Topology-Preserving Smoothing of Vector Fields

Rüdiger Westermann, Christopher Johnson, *Member*, *IEEE*, and Thomas Ertl, *Member*, *IEEE*

Abstract—In this paper, we propose a technique for topology-preserving smoothing of sampled vector fields. The vector field data is first converted into a scalar representation in which time surfaces implicitly exist as level-sets. We then locally analyze the dynamic behavior of level-sets by placing geometric primitives in the scalar field and by subsequently distorting these primitives with respect to local variations in this field. From the distorted primitives, we calculate the curvature normal and we use the normal magnitude and its direction to separate distinct flow features. Geometrical and topological considerations are then combined to successively smooth dense flow fields, at the same time retaining their topological structure.

Index Terms—Flow visualization.

1 Introduction and Related Work

Visualizing vector field data is challenging because no existing natural representation can visually convey large amounts of three-dimensional directional information. In fluid flow experiments, external materials such as dye, hydrogen bubbles, or heat energy are injected into the flow. The advection of these external materials can create stream lines, streak lines, or path lines to highlight the flow patterns. Analogues to these experimental techniques have been adopted by scientific visualization researchers. Numerical methods and three-dimensional computer graphics techniques have been used to map the local flow characteristics to graphical icons such as arrows, motion particles, stream lines, stream ribbons, and stream tubes, which also provide three-dimensional depth cues. While these techniques are effective in revealing the flow fields' local features, the inherent two-dimensional display of the computer screen and its limited spatial resolution restrict the number of graphical icons that can be displayed at one time.

Additional techniques for flow field visualization include, among others, global imaging techniques. Crawfis and Max [4], [5] proposed direct volume rendering methods to create images of entire vector fields. Here, vector kernels and texture splats are used to construct three-dimensional scalar signals from the vector data. Van Wijk [30] proposed a Spot Noise method using stretched ellipses to create two-dimensional textures that can be mapped onto parametric surfaces. Max et al. [19] further utilized the spot noise method to visualize three-dimensional velocity fields near contour surfaces. Since Cabral and Leedom [2] presented a Line Integral Convolution (LIC) method, which makes use of a one-dimensional low pass filter to convolve an input texture along the principal curves of the vector field, a number of additional related techniques have been proposed. These attempt to optimize the LIC method in terms of computational cost and image quality, to visualize flow over surfaces, and, more recently, to visualize 3D flow in a volume [24], [9], [1], [23], [15], [22].

These methods can successfully illustrate the global behavior of vector fields; however, it is difficult when using such methods to effectively control stream line density in a way that depicts both the direction structure of the flow *and* the flow magnitude. Furthermore, because of the tremendous information density they produce and their inherent occlusion effects, LIC methods generally fail if utilized to globally visualize 3D flow fields.

One approach to overcoming these limitations is to interactively but manually modify the renderable representation in order to highlight the interesting structures [20]. Although visually pleasant results can be achieved using hardware-accelerated 3D texture mapping, in particular for large-scale vector fields it turns out to be rather difficult to pick the relevant structures without explicit knowledge concerning the underlying flow. Despite its inherent interactivity, this approach does not necessarily guarantee that the characteristic flow features will be found.

A different approach is to inspect the flow field in order to detect and analyze critical points [13]. Topological skeletons, which are defined by those stream lines starting at a critical point in the direction of the eigenvectors, are extracted and displayed. Although these techniques provide an effective tool to determine the topological equivalence of different flows, they do not allow an intuitive analysis of the principal streams and their direction of complex flows. First attempts to further simplify the topology by merging close critical points, as presented in [27], are restricted to 2D flows.

- *R. Westermann is with the Scientific Visualization and Imaging Group, Aachen University of Technology, D-52056 Aachen, Germany. E-mail: westermann@sc.rwth-aachen.de.*
- *C. Johnson is with the Scientific Computing and Imaging Institute, School of Computing, Merrill Engineering Building, 50 South Campus Central Dr., Room 3490, University of Utah, Salt Lake City, UT 84112-9205. E-mail: crj@cs.utah.edu.*
- *T. Ertl is with the Institut für Informatik der Universität Stuttgart, Abt. Visualisierung und Interaktive Systeme, Breitwiesenstrasse 20-22, D-70565 Stuttgart, Germany. E-mail: Thomas.Ertl@informatik.uni-stuttgart.de.*

Manuscript received 30 Mar. 2001; accepted 7 May 2001.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 114131.

Other techniques reduce the primitives used to depict the structure of the flow in such a way that the result still represents the original data sufficiently. While, in [28], stream line placement in 2D flows was guided by visual attributes, in [16], evenly spaced stream lines were generated based on a distance criterion. Explicit consideration of the flow topology was used in [32] to optimize the streamline seeding strategy.

On the contrary, the main concern of the work presented in [12], [26], [11], [18] was to effectively simplify the underlying data without loss of relevant information. In general, however, these hierarchical techniques are local, in that they usually consider only the vector field in the geometric neighborhood around each position, but do not take into account the global structure of the flow.

In this paper, we extend our novel approach, presented first in [33], for the analysis and display of stationary vector field data, which includes effective techniques for the classification, segmentation, and topology-preserving smoothing of flow fields. Rather than analyzing the flow field as such, we first convert it into a scalar field. We then analyze the spatial and temporal evolution of level-sets or time surfaces in this field. In particular, we show how to obtain the flow data at ever-coarser resolution by dispersing small disturbances across the time surfaces.

The goal of our approach is twofold: to present a method that allows for the automatic segmentation of flow fields *and* to build an algorithm for the topology-preserving smoothing of vector field data on top of it, by means of which a multiscale representation can be obtained. In contrast to the work in [33], the focus of this work is on developing an efficient scheme for the topology-guided smoothing of flow fields. We considerably improve our previous work in terms of accuracy and efficiency by introducing a novel concept that allows us to accurately measure distortions in the flow.

The reminder of this paper is organized as follows: First, we introduce the basic idea of converting the vector field data into a level-set representation of time surfaces and we describe how the effective exploitation of intermediate results of the numerical integration leads to significant acceleration of the process. We then explain our concept of flow analysis based on stream boundaries which are extracted from the flow by a curvature based analysis of the time surfaces involving an efficient approximation of the curvature normal by means of flow-warp icons. Finally, we propose an explicit scheme for the topology-preserving smoothing of flow fields and we conclude the paper with a detailed discussion of our results.

2 Flow Surfaces

In fluid dynamics, flow surface techniques have become important to the investigation of the dynamics of vector field data. A flow surface can be seen as a variation of path lines in nonstationary flows where several lines are joined to form a surface. A dense set of particles is released into the flow and their subsequent positions, as well as the distortions of the so-defined surfaces, are monitored.

In computational flow visualization, techniques for simulating different kinds of flow surfaces have been developed in the past [14], [29], [3]. In its most general form, it simulates flow surface by placing an initial surface in the flow and then by successively moving all vertices defining the surface within constant intervals along the integral curves of the flow. The integral curves emanating at a given position are the solutions to the ordinary differential equation

$$\frac{d\mathbf{r}(\mathbf{t})}{dt} = \mathbf{v}(\mathbf{r}, t) \tag{1}$$

with initial boundary condition $\mathbf{r}(t_0) = \mathbf{r}_0$. Here, $\mathbf{r}(t)$ denotes the position of a particle at time t, and $\mathbf{v}(\mathbf{r}, t)$ represents the in-stationary velocity field. Notice in particular that this technique can be extended to the stationary case in which a time-independent velocity field is considered and its integral curves are defined with respect to any other parameterization.

In general, flow surfaces can be placed everywhere in the flow; however, without loss of generality, let us assume that particles are initially released at the inflow boundaries and at every source into all possible directions of the velocity at that source. Thus, the evolving surfaces formed by connecting particles at time t_n contain all positions within the domain that can be reached from a source in that time. In other words, any particle that is released from this surface and traverses its integral curve backward will reach a source or the boundary of the domain in that time. We will subsequently call these kinds of particles the *antiparticles*, and the particular kind of flow surface the *time surface*.

A particular time surface in a flow is described by the implicit equation $T(x, y, z) = t_n$, where T is the time the antiparticle released at position (x, y, z) requires to reach a source or the boundary. Therefore, a time surface can be computed either by distorting the initial surface with respect to the flow field or by computing the scalar function T for all necessary positions and by fitting the surface using traditional techniques. The first approach has two major limitations: It requires the generation and display of a huge number of primitives and the generated surfaces are likely to become nonmanifolds including self-intersections. In comparison, the second, implicit approach has various advantages and will be sketched out in the following sections.

2.1 Implicit Time Surfaces

We aim to construct a volumetric representation in which the time surfaces implicitly exist as level-sets [21]. Level-sets in our approach may not exactly be "level-sets" in a strict sense as introduced by Sethian in his original work; however, they are still level sets in the mathematical sense. The level set of f at a particular value of time, t, is the set of all points (x, y, z) such that $f(x, y, z) = t$. Another name for this is the contour curve of f at level t.

Therefore, for each grid point, an antiparticle is released and its integral curve is traced until a source or the boundary of the domain is reached. This procedure is equivalent to the backward tracing approach proposed in [31] for the calculation of stream surface functions. We employ a fourth-order Runge-Kutta scheme in order to find successive points along the curves.

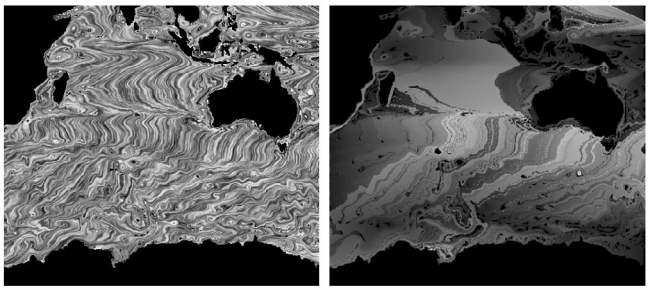


Fig. 1. In the left image, the ocean flow is depicted by means of LIC. On the right, the traveling times of antiparticles to the boundaries or sources are shown as scalar values at each grid point.

In the current implementation, however, the grid is processed stream line by stream line. Once a certain position has been selected to start a new particle line, this line is traced backward until a source or the boundary is reached. During this walk, the positions of all points close to the current line are stored. For each of these points, the time is recorded that was needed to reach the point from the starting position. The time it takes to reach the end of the trace from each of these points is simply the overall time minus the time stored for the point during runtime. In this way, a huge number of points to be processed separately can be saved, which considerably accelerates the generation of the time field.

Prior to this procedure we determine critical points in the flow where the magnitude of the velocity vanishes. Thus, during runtime, we detect antiparticles reaching a critical point. These particles have to be stopped in order to avoid nonterminating traces. Additionally, a stopping criterion is employed for antiparticles, moving on closed orbits, for which no valid but unique time values are assigned.

If the distance from point p_i to point p_{i+1} is d_i, then the time an antiparticle needs to travel from p_i to p_{i+1} is $t_i = d_i \cdot \frac{1}{|\mathbf{v}(p_i)|}$. Integrating the distances along the path yields the time the antiparticle needs to move from the grid point it was released from until it leaves the domain or reaches a critical point. Thus, by storing all times at each grid point, we have converted the vector field into a scalar field, which provides alternatives for display and analysis of the flow (see Fig. 1).

3 Flow Analysis

3.1 Stream Boundaries

As we have claimed in the introduction, our approach should be effective in revealing homogeneous streams in the flow which, in general, cannot be determined by simply analyzing the vector data locally. Even if the vector data is locally homogeneous in terms of direction *and* speed, we will find regions in which different streams proceed parallel to each other over a certain distance, but will be separating again. As a solution to this problem, we have developed a local technique that takes into account global information in that it accumulates flow quantities along the integral curves.

To illustrate the concept, let us interpret particles moving along a path line as a container filled with liquid. At the beginning, each container is empty. At each position in the field, the particle carries all the liquid that was injected along the path line up to this position. The injected amount at each position is equal to the magnitude of the vector data at this position. We then try to extract the exact regions in which adjacent particles carry different amounts of liquid because this implies that the particles have a different history in terms of what they collected along their paths. In the current scenario, the amount of liquid carried by a particle equals the time a particle was traveling along a particle line. The difference between adjacent values on neighboring lines now indicates how much the cumulated matter along the line differs. Consequently, within homogeneous streams, the distortions of level-sets corresponding to equal cumulative amounts are small in general, whereas they are high between different streams. We will subsequently call these kinds of boundaries the *stream boundaries* and we will proceed by studying the curvature of time surfaces by means of which areas where differences occur can be identified.

3.2 Curvature Based Analysis of Time Surfaces

The study of time surfaces is of particular interest because they effectively visualize the geometric and topological modifications of their evolving structures. By helping us to discriminate among areas of flow showing different characteristics, these modifications should allow us to more accurately analyze the flow under consideration. As a consequence, we need to develop a measure for the variations of the flux as specified above that can be used to indicate the presence of stream boundaries.

One approach to detect and characterize "surface features" in geometric modeling is to analyze the local curvature across the surface. Methods for the efficient calculation of geometric attributes on meshes can be found in [8], [25], [17], [7], [6], where this kind of information has been used primarily for the analysis and smoothing of geometric shapes. In the following, we will make use of some of these concepts to analyze the implicit flow surfaces by means of their curvature.

In the present scenario, the curvature of the time surfaces tells us where distortions of these surfaces with respect to the influence of the flow field are most significant. At first glance, one might conclude that the gradient of the scalar time distribution already suffices to locally characterize these distortions. In general, however, a high gradient may point into the flow direction or it may not coincide with the direction though the time surface it belongs to is locally flat, e.g., in a perfect shear flow. Based on our previous remarks on flow boundaries, we are thus interested in finding those positions in the data exhibiting high curvature.

Therefore, however, we first have to derive a method that can be used effectively to compute the curvature at any grid point. In [33], an oracle was proposed that allows for the estimation of the mean curvature at arbitrary points within cells of the scalar time field. The curvature was then computed at randomly selected points in the interior of that cell and the maximum value was used as the curvature measure. Each cell exhibiting high curvature was assumed to belong to a stream boundary that isolates distinct streams from each other.

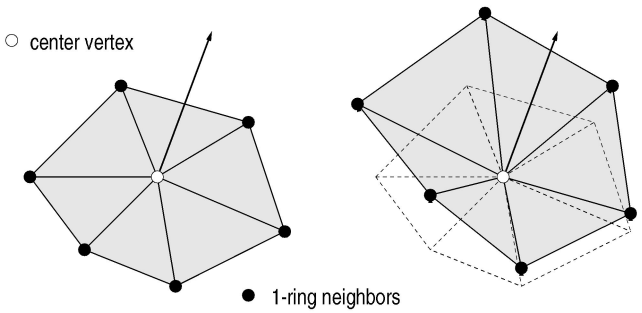


Fig. 2. The flow-warp icon is a planar polygon that is placed in the flow and deformed with respect to the flow direction and to the differences in the time values. The curvature operator is then evaluated on the deformed geometric shape.

Unfortunately, the proposed operator has turned out to be rather impractical for several reasons. First, this measure as such might give improper results since the estimated curvature doesn't take into account the range of data samples in each cell. As a consequence, time surfaces within different cells might have similar curvature, even if the variation of scalar values within the cells differs significantly. Second, the evaluation of this measure is expensive in terms of numerical operations because, at a number of points, the scalar field has to be resampled to compute partial derivatives. Third, computed values might become very small and it has turned out to be rather cumbersome to appropriately scale the values to a range that can be effectively used further on.

In order to circumvent the mentioned drawbacks, we take advantage of a discrete operator well-suited for the estimation of the mean curvature in meshes. Therefore, we first have to obtain the mesh on which this particular geometric attribute can be computed. Because we are only interested in classifying each grid point with respect to the curvature of the surface passing through that point, it suffices to locally reconstruct the time surface in the vicinity of each point. However, instead of reconstructing the surface as it is, we have developed a much more efficient and stable alternative.

Let us proceed by introducing the concept of a *flow-warp* icon. The flow-warp icon is simply a regular, planar polygon consisting of a center vertex and a 1-ring neighborhood, as illustrated in Fig. 2. The number of 1-ring neighbors is chosen in such a way as to allow for the accurate resampling of the scalar field around the center vertex. In our example, the center vertex has valence six and 1-ring neighbors are equally distributed around the center.

At each grid point, a flow-warp icon is positioned in the flow such that the flow vector at this point is orthogonal to the planar polygon. Then, the distortions of the time surface are simulated by distorting the iconal representation correspondingly. At each vertex, the scalar time value is evaluated and, for every 1-ring neighbor, the difference between this value and the value at the center vertex is computed. In addition, at every 1-ring neighbor, the vector field is reconstructed. Every vertex but the center one is then shifted into the direction of the flow. See right of Fig. 2. The strength of the shift is proportional to the difference between the time values. The distortion is thus applied with respect to flow direction and to the variation of the flux.

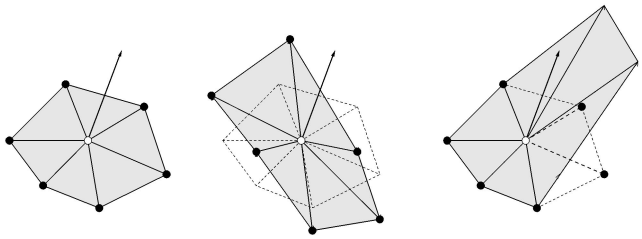


Fig. 3. Different displacements are shown. In the middle image, as a result of opposite displacements, the flow-warp icon remains almost planar, e.g., in a shear flow. On the right, the highly deformed shape indicates 1-ring neighbors belonging to different streams.

One of the nice features of this approach is that it allows us to compute the mean curvature in terms of magnitude and direction. From the deformed icon, we can easily determine the curvature normal, which provides us with an additional attribute to classify grid points. Now, stream boundary cells are characterized by grid points exhibiting high curvature magnitude *and* opposite curvature normals.

For computing the mean curvature normal, we use the following well-known approximation of the Laplacian operator, the so-called scale-independent umbrella operator [10], [7]:

$$L(X_i) = \frac{2}{E} \sum_{j \in N_1(i)} \frac{X_j - X_i}{|e_{ij}|}, \quad with \quad E = \sum_{j \in N_1(i)} |e_{ij}|,$$

where $|e_{ij}|$ specifies the length of the edge connecting vertices X_i and X_j.

In particular, both the location of vertices and the length of the edges connecting 1-ring neighbors with the center vertex enter into this operator. In this way, we keep track of the direction of the applied distortions and its strength. As we would expect, two distortions performed into the same direction but with different length result in different curvature values. On the other hand, if the displacement is performed with equal strength but into different directions, the curvature differs as well. See Fig. 3 for some examples. Flows that proceed parallel to each other but with different speed thus reveal higher curvature than diverging flows having the same variation in speed.

In 2D, things become even simpler because we only have to consider two line segments originating at the current grid point. From the orientation and the length of both segments, the mean curvature can easily be approximated. Note that, again, the mean curvature normal is evaluated, which depends on the direction of the distortions as well as their strengths.

In contrast to calculating the mean curvature as proposed in [33], the current method is superior in several ways.

- It performs much faster because the curvature is approximated by means of the discrete umbrella operator that only involves simple vector operations.
- The accuracy by which the curvature is approximated can be improved using any other operator on discrete meshes, as proposed in [6].

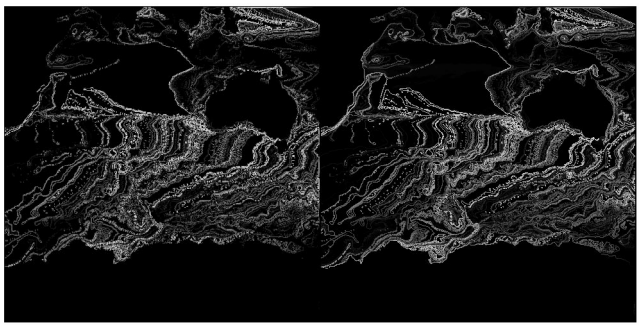


Fig. 4. Two curvature plots that were generated using different methods are shown. On the left, curvature was computed by means of the measure proposed in [33]. On the right, curvature was approximated by evaluating the scale-independent umbrella operator on the deformed flow-warp icons.

- Curvature magnitude and curvature normal allow for a more precise classification of stream boundary cells.
- A conservative bound for the curvature is given by the maximal displacement that can occur. This, however, is restricted by the maximal difference between time values at neighboring points.

In Fig. 4, two different curvature plots of the time field that was generated from the ocean data set are shown. On the left, the method proposed in [33] was employed, while, on the right, the result as computed using the current technique is displayed. As one can immediately see, the technique proposed in this work determines stream boundaries more accurately and is less sensitive to small disturbances in the data.

In any case, however, the curvature plot naturally leads to the discrimination of separate streams that flow in different directions and/or with different speeds. In laminar streams where the distortions of the time surfaces are low, the curvature will be low as well. In the next section, we will demonstrate how to use this information to successively smooth flow fields at the same time revealing the flow topology.

4 Topology-Preserving Smoothing of Flow Fields

In the following, our goal is to develop a technique that allows us to successively smooth the flow field with respect to the curvature of the time surfaces. Separate streams should not be merged, whereas small deviations between vector values within them should be removed.

4.1 Iterative Smoothing Scheme

So far, smoothing schemes for vector data almost entirely rely on the local evaluation of the flow field. The variation of the flow direction between adjacent grid points is considered, but the smoothing operator doesn't account for accumulated flow quantities as proposed in our approach. In [6], for instance, an anisotropic smoothing scheme for vector fields has been proposed that diffuses small deviations into the direction of homogeneous vector components.

However, also in this approach, the local smoothing process may result in the dispersion of disturbances across different streams as well as along the stream lines. Neither effect is suitable since both lead to undesirable smoothing orthogonal to the streams and equally undesirable distortion of the stream lines' main shape. In order to avoid these drawbacks, we have incorporated the local curvature of the accumulated flow quantity, the time-surfaces, into the smoothing process.

We start with a Cartesian grid and the initial distribution of the function values $T(x, y, z)$ at each grid point. We then subsequently visit each grid point and the vector data is smoothed by averaging the contributions from all 26 neighbors around that point:

$$V_i = \sum_{j \in N(j)} w_j \cdot V_j; \quad w_j = \begin{cases} 0 & : \ i \neq j \quad \& \\ & : \ L_j < \epsilon \quad \& \quad op(j,i) \\ \tilde{w}_j & : \ otherwise. \end{cases}$$

Here, $N(j)$ includes the point itself, $\tilde{w}_j$ specifies precomputed weights that are stored in a $3 \times 3 \times 3$ filter mask, L_j is the curvature magnitude, and $op(j, i)$ returns True if both curvature normals at N_j and N_i are opposite to each other and False otherwise.

At the current grid point and all of its neighbors, we place flow-warp icons into the stream. The distortions of these icons are simulated as described and the curvature normals are approximated by means of the scale-independent umbrella operator. Only those neighbors that belong to the same stream are considered in the smoothing process. All others indicated by high curvature and opposite curvature normal will be ignored. Note that vector samples at grid points belonging to closed orbits will only be averaged with other samples in the same orbit. These are classified by unique ids which were ascertained in the preprocessing step.

5 Results and Analysis

In this section, we show some results of our approach and we analyze the main modules and features of our system. All tests were run on an SGI Octane equipped with one R12000 400 MHz processor and 256 MB main memory. All of our tests were executed on 2D or 3D Cartesian grids, but, also, other grid types can be processed with only slight modifications. In particular, the particle tracer has to be modified appropriately and an algorithm is required that allows for the resampling of the vector data and the time distribution. Then, the deformation of flow-warps can be computed straightforwardly with the only modification that the smoothing process incooperates inverse distance weights with respect to the length of the edges between grid points rather than precomputed weights.

As already stated in [33], the most time consuming element of the presented approach is the computation of the time distribution T, which is accomplished by tracing the integral curves back in time until a source or the inflow boundaries are reached. The previous system implemented the Runge-Kutta scheme in a straightforward way without taking advantage of coherence in the data. The naive approach, in which every particle trace is computed from

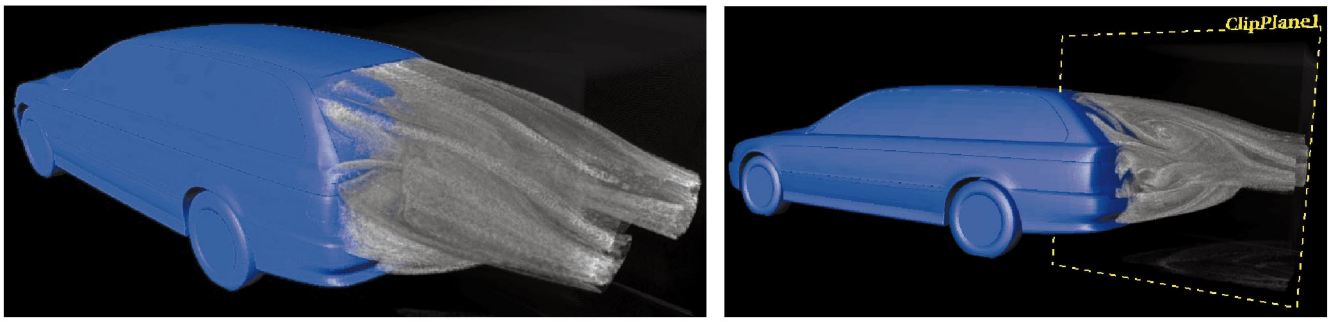


Fig. 5. Both images show the curvature volume computed from the flow field around the back of the car. Dark gray indicates high curvature. The relevant structures can be clearly distinguished even without any manual modifications.

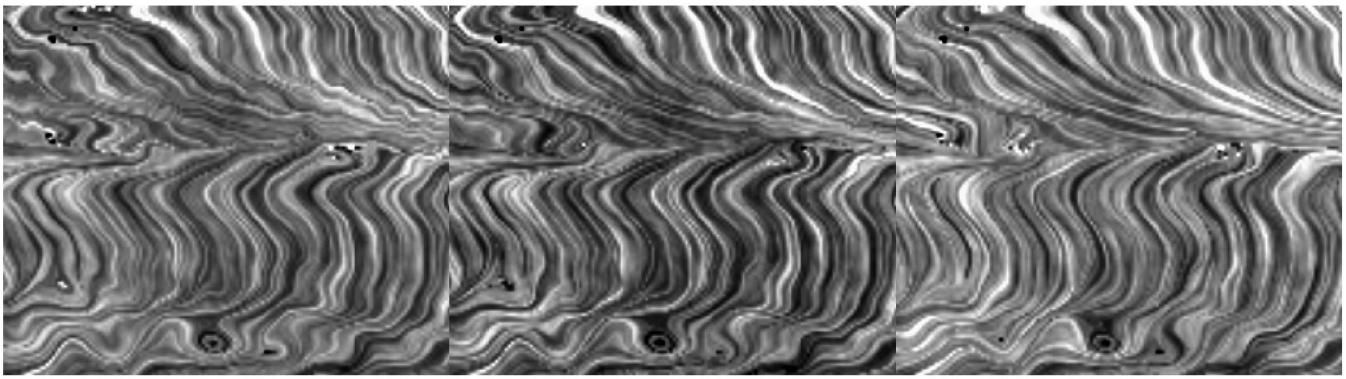


Fig. 6. In this sequence, the ocean data set is continuously smoothed using the proposed iteration scheme. In the middle image, the result after two iterations is shown. Five iterations were performed in the rightmost image.

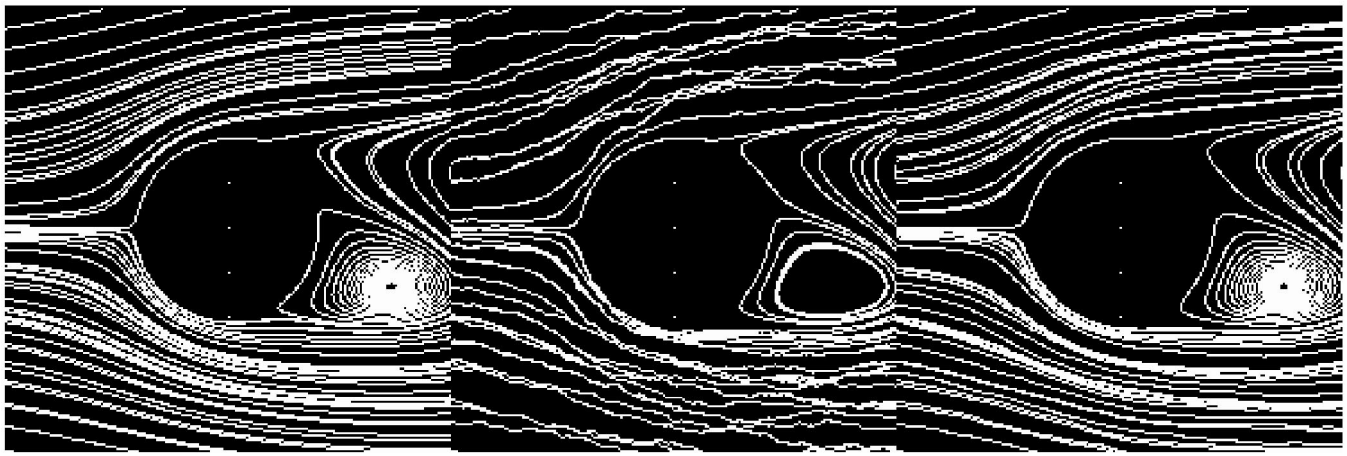


Fig. 7. The vector field showing the flow around a cylinder (left) was modified by adding noise (middle). On the right, the data was smoothed using three iterations.

scratch, takes roughly 28 minutes for the 256^3 flow data set shown in Fig. 5.

As proposed earlier in the current implementation, the grid is processed stream-line by stream-line in order to geneate the time field. In this way, the proposed method could be improved considerably. This also makes it suitable for the processing of large-scale 3D vector data. For example, this acceleration technique could reduce to 5 minutes the time needed to process the aforementioned example. For the same example, computing the curvature by means of flow-warp icons and smoothing the flow field took approximately 48 seconds.

So far, the proposed multiscale representation enables us to remove noise from flow fields and to generate copies of the original flow at ever coarser resolution. Fig. 6 and Fig. 7 show different data sets after some iterations were performed. One can easily recognize that the main shape of the stream lines is retained as high frequency oscillations are successively removed. In particular, random noise was added to some of the vector fields and the original data was compared to the results before and after smoothing. As can be clearly seen from the images, the original data set can be recovered by our method without any noticeable artifacts.

As we have pointed out, our technique is intended to extract stream boundaries based on the proposed discrete

curvature criterion. As a matter of fact, the classification of stream boundaries and, consequently, the smoothing process strongly relies on the curvature threshold we select as the importance measure. The specification of a proper error tolerance raises the same intrinsic problem as in other areas where techniques attempt to discriminate noise from features. On the other hand, the incooperation of the curvature direction as an additional importance measure considerably improves the accuracy of this process.

6 Conclusion

In this work, we have emphasized a general approach for the topology-preserving smoothing of flow fields by means of the dynamics of time surfaces. The major contribution here is to consider time surfaces within this field as the fundamental structures showing the dynamics of the flow. The evolution of these level-sets in space and time is analyzed in terms of their curvature normal. This technique enables us to separate homogeneous streams from each other.

Although the discrete curvature is locally investigated, it gives a global measure because a point on the time surface carries information along the entire stream up to the current position. Consequently, noise along the stream lines is increasingly removed due to integration, while turbulence introduces high frequency oscillations as significant changes to flow direction alter the stream lines' main shape.

We introduced an explicit scheme to iteratively smooth flow fields. In particular, we have shown how to remove noise from vector data by dispersing small disturbances within separate streams but not across them. As a matter of fact, the integral curves' main shapes can be retained. Even if highly turbulent sections are present in the data to the extent that no regular stream boundaries can be detected, our approach is able to detect these regions and smoothing is not performed.

In contrast to the work presented in [33], we introduced two major extensions: the efficient approximation of the curvature normal by means of flow-warp icons as the fundamental primitives and the acceleration of the integration process due to the effective exploitation of intermediate results. Both extensions have led to improved accuracy of the classification procedure and to a considerable acceleration of the entire smoothing process.

Acknowledgments

We would like to thank H. Hagen for his valuable reference to the curvature normal. We also thank NASA for providing the ocean data set. This work was supported, in part, by grants from the US National Science Foundation, National Institutes of Health, and Department of Energy.

References

[1] H. Battke, D. Stalling, and H.-C. Hege, "Fast Line Integral Convolution for Arbitrary Surfaces in 3D," *Visualization and Math.,* pages 181-195, Springer, 1997.

[2] B. Cabral and C. Leedom, "Imaging Vector Fields Using Lne Integral Convolution," *Computer Graphics (SIGGRAPH '93 Proc.),* pp. 263-270, 1993.

[3] W. Cai and P.-A. Heng, "Principal Stream Surfaces," *Proc. IEEE Visualization '93,* pp. 75-80, 1997.

[4] R. Crawfis and N. Max, "Direct Volume Visualization of Three-Dimensional Vector Fields," *Proc. ACM Workshop Volume Visualization,* pp. 55-60, 1992.

[5] R. Crawfis and N. Max, "Texture Splats for 3D Scalar and Vector Field Visualization," *Proc. IEEE Visualization '93,* pp. 261-265, 1993.

[6] M. Desbrun, M. Meyer, P. Schroeder, and A. Baar, "Discrete Differential-Geometry Operators in ND," technical report, California Inst. of Technology, 2000.

[7] M. Desbrun, M. Meyer, P. Schröoder, and A. Barr, "Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow," *Computer Graphics (SIGGRAPH '99 Proc.),* pp. 317-324, 1999.

[8] G. Farin, *Curves and Surfaces for CAGD,* third ed. Academic Press, 1993.

[9] L.K. Forssell and S.D. Cohen, "Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation, and Unsteady Flows," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 2, pp. 133-141, 1995.

[10] K. Fujiwara, "Eigenvalues of Laplacians on a Closed Riemannian Manifold and Its Nets," *Proc. AMS 123,* pp. 2585-2594, 1995.

[11] H. Garke, T. Preussner, M. Rumpf, A. Telea, U. Weikard, and J. van Wijk, "A Continuous Clustering Method for Vector Fields," *Proc. Visualization '00,* pp. 351-358, 2000.

[12] B. Heckel, G. Weber, B. Hamann, and K. Joy, "Construction of Vector Field Hierarchies," *Proc. IEEE Visualization '99,* pp. 19-27, 1999.

[13] L. Hesselink and T. Delmarcelle, "Visualization of Vector and Tensor Data Sets," *Scientific Visualization—Advances and Challenges,* pp. 367-390, Academic Press, 1994.

[14] L. Hultquist, "Constructing Stream Surfaces in Steady 3D Vector Fields," *Proc. IEEE Visualization '92,* pp. 171-177, 1992.

[15] V. Interrante and C. Grosch, "Strategies for Effectively Visualizing 3D Flow with Volume LIC," *Proc.s IEEE Visualization '97,* pp. 421-425, 1997.

[16] B. Jobard and W. Lefer, "Creating Evenly-Spaced Streamlines of Arbitrary Density," *Proc. EG-ViSC '97,* pp. 102-107, 1997.

[17] L. Kobbelt, "Discrete Fairing," *Proc. IMA Conf. Math. of Surfaces,* pp. 101-131, 1997.

[18] S. Lodha, J. Renteria, and K. Roskin, "Topology Preserving Compression of 2D Vector Fields," *Proc. Visualization '00,* pp. 343-350, 2000.

[19] N. Max, R. Crawfis, and C. Grant, "Visualizing 3D Velocity Fields Near Contour Surfaces," *Proc. IEEE Visualization '94,* pp. 248-255, 1994.

[20] C. Rezk-Salama, P. Hastreiter, and T. Ertl, "Interactive Exploration of Volume Line Integral Convolution Based on 3D-Texture Mapping," *Proc. IEEE Visualization '99,* pp. 233-240, 1999.

[21] J.A. Setian, *Level Set Methods and Fast Marching Methods.* Cambridge Univ., Cambridge, U.K., 1999.

[22] H.-W. Shen and D. Kao, "Uflic: A Line Integral Convolution Algorithm for Visualizing Unsteady Flows," *Proc. IEEE Visualization '97,* pp. 317-323, 1997.

[23] H.W. Shen, K.L. Ma, and C.R. Johnson, "Global and Local Vector Field Visualization Using Enhanced Line Integral Convolution," *Proc. ACM Symp. Volume Visualization,* pp. 63-70, 1996.

[24] D. Stalling and H.-C. Hege, "Fast and Resolution Independent Line Integral Convolution," *Computer Graphics (SIGGRAPH '95 Proc.),* pp. 249-256, 1995.

[25] G. Taubin, "A Signal Processing Approach to Fair Surface Design," *Computer Graphics (SIGGRAPH '95 Proc.),* pp. 351-358, 1995.

[26] A. Telea and J. Wijk, "Simplified Representation of Vector Fields," *Proc. IEEE Visualization '99,* pp. 35-43, 1999.

[27] X. Tricoche, G. Scheuermann, and H. Hagen, "A Topology Simplification Method for 2D Vector Fields," *Proc. Visualization '00,* pp. 359-366, 2000.

[28] G. Turk and D. Banks, "Image-Guided Streamline Placement," *Computer Graphics (SIGGRAPH '96 Proc.),* pp. 453-460, 1996.

[29] J. van Wijk, "Implicit Stream Surfaces," *Proc. IEEE Visualization '93,* pp. 245-260. 1993.

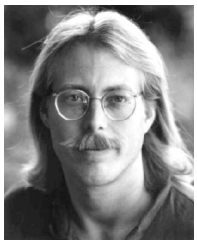
[30] J.J. van Wijk, "Spot Noise: Texture Synthesis for Data Visualization," *Computer Graphics,* vol. 25, no. 4, pp. 309-318, 1991.

[31] J.J. van Wijk, "Implicit Stream Surfaces," *Proc. IEEE Visualization '93,* pp. 245-253, 1993.

[32] V. Verma, D. Kao, and A. Pang, "A Flow-Guided Streamline Seeding Strategy," *Proc. Visualization '00,* pp. 163-170, 2000.

[33] R. Westermann, C. Johnson, and T. Ertl, "A Level-Set Approach for Flow Visualization," *Proc. IEEE Visualization '00,* pp. 147-155, 2000.

Rüdiger Westermann pursued his doctoral thesis on multiresolution techniques in volume rendering and he received the PhD degree in computer science from the University of Dortmund in Germany. He is a professor of computer science at the University of Technology Aachen. He is the head of the Scientific Visualization and Imaging Group. His research interests include hierarchical methods in scientific visualization, volume rendering of structured and unstructured grids, hardware accelerated image synthesis, flow visualization, and parallel graphics algorithms. He was a visiting professor at the University of Utah in Salt Lake City.



Christopher Johnson directs the Scientific Computing and Imaging Institute (www.sci.utah.edu) at the University of Utah and holds faculty appointments in the School of Computing and Departments of Physics and Bioengineering. His research interests are in the area of scientific computing. Particular interests include inverse and imaging problems, adaptive methods for partial differential equations, problem solving environments, large scale computational problems in medicine, and scientific visualization. He is a member of the IEEE.

Thomas Ertl received the master's degree in computer science from the University of Colorado at Boulder and the PhD degree in theoretical astrophysics from the University of Tuebingen. Currently, Dr. Ertl is a full professor of computer science at the University of Stuttgart, Germany, and the head of the Visualization and Interactive Systems (VIS) group within the Institut for Informatics. Prior to that, he was a professor of computer graphics and visualization at the University of Erlangen, where he led the Scientific Visualization Group. His research interests include visualization, computer graphics, and human computer interaction in general, with a focus on volume rendering, flow visualization, multiresolution analysis, parallel and hardware accelerated graphics, large datasets, and interactive steering.