Applications of Computational Geometry to Volume Visualization

PETER L. WILLIAMS
University of Illinois at Urbana-Champaign

1 Introduction

The purpose of this extended abstract is to report on some recent applications of computational geometry to the field of volume visualization, and to outline a number of unsolved computational geometry problems in that field.

In computational science, the solution generated by the finite element method will often be a data set consisting of several scalar values defined at each vertex of a 3D mesh. These meshed data sets can be large, typically with from 100,000 to 5,000,000 polyhedral cells. One rendering technique used to visualize a scalar field, is to display a 2D projection of a colored semitransparent 3D volume or cloud, where the color and opacity of the 3D cloud are functions of the scalar field. This technique is called *direct volume rendering*. It can be accomplished by ray tracing or by projection of each cell onto the screen in visibility order. This latter method is referred to as *direct projection volume rendering* (DPVR) [4, 5, 6, 7]. A visibility ordering of a set of objects from some viewpoint is an ordering such that if object a obstructs object b, then b precedes a in the ordering. The visibility ordering is needed in DPVR so that the color and opacity can be correctly composited at each pixel.

Ray traced images typically take from 15 minutes to several hours to generate. With DPVR, it is possible to create images interactively so the scientist can get a holistic feeling for the spatial orientation of the field and the location of its extrema by rotating the image. We refer to this as interactive DPVR. The goal of interactive DPVR is to give a general idea of the scalar field; therefore approximations in the process are acceptable. This is important because it is unlikely that very large data sets can be displayed interactively without approximations.

For interactive DPVR of a data set of any finite size, defined on a mesh of any topological or geometric shape, problems in four areas need to be addressed: visibility ordering, point location, cell projection and opacity calculation, and lower resolution meshes. Mesh preprocessing methods that maximize run-time performance, and efficient serial and parallel run-time algorithms are required in all four areas. Efficient implementations of both the preprocessing and run-time algorithms are also required; this often entails serious consideration of degeneracies. Since the data sets can be large, efficient use of storage is essential; $O(n^2)$ storage is generally out of the question. After some preliminary definitions, we take up each of the four areas, and describe the current solutions and outline what is still needed.

2 Preliminary Definitions

A *mesh* is a cell complex S in $\mathbf{E}^3$, whose cells are convex three-dimensional polytopes, such that for any partition of S into two subsets there is always at least one polygon that is a facet of a cell from each subset. If a facet f of some cell in a mesh S is not shared by any other cell in S, then f is an *exterior facet*. The union of all exterior facets of S constitutes the *boundary* of S. An *exterior cell* has at least one exterior facet. If the boundary of a mesh S is also the boundary of the convex hull of S, then S is called a *convex* mesh; otherwise it is called a *nonconvex* mesh. Let S be a nonconvex mesh. A three-dimensional polytope p, not necessarily convex, which is not a member of S, such that each facet of p is shared by some cell in S, is called a *void*. If p is convex, then we call it a *convex void*. A nonconvex mesh may have zero or more voids. The union of all facets of a void is referred to as the *void boundary*. The union of the facets in the boundary of S that are not facets of a void is the *outside boundary* of S. A nonempty region between the boundary of the convex hull of a mesh S and the outside boundary of S is referred to as a *cavity*. In some

cases a cavity may be a tunnel or network of tunnels. The cells in a mesh may have differing numbers of facets; often, however, all cells will be tetrahedra.

3 Visibility Ordering

3.1 The Meshed Polyhedra Visibility Ordering Algorithms

The Meshed Polyhedra Visibility Ordering (MPVO) Algorithm, described by Williams and Shirley [8], is a simple and efficient algorithm for visibility ordering the cells of any acyclic convex set of meshed convex polyhedra. This algorithm, based on ideas in Edelsbrunner's paper on the acyclicity of cell complexes [1], orders the cells of a mesh in $O(n)$ time using $O(n)$ space, where n is the size of the mesh. The Binary Space Partition (BSP) tree algorithm [3] is not suitable for visibility ordering large meshes because splitting planes are used which could result in an explosion in the total number of cells.

DEFINITIONS: An *obstructs* relation is formally defined in [1, 8]. The *behind* relation $<_{vp}$ is defined such that $p_1 <_{vp} p_2$ if and only if p_1 and p_2 are adjacent, that is, share a facet, and p_2 obstructs p_1. Diagrammatically, we represent $p_1 <_{vp} p_2$ as an arrow through the facet shared by p_1 and p_2 pointing from p_1 to p_2. We denote the transitive closure of $<_{vp}$ as $<^*_{vp}$. If for every viewpoint, the obstructs relation on a mesh S is acyclic, then we say S is an *acyclic mesh*.

THE MPVO ALGORITHM: Given an acyclic convex mesh S, construct the adjacency graph G for the cells of S. For a given viewpoint vp, for every pair of adjacent cells p_1 and p_2 in S, compute the $<_{vp}$ relation. If $p_1 <_{vp} p_2$, then direct the corresponding edge in G from p_1 to p_2, otherwise from p_2 to p_1. G is now a directed acyclic graph (DAG). Perform a topological sort of G; the resulting ordering will be a visibility ordering of the cells of S.

The adjacency graph G and the plane equation coefficients for each shared facet can be computed and stored once in a preprocessing step. Then, for any given viewpoint, the direction to assign to each edge in G can be determined as follows. Each edge corresponds to a facet shared by two cells. That facet defines a plane which in turn defines two half-spaces, each containing one of the cells. The direction of the edge is towards the cell whose half-space contains the viewpoint. To compute this direction, the plane equation for each facet can be evaluated with the coordinates of the viewpoint.

DEFINITIONS: If certain nodes of DAG G are marked, the set $\text{upreds}^*(b)$ is defined as follows. If b is marked then $\text{upreds}^*(b) = \emptyset$, otherwise $\text{upreds}^*(b)$ is the union of $\{b\}$ and $\text{upreds}^*(a)$ over all predecessors a of b. Let S be a mesh and vp a viewpoint, if p is an exterior cell in S which has an exterior facet f such that the plane defined by f separates vp and p, then we say p is a *front facing* exterior cell. Let dist() be the Euclidean metric and cen() be the centroid. Let S be an acyclic mesh. If for any viewpoint vp, (1) there exist front facing exterior cells p and q in S such that $\text{dist}(vp, \text{cen}(q)) \geq \text{dist}(vp, \text{cen}(p))$, and either (a) there exists a cell $c <^*_{vp} q$ such that c obstructs p, and $p \not<^*_{vp} c$, or (b) q obstructs p, and $p \not<^*_{vp} q$, and (2) for all front facing exterior cells r in S with $\text{dist}(vp, \text{cen}(r)) > \text{dist}(vp, \text{cen}(q))$, $p \not<^*_{vp} r$, then we call this an instance of a *boundary anomaly* (BA).

THE MPVO ALGORITHM FOR NONCONVEX MESHES: Given an acyclic mesh S, construct the adjacency graph G for the cells of S. For a given viewpoint vp, initialize a list L to empty. Place all front facing exterior cells in S on L. Sort the cells on L by decreasing distance from vp to the centroid of the cell. For every pair of adjacent cells p_1 and p_2 in S, compute the $<_{vp}$ relation. If $p_1 <_{vp} p_2$, direct the corresponding edge in G from p_1 to p_2, and otherwise from p_2 to p_1. For each cell c on L, starting with the leftmost one, mark and output in topological order the cells of $\text{upreds}^*(c)$. (If a cell d is marked, then by definition of a topological sort, all cells q such that $q <^*_{vp} d$ will also be marked.) Provided S has no boundary anomalies, the output will be a visibility ordering of the cells of S.

Efficient methods are needed for detecting the occurrence of a BA during an ordering and for efficiently determining whether a mesh has any BAs. The MPVO Algorithm for Nonconvex Meshes has been tested on a number of actual computational meshes and it has been found that it gives a correct ordering for all viewpoints examined; therefore it has proven quite useful. However, it is easy to generate counterexamples by hand. We regard this algorithm as a heuristic that is quick and easy to implement. A better $O(n)$ algorithm for visibility ordering nonconvex meshes is needed.

3.2 Cyclic Meshes

An efficient preprocessing algorithm is needed for determining whether a mesh is acyclic for all viewpoints. Edelsbrunner [1] has proven an acyclicity theorem for the obstructs relation for the Delaunay triangulation (DT). The vertices of a cyclic mesh can be retriangulated with the DT to obtain an acyclic mesh. For some point set topologies, a DT can be $O(n^2)$ both in time and in size. An efficient parallel DT algorithm is needed. If a nonconvex mesh is retriangulated with a DT, it will be necessary to ensure that the facets in the original boundary are also facets in the new triangulation, and to mark all tetrahedra outside the original boundary as 'imaginary'. This is called a *conformed DT* [8]. Usually a conformed DT is possible only if new points are added. The process of adding points and retriangulating is repeated until no tetrahedron intersects the original boundary. The convergence properties of this process is an unsolved problem. A conformed DT algorithm in $\mathbf{E}^3$ is needed.

The DT is considered to be one of the three most attractive methods for 3D automatic mesh generation in computational science. Often a DT will be used to triangulate a nonconvex region by relaxing the DT criteria at the boundary. The incidence of cycles may be less than other completely non-DT methods; also, it is possible that cycles may occur only at the boundary area; but, there is no proof or analysis. This needs more investigation. The proof of acyclicity of the DT is based on exact arithmetic and the slightest inaccuracy can destroy the acyclicity property. Mesh generation algorithms usually are implemented in floating point. Therefore this issue also needs investigation.

For large cyclic meshes with a nonuniform vertex distribution, due to storage limitations it will not be possible to use the DT to retriangulate the mesh; therefore, another means of dealing with cycles is needed.

The MPVO Algorithm, using a depth first search (DFS) for the topological sort [8], will output an ordering of all cells of a cyclic mesh; but, the ordering will not be a correct visibility ordering for the cells involved in the cycle. In cases where the number of cells involved in the cycle is small in comparison to the total number of cells, it is possible, due to the high degree of abstraction inherent in the process of DPVR, that cycles may have no noticeable effect on the image. An efficient algorithm is needed to count the number of cells involved in cycles so this information can be made available to the user. When the MPVO Algorithm uses a breadth first search (BFS) [8], it will halt on encountering a cycle. The BFS version is preferred since it is parallelizable; whereas, the DFS version is not amenable to parallelization. Heuristics are needed to break cycles encountered by the BFS version and thus output all the cells even though the resulting ordering will not be entirely correct. The general problem of minimizing the number of cells incorrectly ordered, which is related to the feedback arc set problem, is NP-complete. Heuristics need to be developed for this problem.

Although the MPVO Algorithm is parallelizable [7], it is quite possible that an algorithm initially designed as a parallel algorithm could be more efficient. It is known that a plane or spherical sweep is easy to parallelize; therefore, it is possible that such a technique could be used to create a global ordering of all cells that could be used in conjunction with the local ordering established by the $<_{vp}$ behind relation.

3.3 Nonconvex Meshes

Meshes often are nonconvex. Therefore, an important area of research is to find ways to convert nonconvex meshes into convex meshes so the MPVO Algorithm can be used. This will entail algorithms, and their implementations, for locating voids and cavities, for determining whether voids and cavities are convex/nonconvex, and for triangulating nonconvex voids or cavities; see [8]. Another approach to investigate is the use of α-shapes [2], properly generalized from $\mathbf{E}^2$ to $\mathbf{E}^3$.

4 The Spatial Point Location Problem

In visualization, the solution of the spatial point location problem is required for interactive probing, stream line generation and ray tracing applications. The data structures for the MPVO Algorithm can be used to solve the point location problem. If the viewpoint is set to the query point, then, informally, for a convex mesh, start from any cell and follow any 'outbound' arrow to the next cell until a cell with no outbound arrows is reached; this is the target cell. If an exterior facet with an outbound arrow is reached, then the point doesn't lie within the mesh. The direction of the arrows is computed on the fly and not for all cells in

advance as is done in the MPVO Algorithm. A complete description is given in [8]. The mesh need not be acyclic. If the mesh is nonconvex a similar technique will work but is not as efficient; if the query point is outside the mesh, every cell will have to be searched. For nonconvex meshes, a better point location method utilizing the MPVO data structures is needed.

5 Cell Projection and Opacity Calculation

In DPVR a scalar field over a region of 3-space can be visualized by modeling that region as though it were filled with tiny glowing particles whose color R, G, B and density ρ are assumed to vary continuously over the region and are determined by four user-specified functions of the scalar field. In effect the region is represented as a semitransparent medium which emits its own light. A cell's contribution of color and opacity to each pixel is determined by integrating the color and density functions along a ray from the pixel through the cell [4, 5, 6]. The color $RGBvalue$ and opacity α are then composited at each pixel as, $newPixelRGBvalue = \alpha RGBvalue + (1 - \alpha) oldPixelRGBvalue$.

The ray integration is a time consuming process; also, the integrand exists as a closed form expression only in special cases. Therefore our goal is to develop a hierarchy of approximations that trade off accuracy for speed by taking advantage of the capabilities of existing graphics hardware. High-end graphics workstations have algorithms in hardware that, given the vertices of a polygon in world coordinates and viewing parameters, perform the transformation from world to screen coordinates, either as an orthographic (OP) or as a perspective projection (PP), and then linearly interpolate a color and opacity specified for each vertex over the pixels covered by the polygon. Polyhedron rendering hardware is not commercially available; it is necessary to decompose a polyhedral cell into a set of polygons in order to render it.

In the discussion that follows, it is assumed that a PP is used and that the cells are tetrahedra; however, an OP may also be used, and cells which are wedges, pyramids and bricks are quite common. A polyhedral cell c may be modeled by its polygonal projection onto the viewing plane, called its *footprint*, where it is assumed that the "thickness" t of the footprint at any pixel is $r \cap c$, where r is a ray from the viewpoint through the pixel, and a and b are the points where r enters and exits the cell. The cell's contribution of opacity to the pixel can be calculated as $(a - b)((\rho(a) + \rho(b))/2)$ and color as $(R(a) + R(b))/2$, etc. The compositing of these footprints in visibility order into an image is called *splatting*.

A gross approximation, but one which still results in the generation of useful images, is to render only the front facing faces of a cell, assuming each face is a slab of uniform thickness t [7]. Every front face of every cell is assumed to have the same thickness t regardless of viewpoint. The opacity for each cell vertex v is calculated in a preprocessing step as $t\rho(v)$. Since the MPVO data structure contains information about which faces are front facing, this approximation is very fast. A somewhat more accurate method [7] is to decompose the projected cell into from one to four triangles, as shown in [5], based on a case analysis of the number of front/back facing faces. There are four cases: one front facing face (FFF) and three back facing faces (BFF), or vice versa, which result in three triangles each of which uses the vertex opposite the single FFF or BFF as a non-zero thickness (NZT) vertex; two FFF and two BFF, which yield four triangles, each of which uses the centroid as a NZT vertex; two FFF and one BFF, or vice versa, which yield two triangles which use the vertex opposite the single FFF or BFF as a NZT vertex; one FFF and one BFF which yield one triangle which uses the vertex opposite the FFF as a NZT vertex. The opacity is set to zero at the zero thickness vertices. Each triangle will always have one NZT vertex v; at v, we assume a uniform thickness t for all cells and use the precalculated opacity $t\rho(v)$.

A much better approximation can be achieved if a new vertex v is introduced in the interior of a cell c so that v lies on a ray r from the viewpoint such that $r \cap c$ is maximized. This new NZT vertex v, which can be at the ray entry or exit point a or b or anywhere in between, is used along with the existing zero thickness vertices to triangulate the polygonal projection; v and its associated opacity and color are calculated at run-time. A method for quickly calculating a and b is needed, as is a method for approximating their calculation. Other algorithms for cell projection and opacity calculation are also needed to fill out the hierarchy of approximations. Any calculations that can be done as a preprocessing step, and that make minimal use of runtime memory, are valuable.

6 Lower Resolution Meshes

For very large data sets, it still may not be possible to generate images interactively, even with parallelization and the use of approximations for cell projection and opacity calculation. For example the hardware rendering pipeline may have an upper bound on the number of polygons per second that it can handle. Therefore, to achieve interactivity, it may be necessary to reduce the number of cells rendered by filtering techniques such as: rendering only selected cells or by coalescing adjacent cells. We refer to this concept as *lower resolution meshes* [7]. How best to create such meshes is an open question.

Rendering only selected cells will still require visibility ordering the entire mesh. The coalescing method can result in nonconvex cells not amenable to ordering by the MPVO Algorithm. One way to render only selected cells is by rendering the maximal independent set of the adjacency graph of the cells of the mesh. Such a result would seem to have a good distribution of the data points. A disadvantage of this method is that the degree of the nodes imposes bounds on the level of filtration allowed. More flexibility in the level of filtration can be achieved by selecting cells at random in the mesh, provided a truly random selection procedure is used. It is expected that this method will also have a nice distribution property. Another technique is to discard every n-th cell that is output from the visibility ordering, possibly tagging adjacent cells to prevent their removal. The use of shelling also needs to be investigated. More ideas in this area is needed.

Acknowledgement

The author is most grateful to Herbert Edelsbrunner for many very valuable conversations and to Allan Tuchman for a number of very helpful comments. Greg Shannon suggested the possibility of parallelization of a plane sweep in conjunction with the MPVO Algorithm.

References

[1] EDELSBRUNNER H. An Acyclicity Theorem for Cell Complexes in d Dimension. *Combinatorica 10 (3)* (1990), 251-260.

[2] EDELSBRUNNER, H. *Algorithms in Combinatorial Geometry.* Springer-Verlag, Heidelberg, 1987.

[3] FUCHS, H., KEDEM, Z., and NAYLOR, B. On Visible Surface Generation by A Priori Tree Structures. *ACM SIGGRAPH Comput. Gr. 14* (July 1980), 124-133.

[4] MAX, N., HANRAHAN, P., and CRAWFIS, R. Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions. *San Diego Workshop on Volume Visualization, Comput. Gr. 24 5* (Dec 1990), 27-33.

[5] SHIRLEY, P. and TUCHMAN, A. A Polygonal Approximation to Direct Scalar Volume Rendering. *San Diego Workshop on Volume Visualization, Comput. Gr. 24 5* (Dec 1990), 63-70.

[6] WILHELMS, J. A Coherent Projection Approach for Direct Volume Rendering. Tech. Rep. UCSC-CRL-90-38, Computer Research Laboratory, University of California, Santa Cruz, Aug. 2, 1990.

[7] WILLIAMS, P. L. Issues in Interactive Direct Projection Volume Rendering of Nonrectilinear Meshed Data Sets. Work in Progress Report, San Diego Workshop on Volume Visualization Dec. 1990, available as Report 1059, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign Dec. 1990.

[8] WILLIAMS, P. L. and SHIRLEY, P. Visibility Ordering Meshed Polyhedra. Report 1097, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign April 1991.