A Complete Treatment of D1 Discontinuities in a Discontinuity Mesh

Sherif Ghali A. James Stewart

Department of Computer Science
University of Toronto
{ghali,jstewart}@dgp.toronto.edu

Abstract

This paper presents a treatment of first–order discontinuities (D1) that arise in discontinuity meshes. An algorithm is described that, given a planar D1 discontinuity surface in a polyhedral scene, computes the corresponding discontinuity curves on the faces of the scene. A method is described to efficiently update the backprojection across a D1 discontinuity curve. An alternative update method is presented which is novel and numerically robust.

Keywords: D1 discontinuity, discontinuity meshing, shadows, radiosity, global illumination.

1 Introduction

For global illumination algorithms operating on polyhedral scenes illuminated by an area light source, the discontinuity mesh is used to model accurately sharp changes in illumination [Hec91, Hec92a, LTG92, LTG93, Tam93].

The *backprojection* is a topological description of the boundary of the visible light source, as seen from some viewpoint. The *discontinuity mesh* is a set of maximal cells on the faces of the scene such that the backprojection is constant for all viewpoints within each cell. Adjacent cells on a face are divided by *discontinuity curves*, which are the intersection of the face with *discontinuity surfaces* [PD86]. Discontinuity curves are line segments or conic sections, whereas discontinuity surfaces are subsets of planes or quadric surfaces.

The backprojection in one cell can be efficiently updated to compute that in adjacent cells and hence can be propagated to all other cells of the mesh [GM90, GCS91, SG93, SG94, DF94, Dre94]. Given the backprojection within a cell of the discontinuity mesh, the exact primary irradiance of any point in the cell can be computed with a simple contour integral around the boundary of the visible source for a source with constant emission [Moo36, NN85, BRW89].

First–order discontinuities (termed “D1”) correspond to discontinuities in the first derivative of the illumination [Hec91]. There are two ways in which a D1 discontinuity surface can arise: (a) two edges of the scene (one of which might be on the light source) are coplanar and the planar surface embedding the edges intersects the light source, and (b) the plane embedding a face of the scene intersects the light source. An example of the former is shown in Figure 1.

D1 discontinuities are important for two principal reasons: First, the D1 discontinuities can be perceptually important, which most often occurs when a D1 discontinuity is defined by an edge in the scene that is parallel to an edge on the light source. Second, if present in a scene, they must be incorporated into the discontinuity mesh: If a discontinuity line is missing from the mesh, the two cells adjacent to the missing line will appear in the mesh as one combined cell. That combined cell will no longer have a single backprojection, preventing efficient computation of exact primary irradiance and efficient propagation of the backprojection.

Two operations are necessary to treat D1 discontinuities. First, a D1 surface must be *cast*; this operation determines the curves that are the intersection of the D1 surface with the faces of the scene. This is not a simple surface/face intersection problem because only those curves that are visible from the light source are counted. Second, the backprojection must be *updated* across a D1 surface in order to propagate it from cell to adjacent cell of the discontinuity mesh. Both operations are described in this paper, as well as an alternative update method that is novel and robust.

D2 surfaces have been treated elsewhere ([GM90, SG93, DF94], among others). Although D2 surfaces can be of higher algebraic degree than D1 surfaces, their treatment is simpler. (This is partly because a viewpoint on a D2 surface can only look toward the light source in one

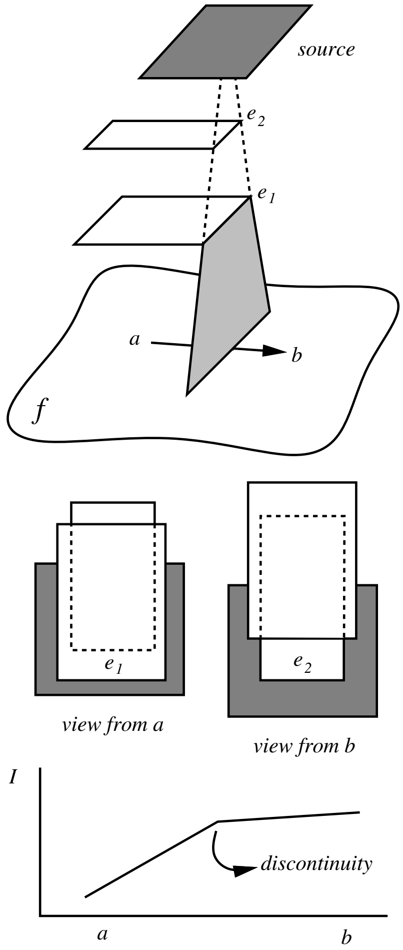

Figure 1: A D1 discontinuity surface (lightly shaded) and its intersection with a face f of the scene. As a viewpoint crosses the surface from a to b, the area of the visible light source is discontinuous in its first derivative since the boundary of the source (darkly shaded), which was delimited by edge e_1, becomes delimited by e_2, which is more distant.

direction, whereas a viewpoint on a D1 surface can look in many directions.) D2 surfaces will not be discussed further in this paper.

2 Discontinuity Meshing in Flatland

The problem of casting a D1 surface is, in fact, a special case of discontinuity meshing, which occurs in the plane. This section describes general two–dimensional meshing. Section 3 will show how it applies to D1 casting.

Heckbert was the first to use discontinuity meshes in

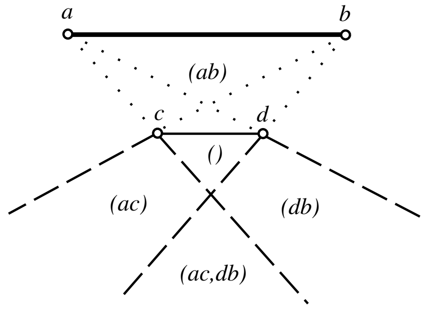

Figure 2: A source ab and a segment cd. The scene below the source is partitioned by the dashed discontinuity lines into five regions; different parts of the source are visible in each region. For example, in the region labelled (ac, db), two pieces of the source are visible, one delimited by the endpoints a and c, the other delimited by d and b.

computer graphics. He showed [Hec92b] how to use a visibility algorithm to compute the discontinuity mesh. Discontinuity meshes are akin to aspect graphs [PD86], which are used in computer vision.

We extend Heckbert's work with (a) a new algorithm — based on the plane sweep paradigm — to compute the discontinuity mesh and (b) an efficient computation of the backprojection in each cell of the mesh.

In the plane, a *scene* consists of a set of simple, non-intersecting polygons and a distinguished line segment called the *light source*. Each polygon consists of a cycle of *segments*. We assume without loss of generality that the source is parallel to the x axis and emits light in the $-y$ direction, which is termed *downward*. Parts of polygons above the source are ignored since they receive no direct illumination.

From any viewpoint in the scene, the source may be partially or fully obscured by other segments. In general, the source appears as a (possibly empty) sequence of disjoint pieces, ordered radially around the viewpoint. This sequence of disjoint pieces is called the *backprojection* of the viewpoint. Each piece of the backprojection is delimited by two endpoints; these can be endpoints of the source or endpoints of other segments in the scene. See Figure 2.

In the plane, the *discontinuity mesh* is a set of maximal cells on the segments of the scene such that the backprojection is constant within each cell. Adjacent cells on a segment are separated by *discontinuity points*, which are the intersection of the segment with *discontinuity lines*. A discontinuity line passes through two segment endpoints (call them d and e) and intersects the source *without* intersecting any segment between d, e, and the source. See Figure 3.

2.1 Computing the Mesh

Following is the algorithm to compute the discontinuity mesh.

1. (Refer to Figure 4.) Position a point p at one endpoint of the source segment. From p draw rays to each segment endpoint in the scene. Sort these rays radially around p. For each region between pairs of adjacent rays, determine the segments that intersect the region and sort them by increasing distance from p. The list of segments in each region is called the *segment stack*. In each region, the topmost segment on the stack is visible from p.

2. Consider each pair of adjacent regions: If the topmost segments in the two corresponding segment stacks are different, deposit a discontinuity point on the more distant of the two segments. In Figure 4, a discontinuity point would be deposited on segment gh at its intersection with the second ray from the left.

3. Move p toward the other end of the source, maintaining the rays from p. Whenever two adjacent rays become collinear, p lies on a discontinuity line, the treatment of which is described below. After treating the discontinuity, exchange the two rays in the radial ordering around p.

4. When p reaches the other end of the source, repeat Step 2.

This algorithm can be implemented efficiently using a priority queue which contains the positions on p at which rays become collinear. For n rays, the queue will contain $n - 1$ entries, since only adjacent rays can become collinear. Step 3 above consists of removing from the queue the next closest position to p, moving p to that position, treating the discontinuity at p, and adding new positions to the queue after the two rays are exchanged in the radial order.

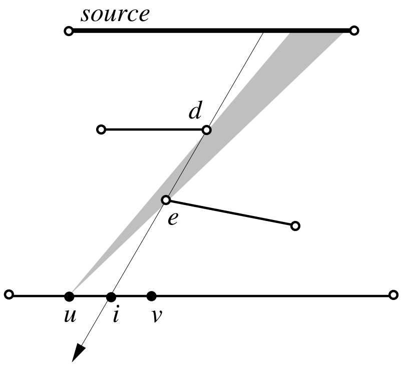

Figure 3: A two–dimensional scene with a source and three segments. A viewpoint at u can see a part of the source. This part of the source disappears from view when the viewpoint moves to v. The line through d and e is a discontinuity line and i is a discontinuity point.

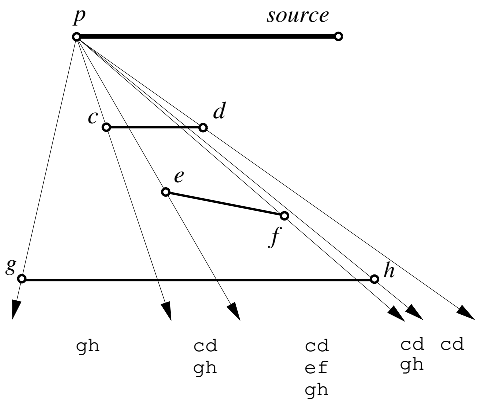

Figure 4: The algorithm initially draws a ray from a point p (positioned at one end of the source) to every segment endpoint in the scene. For each region between adjacent rays, the segments that intersect that region are recorded in order of increasing distance from p (these are shown at the bottom).

Treatment of Discontinuities

A discontinuity is detected in Step 3 above when two adjacent rays become collinear and the region between them "collapses." As p moves past the discontinuity, the rays are exchanged in the radial order and a new region appears between them. Two things must be done here: (a) the segment stack of the new region must be computed, and (b) a discontinuity point might have to be deposited on some segment.

The new segment stack is easily obtained by modifying the segment stack from the old, collapsed region. Let the discontinuity line be defined by two endpoints u and v, where p is closer to u than it is to v. Looking along the discontinuity line toward p, the segments adjacent to u and v (of which there are three or four, since the segments belong to polygons) will appear to the left of, to the right of, or collinear with the discontinuity line (see Figure 5). Modify the segment stack of the old, collapsed region by (a) removing any left edges adjacent to u and any right edges adjacent to v and (b) adding any right edges adjacent to u and any left edges adjacent to v.

Next, we determine whether to deposit a discontinuity point. Consider Figure 5. The rays pu and pv are adjacent

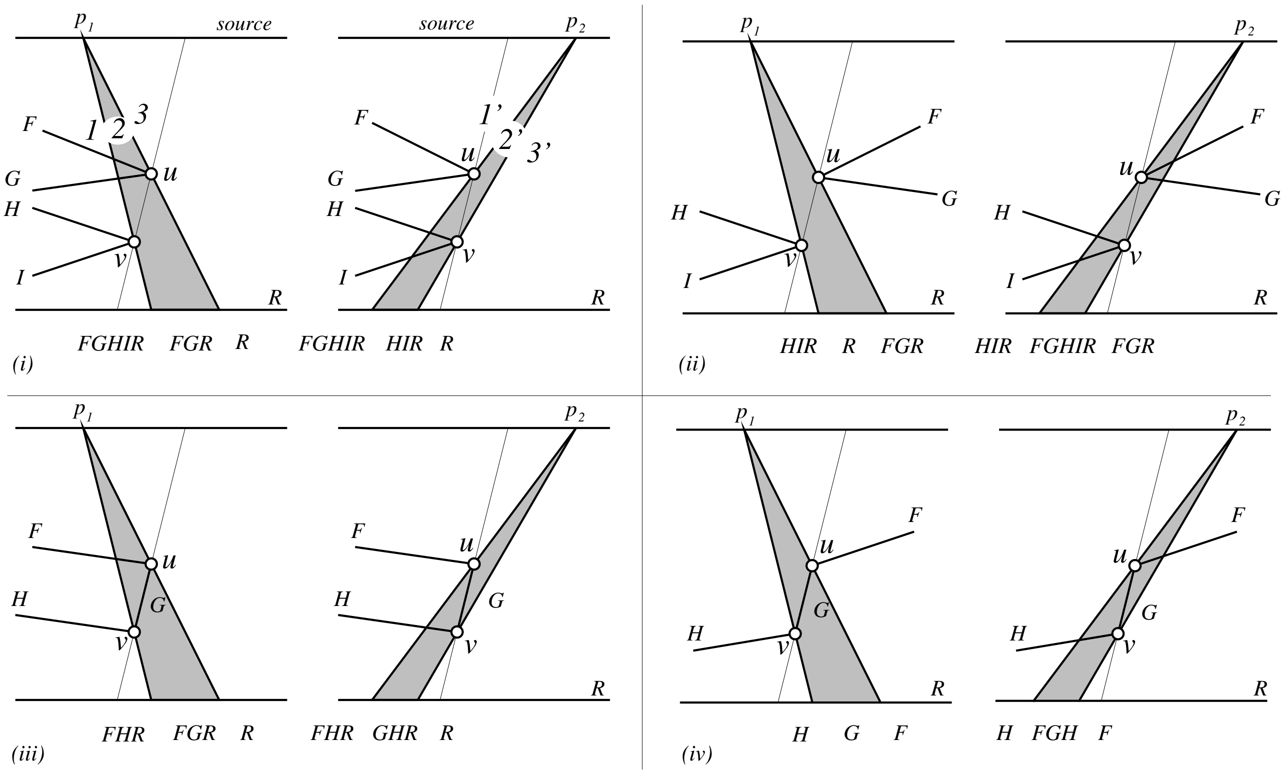

Figure 5: As point p moves from p_1 to p_2 along the light source, the two rays pu and pv exchange position in the radial order around p. The segment stacks in each region defined by these rays (regions 1,2,3 at p_1 and regions $1'$,$2'$,$3'$ at p_2 above) are updated to reflect the crossing. Below each region, these stacks are shown before and after p crosses the discontinuity. In cases (i), (ii), and (iii), a discontinuity point is deposited on the bottom segment R.

to three regions; let s_1, s_2, and s_3 be the topmost segment of each region (in counterclockwise radial order around p) *before* p has passed the discontinuity (e.g. at p_1 in Figure 5). Let $s_{2'}$ be the topmost segment of the middle region *after* p has passed the discontinuity (e.g. at p_2 in Figure 5). For example, in Figure 5(i), $s_1 = F,\ s_2 = F,\ s_3 = R$, and $s_{2'} = H$.

Given s_1, s_2, $s_{2'}$, and s_3, the applicable case in Figure 5 is easily determined:

- *Case i:* u and v have edges to the same side of the uv line and either s_1 or s_3 is different from both s_2 and $s_{2'}$. A discontinuity point is deposited on the one of s_1 and s_3 that is different from both s_2 and $s_{2'}$. In Figure 5(i), segment R is different from the segments F and H.

- *Case ii:* u and v have edges to opposite sides of the uv line and either s_2 or $s_{2'}$ is different from both s_1 and s_3. A discontinuity point is deposited on the one of s_2 and $s_{2'}$ that is different from both s_1 and s_3. In Figure 5(ii), segment R is different from the segments F and H.

- *Case iii:* u and v are joined by a segment and the other two segments adjacent to u and v are to the same side of the uv line. Once again, either s_1 or s_3 is different from both s_2 and $s_{2'}$. A discontinuity point is deposited on the one of s_1 and s_3 that is different from both s_2 and $s_{2'}$. This is segment R in Figure 5(iii).

- *Case iv:* u and v are joined by a segment and the other two segments adjacent to u and v are on opposite sides of the uv line. No discontinuity point is deposited.

Some other situations exist, which are not shown in Figure 5: *(Case v)* If either u or v has two adjacent segments on opposite sides of the uv discontinuity line, no discontinuity point is deposited, since a viewpoint crossing the uv line below these segments cannot see the source at all when looking upward along the uv line; *(Case vi)* If another segment of the scene intersects the uv line above u, no discontinuity point is deposited (in this case, s_2 and $s_{2'}$ are identical); *(Case vii)* If another segment of the scene intersects the uv line between u and v, no discontinuity point is deposited (in this case, each of s_2 and $s_{2'}$ is identical to one of s_1 and s_3).

The segment stacks must be maintained during p's progress along the source in order to determine quickly, whenever p crosses a uv line, whether a discontinuity point needs to be deposited. We could do this without using segment stacks, but then we would have to determine, whenever p crosses a uv line, what parts of the scene are visible from p. This would take time proportional to the number of segments in the scene *every time* p crosses a uv line. The extra effort involved in maintaining the segment stacks eliminates this potentially expensive step.

The complete mesh algorithm of this section was implemented by the authors in two days with about 1000 lines of C code. For a scene of 100 segments the program took 2 seconds on a 166 MHz Pentium PC. We expect this to become much faster when the code is improved with more sophisticated data structures for the priority queue and the segment stacks.

2.2 Computing the Backprojections

Each cell of the mesh has a backprojection which describes the topology of the visible source, as seen from that cell. We will compute backprojections in a few key cells of the mesh and will propagate them from cell to adjacent cell throughout the mesh. The propagation will involve updating the backprojection incrementally as the viewpoint crosses discontinuity lines that separate adjacent cells.

Updating the Backprojection Across a Discontinuity

Each discontinuity line is defined by two segment endpoints u and v and is denoted $\langle uv \rangle$. There are several ways that the backprojection can be updated as the viewpoint crosses $\langle uv \rangle$:

- If neither u nor v is on the source:
  - If u and v bound segments to the same side of $\langle uv \rangle$, then one of u and v replaces the other in the backprojection (e.g. $\langle df \rangle$ in Figure 6).
  - If u and v bound segments on opposite sides of $\langle uv \rangle$, then segment uv appears or disappears in the backprojection (e.g. $\langle de \rangle$ in Figure 6).
- If either u or v is on the source:
  - If u and v bound segments to the same side of $\langle uv \rangle$, then segment uv appears or disappears in the backprojection (e.g. $\langle bf \rangle$ and $\langle ac \rangle$ in Figure 6).
  - If u and v bound segments on opposite sides of $\langle uv \rangle$ then one of u and v replaces the other in the backprojection (e.g. $\langle be \rangle$ in Figure 6).

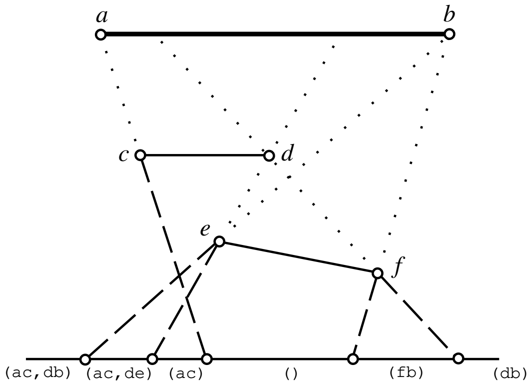

Figure 6: A scene with a source ab and three segments. The backprojection in each cell of the bottom segment is shown below the cell.

Computing Backprojections in Cells

This section describes how to compute the backprojection in each cell of the discontinuity mesh, which is computed with the algorithm of Section 2.1.

From some segments of the scene, the whole light source is visible from every point of the segment. These segments are said to be *fully visible*. For example, in Figure 6, segment cd in is fully visible but segment ef is not.

The algorithm of Section 2.1 can be modified to determine these segments, as follows. That algorithm initially computes the segments that are visible from one end of the source. Any segment that is *not* initially visible is flagged as "not fully visible." While the algorithm proceeds, discontinuity points are deposited on segments. Any segment on which a discontinuity point is deposited is also flagged as "not fully visible." (Since a discontinuity point separates two cells in which the topology of the visible source is different, in at least one of those cells, the whole source can't be visible.) Once the algorithm terminates, any segment that has not been flagged as "not fully visible" *is* fully visible. Note that at least one segment of the scene will be fully visible, since it is not possible to establish a cycle of overlapping segments in two dimensions.

On each fully visible segment is a single cell of the discontinuity mesh. The backprojection of this cell consists of the whole light source. Thus, we can determine very quickly the backprojection on at least one segment of the scene.

In what follows, we describe how to propagate the backprojection from a cell of one segment into a cell of another segment. Once on the second segment, the backprojection can be propagated to the other cells of the second segment by incrementally updating it across the discontinuity lines that separate the cells (as described above). By repeating the segment–to–segment and cell–to–cell propagation, the backprojection can be efficiently computed for every cell in the mesh.

Consider a discontinuity line, defined by endpoints u and v, that intersects the interior of a segment at a point w. Without loss of generality, assume that u is closer to the segment than v. Let C_u be the cell adjacent to u and let C_w be the cell adjacent to w that is on the side of $\langle uv\rangle$ opposite to C_u. Refer to Figure 7.

The backprojection in C_u is propagated to C_w by updating it across the discontinuity lines that pass through u. The update is performed in radial order around u and until $\langle uv\rangle$ is reached.

For example, in Figure 7, the discontinuity lines are ordered clockwise around u: $\langle ua\rangle$, $\langle ub\rangle$, $\langle uv\rangle$, $\langle uc\rangle$. Note that no change occurs in the backprojection when the viewpoint crosses a discontinuity line above u; only by crossing below u will u have an effect on the backprojection.

In summary, we have described how to compute efficiently the two–dimensional discontinuity mesh and how to compute efficiently the backprojection of each cell of the mesh.

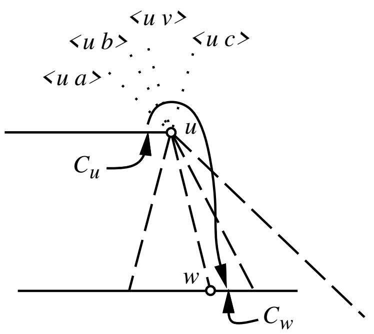

Figure 7: The backprojection in cell C_u is propagated to cell C_w by updating it across the discontinuity line $\langle ua\rangle$, followed by $\langle ub\rangle$. The light source, not shown, is above everything.

3 Casting a Discontinuity Surface

In a three–dimensional scene, a D1 discontinuity surface occurs in two situations: either (a) two segments are coplanar and the plane embedding them intersects the light source or (b) the plane embedding a face of the scene intersects the light source. Identifying D1 surfaces has been studied elsewhere [GM90, Hec92a, LTG92, DF94] and will not be treated here.

A D1 surface is a bounded, infinite region of a plane. If two edges define the surface, the surface consists of those points x from which a ray can be drawn that intersects one edge, then the other edge, then the light source (in that order) without intersecting the interior of any object between the first edge and the source (recall Figure 1). Similarly, if a face defines the surface, the ray must intersect the face and the source (in that order) without intersecting the interior of any object between the face and the source.

3.1 Using Two–Dimensional Meshing to Cast a D1 Surface

Given a D1 discontinuity surface in a three–dimensional scene, the casting operation determines the discontinuity segments that (a) are intersections of the D1 surface with scene faces and (b) are visible from the source.

It is easy (although not very efficient) to intersect the D1 surface with each face of the scene to form a list of potential discontinuity segments. Drettakis and Fiume [DF94] use a heuristic to speed this up: The ambient space is divided into voxels, each of which records the scene faces that intersect it. The D1 surface is first intersected with the voxels to determine a set of candidate faces, each of which is intersected to determine the potential discontinuity segments.

The casting procedure follows: The potential discontinuity segments, along with a segment corresponding to the light source, are input to the two–dimensional discontinuity meshing algorithm of Section 2. That algorithm outputs the description of a two–dimensional discontinuity mesh consisting of discontinuity cells and their backprojections. Only those discontinuity cells from which some part of the source is visible correspond discontinuity edges in the three–dimensional mesh. The casting operation simply outputs the three–dimensional segments corresponding to these cells! Note that the discontinuity points that bound these cells in the two–dimensional mesh correspond exactly to discontinuity points in the three–dimensional mesh. Figure 8 shows an example.

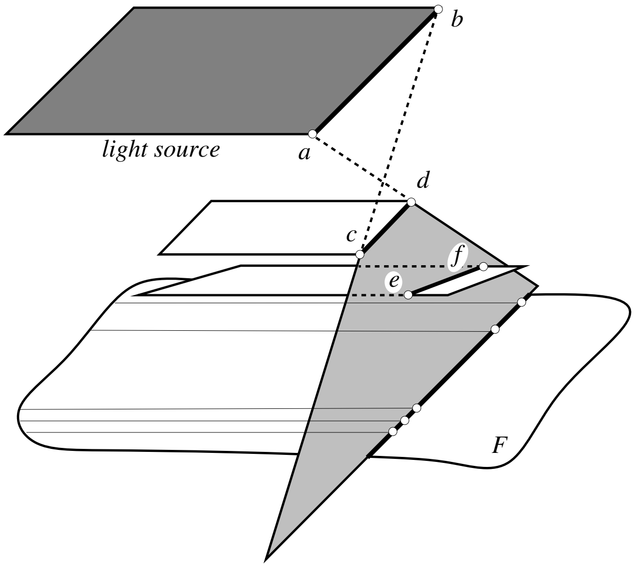

Figure 8: The three-dimensional scene that gave rise to the two–dimensional scene of Figure 6. The lightly shaded D1 surface is defined by edges ab and cd. The cells of the two–dimensional discontinuity mesh that have non–empty backprojections (five of them in Figure 6) correspond to D1 discontinuity lines in the three–dimensional scene. Note that the discontinuity points in the two–dimensional mesh of Figure 6 correspond exactly to discontinuity points in the three–dimensional mesh above. Some other mesh edges touching these points are shown as thin lines.

4 Updating the Backprojection Across a General D1 Surface

This section describes methods to update the backprojection across a D1 discontinuity surface. The particular method used depends upon how the discontinuity surface is defined.

4.1 Surfaces Defined by an Arbitrary Face

The following algorithm is a based upon the work of Gigus and Malik [GM90] and takes advantage of the fact that we only update the boundary of the visible light source, while their application must update the more complex visible surface map.

Consider the case in which the D1 surface is defined by a face F. For a viewpoint on the surface, the *lower chain* of F is the sequence of (parts of) edges visible from the viewpoint. The *upper chain* is the sequence of (parts of) edges visible looking from a large circle — which is centred at the viewpoint — toward the viewpoint but not through the viewpoint. As Gigus and Malik [GM90] point out, for a face of n segments, both can be computed in $\mathcal{O}(n)$ time with a two–dimensional visible surface algorithm that exploits the fact that F is a simple polygon.

The D1 surface partitions the ambient space into two halfspaces: an *outside* halfspace, containing the outward–pointing normal of the face, and an *inside* halfspace.

As the viewpoint traverses the D1 surface, edges of F that are present in the backprojection will disappear, to be replaced by other edges of F. If the viewpoint traverses from outside to inside, the edges that disappear will be from the upper chain, while those that replace them will be from the lower chain. Conversely, if the viewpoint traverses from inside to outside, lower chain edges will disappear, to be replaced by upper chain edges.

The following procedure updates the backprojection as the viewpoint traverses the D1 surface: For each edge e of face F present in the backprojection, locate the edges in the opposite chain to that of e which have the same central projection onto the viewpoint (the endpoints of the chain can be found in $\mathcal{O}(\log n)$ time with binary search). Replace e with those found in the opposite chain. Alternatively, we can merge the upper and lower chains in radial order around the viewpoint, replacing an edge in one by an edge in the other whenever it occurs in the backprojection.

4.2 A Symbolic Approach for Surfaces Defined by Two Separate Edges

This section describes a novel, alternative, method to update the backprojection across a D1 surface that is defined by two edges of the scene. In this approach, we break each D1 surface into a number of simpler D2 surfaces. We then determine the order in which these new surfaces must be traversed to update correctly the backprojection. This approach is more robust than that of the previous section because almost all of the computation is done symbolically.

We break the discontinuity into a number of EV surfaces by jittering one endpoint of one of the edges infinitesimally in a direction out of the plane. This results in four D2 surfaces which are infinitesimally different from the original D1 surface. See Figure 9. The key point is the following: An update of the backprojection across the D1 surface is equivalent to a sequence of updates across the four D2 surfaces, in some order, called the *crossing order*. However, not every crossing order of the four D2 surfaces admits successful updates; a crossing order is *realizable* if the backprojection can be successfully updated across the D2 surface in that order, resulting in the same change to the backprojection as an update across the original D1 surface.

In Figure 9(i), vertex a is jittered out of the plane defined by $abcd$. The discontinuities that result on face f are the four lines in Figure 9(ii) which are labelled by the corresponding discontinuity surfaces.

Any sequence of updates across the four discontinuity surfaces that follows a path from point p to point q in Figure 9(ii) is sufficient. There are twelve realizable and twelve unrealizable such sequences. To find a realizable sequence, it is sufficient to test the dot product $(b - a) \cdot (d - c)$. If it is positive, then the sequence $[acd, bcd, cab, dab]$ is realizable. If it is negative, then that sequence would correspond to an unrealizable update from point u to point v in the figure, but a realizable update would be the sequence $[bcd, acd, cab, dab]$ (or, in general, exchange either pair of surfaces (acd, bcd) or (cab, dab) in the original sequence).

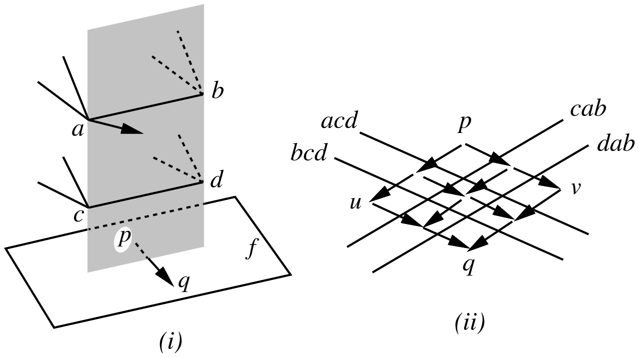

Figure 9: (i) A D1 surface (shaded) defined by edges ab and cd. Moving vertex a infinitesimally out of the surface results in four D2 surfaces: acd, bcd, cab, and dab. (ii) These surfaces intersect the face f in an arrangement of four lines.

Intuitively, the unrealizable crossing orders are unrealizable because the regions of u and v don't correspond to either side of the original D1 surface, unlike the regions of p and q. Note that the twenty-four crossing orders described above are the only possible crossing orders, since this is the number of permutations of four items.

Performing the infinitesimal perturbation is not necessary in practice; it is sufficient to know that some perturbation exists for the chosen crossing sequence. The previous discussion is summarised in the following lemma.

Lemma 1 *Given two coplanar edges not belonging to the same face and defining a D1 discontinuity surface, the backprojection can be updated across the D1 surface. This update can be done in constant time for each such pair of edges.*

4.3 A Symbolic Approach for Surfaces Defined by Triangular Faces

The simplest face to embed a D1 surface is a triangular face (see Figure 10). The description of the update in this case will simplify the presentation of the convex face case which follows.

Updating the backprojection across a triangular face is simple. In Figure 10, imagine a viewpoint crossing the discontinuity line from p to q. At p, edge z appears in the backprojection and at q, edges x and y appear. If the viewpoint is moving from p to q, we simply replace the sequence of edges $[z]$ in the backprojection by the sequence of edges $[x, y]$ (or a portion thereof). To determine the portion of $[x, y]$ that replaces $[z]$, we project the triangle vertices onto the plane of the light source (with a central projection through the point at which the viewpoint crosses the D1 surface). Only those edges of $[x, y]$ whose projections falls inside the visible part of the source (defined by the backprojection) are inserted. This ensures that an occluded edge is not added to the backprojection. This algorithm is just a special case of that of Gigus and Malik [GM90].

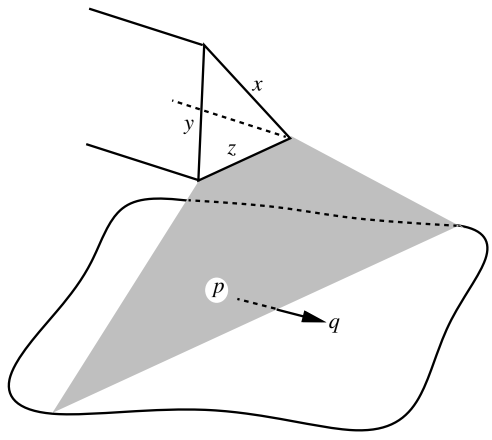

Figure 10: Crossing a D1 discontinuity defined by a triangular face. From p to q, edge z in the backprojection is replaced by edges x and y. From q to p, x and y are replaced by z.

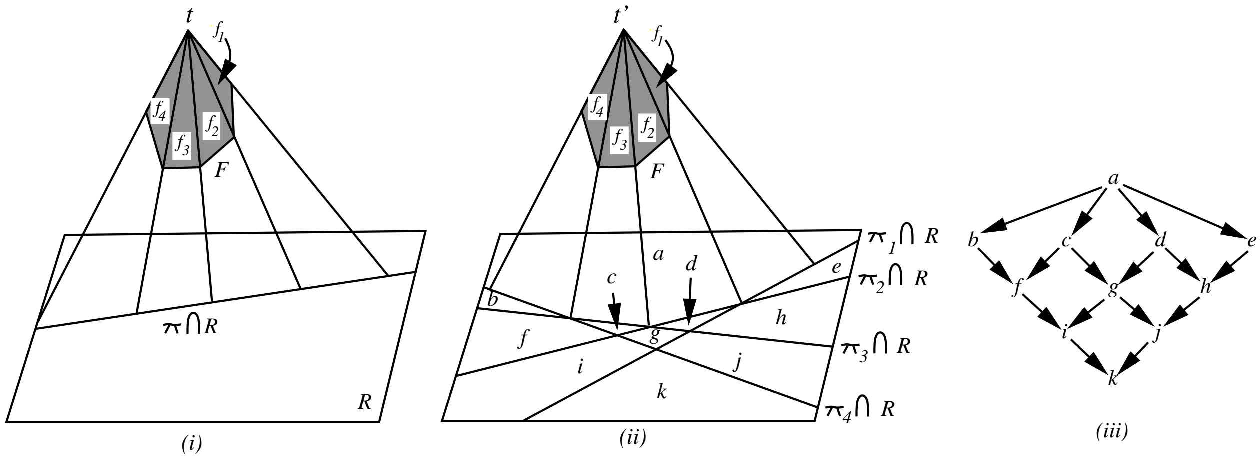

Figure 11: (i) D1 discontinuity surface defined by a convex face. The four discontinuity surfaces intersect the face R in the same line. (ii) EV discontinuity surfaces $\pi_1, \pi_2, \ldots \pi_{k-2}$ induced by perturbing the topmost vertex of a face F of k vertices. (iii) There exists at least one feasible path from a to k.

4.4 A Symbolic Approach for Surfaces Defined by Convex Faces

A more difficult problem arises with convex faces of more than three edges. It is possible to dissect the D1 surface that embeds the face into a set of EV surfaces (one for each vertex/edge pair), but the size of such a set would be in $\Theta(k^2)$ for a face of k edges. The naive approach of computing all $\Theta(k^2)$ surfaces fails, however, since an arbitrary crossing order of the surfaces will not work.

In what follows, we describe a jittering approach to break a convex face of k edges defining a D1 surface into $k-2$ triangular faces, each defining a single EV surface. This decomposition admits many crossing sequences, any one of which can be chosen to update the backprojection across the set of $k-2$ EV surfaces.

In Figure 11, assume that plane π embedding the face F intersects the light source. Thus π embeds a D1 discontinuity surface. Consider a viewpoint on face R crossing the discontinuity line $\pi \cap R$. Behind π the visible portion of the light source contains some lower chain of edges of F. In front of π the visible portion contains some upper chain from the face. Note that the upper and lower chains are always disjoint.

A backprojection on the face R is updated across π as follows. Let t be the vertex on F most distant from the plane of R. Displace t infinitesimally in the direction of the outward–pointing normal of F and call the new vertex t'. Create $k-2$ triangular *subfaces* by connecting t' to adjacent pairs of the other $k-1$ vertices of F. Number the subfaces from f_1 to f_{k-2} in clockwise radial order around t'. See Figure 11(ii).

The discontinuity curves that now arise on R are formed from the intersection of R with the planes embedding the subfaces. These planes are called *subdiscontinuity surfaces* and are labelled $\pi_1 \ldots \pi_{k-2}$. It is clear from the construction (and from Figure 11) that the subdiscontinuity surfaces will be tangent to a convex curve.

We now describe how to choose a realizable crossing sequence of $\pi_1 \ldots \pi_{k-2}$.

Consider a viewpoint moving in a direction with a positive dot product with F's normal. (If the viewpoint is moving in the direction opposite to the normal, we perturb t in that direction.) Any one of the subdiscontinuity surfaces $\pi_1 \ldots \pi_k$ may be crossed first, since all surfaces abut the region containing the viewpoint (region a in Figure 11). Choose the subdiscontinuity π_i to be crossed first such that at least one edge of F defining π_i is present in the backprojection and thus the initial update is feasible.

After crossing π_i, the viewpoint is forced to cross the subdiscontinuities π_{i+1} to π_{k-2} in increasing order and is forced to cross π_{i-1} to π_1 in decreasing order. Each interleaving of the two sequences $<\pi_{i-1}, \pi_{i-2}, \ldots \pi_1>$ and $<\pi_{i+1}, \pi_{i+2}, \ldots \pi_{k-2}>$ corresponds to some path through the arrangement of discontinuity lines on R. Thus every interleaving is realizable and it is sufficient to choose one. The previous discussion is summarised in the following lemma.

Lemma 2 *Given a convex face of k edges which defines a D1 discontinuity surface, the backprojection can be updated across the surface in $\mathcal{O}(k)$ time.*

In Figure 11(iii), this realizable update corresponds to some path from a to k in the directed graph shown. As noted above, the infinitesimal perturbation is not used in practice; it is sufficient to know that some path in some perturbation exists for the chosen crossing sequence. This update is robust because few numeric computations are used: one to find the highest vertex and one to find the direction of traversal of the D1 surface. Most EV updates can be performed symbolically [SG94].

5 Conclusions

The D1 surfaces considered here form an important part of the complete discontinuity mesh. The existence of such surfaces was previously shown [PD86, GM90, Hec91, Hec92a, LTG92]. A complete treatment of these surfaces has been presented, along with a novel backprojection update based upon a jittering argument.

This paper shows that the casting of D1 surfaces is not as simple as it first seems. This presentation unifies several methods involved in computing discontinuities: The casting step of D2 surfaces considers visibility in one dimension, that of D1 surfaces considers visibility in two dimensions, and the general problem of handling discontinuity surfaces is concerned with visibility in three dimensions.

Two methods are presented to update the backprojection from cell to cell in a discontinuity mesh when the segment crossed is a D1 discontinuity edge. The first, rather "brute force," method is based on [GM90]. The second method uses a jittering approach to break a D1 surface into a set of D2 surfaces. This jittering does not need to be performed in practice and in fact allows us to update the backprojection by making use primarily of symbolic information. It is not clear that the jittering approach can be extended to nonconvex faces.

References

[BRW89] D. R. Baum, H. E. Rushmeier, and J. 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.

[DF94] G. Drettakis and E. Fiume. A fast shadow algorithm for area light sources using backprojection. *Computer Graphics Proceedings, Annual Conference Series 1994*, 28:223–230, August 1994.

[Dre94] G. Drettakis. *Structured Sampling and Reconstruction of Illumination for Image Synthesis*. PhD thesis, University of Toronto, January 1994.

[GCS91] Z. Gigus, J. Canny, and R. Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):542–551, June 1991.

[GM90] Z. Gigus and J. Malik. Computing the aspect graphs for line drawings of polyhedral objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):113–122, February 1990.

[Hec91] P. Heckbert. *Simulating Global Illumination Using Adaptive Meshing*. PhD thesis, CS Division (EECS), Univ. of California, Berkeley, June 1991.

[Hec92a] P. Heckbert. Discontinuity meshing for radiosity. *Third Eurographics Workshop on Rendering*, pages 203–215, May 1992.

[Hec92b] P. Heckbert. Radiosity in flatland. *Eurographics*, 11(2), 1992.

[LTG92] D. Lischinski, F. Tampieri, and D. Greenberg. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics & Applications*, pages 25–39, November 1992.

[LTG93] D. Lischinski, F. Tampieri, and D. P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 199–208, August 1993.

[Moo36] P. Moon. *The Scientific Basis of Illuminating Engineering*. McGraw-Hill, 1936.

[NN85] T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 23–30, July 1985.

[PD86] W. Plantinga and C. Dyer. An algorithm for constructing the aspect graph. In *Proc. 27th Symp. Foundations of Computer Science*, pages 123–131, 1986.

[SG93] A. J. Stewart and S. Ghali. An output sensitive algorithm for the computation of shadow boundaries. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 291–296, Waterloo, Canada, 1993.

[SG94] A. J. Stewart and S. Ghali. Fast computation of shadow boundaries using spatial coherence and backprojections. *Computer Graphics Proceedings, Annual Conference Series 1994*, 28:231–238, Aug 1994.

[Tam93] F. Tampieri. *Discontinuity Meshing for Radiosity Image Synthesis*. Ph.D. thesis, Cornell University, Ithaca, NY, 1993. available as Cornell technical report 93–1346.

[Tel92] S. J. Teller. Computing the antipenumbra of an area light source. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 139–148, July 1992.