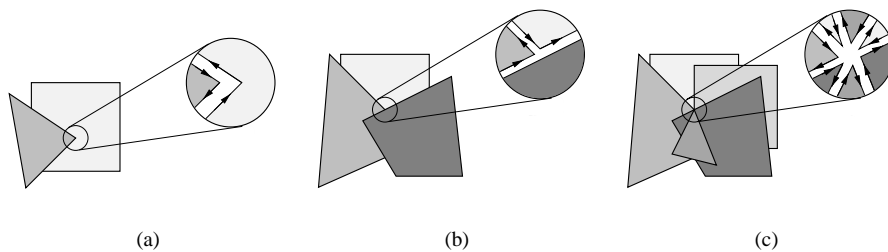


Fig. 3. Common cases of vertex behavior: (a) an intrinsic vertex against a background polygon, (b) an apparent vertex caused by the intersection of two edges against a background polygon, (c) a particularly unfortunate conjunction of three edges and one polygon vertex.

3 Vertex Tracing and Angular Spans in Polygonal Environments

Equation (6) provides a method for computing the irradiance due to a polygon which is no longer constrained by the vertex order. 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 vertices to compute the total irradiance.

The projection of the scene onto the image plane is equivalent to the construction visibility map [18, 22]. That is, once the visibility map is constructed, our formula may be directly applied to compute the irradiance. The formula applies equally well to an approximate visibility map, such as the one developed by Stewart and Karkanis [22]. However, the visibility map by definition include 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. Later we discuss optimization methods. The algorithm provides a simple, low-overhead and potentially efficient method for exact irradiance computation which is likely to fit easily into existing illumination computation systems. Furthermore, 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 and to radiosity reconstruction, where every polygons in the scene is assumed to emit.

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 are M and M_B of the foreground and background polygons, 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 frontmost polygon because the incoming and outgoing slopes are the same.

Sub-figures (a) and (b) of figure 3 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 [16, 10]. Figure 3(c) shows an example of a conjunctive vertex from three apparent vertices and one intrinsic vertex. But despite the complexity of the interaction, the local behavior is still sufficient to com-

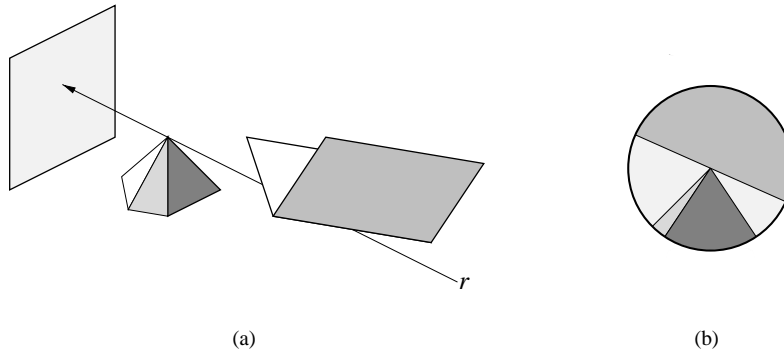


Fig. 4. (a) Tracing a conjunctive vertex, and (b) the resulting angular spans.

pute 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, and an slope/quadrant pair to represent the angular boundary.) Angular spans are similar to linear spans used in scan-line rendering (*e.g.* [24]) except that the opposite ends of a scan line coincide. 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.

3.3 Naive Vertex Tracing

To determine the irradiance, all the visible vertices in the scene must be examined, which means the visible vertices must be found. The method accomplishes this task by tracing the ray from the point of evaluation r through the vertex. This way, all the participating polygons in a conjunctive vertex are found during the visibility test. The vertex tracing algorithm works by constructing an angular span list for each visible vertex (intrinsic or apparent). Then, when the span list is completed, the contribution

for the vertex, consisting of contributions from all the incident polygons with 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.

The list is constructed by incrementally adding a span for each vertex or edge incident on the ray from the point of evaluation r and the defining vertex. The depth value z for each span comes from the distance along the ray from r to the point of intersection. 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) , $\theta_1 = \arctan_2(y_{i-1} - y_i, x_{i-1} - x_i)$ and

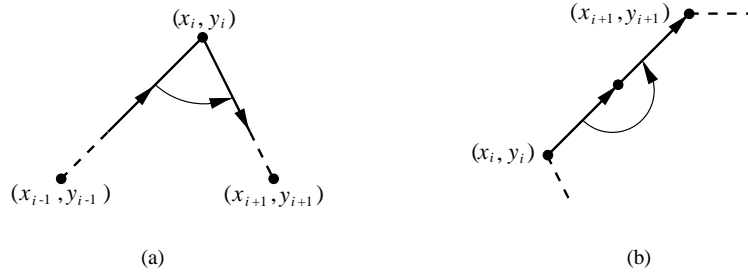


Fig. 5. (a) the span for an intrinsic vertex, (b) the span for an incident edge.

$\theta_2 = \arctan(y_{i+1} - y_i, x_{i+1} - x_i)$. For an edge span, $\theta_1 = \arctan(y_i - y_{i+1}, x_i - x_{i+1})$ and $\theta_2 = \theta_1 + \pi$.

How the span list is initialized at a particular vertex v depends on the type of the vertex. For an intrinsic vertex, a span for the vertex is computed and inserted into the list. For an apparent vertex, a span is added for each of the two edges. In both cases, the remaining angular region is taken as a span with depth field z set to ∞ , the “background” depth.

If the ray intersects the interior of a polygon in front of v (in front of the more distant edge in the case of an apparent vertex), the computation for v is finished, as v is not visible. The nearest polygon the ray hits behind v (if there is one) is stored as the background polygon B , and the emission constant for B is used for spans having the background depth of ∞ .

Once the angular span list has been fully constructed, the contribution of the incident polygons 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.

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 will be included more than once. For example, if two intrinsic vertices line on the same ray, the contribution for the resulting conjunctive vertex will be added when the first vertex is examined, then again when the second vertex is examined. There are two obvious solutions. One is to mark vertices

Algorithm 1 General vertex tracing

 $\Sigma \leftarrow 0$ **for each** visible vertex v **do**reset the span list, add a span for v , or the edges forming v set the background polygon B to none, $z_B \leftarrow \infty$ set z_v to the distance to v (or the distance to the further edge)**for each** polygon P which intersect the ray $r\vec{v}$ **do****if** $r\vec{v}$ intersects the interior of P **then****if** $z_P < z_v$ **then**vertex v is covered, there is no contribution**else if** $z_P > z_B$ **then** $B \leftarrow P$ **end if****else if** $r\vec{v}$ intersects edge i of P **then**add a span for edge i of P mark (x, y) as visited**else if** $r\vec{v}$ intersects vertex i of P **then**add a span for vertex i of P mark (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**

(in the image plane) as “visited”, another is to determine how many times the ray will be visited, by counting the number of distinct inherent vertices there are along the ray, then divide the contribution by this number. The first method can be done in essentially constant time using a bucketing or hashing scheme at the expense of extra memory. The second method incurs no more memory, but there will be wasted time in re-visiting the incident vertices. The multiplicity of the contribution is easily computed; if there are N incident intrinsic vertices and E incident edges, the multiplicity is $N + E(E - 1)/2$. In polyhedral and deeply subdivided environments, most vertices will be shared so the second option is likely to be too wasteful.

The pseudocode for a sketch of the general algorithm is shown in algorithm 1.

3.5 Polygonal and Polyhedral Environments

In polyhedral environments, where all the objects are 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. In less structured polygonal environments, the back faces of two-sided polygons may be present. This complicates the algorithm in several ways. First, a back-face polygon will not incur the reversal of orientation, so the contours of the projected polygon will be counter-clockwise instead of clockwise. Second, in the cases of shared edges and vertices, all the incident polygons will have the same depth. Extra depth information (such as a pointer to the polygon) is required to assure an invisible polygon does not incorrectly contribute a span. This last problem magically vanishes in polyhedral environments, because back-faced polygons are never visible, and can be excluded from consideration in the inner loop.

If the geometry of the scene has more structure than simply a list of polygons, such as a winged-edge data structure, the performance of the algorithm can be significantly improved. First of all, the vertices and edges are separate data entities, so shared vertices and edges need only be examined once. In this case, conjunctive vertices can be defined as those where separate *shared* intrinsic vertices and/or apparent vertices appear to coincide. Conjunctive vertices are much more rare in this situation, which will reduce the bookkeeping cost to practically nothing.

4 Efficiency in Radiosity and Other Subdivided Environments

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 intrinsic vertices, and a trace time of $O(N)$ for each vertex. In a typical configuration, there are probably more like $\Theta(N)$ intrinsic and apparent vertices, so the performance even in the naive approach is likely to be something like a quadratic, but even this is too slow to be practical for large environments.

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. A bounded volume hierarchy can help with this: if two volumes appear to not intersect, then none of their contents can intersect either. A bounding-spheres hierarchy is likely to work well for this, as the test for apparent intersection is very simple and fast.

4.1 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 greatly reduced, even in the absence of a bounding volume hierarchy. 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, which are straightforward to evaluate.

5 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. The formulation can be used to form a hybrid approach in the computation of a visibility map or in the backprojection algorithm instance computation [9]. We described an algorithm based on vertex tracing using angular spans to store polygon behavior at the vertices can exploit this formulation. The algorithm is particularly applicable to polyhedral environments where all or most of the polygons emit. Furthermore, the algorithm is relatively simple, incurs a low overhead, and fits in nicely with 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.

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

References

1. J. Arvo. *Analytic Methods for Simulated Light Transport*. PhD thesis, Yale University, 1995.
2. James Arvo. The irradiance Jacobian for partially occluded polyhedral sources. In *Siggraph '94*, pages 343–350, July 1994.
3. P. Atherton, K. Weiler, and D. Greenberg. Polygon shadow generation. volume 12, pages 275–281, August 1978.
4. Daniel R. Baum, Holly E. Rushmeier, and James M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 325–334, July 1989.
5. A. T. Campbell and Donald S. Fussell. Adaptive mesh generation of global diffuse illumination. *Computer Graphics*, 24(3):155–164, August 1990. ACM Siggraph '90 Conference Proceedings.
6. 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.

7. Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Cambridge, MA, 1993.
8. Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 137–45, jul 1984.
9. George Dretakkis 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.
10. G. Drettakis. *Structured Sampling and Reconstruction of Illumination for Image Synthesis*. PhD thesis, University of Toronto, 1994.
11. 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.
12. Steven J. Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Computer Graphics*, pages 221–230, August 1993. ACM Siggraph '93 Conference Proceedings.
13. Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.
14. Paul Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–226, May 1992.
15. Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 67–74. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
16. Daniel Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications*, 12(6):25–39, November 1992.
17. T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and inntereflection. In *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, pages 23–30, July 1985.
18. H. Plantinga and C.R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, 1990.
19. Peter Schröder and Pat Hanrahan. On the form factor between two polygons. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 163–164, 1993.
20. F. X. Sillion and C. Puch. *Radiosity and Global Illumination*. Morgan Kaufman, 1994.
21. Cyril Soler and François Sillion. Fast calculation of soft shadow textures using convolution. In *Siggraph '98*, pages 321–332. ACM SIGGRAPH, New York, 1998.
22. A. James Stewart and Tasso Karkanis. Computing the approximate visibility map, with applications to form factors and discontinuity meshing. *Eurographics Workshop on Rendering*, June 1998.
23. 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.
24. Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques*. ACM Press, 1992.
25. K. Weiler and K. Atherton. Hidden surface removal using polygon area sorting. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2):214–222, July 1977.
26. Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, November 1990.