Streamline Visualization of Multiple 2D Vector Fields

Timothy Urness[a] and Victoria Interrante[b]

[a]Drake University, 2507 University Avenue, Des Moines, IA
[b]University of Minnesota, 4-192 EE/CS Building, 200 Union Street SE, Minneapolis, MN

ABSTRACT

The analysis of data that consists of multiple vector fields can be greatly facilitated by the simultaneous visualization of the vector fields. An effective visualization must accurately reflect the key physical structures of the fields in a way that does not allow for an unintended bias toward one distribution. While there are several effective techniques to visualize a single vector field through equally-spaced streamlines, applying these techniques to individual vector fields and combining them in a single image yields undesirable artifacts. In this paper, we present strategies for the effective visualization of two vector fields through the use of streamlines.

Keywords: Flow Visualization, Vector Field Visualization

1. INTRODUCTION

The visualization of co-existing vector fields has several practical applications. Examples include: the analysis of co-existing magnetic and electrical fields, the fluid dynamics application of velocity and vorticity vector fields, and the astronomical analysis of velocity and magnetic vector fields in magnetohydrodynamic supersonic jets.[1] The analysis of the data can be greatly facilitated by a better understanding of the complicated interaction between the two vector fields. The significant challenge in interpreting the results of these scientific applications is determining how and where the separate vector fields are correlated or not correlated. The visualization methods presented in this paper are designed to allow each field to be understood and analyzed both individually and in the context of the other without creating an unintended bias toward one distribution.

In an effort to illustrate how to visually represent two vector fields simultaneously, we examine two streamline representations distinguished by color. In figure 1, we provide two examples in which different equally-spaced streamline representations are simply overlaid. While the color of the streamlines distinguishes each vector field appropriately, the field in which the streamlines are "on top" is more visually prominent than the underlying vector field. As one of the goals of this research is to represent both vector fields without one being more salient than the other, this simple technique produces an undesirable result. This paper focuses on methods that will produce images representing two vector fields in which the key structures and characteristics of both fields can be seen without a bias of one vector field over another. The algorithm for constructing overlapping streamlines is presented in section 3.

2. RELATED WORK

2.1 Streamline Visualization

A streamline is a line that is tangent to the velocity vector at every point. Streamlines are traditionally used in flow visualizations as they give the direction and orientation of the flow at each point along the line. In the simplest case, we define a stationary 2D vector field as a map $v : \mathbb{R}^2 \to \mathbb{R}^2, x \mapsto v(x)$. A streamline is defined by its *integral curve* or path, $\sigma(u)$, whose tangent vectors coincide with the vector field:

$$\frac{d}{du}\sigma(u) = v(\sigma(u)). \tag{1}$$

Further author information: (Send correspondence to T.U.)
T.U.: E-mail: timothy.urness@drake.edu, Telephone: 1 515 271 2118
V.I.: E-mail: interran@cs.umn.edu, Telephone: 1 612 625 3543

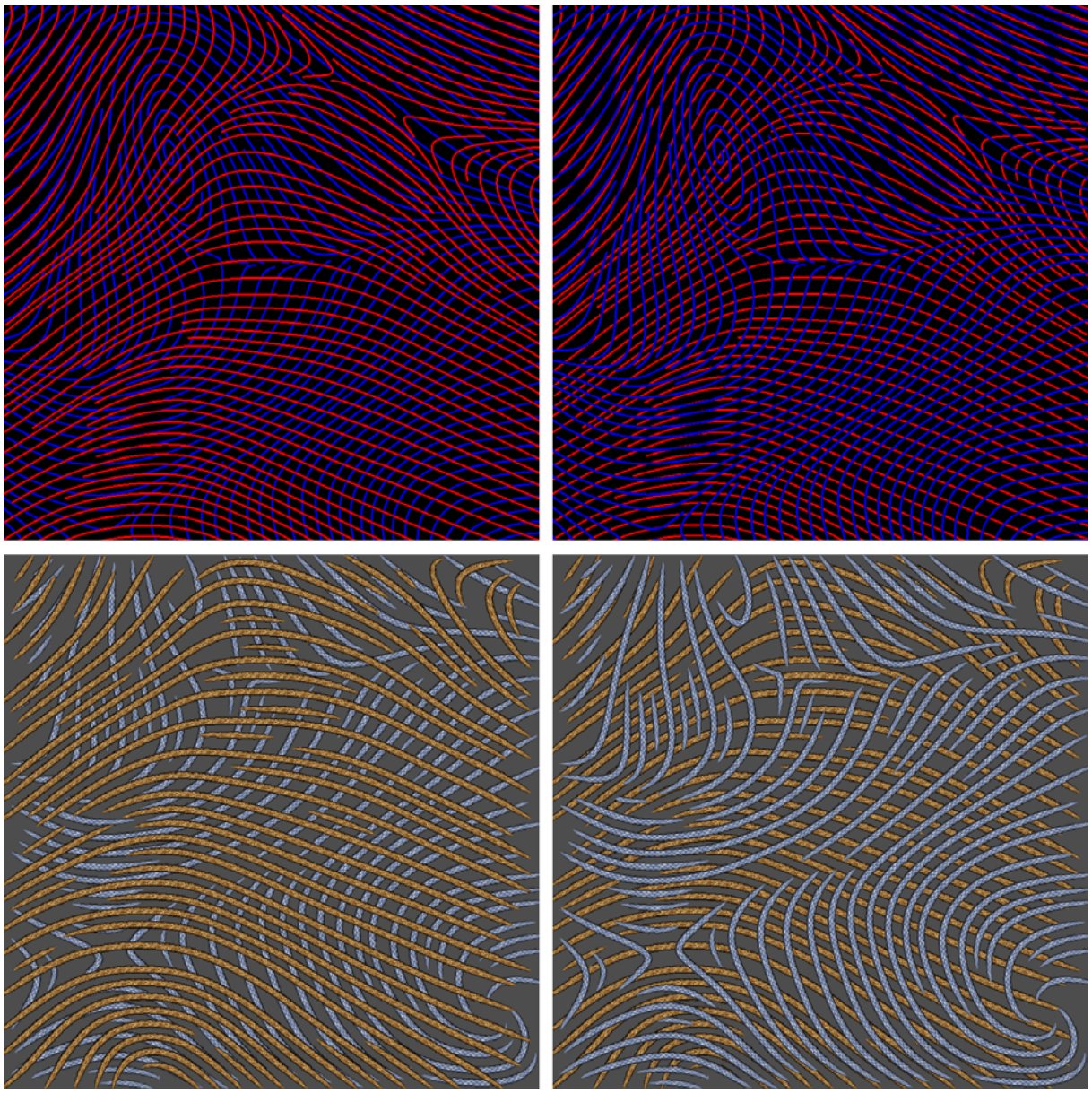

Figure 1. Overlaying streamline images results in the vector field "on top" appearing more prominent than the underlying vector field. Upper Left: red streamlines over blue streamlines. Upper Right: blue streamlines over red streamlines. Lower Left: textured tan streamlines over textured light-blue streamlines. Lower Right: textured light-blue streamlines over textured tan streamlines.

In practice, a streamline is constructed from an initial seed point by numerically integrating the above formula. Many algorithms exist for the placement or seeding of streamlines of a single vector field.[2–6] The goal of these algorithms is to place streamlines in a way to avoid visual clutter and accurately depict the key physical structures of the vector field.

The pioneering technique developed by Turk and Banks utilizes an energy function to guide the placement of streamlines at a specified density for a single vector field.[4] The algorithm applies energy-decreasing functions to refine the placement and length of individual streamlines. A low-pass filter of the streamline image measures the *quality* of the placement of the streamlines – ensuring a uniform balance of streamlines over the domain.

Quantifying the *quality* by using the low-pass filtered image allows for a measure of the difference between successive iterations of the method. While computationally expensive, the algorithm effectively creates an image of equally-spaced streamlines for a single vector field.

An alternative technique for streamline placement developed by Jobard and Lefer incorporates a computationally-efficient greedy algorithm.[2] The algorithm begins by placing a seed point at a random location in the domain and computing the interpolating streamline. New seed candidates are selected that are a user-defined distance from the points on the previously defined streamline (figure 2). Streamlines are constructed from these seed points continuing until the streamline reaches a singular point in the vector field, the streamline reaches the boundary of the domain, or the streamline comes within a user-supplied distance of another previously defined streamline. The process of determining new points at a distance from existing streamlines continues until no other valid seed points can be found.

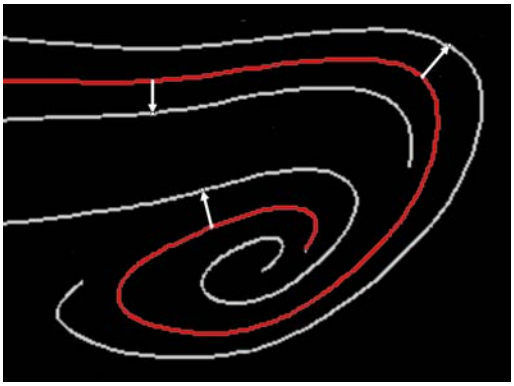

Figure 2. The Jobard and Lefer algorithm selects seed points at an equal distance away from a calculated streamline. The original streamline is colored red. Equally-spaced streamlines are colored grey.

2.2 Multiple Field Visualization

Transparency or layering is one of the most effective methods to portray relationships between overlaying or co-existing components. Kirby et al. introduced a method inspired by the layering techniques used in oil paintings to visualize components of 2D flow data.[7] Interrante has used transparent stroked textures on 3D surfaces superimposed over underlying opaque structures.[8] Weigle et al. demonstrated a texture generation technique, based on the layering of oriented slivers, using orientation and luminance to encode multiple overlapping scalar fields.[9] Hotz et al. vary the input texture density and spot size in addition to kernel length and overlay sparse Line Integral Convolution images to visualize features of a tensor field.[10] Urness et al. developed strategies for visualizing multiple vector fields by utilizing different methods for representing each field.[1] Our research is inspired by these techniques, and seeks to expand the applications in the visualization of multiple related vector fields.

In the following sections, we describe how these algorithms can be adapted for use in visualizing multiple, coincident vector fields. We present techniques that represent both fields accurately, with equal prominence, and minimize visual artifacts.

3. STREAMLINE WEAVE

As an alternative to simply overlaying one vector field representation over another, we seek to *interweave* the streamlines as to give a balanced representation of both vector fields simultaneously.

3.1 Streamline Calculation and Visualization

Streamlines are initially calculated using the given vector field and a numerical integration technique such as the adaptive-step-size fourth-order Runge-Kutta method. As introduced by Taponecco et al.,[11] we construct

a 2D *thick streamline* by considering the normal component along each point of a calculated streamline. A user-specified width is multiplied by the normal component to give the location for each point of the thick streamline. The coordinates of the thick streamline are used to construct polygons in which a texture can be easily applied using standard texture mapping techniques. This results in a streamline representation that can be texture-mapped to reflect different colors and texture patterns that provide a richly diverse set of possibilities for the visualization of flow data.

The visualization of streamlines on a 2D domain requires that, in certain cases, the streamlines be terminated. This typically occurs at singularities in the vector field or in the event of maintaining a uniform spacing of the streamlines. To reduce the abrupt truncation of streamlines, we employ a technique of tapering the ends of streamlines.[2,4] In the case of the calculated *thick streamlines*, the coordinates of the polygon vertices are computed to converge at the end of the streamline.

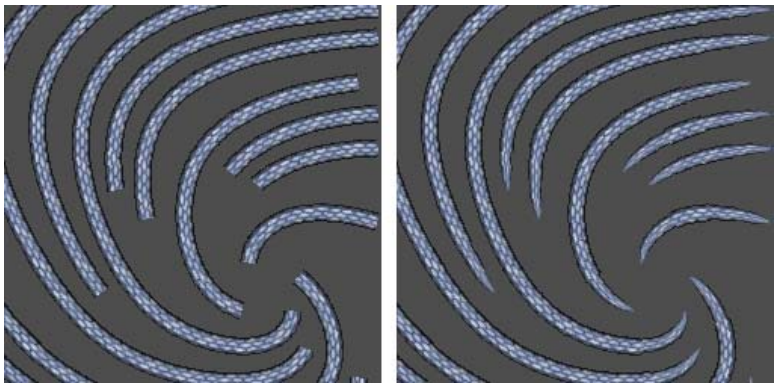

Figure 3. Tapering streamlines reduces the visual artifact that occurs when thick streamlines are terminated. Left: non-tapered streamlines. Right: tapered streamlines.

3.2 Streamline Halo

One of the most important concerns in depicting the interweaving of streamlines is how to graphically represent the overlapping elements to be distinct from each other. Artists and illustrators have traditionally used the technique of line continuity to depict one line passing underneath another. The size, quality, thickness, and discontinuity of lines allow for an enriched vocabulary in the representation of visual objects.[12,13]

Inspired by Interrante and Grosch,[14] we define a texture-mapped 2D texture "halo" around the streamlines, to not only give it a 3D appearance, but also to visually separate it from streamlines from a different vector field at a point of intersection (figure 4). While the vector fields represented are defined on the same 2D plane, discontinuities in the lines are utilized to allow for the perception that one line is underneath the other when two lines cross.

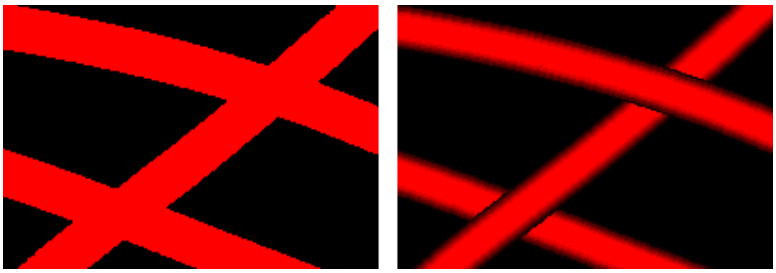

Figure 4. Left: Overlaid streamlines result in an image in which it is difficult to determine distinct streamlines. Right: A textured halo around the streamlines gives each streamline a 3D appearance and also creates discontinuity between overlapping streamlines.

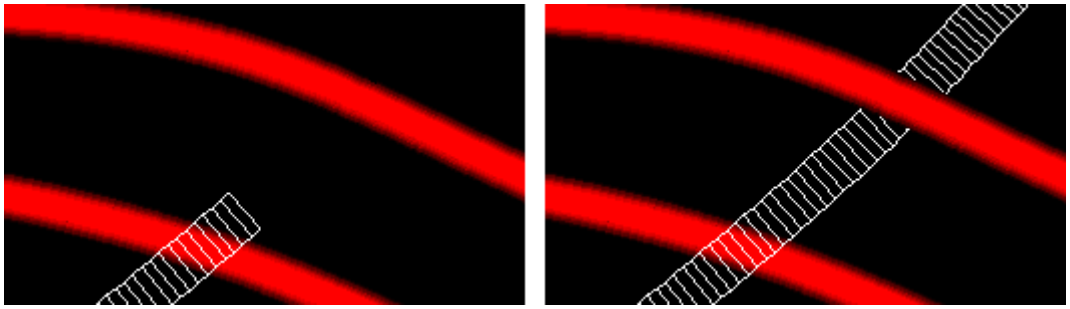

Figure 5. Left: A magnified illustration of how the weaving streamline is constructed. Initially the blending function is set so the streamline will pass over the previously placed streamline (left). When four corners of a polygon are clear from the underlying streamline, the blending function is changed to allow the streamline to pass under the next streamline it encounters (right).

3.3 Algorithm

The effect of streamlines weaving over and under other streamlines from a different vector field is achieved by altering the opacity values of the polygons that constitute the streamline. This method allows for the computational complexity of the algorithm to be minimized as the geometry for intersecting polygons does not have to be explicitly calculated. The intersections are handled through changing the opacity of the streamline as it intersects with the streamlines already placed.

Assuming the seed points for the streamline placement have already been determined by an equally-spaced streamline algorithm, the geometry for the thick streamline can been calculated.[11] The algorithm works by weaving one vector field amongst the streamlines defined by the accompanying vector field. Initially, the streamlines from the first vector field can be placed without the need to consider the second field. The second vector field is introduced one streamline at a time. Each streamline is rendered starting at the furthest point of negative integration and continues forward until it reaches the complete streamline length. At the beginning of each streamline, a random variable determines whether it will begin by passing over or under the first streamline of the opposing vector field it encounters.

If it has been determined that the streamline being rendered is to pass over the first streamline it encounters, the opacity function is set to replace the pixels previously stored in the frame buffer. This blending function is maintained until the streamline being drawn has completely crossed the streamline from the first vector field. This condition is met when the value of the alpha buffer for all four corners of the calculated polygon have opacity values of 0.0. The blending function is then changed to allow for the streamline to pass under the next intersection in a similar fashion (figure 5). This process continues for all streamlines and polygons, alternating between the blending function parameters to allow for the streamlines to appear to be passing over and under each other. The consistent pattern of overlapping streamlines, however, can lead to a visual artifact similar to a checkerboard pattern. To minimize this effect, a random variable determines whether the streamline should pass over or under a previous streamline.

A limitation of the streamline weaving algorithm is that the streamlines must be spaced far enough apart that the four corners of the polygon that constitutes a sliver of the streamline are able to clear the previously placed streamline before it encounters another streamline. In essence, this requires that the streamlines be separated by at least the width of the thick streamline. In our experience, this restriction leads to an effective spacing to allow both vector fields to be represented equally while sufficiently covering the entire domain.

3.4 Artifacts

Standard streamline seeding algorithms only consider a single vector field. The algorithm that determines the placement of the streamlines for each individual vector field is designed to avoid situations in which streamlines become "too close." While effective with individual vector fields, artifacts can arise when applying these techniques by interweaving two equally-spaced streamline representations. Namely, when the placement of the

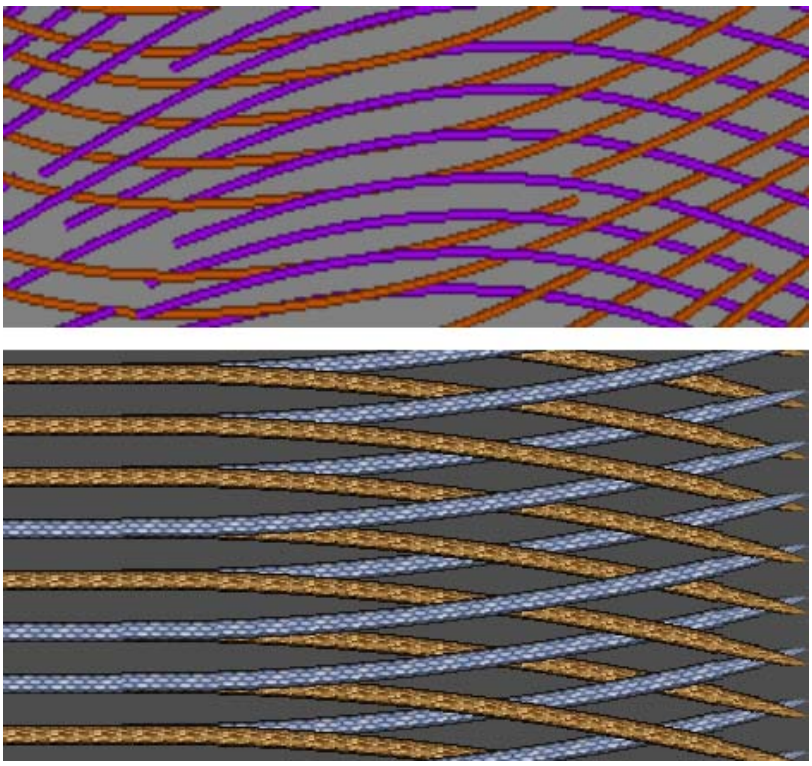

Figure 6. The equally spaced streamline algorithms do not account for a second vector field. This results in the potential for streamlines to overlap in regions where vector fields are parallel causing an undesired clustering artifact.

single-vector-field algorithms are combined, it is possible that streamlines will overlap in areas where the vector fields are parallel. The streamlines become coincident, which results in a clustering artifact (figure 6).

In the next sections, we describe algorithms that consider both vector fields when determining the location for streamlines in order to visually represent both vector fields with equal prominence and minimize clustering artifacts.

4. IMAGE-GUIDED STREAMLINE PLACEMENT

In this section we describe methods based on the Turk and Banks image-guided approach to seeding streamlines.[4] The algorithm works by considering a low-pass filtered version of the streamlines and minimizing a correlated energy function. When streamlines from two vector fields are introduced to the shared domain, each streamline must be assigned to the appropriate field. Care must be taken to ensure that nearly-touching streamlines are from the same vector field prior to joining, as it is possible that two abutting streamlines may not be from the same vector field.

4.1 Original Metric with Two Vector Fields

The quality metric of the Turk and Banks algorithm ensures that streamlines are equally-spaced and a uniform density is maintained throughout the image. This ensures that the entire domain is covered with a similar density of streamlines and no streamlines are "too close" to each other. By introducing a second vector field, it is inevitable that streamlines from different vector fields will cross. Intersecting streamlines, however, induce a high penalty in the metric defined by the low-pass filtered image, as regions of intersection will not have a similar density as regions of non-intersection. Thus, the areas of intersection result in an area of higher density in the low-pass filtered image (figure 7). These regions are above the target density and the intersections are avoided in the final image (figure 8, left).

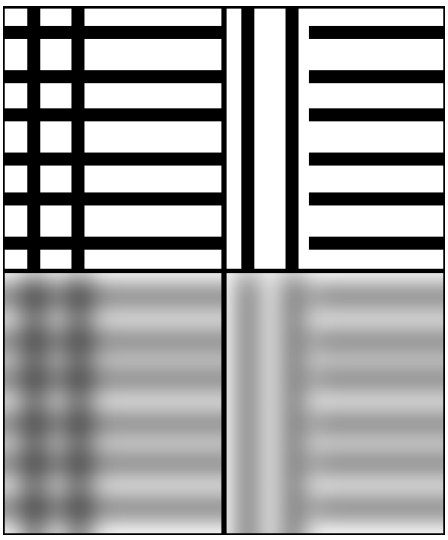

Figure 7. Example of a low-pass filter on two different line patterns. Top: two images of lines. Bottom: a low-pass filter of the above line images. The tendency for the original image-guided streamline placement algorithm to avoid intersecting lines is explained by the even distribution of the low-pass filter on the non-intersecting lines (lower right) compared to the intersecting lines (lower left).

4.2 Improved Metric

As the adaptation of original Turk and Banks technique does not give a representation in which the fields appear *integrated*, we further modify the algorithm to allow the coverage of each individual vector field to contribute to the energy function. This is done by redefining the quality metric to include a low-pass filter image of each of the individual vector fields, in addition to the global set of streamlines.

We initially create three different low-pass images: one that contains all of the streamlines, one that contains streamlines only from the first vector field, and one that contains streamlines from only the second vector field. Each streamline is added to the appropriate two of three low-pass filter images. The density of the low-pass filter of the three images is added, in equal parts, to define the final quality metric. This ensures that areas in which no streamline for a vector field exist are minimized, as this would result in a high penalty from the low-pass image that represents the coverage from that particular vector field. The result is that streamlines are forced to overlap, as desired. The algorithm continues as defined by Turk and Banks and each streamline is placed, altered, and adjusted according to the new metric. The results of the improved algorithm are displayed in figure 8 (right).

5. SEED POINT STREAMLINE PLACEMENT

The equally-spaced streamline approach introduced by Jobard and Lefer provides a computationally efficient method to place streamlines for a single vector field.[2] We extend the Jobard and Lefer algorithm to incorporate two vector fields and minimize the coincident intersections.

5.1 Algorithm

The original algorithm considers two different spacing distances, d_{sep} and d_{test}. d_{sep} is the user-defined distance for the seed points from a calculated streamline. d_{test} is the stopping criteria; it is the distance a new streamline can be within another streamline from the same vector field before it is stopped because it is "too close." Note that in the multiple vector field implementation, the d_{test} criteria is only enforced with streamlines of the same vector field, as intersections between the vector fields will occur naturally and should not be suppressed.

The algorithm begins by selecting a random seed point and constructing a streamline. The algorithm progresses by alternating the vector fields in which the streamlines are drawn. The next candidate seed point is

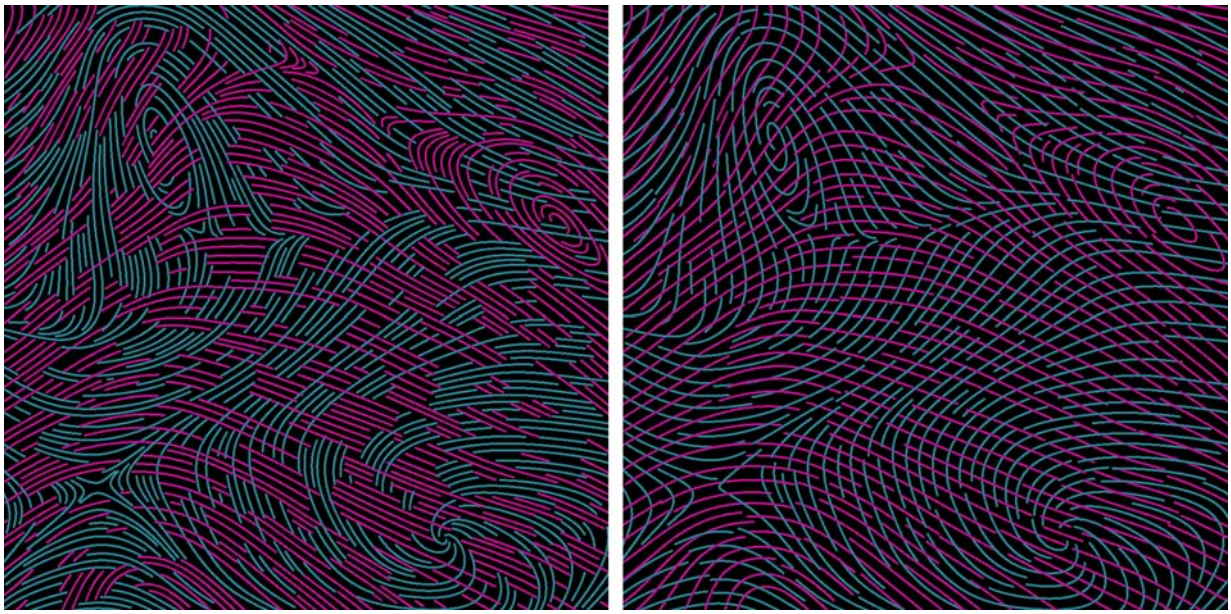

Figure 8. Left: An image depicting two vector fields using the original Turk and Banks energy-minimizing metric. The energy function induces a high penalty for intersecting streamlines resulting in areas where streamlines from only one vector field are present. Right: A representation of two vector fields in which streamlines are placed using the improved quality metric.

selected to be a user-defined distance, d_{sep}, away from the previous streamline drawn. However, unlike the original algorithm (designed for a single vector field), we also consider the second vector field when selecting potential seed points.

As we wish to eliminate the phenomenon of overlapping streamlines when the vector fields are parallel to one another, we scan all points on the previously drawn streamline to find the location in which the two vector fields are most closely aligned. The point in the streamline where the vector fields are the most closely aligned is used to calculate the next seed point at a distance of d_{sep} perpendicular from the streamline. This guarantees that the streamlines of both vector fields will not be coincident at this point, and will avoid a clustering artifact in the final image (figure 9).

The process of computing seed points alternates between the two vector fields and continues until a valid seed point does not exist. Seed points may not be valid for a number of reasons: the seed point is within the distance of d_{sep} of a streamline from the same vector field; the resulting streamline length is below a user-defined tolerance; or the seed point is at a critical point. In the case that a seed point is not valid, the algorithm begins again with a previously computed streamline to find an appropriate seed point. As with the original Jobard and Lefer algorithm, streamlines are drawn in this fashion until no other valid seed points can be found.

5.2 Computational Complexity

The process of determining the most appropriate point along a streamline to use as the next seed is computationally inexpensive. The goal is to minimize the overlapping of streamlines where the two vector fields are parallel. The angle between both vector fields is calculated by evaluating the dot product between the corresponding vectors. Given vector A from the first vector field and the corresponding vector B from the second vector field, the measure of the angle between them, θ, can be calculated as

$$A \cdot B = |A||B| \cos\theta. \tag{2}$$

Note that the vector fields will be aligned when $\theta = 0$ or $\theta = \Pi$. In either case, $\cos\theta = 1.0$. Assuming the vectors are normalized, the location in which the vector fields are the most aligned will result at the location

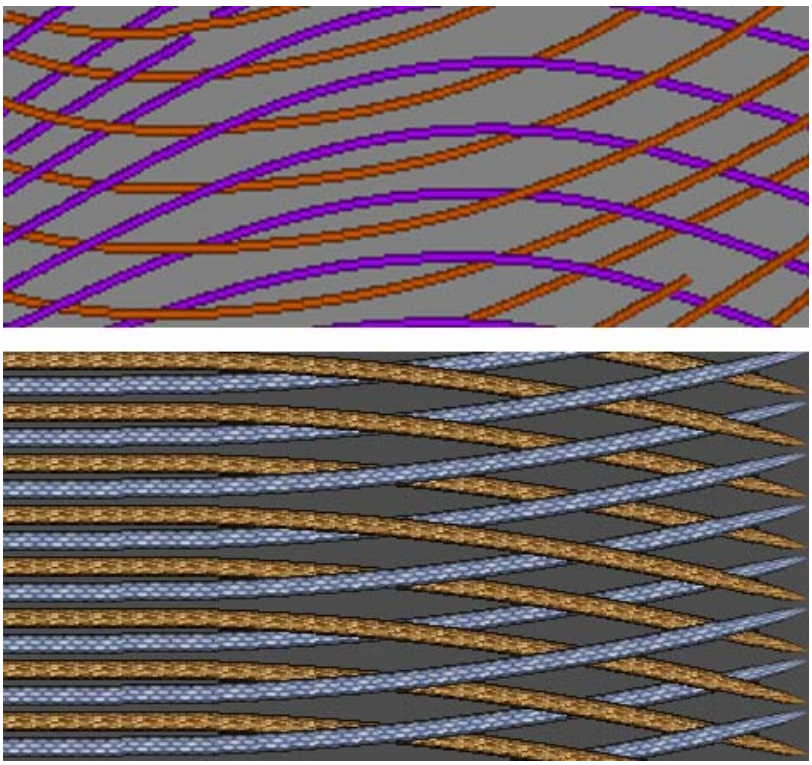

Figure 9. Adaptations to the Jobard and Lefer algorithm allow for effective placement for seed points of two vector fields. Alternating streamlines from different vector fields allows the streamlines to be between other equally-spaced streamlines.

on the streamline where the dot product is minimal. Thus, locating the optimal location on the streamline to calculate the next seed point requires a few additional operations for each point on the streamline to compute the dot product.

In the next section we illustrate the usefulness of the algorithm by visualizing two vector fields that exist within a turbulent boundary layer.

5.3 Application – Turbulent Flow

Turbulent flow is chaotic in nature and is characterized by swirling vortex structures, or eddies, of various size, strength, and orientation. One of the most basic types of turbulent flow is that which occurs when a fluid passes a boundary, such as air over an airplane wing or the flow of oil within a pipeline. The velocity of the flow must match the velocity of the wall at the surface of the boundary. A boundary layer or a turbulent boundary layer is the zone over which the average fluid velocity decreases from freestream value to zero.

In addition to the velocity vector field of the flow ($\overrightarrow{v}$), several other variables can be numerically calculated and are significant to theories related to turbulent flow. For example, the *vorticity* vector field ($\overrightarrow{w} = \nabla \times \overrightarrow{v}$) can be numerically calculated and analyzed with the velocity vector field. The vorticity vector field is a measure of the rotation of fluid flow. Analysis of the vorticity vector field is useful as it can indicate the strength and orientation of the swirling eddies within in the flow.

Particle image velocimetry (PIV) is a technique that can be used to experimentally measure instantaneous components of a velocity field in the plane of a turbulent boundary layer. The corresponding vorticity vector field can be numerically calculated for simultaneous analysis. Figure 10 shows dual-vector field visualization techniques applied to data acquired from a dual plane, particle image velocimetry (PIV) experiment. This image can clearly delineate regions where the velocity and vorticity vectors are parallel (outlined by rectangles) and regions where the vectors are orthogonal (outlined by ovals). Understanding the coincidence of the velocity and

Figure 10. Visualization of multiple vector fields from a dual plane particle image velocimetry experiment. The tan texture mapped streamlines represent the velocity vector field. The light-blue textured streamlines represent the vorticity vector field. The regions within each rectangle depict areas where the vorticity field and the velocity field are parallel. The regions within each oval depict areas where the vorticity field and velocity field are orthogonal. The combination of vector fields within an image allows for a better understanding of the interactions between velocity and vorticity vector fields.

vorticity fields is of interest to investigators, particularly in conjunction with the visualization of vortex core locations.

6. CONCLUSION

The analysis of data that consists of multiple vector fields can be greatly facilitated by a simultaneous visualization of both vector fields. In an effort to better understand the key physical characteristics of coincident vector fields, we have developed several techniques designed to represent both fields accurately – without one field appearing more prominent than the other. These techniques involve streamlines being *interwoven* in a computationally-efficient algorithm that utilizes alterations to classic streamline seeding techniques. The algorithms minimize the artifacts that occur when simply applying streamline seeding strategies to a vector field and overlaying the resulting streamlines by considering where the vector fields are parallel and minimizing the area that the

streamlines may overlap. Lastly, we have showed how this approach can be effective in analyzing the velocity and vorticity vector fields in a turbulent boundary layer from data collected via a PIV experiment.

ACKNOWLEDGMENTS

We would like to acknowledge Ellen Longmire, Ivan Marusic, and Bharathram Ganapathisubramani for their assistance in obtaining and analyzing the data from the PIV experiment. This research was supported by the National Science Foundation (CTS-0324898).

REFERENCES

1. T. Urness, V. Interrante, E. Longmire, I. Marusic, S. O'Neill, and T. W. Jones, "Strategies for the visualization of multiple 2d vector fields," *IEEE Computer Graphics and Applications* **26**(4), pp. 74–82, 2006.
2. B. Jobard and W. Lefer, "The motion map: Efficient computation of steady flow animations," *Proceedings of IEEE Visualization 97* , pp. 323–328, 1997.
3. A. Mebarki, P. Alliez, and O. Devillers, "Farthest point seeding for efficient placement of streamlines.," *Proceedings of IEEE Visualization 2005* , pp. 479–486, 2005.
4. G. Turk and D. Banks, "Image-guided streamline placement," in *Proceedings of SIGGRAPH 96*, pp. 453–460, 1996.
5. V. Verma, D. T. Kao, and A. Pang, "A flow-guided streamline seeding strategy," pp. 163–170, 2000.
6. X. Ye, D. Kao, and A. Pang, "Strategy for seeding 3d streamlines.," *Proceedings of IEEE Visualization 2005* , pp. 471–478, 2005.
7. R. Kirby, H. Marmanis, and D. H. Laidlaw, "Visualizing multivalued data from 2d incompressible flows using concepts from painting," *Proceedings of IEEE Visualization 99* , pp. 333–340, 1999.
8. V. Interrante, "Illustrating surface shape in volume data via pricipal direction-driven 3d line integral convolution," in *Proceedings of SIGGRAPH 97*, pp. 109–116, 1997.
9. C. Weigle, W. Emigh, G. Liu, R. Taylor, J. Enns, and C. Healey, "Oriented sliver textures: A technique for local value estimation of multiple scalar fields," *Graphics Interface* , pp. 163–170, 2000.
10. I. Hotz, L. Feng, B. Hamann, K. Joy, and B. Jeremic, "Physically based methods for tensor field visualization," in *Proceedings of IEEE Visualization 2004*, pp. 123–130, 2004.
11. F. Taponecco, T. Urness, and V. Interrante, "Directional enhancement in texture-based vector field visualization," in *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pp. 197–204, ACM Press, (New York, NY, USA), 2006.
12. D. Dooley and M. F. Cohen, "Automatic illustration of 3d geometric models: Surfaces," in *VIS '90: Proceedings of the 1st conference on Visualization '90*, pp. 307–314, IEEE Computer Society Press, (Los Alamitos, CA, USA), 1990.
13. T. Saito and T. Takahashi, "Comprehensible rendering of 3-d shapes," *Proceedings of SIGGRAPH 90* , pp. 197–206, 1990.
14. V. Interrante and C. Grosch, "Strategies for effectively visualizing 3d flow with volume lic," *Proceedings of IEEE Visualization 97* , pp. 421–424, 1997.