Flow Visualization with Quantified Spatial and Temporal Errors Using Edge Maps

Harsh Bhatia, *Student Member*, *IEEE*, Shreeraj Jadhav, *Student Member*, *IEEE*, Peer-Timo Bremer, *Member*, *IEEE*, Guoning Chen, *Member*, *IEEE*, Joshua A. Levine, *Member*, *IEEE*, Luis Gustavo Nonato, *Member*, *IEEE*, and Valerio Pascucci, *Member*, *IEEE*

Abstract—Robust analysis of vector fields has been established as an important tool for deriving insights from the complex systems these fields model. Traditional analysis and visualization techniques rely primarily on computing streamlines through numerical integration. The inherent numerical errors of such approaches are usually ignored, leading to inconsistencies that cause unreliable visualizations and can ultimately prevent in-depth analysis. We propose a new representation for vector fields on surfaces that replaces numerical integration through triangles with maps from the triangle boundaries to themselves. This representation, called *edge maps*, permits a concise description of flow behaviors and is equivalent to computing all possible streamlines at a user defined error threshold. Independent of this error streamlines computed using edge maps are guaranteed to be consistent up to floating point precision, enabling the stable extraction of features such as the topological skeleton. Furthermore, our representation explicitly stores spatial and temporal errors which we use to produce more informative visualizations. This work describes the construction of edge maps, the error quantification, and a refinement procedure to adhere to a user defined error bound. Finally, we introduce new visualizations using the additional information provided by edge maps to indicate the uncertainty involved in computing streamlines and topological structures.

Index Terms—Vector fields, error quantification, edge maps.

1 Motivations

Vector fields are a common form of simulation data appearing in a wide variety of applications ranging from computational fluid dynamics (CFD) and weather prediction to engineering design. Visualizing and analyzing the flow behavior of these fields can help provide critical insights into simulated physical processes. However, achieving a consistent and rigorous interpretation of vector fields is difficult, in part because traditional numerical techniques for integration do not preserve the expected invariants of vector fields.

To better understand this challenge inherent in traditional numerical techniques, we reconsider the most common way to store vector fields. Both a discretization of the domain of the field (often in the form of a triangulated mesh) as well as a set of sample vectors (defined at the vertices of the mesh) are required. The vector field on the interior of a triangle is approximated by interpolating vector values from the samples at the triangle's corners. Subsequently, computing properties that require integrating these vector values presents a significant computational challenge. For example, consider computing the flow paths (streamlines) of massless particles that travel using the instantaneous velocity defined by the field. Naive integration techniques may violate the property that every two of these paths are expected to be pairwise disjoint (i.e., the uniqueness of the solution of an ordinary differential equation). Fig. 1 gives one such example, where a fourth-order Runge-Kutta integration technique creates two crossing streamlines.

Despite these problems, many of the standard techniques used for vector fields rely on variants of Runge-Kutta methods. Consequently, robustly computing flow becomes a formidable task. Integration is confounded by numerical errors at each step, in particular near unstable regions where the flow bifurcates or spirals slowly. These errors can compound quickly to produce inconsistent views of the vector field, resulting in inaccurate visualization and analysis of the field.

Apart from the obvious problem of potentially including an unknown structural error in the analysis, traditional techniques can cause a more subtle yet equally important problem. By hiding the errors inherent in the numerical integration these techniques create the perception of certainty. The user is presented with crisp lines and clean segmentations which imply a false level of accuracy. Instead, a more nuanced approach that clearly indicates which information is known and where possible instabilities might arise would provide a more candid view of the data.

- *H. Bhatia, S. Jadhav, G. Chen, J.A. Levine, and V. Pascucci are with the Scientific Computing and Imaging Institute (SCI), University of Utah, 72 S Central Campus Drive, Room 3750, Salt Lake City, UT 84112. E-mail: {hbhatia, jadhav, chengu, jlevine, pascucci}@sci.utah.edu.*
- *P.-T. Bremer is with the Center of Applied Scientific Computing, Lawrence Livermore National Laboratory, Box 808, L-422, Livermore, CA 94551-0808. E-mail: bremer5@llnl.gov.*
- *L.G. Nonato is with the Depto de Matemática Aplicada, Instituto de Cincias Matemáticas e de Computação, Universidade de São Paulo, Av. Trabalhador São-carlense 400, São Carlos 13560-970, SP, Brazil. E-mail: gnonato@icmc.usp.br.*

Manuscript received 10 June 2011; revised 21 Sept. 2011; accepted 5 Oct. 2011; published online 14 Oct. 2011.
Recommended for acceptance by G. Di Battista, J.-D. Fekete, and H. Qu.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2011-06-0125.
Digital Object Identifier no. 10.1109/TVCG.2011.265.

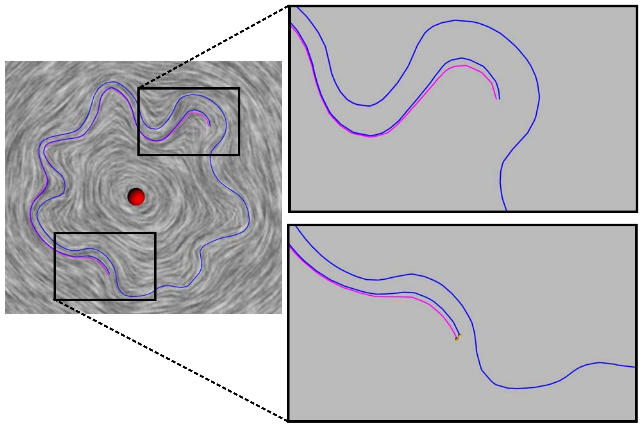


Fig. 1. Left: Two streamlines are seeded traveling clockwise around this sink (red ball) in a domain $[-1, 1] \times [-1, 1]$. Right bottom: Initially, the magenta streamline is seeded outside of the blue streamline with respect to the center of the domain. Right top: After integration with a step size of 0.025 the streamlines cross, now the magenta streamline is inside the blue streamline with respect to the center.

Considering these motivations, we propose a new data structure, called *edge maps*, to represent vector fields. Edge maps provide an explicit representation of flow by mapping entry and exit points of flow paths on the edges of the triangle. Thus, they encode the property most often needed by common analysis tools to compute visualizations and topological decompositions. We show how to compute many of the same primitives robustly and directly on the edge maps themselves. Moreover, the edge map data structure encodes numerical error, presenting a more complete view of the data illuminating the major features that demonstrate where numerically unstable regions exist. This encoding enables refinement of the maps to bound the amount of error incurred by this representation.

While a method is required to compute the initial flow within each triangle, any subsequent computation assumes the edge map to be ground truth. Such a strategy is akin to recent techniques that robustly compute scalar field topology. Gyulassy et al. [11], for example, convert a scalar field into a discrete gradient from which global properties such as the topology can be extracted consistently. In both scalar and vector fields the initial conversion can create discretization artifacts. However, the net gain is significant. Using edge maps, we can accumulate the error while performing computation. Where discretization artifacts have occurred, we show these unavoidable errors explicitly to the user. Consequently, instead of providing a black box representation of the data that ignores the impact of discretization, we can provide analysts a visualization of the data that accounts for these artifacts and indicates how errors may have affected the apparent flow behavior.

1.1 Contributions

This research focuses on a new representation of vector fields, called edge maps, aiming at encoding the flow behavior with a bounded error. Edge maps encode the inflow/outflow behavior over the boundary of a triangle, allowing the replacement of integration with a simple map lookup. The analysis and visualizations generated using edge maps are guaranteed to be consistent up to floating point precision, as a byproduct of the removal of numerical integration of streamlines.

A preliminary discussion on edge maps without a notion of time appeared in [1], and a discussion on the mathematical properties of edge maps, a proof of their consistency, and a list of the possible configurations of flow within each triangle can be found in [17]. In the current paper, we discuss edge maps, and explain how edge maps enable consistent visualization of flow augmented by visualization of errors. The main contributions of this work are:

- The definition of edge maps, with time, for vector fields defined on triangulated surfaces, and an algorithm to compute their approximation;
- Quantification of both spatial and temporal error bounds due to this approximation;
- A refinement procedure for reducing both spatial and temporal mapping error; and
- New visualizations of flow that highlight instabilities by showing manifestations of these errors in space and time.

2 Related Work

Since vector-valued data are a natural way to represent fluid flow in simulations as well as other dynamical systems [15], analyzing vector fields has received a significant amount of attention in the visualization community. In addition, computer graphics researchers have used vector fields for applications ranging from texture synthesis and nonphotorealistic rendering [6], [44] to mesh generation [2], [30]. Regardless of the application, there is a universal need to represent large, complex fields concisely. A reliable visualization must encode the important features of the field and ensure that the methods used do not create contradictory views.

Kipfer et al. [20], following the lead of Nielson and Jung [26], propose a local exact method (LEM) to trace a particle on linearly interpolated vector fields defined on unstructured grids. LEM solves an ODE representing the position of the particle as a function of time, starting at a given position. Consequently, it removes the need to perform stepwise numerical integration, and hence is free from the cumulative integration error and is as accurate as numerical precision. Given an entry point of a particle to a simplex, LEM gives its exit point from the simplex. We use this exact method during the construction of edge maps, which removes the need for on-the-fly numerical integration.

Consistency is particularly desirable when computing structural properties of vector fields. Helman and Hesselink [14] compute a vector field's topological skeleton by segmenting the domain of the field using streamlines traced from each saddle of the field along its eigenvector directions. The nodes of the skeleton are critical points of the vector field and streamlines that connect them are called separatrices. Subsequently, the skeleton extraction has been extended to include periodic orbits [42]. By detecting closed streamlines, Wischgoll and Scheuermann [42] propose a technique to detect limit cycles in planar vector fields. Scheuermann et al. [34] look for the areas of nonlinear behavior in the field, and use higher order methods to preserve such features, which otherwise would have been destroyed due to linear assumption. Three dimensional variants of the topological skeleton have also been proposed [10], [18], [38], [41]. The readers should refer to [22], [35] for more detailed surveys.

However, it is well known that computing the topological skeleton can be numerically unstable due to errors inherent in the integration of separatrices and inconsistencies among neighboring triangles [9], [16], [26], [33]. As a result, some of the fundamental topological invariants of a vector field may not be preserved, such as the Poincaré-Hopf index formula or the fact that streamlines are pairwise disjoint. Consequently, computing the topological skeleton numerically is adequate for visualizing the resulting structures but less suitable for further analysis. A number of techniques have been proposed to extract the topological skeleton in a stable and efficient manner. Chen et al. [3] introduce the Entity Connection Graph (ECG) as a more complete topological representation of vector fields on piecewise linear (PL) manifolds. They further introduced the Morse Connection Graph (MCG) as a more robust representation of vector field topology [4] based on *Morse decomposition*. Recently, Szymczak and Zhang [37] propose an algorithm to compute the Morse decomposition for piecewise constant (PC) vector fields by representing the set of all trajectories in the field as a finite transition graph. Szymczak [36] extends this technique to produce a superset of transition graphs to reflect the error in the field. While the PC field is created from the original piecewise linear field, there is no guarantee that the resulting Morse decomposition reflects the original vector field.

Recent work of Reininghaus and Hotz [31] construct a combinatorial vector field based on Forman's discrete Morse theory [8]. Using combinatorial fields allows the extraction of a consistent topological structure. However, the application of the original solve for the combinatorial vector fields is limited by the high computational complexity, leading to later improvements to the algorithm [32]. While provably consistent, it is unclear how close the resulting combinatorial field is to the original field. By comparison, this work proposes an integration technique that is both consistent and has a bounded error with respect to the LEM.

A large section of visualization community concentrates on error and uncertainty visualization [19]. Pang et al. [29] identified three sources of uncertainty: 1) Uncertainty in data, 2) Uncertainty due to derivation, 3) Uncertainty due to visualization. The earlier work on vector field uncertainty visualization concentrated on visualizing extended glyphs for vector samples to represent uncertainty [29], [43] and comparisons of streamlines using different integration schemes, step sizes, and data resolutions [40]. Recently, Otto et al. described a method to obtain uncertain topological segmentations by sampling in a random 2D vector field [27], and later extending it to 3D vector fields [28]. While their method tries to simulate the uncertainty in data probabilistically, disregarding the uncertainty due to the computation and visualization itself, the edge maps assume the given data as certain, and enable visualization of uncertainty introduced during computation and visualization only.

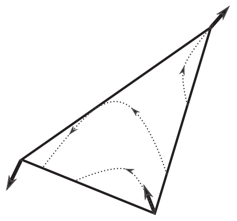


Fig. 2. Flow through a triangle with vectors defined on its vertices.

3 Edge Maps

In the following, we define edge maps, describe their construction, and discuss how edge maps can be used for a consistent propagation.

3.1 Foundations

Let $\vec{V}$ be a tangential vector field defined on a 2-manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$. $\vec{V}$ is represented as the set of vector values sampled at the vertices of a triangulation of $\mathcal{M}$. Specifically, each vertex p_i has the vector value $\vec{V}(p_i)$ associated with it. The vector values on the interior of each triangle Δ with vertices $\{p_i, p_j, p_k\}$ in the triangulation are interpolated linearly using $\vec{V}(p_i), \vec{V}(p_j), \vec{V}(p_k)$. Fig. 2 depicts the field defined in this way for a single triangle. Note that we require that these three vectors reside in the plane of the triangle. A suitable projection might be needed for this, while ensuring that the vector field is consistent across edges.[1]

Given a vector field $\vec{V}$, we can define the *flow* $x(t)$ of $\vec{V}$. Treating $\vec{V}$ as a velocity field, the flow describes the parametric path that a massless particle travels according to the instantaneous velocity defined by $\vec{V}$. $x(t)$ can be defined as the solution of the differential equation

$$\frac{dx(t)}{dt} = \vec{V}(x(t)).$$

The path $x(t)$ with $x(0) = x_0$ is called the *streamline* starting at x_0. Since generally, the analytic form of the vector field is unavailable, solving this differential equation for a single streamline is typically accomplished using numerical integration such as Euler or Runge-Kutta methods.

For a PL vector field defined by the three vector samples at the vertices of a triangle, we begin with assuming that: 1) the vectors at all the vertices of the triangle are nonzero, 2) the vectors at any two vertices sharing an edge are not antiparallel, and 3) the vectors at two vertices on an edge e are not both parallel to e. Any such configuration violating one of these conditions is unstable, and can be avoided by a slight perturbation. This perturbation ensures that no critical point lies on the boundary of the triangle and there is no streamline along any edge of the triangle, which significantly simplifies the analysis of edge maps, as will be clear in the following sections.

1. For each vertex p_i, we use mean value coordinates in 3D [7] to project the star of p_i, $\mathcal{S}(p_i)$ onto a 2D plane. The vector $V(p_i)$ is then projected onto every triangle in $\mathcal{S}(p_i)$, and an average vector is computed in 2D. The inverse projection (to 3D) of the average vector is then computed for every triangle in $\mathcal{S}(p_i)$ giving a consistent representation of $V(p_i)$.

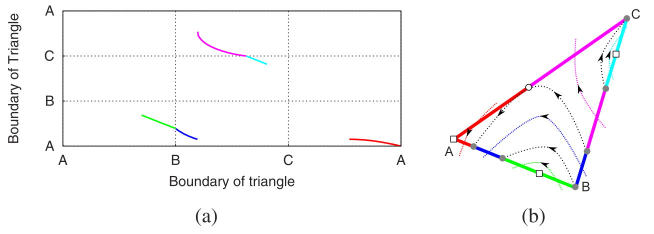


Fig. 3. Edge map for the triangle from Fig. 2. (a) The edge map visualized as a plot, mapping the origin points (horizontal axis) to destination points (vertical axis). Each point on the plot represents a streamline between an o-d pair. Since the edge map is only defined for origin points on the boundary, the plot is disconnected at the points where the boundary switches from inflow to outflow, or vice versa. (b) Edge maps subdivide the boundary into a set of links, which map inflow to outflow for a triangle.

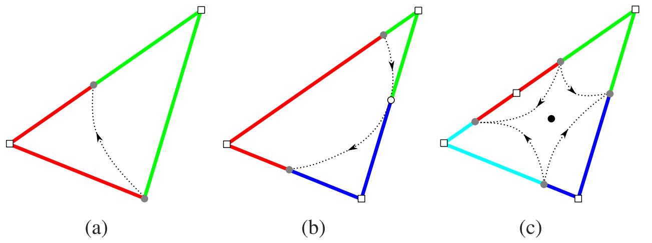


Fig. 4. Splitting of the boundary of a triangle into intervals: (a) A triangle with a forward vertex image (gray dot) of the lower right vertex; (b) A triangle with a single transition point (white circle) from internal flow and its forward and backward image (gray dots); (c) A triangle with a saddle point (black dot), its four sepx points (gray dots), and a transition point from external flow (white square). Note that in most cases, vertices also act as external transition points.

3.2 Definition

To understand and represent the flow behavior through a triangle, we first summarize the formal definitions given in [17]. Let Δ be a triangle with boundary $\partial\Delta$. An *origin-destination (o-d) pair* is a pair of points (p, q), where both p and q lie on $\partial\Delta$ and there exists a streamline between them which lies entirely in the interior of Δ. We call p an origin point and q a destination point. Let P be the set of all the origin points on $\partial\Delta$, and Q be the set of all the destination points on $\partial\Delta$. The *edge map of* Δ, $\xi : P \to Q$, is defined as the point-to-point mapping between the boundary of the triangle, such that $\xi(p) = q$ if (p, q) is an o-d pair. q is called the image of p under ξ. If there exists a critical point in the interior of the triangle, some points on $\partial\Delta$ will not be a part of any o-d pair, since they flow to or emerge from the critical point.

Edge maps provide a point-to-point mapping between entry and exit points of streamlines through a triangle. Fig. 3a visualizes an edge map as a graph of the point-to-point mappings. To efficiently represent the edge maps, we merge adjacent origin points that have adjacent destination points. This merging creates a more compact representation of the point-to-point mapping as a mapping between connected subsets of the boundary of a triangle, called *intervals*. The interval obtained by merging adjacent origin points is called the *origin interval*, while the interval obtained by merging their respective destination points is called the *destination interval*. Pairing up of an origin and its corresponding destination interval forms a *link*. A link is an interval-interval map, representing a region of unidirectional flow. Fig. 3 depicts the results of this merging process.

To facilitate practical applications like streamline integration, the edge map needs to account for the critical points as well. We define a *forward edge map* $\xi^+ : P \to \Delta$ such that given a point p where a streamline enters the triangle, the map gives us the unique point where it exits the triangle. If a critical point exists within the triangle, the flow may never exit, and ξ^+ returns the location of the critical point. Hence, the range of ξ^+ can include the interior of the triangle. On the points on the boundary, where flow does not enter the triangle, but instead, exits it, we define a *backward edge map* $\xi^- : Q \to \Delta$. For a point q on the boundary of Δ, $\xi^-(q)$ describes the unique point where flow entered the triangle on its path to q, or the critical point where p originated from.

We note that the edge map ξ as defined in [17] is a bijection and its inverse ξ^{-1} represents the edge map of inverted flow. However, according to the definition presented here, ξ^+ (or ξ^-) is a bijection if and only if there is no critical point present in the triangle. For such triangles, $\xi^- = (\xi^+)^{-1}$, since for points $p, q \in \partial\Delta$, $\xi^+(p) = q$ if and only if $\xi^-(q) = p$. As for triangles with critical point, this inverse relationship does not hold because ξ^+ (or ξ^-) is no more a one-to-one map. In either case, for a triangle Δ, ξ^+ and ξ^- completely describe the behavior of the flow through Δ.

3.3 Edge Map Generation

To generate a compact representation of the flow, edge maps subdivide $\partial\Delta$ into connected maps of flow between origin and destination interval, called links. While the endpoints of the links represent the flow accurately, an approximation can be made to represent the flow at the interior of links. To ensure consistent integration, any approximation can be chosen as long as it preserves the ordering of streamlines within a link. In this work, we use a linear approximation of the links. Thus, the flow representation in edge maps is approximate but consistent. Edge maps, however, do not ignore this approximation error, but represent it explicitly (as will be discussed in Section 4).

An edge map (forward and backward) can be encoded concisely as a collection of links of a triangle, such that the intervals are nonintersecting other than at their endpoints, and covers the entire boundary of the triangle (see Fig. 3). If there is a critical point present in the triangle, some links may include the critical point as a source or destination interval. Thus, to store the edge map for a triangle, we only need to encode a collection of links.

As discussed in Section 3.2, intervals are constructed by merging adjacent origin points whose destinations are also adjacent. At the maximum level of merging, the intervals are bounded by either:

1. *vertices* of the triangle;
2. images of vertices (Fig. 4a);
3. *transition points*: points where the flow changes between inflow and outflow (Fig. 4b);
4. images of transition points, and
5. *sepx points*, where the separatrices of a saddle exit or enter (Fig. 4c).

Fig. 5 gives the algorithm for computing the edge map for a triangle without a critical point. When the triangle has

```
ConstructEdgeMap(Δ):
  1) Identify the transition points on ∂Δ. Advect the
     internal transition points forward and backward to
     find their images.
     (There can be at most 6 transition points in a
     triangle: 1 per edge and 1 per vertex. [17])
  2) Advect any vertices of Δ that are not transition
     points forward (resp. backward) to find their images.
     (The transitions points, vertices, and their images
     cut ∂Δ into intervals of unidirectional flow.)
  3) Using the direction (inflow/outflow), and connectivity
     implied by advecting, pair intervals to form links.
     (Collection of these links compose the edge map.)
```

Fig. 5. Algorithm for creating the edge map for a triangle. The advection of vertices and transition points are computed using LEM [26].

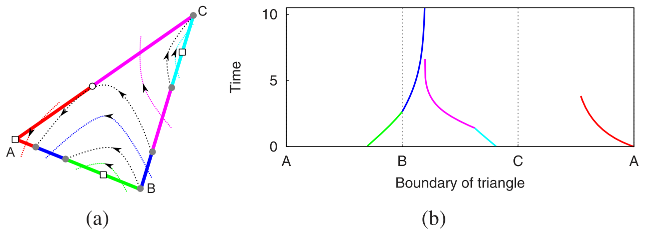


Fig. 6. (a) Edge map from Fig. 3. (b) The forward time-edge map (ξ_t^+) maps origin points to the time taken by them to reach their respective destinations.

a critical point (detected using [24]), the algorithm is similar except that there can be additional cuts (from separatrices) and the critical point itself can act as an interval.

Recently, we showed that there exist 23 equivalent classes of edge maps for linearly varying flow [17]. Here, equivalence is defined as invariance under rotation of triangle and inversion of flow. Linearity of the flow implies that there is a bounded number of intervals on the boundary of a triangle, which can be connected only in a limited number of ways to create a valid edge map. Since the number of classes is limited, the overhead for storing the edge maps of a single triangle is both bounded and relatively low.

3.4 Encoding Time in Edge Maps

While some flow properties like the topological segmentation are invariant to the vector magnitudes, others like streamline propagation are not. To represent the flow accurately, it is important that the edge maps encode the speed of the flow as well.

Every origin point on $\partial\Delta$, whose destination is also on $\partial\Delta$, takes a finite time to reach its respective destination point. We define a *time-edge map*, ξ_t, as the mapping between origin points on $\partial\Delta$ to the time they take to reach their corresponding destination, i.e., $\xi_t : P \rightarrow \mathbb{R}$. Thus, a complete description of the flow through Δ is encoded using ξ and ξ_t. Fig. 6b shows the time-edge map for the triangle in Fig. 3.

During the generation of edge maps, LEM also computes the time taken by the vertices and transition points to reach their images. This time information can be stored in edge maps for every point that is advected. Thus, the endpoints of origin intervals are assigned the time they take to reach the endpoints of their respective destination intervals.

Note that, however, if a triangle contains a critical point C, the origin points that flow in to the critical point take an infinite amount of time to reach the critical point. Edge maps replace the infinite time with an approximation at the endpoints p of the links, $t = d_p/|\vec{v}_p|$, where $d_p = \|p - C\|$, and $\vec{v}_p$ is the vector at p. During the analysis or visualization of a streamline, this approximation assumes a constant speed of the particle between p and C, and is important to ensure a computationally feasible termination of the streamlines going to critical points. The error due to this approximation is limited to the last step of propagation.

The forward time-edge map is then extended to $\xi_t^+ : P \rightarrow \mathbb{R}$, such that for an origin point, it defines a unique destination point, and the time taken to reach the destination point $\xi_t^+(p) = t$.

3.5 Linear Approximation of Edge Maps

The encoding of flow as edge maps ultimately allows us to determine structural properties of the flow through the triangle using a simple lookup. Consequently, this leads to computing flow-based properties efficiently. For example, we can query the edge maps to determine destinations of points under the flow by trivially performing lookup and composition on the maps. At each lookup, we have preserved the property that origin intervals are mapped

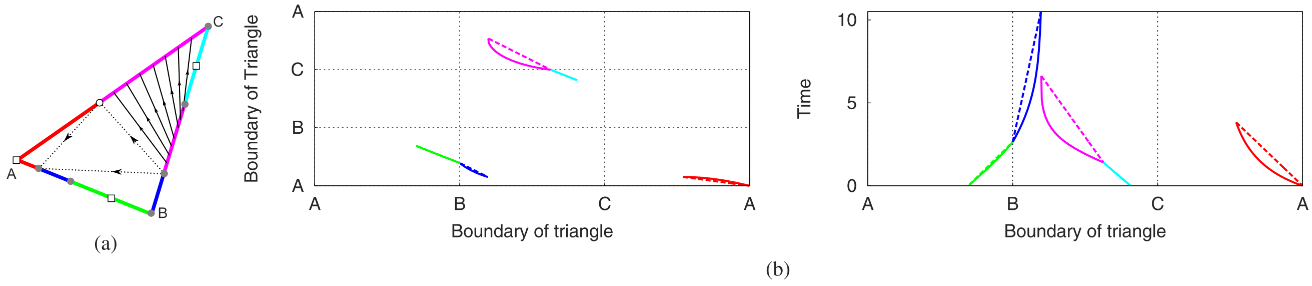


Fig. 7. (a) Linear approximation of the edge map from Fig. 3b. (b) We reproduce the plots of ξ and ξ_t from Figs. 3a and 6b here, and compare them (solid) with their linear approximation made by edge maps (dashed). The deviation between the two curves is the error of edge maps due to this approximation, shown in Fig. 8.

onto the same destination intervals they would have in the original PL flow in a consistent manner.

In particular, streamlines can be approximated on a per-link level by linearly interpolating between the origin and destination intervals. Hence, edge maps provide a way to approximate streamlines. As Fig. 7a shows, for an origin point on $\partial\Delta$, its path to its destination can be approximated by linearly interpolating in the origin interval corresponding to the origin point and projecting that point to the same barycentric coordinate in the destination interval.

Using the precomputed edge maps, any numerical integration to calculate streamlines (such as the simplest Euler integration) given by

$$x_{n+1} = x_n + (t_{n+1} - t_n) \cdot \vec{V}(x_n),$$

can be replaced by a simple lookup

$$\begin{aligned} x_{n+1} &= \xi^+(x_n), \\ t_{n+1} &= t_n + \xi_t^+(x_n). \end{aligned} \tag{1}$$

Hence, edge maps enable computation of streamlines in a consistent manner, and as will be discussed below, have a bounded error that can be explicitly computed and reduced.

4 Representing Error

The most common approach for tracing streamlines is numerical integration. From a given starting point these techniques repeatedly take small steps to approximate the next position in the path. The resulting error is controlled only indirectly by choosing a step size [12]. Since typically the true streamline is not known, this error cannot be quantified explicitly. While some schemes are more accurate than others and sophisticated techniques exist to locally adapt the step size, the indirect control over an unknown error represents a fundamental restriction. On the contrary, edge maps represent and control the error in representation explicitly and do not require setting a step size.

Furthermore, integrating streamlines numerically can also lead to inconsistencies, such as intersecting streamlines and significant differences between forward and backward traced lines. Edge maps replace integration with a one dimensional barycentric mapping that guarantees nonintersecting streamlines and consistency between forward and backward traces up to the floating point precision of the linear interpolation. Here, we study the approximation errors in edge maps, and provide tools to quantify and visualize them.

4.1 Approximation Errors in Edge Maps

As discussed in Section 3.5, streamline propagation can be performed with a bounded error using edge maps as given in (1). Since the edge map approximates the true exit point q and the true exit time t of a point p by linearly interpolating within the link as q' and t', respectively, it incurs some error. The spatial error can be calculated as the deviation of the exit point given by the map, from that given by the exact method ($\varepsilon = \|q - q'\|$). Similarly, the temporal error can be calculated as the deviation of the exit time given by the map, from that given by the exact method ($\delta = t - t'$). Fig. 7b shows this deviation using a comparison between the true flow and its linear approximation using edge maps, and Fig. 8 shows the error plots for the spatial and temporal error.

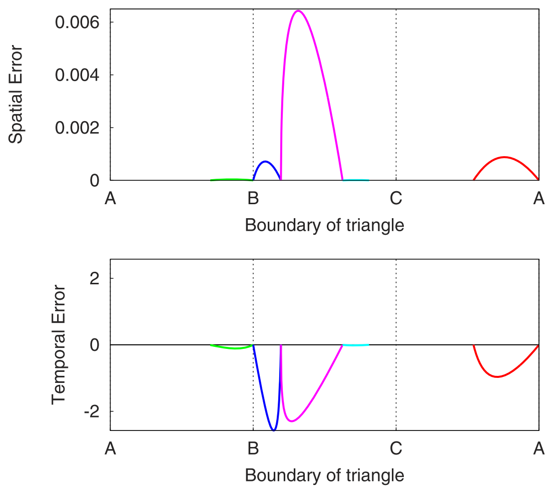


Fig. 8. The error plots for (top) spatial and (bottom) temporal errors for the linearly approximated map shown in Fig. 7a. Note that the average edge length of the triangle in consideration (Fig. 3) is 0.03.

The errors for both the spatial and temporal deviations are computed by sampling the link. Fig. 8 plots the spatial and temporal errors as a function of the boundary of the triangle. We use the maximum spatial error in a link l as the *spatial error of the link*, ε_l. Similarly, we can define the *temporal error of the link* δ_l as the maximum temporal error within a link.

To account for the amount of error incurred during the propagation, we define $\xi_\varepsilon^+ : P \rightarrow \mathbb{R}$ and $\xi_\delta^+ : P \rightarrow \mathbb{R}$ such that

$$\begin{aligned} \xi_\varepsilon^+(p) &= \varepsilon_l, \\ \xi_\delta^+(p) &= \delta_l, \end{aligned}$$

where ε_l and δ_l are the spatial and temporal errors of the link l which contains p. Thus, two floating-point values are stored per link to represent the mapping error in edge maps.

As explained in Section 3.3, the vertices, saddle separatrices, and transition points are advected to split the triangle perimeter into intervals. Since we use the LEM for this advection, the endpoints of the intervals are accurate up to the floating point precision of the system. These intervals are paired into links to construct the edge map. Hence, the mapping errors (both the spatial and temporal errors) are zero at the endpoints of the link.

In our experiments, we found that typically the spatial error is unimodal in a link. However, the error can also be bimodal, as shown in Fig. 9a. Similarly, the temporal error can be both positive and negative within the same link, as shown in Fig. 9b.

We observe that certain types of flow are less prone to error than others. For example, consider concentric circular orbits or a linear flow where any two streamlines do not diverge from or converge to each other. The mapping error is zero for such a case since the actual flow agrees to the

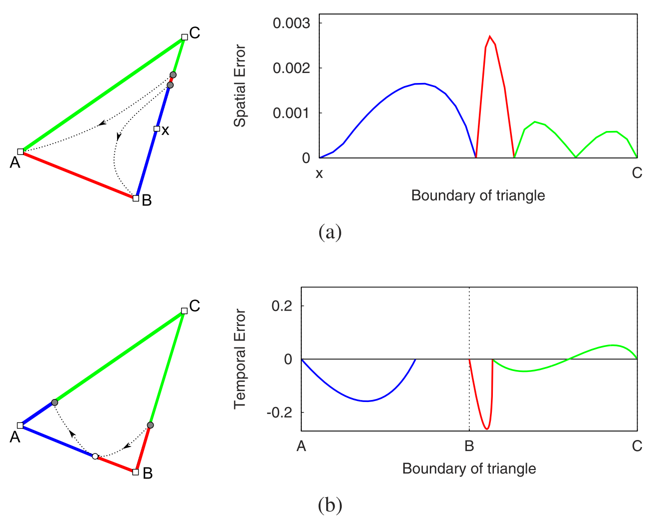


Fig. 9. (a) A (forward) edge map with bimodal spatial error in the green link). Note the horizontal axis has been scaled to the range $[x, C]$ to illustrate the error. (b) A (forward) edge map with both positive and negative temporal error within the green link. The horizontal axis has been scaled to range $[A, C]$ to illustrate the mapping error. In the destination interval on edge AB, there is no temporal error defined under the forward edge map. The average edge length of both the triangles in consideration is 0.3.

linear mapping within each link. Fig. 10 corroborates this intuition by testing the edge map propagation in a purely rotational flow. Hence, in the absence of mapping error, the propagation using edge maps is as accurate as the underlying method for advection used for map generation.

An upper limit to the mapping errors can be imposed by user parameters, ε_{MAX} and δ_{MAX}. If for a link, $\varepsilon_l > \varepsilon_{MAX}$ or $\delta_l > \delta_{MAX}$, we split the link at the point of maximum error to improve the accuracy of the map. We call this process *refinement of edge maps*. Fig. 12 shows the effect of spatial (Fig. 12b) and temporal (Fig. 12c) refinement on the edge map shown in Fig. 12a.

The level of refinement needed for a given flow is subject to the nature and magnitude of flow and the triangle size. Thus, the computation time and memory consumption of refined maps depend upon these factors.

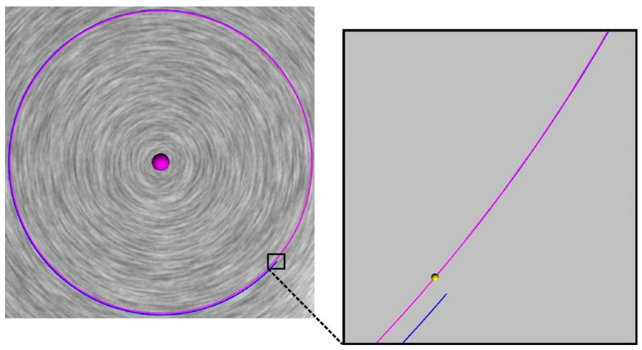


Fig. 10. Comparison between propagation using RK45 (blue) and edge maps (magenta) on a vector field defined by a counter-clockwise orbit seeded at the same point (yellow). The magenta and blue lines overlap in the beginning but a substantial deviation in RK45 streamline is observed after only one revolution around the critical point (purple). In the absence of mapping error, the mapped lines are accurate up to floating-point precision.

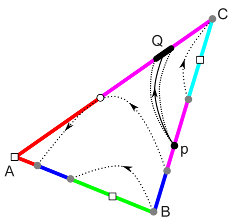


Fig. 11. Expansion of exit points represents error as a range of possible destinations.

5 Visualizing Spatial Error

Explicit representation of error in edge maps enables error visualizations of flow. Here, we discuss how to generate visualizations of spatial error using edge maps.

5.1 Expansion of Exit Points

We have been using the forward (ξ^+) and backward (ξ^-) edge maps as tools to look up the streamlines of individual particles. However, we can also represent the spatial error explicitly by redefining the edge maps as a one-to-many map.

$$\xi^+(p, \omega) = Q,$$

where, for an entry point p, instead of a single exit point q the map gives a range of possible exit points, a segment Q, under the *expansion factor* ω. This is illustrated in Fig. 11. The length of the segment Q is directly proportional to the expansion factor. Thus, we call Q the *expansion* of the exit point q.

The spatial error ε for p encodes the deviation of its exit point $\tilde{q}$ defined by the edge map from the true exit point q. Therefore, the expansion of the exit point Q calculated using $\omega = \xi^+_\varepsilon(p)$ provides an upper bound on the possible exit points of p. Furthermore, since the streamlines at the endpoints of the links are accurate, the expansion cannot span across links and thus is truncated at the endpoints of the link containing both p and q.

5.2 Streamwaves

The one-to-many mapping given by edge maps under the consideration of spatial error can be visualized by a *streamwave*, which is defined as the set of possible destinations that a massless particle may reach when accounting for possible expansions. Alternatively, a streamwave can be seen as the expansion of a streamline due to spatial uncertainties. In the current work, we quantify and visualize the spatial error as streamwaves propagate, by setting $\omega = \varepsilon_l$. However, any other kind of error can be modeled as the expansion ω for streamwaves.

Using edge maps, we can compute the streamwaves as follows:

$$X_{n+1} = \xi^+(X_n, \omega),$$

where $X_0 = \{x_o\}$ represents the seed point of the wave and X_n the set of points currently at the front of the wave. Since a streamwave models the spatial error only, the speed of the wave is reflective of the linear time approximation of the edge maps. The temporal error in the edge maps is ignored. Using traditional techniques to compute streamwaves as a collection of streamlines can become computationally expensive and requires delicate processing in regions of high divergence. Using edge maps, however, propagating a

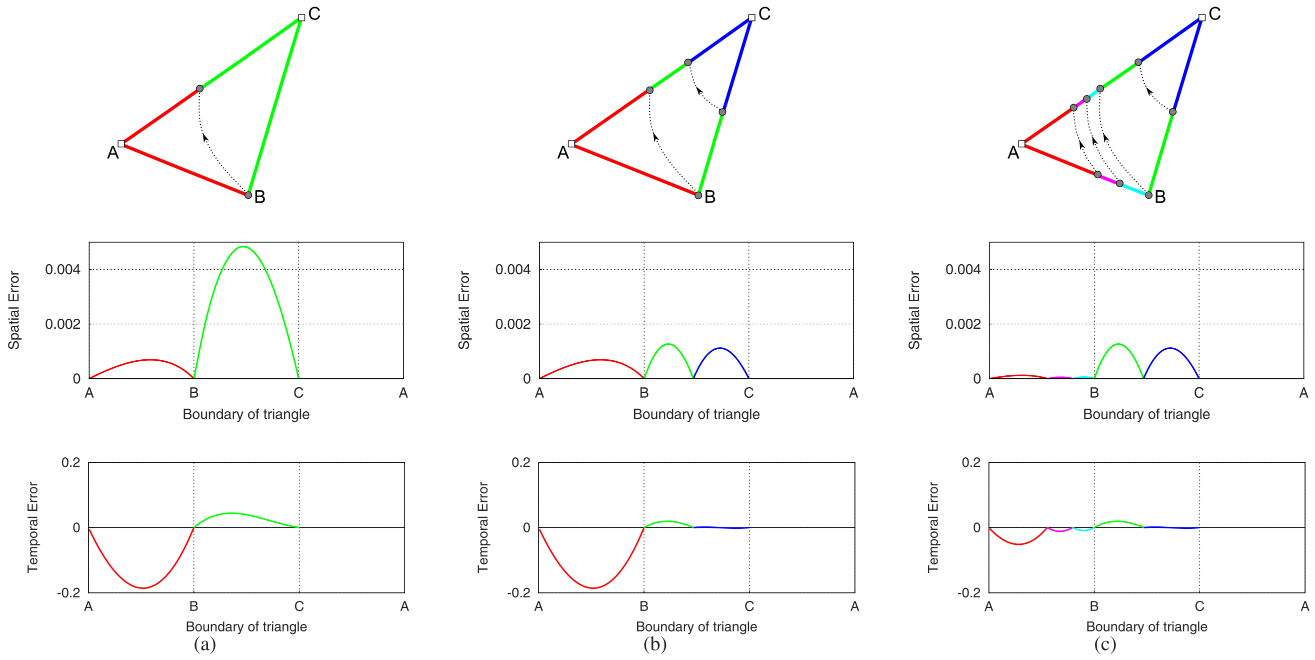


Fig. 12. Reducing the mapping error (middle row: spatial error, bottom row: temporal error) by refinement of edge maps (top row). Level of refinement increases from left to right. The length of the edge AC is 0.0354, and the average time taken by a particle to travel across the triangle is 1.7. (a) No refinement. (b) Spatial refinement with $\varepsilon_{MAX} = 0.003$ splits the green link into two, creating two new links (green and blue) with smaller spatial and temporal errors. (c) Temporal refinement with $\delta_{MAX} = 0.06$ splits the red link twice, creating three new links (red, cyan, and magenta) with even smaller spatial and temporal errors.

wave is only as expensive as the number of links in the triangles currently at the front.

Furthermore, if there exist no bifurcations in a triangle, then only extremes of the range of exit points X_{n+1} are of interest, and all intermediate points are handled implicitly. For triangles with bifurcation, a streamwave may split into two streamwaves, each of which can be propagated independently.

Fig. 13 shows streamwaves computed on simulation of a slice of a homogeneous charge compression ignition (HCCI) engine combustion [13]. The computation of edge maps for 816,642 triangles in the data set took 223 seconds and 200 MB. As shown in the figure, a streamwave is the superset of a single streamline, so analyzing only the streamline in the presence of error is an incomplete analysis. Since expansion of a streamwave in the presence of error may cause it to revisit a certain region, we truncate the streamwave so as to avoid going into infinite flow loops. This is consistent with our definition of streamwave since we only want to visualize the region that can be visited (at least once) by the streamwave. The shape of streamwave reflects the nature of the underlying flow. A "linear" streamwave will be obtained if the flow is mostly linear (little or no rotation), e.g., Fig. 14. If the flow is highly rotational, the streamwave revisits certain regions and becomes "blob-like," e.g., Fig. 13. The color of the streamwave progresses from green to red as it propagates forward in time, as an indication of the speed of the streamwave.

Streamwaves also present a method to visualize error bounds of other integration techniques. For example, Fig. 14 shows the integration of a streamline connecting a source to a sink using three different techniques. By showing a streamwave, whose expansion is set larger than the maximum error for Euler integration, we can visualize a comparison between Euler integration, fourth-order Runge-Kutta, and LEM.

5.3 Visualization of Fuzzy Topology

Topological structures in vector fields, such as their topological skeleton [14], are one of the key features used

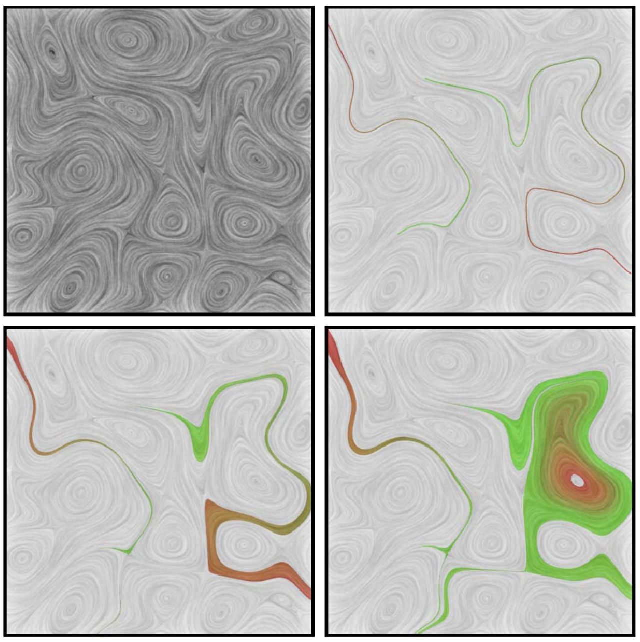


Fig. 13. Visualizations of streamwaves on HCCI data set [13]. The 640×640 data are mapped to a $[-1, 1] \times [-1, 1]$ plane. Top left: original vector field visualized with IBFV [39]. Top right: two streamlines, one near a saddle's separatrix. Bottom: Two images show streamwaves at different levels of error (0.0001 and 0.0002). Streamwaves are colored from green to red, showing the distance the flow has propagated as a measure of the number of maps the streamwave has traveled through. Note that the error levels have been exaggerated to illustrate expansion and bifurcation of streamwaves.

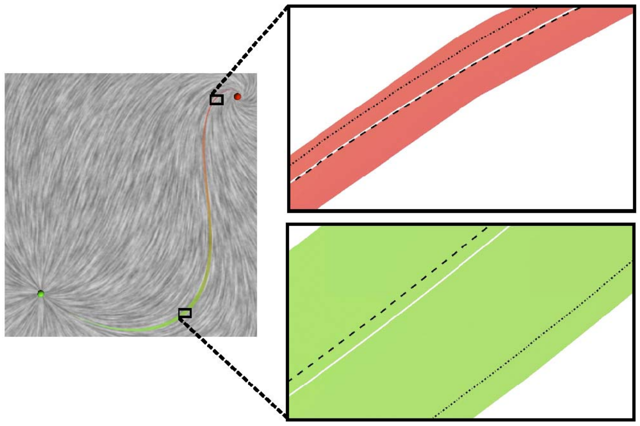


Fig. 14. A streamline using RK4 (black dashed) using stepsize $\Delta t = 0.005$, Euler (black dotted) using stepsize $\Delta t = 0.005$ and local exact method (white solid), and a streamwave using edge maps with $\omega = 0.0001$ were seeded at the same point. Considering the local exact method to be the ground truth, some deviation is observed in Euler and RK4 streamlines. It is also observed that the streamwave, centered around the LEM streamline, bounds the two erroneous streamlines at all the times.

to analyze vector field data. Traditionally, the skeleton is computed by tracing four separatrices out of each saddle (two forward and two backward) by computing streamlines starting in the directions of the eigenvectors of all the saddles. These separatrices terminate when they arrive at another critical point or leave the boundary of the domain. However, this approach faces challenges since compound integration error can cause the trace to end at an incorrect critical point. In particular, unstable topologies, such as when a pair of saddles is connected by a separatrix, suffer from this form of inconsistency.

We can use the streamwave construction to study the robustness of topological representations. By growing a streamwave with $\omega = 0$, in the forward direction from all sources and in the reverse direction from all sinks we can perform a partial topological decomposition of a vector field that is analogous to stable and unstable manifolds in scalar field topology [5]. These streamwaves are initiated from the segments of the boundary flowing out of and into the triangles containing the sources and sinks, respectively. While we cannot account for centers, streamwaves can provide important information about the structure of the flow. In particular, in the absence of closed orbits, the union of the forward and backward streamwaves creates a

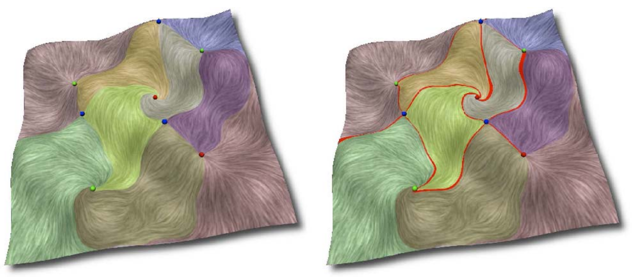


Fig. 15. Visualization of the (left) topology and (right) fuzzy topology for a synthetic data set containing three sources (green), two sinks (red), and three saddles (blue). Streamwaves with (left) $\omega = 0$ and (right) $\omega > 0$ are used for such a construction.

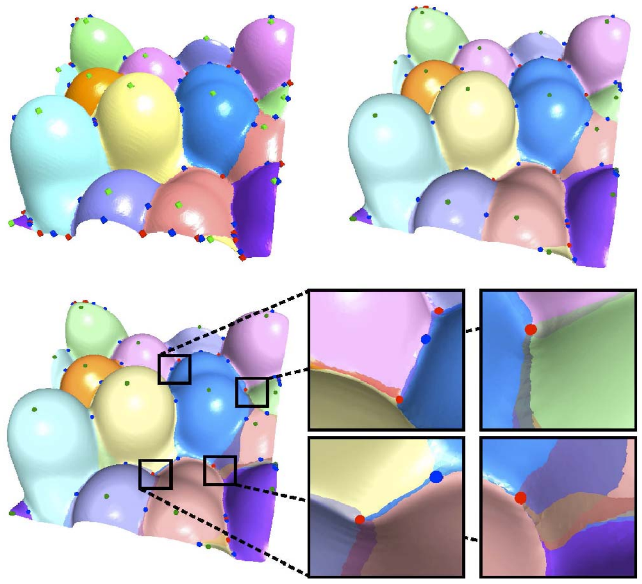


Fig. 16. Visualizations of a Rayleigh-Taylor instability. Top row: We reproduce the results from Laney et al. [21] (left) side-by-side with our edge map computation of the unstable manifolds using $\omega = 0$ (right). Bottom row: when the error factor, $\omega > 0$ is accounted for (left) we can observe the emerging overlaps (right).

covering of domain similar to the segmentation induced by traditional vector field topology.

However, in our construction for $\omega > 0$, each point in the domain may be part of several streamwaves creating a notion of *fuzzy* topology as shown in Fig. 15. Setting $\omega = \varepsilon_l$ highlights the regions of the domain that are unstable/uncertain under the approximation errors of edge maps as thick red bands. This provides important information about any potential instabilities in the topological segmentation. In particular it provides users with an intuitive measure of how certain a given structure is.

To illustrate the new concept of fuzzy topology we compare streamwaves with traditional scalar field techniques, see Fig. 16. Laney et al. [21] use topological analysis on the interface surfaces between heavy and light fluids in a *Rayleigh-Taylor instability*. In particular, the unstable manifolds of the height function segment the surfaces into bubbles, the primary feature of interest. Similarly, we can compute the gradient field of the same data set, and construct the manifolds using streamwaves. Both techniques provide a similar view of the data but our representation is richer by also showing the inevitable inconsistencies at the boundaries of the bubbles.

Fig. 17 shows an additional example of fuzzy topology computed on a combustion chamber data set [23] with refined maps, indicating the swirling structure on its surface. Since, the spatial error in the maps is very low, the fuzzy regions are negligible.

6 Visualizing Temporal Error

For a streamline that is computed assuming error-free propagation, there is a unique time instant given for every position, and vice versa. While all numerical integration techniques have some associated error, however small or big, this error is generally ignored [29]. Using edge maps, we can bound and visualize the temporal error in streamlines,

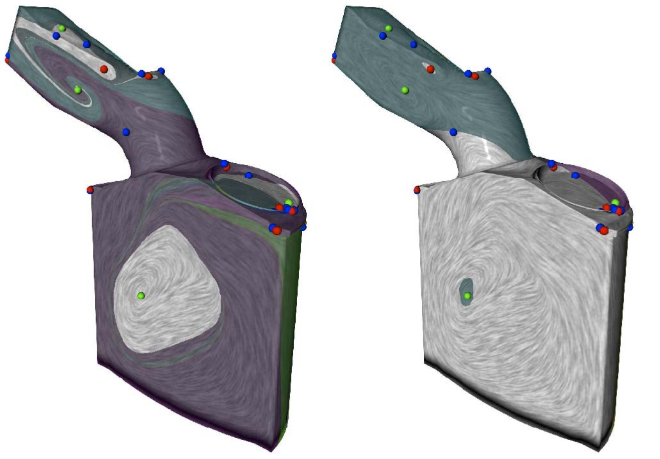


Fig. 17. Visualization of the stable (left) and unstable (right) manifolds of the flow defined on the surface of a combustion chamber [23]. The flow has 13 sources (green), 15 sinks (red), and 27 saddles (blue). With sufficiently refined maps, the fuzzy regions are negligible indicating a more accurate representation.

as discussed below. Here, we consider the temporal errors in the map only, and assume no spatial error.

6.1 Temporal Error in Streamlines

The temporal error of a link (δ_l) represents the maximum possible deviation in edge map's approximation of time from the true time. The points on the streamline corresponding to a temporal error will span across triangles. Since an edge map is not aware of the vector field in neighboring triangles, it can not return a temporal expansion of points as in the case of the spatial expansion (see Section 5.1). Instead, the temporal expansion of points is computed indirectly using the temporal error. When a streamline is integrated using edge maps, the temporal error of the links it passes through is accumulated

$$\delta_{n+1} = \delta_n + \xi^+_\delta(x_n).$$

Thus, every point x_n on the streamline is associated with a time t_n, and a temporal error δ_n.

Fig. 18 illustrates the accumulation of temporal error along a streamline. Given a point p on the streamline, the time and the temporal error can be easily found by interpolating the data stored in the streamline. The accumulated temporal error is visualized as an error field along the streamline in Fig. 19.

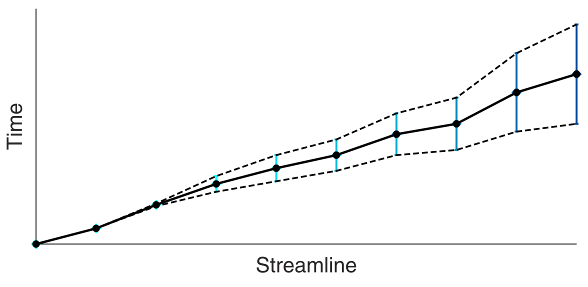


Fig. 18. The accumulation of temporal error shown as an error plot for a streamline in time space. Every point on the streamline has an associated temporal error, shown by the vertical thickness of the dashed envelope. (The vertical thickness of the envelope is twice the amount of temporal error.)

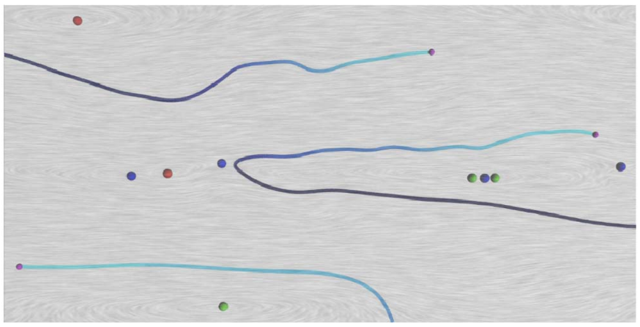


Fig. 19. The accumulated temporal error in streamlines is shown for the ocean data set on a regular grid of 573 × 288 vertices. The flow contains three repelling foci (green balls), two attracting foci (red ball), and four saddles (blue balls). The seed points of the streamlines are shown as small magenta balls. The color map cyan-dark blue is mapped to increasing temporal error along the streamline. A sharp increase in accumulated errors near the saddle at the center of the field indicates a region of high-temporal error.

6.2 Visualizing Temporal Error Spatially

The temporal error δ_n for a point x_n on a streamline defines a range of possible values of time $[t_n - \delta_n, t_n + \delta_n]$ that can be associated with x_n. This range is shown as the vertical thickness of the error envelope in Fig. 18. Another way to visualize the temporal error is to understand its spatial manifestation, which gives a more intuitive understanding of flow behavior under error.

A streamline generated using edge maps can be queried for all the points which could be visited at a given time τ. This query returns every point whose time range contains τ. However, since an edge map streamline contains only the points lying on the edges of the triangulation, we interpolate the time ranges between adjacent points on streamline to include the points on the interior of triangles. This is analogous to intersection of a horizontal line representing some value of time, with the time envelope, and projecting this intersection onto the horizontal axis as shown in Fig. 20. Thus, we get a spatial range centered around the true (in absence of temporal error) position of the particle at every time instant.

Fig. 21 shows the visualization of the spatial range due to temporal error. The spatial range presents a way to visualize the error bounds on the uncertainty in the position of the

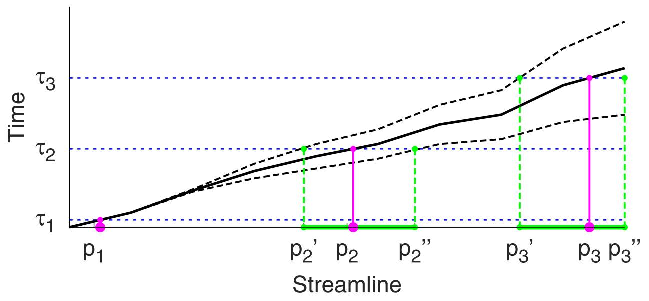


Fig. 20. Computation of the spatial range corresponding to the temporal error in a streamline. For every time instant τ_i, represented by the (dashed blue) horizontal line, its intersections with the temporal error envelope (black dashed) are computed, and then projected on to horizontal axis (the streamline). This gives the spatial range of points (p'_i, p''_i) on the streamline, shown in green. To compute the true position (considering no error), the intersection with the actual time curve (black solid) is projected (p_i), shown in magenta.

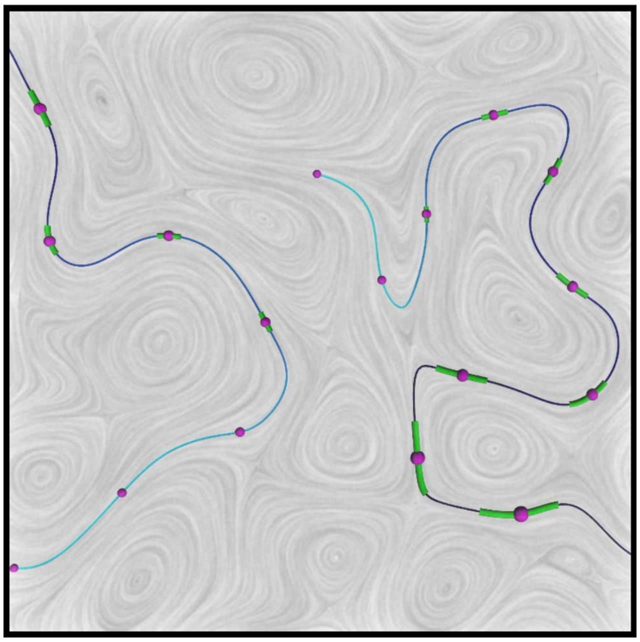


Fig. 21. Visualization of temporal error in streamlines on vector field from Fig. 13. Two streamlines are seeded in the flow: one in the bottom left (exiting from top left), and the other in the upper center (exiting from the bottom right). Increasing time steps are shown with increasing size of the magenta ball, and thickness of the green tube. With increasing time, the magenta ball travels along the streamline representing the movement of the particle. The green tube represents the spatial range due to the temporal error in the propagation. Spatial range is shown for 10 equidistant time steps starting from 0.

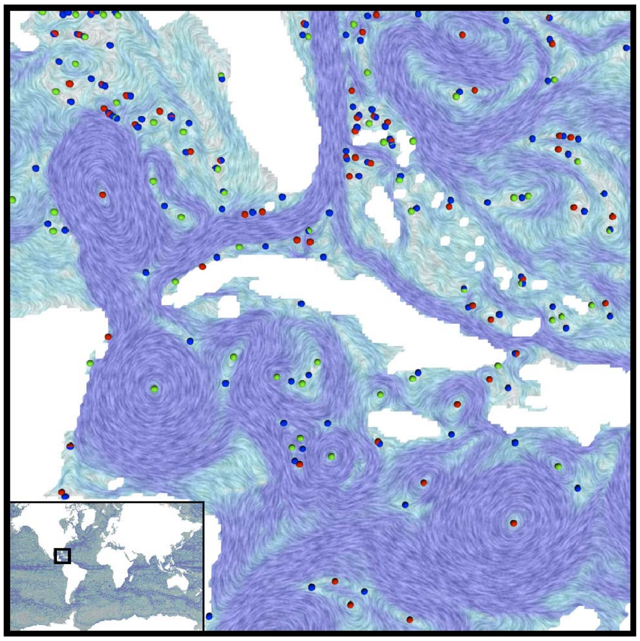


Fig. 22. The oceanic currents in the Gulf of Mexico and the Caribbean Sea. This 200×200 tile is taken from a larger simulation of oceanic currents [25] (inset), and mapped to a $[-1,1] \times [-1,1]$ plane. The region has 56 sources (green balls), 63 sinks (red balls), and 117 saddles (blue balls). The increasing speed of the flow is mapped to color from cyan to purple. We demonstrate our visualization techniques using edge maps on this flow data.

particle at a given time. Note that the extracted spatial range is only a spatial manifestation of temporal error. An equal amount of temporal error will produce a longer spatial range for higher vector magnitudes. By showing the positional extents of these temporal errors, we can investigate the interplay between velocity magnitude and time.

7 Demonstrations

We demonstrate each of the visualization techniques discussed above on a 200×200 tile of a larger depth slice of a 3D simulation of global oceanic currents as shown in Fig. 22. These simulations were produced using a technique based on a boundary impulse response functions [25]. The tile is mapped to a planar domain of size $[-1,1] \times [-1,1]$. Although the flow in the original 3D simulation is incompressible, a depth slice from this 3D domain does not satisfy incompressibility anymore. The slicing creates sinks and sources due to the flow across depths. We visualize the streamwaves, spatial range due to temporal error, and fuzzy topology (unstable manifolds) in Fig. 23 (left to right) at increasing refinement levels (top to bottom).

Fig. 23a shows a streamwave at two different error thresholds. We notice that the streamwave hits a saddle before exiting the domain from top-left. At high error, the streamwaves flows into many slowly rotating critical points. A sudden and substantial increase in the thickness of the streamwave reveals the sensitivity of this region to the spatial error.

Fig. 23b shows the spatial range corresponding to the temporal error as a sequence of time instants starting from 0 with a step of 0.3. A steady increase in the lengths of the green tubes is observed reflecting a steady increase in the temporal error. This behavior is in conjunction with a steady increase in the thickness of the streamwave (Fig. 23a) until it hits the saddle. Visualizing both these error together is important to get a complete understanding of how the flow behaves under error.

Fig. 23c shows the unstable manifolds grown from all the sources in the flow. These regions overlap with each other based on the approximation error in the flow. This overlap indicates the fuzziness in the boundary between them due to this error. Observe that the overlaps reduce with higher refinement. There are many sources in the domain with very low divergence, in which case the corresponding manifold does not grow enough to be noticeable at this scale, especially at the higher refinement (bottom). Spatial refinement of edge maps increases their fidelity to piecewise linear flow. This can create cycles which were previously absent in the flow. Such cycles bound these growing regions. It is observed that large regions of the data set remain untouched by these manifolds, suggesting that the flow in these regions is not associated with the sources in the domain. Hence, the flow in these untouched regions must either be bounded by orbits, or flowing in from the boundary of the domain.

The computation of edge maps for this data set containing 63,010 triangles took 19.4 seconds and 19 MB, giving, on an average 2.23 links per triangle. Spatial refinement (from unrefined maps) of 0.0001 took 29.5 minutes and 36 MB giving 4.2 links per triangle, and temporal refinement (from unrefined maps) of 0.0001 took 25 minutes and 35 MB, giving 4.1 links per triangle.

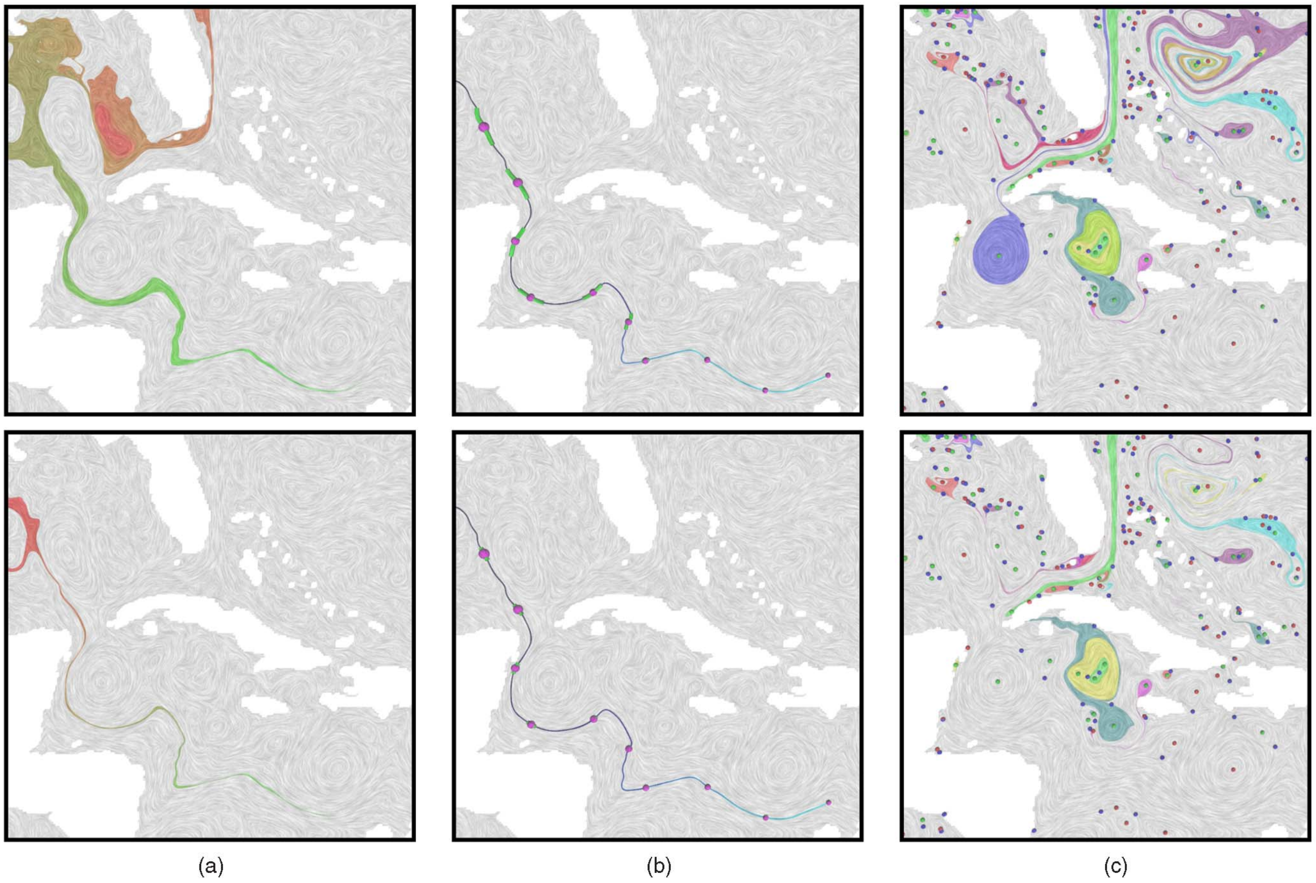


Fig. 23. Visualizations using edge maps on the oceanic current data from Fig. 22. (a) Streamwave visualization at spatial error refinement of 0.1 (top) and 0.0001 (bottom). (b) Spatial range due to temporal error at temporal refinement of 0.01 (top) and 0.0001 (bottom). (c) Unstable manifolds at a spatial refinement of 0.0001 (top) and 0.00001 (bottom).

8 Discussion and Future Work

Edge maps establish a novel way to represent and analyze sampled vector fields. Compared to traditional interpolation schemes they have several attractive properties: 1) numerical integration (and thus all error accumulation) is confined to the map construction; 2) unavoidable errors accumulated during integration or inherent in the representation can be explicitly encoded; and 3) flow information extracted from the maps is guaranteed to be consistent. These advantages translate into a number of useful visualization and analysis tools such as streamwaves, topological descriptions, and visualizations of how temporal errors manifest spatially. The edge map representation can also reproduce published results (using integration schemes), as well as provide richer interpretations that are not possible using existing techniques.

Nevertheless, edge maps have some disadvantages, most notably the storage overhead per triangle. Furthermore, applying texture-based flow visualization techniques for edge maps, such as IBFV, requires some additional effort. Extending the edge map construction to volumetric domains could pose a significant challenge given the number of potential map classes per tetrahedral element. While edge maps remove the need for numerical integration, the consistency guaranteed by edge maps is still up to floating-point precision as the round-off error during map-lookups still needs to be accounted for.

In this work, we have discussed techniques to visualize spatial and temporal errors using edge maps, independently, in the form of streamwaves and spatial ranges due to temporal error. An obvious next step in this research is to integrate the way errors are propagated to produce a single visualization of spatiotemporal errors.

We have presented edge maps for triangulated domains; however, as a general concept, the idea of edge maps is applicable to other kinds of surface domains as well. For example, for structured grids and unstructured quadrilaterals edge maps can be created between the boundaries of the cells. In these domains, different interpolations in the interior of cells will be required and the types of flow behaviors shown in [17] will need to be redefined. However, on a conceptual level of replacing integration with a boundary mapping, the idea of edge maps is both extensible as well as applicable to different discretizations of domain. Also, a different approximation scheme could be used to approximate the edge maps instead of the linear scheme used here.

There exist some interesting opportunities to exploit the consistency and discrete nature of edge maps. One such potential application of edge maps is in vector field simplification. Because the flow can be represented discretely and error can be encoded explicitly, we can merge edge maps to reduce the complexity of the flow fields, or to perform domain simplification keeping the error in the flow bounded.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation awards IIS-1045032, OCI-0904631, OCI-0906379, and CCF-0702817, and by King Abdullah University of Science and Technology (KAUST) Award No. KUS-C1-016-04. This work was also performed under the auspices of the US Department of Energy (DOE) by the University of Utah under contracts DE-SC0001922, DE-AC52-07NA27344, and DE-FC02-06ER25781, and Lawrence Livermore National Laboratory (LLNL) under contract DE-AC52-07NA27344. The authors are grateful to Jackie Chen for the data set from Fig. 13, Robert S. Laramee for the combustion chamber data set from Fig. 17, and Paul Miller, William Cabot, and Andrew Cook for the bubbles data set from Fig. 16. We also thank Mathew Maltude from the Climate, Ocean and Sea Ice Modelling program at Los Alamos National Laboratory (LANL) and the BER Office of Science UV-CDAT team for providing us access to the ocean data from Figs. 22 and 23. Attila Gyulassy and Philippe P. Pebay provided many useful comments and discussions. LLNL-JRNL-485511.

REFERENCES

[1] H. Bhatia, S. Jadhav, P.-T. Bremer, G. Chen, J.A. Levine, L.G. Nonato, and V. Pascucci, "Edge Maps: Representing Flow with Bounded Error," *Proc. IEEE Fourth Pacific Visualization Symp.*, pp. 75-82, Mar. 2011.

[2] D. Bommes, H. Zimmer, and L. Kobbelt, "Mixed-Integer Quadrangulation," *ACM Trans. Graphics*, vol. 28, no. 3, p. 1, 2009.

[3] G. Chen, K. Mischaikow, R.S. Laramee, P. Pilarczyk, and E. Zhang, "Vector Field Editing and Periodic Orbit Extraction Using Morse Decomposition," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, no. 4, pp. 769-785, July/Aug. 2007.

[4] G. Chen, K. Mischaikow, R.S. Laramee, and E. Zhang, "Efficient Morse Decompositions of Vector Fields," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 4, pp. 848-862, July/Aug. 2008.

[5] H. Edelsbrunner, J. Harer, and A. Zomorodian, "Hierarchical Morse-Smale Complexes for Piecewise Linear 2-Manifolds," *Discrete and Computational Geometry*, vol. 30, no. 1, pp. 87-107, 2003.

[6] M. Fisher, P. Schröder, M. Desbrun, and H. Hoppe, "Design of Tangent Vector Fields," *ACM Trans. Graphics*, vol. 26, no. 3, p. 56, 2007.

[7] M.S. Floater, G. Kos, and M. Reimers, "Mean Value Coordinates in 3D," *Computer Aided Geometric Design*, vol. 22, pp. 623-631, 2005.

[8] R. Forman, "A User's Guide to Discrete Morse Theory," *Proc. Int'l Conf. Formal Power Series and Algebraic Combinatorics*, p. 48, 2001.

[9] C. Garth, H. Krishnan, X. Tricoche, T. Tricoche, and K.I. Joy, "Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields," *IEEE Trans. Visualization Computational Graphics*, vol. 14, no. 6, pp. 1404-1411, Nov./Dec. 2008.

[10] A. Globus, C. Levit, and T. Lasinski, "A Tool for Visualizing the Topology of Three-Dimensional Vector Fields," *Proc. IEEE Visualization*, pp. 33-41, 1991.

[11] A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann, "Efficient Computation of Morse-Smale Complexes for Three-Dimensional Scalar Functions," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1440-1447, Nov./Dec. 2007.

[12] E. Hairer, S.P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer Series in Computational Mathematics. Springer-Verlag, 1993.

[13] E.R. Hawkes, R. Sankaran, P.P. bay, and J.H. Chen, "Direct Numerical Simulation of Ignition Front Propagation in a Constant Volume with Temperature Inhomogeneities: II. Parametric Study," *Combustion and Flame*, vol. 145, nos. 1/2, pp. 145-159, 2006.

[14] J. Helman and L. Hesselink, "Representation and Display of Vector Field Topology in Fluid Flow Data Sets," *Computer*, vol. 22, no. 8, pp. 27-36, Aug. 1989.

[15] M.W. Hirsch, S. Smale, and R.L. Devaney, *Differential Equations, Dynamical Systems, and An Introduction To Chaos*, second ed. Elsevier Academic Press, 2004.

[16] J.P. Hultquist, "Constructing Stream Surfaces in Steady 3D Vector Fields," *Proc. IEEE Visualization*, pp. 171-178, 1992.

[17] S. Jadhav, H. Bhatia, P.-T. Bremer, J.A. Levine, L.G. Nonato, and V. Pascucci, "Consistent Approximation of Local Flow Behavior for 2D Vector Fields Using Edge Maps," *Proc. Topological Methods in Data Analysis and Visualization II - Theory, Algorithms, and Applications*, Part 3, pp. 141-159, 2012.

[18] K.M. Janine, J. Bennett, G. Scheuermann, B. Hamann, and K.I. Joy, "Topological Segmentation in Three-Dimensional Vector Fields," *IEEE Trans. Visualization and Computer Graphics*, vol. 10, no. 2, pp. 198-205, Mar./Apr. 2004.

[19] C.R. Johnson and A.R. Sanderson, "A Next Step: Visualizing Errors and Uncertainty," *IEEE Computer Graphics and Applications*, vol. 23, no. 5, pp. 6-10, Sept./Oct. 2003.

[20] P. Kipfer, F. Reck, and G. Greiner, "G.: Local Exact Particle Tracing on Unstructured Grids," *Computer Graphics Forum*, vol. 22, pp. 133-142, 2003.

[21] D.E. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci, "Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1053-1060, Sept./Oct. 2006.

[22] R.S. Laramee, H. Hauser, L. Zhao, and F.H. Post, "Topology Based Flow Visualization: The State of the Art," *Topology-Based Methods in Visualization*, Mathematics and Visualization, pp. 1-19, Springer, 2007.

[23] R.S. Laramee, D. Weiskopf, J. Schneider, and H. Hauser, "Investigating Swirl and Tumble Flow with a Comparison of Visualization Techniques," *Proc. IEEE Visualization*, pp. 51-58, 2004.

[24] Y. Lavin, R. Batra, and L. Hesselink, "Feature Comparisons of Vector Fields Using Earth Mover's Distance," *Proc. IEEE/ACM Visualization*, pp. 103-109, Oct. 1998.

[25] M. Maltrud, F. Bryan, and S. Peacock, "Boundary Impulse Response Functions in a Century-Long Eddying Global Ocean Simulation," *Environmental Fluid Mechanics*, vol. 10, pp. 275-295, 2010.

[26] G. Nielson and I.-H. Jung, "Tools for Computing Tangent Curves for Linearly Varying Vector Fields over Tetrahedral Domains," *IEEE Trans. Visualization and Computer Graphics*, vol. 5, no. 4, pp. 360-372, Oct. 1999.

[27] M. Otto, T. Germer, H.-C. Hege, and H. Theisel, "Uncertain 2D Vector Field Topology," *Computer Graphics Forum*, vol. 29, no. 2, pp. 347-356, 2010.

[28] M. Otto, T. Germer, and H. Theisel, "Uncertain topology of 3D Vector Fields," *Proc. IEEE Fourth Pacific Visualization Symp.*, pp. 65-74, 2011.

[29] A.T. Pang, C.M. Wittenbrink, and S.K. Lodha, "Approaches to Uncertainty Visualization," *The Visual Computer*, vol. 13, pp. 370-390, 1996.

[30] N. Ray, B. Vallet, W.-C. Li, and B. Lévy, "N-Symmetry Direction Field Design," *ACM Trans. Graphics*, vol. 27, no. 2, pp. 1-13, 2008.

[31] J. Reininghaus and I. Hotz, "Combinatorial 2D Vector Field Topology Extraction and Simplification," *Proc. Workshop Topological Methods in Data Analysis and Visualization*, 2009.

[32] J. Reininghaus, C. Löwen, and I. Hotz, "Fast Combinatorial Vector Field Topology," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 10, pp. 1433-1443, Oct. 2011.

[33] G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, K.I. Joy, and W. Kollmann, "A Tetrahedra-Based Stream Surface Algorithm," *Proc. Conf. Visualization (VIS '01)*, pp. 151-158, 2001.

[34] G. Scheuermann, H. Krüger, M. Menzel, and A.P. Rockwood, "Visualizing Nonlinear Vector Field Topology," *IEEE Trans. Visualization and Computer Graphics*, vol. 4, no. 2, pp. 109-116, Apr.-June 1998.

[35] G. Scheuermann and X. Tricoche, "Topological Methods in Flow Visualization," *Visualization Handbook*, pp. 341-356, Elsevier, 2004.

[36] A. Szymczak, "Stable Morse Decompositions for Piecewise Constant Vector Fields on Surfaces," *Proc. Joint Eurographics - IEEE Symp. Visualization (EuroVis '11)*, 2011.

[37] A. Szymczak and E. Zhang, "Robust Morse Decompositions of Piecewise Constant Vector Fields," *IEEE Trans. Visualization and Computer Graphics*, 2011.

[38] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel, "Saddle Connectors - An Approach to Visualizing the Topological Skeleton of Complex 3D Vector Fields," *Proc. IEEE Visualization*, pp. 225-232, 2003.

[39] J.J. van Wijk, "Image Based Flow Visualization," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 754-754, 2002.

[40] V. Verma and A. Pang, "Comparative Flow Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 10, no. 6, pp. 609-624, Nov./Dec. 2004.

[41] T. Weinkauf, H. Theisel, K. Shi, H.-C. Hege, and H.-P. Seidel, "Extracting Higher Order Critical Points and Topological Simplification of 3D Vector Fields," *Proc. IEEE Visualization,* pp. 559-566, 2005.

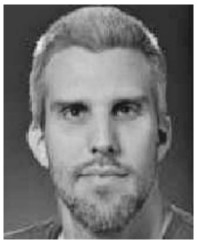
[42] T. Wischgoll and G. Scheuermann, "Detection and Visualization of Closed Streamlines in Planar Flows," *IEEE Trans. Visualization and Computer Graphics,* vol. 7, no. 2, pp. 165-172, Apr.-June 2001.

[43] C.M. Wittenbrink, A.T. Pang, and S.K. Lodha, "Glyphs for Visualizing Uncertainty in Vector Fields," *IEEE Trans. Visualization and Computer Graphics,* vol. 2, no. 3, pp. 266-279, Sept. 1996.

[44] E. Zhang, K. Mischaikow, and G. Turk, "Vector Field Design on Surfaces," *ACM Trans. Graphics,* vol. 25, no. 4, pp. 1294-1326, 2006.

Harsh Bhatia received the BTech degree in information and communication technology from DA-IICT, India in 2007. He is currently a graduate student in computer science at the Scientific Computing and Imaging (SCI) Institute at the University of Utah. His research interests include topological analysis of scalar and vector fields, uncertainty visualization, computer graphics and scientific visualization, and modeling and simulation. He is a student member of the IEEE.

Shreeraj Jadhav received the BE degree in mechanical engineering in 2006 from the University of Pune, India. He is a graduate student at the University of Utah, School of Computing. Currently, he is working as a research assistant at the Scientific Computing and Imaging (SCI) Institute at the University of Utah. His research interests include topology-based methods in visualization and data analysis, uncertainty quantification, progressive and multiresolution methods. He is a student member of the IEEE.


Peer-Timo Bremer received a diploma in mathematics and computer science from the Leipniz University in Hannover, Germany in 2000 and the PhD degree in computer science at the University of California, Davis in 2004. He is a computer scientist and project leader at the Center for Applied Scientific Computing at the Lawrence Livermore National Laboratory (LLNL) and a research professor at the SCI Institute of the University of Utah. Prior to his tenure at CASC he was a postdoctoral research associate at the University of Illinois, Urbana-Champaign. He is a member of the IEEE Computer Society and the IEEE.

Guoning Chen received the bachelor's degree in 1999 from Xi'an Jiaotong University, China and the master's degree in 2002 from Guangxi University, China. In 2009, he received the PhD degree in computer science from Oregon State University. His research interests include scientific visualization, computational geometry and topology, and computer graphics. Currently, he is a postdoctoral research associate at the Scientific Computing and Imaging (SCI) Institute at the University of Utah. He is a member of the IEEE and the ACM.

Joshua A. Levine received the bachelor's degrees (2003) and the master's degree (2004) from Case Western Reserve University. He achieved the PhD degree in computer science from The Ohio State University (2009). He is currently a postdoctoral research associate at the Scientific Computing and Imaging (SCI) Institute at the University of Utah. His research interests include mesh generation, geometric modeling, computational geometry, and scientific visualization. He is a member of the IEEE and the ACM.

Luis Gustavo Nonato received the PhD degree from the Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, in 1998. He joined the Universidade de So Paulo in 1999 and has been an associate professor since 2006. His research interests include visualization and geometry processing. His teaching activities include geometry processing, numerical analysis, and geometric modeling. He is a member of the IEEE.

Valerio Pascucci received the EE laurea (master's) degree from the University La Sapienza in Rome, Italy, in December 1993, as a member of the Geometric Computing Group, and the PhD degree in computer science from Purdue University in May 2000. He is a faculty member at the Scientific Computing and Imaging (SCI) Institute at the University of Utah. Before joining SCI, he served as a project leader at the Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, (from May 2000) and as an adjunct professor at the Computer Science Department of the University of California, Davis (from July 2005). Prior to his tenure at CASC, he was a senior research associate at the University of Texas at Austin, Center for Computational Visualization, CS, and TICAM Departments. He is a member of the IEEE.