Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2003)
G.-P. Bonneau, S. Hahmann, C. D. Hansen (Editors)

Improving Topological Segmentation of Three-dimensional Vector Fields

Karim M. Mahrous, Janine C. Bennett, Bernd Hamann and Kenneth I. Joy

Center for Image Processing and Integrated Computing
Department of Computer Science
University of California, Davis
Davis, CA 95616-8562
{mahrous,bennettj,hamann,joy}@cs.ucdavis.edu

Abstract
We present three enhancements to accelerate the extraction of separatrices of three-dimensional vector fields, using intelligently selected "sample" streamlines. These enhancements reduce the number of needed sample streamlines and their propagation length. Inflow/outflow matching supports the simultaneous extraction of topologically significant inflow and outflow separatrices in a single pass. An adaptive sampling approach is introduced and used to seed streamlines in a more meaningful and efficient manner. Cell-locking is a new concept that isolates regions of a data set that do not contain separatrices. This concept makes streamline propagation more efficient as streamlines are not propagated through cells that do not influence or contain separatrices. These enhancements enable us to perform separatrix construction for three-dimensional vector field data requiring less overall computation.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Viewing algorithms I.3.4 [Computer Graphics]: Application packages

1. Introduction

Most topological approaches for bivariate vector field segmentation construct the separatrices defining the *topological skeleton* of a flow field. The topological skeleton is given by sets of curves (in two dimensions) or surfaces (in three dimensions) that separate the domain of the field into regions of similar flow. In two dimensions, these approaches have their origin in critical point analysis. One identifies all the critical points in the field, finds the attachment and separation points on the domain boundary and uses these points, propagated through the flow field, to define its topological skeleton.

These approaches, until recently, had not been extended to three dimensional vector fields, at least not from a computational perspective. There are two main reasons for this fact: First, critical points rarely appear in three dimensional vectors fields. In fact, vector fields that contain a large number of vortices (the three-dimensional generalization of a critical point in two dimensions) may contain no isolated critical points at all. Second, in three dimensions, the instabilities of propagation (stream line integration) methods, such as Runge-Kutta, are inherently more noticeable and greatly affect the overall structure of the separatrices. From a computational point of view it is extremely challenging to devise sound, effective, and efficient methods to extract and represent geometrically the separatrix structure of a three-dimensional vector field.

Recent work by Mahrous et al.[12] allows for topological separation of vector fields using a "segmented data set" approach. It is similar to previous two-dimensional approaches in that it isolates the separatrices in three-dimensional vector fields. It is a two-step algorithm: First it replaces the vector field with a segmented data set; second, that data set is processed using a material interface algorithm to generate the final separatrices. The algorithm determines both *local separatrices*, separatrices generated from attachement/separaration points on the boundary, and *critical point separatrices*, separatrices formed by connections among critical points contained in the field.

The basic idea of the algorithm by Mahrous et al. is to

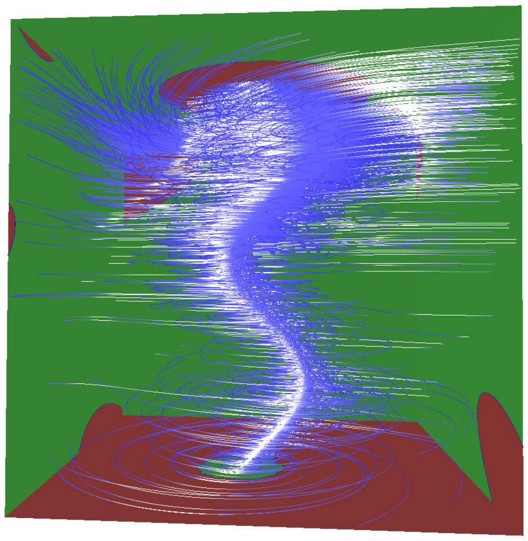

Figure 1: ***Topological boundaries of the tornado data set.*** *The boundary of the tornado data set is shown, where red signifies inflow portions of the boundary and green outflow. Illuminated streamlines*[22] *are used to show the flow and structure of the tornado's core (from Mahrous et al.*[12]*). Data set courtesy of LLNL*[2].

use "property markers" on each *inflow region* or *outflow region*. These property markers are used in conjunction with streamlines to assign to each vertex (of a mesh discretizing the domain) the location that flow near that vertex ultimately tends to travel towards. A Voronoi diagram is constructed inside each cell by using the the vertices of the cell as the Voronoi points. Each vertex, v, thus obtains a fractional representation of the travel paths of the streamlines that travel through the Voronoi cells that contain v, see Figure 2. The tangential boundaries between these travel paths can be expressed as surfaces, which are the topological boundaries, or separatrices, of the vector field.

This approach, while able to extract the topological surfaces of the field, is inefficient in some stages: some regions of the vector field are continuously refined when they contain no portion of the separatrix. Also, some streamlines are propagated through sections of the vector field when their property could be determined earlier. Furthermore, to completely capture all separatrices of a field, two passes would need to be made, one propagating the streamlines forward and one propagating the streamlines backward.

The methods presented in this paper extract the same topological surfaces in a single pass and in a much more efficient manner. Section 2 discusses motivation and related work, while Section 3 summarizes the original algorithm. Section 4 presents the improvements made to the original algorithm and in Sections 5 and 6 we discuss results and conclusions respectively.

2. Motivation and Related Work

Traditional bivariate vector field segmentation techniques characterize two-dimensional fields by their critical points. Classification of these points and segmentation by separatrices, coupled with identification of attachement/separation points, completely describes the flow in the two-dimensional case. However, three-dimensional fields can be arbitrarily complex, may not contain isolated critical points (consider, for example, the NASA delta wing[9]), and may contain complex features that are difficult to visualize. As a result, classical bivariate approaches are difficult to extend to three-dimensional fields.

Current three-dimensional vector field visualization techniques rely mainly on streamline and streamsurface generation[6], [13]. These approaches trace massless particles through the vector field, representing their trajectories by linking them in lines (streamlines) or surfaces (streamsurfaces). Level sets[20] were used to enhance streamline generation. However, the most important vector field segmentation technique is the generation of separatrices. Separatrices are streamsurfaces that separate the flow into topologically similar regions. Scheuermann et al.[15, 16, 17], and others,[18, 21] have done research in this area for two-dimensional vector fields. Recent work has been done as well[12, 14] to identify these separatrices for three-dimensional vector fields.

Various vector field visualization techniques have been proposed to visualize certain features of the data set, rather than partitioning the vector fields based on topological surfaces. Certain phenomena are found within the vector field[3, 7, 6, 9, 10] and then visualized.

Van Wijk's method[19] creates implicit stream surfaces by calculating streamlines at all grid points. Values are assigned to the streamlines in regions of interest, defining a scalar function that is constant on streamlines. Streamsurfaces are then obtained as isosurfaces of this scalar function.

Work done by Mahrous et al.[12] to generate local separatrices[16] also samples a given vector field using streamlines. Sampling is used to generate the information needed to derive a segmentation of the field. The original vector field data at each vertex is replaced by a barycentric coordinate tuple that represents the "probability" for a local separatrix to be close to that vertex. A material interface algorithm, similar to that of Bonnell et al.[1], can then be used to generate local separatrices. The local separatrices that divide the vector field into regions of similar flow behavior are the separating surfaces defined by the barycentric coordinate tuples. Our approach enhances the work done by Mahrous et al.[12] by improving the underlying sampling technique. Furthermore computation time is reduced substantially by adding "cell locking" and inflow-outflow matching.

3. Converting Vector Fields

Given a vector field defined on a three-dimensional simplicial (tetrahedral) grid, we define a labelling scheme for its vertices and use it to determine "terminating cells." Streamlines are seeded and propagated until they encounter one of these terminating cells. The streamlines acquire the "property marker" of the terminating cell encountered and pass this property to the vertices along discretely sampled paths. These properties are then used to create a "segmented data set." Finally, a material interface algorithm approximates local separatrices. Summarized, the algorithm consists of three steps:

1. Labelling: Locate all terminating cells in the data set and using an area-growing approach[12] label each contiguous group with distinct property markers.
2. Sampling and Tabulating: Seed streamlines and allow them to propagate through the data set, inheriting the properties of their associated terminating cells. Then, distribute the properties from the streamlines to the original data locations, replacing the original vector field with a segmented data set.
3. Creating Separatrices: Apply a material interface algorithm to the segmented data set to produce the desired separatrices.

3.1. Marking Terminating Cells

A cell C is considered a terminating cell when a streamline s exists such that, for an infinitely small stepsize t and a propagation method (such as Runge-Kutta), repeated iteration of the advection function will not move s into another cell in the data set. A cell C can be a terminating cell due to practical reasons (e.g., the boundary of the data set is encountered) or due to theoretical reasons (e.g., C contains a sink as a critical point).

Each terminating cell in the data set needs to be located and labelled, which is done in two stages. The first stage locates critical points in the data set, see Figure 3. Often none exist due to the fact that even extremely complex three-dimensional vector fields rarely contain critical points. However, if any do exist they must be labelled with an appropriate inflow or outflow property. The second stage locates and appropriately labels the terminating cells on the domain boundary. This step is done using an area-growing approach on the boundary.

First, we construct a list B of all boundary faces (triangles) of the mesh. We determine an element $b \in B$ such that b is not yet labelled with a property. The element b can have as its associated property inflow, outflow, or tangential flow. Let

$$\vec{v}_1, \ldots, \vec{v}_n$$

be the vectors associated with the vertices of b and $\vec{n}$ be the inward normal of b with respect to the boundary of the domain. Then, b is an inflow face if

$$\vec{v}_1, \ldots, \vec{v}_n \text{ all satisfy } \vec{v}_i \cdot \vec{n} > 0$$

and an outflow face if

$$\vec{v}_1, \ldots, \vec{v}_n \text{ all satisfy } \vec{v}_i \cdot \vec{n} < 0.$$

If neither of these two conditions hold, i.e.

$$\exists i \text{ such that } \vec{v}_i \cdot \vec{n} > 0 \text{ and } \exists j \text{ such that } \vec{v}_j \cdot \vec{n} < 0,$$

then b is called a tangential flow face. Tangential flow faces have two properties, an inflow property and an outflow property. When b's type is determined, it is appropriately labelled, see Figure 4.

When b is labelled, we queue all of its neighbors. Next we label each neighbor with the same property if it has the same boundary flow condition, i.e., if b is an inflow face and

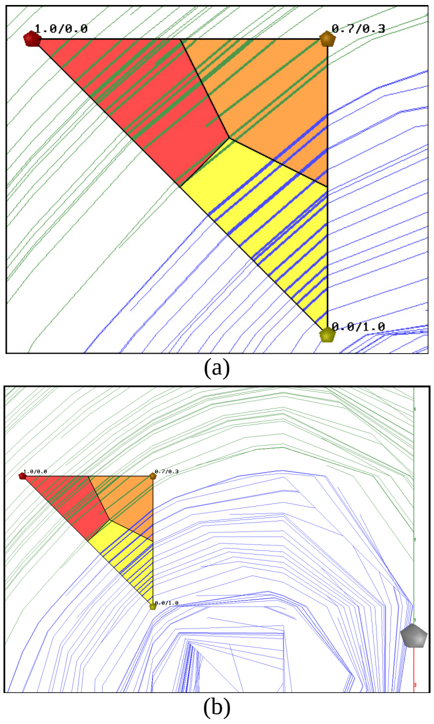

Figure 2: ***Sampling with streamlines and tabulating their values.*** *Figure (a) shows a close-up view of a cell in a 2D data set. The Voronoi cell decomposition is shown to illustrate tabulation assignments (e.g. all portions of the streamlines that are in the red section of the cell contribute to the red vertex). The accumulated properties are also shown, on a per-vertex basis. Thus the red vertex has 100% contribution from the green streamlines, the yellow vertex has 100% contribution from the blue streamlines and the orange vertex has 70% green contribution and 30% blue contribution. Figure (b) shows the cell embedded in the data set. Notice the green streamlines accumulate their properties from the green section of the boundary, while the blue streamlines are accumulating their properties from an internal "orbit."*

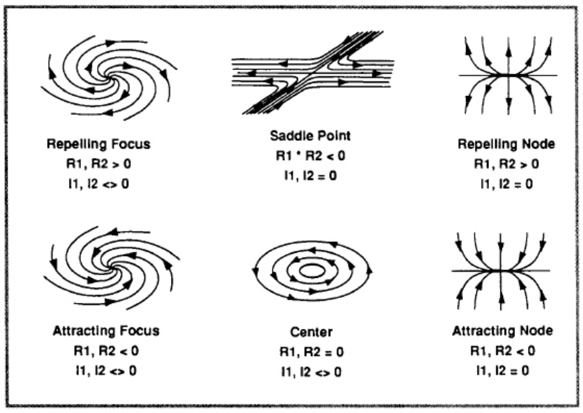

Figure 3: ***Critical point classification.*** *Classification criteria for critical points. "R1 and R2 denote the real parts of the eigenvalues of the Jacobian, i1 and i2 the imaginary parts"*[7]*. (from Helman and Hesselink* [7]*).*

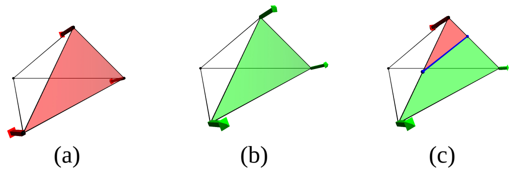

Figure 4: ***Inflow, outflow, and tangential faces on a tetrahedron.*** *Figure (a) shows a tetrahedron with an inflow face marked in red. Figure (b) shows a tetrahedron with an outflow face marked in green. Figure (c) shows a tetrahedron with a tangential flow face. The inflow section of the face is marked in red, the outflow section is marked in green, and the linearly interpolated tangential flow line is marked in blue.(We use linear interpolation over a tetrahedron.)*

is labelled with an inflow property, then we only label those of b's neighbors that are also inflow. If the neighbor is a tangential flow boundary face, then it is labelled since it represents both inflow and outflow. If the current queue element is labelled, then we queue its neighbors and continue this process until the queue is empty. When the queue is empty, we repeat the process again by first incrementing the label that was used in the previous pass, and then identifying another unlabelled cell b.

At the end of this procedure, all terminating cells in the data set are labelled, and these labels can be assigned to those streamlines that terminate in these cells.

3.2. Sampling and Tabulating

The first step in sampling the data set is to determine the seed locations for streamlines. While this step is one of the advancements to the algorithm, its general form will be discussed here, leaving the specifics to section 4.3. The "ideal" sampling will require the minimal number of seeded streamlines, covering the maximal number of cells in the data set. Thus, the final property distribution will ideally cover the maximal number of vertices.

The actual propagation of the streamlines is performed using a standard integration technique (Runge-Kutta). When a streamline s encounters a cell from which it cannot proceed to another cell (i.e., when encountering a critical point or the boundary) an appropriate property label L_s is determined and the cell acquires that label, see Figure 2. Then, each point p in s "finds" the vertex v in the data set that it is closest to (using the largest barycentric coordinate component from the tetrahedral cell that contains p) and "informs" v of the inherited property L_s. Each vertex has an associated array of property counters (which we call its associated barycentric property tuple) and increments the appropriate counters (or components) according to the properties of the streamlines "passing by".

When this step is done for all streamlines used to sample the data set, all vertices' associated barycentric property tuples are normalized. At this point, the data set is segmented and its separatrices can be determined via a material interface algorithm.

3.3. Creating Separatrices

Definition: Given a data set based on an unstructured tetrahedral mesh, and an associated set of material "properties" $c_1, c_2, \ldots, c_m$, we associate with each vertex p in the data set an m-tuple $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$, where α_i is the fraction of property c_i present (or "valid") at p. We assume that $0 \le \alpha_i \le 1, i = 1, \ldots, m$, and $\Sigma_1^m \alpha_i = 1$. We call data sets of this kind *segmented*.

Following the principles established by Bonnell et al.[1], we consider a 3-simplex (tetrahedron) T in a tetrahedral grid containing m properties. Each vertex is of the form (p, α), where p represents the Euclidean coordinates of the vertex, and $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ is the associated barycentric coordinate tuple. To determine the possible segment boundaries in T, we first determine the number of "active" properties in T. A tetrahedron T contains k active properties if there are k indices $i_1, i_2, \ldots, i_k$, such that the associated barycentric property tuple $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ of each vertex of T has the property that $\alpha_i = 0$ for $i \neq i_1, \ldots, i_k$.

Next, the barycentric property tuples associated with the vertices of T are mapped into a tetrahedron T_α in a $(k-1)$-simplex in "property space," see Figure 5. This $(k-1)$-simplex has k vertices, where the ith vertex is associated with a barycentric coordinate that has a value of one in the ith component, and zeros in the remaining components. We construct a Voronoi diagram in the $(k-1)$-simplex, using the vertices of the simplex as the points for which to construct the Voronoi diagram. The boundaries of the resulting

Voronoi cells consist of the faces of the $(k-1)$-simplex and the $\binom{k}{2}$ hyperplanes defined by the equations $\alpha_i = \alpha_j$, where $1 \le i < j \le k$. Next, we calculate the intersections of the 3-simplex T_α with the Voronoi cells in the $(k-1)$-simplex. These intersections define barycentric coordinates that are used to calculate intersections in T in Euclidean three-space. By triangulating these points, we obtain the separating surface(s) in T. This step is applied to each tetrahedron, and we obtain as a result the separating surfaces of the properties $c_1, c_2, \ldots, c_m$.

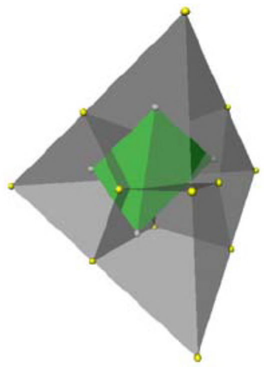
Figure 5: ***Property space three-simplex.***
The "property space" three-simplex in the case $m = 4$ (four properties). The figure illustrates a three-dimensional projection of the three-simplex with an embedded tetrahedron.

Intersections in the property space $(k-1)$-simplex can be found by a 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 a parameter r such that

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

If the first two coordinates of α are equal, 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[4] 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. The clipped object is always convex, and the capping procedure is straightforward. The polygons of T_α determined by the clipping algorithm are used to define polygons in T in Euclidean three-space which represent the segment boundaries in T.

4. Improvements

The improvements of the algorithm are motivated by the desire to make the algorithm more robust and more efficient. Three improvements are introduced in this paper: inflow/outflow matching, adaptive sampling, and cell-locking. These improvements are explained in detail in the next sections.

4.1. Inflow/Outflow Matching

The previous version of the algorithm, as discussed in Mahrous et al.[12], while capturing all topological structures related to forward propagation does not capture some structures that can be considered necessary for a "complete" topological skeleton. To determine remaining portions of the topological skeleton, not only is it important to determine the forward advection location of each seed point, but it is also important to determine the backward advection of the seed point. Thus, when propagating a streamline we must determine both its "exit property" as well as its "entrance property." Note that a streamline can enter the data set at a source or at a point on the boundary.

In the original version of the algorithm, while both inflow and outflow terminating cells are labelled, inflow labels remain essentially unused during the remaining portion of the algorithm. This extension, in a single pass, propagates streamlines forward and backward. As a result, the inflow and outflow label markers are now used simultaneously. This approach raises several issues.

We use a tetrahedral decomposition and linear interpolation within cells. Due to numerical inaccuracies, it can be quite difficult to determine through which face of a tetrahedron a streamline exits. For example, when a streamline leaves a tetrahedron close to an edge, it can become very hard to determine its exit face. This can lead to problems when determining a property for a streamline if it leaves the data set through the edge of a cell that does not contain a boundary face (and therefore does not contain a property marker). In order to deal with this issue, we employ search heuristics (such as breadth-first neighbor search) to determine the most appropriate label for a streamline.

The process of updating the barycentric property tuples at each vertex is more complicated than in the original method. Previously, the number of components in the barycentric property tuple equaled the total number of property markers used by the algorithm (since only exit property markers were considered). Now that both entrance and exit property markers are accounted for, the total number of components in each barycentric property tuple must equal the number of distinct entrance/exit property pairs. This consideration dramatically increases the size of the tuples. However, the run time performance of the material interface algorithm is not impacted.

4.2. Cell-locking

The process of seeding and computing streamlines is important for property determination. If this property could be

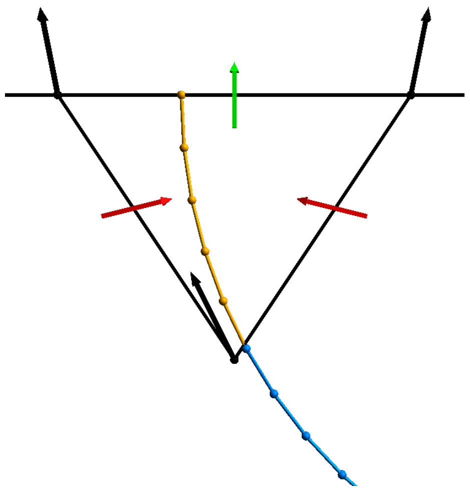

Figure 6: ***Locking cells that initially contain a property.*** *This boundary cell's boundary face has a specific property P. The green vectors indicate outflow, and the red vectors indicate inflow. Thus, the only outflow location is on the boundary face and a streamline entering this cell has only one possible exit location. The cell can therefore be "locked" with the property P.*

determined without running the streamline fully (or at all), then calculation time could be saved. The basic idea underlying our cell-locking strategy is driven by the observation that propagation through some cells is not necessary when there is only one possible resulting property. Consider, for instance, the simple case of a cell C on the boundary of the domain. Let C have only one outflow face f, and let that face be on the boundary, see Figure 6. Any streamline s entering C must exit C on face f. Cell-locking allows any streamline that enters C to stop when entering C and inherit the respective property.

This insight can be used to save streamline computations, and generally allows us to lock a few boundary cells. However, to achieve significant speed-up, a larger number of cells throughout the data set must be locked. In order to accomplish this goal (having determined the locked boundary cells) we queue neighbor cells of locked boundary cells. To illustrate this step, consider a face-neighbor of C (the locked boundary cell) with only one outflow face that leads to C; this cell can also be locked with the same property as C. In general, a cell on the queue can be locked if all of its outflow faces are locked with the same property, see Figure 7. The property can be propagated to subsequent neighbors until some portion of the original data set is locked. An analogous approach exists for locking cells based on inflow properties. Furthermore, more aggressive and more clever heuristics can be used to find even more "lockable" cells.

When all lockable cells are locked, all streamlines that would have been seeded in these cells can be ignored since

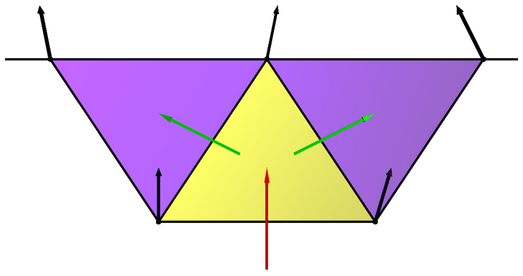

Figure 7: ***Locking cells that do not initially contain a property.*** *Demonstrating cell-locking in cells that do not originally contain a property: The purple cells are locked with some property, and it is being determined whether the yellow cell can be locked. The vectors are color-coded with respect to the yellow cell, green indicating outflow and red indicating inflow. Since both of the yellow cell's outflow neighbors are locked with the same property it also can be locked with that property.*

their properties are know a priori. Furthermore, all streamlines that encounter a locked cell terminate since their property is known as soon as they enter a locked cell.

4.3. Adaptive Sampling

The overall algorithm described is not designed to be interactive. Rather, it is expected that it might require a significant amount of time to compute the separatrices (which are stored in polygonal format for visualization purposes). However, acceleration strategies also lead to a "more accurate" surface. The original algorithm relies heavily on "processing power" to generate reasonably good approximations of the separatrices; it basically uniformly seeds streamlines in different cells. While this approach produces correct results, it adds to computational cost. No fully automatic streamline seeding algorithm is known today. However, even without such a tool, great improvements can be made over uniform sampling. We employ a four-step sampling process: "Initial", "unsampled," "undersampled," and "separatrix" seeding.

Initial seeding is the process of defining the first set of seed points for streamlines. While using field curvature information can determine good seed locations, we have found that the time necessary to compute the additional curvature information does not pay off. A user could determine initial seed locations, for example, through an input file. In our experience, however, seeding the initial set of streamlines manually does not (in most cases) greatly affect the overall number of streamlines necessary. Thus, we still choose initial seed locations by uniform sampling, as in the original algorithm. This step is executed in a single pass. Each of the adaptive sampling steps (unsampled, undersampled, and separatrix) require multiple passes.

The first step of each of the adaptive seeding approaches is memory calculation to determine the maximal number of seedable streamlines (based on average length of streamlines seeded thus far). The unsampled seeding process checks the vertices of the data set and seeds streamlines near those vertices whose barycentric property tuples have not been modified (i.e., vertices whose properties are still zero). This step terminates when the barycentric property tuples for all vertices have been modified.

Next, we perform undersampled seeding. For each vertex v in the data set, we consider its associated barycentric property tuple b_v. We then calculate S_v, the sum of each of the components of b_v. The value of S_v represents the number of points along streamlines that contribute to b_v. For each undersampled seeding pass, we seed additional streamlines near vertices in increasing order of S_v.

Our last adaptive step is separatrix seeding. The purpose of this step is to seed additional streamlines near vertices that lie close to local separatrices. For each vertex v we consider its associated barycentric property tuple b_v and S_v, the sum of the components of b_v. We also consider G_v, the number of components of b_v that are greater than zero, i.e., the number of "active" properties. If G_v is greater than one, then there are streamlines near v that begin/end at different sources and terminal locations. Therefore, it is possible that v lies near a separatrix. In this step, we seed additional streamlines near vertices in increasing order of G_v. Many vertices v in the data set only have one or two active properties, and, as a result, there will be many vertices having the same number of active properties. Among vertices with the same number of active properties, we seed additional streamlines in increasing order of S_v, similarly to the undersampled seeding step.

The total number of passes performed for each of the adaptive steps is a user-defined parameter that should be a balance between desired accuracy and computation time.

5. Results

To demonstrate the adaptive sampling and cell-locking results achieved, we used the tornado data set from the original method[12]. The tornado data set is a flow field on a $64 \times 64 \times 64$ grid, see Crawfis and Max[2]. Figure 8 shows the tornado data set at several different sampling resolutions. The adaptive sampling steps generate a good approximation of the separatrix with a fraction of the streamlines required by the original method.

When locking cells based on outflow, we were able to lock 11 percent of the total 1,572,864 tetrahedral cells. Figure 9 shows the boundary of the locked outflow cells in green. All tetrahedra that lie between the green surface and the data set boundary are locked. We achieve similar results when locking inflow cells with a total of around 20 percent of cells locked. This reduces the average streamline length as well as eliminates 20 percent of the streamlines required to generate the separatrices.

Figure 10 shows the tornado data set using the inflow/outflow enhancement. The core separatrix generated by the original method is diminished as well as additional separatrices separating regions where streamlines enter and/or exit the data set domain from boundary regions with differing property values.

We did not have access to any three-dimensional data sets that contain critical points. Therefore, we generated one on a $20 \times 20 \times 20$ grid with diagonally-constant flow. A single critical point was generated on an edge shared by four tetrahedra by reversing the direction of the flow along a row of vectors on one of the boundaries of the data set. This critical point data set was generated to illustrate the results of inflow/outflow matching. Figure 11 shows streamlines in the critical point data set, as well as the separatrices generated using inflow/outflow matching. This data set highlights why it is important to consider both the entrance and exit properties of a streamline. The original algorithm would not have created any separatrices due to the fact that only one exit property exists. Since all streamlines exit the data set from faces with the same property, no separatrices are generated. However, using the inflow/outflow enhancement, we see that two surfaces exist that separate the flow into three distinct regions.

6. Conclusions

We have presented three enhancements to an algorithm that generates separatrix surfaces in three-dimensional vector fields: inflow/outflow matching, cell-locking, and adaptive sampling. These enhancements allow us to generate a more topologically significant set of surfaces using overall less computational work.

Future work can be done to create even more aggressive cell-locking and adaptive sampling techniques. Furthermore, we would like to apply this to time-varying data and move away from the use of the boundary information entirely, relying only on the internal flow of the field to generate separatrices.

Acknowledgements

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award) and ACI 0222909, 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 Institute of Mental Health and the National Science Foundation under contract NIMH 2 P20 MH60975-06A2; and the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159. We also acknowledge the support of ALSTOM Schilling Robotics and SGI. We thank the members of the Visualization and Graphics

Research Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis. We thank especially Oliver Kreylos and David Wiley for their help and suggestions.

References

1. K. Bonnell, D. Schikore, M. Duchaineau, B. Hamann, and K. I. Joy. Constructing material interfaces from data sets containing volume fraction information. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings of IEEE Visualiztion 2000*, pages 367–372, Los Alamitos, October 2000. IEEE, IEEE Computer Society Press.
2. 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.
3. U. Dallmann. "topological structures of three-dimensional flow separations". Technical report 221-82, "Deutsche Forschungs- und Versuchsanstalt f(ü)r Luft- and Raumfahrt", 1983.
4. A. J. Hanson. Geometry for N-dimensional graphics. In P. Heckbert, editor, *Graphics Gems IV*, pages 149–170. Academic Press, Boston, 1994.
5. J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *Computer*, 22(8):27–36, August 1989.
6. J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *Visualization in scientific computing*, pages 61–73, 1990.
7. J. Helman and L. Hesselink. Visualization of vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991.
8. V. Interrante and C. Grosch. Visualizing 3D flow. *IEEE Computer Graphics & Applications*, 18(4), July – August 1998. ISSN 0272-1716.
9. D. N. Kenwright. Automatic detection of open and closed separation and attachment lines. In *IEEE Visualization ’98*, pages 151–158, Washington - Brussels - Tokyo, October 1998. IEEE.
10. D. N. Kenwright, C. Henze, and C. Levit. Feature extraction of separation and attachment lines. *IEEE Transactions on Visualization and Computer Graphics*, 5(2), April 1999.
11. 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.
12. K. M. Mahrous, J. C. Bennett, G. Scheuermann, B. Hamann, and K. I. Joy. Topological segmentation in three-dimensional vector fields. Technical Report 36, Computer Science Department, University of California, Davis, 1 Shields Ave, Davis, CA, 95616, 2002.
13. G. M. Nielson, H. Hagen, and H. Müller. *Scientific Visualization: Overviews, Methodologies, and Techniques*. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1997. IEEE catalog number BP07777.
14. G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, and K. I. Joy. A tetrahedra-based stream surface algorithm. In *Proceedings of IEEE Visualization 2001*, pages 83–91, 2001.
15. G. Scheuermann, H. Hagen, and H. Krüger. Clifford algebra in vector field visualization. In Hans-Christian Hege and Konrad Polthier, editors, *Mathematical Visualization*, pages 343–351. Springer Verlag, Heidelberg, 1998.
16. G. Scheuermann, B. Hamann, K. I. Joy, and W. Kollmann. Visualizing local vector field topology. *SPIE Journal of Electronic Imaging*, 9(4):356–367, oct 2000.
17. G. Scheuermann, X. Tricoche, and H. Hagen. C1-interpolation for vector field topology visualization. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization ’99*, pages 271–278, San Francisco, 1999. IEEE.
18. X. Tricoche, G. Scheuermann, and H. Hagen. A topology simplification method for 2D vector fields. In *Proceedings Visualization 2000*, pages 359–366. IEEE Computer Society Technical Committee on Computer Graphics, 2000.
19. J. J. van Wijk. Implicit stream surfaces. In Gregory M. Nielson and Dan Bergeron, editors, *Proceedings of the Visualization ’93 Conference*, pages 245–252, San Jose, CA, October 1993. IEEE Computer Society Press.
20. R. Westermann, C. Johnson, and T. Ertl. A level-set method for flow visualization. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings Visualization 2000*, pages 147–154. IEEE Computer Society Technical Committee on Computer Graphics, 2000.
21. T. Wischgoll and G. 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.
22. M. Zöckler, D. Stalling, and H.-C. Hege. Interactive visualization of 3D-vector fields using illuminated streamlines. In *Proceedings of IEEE Visualization ’96, San Francisco*, pages 107–113, October 1996.

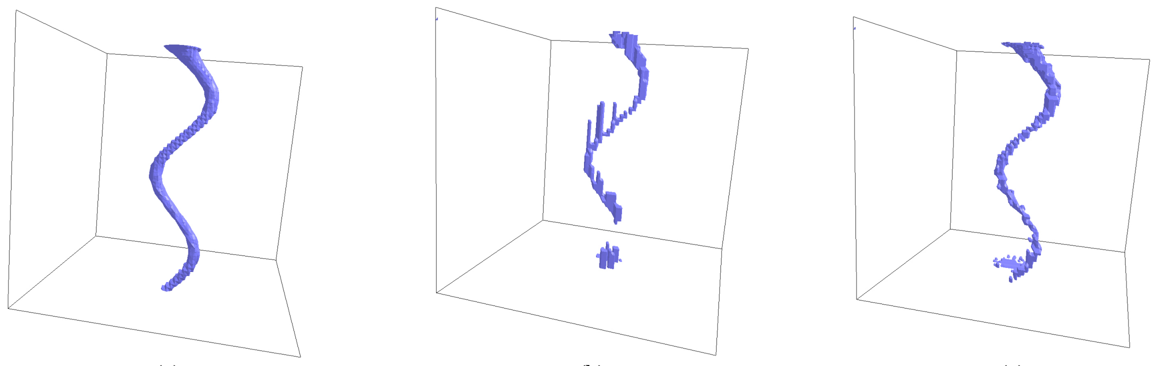

Figure 8: ***Tornado separatrices at different sampling intervals.***
Figure (a) shows the tornado data set uniformly sampled with several million uniform *samples. Figure (b) shows the tornado data set with 32,000* uniform *samples. Figure (c) shows the tornado data set with 1,800* adaptive *samples. During creation of the separatrices, if a vertex remains un-sampled a property must be artificially assigned to it. Techniques such as interpolation with neighbor quantities can be employed. However, in some cases artifacts are still produced such as in Figure (b).*

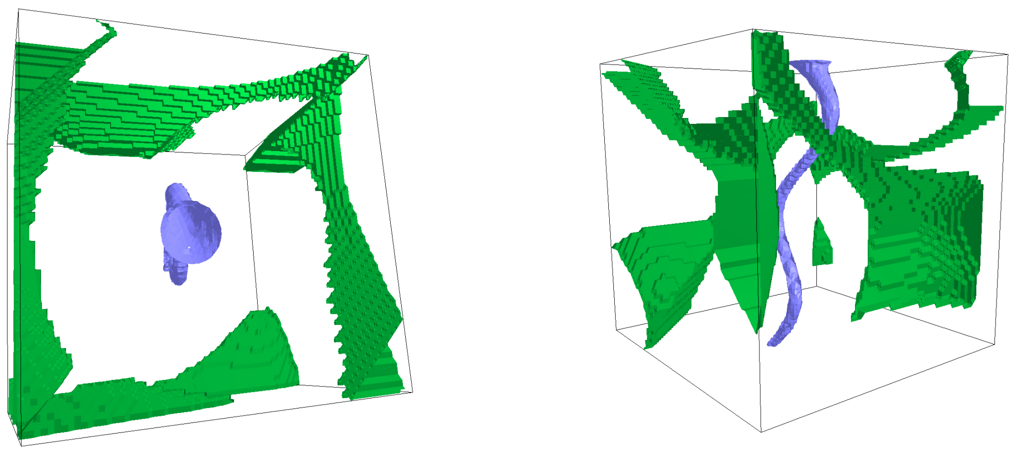

Figure 9: ***Cell-locking in tornado data set.***
Figures (a) and (b) show the boundary of the locked outflow cells in green. All tetrahedral cells that lie between the green surface and the data set boundary are locked.

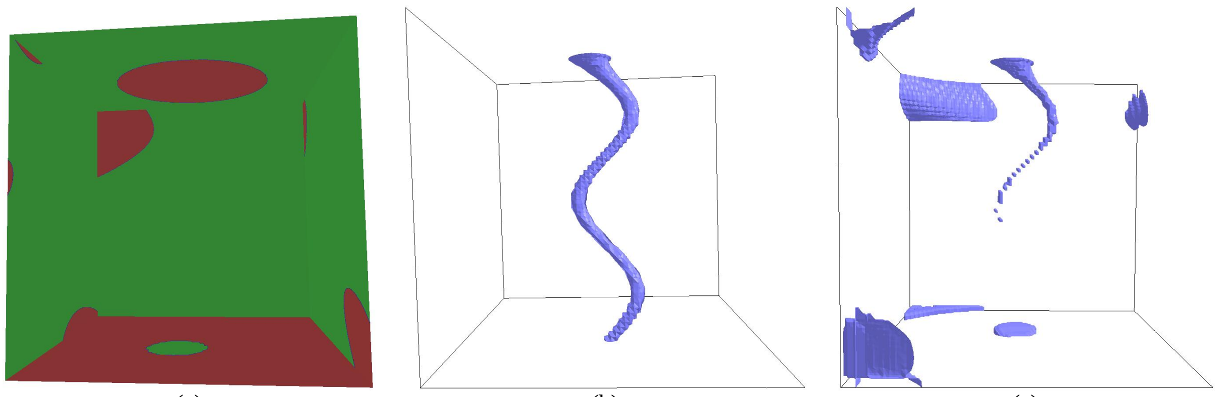

Figure 10: ***Separatrices found using inflow/outflow improvement in tornado data set.***
Figure (a) shows the topological separation of the boundary of the tornado data set. Figure (b) shows the topological separation generated by the original algorithm. Figure (c) shows the topological separatrices generated using the inflow/outflow improvement. This improvement separates regions of flow that have distinct entrace/exit conditions, thus the “pockets” of separated flow in the corners of the data set.

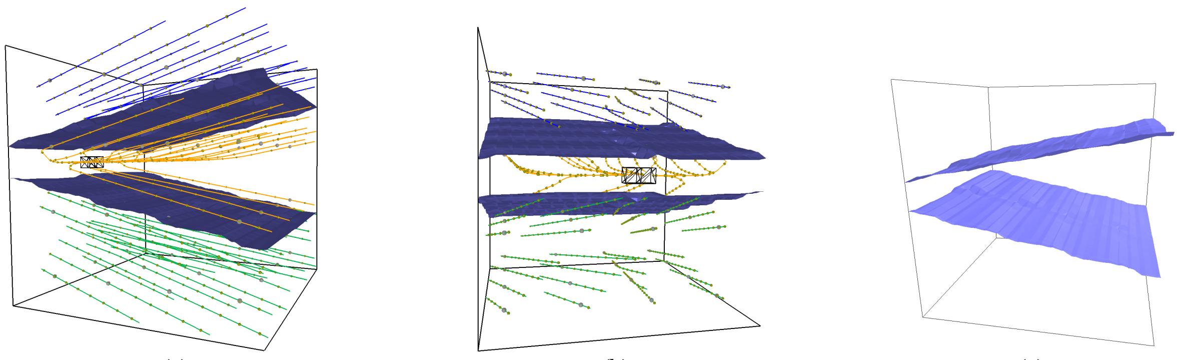

Figure 11: ***Separatrices found using inflow/outflow enhancement in the critical point data set.***
Streamlines can have one of three different entrance properties, however, they all have the same exit property. Figures (a) and (b) show streamlines in addition to the separatrices. The critical point is located on the shared diagonal of the tetrahedra outlined in black. Figure (c) shows the separatrices that would not have been generated by the original algorithm.