On Constructing and Visualizing the Topological Structure of the Visibility and Radiance of Architectural Models

Jared Hoberock[1], Samuel Hornus[2], and John C. Hart[3]

[1] NVIDIA Corp., jaredhoberock@gmail.com
[2] INRIA Sophia Antipolis, Samuel.Hornus@sophia.inria.fr
[3] University of Illinois Urbana-Champaign, jch@illinois.edu

Abstract. The visibility complex organizes the space of maximal segments in the free-space of a 3-D scene into a four-dimensional structure. The cells of the visibility complex have applications in image-based rendering, form-factor computations, global illumination, shadowing and many other methods, but to date only the 0-D and 1-D cells have been constructed. This paper expands the frontier of our understanding of the visibility complex by finding and adding the 2-D cells to the visibility complex. We demonstrate the capabilities of these 2-cells with a rendering application that sparsely samples the illumination of a scene, and interpolates these samples only within the boundaries of the 2-cells, thus avoiding aliasing effects while respecting sharp changes in radiance across boundaries in the visibility.

1 Introduction

In a closed 3-D scene, both sightlines and light transport occur along extremal line segments that extend between one surface point to a second surface point. Since surfaces are two dimensional, the topological space of these suface-to-surface segments is four dimensional subset of the Plücker quadric. This 4-D space is partitioned by the *visibility complex* (a generalization of the cell complex where "cells" need not be topologically disks) into subspaces whose dimensionality corresponds to the degrees of freedom of the maximal segments they represent.

The goal of this paper is to consider the visibility complex as a structure for the analysis of architecture. The visibility complex can be used to visualize and analyze the space of sightlines for both the beauty and security of interior spaces, or for urban warfare and sniper prediction in exterior cityscapes.

The visibility complex can also be used as a support for measuring radiance in a scene. This enables previsualization of an architectural design's light distribution for improving lighting efficiency, security and aesthetics. The radiance (power of light along a ray) is constant along a maximal line segment (in a vacuum), but unlike visibility it is not symmetric and each maximal line segment supports two radiances, one in each direction. Thus there is a 2:1 map from

the space of radiance to the space of visibility, and the "radiance" complex is a double cover of the visibility complex.

Image based modeling and rendering methods use a variety of spaces to organize spaces of radiance samples. The two-plane parameterization stores all of the external radiance passing through a double-paned window in $I^2 \times I^2$ [1, 2]. Surface light fields store a hemisphere of internal radiance samples over the surfaces in the scene S in the space $I^2 \times \partial S$ but requires a parameterization of every surface in the scene. The visibility complex's subdivision of the extremal line segment space seems a more natural space to store radiance samples.

Though the visibility complex of a 3-D scene is well understood [3], it has never been constructed. Its size is $O(n^4)$ which would be quite large for modern scenes of millions of polygons. The 1-skeleton (nodes, arcs) of the visibility complex has been constructed procedurally for on-demand evaluation of shadow boundary information that can be used for discontinuity meshing [4]. This paper constructs the visibility 2-skeleton which expands the utility of the visibility complex beyond discontinuity meshing to more substantial applications.

Contributions. Our study of the 2-cells of the visibility complex focuses on V cells, which represent the family of segments between two faces passing through a single vertex. We show how these V-cells can be interrogated by finding the 1-D VE cells on their boundary. We have extended the half-edge mesh representation to these VE cells such that finding the VE-cell boundary of a V cell is similar to walking a half-edge data structure around a mesh face. This analysis leads to the following contributions.

- **Construction.** A top-down construction of the full 3-D visibility complex, which includes the higher (2-, 3- and 4-) dimensional cells in a more natural setting than previous work [4, 3].
- **Implementation.** An implementation, complete with results and discussion, of the full 3-D visibility complex. Previous implementations have been limited to the 2-D visibility complex or the 1-skeleton of the 3-D complex [4]. Durand *et al.* [3] describe a construction algorithm for the 3-D complex but do not implement it.
- **Sampling.** As alluded to previously [3], the visibility complex can be used to store radiance, making it a natural data structure for image-based rendering. Our implementation allows us to not only store but organize radiance samples in light space.
- **Interpolation.** The full 3-D visibility complex allows us to interpolate these samples relative to visibility events, which extends Bala *et al.* [5], which was limited to a subset of the visibility 1-skeleton.

2 Previous Work

Data Structures. There have been a variety of data structures devised to organize information about visibility in a scene. A simple quadratic visibility graph [6] connects mutually visible vertices of a scene with edges, but ignores visual events

involving faces and edges. The aspect graph partitions space into viewpoints from which hidden-line renderings are isomorphic [7, 8], but an n-polygon scene leads to worst-case $O(n^6)$ orthographic and $O(n^9)$ perspective aspect graphs. The visibility complex [9, 10] organizes visual events into a more tractable data structure which is worst-case $O(n^4)$ but probabilistically $O(n^2.67)$ [3] which makes it attractive for exact visibility computations.

Discontinuity Sensitive Reconstruction. To accelerate global illumination algorithms, much research has been spent on sparse sampling and reconstruction. Early methods reconstructed images by interpolating samples without regard to discontinuities such as the edges of objects and shadows [11]. More recent approaches account for discontinuities explicitly in the reconstruction process [5]. The color at a pixel is approximated by interpolating only the nearby samples that can be reached without crossing a discontinuity, though until now these boundaries were limited to the visibility 1-skeleton.

Robust Epsilon Visibility. Duguet and Drettakis [12] construct a more robust implementation of the visibility 1-skeleton by merging small visibility features. They thicken vertices into spheres and edges into cylinders by an ϵ factor to avoid small features that create numerical problems and increase the size of the visibility complex. Our work on the 2-skeleton is based on their 1-skeleton approach, and we have used their ϵ-thickening to avoid small features in our construction of the 2-skeleton.

3 Background

Half-Edge Mesh Datastructure. Let $S \subset \mathbf{R}^3$ be a closed polygonal manifold scene represented by a half-edge data structure [13]. The directed edge $\mathsf{e} \in S$ emanates from the vertex $\mathsf{e.start}$ and terminates at the vertex $\mathsf{e.end}$, whereas its opposite $\mathsf{e.opp}$ extends from $\mathsf{e.end}$ to $\mathsf{e.start}$. The next edge counterclockwise when viewed from outside the manifold is $\mathsf{e.next}$ and these two edges share the face $\mathsf{e.left}$.

Segment Space. The collection of all oriented segments is $\mathbf{R}^6$, the space consisting of all line segments from $\mathbf{x} \in \mathbf{R}^3$ to $\mathbf{y} \in \mathbf{R}^3$. The collection of undirected segments is *segment* space, the quotient of $\mathbf{R}^6$ with the equivalence $(\mathbf{x}, \mathbf{y}) = (\mathbf{y}, \mathbf{x})$ which essentially folds it in half.

Visibility Complex. The visibility complex $\mathcal{V}(S)$ is the space of maximal free segments in a scene S. Since the endpoints of these segments must lie on surfaces to be of maximal length, the segments have four degrees of freedom (two for each surface-constrained endpoint) and thus $\mathcal{V}(S)$ is four-dimensional.

The visibility complex organizes maximal segment space into cells corresponding to the dimensionality of a family of segments. The family of maximal segments between two faces form 4-cells. The collection of segments tangent to a

single edge form a 3-cell, since one dimension is reduced by the loss of a degree of freedom do to the constraint that the segments intersect the edge. The segments tangent to two edges (or a vertex if the two edges meet) form a 2-cell (a patch in maximal segment space). The segments tangent to three edges (or a vertex and an edge) form a 1-cell (an arc in maximal segment space). Finally, the segment tangent to four edges (or a vertex and two edges, or two vertices) form a 0-cell, e.g. a point in maximal segment space.

The boundaries of the families of segments in each cell occur when the segments reach an additional edge. This organizes these cells into a complex. Thus segment space is organized into the *visibility complex* $\mathcal{V}$, though we must be careful to note that each "cell" in this complex is a manifold-with-boundary of the appropriate dimension. (In other words, a cell may not necessarily be homeomorphic to a disk. It may have a hole.) We denote by $\mathcal{V}_k(S)$ the k-skeleton of the visibility complex of scene S containing the subset of cells of dimension k or less.

$\mathcal{V}_0$ is a collection of 0-cells of type E4, VEE and VV corresponding to maximal free segments with no degrees of freedom. We can construct $\mathcal{V}_0$ by enumerating all combinations of four edges and determining which can be stabbed by a single free segment. We include the endpoints of the edges, which yield VEE and VV nodes when the free segment stabs a vertex shared by the edges.

4 The Visibility 1-Skeleton

We first construct its 0-cells $\mathcal{V}_0(S)$ with an $O(n^4)$ iteration over all sets of four edges. Each set of four edges can yield zero, one or two E4 cells [14]. These sets of four edges may contain one or two pairs that share endpoints, yielding VEE- and VV-cells respectively.

We construct the 1-skeleton $\mathcal{V}_1(S)$ of the visibility complex following the methods of Durand *et al.* [4], and Duguet *et al.* [12], but with the addition of oriented VE-cells computed on a half-edge mesh data structure that eases the implementation of the topological reasoning needed to connect the 1-cells to each other through the existing 0-cells.

4.1 VE Half-Cells

Each VE-cell corresponds to the set of maximal free segments passing through a vertex v and a maximal subset of an edge e. We first define a canonical VE half-cell as

$$\mathsf{halfve} = (\mathsf{upBlocker}, \mathsf{vertex}, \mathsf{edge}, \mathsf{downBlocker}, \mathsf{startPoint}, \mathsf{endPoint}). \quad (1)$$

The upBlocker and downBlocker correspond to faces the family of segments extend between, and the half-cell is oriented such that the "down" direction (e.g. downstream) is *from* the vertex *to* the edge.

Using the values indicated in Fig. 1 we represent a VE cell by the two half-VE-cells

$$\mathsf{ve} = (\mathsf{up.right}, \mathsf{v}, \mathsf{e}, \mathsf{e.left}, \mathsf{start}, \mathsf{end}), \tag{2}$$

$$\mathsf{ve.opp} = (\mathsf{up.left}, \mathsf{v}, \mathsf{e.opp}, \mathsf{down.left}, \mathsf{end}, \mathsf{start}). \tag{3}$$

Recall e.opp extends from e.dest to e.org [13].

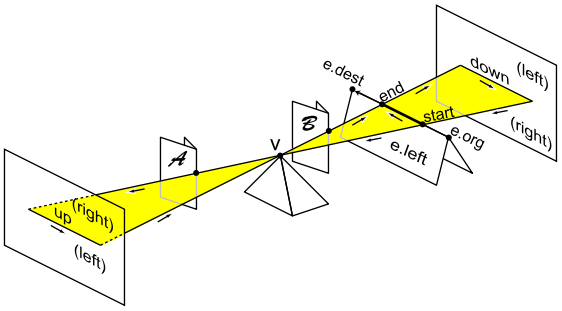

Fig. 1. An oriented VE cell representing the set of maximal segments passing through vertex v and directed half-edge e. In this example, occluders $\mathcal{A}$ and $\mathcal{B}$ reduce the portion of the edge generating the VE cell to the segment from start to end. The arrows indicate the propagation of the orientation of e to edges up and down embedded in the up and down blockers, respectively.

Each VE half-cell has a blocker pair corresponding to the two faces that terminate the family of segments. Dividing the VE cell into two half cells differentiates the blockers on either side of the VE family of segments. The blocker pair $(\mathsf{upBlocker}, \mathsf{downBlocker})$ for the half-cell ve are $(\mathsf{up.right}, \mathsf{e.left})$ whereas $(\mathsf{up.left}, \mathsf{down.left})$ are the blocker pair for ve.opp, as illustrated in Fig. 2.

The edge up is a projection of e through the vertex v. In general position, this projection falls in the middle of a face, and up.left = up.right return a pointer to this face. However, the projection of the edge e through a vertex v may fall on an edge, in which case up.left $\neq$ up.right, which occurs often in highly aligned (e.g. architectural) scenes.

When v and e share the same face, then we indicate this condition by setting the downBlocker member of the ve half-cell to the vertex v.

The edge e need not be a silhouette edge in v's view. In this case v can see both e.left and e.right, and we set the downBlocker member of ve.opp half-cell to e.right.

The vertex v may occur on the side of an object such that its "up" side is completely interior to the object. In this case, the vertex v serves as the upBlocker member for both ve and ve.opp.

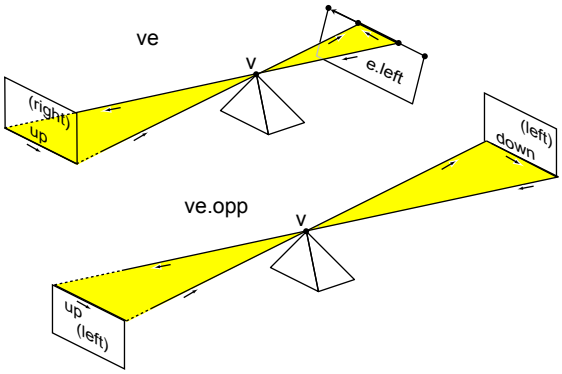

Fig. 2. A VE cell and its opposite.

4.2 EEE-Cells

The 1-skeleton $\mathcal{V}_1 \supset \mathcal{V}_0$ augments the 0-skeleton with 1-cells of type EEE and VE that serve as arcs connecting the 0-cells in $\mathcal{V}_0$. These 1-cells describe a 1-parameter family of lines and the direction of parameterization is useful when determining their topological organization in the visibility complex (e.g. to which 0-cells they attach). The VE cell is parameterized by its single edge. The E^3 cell, given as $\{\mathsf{e}_1, \mathsf{e}_2, \mathsf{e}_3\}$, can be parameterized by any of its three edges. In fact, we use the expression

$$\operatorname{sgn}\left(\frac{d\mathsf{e}_i}{d\mathsf{e}_j}\,|_{\{e_i,e_j,e_k\}}\right)$$

to determine the parametric orientation of an EEE cell, where the derivative $d\mathsf{e}_i/d\mathsf{e}_j$ represents the change in the position of the stabbing line along edge i that occurs in response to a change in the position of the stabbing line along edge j. These derivatives are constant for a given EEE cell and can be computed exactly using forward differences.

5 2-Skeleton Construction

The visibility complex contains two kinds of 2-D elements. A V cell represents the space of all maximal free segments stabbing a vertex between the same two faces, and resembles a double prism. An EE cell represents the space of all maximal free segments stabbing two edges, which forms a tetrahedron between the two edges, along with the extensions of the segments beyond the edges.

5.1 V Cells

We will focus on the V cell. Formally, a V cell is defined

$$\mathsf{vcell} = (\mathsf{up}, \mathsf{v}, \mathsf{down}, \{\partial_i\}) \tag{4}$$

where up and down are the two face blockers (in either order) and $\{\partial_i\}$ are a list of the outer ∂_0 and possibly inner $\partial_{i\geq 1}$ boundaries of the V cell, each represented as closed loops of VE-cells. Since there is only a vertex and two blockers, there is no basis for a preferred orientation with respect to up and down. However, if v is one of the blockers, then it is always the "up" blocker. Some sample V cells appear in Figure 3.

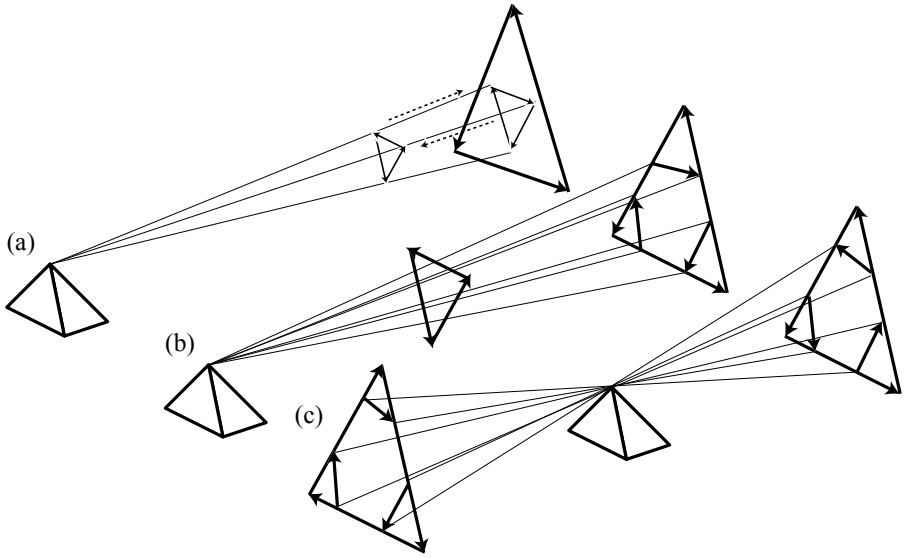

Fig. 3. Examples of V cells. A V cell is formed from the vertex to the small triangle (a), which creates a hole in the second V cell on the larger triangle. The V cell for the smaller triangle in (b) creates three V cells on the larger triangle. The V cell between two triangles (c) is a projected hexagon.

Figure 3(a) demonstrates that a V cell may not be simply connected. It may contain one or more holes each created by an occluder between v and one of its blockers that projects to the interior of the blocker. There does not exist a path of visual events which connect the inner boundary to the outer boundary of the VE cell, which is one reason the interrogation of 2-cells has been elusive.

5.2 V-Cell Boundaries

The VE cells can be linked into a chain based on the connectivity of the oriented half-edge segments. In other words, the end of one VE cell will correspond to the start of the next VE cell in a chain. These chains eventually form closed loops, such that the end of the tail VE cell corresponds to the start of the head VE cell. In general position, these VE-cell chains are simple, but in degenerate cases these chains can merge and divide at 0-cells.

Given a VE half-cell and its corresponding blocker pair (upBlocker, downBlocker) we form a boundary loop that wraps around this blocker pair. For example, the hexagonal V cell in Fig. 3(c) consists of six ve half-cells of type (2), though their "downstream" direction from vertex to edge alternates. The V cell corresponding to the top of the rear triangle in Fig. 3(b) consists of one ve.opp and two ve half-cells. Thus we interrogate V-cells in $\mathcal{V}_2$ by finding chain boundary loops of VE half-cells which is very similar to finding faces in a manifold mesh by traversing half-edges.

Since we know a VE cell's opposite exists, we store only a single half-VE-cell and discard its opposite. If we find a half-VE-cell whose orientation is the opposite of what is needed, we reconstruct its opposite on the fly.

As shown in Fig. 3(a), a V-cell can contain a hole, which means the family of lines from a vertex through a blocker pair can completely surround another V-cell.

In general, the ve half-cells (2) wrap counterclockwise, constructively surrounding the VE cell, whereas the ve.opp half-cells (3) wrap clockwise, destructively trimming the V-cell.

5.3 V Cell Enumeration

The V cells of the visibility complex are interrogated by the algorithm described in Fig. 4.

$VE = \text{FindVECellsWithVertex}(\mathcal{V}_1, v)$
Foreach $\{b1, b2\} \in \text{AllBlockerPairs}(VE)$
    $VEb1b2 = \text{FindVECellsWithBlockers}(VE, \{b1, b2\})$
    $L = \text{FormLoops}(VEb1b2)$
    $\{\partial_i\} = \text{OuterInnerBoundaries}(L)$
    $V = V + (b1, v, b2, \{\partial_i\})$
Endfor

Fig. 4. V Cell enumeration algorithm.

This algorithm begins by creating a set VE containing all VE half-cells generated by vertex v. We then pick each blocker pair in turn and seek to establish a V cell for it. The variable $VEb1b2$ contains all of the VE half-cells that contain $b1$ and $b2$ as blockers, in either order.

We then take these VE half-cells and form loops. The procedure FormLoops($VEb1b2$) picks a random VE half-cell from $VEb1b2$ and begins traversing it, finding another VE half-cell in $VEb1b2$ whose start point corresponds to the current cell's end point. This process repeats until the sequence of VE half-cells loops back onto itself, at which point it is stored as a loop L_i and another random VE

half-cell is chosen to initiate the next boundary loop. This procedure terminates when all VE half-cells have been added to loops, and the loops are returned into the list L.

After their creation, each boundary loop must be identified as the outer boundary of a V cell or as the inner boundary indicating a hole in the V cell. We assume a closed scene, or at least that one of the blockers is a polygon (e.g. we do not consider $\{\infty, v, \infty\}$ V cells) so we can project the boundary loop of a V cell onto the plane P of one of its blockers.

An outer boundary loop will wind counterclockwise about the V cell, whereas an inner boundary loop will wind clockwise about the hole it creates. We compute the winding of the polygon as the sum of the turning angles at each vertex of the projected boundary loop. If this sum is 2π then the boundary loop winds counterclockwise, otherwise the sum is -2π, indicating a clockwise winding.

5.4 Matching Inner Chains With Outer Chains

After chains have been created, and outer and inner chains (holes) identified, there is the problem of deciding which holes go in which outer boundary. A solution cannot be inferred, since we note that there is no topological connection between a hole and a outer boundary. We must use geometry: it is a non-convex polygon inside non-convex polygon problem.

The procedure *Find-Holes,* which takes as parameters a single outer boundary ∂, and a list of possible holes (inner chains) H, returns a list of holes $\overline{\Delta}$, contained in ∂. Let $\mathbf{P}$ be the plane of one blocker of the V cell. Once the chains projected on $\mathbf{P}$, we count the intersections between the (projected) outer chain and a ray cast from a vertex of the potential inner chain in some direction on the plane. An odd number indicates that the inner chain is inside the outer chain.

5.5 Reconciling V Cells with Robust Epsilon Visibility

Our previous general position assumption ensures that the *next* operation at a 0-cell is unambiguous. However, if we allow our scenes to contain degeneracies, what potential issues arise?

Consider the figure 5. The extremal stabbing line natively generated by $(v, v2)$, also stabs edge $e5$ degenerately. Figure 6 shows there are two VE cells originating at $vv2e5$ winding counter-clockwise to any V cell of the form $\{upBlocker, v, f2\}$.

How do we deal with this problem? Ideally, we want some function to map a unique previous VE cell terminating at a 0-cell to a unique next VE cell emanating from the 0-cell. Note that at each 0-cell, the number of terminating VE cells must equal the number of originating VE cells. This implies there must be some mapping. Our mapping is arbitrary. When performing the next (previous) operation, we collect all the next VE cells and all the previous VE cells into two lists, N and P, respectively. Sort the list by some arbitrary unique ID per VE cell (pointer value). In this way, we can create a bijection from previous VE cells to next VE cells. This mapping requires that the next (previous) operation at a 0-cell takes as an argument the previous (next) VE cell visited.

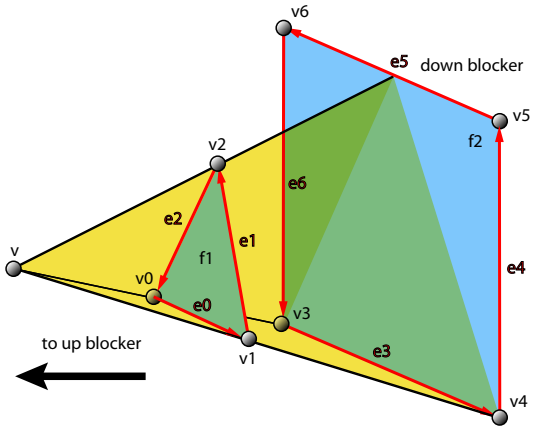

Fig. 5. A degenerate scene. The 0-cell $vv2e5$ stabs a degenerate edge.

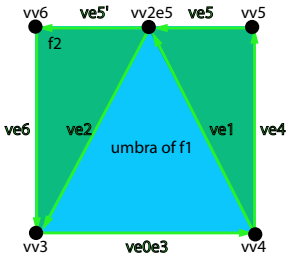

Fig. 6. The view through v of the degenerate scene of figure 5. Note that there are two next cells at $vv2e5 : ve5'$ and $ve2$.

6 Enumeration of EE-cells

The procedure to enumerate EE-cells proceeds almost in the same way. Two notions need to be defined for EE-cells boundaries: the notion of being locally to the right or to the left of the EE-cell, and the notion of (counter-)clockwise winding around a cell.

To do this we first describe a handy EE-cell parameterization. An EE-cell E is a connected set of segments tangent to two edges e_1 and e_2. We parameterize each edge e_i with $x_i \in [0, 1]$. E is thus represented as a connected region in the square $[0, 1]^2$. Its boundaries are made of hyperbolas' arcs [15].

With regard to this parameterization of EE-cells, the notion of the cell lying locally to the left of to the right of it's boundary elements can now be defined. The winding direction can also be computed as explained in figure 7.

We now have all the tools necessary to enumerate the EE-cells in the same way as we did for V-cells.

7 Applications and Results

We present here possible applications of the 2-skeleton. The first ones are presented here as possible future work, while we implemented the last two, for which we present our results.

A major applications of V-cells construction is that of computing the view of a point. Consider a point P in space (either a scene vertex or not). All V-cells adjacent to P consitutes the visibility polyhderon V_P of P, i.e., they encode all scene points visible from p. Applications of V_P include:

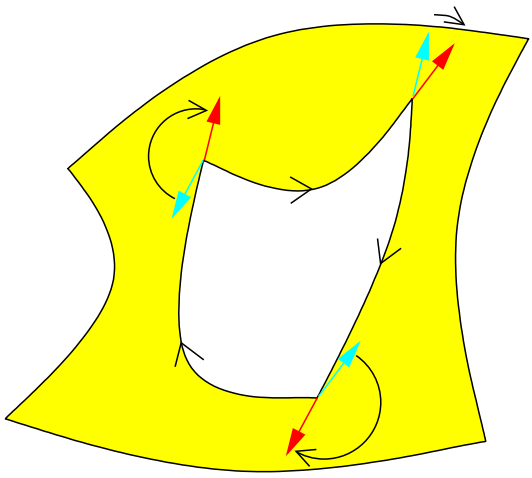

Fig. 7. Computing the winding of a chain in an EE-cell. Circle arcs angles are added. Red arrow are tangent vector at the end of each arc. Blue arrows are copies of the previous vertex' tangent vector.

7.1 Hidden Surface Removal (HSR)

Clearly, V_P describes which parts of which polygons are visible from P. One can extract from it the set of visible polygons (the set of blockers of V-cells inside V_P). Alternatively, one can gets more precise HSR by cutting every polygon along their intersection with the boundaries of the V-cells of V_P.

7.2 Shadow Boundaries

Let L be a point light source. It is straightforward to compute the boundaries of shadows it casts. The situation is illustrated in figure 8. We see some applications of this:

If the light does not move, we can precompute a highly efficient shadow volume cast by the fixed parts of the scene. The precomputed shadow volume is efficient since it contains no "shadow inside shadow", thus speedind up stencil shadow algorithms (there are less extruded quads to rasterize). An alternative to the stencil shadow algorithm becomes possible if the geometry has been cut as in the previous section about HSR: For each point-light, store the list of visible (cut) polygons. Render the whole scene unlit. Then for each point light, render the list of lit polygons with illumination.

7.3 Shooting photons on specular objects

In the photon mapping algorithm [16], a secondary photon map called *Caustic photon map* is used to render sharper caustics. When constructing this map,

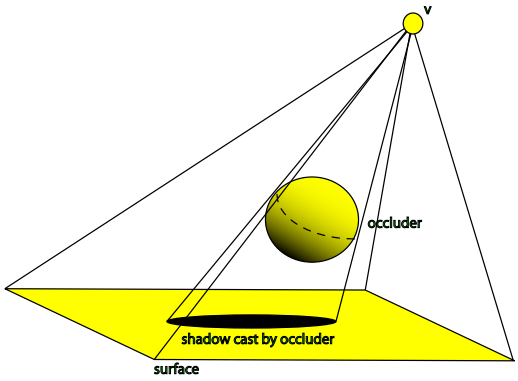

Fig. 8. The shadow boundaries cast by a point light and an occluder create a hole in the V-cell defined by the light and a surface. Inside the hole is another V-cell: the illuminated portion of the occluder.

photons are shot toward highly specular objects only. One of the problems arising is to acurately sample directions leading to these specific objets in the scene. Computing the set V_P of V-cells around sample points P on an area light can help in choosing a direction for each caustic photon and in evenly distributing photons in allowed directions (where specular objects lie). We believe the speed bottleneck could be overcome since the whole set V_P need not be computed: one needs only V-cells one of whose blocker is a specular object.

The following subsections present applications of V-cells that we did implement.

7.4 Point to polygon exact form-factor computation

In a ray-tracer, the direct illumination at a point P by an area light is generally computed by sampling the area light. If we compute V_P, not only can we use an analytical solution to compute this direct illumination, but one can also achieve better sampling of the directions around P for integrating indirect illumination in a monte-carlo algorithm, as in the previous section. The exact direct illumination of P from an area light can be computed since we know the exact parts of the area light that are visible from P, and in such a case, a well known formula gives us the form factor [17]. Figure 9 shows 2 ray-traced images with exact direct illumination.

7.5 Radiance Discontinuities for Reconstruction from Spare Samples

If C is the camera location, then the shape of the V-cells in V_C are the visible parts of the scene, as seen in subsection 7.1. In particular, when projected on the

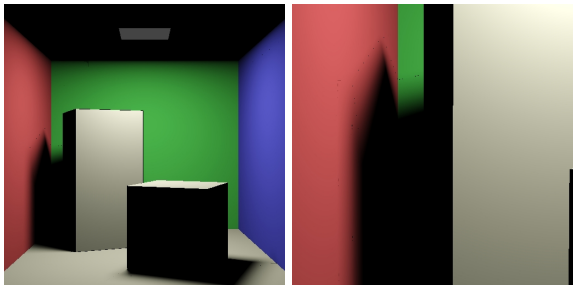

Fig. 9. *Left.* Ray-traced Cornell box with exact direct illumination from an area light. *Right.* Zoom on the shadow of the larger box. Notice some black pixels at the boundary between full light and penumbra, probably due to bugs in our implementation. Rendering time was about 27 minutes for each of these 512x512 images. However our implementation does not employ any optimization strategies.

image plane, their boundaries form the loci of radiance discontinuities. Using this we are able to subdivide the image plane into regions, each being the boundary of a visible polygon. We used this subdivison for rendering an image using a few illumination samples only: each region is triangulated using the samples locations as vertices. The benefit is that no illumination samples bleeds across visible edges of the scene.

Figure 10 illustrates this. On the top left, one can see the regions corresponding to V-cells in V_C. The top right picture shows the same regions from a different point of view, illustrating what has been computed; notice that this in essence corresponds to hidden surface removal. Below, the left images show the triangulated regions. The illumination has been computed only at the vertices of the triangles. The right images are shown without the triangulations. The number of samples is, from top to bottom, 100, 214, 514 and 1014. The top right image has been created using 2014 samples.

In the top right image, we can note a wrong interpolation of the shadow at the bottom left of the larger box. This is due to the simplistic heuristic we used for the addition of samples. Here is how one could overcome this problem:

Assume we have computed the sets V_P of each vertex P of a polyhedral area light, together with the sets E_e of EE-cells adjacent to each edge e of that area light. The boundaries of the 2-cells in $\bigcup_{P,e} V_P \cup E_e$, when projected on their blockers, draw the loci of discontinuities in illumination. An offline preprocessing step could insert these shadow boundaries into a discontinuity mesh used for rendering later Also, sampling an EE-cell in the $[0,1]^2$ parameterization can be used as indicators of important places where to place samples on the scene polygons for reconstruction from sparse samples [5]. This is indeed where penumbra variation are the more subtle. We did not however implement EE-cells, and were therefore unable to detect the loci of penumbrae in our test scene, hence the artifacts in the penumbrae in figure 10.

References

1. Levoy, M., Hanrahan, P.M.: Light field rendering. In: Proc. SIGGRAPH 96. (August 1996) 31–42
2. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proc. SIGGRAPH 96. (August 1996) 43–54
3. Durand, F., Drettakis, G., Puech, C.: The 3d visibility complex. ACM TOG **21**(2) (April 2002) 176–206
4. Durand, F., Drettakis, G., Puech, C.: The visibility skeleton: A powerful and efficient multi-purpose global visibility tool. Proc. SIGGRAPH 97 (Aug. 1997) 89–100
5. Bala, K., Walter, B., Greenberg, D.P.: Combining edges and points for interactive high-quality rendering. In: Proc. SIGGRAPH 03. (July 2003) 1–2
6. Welzl, E.: Constructing the visibility graph of n line segments in the plane. Information Processing Letters **20** (1985) 167–171
7. Gigus, Z., Malik, J.: Computing the aspect graph for the line drawings of polyhedral objects. IEEE Transactions on Pattern Matching and Machine Intelligence **12**(2) (Feb. 1990)
8. Plantinga, H., Dyer, C.R.: Visibility, occlusion, and the aspect graph. International Journal of Computer Vision **5**(2) (1990) 137–160
9. Pocchiola, M., Vegter, G.: The visibility complex. International Journal of Computational Geometry and Applications **6**(3) (Sept. 1996) 279–308
10. Durand, F., Drettakis, G., Puech, C.: The 3d visibility complex, a new approach to the problems of accurate visibility. Proc. Eurographics Workshop on Rendering (June 1996) 245–256
11. Walter, B., Drettakis, G., Parker, S.: Interactive rendering using render cache. In: Proc. Eurographics Rendering Workshop. (June 1999) 19–30
12. Duguet, F., Drettakis, G.: Robust epsilon visibility. In: Proc. SIGGRAPH 02. (July 2002) 1–2
13. Guibas, L., Stolfi, J.: Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. ACM Trans. on Graphics **4**(2) (1985) 74–123
14. Teller, S., Hohmeyer, M.: Determining the lines through four lines. J. Graphics Tools **4**(3) (1999) 11–22
15. McKenna, M., O'Rourke, J.: Arrangements of lines in 3-space: A data structure with applications. In: Proc. Symp. Comp. Geom., ACM (1988) 371–380
16. Jensen, H.W.: Global illumination using photon maps. In Pueyo, X., Schrder, P., eds.: Rendering Techniques 96, Springer-Verlag (1996) 21–30
17. Sillion, F., Puech, C.: Radiosity and Global Illumination. Morgan Kaufmann Publishers, San Francisco (1994) ISBN 1-55860-277-1.

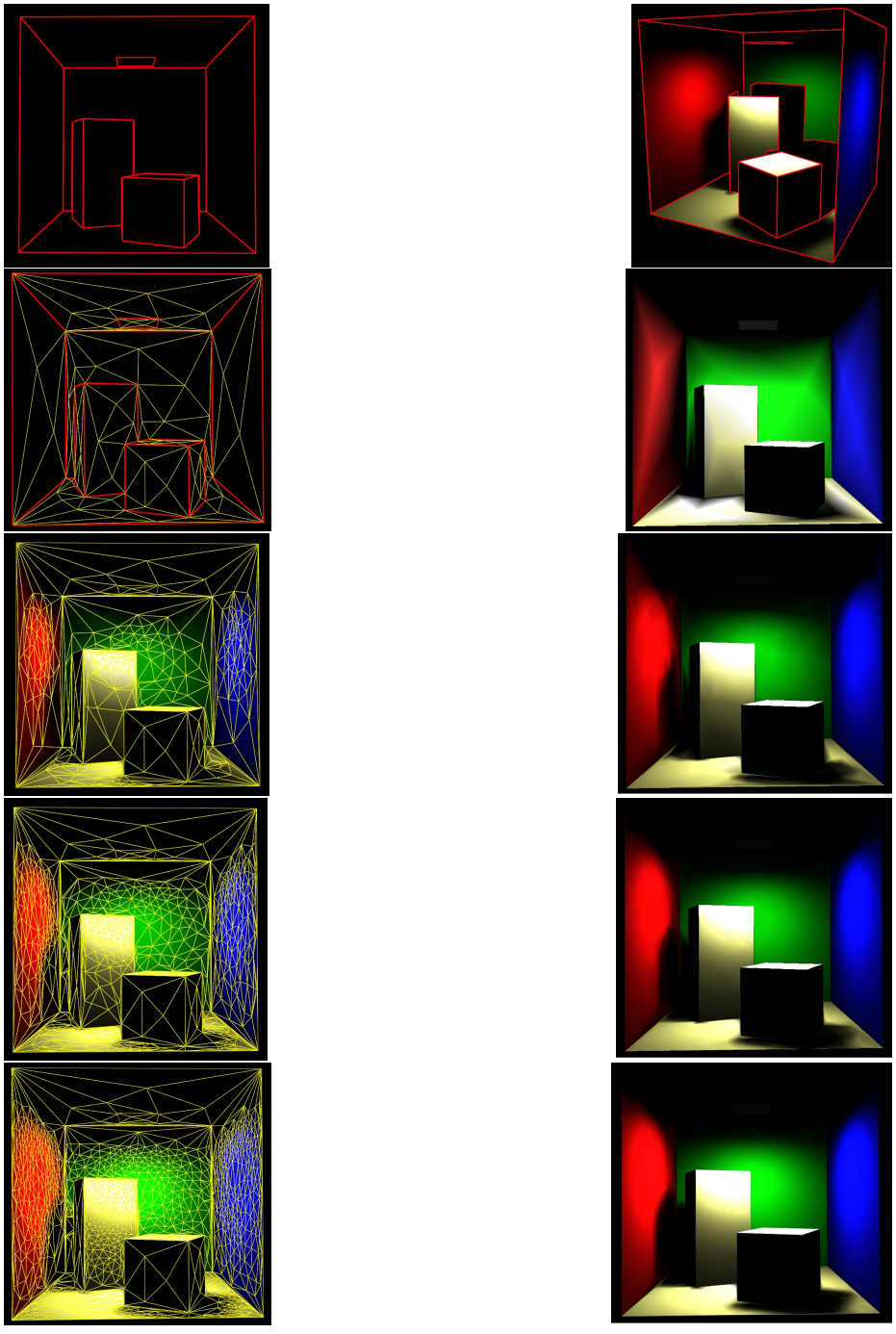

Fig. 10. Direct illumination reconstruction from spare samples. Notice the wrong interpolation of the shadow at the bottom left of the larger box. This is due to the simplistic heuristic we used for the addition of samples. Light at the bottom left of the smaller box can be seen because the boxes don't touch the ground. (These images look different than those in figure 9 because of a different tone-mapping.)