Flow Volumes for Interactive Vector Field Visualization

Nelson Max
Barry Becker
Roger Crawfis
Lawrence Livermore National Laboratory
P.O. Box 808 / L-301
Livermore, California 94551
(max2@llnl.gov)

Abstract

Flow volumes are the volumetric equivalent of stream lines. They provide more information about the vector field being visualized than do stream lines or ribbons. Presented is an efficient method for producing flow volumes, composed of transparently rendered tetrahedra, for use in an interactive system. The problems of rendering, subdivision, sorting,compositing artifacts, and user interaction are dealt with. Efficiency comes from rendering only the volume of the smoke, using hardware texturing and compositing.

Introduction

Understanding 3D vector fields is a current challenge for scientific visualization. When the vector field is, or can be interpreted as, a flow velocity, objects can be traced in the flow. For example, particles can be released and advected by the flow to produce animated motion. Stream lines can also be generated as the tracks left behind by the moving particles. Once the position of a moving particle has been computed for each time step, it is trivial to connect successive points by line segments.

Similarly, if two adjacent streamlines are known, they can be connected by a sequence of triangles to give a ribbon, or stream surface[2][9]. If the two stream lines diverge from each other, Hultquest[3] gives a method of increasing the number of triangles across the width of the ribbon, in order to maintain a smooth-appearing surface that closely approximates the smooth stream surface. Again, the triangles are rendered efficiently by the graphics hardware. Each time the dimensionality increases, from points to lines to areas, there is an increase in the interpretability of the visual representation, without a proportional increase in computational cost to solve the differential equation for the flow. In this paper, we take this progression one step further, to flow volumes.

In physical experiments, smoke is often released into a gas flow, or dye into a liquid flow, as an aid to visualization. The flow past a smoke or dye generator advects the tracer substance into a flow volume, which can be rendered as a semi-transparent volume density. We describe below how this flow volume can be rendered efficiently using graphics hardware. The user can interactively move and size the generating polygon, which is automatically oriented normal to the flow field. As the medium moves through the initial polygon, it becomes colored by the tracer. The result is an image or interactive animation simulating the results of the familiar physical experiments.

Volume Rendering

We divide the flow volume into a collection of tetrahedra, which are rendered by the method of Shirley and Tuchman[10]. This method divides the projection of a tetrahedron into up to four triangles. Figure 1 shows the two non-degenerate cases, which require three and four triangles respectively.

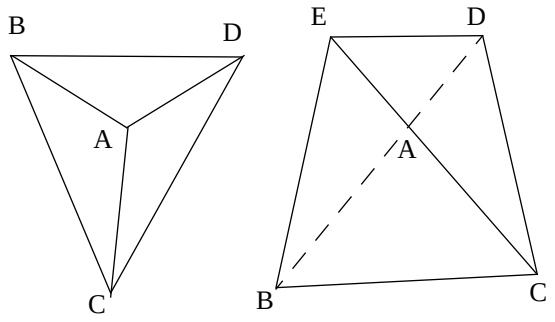

Figure 1

The vertex marked A in each projection corresponds to a viewing ray segment through the tetrahedron, whose length l can be computed from the geometry. The smoke's color and opacity along this ray segment can then be computed by the density emitter model of Sabella[8]. This model assumes the smoke particles absorb a "differential opacity" fraction τ per unit length of the

light traversing the ray, and emit or reflect extra light c per unit length. For colored images, c will be wavelength dependent, with red, green, and blue components.

One can show by integration[4][8] that the opacity α at A is $1\text{-}e^{-\tau l}$, and the color added to the viewing ray is $c\alpha/\tau$. The method of Shirley and Tuchman[10] is to evaluate the color and opacity once at the "thick" vertex A. The color and opacity are zero at the other "thin" vertices on the profile. Bilinear interpolation (linear on triangles) in the hardware rendering pipeline is used to interpolate the color and opacity across the triangle, and composite each triangle over the background. The linear interpolation of color and opacity causes artifacts, which can reveal the separation of the flow volume into tetrahedral cells. Max *et al.*[4] used a software renderer, and computed the necessary exponential at every pixel. They were thus able to deal easily with arbitrary convex cells. Here, we use Shirley and Tuchman's triangle method on tetrahedra. However it is possible to use the texture mapping hardware available on some workstations to get an effective exponential per pixel. The quantity τl is used as a texture coordinate at each vertex, and the value $1\text{-}e^{-\tau l}$ is put in the texture table. The shading system on the Silicon Graphics VGX workstation can then use this as the value in compositing each pixel, at the high throughput rate of its parallel pipeline. This technique reduces polygonal artifacts with only a slight degradation of performance (see figure 7). Since it renders the tetrahedra more accurately fewer are needed and hence opacity round-off will be reduced when the relative thickness increases.

As noted by Wilhelms[12] the eight bits allowed for representing opacity can lead to problems when compositing many thin volumes consecutively. Color shifting became apparent when the numerical error was greater in one color channel than another. The effects were minor and will mostly disappear if a machine having twelve bits per channel is used.

Volume rendering is often slow because of the huge number of cells in a typical volume. However, in rendering a flow volume of smoke, only the cells in the small flow volume need be rendered. Everything else is completely transparent, and may be skipped. This makes interactive use possible.

Sorting

The compositing scheme of [4] and [8] require that the volume cells be composited in back to front order. In general, sorting for the back to front order is a difficult problem. There are easy sorts for special cases such as rectilinear grids, but a flow volume is not rectilinear. Max *et al.* [4] propose using a general topological sort of a directed graph, but this method only works if a convex data volume is completely filled with convex cells. Williams[13] proposes a generalization to non-convex data volumes, but it is not guaranteed to be correct in all cases.Both these methods can return with failure if a depth order cycle exists.

In the current interactive system, we avoid sorting by assuming the color of the smoke is uniform, a reasonable assumption for the visual effect we desire. We now explain why we can do this. Consider a pixel, with initial background intensity F_0 which is covered in back to front order by cell projections of opacity $\alpha_i=(1-e^{-\tau l})$ and color $C\alpha_i = c\,\alpha_i/\tau$. Although the hardware is based on the opacity α_i we will use the transparency $t_i = 1\text{-}\alpha_i=e^{-\tau l}$ for ease in the derivation. The compositing step to update the frame buffer value F is then

$$F_i = t_i F_{i-1} + (1\text{-}t_i)C \qquad (1)$$

We will prove by induction that

$$F_i = \left(\prod_{j=1}^{i} t_j\right)F_0 + \left(1 - \prod_{j=1}^{i} t_j\right)C \qquad (2)$$

The initial step, for i=1, follows from the first compositing step, in formula(1). The induction step then assumes (2) is true for i-1, and derives it for i, using formula (1):

$$F_i = t_i F_{i-1} + (1 - t_i)\,C$$

$$= t_i\left[\left(\prod_{j=1}^{i} t_j\right)F_0 + \left(1 - \prod_{j=1}^{i} t_j\right)C\right] + (1 - t_i)\,C$$

$$= t_i\left(\prod_{j=1}^{i-1} t_j\right)F_0 + \left(t_i - t_i\prod_{j=1}^{i-1} t_j + 1 - t_i\right)C$$

$$= \left(\prod_{j=1}^{i} t_j\right)F_0 + \left(1 - \prod_{j=1}^{i} t_j\right)C$$

The product $\prod_{j=1}^{i} t_j$ is independent of the order of thfactors t_i since multiplication is commutative, and we can thus composite the cells in any order without sorting. Nielson[7] has made a similar observation. This order independence also means that the depth order of the smoke trails in the fluid volume is ambiguous in a still frame. However, with the ability to rotate the scene in real time, the full 3D configuration is revealed.

In rendering an image, we first scan convert the opaque polygons in the environment into the z buffer. Then, when scan converting the triangles from the projections of the tetrahedra, we use the hardware feature which compares the triangle z with the z buffer to determine whether to composite a pixel, but does not update the z buffer. Thus unsorted smoke can still be hidden by opaque objects.

Adaptive Subdivision into Tetrahedra

Some volume cells may have non-planar faces. These faces may become self-intersecting polygons when projected onto the picture plane and hence cause problems in the volume compositing scheme. While such problem cells may be rare in projecting a fixed curvilinear grid, they will be more common in flow volumes, since small scale variation in the velocity field can easily distort the faces. Therefore, we have chosen to decompose the volume into tetrahedra. A method for doing this consistently is the topic of this section.

Let S_0 be the initial polygon generating the flow volume, and let S_n be the surface into which S_0 is carried by the flow, after n time steps. We maintain an approximation of S_n into triangles, which are subdivided adaptively, if they become too large or too curved. To construct the layer of volume cells between S_n and S_{n+1}, we use a collection of prisms, with the triangles on S_n as a base. Each prism is then subdivided into three tetrahedra.

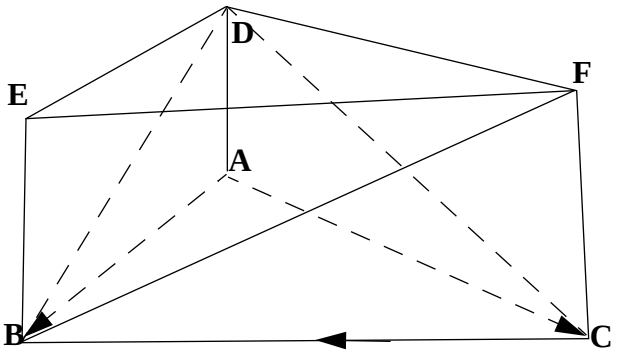

Figure 2

The subdivision of a prism is specified by choosing one of the two possible diagonals on each of its three quadrilateral faces. For example, the choice of three diagonals CD, BD, and BF for the prism ABCDEF shown in Figure 2 implies that it is subdivided into the three tetrahedra ABCD, BCDF, and BDEF. In order to specify the diagonal choices in a way which is consistent across the common quadrilateral faces between adjacent prisms, we use a direction on each edge of the subdivision of S_n, indicated by an arrow on Figure 2. For example, the edge AB on S_n is listed with A as its first vertex, so the point D on S_{n+1} corresponding to A on S_n is the first vertex of the diagonal, and B is the second vertex. Of the eight possible choices of direction for the three sides of triangle ABC, six of them, which have a vertex like B with two arrows pointing towards it, define good subdivisions of the prism. The other two, with all the arrows going around in a clockwise or couterclockwise cycle, do not. Therefore, we must consistently orient the edges of the triangulation of S_n so that no triangle is bounded by a cycle of directed edges. Since every triangulation of a planar region can be built up by adding one-by-one triangles with at least one free side, one can always build up consistent edge orientations by choosing the orientation of the free edge of each triangle appropriately. Instead of using this method, we maintain the edge orientation incrementally. We start with a trivial orientation of the sides for a triangulation of the initial polygon S_0, and then maintain consistency as we subdivide adaptively.

The subdivision of S_n is initially copied from the subdivision of S_{n-1}. If an edge is too long, or too poor an approximation to the correct curved edge, we subdivide it into two edges, with orientations consistent with the parent edge. Once all edges have been subdivided, we loop over all triangles, subdividing them consistently. Figure 3, with several representative cases, shows that the directions of the new edges in the subdivision can be chosen to avoid cycles.

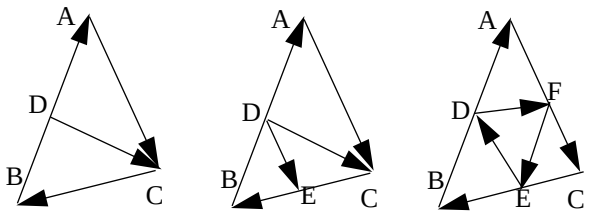

Figure 3

In the center case, when exactly two edges have been subdivided, one of two possible diagonals to a quadrilateral must be chosen, and the shorter one is used.

The curvature subdivision is based on advecting edge midpoints. Suppose, as in Figure 3, D is the midpoint of edge AB. After n time steps, A, B, and D have moved to points A_n, B_n and D_n. The edge A_nB_n is subdivided if the angle between the vectors D_n-A_n and B_n-D_n becomes too large. On the surface S_n, we must still use the actual midpoint of the edge A_nB_n, in order to assure consistency with the flow volume between S_{n-1} and S_n, But on S_{n+1}, we can use the advected midpoint D_{n+1}.

The actual midpoints of the triangle edges on the subdivision of the initial flat polygon are used on surface S_0. Every time a new subdivision edge is created on S_n, we must estimate its midpoint. For edges like AD in Figure 3, which arise from subdividing a longer edge AB, we quadratically interpolate the midpoint using a parabola through A_n, D_n, and B_n. For triangle-crossing edges like DC, linear interpolation is used to find the actual midpoint of D_nC_n.

Figure 4 shows a wire-frame image of curvature-based adaptive tetrahedral subdivision, applied to a divergent flow. The subdivision of the surfaces S_n remains good as these surfaces grow and bend, so the flow volume stays accurate. This volume subdivision also effectively subdivides the stream surface ribbons formed from the edges of the generating polygon. Subdivision improves the accura-

cy of the flow volume by increasing the number of tetrahedra, but makes interaction slow while turned on.

The User Interface

Having a powerful visualization tool is of little value without a scheme for smoothly interacting with it. We present here some ideas which give the user maximum control over the smoke trail. Flow volume interaction is done by manipulating a 3D cursor, and an editing window containing various control widgets for color, transparency, time step size and count, and other characteristics of the flow volume.

The cursor consists of a jack manipulator customized from SGI's Inventor package. It is attached to a polygon which is always perpendicular to the vector field. There are six scale knobs, 2 for each major axis through the polygon, that when selected, will scale the size of the cursor. The *n* sided polygon, which is used as the initial condition for smoke advection, is scaled along with the cursor. When the user clicks on the cursor (but not on a scale knob) three orthogonal translation axes appear. Using the mouse to move the cursor in either direction along one of these axis allows for easy translation in 3D. The Shift and Alt keys may be used to constrain cursor motion to the nearest axis or plane, respectively. The cursor may move anywhere within the domain of the vector field. If a user tries to move beyond those limits the cursor is constrained to the border. For faster interaction the user may choose to have no smoke drawn while the cursor is moving. Even if no smoke is advected, the central polygon is constantly rotating so that its normal remains oriented perpendicular to the direction of the flow. When the mouse button is released the translation axes are removed and a new smoke trail is drawn. Computing and rendering the flow volume is fast enough for smoke to be drawn continuously while the cursor is moving or the scene is rotating.

The smoke puff option simulates an intermittent smoke generator, by making the opacity depend on a time varying functino of the step index *n*. The puffs blow along in real time, as long as no other parameters change, bunching up where the current is slower. Interaction slows if the number of steps is too high.

Active along with the cursor is an editing window containing sliders for controlling the length of the time step, the number of time steps used, and the smoke's transparency. A color wheel is used to control the smoke color. Toggle buttons for specifying transparency texture mapping, compressible/incompressible flows, puff or growing smoke animation, and wire-frame drawing are available. For compressible flows, we make the differential opacity τ inversely proportional to the volume of the tetrahedron. The Inventor Scene Viewer already provides menu options for various drawing styles such as dithered or blended transparency, picking styles, and facilities for customized lighting. All operations of translation, rotation and zooming are handled by a variety of convenient viewers provided by the Scene Viewer.

The IRIS Inventor Toolkit, a Silicon Graphics C++ environment, was of immense help for speedy prototyping.

Results

Flow volumes offer a more powerful visualization tool than streamlines or ribbons. Its effectiveness is shown by the way it can reveal phenomena the other two methods cannot. If compressible flow is specified, changes in pressure can show up as varied opacity. Vector magnitude changes speed or slow puffs of smoke when that animation option in selected. Diverging or converging flows, turbulence, spiraling, shearing, and splitting as flow moves over solid obstacles, are difficult to represent with a single streamline, but easy for volumes. Scaling of the initial polygon allows you to get all of these benefits whether your focus is global or very local. All these features can be seen at the same interactive rates as lines or ribbons because of the use of hardware and a simple rendering algorithm.

For a typical flow volume consisting of 1000 tetrahedra, rendering takes about .4 seconds, transforming and rendering .5 seconds, and recomputing then rendering1 second. Times were approximated on a Silicon Graphics Indigo 2 workstation with extreme graphics.

In Figure 5 we see an example of a hurricane visualized using the system. In it we can see that the wind is moving slowly until the perimeter of the hurricane is reached. Then the velocity increases and the tetrahedra become long and thin. By animating puffs of smoke, varying velocities are more clearly visualized. Figure 6 shows the result of making a flow compressible. In this figure opacity corresponds to density of smoke particles. The effect of adding texture mapping to reduce artifacts can be seen by comparing figure 7*a* which is drawn without texturing to figure 7*b* which uses texturing to produce the correct exponential at each pixel.

Vector fields from electro-magnetics, waveguide simulation, air flow through aerogel material, and a simulated tornado have all been successfully visualized. Each have unique properties which can be explored using this tool.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-ENG-48, with specific support from an internal "LDRD" research grant. We wish to thank Leonore Max, and the IEEE computer Society reviewers for comments which improved the paper.

References

[1] Crawfis, R. and Max, N. (1992) “Direct Volume Visualization of Three-Dimensional Vector Fields”, Proceedings of the 1992 Workshop on Volume Visualization, Kaufman and Lorensen (eds), ACM SIGGRAPH, NY pp 55 - 60

[2] Hellman, J. and Hesselink, L.(1991) “Visualizing Vector Field Topology in Fluid Flows.” IEEE CG&A, Vol 11, No. 3, May, pp36-46

[3] Hultquist, J., (1992) “Constructing Stream Surfaces in Steady 3D Vector Fields”, Proceedings of Visualization ‘92, IEEE Computer Society Press, Los Alamitos, CA pp 171-178

[4] Max N., Hanrahan P., and Crawfis R. (1990) “Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions”, Computer Graphics Vol. 24 No. 5, pp 27-33

[5] Max, N., and Crawfis, R., (1992) “Visualizing Wind Velocities by Advecting Cloud Textures” Proceedings of Visualization ’92, IEEE Computer Society Press, Los Alamitos CA, pp 179 - 184

[6] Max, N., (1993) “Sorting for Polyhedron Compositing”, in “Focus on Scientific Visualization” Hagen H., Müller H. and Nielson G. (eds) Springer Verlag, Berlin, pp 259-268

[7] Nielson, G. N., (1993) “The Volume Rendering Equations”, TR-93-013, Dept. of Computer Science, Arizona State University, Tempe

[8] Sabella, P., (1988) “A Rendering Algorithm for Visualizing 3D Scalar Fields”, Computer Graphics Vol. 22 No. 4 (Siggraph ’88 Proceedings) pp 51 - 55

[9] Shirley, P. and Neeman, H.(1989) “Volume Visualization at the Center for Supercomputing Research and Development”, Proceedings of the Chapel Hill Workshop on Volume Visualization, Department of Computer Science, University of North Carolina, Chapel Hill, NC, pp 17 - 20

[10] Shirley, P. and Tuchman, A.(1990) “A Polygonal Approach to Direct Volume Rendering”, Computer Graphics, Vol. 24 No.5, pp 63-70

[11] Westover, L., (1989) “Interactive Volume Rendering”, Proceedings of the Chapel Hill Workshop on Volume Visualization, Department of Computer Science, University of North Carolina, Chapel Hill, NC, pp 9 - 16

[12] Wilhelms, J. and Van Gelder, A.(1991) “A Coherent Projection Approach for Direct Volume Rendering”, Computer Graphics, Vol. 25, No. 4 (Siggraph ‘91 Proceedings) pp. 275-284.

[13] Williams, P. (1992) “Visibility Ordering Meshed Polyhedra” ACM Transactions on Graphics, Vol. 11, No. 2, pp 103-126

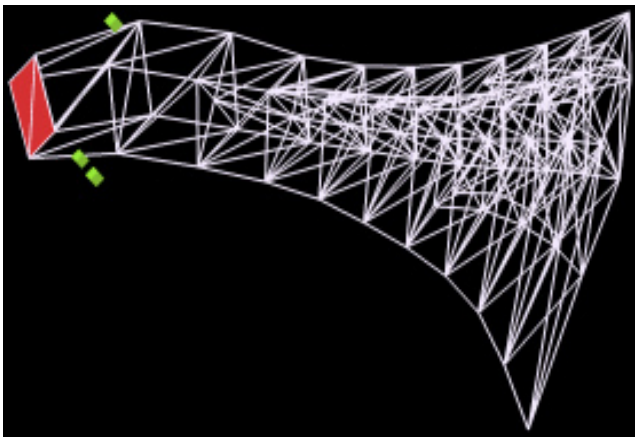

Figure 4. Curvature based adaptive subdivision applied to a divergent flow.

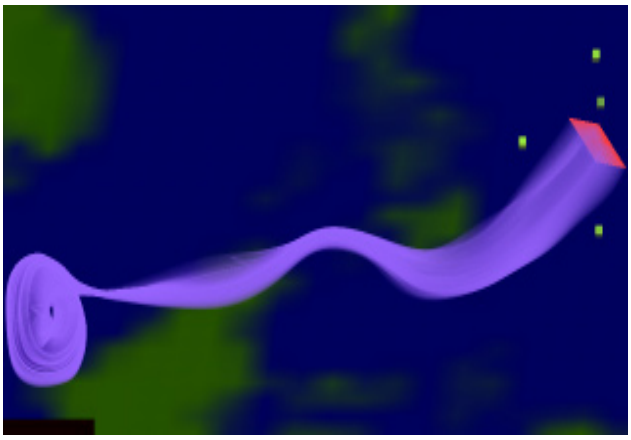

Figure 5. Visualization of a hurricane in global wind data.

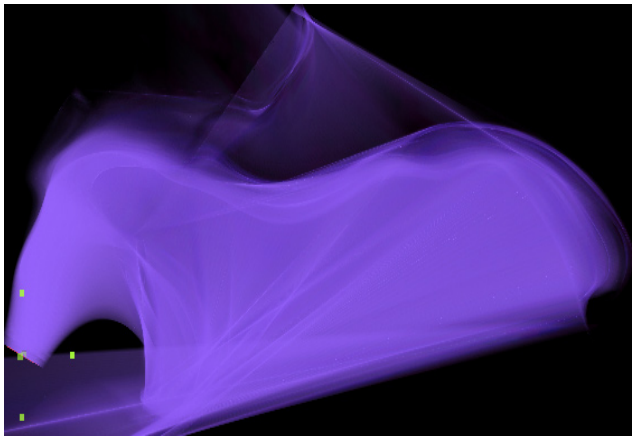

Figure 6a. A flow volume in an incompressible medium.

Figure 6b. The same flow volume, but now we are assuming the medium is compressible.

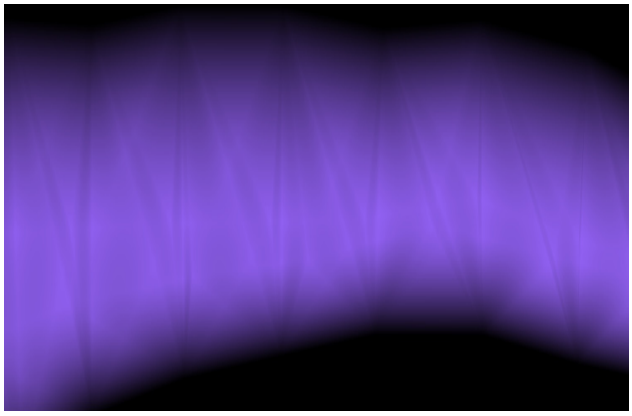

Figure 7a. Smoke without hardware texture mapping.

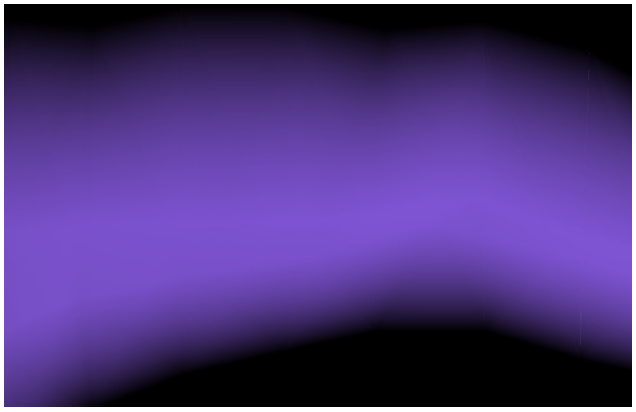

Figure 7b. Smoke with hardware texture mapping.