Approximating the Location of Integrand Discontinuities for Penumbral Illumination with Area Light Sources

Marc J. Ouellette Eugene Fiume
Department of Computer Science
University of Toronto, Toronto, Canada
Toronto, ON M5S 3G4 Canada
e-mail: {vv1|elf}@dgp.toronto.edu

Abstract. The problem of computing soft shadows with area light sources has received considerable attention in computer graphics. In part, this is a difficult problem because the integral that defines the radiance at a point must take into account the visibility function. Most of the solutions proposed have been limited to polygonal environments, and require a full visibility determination preprocessing step. The result is typically a partitioning of the environment into regions that have a similar view of the light source. We propose a new approach that can be successfully applied to arbitrary environments. The approach is based on the observation that, in the presence of occluders, the primary difficulty in computing the integral that defines the contribution of an area light source, is that of determining the visible domain of the integrand. We extend a recent shadow algorithm for linear light sources in order to calculate a polygonal approximation to this visible domain. We demonstrate for an important class of shadowing problems, and in particular, for convex occluders, that the shape of the visible domain only needs to be roughly approximated by a polygonal boundary. We then use this boundary to subdivide an area light source into a small number of triangles that can be integrated efficiently using either a deterministic solution, or a low degree numerical cubature.

Keywords: numerical cubatures, random seed bisection, area sources, soft shadows.

1 Introduction

The efficient computation of soft shadows due to area light sources remains one of the most challenging problems in computer graphics. The synthesis of realistic images depends on this computation; however, the discontinuities that arise, and the complexity of the visibility computation itself, can conspire to create renderings that are either too slow to compute or unsatisfactory in appearance. The problem of determining the direct illumination reaching a specific point from a given light source can be separated into two tasks: determining the visible portion of the source from that point, and calculating the reflected light due to this visible portion. In this paper, we propose a new solution to the first half of this problem, based on providing a polygonal approximation to the visible portion of the source. The resulting integrals can be solved efficiently using either an analytical solution for diffuse surfaces [10], or a numerical cubature of low degree. This algorithm naturally permits nonpolyhedral scene geometries.

In the general setting of area light sources, three main techniques have been used to determine the visibility of a source. The earliest techniques determined visibility of a source by either approximating it by point light sources [1], or by point sampling the source itself [4, 15]. This approach is subject to aliasing if too few samples are used. Images of a higher quality can be achieved using algorithms that use shadow volumes and/or discontinuity meshing to determine the exact visibility of a source [2, 3, 5, 7, 9, 14]. These techniques are very expensive and have only been designed to compute exact visibility for polygonal environments. Finally, shadow maps have also been used to texture map soft shadows, either by pre-calculating an approximation using multiple light source samples and combining them in an accumulation buffer [6], or, most recently, by convolving source and occluder images to produce a soft shadow texture[12, 13]. The latter algorithm can produce convincing shadows for environments with arbitrary types of objects; however, correct shadows are sometimes difficult to produce, for example, if a large occluder touches a receiver.

In this paper, we use *Random Seed Bisection* (RSB) and the *Two Discontinuity Finding* (TDF) algorithm developed in [11] to approximate the visible portion of a polygonal light source. RSB and TDF efficiently find the approximate location of (two or fewer) discontinuities in a one-dimensional integrand caused by (arbitrary) occluders. Given a triangular source, we use RSB and TDF to determine the number and approximate location of the discontinuities caused by occluders along each edge of the source. Given the number of discontinuities along each edge, we classify the triangle into one of six edge visibility configurations. Depending on the resulting configuration, we may compute one or two additional interior discontinuities. We then approximate the visible portion of the triangle using the polygonal boundary defined by the vertices and the discontinuities, and partition the visible domain into sub-triangles. Finally, we calculate the integral over each sub-triangle using either an analytical solution (in the case of a diffuse emitter and a diffuse surface), or a low degree numerical cubature.

This paper is organized as follows: in Section 2, we review the one dimensional discontinuity finding algorithms of [11]. In Section 3, we present the algorithm for calculating a polygonal approximation to the visible portion of a triangular light source. In Section 4, we present the extension of the algorithm to quadrilateral and general polygonal sources. In Section 5, we present results for triangular sources and a discussion of the algorithm developed. Finally, in Section 6, we present our conclusions.

2 One Dimensional Discontinuity Finding Algorithms

In this section, we briefly review the one dimensional discontinuity finding algorithms that were introduced in [11]. We want to find the approximate location of the discontinuities in a visibility function $V(x)$ defined over $[0,1]$ such that $V(x)=1$ if x is visible, and $V(x)=0$ otherwise.

2.1 Random Seed Bisection

If V has at most one discontinuity in $[0,1]$, then V has one discontinuity $\lambda \in [0,1]$ iff $V(0) \neq V(1)$. In such a case, V is a step function, and given a tolerance ε, we calculate $\tilde{\lambda}$, the approximate location of λ, such that $\|\tilde{\lambda} - \lambda\| \leq \varepsilon$.

Suppose that V has exactly one discontinuity over an interval $[a,b]$ such that $V(a) \neq V(b)$. Let $m \in [a,b]$. If $V(a) \neq V(m)$, then V has a discontinuity over $[a,m]$, otherwise V has a discontinuity over $[m,b]$. The *Random Seed Bisection* (RSB) algorithm is based on this observation. In RSB, $[a,b]$ is initially set to $[0,1]$, and m is set to a random seed taken from a uniform distribution over $[0,1]$. At each iteration i, $V(m)$ is evaluated and compared to $V(a)$. The appropriate subinterval is chosen as the $(i+1)^{\text{st}}$ interval, and

m is set to its midpoint. The iteration terminates when the $||a-b|| < 2\varepsilon$, and we set $\tilde{\lambda} = m$. The expected error of RSB after n iterations is $(2/3)(1/2)^{n-1}$, which is almost optimal.

The function $E_n(\tilde{\lambda})$ is the expected value of $\tilde{\lambda}$ in terms of λ and of the number of iterations n. For RSB, $E_n(\tilde{\lambda})$ is a continuous function of λ over $[0,1]$. Because $E_n(\tilde{\lambda})$ is continuous, we can use RSB to efficiently find the approximate location of a discontinuity for an integrand in a linear light source, while avoiding banding problems inherent to purely deterministic methods, as was shown in [11].

2.2 Two Discontinuity Finding Algorithm

Suppose that V has at most two discontinuities in $[0,1]$. If $V(0) = V(1)$, then V has either zero or two discontinuities in $[0,1]$. If we can find a point $m \in (0,1)$ such that $V(0) \neq V(m)$, then we can use RSB to find a discontinuity on each side of m. The *Two Discontinuity Finding* (TDF) algorithm uses heuristics based on scene coherence to determine if such a point m exists. The heuristics either succeed in finding such a value m and we find two discontinuities, or they fail and we conclude that there are no discontinuities. Let P be the number of discontinuities detected in the integrand for the previous (and adjacent) pixel, and let these discontinuities be p_1 and p_2, if they exist. We have different heuristics based on the value of P.

State $P = 0$. Let ν be a user-specified tolerance, and let $x_{-2} = 0$ and $x_{-1} = 1$. Choose a random value $x_0 \in [0,1]$ and determine $V(x_0)$. If $V(x_0) \neq V(0)$, then let $m = x_0$ and return. Otherwise, at each step i, choose x_i as the midpoint of the largest subinterval $[x_a, x_b]$ such that $a, b \in [-2, i-1]$ and there is no $c \in [-2, i-1]$ such that $x_c \in (x_a, x_b)$, with ties being broken randomly. The iteration stops as soon as some $V(x_i) \neq V(0)$, in which case we let $m = x_i$ and return. If the largest subinterval becomes smaller than ν, we stop and conclude that, probabilistically,[1] we have zero discontinuities, and return a failed status. We refer to this algorithm as the *Voronoi Search* (VS).

State $P = 1$. Since the previous pixel had one integrand discontinuity p_1, we had $V(0) \neq V(1)$. Since now $V(0) = V(1)$, one of the end points has changed visibility, say v_i. It is likely that if a change of visibility still occurs in the integrand, it does so between v_i and p_1. We let $m = (p_1 + v_i)/2$ and compute $V(m)$. If $V(m) \neq V(0)$, we return successfully. Otherwise, we use VS to look for a change of visibility between p_1 and v_i.

State $P = 2$. The integrand for the previous pixel had two discontinuities p_1 and p_2. It is likely that if we still have two discontinuities, the midpoint $m = (p_1 + p_2)/2$ will be such that $V(m) \neq V(0)$. If this is the case, return successfully. Otherwise, we use VS to look for a change of visibility in $[0,1]$.

3 Triangular Shadow Algorithm

We now present an algorithm for finding the approximate shape of the visible portion of a triangular light source T for an important class of occluders. We define

[1] Specifically, there is no gap $G \subseteq [0,1]$ of size $||G|| > \nu$ such that $V(x) \neq V(0), \forall x \in G$.

the class of *Occluders Causing Two or fewer Discontinuities In Any Linear Subdomain* (OCTDIALS) to be the class of occluders that cause at most two discontinuities along any linear subdomain of T. Given T and a point to be shaded P, an occluder O is of class OCTDIALS(T,P) iff, given any line segment $L \subseteq T$, O causes at most two discontinuities in the visibility function of L, as seen from P. In a sense, this class includes any object, or collection of objects, that causes a locally convex occlusion of T as viewed from P. This class includes, but is not limited to, convex occluders and convex "visibility holes." In the remainder of this section, we will assume that any occluder O is of the class OCTDIALS(T,P).

We define the visibility function $V(\mathbf{x})$, $\forall \mathbf{x} \in T$, such that $V(\mathbf{x}) = 1$ if $\mathbf{x}$ is visible from P, and $V(\mathbf{x}) = 0$ otherwise. We can use RSB and TDF to determine efficiently the number and approximate location of discontinuities in $V(\mathbf{x})$ along any line segment $L \subseteq T$, and in particular, along the edges of T. Given the location of discontinuities along the edges of T, the location of T's vertices, and, if necessary, the location of one or two additional discontinuities inside T, we can approximate the visible domain of T with a polygonal boundary. We now formulate this approximation algorithm by examining the edge configurations and the configuration transitions.

3.1 Edge Configurations

The first step in approximating the visible domain of T is to find the number and approximate location of the discontinuities in $V(\mathbf{x})$ along the edges of T. Let the three vertices of T be $\mathbf{v_0}$, $\mathbf{v_1}$, and $\mathbf{v_1}$. Since the edges share these vertices, we first calculate $V(\mathbf{v})$ at each vertex $\mathbf{v}$. For each edge defined by a pair of vertices $\mathbf{v_i}$ and $\mathbf{v_j}$, we then compare $V(\mathbf{v_i})$ to $V(\mathbf{v_j})$. If the values are different, we use RSB to find the single discontinuity along this edge, otherwise we use TDF to find either zero or two discontinuities.

We classify T according to its *edge configuration*, that is, according to the number of discontinuities found along each edge. There are only two possible classes of vertex visibility: either all the vertices have the same visibility, or one vertex has a visibility that is distinct from the other two. This leads to a natural classification of the possible edge configurations.

If all vertices have the same visibility, then every edge has either zero or two discontinuities. We can then classify T into one of the following four configurations:

C_{000} : All three edges have zero discontinuities.
C_{002} : One edge has two discontinuities, two edges have zero discontinuities.
C_{022} : Two edges have two discontinuities, one edge has zero discontinuities.
C_{222} : All three edges have two discontinuities.

If a vertex $\mathbf{v}$ has a different visibility than the other two, then two edges have one discontinuity, and the other edge either has zero or two discontinuities. We can then classify T into one of the following configurations:

C_{011} : Two edges have one discontinuity, one edge has zero discontinuities.
C_{112} : Two edges have one discontinuity, one edge has two discontinuities.

If the visibility of the triangle is of type C_{000}, we will classify T as fully occluded if its vertices are occluded, and as fully visible otherwise. This assumption will fail only if the occluder O is entirely contained within the tetrahedron defined by P and T. Because of the unlikelihood of this happening, and because the resulting contribution to the penumbra may often be negligible, we have chosen to make this simplifying assumption.

Given a configuration of type C_{112} or C_{222}, we approximate the shape of the blocker by joining discontinuities with non-intersecting lines. These lines partition the triangle into visible and occluded polygons. Given a neighbourhood of pixels that have the same configuration, this type of approximation produces smooth varying shadows.

Unfortunately, given a configuration of type C_{002}, C_{011}, or C_{022}, simply joining the edge discontinuities can lead to dramatic discontinuities in the penumbral shadow. These discontinuities occur at the boundaries with other configurations, and correspond to the traditional discontinuities encountered in discontinuity meshing. To both understand and to alleviate these problems, we must examine the transitions that can happen between the various configurations.

3.2 Configuration Transitions

As we determine visibility from one pixel to the next, the occluder appears to move with respect to the source. We say that a transition pair exists between two configurations C_a and C_b iff it is possible for an occluder to be translated from position α to position β such that:

- At position α, the configuration due to the occluder is C_a.
- At position β, the configuration due to the occluder is C_b.
- For any position $\gamma = \alpha + s(\beta - \alpha)$ such that $s \in [0, 1]$, the configuration due to the occluder is either C_a or C_b.

There are only 8 possible transitions pairs,[2] and these are illustrated in Figure 1.

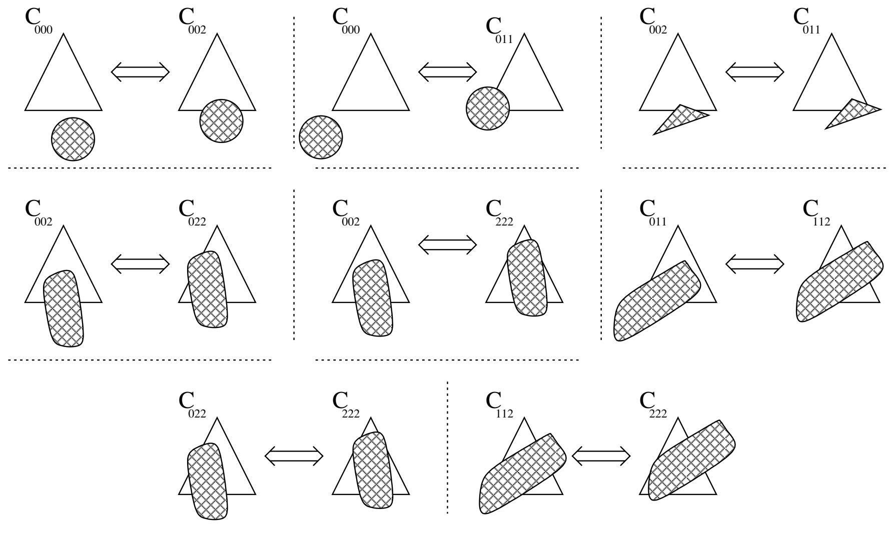


Fig. 1. Configuration Transition Pairs

If we simply use the discontinuities along the edges of T to approximate the shape of the occluder in each of the configurations, some transitions can become *ill-conditioned* (see [8]), in the sense that small changes in the location of the occluder can cause large changes in our approximation of the visible portion. The following transitions are

[2] If the fully-visible assumption about the C_{000} configuration is violated, other transitions are possible.

potentially ill-conditioned: $C_{002} \Leftrightarrow C_{011}$, $C_{002} \Leftrightarrow C_{022}$, $C_{002} \Leftrightarrow C_{222}$, $C_{011} \Leftrightarrow C_{112}$, and $C_{022} \Leftrightarrow C_{222}$. An ill-conditioned transition is illustrated in Figure 2. Notice that the approximation to the visible portion of the source suddenly increases in value in the last transition, as the occluder finally pierces the right edge.

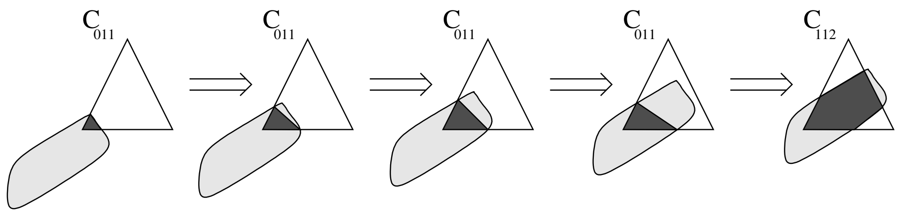


Fig. 2. Ill-Conditioned $C_{011} \Leftrightarrow C_{112}$ transition. Approximate visible domain shown in black.

The problem with the simple approximation is that it is impossible to maintain a consistent approximation across certain configuration transitions. To alleviate ill-conditioning, we must account for the shape of the occluder inside T. This is critical for the *ill-conditioned configurations* C_{002}, C_{011}, and C_{022}. These are the configurations that have one or two edges with zero discontinuities, and for which we have no information about the closeness of the occluder to these edges.

Our goal is to approximate the shape of the occluder such that the approximation varies smoothly across a region of pixels that have the same configuration, and varies continuously at the boundaries where the configuration changes. To achieve this goal, we use effective heuristics to approximate the internal shape of an occluder near one or more edges with zero discontinuities.

3.3 Finding the Closest Point to an Edge

Without loss of generality, we can relabel a triangle T in an ill-conditioned configuration as illustrated in Figure 3. Specifically,

C_{011} : There is a discontinuity $\mathbf{d_1}$ between vertices $\mathbf{v_0}$ and $\mathbf{v_1}$, a discontinuity $\mathbf{d_2}$ between $\mathbf{v_0}$ and $\mathbf{v_2}$, and no discontinuities between $\mathbf{v_1}$ and $\mathbf{v_2}$. See Figure 3(a).

C_{022} : There are two discontinuities $\mathbf{d_0}$ and $\mathbf{d_1}$ between $\mathbf{v_0}$ and $\mathbf{v_1}$, two discontinuities $\mathbf{d_2}$ and $\mathbf{d_3}$ between $\mathbf{v_0}$ and $\mathbf{v_2}$ (with $\mathbf{d_1}$ and $\mathbf{d_2}$ being the discontinuities closest to $\mathbf{v_2}$), and no discontinuities between $\mathbf{v_1}$ and $\mathbf{v_2}$. See Figure 3(b).

C_{002} : There are two discontinuities $\mathbf{d_0}$ and $\mathbf{d_1}$ between $\mathbf{v_0}$ and $\mathbf{v_1}$ (with $\mathbf{d_1}$ being the discontinuity closest to $\mathbf{v_1}$), no discontinuities between $\mathbf{v_0}$ and $\mathbf{v_2}$, and no discontinuities between $\mathbf{v_1}$ and $\mathbf{v_2}$. See Figure 3(c).

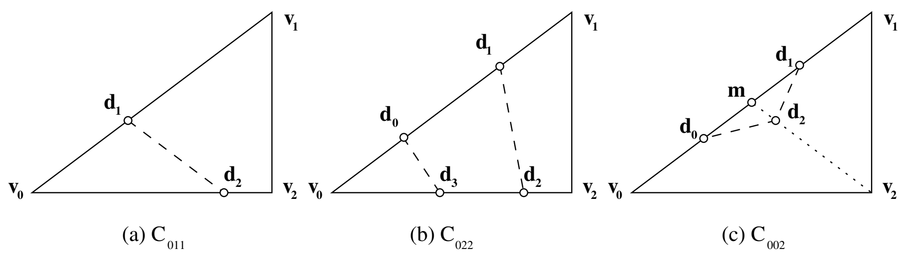


Fig. 3. Relabelling of ill-conditioned triangles, with dashed lines representing linear boundaries.

If a triangle is of type C_{002}, we let $\mathbf{m}$ be the midpoint between the discontinuities $\mathbf{d_0}$ and $\mathbf{d_1}$. Since the occluder is of class OCTDIALS(T,P), then $V(\mathbf{m}) \neq V(\mathbf{v_2})$, and we can use RSB to find a discontinuity $\mathbf{d_2}$ between $\mathbf{m}$ and $\mathbf{v_2}$. The triangle can then be split along the edge $\mathbf{m}$-$\mathbf{v_2}$, which results in two triangles of type C_{011}.

Given that any triangle of type C_{002} can be subdivided into two triangles of type C_{011}, we only need to be able to handle triangles of configuration type C_{011} and C_{022}, to effectively handle all ill-conditioned triangles. Examining Figure 3(a) and Figure 3(b), we see that approximating the internal shape of the visible domain of T can be expressed as follows: given the boundary of the occluder that extends from $\mathbf{d_1}$ to $\mathbf{d_2}$, find the point on this boundary that is closest to the edge $\mathbf{v_1}$-$\mathbf{v_2}$.

Let $\mathbf{v}(f)=\mathbf{v_1}(1-f)+f\mathbf{v_2}$ be a parameterization of the edge $\mathbf{v_1}$-$\mathbf{v_2}$, where $f\in[0,1]$. Let $\mathbf{d}(f)$ be the discontinuity between $\mathbf{v_0}$ and $\mathbf{v}(f)$ that is the furthest from $\mathbf{v_0}$. This is the intersection of the line $\mathbf{v_0}$-$\mathbf{v}(f)$ with the boundary of the occluder that extends from $\mathbf{d_1}$ to $\mathbf{d_2}$, and in fact, $\mathbf{d}(f)$ is the boundary curve. Let $D(f)$ be the perpendicular distance from $\mathbf{d}(f)$ to the edge $\mathbf{v_1}$-$\mathbf{v_2}$. Since the occluder is of class OCTDIALS(T,P), the function $D(f)$ can have only one of three possible shapes:

- Concave upward — if $D''(f) \geq 0$.
- Concave downward — if $D''(f) \leq 0$.
- Straight line — if $D''(f)=0$.

Only if $D''(f) \geq 0$ can D have a minimum at m, with $\mathbf{d}(m)$ closer to the edge $\mathbf{v_1}$-$\mathbf{v_2}$ than both $\mathbf{d_1}$ and $\mathbf{d_2}$. How close $\mathbf{d}(m)$ is to the edge determines the ill-conditioning that would be inherent to an approximation of the visible domain were $\mathbf{d}(m)$ not to be taken into account. Since in practice this is the most common (and most difficult) type of ill-conditioning to detect, we now present an algorithm that determines if D is concave upward, and if so, finds the minimum m and the point $\mathbf{d}(m)$, within a tolerance μ.

In general, the boundary function $\mathbf{d}(f)$ is not explicitly available. However, we can use RSB to calculate $\mathbf{d}(f)$. First, we compute the intersection of the line $\mathbf{v_0}$-$\mathbf{v}(f)$ with the line $\mathbf{d_1}$-$\mathbf{d_2}$, giving the intersection point $\mathbf{i}$. Since the edge $\mathbf{v_1}$-$\mathbf{v_2}$ has no discontinuities, we know that $V(\mathbf{v}(f)) = V(\mathbf{v_1})$. We then compute $V(\mathbf{i})$, and compare it to $V(\mathbf{v}(f))$. If they are the same, the function $D(f)$ is not concave upward, and we cannot find a point on the boundary closer to edge $\mathbf{v_1}$-$\mathbf{v_2}$ than both $\mathbf{d_1}$ and $\mathbf{d_2}$. If they are different, we use RSB to find $\mathbf{d}(f)$, and then compute its perpendicular distance from the edge $\mathbf{v_1}$-$\mathbf{v_2}$.

Figure 4 contains the *Minimum Finding* (MF) algorithm. Given the tolerance μ, MF returns FALSE if the function is concave downward, otherwise returns TRUE and the approximate location of the minimum, within the specified tolerance. MF first determines if $D(f)$ is concave upward by examining a midpoint chosen randomly from a uniform distribution over $[0,1]$, and comparing it to $D(0)$ and $D(1)$ to determine if the function can possibly be concave upward.

If the function is concave upward, MF iteratively refines the interval containing the minimum value of $D(f)$ by applying the *Mean Value Theorem* of calculus. The algorithm terminates when the interval is smaller than μ. The algorithm returns a TRUE value, and the midpoint m of the final interval is the location of the minimum of $D(f)$. The point on the boundary closest to the edge is then $\mathbf{d}(m)$.

If the point $\mathbf{d}(m)$ is closer to $\mathbf{v_1}$-$\mathbf{v_2}$ than both $\mathbf{d_1}$ and $\mathbf{d_2}$, it is used to refine the visible domain of the triangle. Since the occluder is of class OCTDIALS(T,P), the visible domain can be (uniquely) approximated by the polygonal boundary defined by the vertices of the triangle, the discontinuities, and the closest point $\mathbf{d}(m)$. The accuracy of $\mathbf{d}(m)$ is controlled by the tolerance μ, which in turn controls the continuity of the

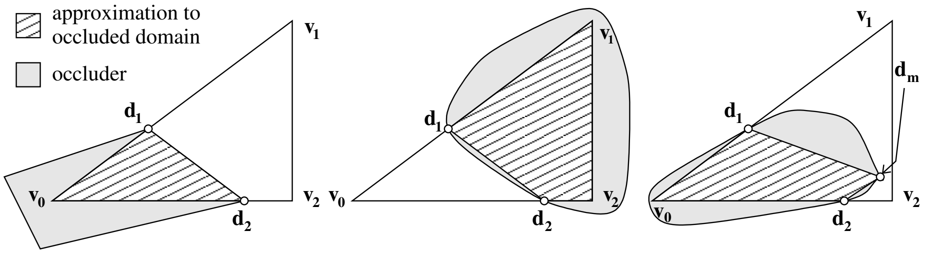
```
lo = 0; hi = 1; mid = choose randomly in [0,1]

// If boundary is concave downward, return FALSE
if( ( D(mid) > D(lo) ) and ( D(mid) > D(hi) ) ){
   return( FALSE );
} else if( D(mid) > min(D(lo),D(hi)) ){
   // lineMid is the value of D(mid) if boundary is a line
   lineMid = ( D(lo)*(hi-mid) + D(hi)*(mid-lo) ) / (hi-lo);
   if ( D(mid) > maxMid ){
      // D(mid) above line, thus boundary is concave downward
      return( FALSE );
   }
}
while( (hi-lo) > tolerance ){
   if( D(mid) > min(D(lo),D(hi)) ){
      if( D(lo) < D(hi) ){
         hi = mid;                  // Minimum is in [lo,mid]
      } else {
         lo = mid;                  // Minimum is in [mid,hi]
      }
      mid = (lo+hi)/2;
   } else { // Test midpoints of lower and upper subintervals
      midlo = (lo+mid)/2; midhi = (hi+mid)/2;
      if( D(midlo) < D(mid) ){
         hi = mid; mid = midlo;         // Minimum in [lo,mid]
      } else if( D(midhi) < D(mid) ){
         lo = mid; mid = midhi;         // Minimum in [mid,hi]
      } else {
         lo = midlo; hi = midhi;   // Minimum in [midlo,midhi]
      }
   }
}
// Desired point is mid, and distance from edge is D(mid)
return( TRUE );
```

Fig. 4. Minimum Finding Algorithm

approximation of the visible domain of the triangle. As the occluder O gets closer to an edge with zero discontinuities, so does the point $\mathbf{d}(m)$, and thus the approximation to the visible domain gradually converges to the approximation that will result when the occluder finally pierces the edge. In Figure 5, the three types of boundaries and the resulting approximation to the visible domain are illustrated for a triangle of type C_{011}.

Similarly to RSB, MF uses a random initial subdivision to avoid banding artifacts. After the initial subdivision, the interval containing the minimum is reduced by half at each iteration. Each iteration requires at most two evaluations of $D(f)$, thus after n iterations, we have evaluated $D(f)$ at most $2n$ times, and the expected error is $O(1/2^{n-1})$.

Finally, a word of caution on the MF algorithm. Since MF relies on RSB, and since both are numerical algorithms, additional care must be exerted in controlling the error levels. As a rule of thumb [8], a nested numerical method should be roughly one order of magnitude more precise than the calling method. In our implementation, we have made the RSB routine (as used within MF) ten times as accurate as the MF routine.

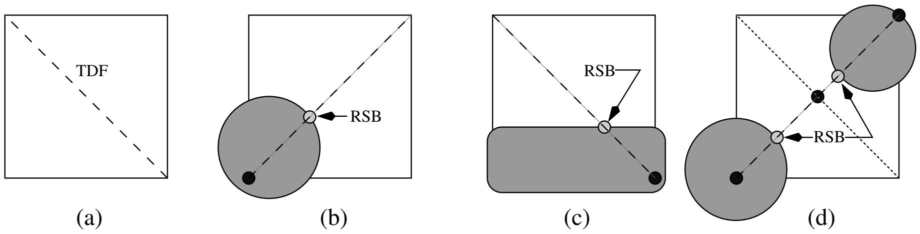


(a) Linear boundary (b) Concave downward boundary (c) Concave upward boundary

Fig. 5. Types of boundaries and resulting approximation of occluded domain.

4 Quadrilateral and General Polygon Algorithms

We now present an algorithm for finding the approximate shape of the visible domain of a (convex) quadrilateral light source Q. Since the four edges share the vertices of Q, we first compute visibility at all four vertices. We then use RSB and TDF to determine the number and approximate location of the discontinuities on the edges of Q.



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

Fig. 6. Four possible vertex configurations for a quadrilateral light source.

Given the visibility at the vertices of Q, there are four different types of configurations possible. We enumerate the configurations and the criteria used to select the diagonal along which Q is subdivided and rendered as two triangles.

1. All four vertices have the same visibility. Choose a random diagonal and determine the number of discontinuities using TDF. See Figure 6(a).
2. One vertex has a different visibility than the other three. Choose the diagonal that has the vertex with a different visibility since we can use RSB to find the discontinuity, and avoid a call to the more expensive TDF routine. See Figure 6(b).
3. Two adjacent vertices have the same visibility and the other two vertices have the opposite visibility. Choose a random diagonal and find the discontinuity using RSB. See Figure 6(c).
4. Two diagonally opposite vertices have the same visibility and the other two vertices have the opposite visibility. Find the intersection point $\mathbf{i}$ of the two diagonals and determine its visibility. Choose the diagonal $\mathbf{v_i}$-$\mathbf{v_j}$ whose end points have a different visibility than $\mathbf{i}$, and use RSB to find the discontinuity between $\mathbf{i}$ and $\mathbf{v_i}$, and to find the discontinuity between $\mathbf{i}$ and $\mathbf{v_j}$. See Figure 6(d).

Once we have chosen the diagonal, we subdivide Q using this diagonal and render the light source as two triangles, using the algorithms of Section 3.

4.1 Extending to Polygons with More than 4 Sides

Our algorithm can also be extended in a straightforward manner to polygons with more than 4 vertices. The key is to first test the vertices for visibility and to split the polygon along one of these types of diagonals, if possible:

1. Diagonals whose end points have opposite visibilities, since we can use RSB to find the discontinuity (e.g., Cases 2 and 3 of the Quadrilateral Algorithm).
2. If two intersecting diagonals are such that one diagonal has two visible vertices and the other has two occluded vertices, then the intersection will have a different visibility than the end points of one diagonal. Choose this diagonal and use RSB to find the two discontinuities (e.g., Case 4 of Quadrilateral Algorithm).

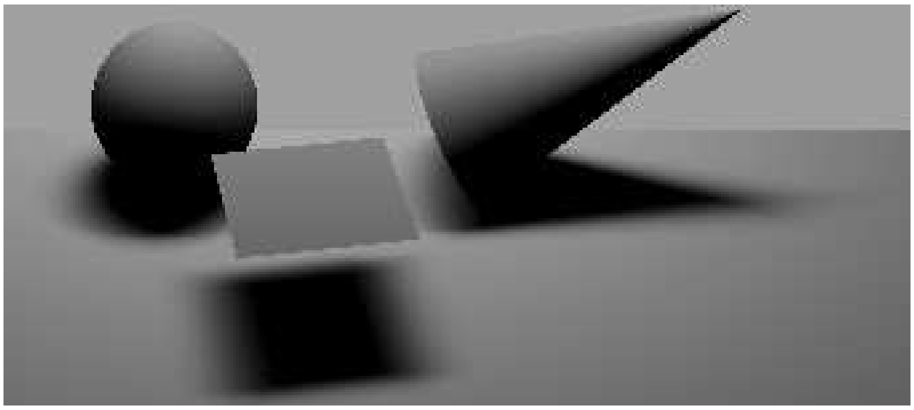
5 Results and Discussion

In Figure 7, we show the result of illuminating three different types of objects with a triangular light source. The source is located parallel to the floor, above and slightly behind the objects. In Figure7(a), the image was computed using the heuristics designed in Section 3: RSB with a tolerance of $\varepsilon = 0.05$, TDF with a tolerance of $\nu = 0.25$, and MF with a tolerance of $\mu = 0.333$. The approximate visibility of the objects required about 26 visibility tests per pixel. The shadows are smooth within each configuration region, and seemless across the boundaries between these regions. Notice that the shadows are rendered nicely for all three different types of objects. In Figure7(b), we show the result of computing the same scene with a super sampling of slightly higher cost (28 samples with visibility testing for each pixel). Slight banding is noticeable in the penumbra of all objects, especially in the shadow cast by the sphere and by the tip of the cone.

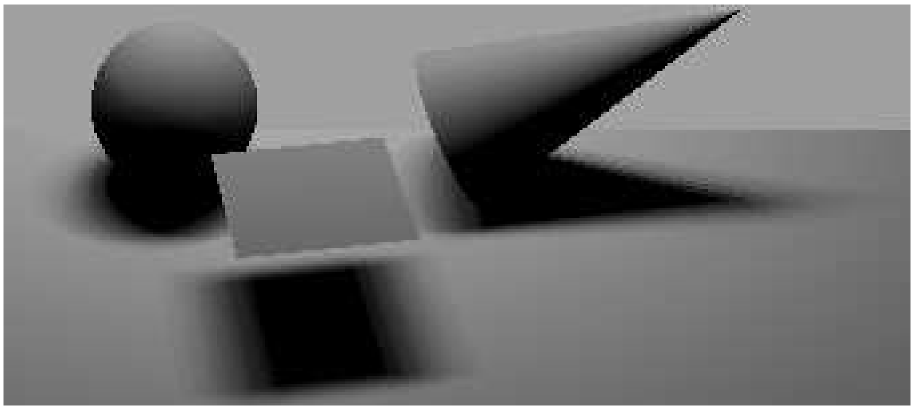
Our results point to an interesting new direction of research involving the approximation of the visible domain of area light sources. We believe that one of the keys to the successful approximation of the visible domain of an integrand lies in the consistency of the approximation. The algorithms we have developed provide an approximate solution to the visible domain that is both smooth within a configuration region, and continuous as we move from one configuration to the other.

The *Minimum Finding* (MF) algorithm developed has several important properties. Ill-conditioned transitions are alleviated by allowing a more consistent approximation of the visible domain of the integrand. Banding is eliminated without sacrificing efficiency by the introduction of a random seed. The tolerance μ provides an effective mechanism for controlling the cost and quality of the approximation. For a given tolerance μ, the complexity of MF is nearly that of a pure bisection method, namely $O((\log 1/\mu)^2)$. This is a result of MF having the RSB method (which has a complexity of $O(\log 1/\varepsilon)$ for a tolerance of ε) embedded within a method of similar complexity. The squared complexity of MF (with respect to RSB) comes as no surprise, since doubling the dimensionality of an integration problem typically squares the cost of the solution. Finally, note that contrary to other sampling methods, the difference between a small occluder being detected or not is bounded by its actual contribution to the penumbral shadow, since the resulting occlusion will be approximated to within μ.

Many improvements are still possible to these approximation algorithms. In particular, approximating the internal shape of the occluder is a difficult (and expensive) problem to solve. Interesting areas of investigation to improve the algorithms presented in this paper include: taking advantage of scene coherence to speed up the outer loop



(a) RSB ($\varepsilon = 0.05$), TDF ($\nu = 0.25$), and MF ($\mu = 0.333$), with 26.2 visibility tests



(b) Super sampling with 28 samples

Fig. 7. Penumbral region comparison for locally convex occluders

of MF, and investigating more efficient methods for finding the minimum point on a concave upward boundary, such as perhaps the *Secant Method* [8].

Finally, an important area of future research is to determine how the algorithms presented in this paper can be extended to more complex computer graphics scenes. Assuming the convex occluder problem can be solved efficiently, how can this solution be extended to an arbitrary class of occluders and larger sets of discontinuity regions? In both cases, a possible approach would be to subdivide either the source or the occluder, so as to reduce the combinatorial complexity in the case analyses.

An interesting hybrid approach could combine the best properties of the algorithms we introduced, with those of shadow mapping techniques [12, 13]. An environment could be partitioned into classes of occluders best suited to each type, with the algorithm of this paper handling large occluders, particularly if an occluder and a receiver are abutting and causing few discontinuities in the integrand, and shadow mapping techniques used to approximate more complex occluders.

6 Conclusions

In this paper, we used the *Random Seed Bisection* (RSB) and the *Two Discontinuity Finding* (TDF) algorithms to approximate the location of discontinuities for polygonal light sources, and for the class of locally convex occluders. Given a triangular light source, we used RSB and TDF to determine the discontinuities along each edge, and to classify the triangle into one of six possible configurations. We introduced the *Minimum Finding* (MF) algorithm to approximate the shape of the visible domain within the triangle. Finally, we approximated the visible domain with a polygonal approximation using the discontinuities found. The resulting integrals can be solved efficiently using either a low degree numerical cubature, or in the case of diffuse surface and light sources, using an analytic solution. We then proposed an extension of these algorithms to general polygons. We believe that this an important first step in addressing the issue of the approximate knowledge of visible domains and its application toward efficient rendering of penumbral shadows.

7 Acknowledgments

The comments by the reviewers helped to improve the paper, and were much appreciated. We gratefully acknowledge the funding of NSERC and CITO in our research.

References

1. Brotman, L.S., Badler, N.I., "Generating Soft Shadows with a Depth Buffer Algorithm", *IEEE Computer Graphics and Applications*, 4(10), Oct. 1984.
2. Campbell, A.T., "Modeling Global Diffuse Illumination for Image Synthesis", Ph.D. Thesis, University of Texas at Austin, Dec. 1991.
3. Chin, N., Feiner, S., "Fast Object-Precision Shadow Generation for Area Light Sources Using BSP Trees", *ACM Computer Graphics (SIGGRAPH Symp. on Inter. 3D Graphics 1992)*.
4. Cook, R.L., Porter, T. and Carpenter, L., "Distributed Ray Tracing", *Computer Graphics*, 18(3), July 1984.
5. Drettakis, G., Fiume, E., "A Fast Shadow Algorithm for Area Light Sources Using Backprojection", *ACM SIGGRAPH Annual Conference Series*, July 1994.
6. Heckbert, P.S., Herf, M., "Simulating Soft Shadows with Graphics Hardware", *Technical Report CMU-CS-97-104*, Carnegie Mellon U., June 1997.
7. Hedley, D., Worrall, A., Paddon, D., "Selective Culling of Discontinuity Lines", *8th Eurographics Workshop on Rendering*, June 1997.
8. Kahaner, D., Moler, C., Nash, S., *Numerical Methods and Software*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.
9. Lischinski, D., Tampieri, F., Greenberg, D., "Discontinuity Meshing for Accurate Radiosity", *IEEE C.G. & Appl.*, 12(6), Nov. 1992.
10. Moon, P., *The Scientific Basis of Illuminating Engineering*, McGraw-Hill, New York, 1936.
11. Ouellette, M.J., Fiume, E., "Approximating the Location of Integrand Discontinuities for Penumbral Illumination with Linear Light Sources", *Graphics Interface 99*, 1999.
12. Soler, C., Sillion, F.X., "Automatic Calculation of Soft Shadow Textures for Fast, High Quality Radiosity", *9th Eurographics Workshop on Rendering*, June 1998.
13. Soler, C., Sillion, F.X., "Fast Calculation of Soft Shadow Textures Using Convolution", *ACM SIGGRAPH Annual Conference Series*, July 1998.
14. Stewart, J.A., Ghali, S., "Fast Computation of Shadow Boundaries Using Spatial Coherence and Backprojections", *ACM SIGGRAPH Annual Conference Series*, July 1994.
15. Wallace, J.R., Elmquist, K.A., and Haines, E.A., "A Ray Tracing Algorithm for Progressive Radiosity", *Computer Graphics*, 23(3), July 1989.