FLOW VISUALIZATION TECHNIQUES FOR CFD USING VOLUME RENDERING

Roger A. Crawfis, The Ohio State University

Han-Wei Shen, The Ohio State University

Nelson Max, University of California, Davis

Keywords: *Scientific Visualization, volume rendering, flow visualization*

ABSTRACT

As simulations have migrated towards three-dimensions, new tools for examining the resulting complex datasets have been required. Much progress has been achieved in the area of scientific visualization towards this goal. However, most of the research has focused on the representation and understanding of a single scalar field. Some nice results have been achieved for vector or flow fields. This paper reviews several of these techniques, organizes them by their approach and complexity and presents some observations on their benefits and limitations. Several example images are used to highlight the character of these techniques.

1 INTRODUCTION

Computational science has advanced rapidly, generating improvements in algorithms and computational efficiency. This had led to the ability to simulate phenomenon at much finer scales, resulting in meshes with hundreds of thousands to millions of sample points. This data is comprised of scalar variables (such as temperature, pressure, etc.), as well as vector (velocity, magnetic flux, vorticity, etc.) and even tensor (stress, etc.) quantities. This data is calculated in three spatial dimensions, plus time dynamics. Research in scientific visualization has seen rapid progress in presenting both scalar three-dimensional fields, as well as vector fields in three-dimensions. This paper presents an overview of these advances for vector field visualization, offering a taxonomy for the myriad of algorithms.

For many years, the staple for visualizing vector data consisted of three possible choices. First, the magnitude of the vector data could be extracted and then scalar visualization algorithms applied to this resulting field. This provides useful information on where the velocity is greatest, but provides no visual cues on the direction or turbulent nature of the flow. A second choice was to place a little vector icon at each grid point. For two dimensions, with enough study a reasonable inference into the flows direction and magnitude was possible. This algorithm suffered severely from excessive clutter if the magnitude of the flow varied greatly. For many grid types, especially uniform, it tended to highlight the grid layout of the sample points rather than the actual flow. These issues were only compounded in three dimensions. This type of plot is commonly called a vector plot, arrow plot or hedgehog plot.

The third classical vector visualization technique simulated the release of a mass-less particle in the vector field. This particle was advected and a history of its path was remembered and traced with a curve. For 2D flows, a streamline separates the flow, such that any particle released to the left of the streamline will remain to the left. For steady state flows these streamlines provided a

good indication of the downstream movement, but only for that particular starting point for the particle. By seeding the computational-space, either randomly or using a rack (or line of starting points), the initial paths can be situated more comprehensively. However, one common drawback was that the particles would tend to cluster together in regions of compression. Once compressed sufficiently, these streamlines stayed bunched together. Thus, large areas of space were left with little or no indication of the flow. Another drawback of this technique was its sensitivity to the numerical precision in which the advection was performed. A simple Euler integration of the resulting ODE can provide misleading paths. Adaptive schemes, such as an adaptive Fourth-Order Runga-Kutta method are thus required. For known analytical functions, these algorithms are robust enough to provide stable results. A common error on many visualization experts was to provide a very robust integrator, only to have it circumvented with an inferior interpolation kernel on the discrete grid data.

Time varying simulations or flow fields present additional challenges to the user and the visualization designer. Streamlines can be applied to a single time step, but may present misleading information. Instead pathlines or streaklines must be used. A streakline mimics the streak left by a virtual dye emitted at a single point in the flow. Pathlines keep a history of the path a particle makes. This particle has a specific starting point and starting time. Both pathlines and streaklines can fold back upon themselves. Streamlines cannot do this, and herein lies their fundamental power, to separate the flow into distinct parts. Pathlines and streaklines therefore require a much greater amount of studying to understand aspects of the flow.

For hedgehogs, no special care is usually taken for time-varying data. Instead, the current plot is replaced with a new one generated using the new vector field. This is akin to have several thousand weather vanes that instantaneously change direction. It is thus extremely difficult, even in 2D, to remember the previous time's direction for all but a hand full of arrows. If the resulting display is cluttered, the effect is as if a random set of cluttered lines is drawn each time. The technique is however useful in separating regions where the flow does not change, from those where it changes drastically.

Perhaps the best classical technique for time-varying flows, is a continuous scalar representation of the vector magnitude. A pseudo-colored plot or volume rendering of the magnitude can show major fronts moving. This technique smoothes out subtle directional changes. If the flow is simply spinning with no lateral movement however, there will be no time-dynamics to represent this.

New research into both steady-state vector field visualization, as well as time-varying vector field visualization was clearly needed. Indeed, a wide variety of algorithms have been presented at the IEEE Visualization conference series and other excellent publications. This paper will attempt to highlight some of these algorithms for this audience. In particular, we will focus on recent algorithms utilizing volume rendering to represent three-dimensional vector fields globally. A categorization or taxonomy of the algorithms will be presented, along with many sample images. Interspersed will be some thoughts on the pros and cons of various algorithms and the need for still further research in this field.

2 A SURVEY AND TAXONOMY

2.1 Advection Techniques

The first class of algorithms all use an advection algorithm similar to streamlines. For the most part, these algorithms are independent of the underlying grid topology of the data. High-quality integration schemes have been developed for all of the major grid types (e.g., regular, curvilinear, and irregular). There are still issues with the precision employed by these algorithms.

For a very large number of advection steps, due to either lengthy integrations or a large number of different particles, these algorithms can become quite slow. However, in practice the number of particles is usually very small and interactive results are obtainable.

2.1.1 Stream Surfaces

As mentioned in the introduction, one of the most valuable aspects of streamlines was its ability to separate steady-state flows. This implies that two streamlines in 2D will not cross each other. In three-dimensions, we are still guaranteed that two streamlines will not cross, but the added dimension allows for streamlines to curl around each other, and it is again very difficult to ascertain the overall flow in areas of substantial directional change. Hultquist [1] solved this problem, by stitching together several adjacent streamlines to build a *stream surface*. This surface can be thought of as the infinite set of streamlines emanating from some starting line segment or curve. The flow is assumed to be continuous, in that a streamline within the neighborhood of an adjacent streamline, can not suddenly diverge in an infinitesimally small time step. For steady-state flows, these surfaces can locally separate the three-dimensional flow, such that streamlines to one side of the surface are restricted to lie within that portion of the volume. If the starting curve for the stream surface is a closed curve, then the separation is into two disjoint parts, the flow bounded by the surface and the flow outside of the surface. This algorithm suffers from some of the same drawbacks as 2D streamlines. Namely, its utility is lost for unsteady flows. A further difficulty is that the flow direction tangential to the surface is lost. For a streamline, we have a single direction, but by merging many streamlines onto a continuous surface, one cannot tell whether the flow went straight down the surface or weaved back and forth. The algorithm is expensive in that it needs to handle severe distortion of the surface mesh as the flow diverges around obstacles. To accommodate this, Hultquist added an algorithm that would adaptively refine the mesh and even allow for tears in the surface.

Van Wijk [2] proposed a simpler scheme to constructing stream surfaces for closed environments. They precompute a discretized volume on a regular grid. For interior point on the grid, they compute a backward streamline until the point intersects the boundary. The position on the boundary is stored for each point. By placing scalar function values on the boundary, this results in a scalar function defined over the regular grid. Van Wijk then takes an iso-contour surface of this function to construct the stream surface. The flow must enter or exit the volume boundaries in order for this technique to work. Artificial boundaries can be placed to rectify this.

One solution to the lack of direction on the surface is to superimpose a texture or pattern on the surface. Interrante and others [3] proposed the use of the Line Integral Convolution (LIC) [4] algorithm to solve this problem.

2.1.2 Flow Volumes

Stream surfaces were a nice extension of streamlines. A single surface can be rendered efficiently using standard graphics hardware. Multiple surfaces would most likely occlude each other. A single surface can have a substantial amount of occlusion, requiring a careful examination of the surface from many viewpoints. If we let the surfaces be semi-transparent, we are restricted to a slow ray-tracing program to handle the correct sorting of the geometry into visibility order. Max, et al. [5] thus introduced the notion of *flow volumes*. Rather than start with a single curve to emanate the streamlines from, they use a small surface patch comprised of quadrilaterals. These quadrilaterals are split along the diagonal to form a triangular surface mesh as the starting surface. As streamlines are advected from these grid points, they bound prism solids. Figure 1. is an example of a constructed flow volume with a

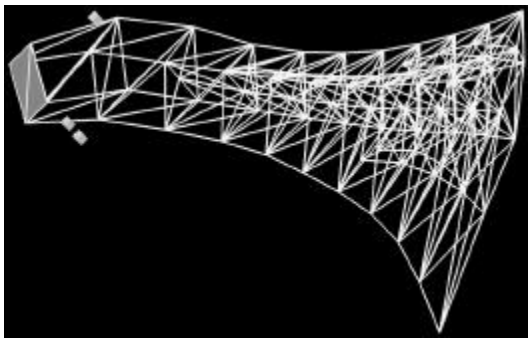

Figure 1. Prism mesh formed from flow volumes

single quadrilateral as the initial surface. The flow volume algorithm also had to deal with adaptively starting new grid points or streamlines in areas where the flow diverged.

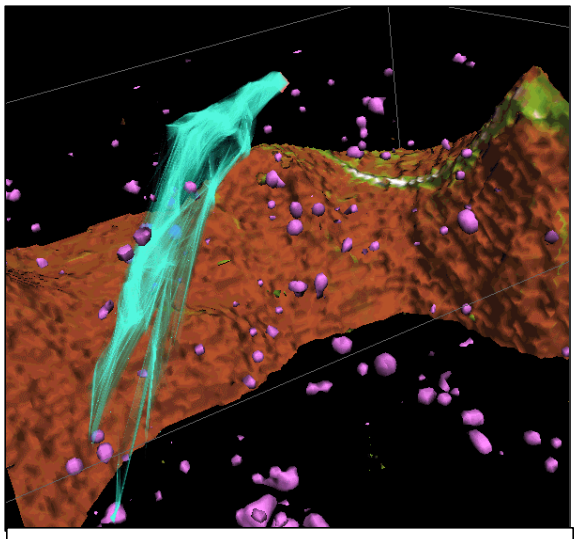

Figure 2. A volume rendered flow volume simulating the release of dye in an underground water simulation.

The wireframe mesh does not aid in representing the flow. The authors developed algorithms to mimic the effect generated from a smoke generator placed in a wind field (or a dye injector placed in a liquid flow). They applied volume rendering algorithms to provide the appearance of low-albedo colored particles filling the volume. This required splitting each individual prism into tetrahedra and applying a special scan-conversion routine [5, 6] to render each tetrahedron. Care again had to be taken to ensure a consistent subdivision of the mesh into tetrahedra [7]. Figure 2 illustrates the technique applied to a simulation of the underground water transport beneath a one-square mile section of the Lawrence Livermore National Laboratory. The brown clay layer illustrates a fault (white stripe), while the purple blobs are iso-contour surfaces of the porosity. The blue-green volume shows the vector field in the water transport model. Here the water eventually spreads out into many small channels. The height or depth of the model has been exaggerated to aid in the understanding and visualization process.

The smoke injector can be interactively moved throughout the volume. The initial surface is rotated automatically such that it is perpendicular to the flow at the center point. This data was particularly difficult in that the heterogeneous soil model required the flow to split in many places. The resulting flow volume therefore had to continually adapt to these divergences. During the rendering, the smoke can be treated as compressible or incompressible, meaning that we can model a fixed density of particles within each tetrahedron, or we can model a fixed number of particles within each tetrahedron. The former option, allows for a constant opacity per unit volume, thus as the tetrahedron's size increases (due to a diverging flow), the opacity increases and the appearance of the smoke gets thicker. The latter approach decreases the per unit opacity of the tetrahedron as the flow diverges. This opacity can also be made to vary as a function of the advection time steps. Using this, the authors were able to simulate puffing smoke interactively in their system. This feature also aided in giving an indication of the relative velocity magnitude of the flow in the flow volume.

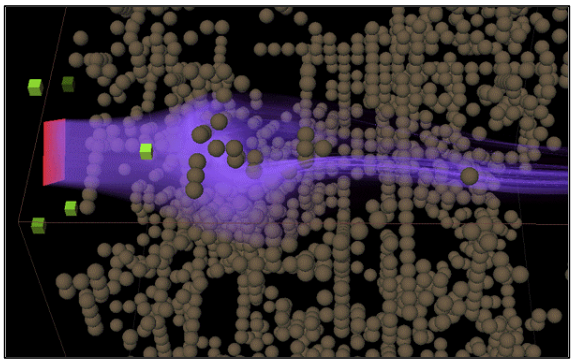

Figure 3. A large flow volume interacting with a simulation of the flow through an aerogel material.

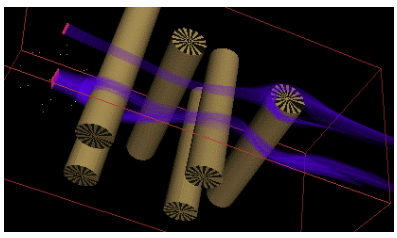

Figure 4. Two flow volumes applied to a HEPA filter simulation.

Since this is a volumetric entity, the size of the virtual smoke generator can be arbitrarily large or small. Figure 3 illustrates a simulation of the flow through a small aerogel substrate. The initial area is quite large, but the flow is constricted as it searches for a good passageway through this material. The small spheres represent the placement of molecules in the simulation. A final example shows two such flow volumes applied to the simulated flow through a HEPA filter. Figure 4 shops how the flow volume can split around obstacles in the flow. The tan cylinders represent the micro-fibers of the HEPA filter.

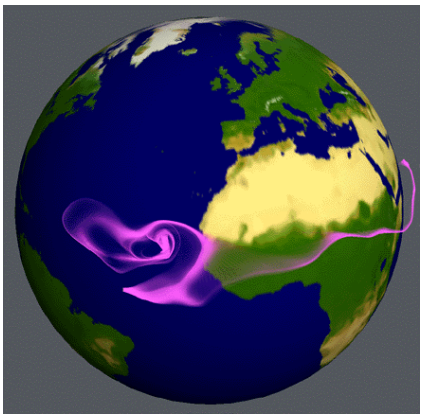

Figure 5. Time varying wind fields used in the construction of an unsteady flow volume.

For time-varying data sets, this technique suffers from the same problems as streamlines and stream surfaces. In a separate paper, Becker, et al. [8] extended this concept for unsteady flows. Their results were inconclusive however. As the flow volume changed over time, the volume would self intersect leading to incorrect rendering. Figure 5. represents a successful case. Here, a global climate simulation is being studied. The wind velocities in the lower hemisphere can clearly be seen as they diverge over Africa and a turbulent pattern is visible over the Pacific ocean.

2.1.3 Stream-balls

An interesting alternative to stream surfaces and flow volumes are the Stream-Balls proposed by Brill [9]. Their concept is similar to van Wijk's implicit stream surfaces, however, they build a scalar function as a potential field from the advected particles or streamlines. Using blobby's [10] or meta-balls [11], they devise a potential field by placing a Guassian function at each advected position for a particle. By summing over all of the time steps for a particle, as well as summing over many different particles, an analytical function is determined. Sampling this function over a regular grid and taking an iso-contour of the scalar function produces surfaces that mimic a stream surface. In areas of high divergence, this surface gracefully splits apart. Hence, it looses the nice property of separation inherent in stream surfaces. The surface will also break apart in areas of high velocity. This helps to indicate the magnitude of the velocity, which is lost in the stream surfaces.

2.1.4 Better Streamlines

Several research efforts have attempted to make classical streamlines more useful for three dimensional flows. Zöckler, Stalling, and Hege applied the curve illumination model of Banks [12] to produce Illuminated Streamlines [13, 14]. The addition of diffuse and specular highlights to these curves greatly aids in the depth perception and illusion of three-dimensionality. To alleviate the problem with streamlines bunching together, leaving large voids of unexplored space, several authors have examined techniques to place streamlines, or streamline fragments, such that in 2D a rather uniform density of curves to empty space is achieved. Turk and Banks [15] developed an algorithm to compute and update the density of streamlines within arbitrary regions of the image plane. New streamlines were started in areas of low density. These were computed for the forward flow, as well as the backward flow until they reached an area of high density. These algorithms border into our next classification, that of producing global textures. Little thought has been given into extending these techniques to three-dimensions. We will not go into many of these, as our focus is on techniques using volume rendering for three-dimensional flows.

2.2 Texture Generation Techniques

This class of techniques attempts to provide a pattern across the entire domain of the problem that provides good visual cues about the direction of the flow. These can be viewed as extensions of arrow plots. Many of the techniques can be combined with color to represent the magnitude of the flow. Additional techniques for showing magnitude have also been researched and these will be

highlighted below as well. A major problem with most of these techniques their lack of usefulness for representing 3D flows in their entirety. Thus, many of the techniques are applied in 2D, on a 2D surface within the 3D flow, or on slice planes through the 3D flow.

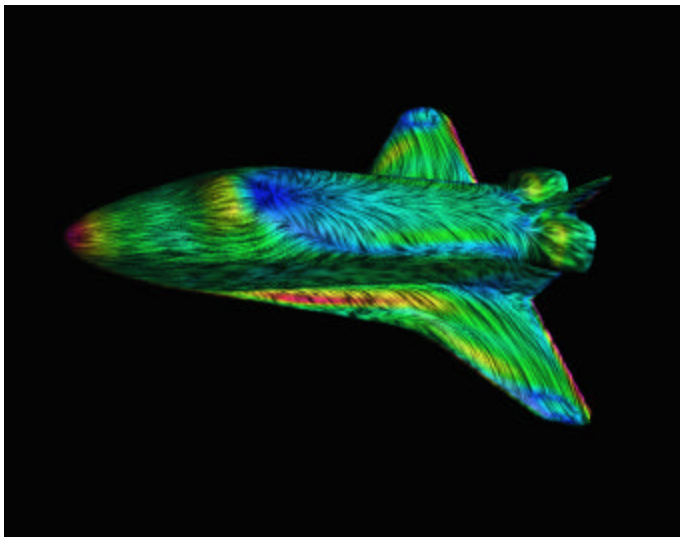

Figure 6. A 2D Line Integral Convolution texture applied to the surface of the space shuttle. This surface is actually a slice plane in the curvilinear grid.

2.2.1 Spot Noise

Jarke van Wijk extended some earlier research on flow visualization [16] to arrive at the concept of Spot Noise [17]. Perhaps the best way to think about spot noise for flow field representation is to imagine little drops of ink falling on a piece of paper. The location of where the ink hits has a uniform distribution, but the paper is tilted, such that the ink runs in the direction of the vector field associated with that point. After many thousands of tiny drops, a pattern is developed that represents the flow field. Van Wijk presented a spectral approach to achieving this as well. The algorithm produces nice results, although for high curvature or vorticity, the spots tend to blur out the flow structure.

2.2.2 Line Integral Convolution

Line Integral Convolution, or LIC [18], has been perhaps the most visible of the recent flow visualization algorithms. The algorithm takes a scalar field and a vector field in as input, and outputs another scalar field. By providing a white noise image as the scalar field, an output image is generated that correlates this noise function along the direction of the input vector field. It operates on a discrete grid by rasterizing a line segment (or streamline) centered at each point and oriented in the direction of the flow, and then integrating (or summing up) the values in the corresponding noise function for those pixels rasterized. Moving to a neighboring point along this rasterized line produces a set of discrete pixels very similar to the current set. Thus, a similar integral value is obtained. On the contrary, selecting a neighboring pixel not in the rasterized set produces a different set of random values and a different integral value.

While LIC is effective in visualizing 2D vector fields, it is quite computationally expensive. Stalling and Hege proposed an extension to speed up the process [19]. Their work is based on two key observations. First, a streamline starting from any point in the domain actually passes through many pixels. For those pixels, only one rasterized streamline is sufficient to produce the convolution results and thus redundant numerical integrations can be avoided. The second observation is that adjacent pixels along the same streamline use very similar sets of pixel values for the convolution. Therefore, the LIC value computed for one pixel can be reused by its neighbors with small modifications, accelerating the convolutions. By reducing the number of streamlines computed and speeding up the LIC convolution, Stalling and Hege's new method can gain a great saving in computing the LIC.

Some work has been directed towards trying to make LIC useful in three-dimensions. Shen *et al.* [20] proposed to introduce dye advection into LIC computation. In their method the "smearing" nature of LIC is used to simulate the dye advection. The visualization produced by their method can include both regular LIC textures and 3D particle traces. To improve the interactivity, they also

proposed an efficient algorithm to locate the "dyed" regions so that a fast update of the LIC result is possible. This feature allows the user to dynamically inject dye at different locations. Recently, Rezk-Salama proposed a volume rendering algorithm towards trying to make LIC useful in three-dimensions. A volume rendering algorithm that utilizes 3D texture mapping memory is explored to quickly adjust slice planes and opacity settings [21]. This provides some clues, but the user is still left to build a three-dimensional model of the flow.

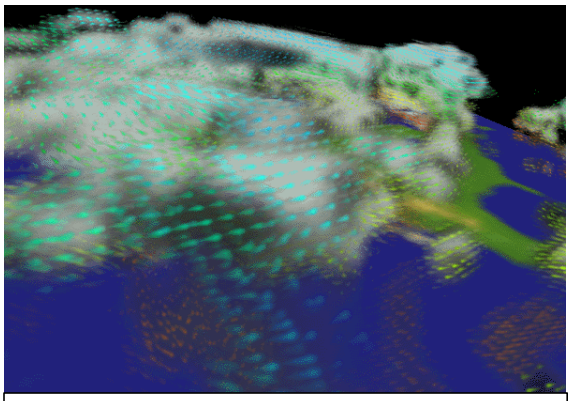

Figure 7. Volume rendering with embedded vector icons. The white clouds represent percent cloudiness and the colored icons represent wind velocity directions in this global climate simulation. The icons are colored according to altitude.

2.2.3 Volume Rendering with embedded Texture

For three-dimensional flows, some research has been directed towards integrating texture or icons into a volume rendering of the flow, or directing volume rendering the flow with an anisotropic texture. Crawfis and Max [22, 23] developed a 3D raster resampling technique where the volume rendering was built up in sheets oriented parallel to the image plane. These sheets were composited [24] in a back-to-front order. The authors modified the volume integral to include the rendering of a tiny cylinder within a small neighborhood. To accomplish this, they took a regular grid sampling on the current sheet. An image reconstruction operator or splat [25] is placed over each grid point. A stochastic heuristic is then used to determine whether this splat should include a cylinder. If so, the cylinder is randomly offset from the center and the volume integral is added for each pixel under the splat and the cylinder. The authors applied this to a time-varying wind field. For each time step, a new rendering was produced and the final frames animated in a movie loop. Due to the stochastic placement of the vector icons, the resulting animation produced the illusion of coherent flow, rather than that of individual icons rotating about a fixed center.

A further refinement of this concept was to embed the vector icons directly into the splat footprint [26]. Here, small billboard images are overlapped and composited together to build up the final image. By placing a small icon within the billboard image, and orienting the image such that it lies both perpendicular to the viewing ray, and parallel to the projected vector direction at the splat's center point, a volume rendered image is produced. A discrete set of such billboard images were designed with the vector icons where reduced in size. This allowed for some indication of the vector direction into the screen. When the flow was parallel to the screen, the largest icon was chosen. For flows parallel to the viewer's line of sight, the icons reduced to a small dot. Figure 7 illustrates this technique applied to the global climate simulation. The white density clouds represent the percent cloudiness in the climate model. The colored vector icons represent the wind velocity direction. For this particular image, the authors used color to represent the altitude of the vector icons. The technique can easily merge the rendering of a scalar field with that of a vector field. The flow is still rather difficult to understand due to the sparse and discrete representation of the vector icons. To alleviate this, the authors devised a set of textures in which the icons moved across the texture, fading in and fading out. By cycling through these textures at rendering time, the illusion of motion is achieved.

The same authors developed several more techniques to represent flow fields using an anisotropic or textured volume rendering. The Line Bundles techniques [27, 28] rely on the graphics hardware to create texture by rendering and compositing many semi-transparent line segments. Using a splatting analogy, each data point is rendered with pre-computed sets of rendered line segments. The colors of these line segments are slightly perturbed to provide a colored pattern. As these line bundles are composited together in a back-to-front order, an anisotropic volume rendering results. Figure 8. illustrates this technique applied to the aerogel simulation. The colors represent the velocity magnitude, with magenta representing high velocity, while yellow and then transparent represent low velocities. This technique allows one to view the velocity magnitude, much like a normal volume rendering. However, the directionality is clearly evident in the volume rendering. Figure 9. shows another example with more opaque line bundles. Here, a test data set is rendered with colors again representing the vector magnitude. Blue represents lower velocities than red. The illumination model for curves of Banks [12] is used in the rendering, providing a better sense of curvature.

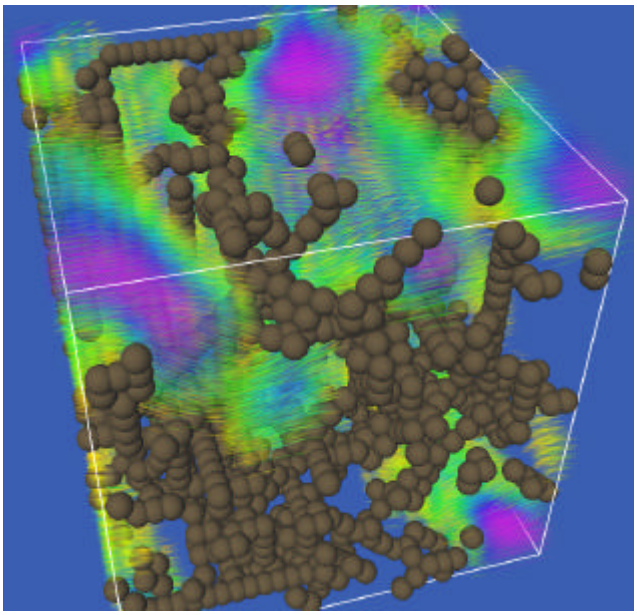

Figure 8. The flow velocity through the simulated aerogel is represented with the *Line Bundles* techniques. The colors represent the velocity magnitude.

2.3 Topology Techniques

Topological techniques can be quite effective in capturing the global features of a flow field. Helman and Hesslink [29] proposed to analyze the vector field topology based on critical point classification, which has been widely used to analyze solution trajectories of ordinary differential equations. Critical points are points that have zero vector magnitude. In two dimensional flow fields, they can be found analytically by solving a simultaneous bilinear equation. For three dimensional field data, numerical method such as Newton-Raphson can be used to locate the critical points. To classify the critical points, the eigenvalues and eigenvectors of their corresponding Jacobian matrices can be computed. Based on the signs of eigenvalues, the critical point can be classified as Repelling Focus, Saddle Point, Repelling Node, Attracting Focus, Center, or Attracting Node. To understand the flow field, a *topology skeleton* can be created by tracing streamlines starting from the neighboring region of the critical point, and following the direction of eigenvectors.

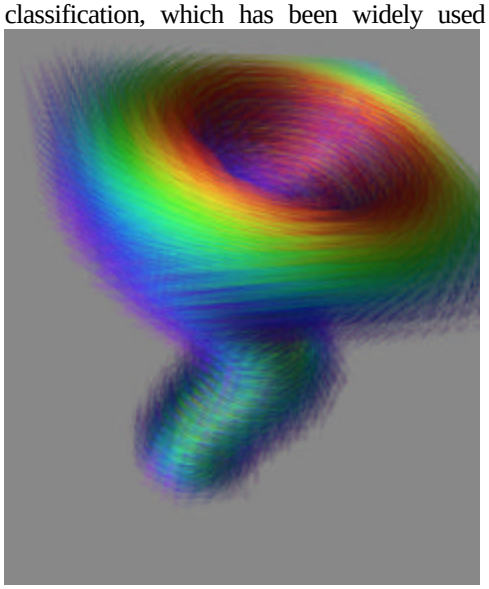

Figure 9. The *Line Bundles* technique using a better illumination model for curves.

Vector topology analysis is an effective tool that can be used to understand the flow field. However, some practical issues can affect the effectiveness of

this approach. First, precise streamlines connecting critical points are difficult to compute using numerical integration method. This is because critical points usually exist in regions with high gradients and discontinuity and most of the numerical methods are not capable of producing accurate results in those regions. The second problem is that flow topology based on the critical point theory does not necessarily exhaust interesting flow phenomena in the field. Sometimes, scientists want to know flow directions in regions surrounding certain objects where no critical points may exist.

3 CONCLUSIONS

This paper has given an overview of many of the techniques available for the visualization of three-dimensional vector fields. A characterization into advection, texture, and topology techniques has been introduced to serve as a discussion for analyzing the task of understanding flow fields. Advection techniques can work quite well for 2D or 3D steady-state flows using streamlines for 2D and flow volumes for 3D. These techniques are particularly beneficial when the task is to understand the downstream behavior of the flow, that is, to understand where some contaminants or pollution will eventually end up. For time-varying flows, this question can still be answered, provided the techniques are stable enough to handle the flow folding back upon itself. In general, the analysis of 3D time-varying flows is very difficult with these techniques. The texture-based techniques provide a comprehensive pattern of the flow direction, so important features are not missed. This works extremely well in 2D and can support time-varying flows with little additional effort. For three-dimensions, the patterns are too dense to provide a view of the entire field. The region of the flow to be represented is usually limited to those areas containing high velocity values. While some nice results have been obtained with the results presented here, much more research is needed before truly useful tools are widely adopted. In particular, even though many of the tools can handle time-varying data, the analysis of time is usually limited to a serial process. This forces the user to remember the previous time-step in the texture models.

4 ACKNOWLEDMENTS

This work was sponsored in part by an NSF CAREER award and conducted in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-ENG-48. The authors wish to acknowledge Bob Corey for the HEPA filter dataset, Steve Ashby for the groundwater flow, Tony Ladd and Elaine Chandler for the aerogel data, Richard Klien for the shockwave dataset and Jerry Potter and Dean Williams for the Global Climate data.

5 REFERENCES

1. Hultquist, J.P.M. *Constructing Stream Surfaces in Steady 3D Vector Fields*. in *Visualization '92*. 1992. Boston, Massachusetts: IEEE Computer Society Press.
2. Wijk, J.J.v. *Implicit Stream Surfaces*. in *Visualization '93*. 1993. Los Alamitos, CA: IEEE Computer Society Press.
3. Interrante, V. and C. Grosch, *Strategies for Effectively Visualizing 3D Flow with Volume LIC*, in *IEEE Visualization '97*, R. Yagel and H. Hagen, Editor. November 1997, IEEE. p. 421-424.
4. Cabral, B. and L.C. Leedom. *Imaging Vector Fields Using Line Integral Convolution*. in *SIGGRAPH 93*. 1993. Anaheim, CA: ACM SIGGRAPH.
5. Max, N., B. Becker, and R. Crawfis. *Flow Volumes for Interactive Vector Field Visualization*. in *Visualization '93*. 1993. Los Alamitos, CA: IEEE Computer Society Press.
6. Shirley, P. and A. Tuchman, *A Polygonal Approximation to Direct Scalar Volume Rendering*. Computer Graphics, 1990. 24(5): p. 63-70.
7. Albertelli, G. and R.A. Crawfis, *Efficient Subdivision of Finite-Element Datasets into Consistent Tetrahedra*, in *IEEE Visualization '97*, R. Yagel and H. Hagen, Editor. November 1997, IEEE. p. 213-220 .

8. Becker, B.G., D.A. Lane, and N.L. Max. *Unsteady Flow Volumes*. in *Proceedings Visualization '95*. 1995. Atlanta, GA: IEEE Computer Society Press.
9. Brill, M., *et al. Streamball Techniques for Flow Visualization*. in *Visualization '94*. 1994. Washington, D.C.: IEEE Computer Society Press.
10. Blinn, J.F., *A Generalization of Algebraic Surface Drawing*. ACM Transactions on Graphics, July 1982. 1 (3): p. 235-256 .
11. Wyvill, G., C. McPheeters, and B. Wyvill, *Soft Objects*, in *Advanced Computer Graphics (Proceedings of Computer Graphics Tokyo '86)*, T.L. Kunii, Editor. 1986, Springer-Verlag. p. 113-128.
12. Banks, D.C., *Illumination in Diverse Codimensions*, in *Proceedings of SIGGRAPH 94*, A. Glassner, Editor. July 1994, ACM SIGGRAPH ACM Press: Orlando, Florida. p. 327-334 .
13. Zöckler, M., D. Stalling, and H. -C. Hege, *Interactive Visualization of 3D-Vector Fields using Illuminated Streamlines* , in *IEEE Visualization '96*, R. Yagel and G.M. Nielson, Editor. October 1996, IEEE. p. 107-114.
14. Stalling, D., M. Zöckler, and H. -C. Hege, *Fast Display of Illuminated Field Lines*. IEEE Transactions on Visualization and Computer Graphics, April - June 1997. 3 (2).
15. Turk, G. and D. Banks, *Image-Guided Streamline Placement*, in *Proceedings of SIGGRAPH 96*, H. Rushmeier, Editor. August 1996, ACM SIGGRAPH Addison Wesley: New Orleans, Louisiana. p. 453-460.
16. Wijk, J.J.v., *A Raster Graphics Approach to Flow Visualization*, in *Eurographics '90*, September 1990, North-Holland. p. 251-259.
17. Wijk, J.J.v., *Spot noise-Texture Synthesis for Data Visualization*, in *Computer Graphics (Proceedings of SIGGRAPH 91)*, T.W. Sederberg, Editor. July 1991: Las Vegas, Nevada. p. 309-318.
18. Cabral, B. and L.C. Leedom, *Imaging Vector Fields Using Line Integral Convolution*, in *Proceedings of SIGGRAPH 93*, J.T. Kajiya, Editor. August 1993: Anaheim, California. p. 263-272.
19. Stalling, D. and H. -C. Hege, *Fast and Resolution Independent Line Integral Convolution*, in *Proceedings of SIGGRAPH 95*, R. Cook, Editor. August 1995, ACM SIGGRAPH Addison Wesley: Los Angeles, California. p. 249-256.
20. Shen, H. -W., C.R. Johnson, and K. -L. Ma, *Visualizing Vector Fields Using Line Integral Convolution and Dye Advection* , in *1996 Volume Visualization Symposium*. October 1996, IEEE. p. 63-70.
21. Rezk-Salama, C., *et al.*, *Interactive Exploration of Volume Line Integral Convolution Based on 3D-Texture Mapping*, in *IEEE Visualization '99*, D. Ebert, M. Gross and B. Hamann, Editors. October 1999, IEEE: San Francisco, California. p. 233-240.
22. Crawfis, R. and N. Max. *Direct Volume Visualization of Three-Dimensional Vector Fields*. in *1992 Workshop on Volume Visualization*. 1992. Boston, MA: ACM SIGGRAPH, NY.
23. Max, N., R. Crawfis, and D. Williams, *Visualization for climate modeling*. IEEE Computer Graphics & Applications, July 1993. 13 (4): p. 34-40.
24. Porter, T. and T. Duff, *Compositing Digital Images*. Computer Graphics, 1984. 18(3): p. 253-259.
25. Westover, L. *Interactive Volume Rendering*. in *Chapel Hill Workshop on Volume Visualization*. 1989. Chapel Hill, NC: Department of Computer Science, Univ. of North Carolina.
26. Crawfis, R. and N. Max. *Texture Splats for 3D Vector and Scalar Field Visualization*. in *Visualization '93*. 1993. Los Alamitos, CA: IEEE Computer Society Press.
27. Max, N., R. Crawfis, and C. Grant. *Visualizing 3D Velocity Fields Near Contour Surfaces*. in *Visualization '94*. 1994. Washington, D.C.: IEEE Computer Society Press.
28. Crawfis, R., N. Max, and B. Becker, *Vector Field Visualization*. IEEE Computer Graphics & Applications, 1994: p. 50-56.
29. Helman, J.L. and L. Hesselink, *Visualizing vector field topology in fluid flows*. IEEE Computer Graphics & Applications, May 1991. 11 (3): p. 36-46.