Topological Segmentation in Three-dimensional Vector Fields

Karim Mahrous, Janine Bennett, Gerik Scheuermann, Bernd Hamann, and Kenneth I. Joy

Center for Image Processing and Integrated Computing
Computer Science Department
University of California, Davis

Abstract

We present a new method for topological segmentation in steady three-dimensional vector fields. Depending on desired properties, the algorithm replaces the original vector field by a derived segmented data set, which is utilized to produce separating surfaces in the vector field. We define the concept of a segmented data set, develop methods that produce the segmented data by sampling the vector field with streamlines, and describe algorithms that generate the separating surfaces. This method is applied to generate local separatrices in the field, defined by a movable boundary region placed in the field. The resulting partitions can be visualized using standard techniques for a visualization of a vector field at a higher level of abstraction.

1. Introduction

Classical approaches for bivariate vector field segmentation are based on constructing the separatrix structure of a field – a set of curves in the plane – defining regions that behave qualitatively similarly [9, 15, 18]. Separatrices are usually generated by computing the critical points in the domain of the vector field, determining the types of these critical points, and using numerical methods to trace streamlines originating from so-called "originators" in the field, see [9]. These streamlines form separatrices that segment the field into regions of similar topological behavior.

Unfortunately, similar techniques have not been developed for three-dimensional vector fields. Most visualization techniques for three-dimensional flow fields concentrate on streamline and stream-surface representations, however these techniques do not give a clear illustration of the field's topological behavior. More sophisticated techniques use streamline analysis to extract features of the field , such as attachment and separation lines on a boundary surface. However, separatrix methods have been unavailable for the analysis of three-dimensional fields. Two basic problems have prevented these studies: First, there are few critical points in three-dimensional fields, and second, the numerical marching methods to trace characteristic stream surfaces are difficult to implement and can create substantial numerical errors. In general, visualization and classification of three-dimensional fields continues to be an open problem.

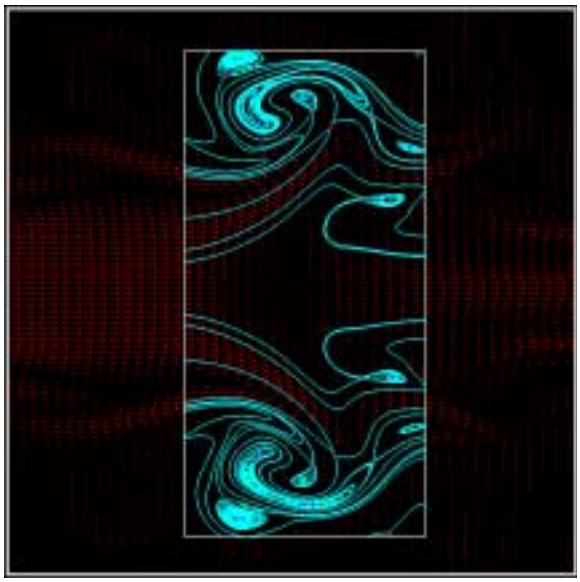

FIGURE 1: Separation of flow in a complex two-dimensional vector field from Scheuermann et al. [19]. Note the separatrices originating from points of tangential flow on the boundary rectangle. This greatly enhances the visualization of the topological flow in the field.

The method presented in this paper is similar to those that generate separatrix structures in two-dimensional vector-fields: focusing on segmentation of the data set based on topological structure. The algorithm is a two-step process that partitions a vector field into regions of topologically similar flow. First, we sample the vector field using streamlines, and replace the original data by a "segmented" data set. Second, a segmentation algorithm generates separating surfaces in the field. We utilize a "local separatrix" concept introduced by Scheuermann et al. [19], which augments the separatrices generated from critical points with "local separatrices" originating from the boundary region of the data set. By segmenting the boundary region into "inflow" and "outflow" regions, a local separatrix is the streamline generated from a point on the boundary where the flow is tangential, see Figure 1. In this way, the algorithm generates a complete separation of the field into regions of similar flow. For visualization of massive data sets, this algorithm can be used to determine similar flow regions within a small region, avoiding the problem of analyzing the complete data set.

We utilize this concept to define a segmentation of a three-dimensional vector field into regions bounded by local separatrices. By manipulation of a "boundary box" in the field, we can create separating surfaces that define regions of similar flow depending on inflow/outflow regions on the boundary of the box. This allows us to place a box in the field and generate separatrices throughout the field, which can be visualized to determine characteristic features of the field.

Given a three-dimensional vector field and a rectangular boundary box B, we define a local separatrix as a stream surface within B that is tangent to the boundary of B. The algorithm generates

these local separatrices by creating a derived "segmented" data set by sampling the original vector field with streamlines. These streamlines terminate either on the boundary or at a critical point. The general idea is to assign a different characteristic marker (or property) to each critical point in the field, and to each contiguous inflow or outflow region of the boundary of B – *i.e.*, regions bounded by lines where the flow is tangential on the boundary of B. Streamlines are initiated at points throughout the data set and traced until they either reach the boundary of B or come arbitrarily close to a critical point. Each of the streamlines is then marked appropriately. We apply the marker information to a vertex of the data set by considering streamlines that lie close to the vertex and assigning markers from these streamlines to the vertex. For example, given a point $\mathbf{p}$ on streamline S that has marker k, if $\mathbf{p}$ lies in tetrahedron T, then we can examine the vertices of the tetrahedron to see which is closest to $\mathbf{p}$. If the closest vertex contains an m-tuple $(m_1, m_2, ..., m_m)$, then we increment m_k. Most vertices will have only one non-zero marker incremented, as most of them will not have local separatrices passing near them. However, some will have multiple markers present.

The m-tuple of markers for each grid vertex is "normalized" to generate a m-dimensional barycentric coordinate tuple at each vertex. This resulting field where each point is associated with a barycentric coordinate tuple is called a *segmented* data set. We utilize "material interface" methods to calculate the boundaries between the regions, using a clipping procedure in barycentric space. The result is a set of local separatrices in the field, separating the flow field according to the inflow and outflow regions of the boundary.

The separating surfaces generated by this method are local separatrices (much like those drawn by Dallmann [5]) of the field defined by the critical points and the field boundary. The user can use a slicing tool (or other technique) to browse through the separatrices, locating vortices and other features.

Section 2 describes work related to streamsurfaces, separatrices and vector fields. Section 3 defines the concept of a *segmented data set*, and discusses methods to find the separating surface between segments. Methods to sample three-dimensional vector fields and convert them to a segmented data set are given in Section 4. Implementation details of the algorithm are discussed in Section 5 and results of the use of this method are given in Section 6.

2. Motivation and Related Work

Classical approaches used for bivariate vector field segmentation are difficult to extend to three-dimensional fields. Two-dimensional vector fields are characterized by their critical points. Classification of these points, and segmentation by separatrices completely characterizes the flow in the two-dimensional case. However, three-dimensional vector fields can be arbitrarily complex, may not

contain critical points (*e.g.*, the NASA delta wing [11]), and may contain complex features that are very difficult to visualize.

Current three-dimensional vector field visualization techniques are based mainly upon streamline and streamsurface generation [9, 15]. These techniques use the definition of the vector field to trace massless particles through the field, tracking their progress by linking them in lines (streamlines) or surfaces (streamsurfaces). Level sets [23] have been used to enhance streamline generation. However, the most important segmentation method for vector fields is the generation of separatrices. Separatrices are streamsurfaces that separate the flow. Scheuermann et al. and others [18, 19, 20, 21, 25] have done research in this area for two-dimensional examples, but little has been done for three-dimensional fields [17].

Other methods of automatic vector field visualization have been proposed that adequately visualize certain features of the data set. However, many of these methods focus on finding certain phenomena within the vector field [5, 8, 9, 11, 12] without attempting to address the partitioning of vector fields based on global topological surfaces. Other techniques to visualize three-dimensional fields [10] also have difficulty in representing topological behavior.

Van Wijk's method [22] creates implicit stream surfaces by calculating streamlines at all grid points. This method assigns values to the stream lines in a region of interest, defining a scalar function that is constant on streamlines. This allows one to obtain streamsurfaces as isosurfaces of this scalar function.

Our method to generate local separatrices [19] is similar to that of Van Wijk as we also sample the vector field using streamlines. We use this sampling to develop a segmentation of the field, effectively replacing the original vector-field information. In this derived representation, the vertices each have an associated barycentric coordinate tuple that represents the "probability" that the local separatrix is close to the vertex. The local separatrices, which are separating surfaces defined by the barycentric coordinate tuples, are generated by a material-interface generation algorithm, similar to that of Bonnell et al. [2]. The output of the algorithm are local separatrices in the field that separate the vector field into regions of similar flow.

3. Segmented Data Sets

Suppose we are given a data set based on an unstructured tetrahedral mesh, and an associated set of "properties" $c_1, c_2, ..., c_m$. For each vertex $\mathbf{p}$ in the data set, we associate an m-tuple $\alpha = (\alpha_1, \alpha_2, ..., \alpha_m)$, where α_i is the fraction of property c_i present (or "valid") at $\mathbf{p}$. We assume that $0 \leq \alpha_i \leq 1$, for $i = 1, ..., m$, and $\sum_{i=1}^{m} \alpha_i = 1$. We will call data sets of this kind *segmented*. A segmented data set is thus one where each vertex $\mathbf{p}$ has an associated barycentric coordinate tuple α.

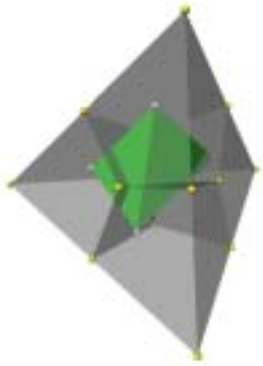

FIGURE 2: The "property space" three-simplex in the case $m = 4$. The figure illustrates a three-dimensional projection of the 3-simplex with an embedded tetrahedron.

Müller [14] and Nielson and Franke [6] defined methods to find the separating surface in an unstructured tetrahedral data set when each vertex is associated with a particular "type" (*i.e.*, exactly one of the α_i values is one for each vertex). Their methods follow the principle of the marching-cubes algorithm of Lorensen and Cline [13], generating a separating surface within each tetrahedron.

Bonnell et al. [2] have solved a more general problem that treats the output from multi-fluid Eulerian hydrodynamics calculations. In their application, grid cells contain fractional volumetric information for each of several fluids, where each cell C of a grid $\mathcal{S}$ has an associated m-tuple $(\alpha_1, \alpha_2, ..., \alpha_m)$ that represents the portions of each of m fluids in the cell. By considering a "dual grid," Bonnell et al. associate the m-tuples with vertices of the dual grid and develop a method that finds a (crack-free) piecewise two-manifold separating surface approximating the boundary surfaces between the various fluids. Given m fluids, the method can potentially calculate $m - 1$ separating surfaces for each tetrahedron.

Following the principles established by Bonnell et al. [2], we consider a 3-simplex (tetrahedron) T in an unstructured three-dimensional grid containing m properties. Each vertex is of the form $(\mathbf{p}, \alpha)$, where $\mathbf{p}$ represents the Euclidean coordinates of the vertex, and $\alpha = (\alpha_1, \alpha_2, ..., \alpha_m)$ is the associated barycentric coordinate tuple. To determine the possible segment boundaries in T, the barycentric coordinate tuples associated with the vertices of T are mapped into a tetrahedron T_α in an $(m-1)$-simplex in "property space." This $(m-1)$-simplex has m vertices, where the kth vertex is associated with a barycentric coordinate that has a value of one in the kth component, and zeros in the remaining components. We construct a Voronoi diagram in the $(m-1)$-simplex, using the vertices of the simplex as the Voronoi points, and calculate the intersections of the 3-simplex T_α with the Voronoi cells in the $(m-1)$-simplex. These intersections define barycentric coordinates

that are used to calculate intersections in the Euclidean-space coordinates of T. Triangulating these coordinates defines the separating surface(s) in T. Executing this method for each tetrahedron of an unstructured simplicial grid will define the separating surfaces of the properties $c_1, c_2, ..., c_m$.

In the case of a tetrahedron with four properties, it is sufficient to assume that each vertex of T has an associated barycentric coordinate tuple $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, where $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$, and $\alpha_i \geq 0$. By considering the 3-simplex having vertices $(1, 0, 0, 0)$, $(0, 1, 0, 0)$, $(0, 0, 1, 0)$, and $(0, 0, 0, 1)$ in property space, a partition of this simplex into Voronoi cells can be defined. The boundaries of these cells are bounded by the faces of the 3-simplex and six 3-dimensional hyperplanes, defined by the set of α such that (i) $\alpha_1 = \alpha_2$, (ii) $\alpha_1 = \alpha_3$, (iii) $\alpha_1 = \alpha_4$, (iv) $\alpha_2 = \alpha_3$, (v) $\alpha_2 = \alpha_4$, and (vi) $\alpha_3 = \alpha_4$. The 3-simplex, resulting Voronoi partition, and an embedded tetrahedron are shown in Figure 2.

Specifically, a tetrahedron T in an unstructured simplicial grid contains k *properties* if there are k indices $i_1, i_2, ..., i_k$, such that the associated barycentric coordinate tuple $\alpha = (\alpha_1, \alpha_2..., \alpha_m)$ of each vertex of T has the property that $\alpha_i = 0$ for $i \neq i_1, ..., i_k$. In the k-property case, a tetrahedron T has an associated tetrahedron T_α in a $(k-1)$-simplex in property space. The $(k-1)$-simplex is partitioned into Voronoi cells whose boundaries consist of the faces of the $(k-1)$-simplex and the $\binom{k}{2}$ hyperplanes defined by the equations $\alpha_i = \alpha_j$, where $1 \leq i < j \leq k$.

Intersections in the property space $(m-1)$-simplex can be found by a straightforward clipping procedure. Suppose that an edge of T_α with endpoints $\alpha^{(1)}$ and $\alpha^{(2)}$ crosses the hyperplane defined by $\alpha_1 = \alpha_2$. If α is the intersection point, we can compute r such that

$$\alpha = (1 - r)\alpha^{(1)} + r\alpha^{(2)}.$$

If the first two coordinates of α are equal, *i.e.*, on hyperplane $\alpha_1 = \alpha_2$, then the first two coordinates of $(1 - r)\alpha^{(1)} + r\alpha^{(2)}$ are also equal. Thus,

$$(1 - r)\alpha_1^{(1)} + r\alpha_1^{(2)} = (1 - r)\alpha_2^{(1)} + r\alpha_2^{(2)},$$

which allows us to calculate r directly. (See Hanson [7] for similar methods.) We utilize a "clipping and capping" algorithm that allows us to iteratively clip against each Voronoi boundary, capping the resulting clipped object at each stage. In this way, the clipped object is always convex, and the capping procedure is straightforward. The polygons of T_α determined by the clipping algorithm are then used to define polygons in the Euclidean coordinates of T, which represent the segment boundaries in T.

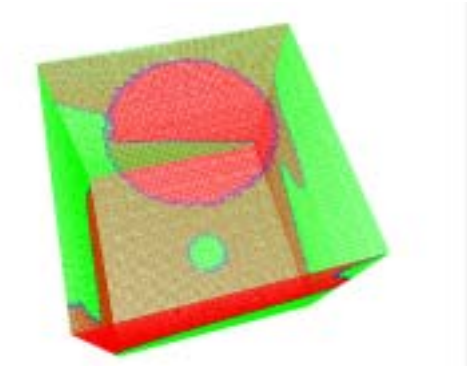

FIGURE 3: The inflow/outflow areas of the boundary box in a tornado data set. The terminating tetrahedra with inflow faces are colored red, and the ones with outflow faces are colored blue. The terminating tetrahedra that have regions of tangential flow are colored green. Note the circles on the top and bottom of the box boundary. These outline the center vortex of the tornado on the boundary of the box.

4. Segmenting Vector Fields

Given a vector field defined over a three-dimensional simplicial grid, we use streamlines to sample the vector field to create a segmentation of the field. The segment boundaries of this field will be approximations of the local separatrices. We first mark all critical points with a unique value (property), and identify and mark all connected inflow regions of the boundary. These marks are the property values of the field. We then sample the field using streamlines, tracing each streamline backward (see [23]) until it reaches either a critical point, or a boundary. Each streamline is associated with a unique marker considering its origin. Next, we transfer the marker information to the vertices, creating barycentric coordinate tuples at the vertices. By applying the segmentation algorithm of Section 3, we generate the local separatrices of the field.

4.1. Marking Boundary Cells

Streamlines terminate at the boundary of the data set, or at a critical point. Tetrahedra lying on the boundary or tetrahedra that contain critical points are called *terminating tetrahedra*. When streamlines encounter a terminating tetrahedron, a marker is assigned to the streamline. Two types of terminating tetrahedra exist: internal and external. Internal terminating tetrahedra contain critical points [19]. External terminating tetrahedra have one or more triangular faces on the domain boundary. There exist three classifications of boundary triangles on an external terminating tetrahedron: A boundary triangle is either an *inflow* triangle, an *outflow* triangle, or it has a area of *tangential flow*. A boundary triangle is an inflow triangle if each vector $\vec{v_1}, \vec{v_2},$ and $\vec{v_3}$ associated with the vertices of

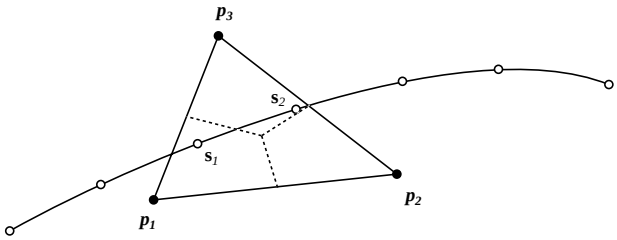

FIGURE 4: Streamline and Triangle. The points $\mathbf{s}_1$ and $\mathbf{s}_2$ cause the markers associated with $\mathbf{p}_1$ and $\mathbf{p}_3$, respectively, to be incremented.

the triangle satisfy

$$\vec{v_i} \cdot \vec{n} > 0$$

where $\vec{n}$ is the inward face normal of the boundary triangle. Similarly, a boundary triangle is an outflow triangle if

$$\vec{v_i} \cdot \vec{n} < 0.$$

If neither property holds, then the face contains points that have tangential flow.

Terminating tetrahedra are marked using two methods: Internal terminating tetrahedra are detected and marked individually while boundary external terminating tetrahedra are marked using an area-growing approach. An unmarked boundary terminating tetrahedron T is identified and its type is determined. A unique mark is then generated and assigned to T. The boundary neighbors of T are identified and inherit the same mark if they have the same boundary flow characteristic as T. The algorithm progresses with the neighbor boundary tetrahedra until all boundary tetrahedra have been marked. Figure 3 illustrates the marking process for terminating tetrahedra on a tornado data set.

4.2. Segmentation

We sample the flow field using streamlines. As streamlines encounter a terminating tetrahedron T, they are assigned the marker k_T associated with T. We then retrace the points that generate the streamline, and for each point $\mathbf{s}$ increment the m_{k_T} property stored at the grid vertex nearest the point $\mathbf{s}$. This is illustrated in Figure 4. Given a streamline point $\mathbf{s}$ in tetrahedron T, we calculate the barycentric coordinate β of $\mathbf{s}$ in T, and increment the m_{k_T} property in the vertex of T that corresponds to the largest component of β. This is illustrated in Figure 5.

After sampling all streamlines, we normalize the property values at the vertices of the grid, creating the required barycentric coordinate tuples. The result is a segmented data set.

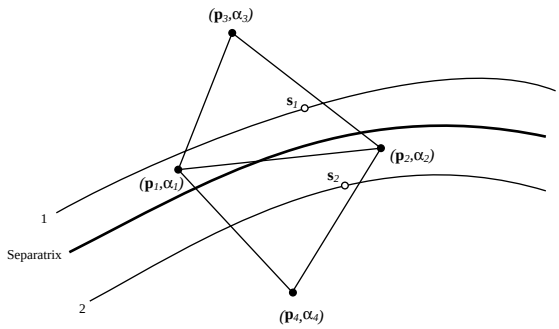

FIGURE 5: Separatrix and Triangle. Points on streamlines on both sides of the separatrix contribute to the barycentric coordinate tuple at a vertex. The streamline with marker 1 and the streamline with marker 2 both contribute to α_2, as the points $\mathbf{s}_1$ and $\mathbf{s}_2$ both lie closer to $\mathbf{p}_2$.

5. Implementation

The algorithm is straightforward to implement and is based upon streamline generation and interface construction. However, we have found that the complexity of vector fields, the size of the data sets, and the sampling strategy creates a number of issues.

- Due to the fact that a large number of inflow/outflow boundary regions may occur on the boundary, the barycentric coordinates tuples may contain a large number of components. However, most of the data points will be associated with only one marker. Several of the data points may be associated with two markers near the local separatrices, and the case where three or more markers are present will be rare. Thus we never store the full barycentric coordinate, but only those components of each coordinate that are non-zero. This enables the algorithm to work with a large number of "properties."
- Many three-dimensional vector fields have *orbits*, *i.e.*, closed streamlines, that never enter a terminating cell, see [26]. In this case, a separate "property marker" must be used for each orbit. We have tested several heuristic algorithms to detect orbits and have implemented one that correctly identifies orbits when two-tetrahedron patterns are repeated along a streamline. This strategy seems to work well in practice and is illustrated in the results below.
- The seed points of streamlines can be calculated in different ways. A randomized Monte Carlo approach can be used, as can a stratified sampling approach, see [3]. In our implementation, we used an approach that distributes points uniformly throughout a tetrahedron and this seems to work well. Effective seeding of streamlines is still an issue of further research.
- To reduce potential numerical errors, the accuracy of the generated streamlines is carefully monitored in our implementation (see [16]). Streamline propagation was restricted to a per-

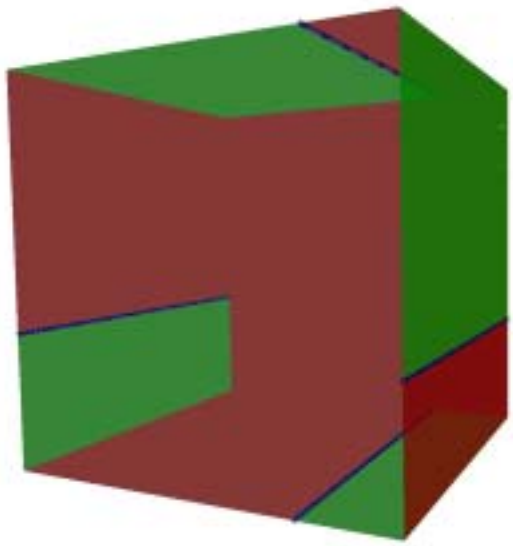

FIGURE 6: The boundary box of the circle data set colored as to the flow of the field across the boundary of the box. Inflow faces are shown in red, outflow in green, and faces that have tangential flow are shown in blue.

tetrahedron basis, *i.e.*, step size is controlled locally by considering the size of a tetrahedron.

6. Results

Notes for the editors: We have kept this illustrations large in this section of the paper for review purposes. The pictures will be reduced to one or two pages in the final version.

We have tested this algorithm on several scientific data sets. The initial data set is a circular field, generated over a $10 \times 10 \times 10$ grid. The boundary box is offset to better illustrate the flow. Figure 6 shows the boundary box with outlines of the tangential flow boundaries. The inflow boundary faces are shown in red, the outflow faces in green, and the tangential faces in blue. Figure 7 shows a streamline representation of the field, colored by the property markers. Figure 8 illustrates the results of the segmentation algorithm on this low-resolution data set.

The second example is a flow field from a simulated tornado. This data set was generated by Crawfis and Max [4] to illustrate flow patterns in three-dimensional flow fields. The data set is $64 \times 64 \times 64$. Figure 9 shows the boundary box and the flow patterns on the faces of the box. There are two outflow regions on the boundary of the box, a circular green region on the bottom of the box, and all other regions shown in green. Figure 10 shows a side view of illuminated streamlines (see Zockler et al. [27]) in the tornado flow field. The faces of the boundary box are again colored

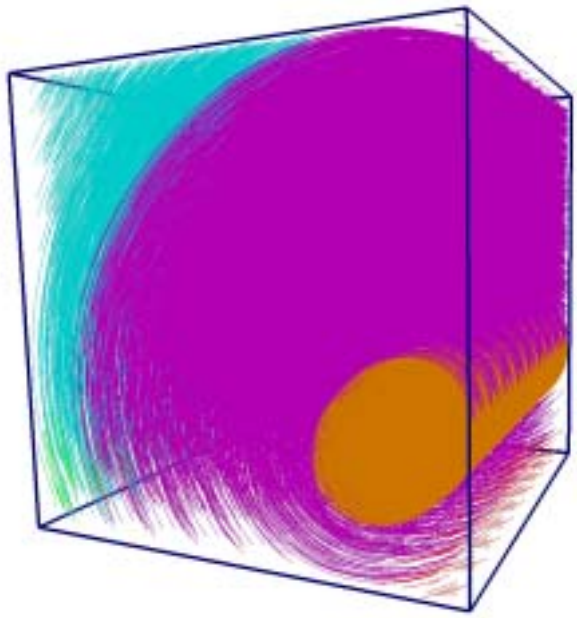

FIGURE 7: Streamline representation of the flow in the circle data set. Regions are colored based upon sample properties. The orange streamlines are identified as an orbit by the algorithm. The small regions of red and green streamlines in the lower-left and lower-right regions of the data set correspond to the separating surfaces seen in Figure 8.

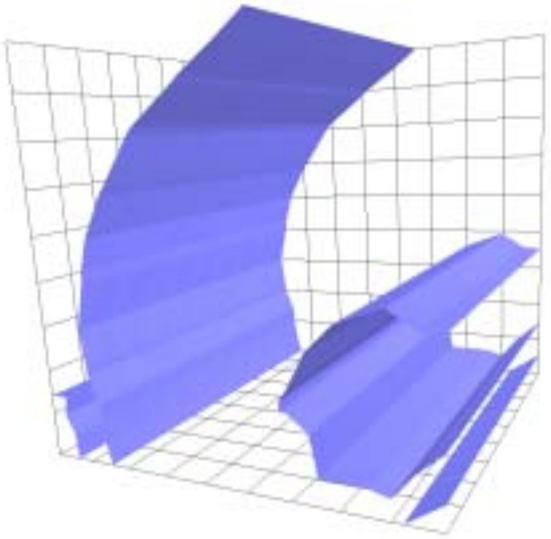

FIGURE 8: The separating surfaces generated by the algorithm on the circle data set. In this low-resolution example, the ”kinks” are produced as an artifact of the “marching tetrahedra” algorithm.

FIGURE 9: The boundary box and the flow patterns on the faces of the box. The inflow faces are shown in red, the outflow faces in green and the faces with tangential flow are shown in blue.

to represent the flow across their faces. The two outflow regions are clearly visible in this picture. Figure 11 illustrates the output of the segmentation algorithm with a view that corresponds to that of Figure 10. Notice that the algorithm correctly identifies the funnel as the boundary of the two outflow regions. Here we marked streamlines only by the outflow region. Figure 12 illustrates the streamlines and segmented surface from above.

The third example is from a computational simulation of a spherical argon bubble that is hit by a 1.25 Mach shock in the air. The bubble is deformed through interaction with the vorticity generated as the shock passes over the bubble. This data set was generated at the Center for Computational Sciences and Engineering at Lawrence Berkeley National Laboratory and has been used in a variety of adaptive mesh refinement methods (see Berger and Colella [1]). The data set is $128 \times 128 \times 256$ and we illustrate the field at time step 500. Here the argon bubble has deformed into a shape with a characteristic "smoke ring." Figure 13 gives a side view of illuminated streamlines that represent the field at this time step. The boundary box is also shown with color coding for the flow over the faces of the box. Figure 14 shows the result of the segmentation algorithm on this data set. The separatrices shown separate the flow between outflow regions on the boundary box, and also correctly identifies the ring with the vortices due to the turbulence. The ring is generated by identifying orbits in the streamlines, see Section 5. Figure 15 and Figure 16 give a front view of the argon bubble flow field and the results of the segmentation algorithm. Figure 17 and Figure 18 show close-up views of the ring identified by the segmentation algorithm.

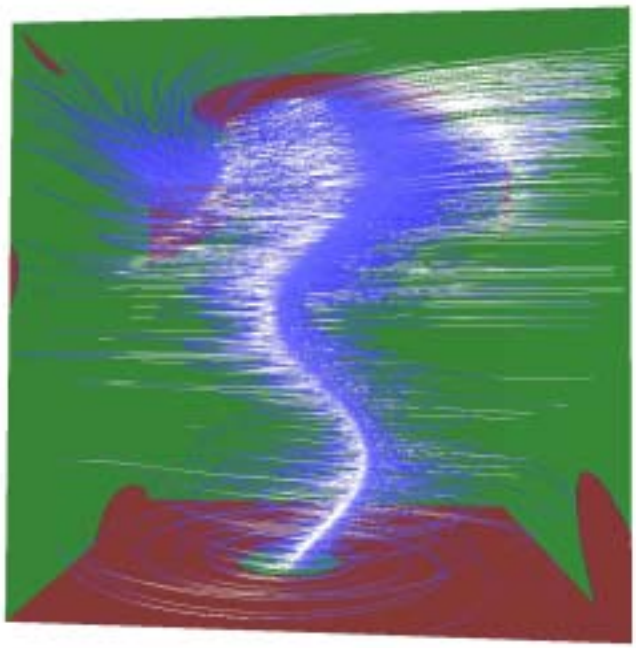

FIGURE 10: A side view of illuminated streamlines in the tornado data set. The faces of the boundary box are again colored according to the flow across the faces. In this illustration, the streamlines seeded outside the funnel travel to outflow regions on the sides and top of the boundary box. Streamlines seeded within the funnel travel to the outflow region at the bottom. on the top of the data set to the outflow regions on the bottom.

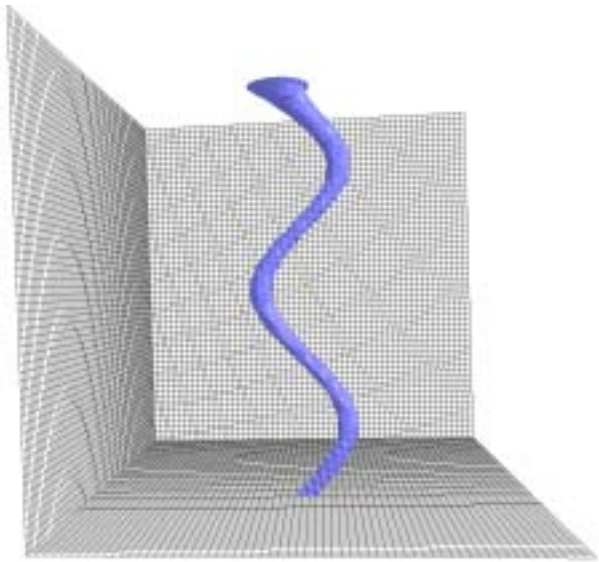

FIGURE 11: The output of the segmentation algorithm on the tornado data set showing the separatrix between streamlines that flow to different outflow regions. This illustration was produced using the same viewpoint as Figure 10.

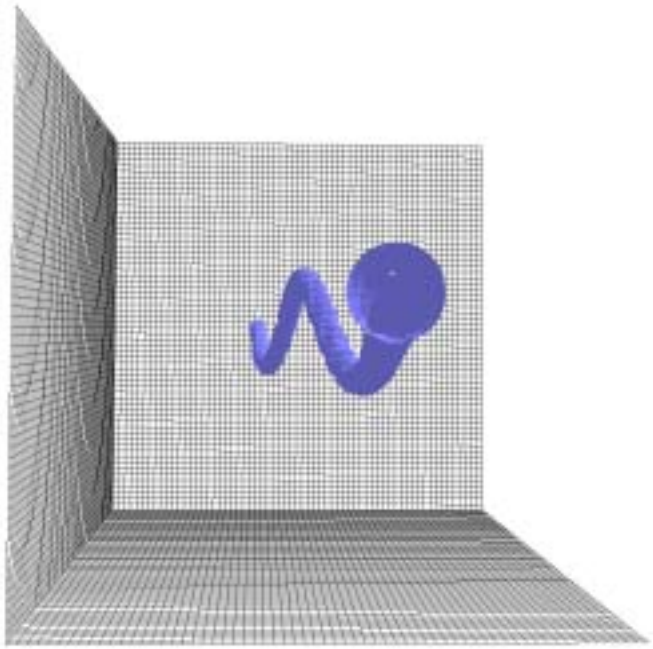

FIGURE 12: A top view of the results of the segmentation algorithm for the tornado data set.

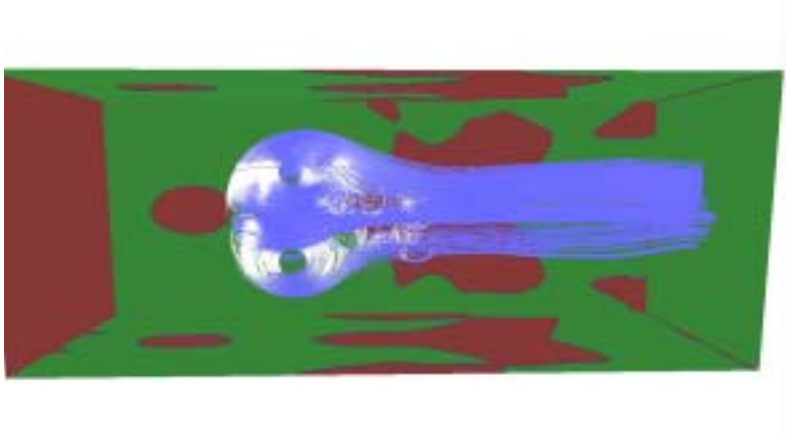

FIGURE 13: A side view of illuminated streamlines representing time step 500 in the argon bubble data set. The boundary box is also shown with faces colored as to the flow across the boundary of the box. The streamlines shown have been selected to illustrate the shockwave.

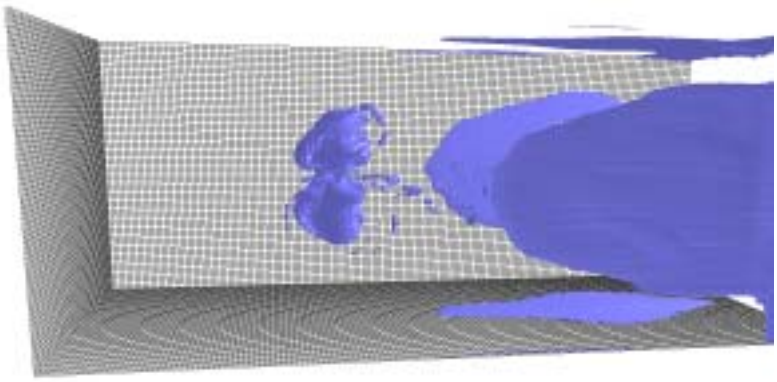

FIGURE 14: A side view of the result of the segmentation algorithm on the argon bubble data set. The shockwave (ring) is clearly identified along with closed surfaces trailing the shockwave due to the turbulence.

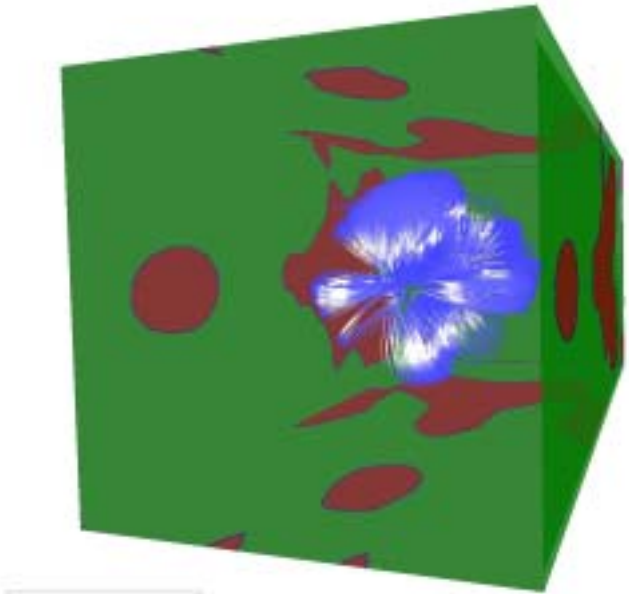

FIGURE 15: A front view of the illuminated streamlines in the argon bubble data set.

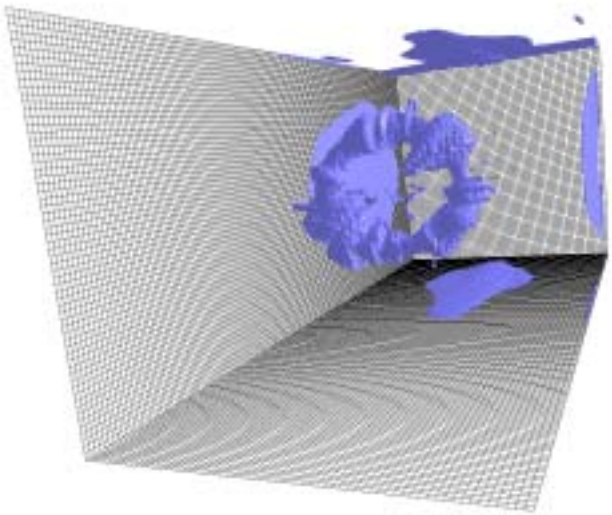

FIGURE 16: A front view of the results of the segmentation algorithm on the argon bubble data set.

FIGURE 17: A close-up view of the "ring" identified by the segmentation algorithm.

FIGURE 18: A close-up view of the “ring” identified by the segmentation algorithm.

7. Conclusions and Future Work

We have presented a new algorithm to generate separating surfaces in three-dimensional vector fields. The algorithm samples the vector field using streamlines, replaces the original vector field by a derived segmented data set, and uses the segmented data to generate the separating surfaces. The algorithm is straightforward to implement and is applicable to a large number of data sets.

The proposed algorithm could be improved in several ways. Using knowledge about regions of tangential flow on the boundary and the critical points, we could seed our streamlines near critical points [24] and near tangential flow regions [19]. This strategy avoids processing streamlines away from the local separatrices. We plan to utilize these techniques to improve the sampling process and produce an "interactive" data exploration environment for vector fields. These separating surfaces should also be generalized to time-varying data.

8. Acknowledgments

This work was supported by the National Science Foundation under contracts ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the National Institutes of Health under contract P20 MH60975-06A2, funded by the National Institute of Mental Health and the National Science Foundation; and the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B528318. We thank the members of the Visualization Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis for their support. We express our special thanks to Oliver Kreylos and David Wiley.

References

[1] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–84, 1989.

[2] Kathleen Bonnell, Dan Schikore, Mark Duchaineau, Bernd Hamann, and Kenneth I. Joy. Constructing material interfaces from data sets containing volume fraction information. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings of IEEE Visualization 2000*, pages 367–372. IEEE Computer Society Press, October 2000.

[3] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, January 1986. also in Tutorial: Computer Graphics: Image Synthesis, Computer

Society Press, Washington, 1988, pp. 283-304. Comments on article in ACM TOG, v. 9, n. 2, p 233-243.

[4] R. A. Crawfis and N. Max. Texture splats for 3D scalar and vector field visualization. In Gregory M. Nielson and Dan Bergeron, editors, *Proceedings of the Visualization '93 Conference*, pages 261–267, San Jose, CA, October 1993. IEEE Computer Society Press.

[5] Uve Dallmann. Topological structures of three-dimensional flow separations. Technical Report 221-82, Deutsche Forschungs- und Versuchsanstalt für Luft- and Raumfahrt, 1983.

[6] Richard Franke and Gregory M. Nielson. Scattered data interpolation and applications: A tutorial and survey. In H. Hagen, H. Müller, and G.M. Nielson, editors, *Focus on Scientific Visualization*, pages 131–159. Springer-Verlag, New York, 1995.

[7] Andrew J. Hanson. Geometry for n-dimensional graphics. In Paul Heckbert, editor, *Graphics Gems IV*, pages 149–170. Academic Press, Boston, 1994.

[8] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *Computer*, 22(8):27–36, August 1989.

[9] James Helman and Lambertus Hesselink. Representation and display of vector field topology in fluid flow data sets. In G. M. Nielson and B. D. Shriver, editors, *Visualization in Scientific Computing*, pages 61–73. IEEE Computer Society, 1990.

[10] Victoria Interrante and Chester Grosch. Visualizing 3D flow. *IEEE Computer Graphics & Applications*, 18(4):49 – 53, July – August 1998.

[11] D. N. Kenwright. Automatic detection of open and closed separation and attachment lines. In Hans Hagen David Ebert and Holly Rushmeier, editors, *IEEE Visualization '98*, pages 151–158, October 1998.

[12] David N. Kenwright, Chris Henze, and Creon Levit. Feature extraction of separation and attachment lines. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):135–144, April 1999.

[13] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21 (4), pages 163–170, July 1987.

[14] Heinrich Müller. Boundary extraction for rasterized motion planning. In Horst Bunke, Tankeo Kanade, and Hartmut Noltemeier, editors, *Modeling and Planning for Sensor Based Intelligent Robot Systems*, pages 41–50. World Scientific Publishers, 1995.

[15] Gregory M. Nielson, Hans Hagen, and Heinrich Müller. *Scientific Visualization: Overviews, Methodologies, and Techniques*. IEEE Computer Society Press, 1997.

[16] Gregory M. Nielson and Il-Hong Jung. Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):360–372, 1999.

[17] Gerik Scheuermann, Thomas Bobach, Hans Hagen, Karim Mahrous, Bernd Hamann, and Kenneth I. Joy. A tetrahedra-based stream surface algorithm. In Thomas Ertl, Kenneth I. Joy, and Amitabh Varshney, editors, *Proceedings of IEEE Visualization 2001*, pages 83–91, October, 2001.

[18] Gerik Scheuermann, Hans Hagen, and Heinz Krüger. Clifford algebra in vector field visualization. In Hans-Christian Hege and Konrad Polthier, editors, *Mathematical Visualization*, pages 343–351. Springer Verlag, Heidelberg, 1998.

[19] Gerik Scheuermann, Bernd Hamann, Kenneth I. Joy, and Wolfgang Kollmann. Visualizing local vector field topology. *SPIE Journal of Electronic Imaging*, 9(4):356–367, October 2000.

[20] Gerik Scheuermann, Xavier Tricoche, and Hans Hagen. C1-interpolation for vector field topology visualization. In David Ebert, Markus Gross, and Bernd Hamann, editors, *Proceedings of IEEE Visualization '99*, pages 271–278, October, 1999.

[21] Xavier Tricoche, Gerik Scheuermann, and Hans Hagen. A topology simplification method for 2D vector fields. In Thomas Ertl, Bernd Hamann, and Amitabh Varshney, editors, *Proceedings Visualization 2000*, pages 359–366, 2000.

[22] Jarke J. van Wijk. Implicit stream surfaces. In Gregory M. Nielson and Dan Bergeron, editors, *Proceedings of the Visualization '93 Conference*, pages 245–252, October 1993.

[23] Rüdiger Westermann, Christopher Johnson, and Thomas Ertl. Topology-preserving smoothing of vector fields. In *IEEE Transactions on Visualization and Computer Graphics*, volume 7, pages 222–229. IEEE Computer Society, 2001.

[24] Rüdiger Westermann, Christopher Johnson, and Thomas Ertl. A level-set method for flow visualization. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings of IEEE Visualization 2000*, pages 147–154, October 2000.

[25] Thomas Wischgoll and Gerik Scheuermann. Detection and visualization of closed streamlines in planar flows. In *IEEE Transactions on Visualization and Computer Graphics*, volume 7(2), pages 165–172. IEEE Computer Society, 2001.

[26] T. Wishgoll, G. Scheuermann, and H. Hagen. Distributed computation of planar-closed streamlines. In Robert Erbacher, Philip Chen, Matti Gröhn, Johnathan Roberts, and Craig Wittenbrink, editors, *Proceedings SPIE Visualization and Data Analysis 2002*, pages 41–50. SPIE, January 2002.

[27] Malte Zöckler, Detlev Stalling, and Hans-Christian Hege. Interactive visualization of 3D-vector fields using illuminated streamlines. In *Proceedings of IEEE Visualization '96, San Francisco*, pages 107–113, October 1996.