Edge Maps: Representing Flow with Bounded Error

Harsh Bhatia* SCI Institute, Univ. of Utah
Shreeraj Jadhav* SCI Institute, Univ. of Utah
Peer-Timo Bremer† Lawrence Livermore National Lab
Guoning Chen* SCI Institute, Univ. of Utah
Joshua A. Levine*, *Member, IEEE* SCI Institute, Univ. of Utah
Luis Gustavo Nonato‡ Universidade de São Paulo, Brazil
Valerio Pascucci*, *Member, IEEE* SCI Institute, Univ. of Utah

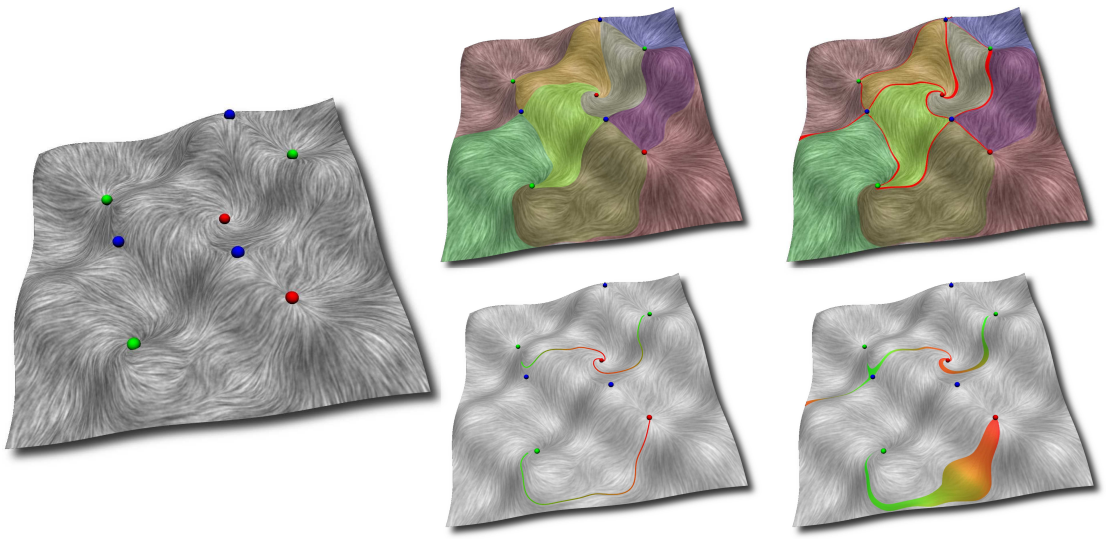


Figure 1: Edge maps enable new views of vector field stability, illustrated with a vector field on this wavy surface. Top row (middle right): A visualization of some colored regions where flow shares the same source (green spheres) and sink (red spheres) is augmented to show how these regions overlap when error is introduced. Bottom row (middle right): Streamwaves (colored green to red as they grow) show the advection of a single particle. In the presence of error, waves can widen and narrow, and bifurcate or merge.

ABSTRACT

Robust analysis of vector fields has been established as an important tool for deriving insights from the complex systems these fields model. Many analysis techniques rely on computing streamlines, a task often hampered by numerical instabilities. Approaches that ignore the resulting errors can lead to inconsistencies that may produce unreliable visualizations and ultimately prevent in-depth analysis. We propose a new representation for vector fields on surfaces that replaces numerical integration through triangles with linear maps defined on its boundary. This representation, called edge maps, is equivalent to computing all possible streamlines at a user defined error threshold. In spite of this error, all the streamlines computed using edge maps will be pairwise disjoint. Furthermore, our representation stores the error explicitly, and thus can be used to produce more informative visualizations. Given a piecewise-linear interpolated vector field, a recent result [15] shows that there are only 23 possible map classes for a triangle, permitting a concise description of flow behaviors. This work describes the details of computing edge maps, provides techniques to quantify and refine edge map error, and gives qualitative and visual comparisons to more traditional techniques.

Keywords: Vector Fields, Error Quantification, Edge Maps.

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation; I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types; G.1.2 [Numerical Analysis]: Approximation—Approximation of surfaces and contours

*emails: {hbhatia, jadhav, chengu, jlevine, pascucci}@sci.utah.edu
†email: bremer5@llnl.gov
‡email: gnonato@icmc.usp.br

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 to 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 inherent issue of traditional numerical techniques, we start with a description of a 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. Computing properties that then 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. Naïve 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). Figure 2 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

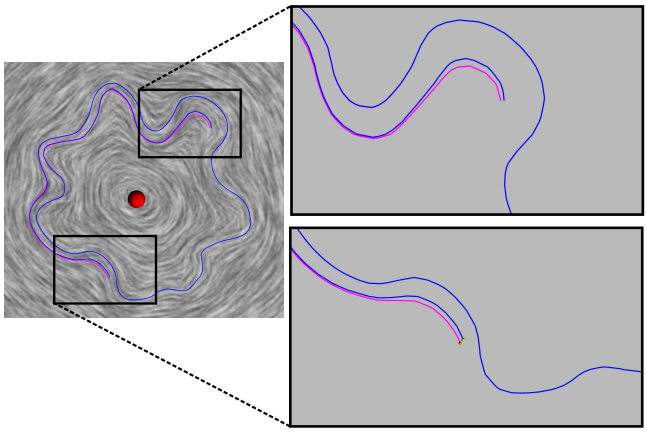


Figure 2: Left: Two streamlines are seeded traveling clockwise around this sink in a domain $[-1,1] \times [-1,1]$. Bottom: Initially, the magenta streamline is seeded outside of the blue streamline. Top: After integration with a step size of 0.025 the streamlines cross, now the magenta streamline is inside the blue streamline.

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. The resulting visualizations and analysis can cause inaccurate interpretations 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.

Considering these motivations, we propose a new data structure to represent vector fields called edge maps. 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, allowing the presentation of a more complete view of the data illuminating the major features that demonstrate where numerically unstable regions exist. By quantifying this error, we can refine 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. [10], 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 carry the error through 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.

Contributions

This paper describes a new data structure for storing the flow behavior of a vector field that does not rely on numerical integration. This structure is complementary to the traditional way of storing vector fields as piecewise linear interpolations over a mesh. Each triangle stores a map which encodes the inflow/outflow behavior over the boundary of the triangle. This allows us to replace the notion of integration with a different primitive: map lookup. Our contributions include:

- The definition of edge maps for triangulated 2D vector fields, and an algorithm to compute the approximate edge maps;
- Quantification of error bounds with this approximation;
- A refinement procedure for reducing mapping error; and
- New visualizations of flow instabilities using edge maps.

A more detailed discussion on the mathematical properties of edge maps and the possible configurations of flow within each triangle appeared in a recent technical report [15].

2 Related Work

Since vector-valued data is a natural way to represent fluid flow in simulations as well as other dynamical systems [13], 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 non-photorealistic rendering [5, 32] to mesh generation [1, 22]. 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. [17], following the lead of Nielson and Jung [21], proposed 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 step-wise 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 [12] 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 [31]. Three dimensional variants of the topological skeleton have also been proposed [9, 16, 28, 30]. The readers should refer to [8, 19, 27] 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 [3, 7, 14, 21, 25, 28]. As a result, some of the fundamental topological invariants of a vector field may not be preserved, such as, the Poincare-Hopf formula or the fact that streamlines are pair-wise 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. [2] introduce the ECG (Entity Connection Graph) as a more complete topological representation of vector fields on piece wise linear manifolds. By detecting closed streamlines, Wischgoll and Scheuermann [31] propose a technique

to detect limit cycles in planar vector fields. Scheuermann et al. [26] look for the areas of non-linear behavior in the field, and use higher order methods so that the features are not destroyed under linear assumption.

Recent work of Reininghaus and Hotz [23] construct a combinatorial vector field based on Forman's discrete Morse theory [6]. Using combinatorial fields allows the extraction of a consistent topological structure. However, combinatorial vector fields were limited by their high complexity, leading to later improvements to the algorithm [24]. While provably consistent, it is unclear how close the combinatorial field is to the original field. By comparison, this work proposes an integration technique that is both consistent and has error bounded with respect to the LEM.

3 Edge Maps

To address the issues of inconsistency, we propose an alternate representation called *edge maps*. In the following, we define edge maps and describe the elements that go into their construction.

3.1 Foundations

Let $\vec{V}: \mathscr{M} \to \mathbb{R}^2$ be a 2-dimensional vector field defined on a manifold $\mathscr{M}$. $\vec{V}$ is represented as the set of vector values sampled at the vertices of a triangulation of $\mathscr{M}$. 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 $\vec{V}(p_i), \vec{V}(p_j), \vec{V}(p_k)$. Figure 3(a) depicts the field defined in this way for a single triangle.

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 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 piecewise linear 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 non-zero, (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, which significantly simplifies the analysis of edge maps.

3.2 Definition

Let Δ be a triangle with boundary $\partial\Delta$. To understand and represent the flow behavior through Δ, we first summarize the formal definitions given in [15]. 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 on 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 of endpoints of streamlines through a triangle. To efficiently represent the edge maps, we merge adjacent origin points that have destination points which also adjacent. This merging helps approximate the point-to-point mapping as a mapping between connected subsets of the boundary of a triangle, called *intervals*. The interval obtained after 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. Figure 3(b) depicts the results of this merging process.

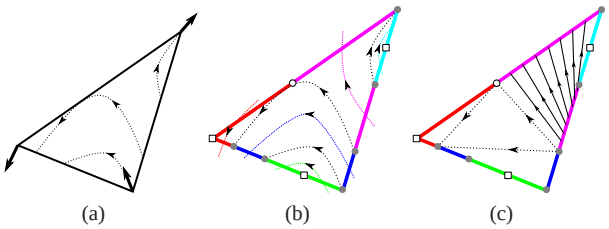


Figure 3: Edge map for a triangle. (a) Within a triangle, the vector is represented by interpolating three vectors at its vertices. (b) Our representation subdivides the boundary into a set of intervals, which map inflow to outflow for a triangle. (c) Given an entry point to a triangle, its corresponding exit point can be obtained by approximating the map linearly, thus replacing streamline integration with a step across the triangle.

3.3 Edge Map Generation

For practical purposes, we extend the definition of an edge map, to include critical points in the map. Such an extension facilitates streamline integration using edge maps, as discussed later. 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, 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.

We note that the edge map ξ as defined in [15] 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 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.

An edge map (forward and backward) can be encoded concisely as a collection of links of a triangle, such that the intervals are non-intersecting other than at their end points, and covers the entire boundary of the triangle (see Figure 3(c)). 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 pairs of intervals.

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: (i) *vertices* of the triangle; (ii) images of vertices (Figure 4(a)); (iii) *transition points*: points where the flow changes between inflow and outflow (Figure 4(b)); (iv) images of transition points and (v) *sepx points*, where the separatrices of a saddle exit or enter (Figure 4(c)).

Figure 6 gives the algorithm for computing the edge map for a triangle without a critical point. The advection of vertices and transition points are done using LEM [21]. When the triangle has a

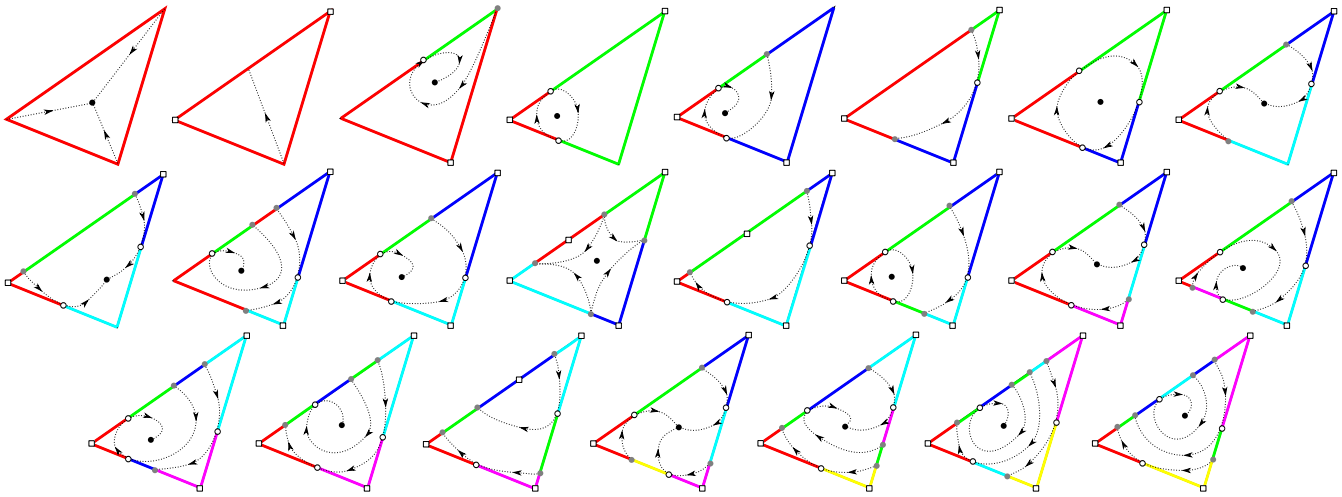
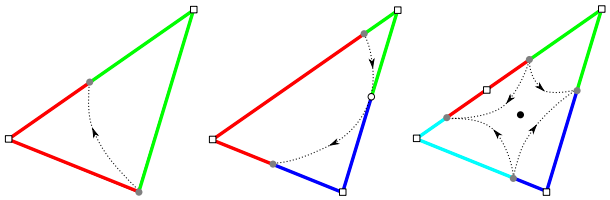


Figure 5: The 23 equivalence classes of edge maps for piecewise linear flow.They are ordered left-to-right, top-to-bottom by the number of links that exist in each edge map. Each link is assigned a different color.



(a) (b) (c)

Figure 4: Splitting of the boundary into intervals: (a) A triangle with a forward vertex image (grey 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 (grey dots); (c) A triangle with a saddle point (black dot), its four sepx points (grey dots), and a transition point from external flow (white square). Note that in most case vertices also act as external transition points.

critical point (detected using [20]), the algorithm is similar except that there can be additional cuts (from separatrices) and the critical point itself can act as an interval.

ConstructEdgeMap(Δ):

1. Identify the transition points on $\partial\Delta$. If necessary advect it forward and backward to find its images.

   (There can be at most 6 transition points in a triangle: 1 per edge and 1 per vertex. [15])

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 $\partial\Delta$ 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.)

Figure 6: Algorithm for creating the edge map for a triangle.

3.4 Streamline Integration using Edge Maps

The encoding of flow as edge maps ultimately allows us to determine structural properties of the flow through the triangle in a fast manner. 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 going to the same destination intervals they would have in the original piecewise linear flow.

In particular, particle trajectories can be approximated on a per-link level by linearly interpolating between the source and destination intervals. As a result, this approach gives a method to approximate streamlines. As Figure 3(c) shows, for a point on any source interval, we can approximate the path to its destination by linearly interpolating where it lies in the origin interval and projecting that point to the same coordinate in the destination interval.

Using the precomputed edge maps, any numerical integration to calculate the streamlines or particle propagation (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

$$x_{n+1} = \xi^+(x_n)$$

Hence, edge maps are faster, and as discussed in Section 4.1, have a bounded error that can be explicitly computed.

3.5 Equivalence Classes of Edge Maps

In a companion report [15], we show that there exist 23 equivalent classes of edge maps for linearly varying flow, see Figure 5. Here, equivalence is defined as invariance under rotation of triangle and inversion of flow. Exploiting the fact that piecewise linear flow can switch between inflow and outflow only once per edge, one can show that the boundary of a triangle can be broken up into at most eleven intervals, with a potential critical point acting as a twelfth interval. To understand topological equivalence, splits caused by vertices and their images (see Figure 4(a)) are discounted. The linearity of the flow implies that these intervals may connect themselves into, only in a limited number of ways to create a valid edge map. For a more descriptive discussion on equivalence, and to understand the rules for validating edge maps based on the properties of linear vector fields, the reader is encouraged to refer to [15].

Since the number of classes is limited, the overhead for storing the edge maps of a single triangle is both bounded and relatively low. We observe that only four of the classes do not contain a critical point. Therefore, the vast majority of triangles in our experiments could be classified as one of these four types.

4 Error Propagation Using Edge Maps

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 [11]. Since typically the true streamline is not known, this error cannot be quantified explicitly. While some schemes are more accurate than others and highly 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 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 the integration with a one dimensional barycentric mapping that guarantees non-intersecting streamlines and consistency between forward and backward traces up to the floating point precision of the linear interpolation.

4.1 Mapping Error

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 end-points 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. Since the edge map approximates the true exit point q of a point p by linearly interpolating within the link as q', it incurs some error. This error can be calculated as the deviation of the exit point given by the map, from that given by the exact method ($\|q - q'\|$). We have derived the expression for this mapping error in a link, $e(\lambda)$, shown as Equation 1 in Appendix A (as a function of the arc length parameter of the link, λ).

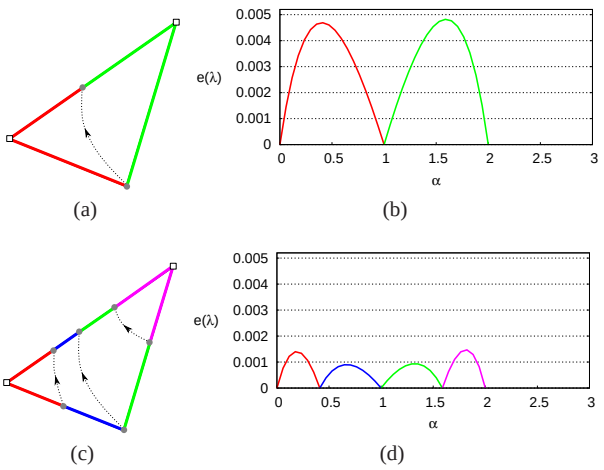


Figure 7: Mapping error in edge maps. The red and green error curves in (b) show the mapping error in red and green links of the triangle in (a). The mapping error is drawn as a function of arc-length parameter of the triangle, α, counter-clockwise from the bottom left vertex. For $2.0 < \alpha < 3.0$, there is no mapping error, since this segment of the triangle is acting as a destination. The average edge-length of the triangle in consideration is 0.025. A refined map (c) contains much smaller mapping error (d) when refined to $\delta = 0.0025$ as compared to the basic map. During refinement, both links in (a) get split into two links each.

Figure 7 shows one such graph of the mapping error as a function of the arc-length parameter of the triangle, α. We use the maximum error value of error in a link as the *mapping error of the link*, ε. An upper limit to this mapping error can be imposed by a user parameter, δ. If for a link, $\varepsilon > \delta$, we split the link to improve the accuracy of the map. We call this process *refinement of edge maps*. Figure 7(c) shows refined map of the flow in Figure 7(a) flow with $\delta = 0.0025$.

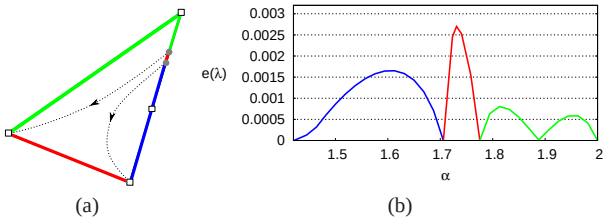


Figure 8: An example of a map with bimodal error (in green link). Note that the x-axis has been scaled to range $1.4 \leq \alpha \leq 2.0$ to illustrate the mapping error. For other values of α, the error is zero, since those segments are destinations.

In our experiments, we found that typically the error is unimodal in a link. However, the error can also be bimodal, as shown in Figure 8. Unsurprisingly, the mapping error is zero in the absence of divergence, since there is no linear expansion of flow, in the direction orthogonal to the flow. Figure 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.

4.2 Expansion of Exit Points

We have been using the forward (ξ^+) and backward (ξ^-) edge maps as tools to look up the streamline of individual particles. However, we can also represent the mapping 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 Figure 9. The length of the segment Q is directly proportional to the expansion factor. Thus, we call Q the *expansion* of the exit point q.

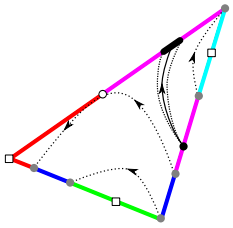


Figure 9: Expansion of exit points represents error as a range of possible destinations.

The mapping 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 $\omega = \varepsilon$ 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 can not span across links and thus is truncated at the endpoints of the link containing both p and q.

4.3 Streamwaves

Under the consideration of mapping error, edge maps no longer describe a one-to-one, but a one-to-many mapping. We define a *streamwave* as set of possible destinations that a massless particle may reach at least once when accounting for possible errors. Alternatively, a streamwave can be seen as the expansion of a streamline due to mapping uncertainties. In the current work, we quantify

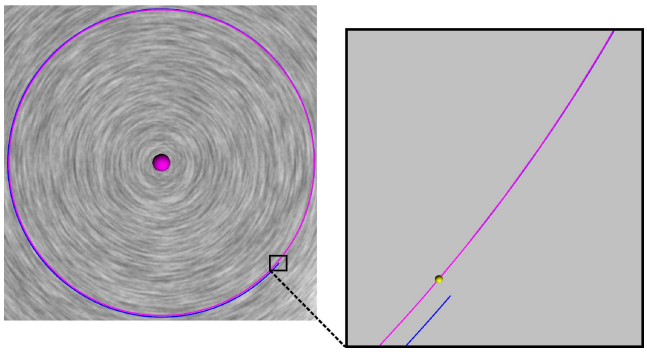


Figure 10: Comparison between propagation using RK45 (blue) and edge maps (magenta) on a vector field defined by a counter-clockwise orbit seeded at the same point (yellow). The magenta and blue lines overlap in the beginning but a substantial deviation in RK45 streamline is observed after only one revolution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

and visualize the mapping error as streamwaves propagate. However, any other kind of error can be modeled as the expansion ω for streamwaves.

Using the 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. Note that, in this form the speed of the wave is determined by the number of triangles that are processed rather than the velocity of the flow. 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 wave is only as expensive as the number of triangles currently at the front and independent of the flow complexity within each triangle.

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 the intermediate points are handled implicitly. For triangles with bifurcation, a streamwave may split into two streamwaves, each of which can be propagated independently.

As shown in Figure 11, 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 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 visually show error bounds of other integration techniques. For example, Figure 12 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 visually show a comparison between Euler integration, fourth-order Runge-Kutta, and the local exact method.

5 Visualization of Fuzzy Topology

Topological structures in vector fields, such as their topological skeleton [12], are one of the key features used 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 eigen-

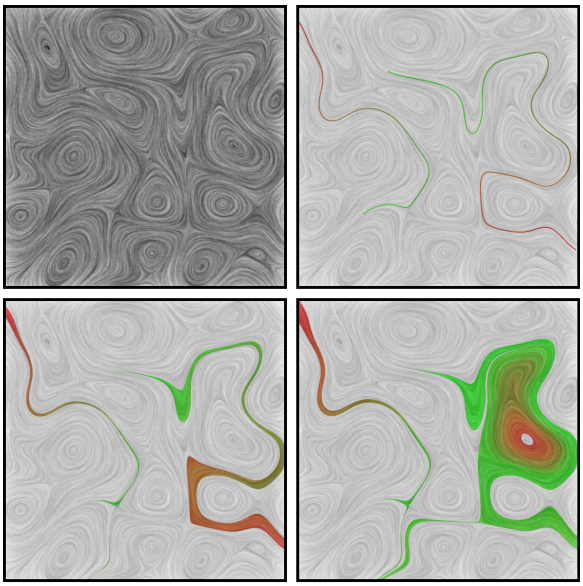


Figure 11: Visualizations of streamwaves. Top left: original vector field visualized with IBFV [29]. Top right: two streamlines, one near a saddle's separatrix. Bottom: Two images show streamwaves at different levels of error. Streamwaves are colored from green to red, showing distance the flow has propagated as a measure of the number of maps the streamwave has travelled through. Note that the error levels have been exaggerated to illustrate expansion and bifurcation of streamwaves.

vectors 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 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 [4]. These streamwaves are initiated from the boundary of the triangles containing the critical points. While we cannot yet 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 covering of domain similar to the segmentation induced by traditional vector field topology. However, in our construction each point in the domain may be part of several streamwaves creating a notion of *fuzzy* topology as shown in Figure 1. Figure 13 shows an additional example of fuzzy topology computed on a diesel engine dataset, indicating the swirling structure around its surface. 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 Figure 14. Laney et al. [18] 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 dataset, 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.

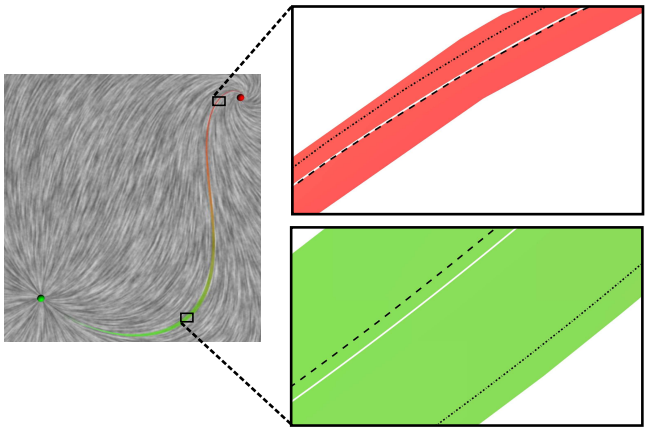


Figure 12: A streamline using RK4 (black dashed) using stepsize $\Delta t = 0.005$, Euler (black dotted) using stepsize $\Delta t = 0.005$ and local exact method (LEM) (white solid), and a streamwave using edge maps with $\delta = 0.0001$ were seeded at the same point. Considering the local exact method 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.

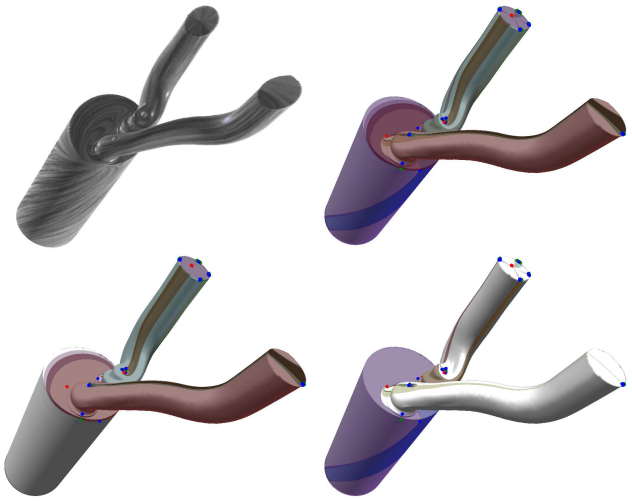


Figure 13: Topology of the diesel engine dataset. Top row: An IBFV rendering of the flow (left) and visualization of the topology (right). Bottom row: the stable and unstable manifolds.

6 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 and the notion of fuzzy topology. 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.

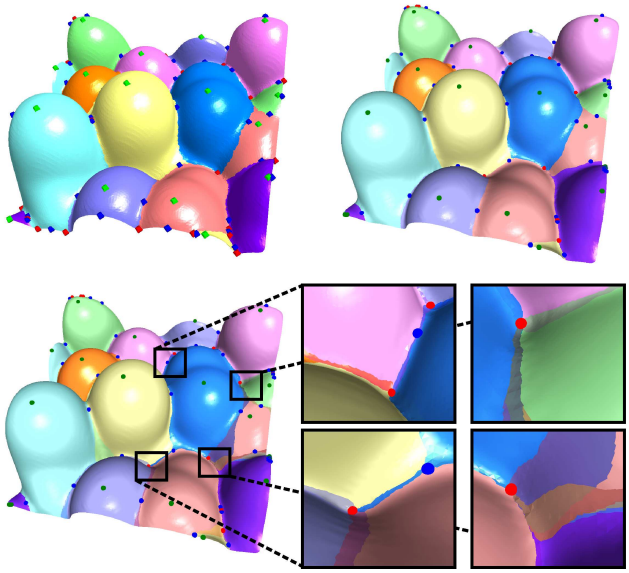


Figure 14: Visualizations of a Rayleigh-Taylor instability. Top row: We reproduce the results from Laney et al. [18] (left) side-by-side with our edge map computation of the unstable manifolds (right). Bottom row: when the error factor is increased (left) we can observe the emerging overlaps (right).

In this work, 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 of the interior of cells will be required and the types of flow behaviors shown in [15] will need to be redefined. However, on a conceptual level of replacing integration with a boundary mapping, the idea of edge maps is both extendible as well as applicable to different discretizations of domain.

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.

Acknowledgements

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 U.S. Department of Energy 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. We are grateful to Jackie Chen for the dataset from Figure 11, Robert S. Laramee for the diesel engine dataset from Figure 13, and Paul Miller, William Cabot, and Andrew Cook for the bubbles dataset from Figure 14. Attila Gyulassy and Philippe P. Pebay provided many useful comments and discussions. LLNL-PROC-463631.

References

[1] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77, 2009.

[2] G. Chen, K. Mischaikow, R. S. Laramee, P. Pilarczyk, and E. Zhang. Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Trans. Vis. Comput. Graph.*, 13(4):769–785, 2007.
[3] G. Chen, K. Mischaikow, R. S. Laramee, and E. Zhang. Efficient Morse decompositions of vector fields. *IEEE Trans. Vis. Comput. Graph.*, 14(4):848–862, 2008.
[4] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry*, 30(1):87–107, 2003.
[5] M. Fisher, P. Schröder, M. Desbrun, and H. Hoppe. Design of tangent vector fields. *ACM Trans. Graph.*, 26(3):56, 2007.
[6] R. Forman. A user's guide to discrete morse theory. In *Proc. of the 2001 Internat. Conf. on Formal Power Series and Algebraic Combinatorics*, page 48, 2001.
[7] C. Garth, H. Krishnan, X. Tricoche, T. Tricoche, and K. I. Joy. Generation of accurate integral surfaces in time-dependent vector fields. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1404–1411, 2008.
[8] C. Garth and X. Tricoche. Topology- and feature-based flow visualization: Methods and applications. In *SIAM Geom. Des. & Comp.*, 2005.
[9] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *IEEE Visualization*, pages 33–41, 1991.
[10] A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of morse-smale complexes for three-dimensional scalar functions. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1440–1447, 2007.
[11] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics, 1993.
[12] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, 1989.
[13] M. W. Hirsch, S. Smale, and R. L. Devaney. *Differential Equations, Dynamical Systems, and An Introduction To Chaos*. Elsevier Academic Press, 2nd edition, 2004.
[14] J. P. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *Proc. of IEEE Vis. '92*, pages 171–178, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.
[15] 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. Technical Report UUSCI-2010-004, Scientific Computing and Imaging Institute, December 2010.
[16] K. M. Janine, J. Bennett, G. Scheuermann, B. Hamann, and K. I. Joy. Topological segmentation in three-dimensional vector fields. *IEEE Trans. Vis. Comput. Graph.*, 10:198–205, 2004.
[17] P. Kipfer, F. Reck, and G. Greiner. G.: Local exact particle tracing on unstructured grids. *Computer Graphics Forum*, 22:133–142, 2003.
[18] 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 TVCG.*, 12(5):1053–1060, 2006.
[19] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post. Topology Based Flow Visualization: The State of the Art. In *Topology-Based Methods in Vis.*, Mathematics and Visualization, pages 1–19. Springer, 2007.
[20] Y. Lavin, R. Batra, and L. Hesselink. Feature comparisons of vector fields using earth mover's distance. In *Proc. of IEEE/ACM Visualization '98*, pages 103 –109, Oct. 1998.
[21] G. Nielson and I.-H. Jung. Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE Trans. Vis. Comput. Graph.*, 5(4):360 –372, oct. 1999.
[22] N. Ray, B. Vallet, W.-C. Li, and B. Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2):10, 2008.
[23] J. Reininghaus and I. Hotz. Combinatorial 2d vector field topology extraction and simplification. In *Topological Methods in Data Analysis and Visualization.*, 2009.
[24] J. Reininghaus, C. Löwen, and I. Hotz. Fast combinatorial vector field topology. *IEEE Trans. Vis. Comput. Graph.*, 99, 2010.
[25] G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, K. I. Joy, and W. Kollmann. A tetrahedra-based stream surface algorithm. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 151–158, Washington, DC, USA, 2001. IEEE Computer Society.
[26] G. Scheuermann, H. Krüger, M. Menzel, and A. P. Rockwood. Visualizing nonlinear vector field topology. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):109–116, 1998.
[27] G. Scheuermann and X. Tricoche. Topological methods in flow visualization. In C. Hansen and C. Johnson, editors, *In Visualization Handbook*, pages 341–356. Elsevier, 2004.
[28] 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. In *IEEE Visualization*, pages 225–232, 2003.
[29] J. J. van Wijk. Image based flow visualization. *ACM Trans. Graph.*, 21(3):754–754, 2002.
[30] 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. In *IEEE Visualization*, pages 559–566, 2005.
[31] T. Wischgoll and G. Scheuermann. Detection and visualization of closed streamlines in planar flows. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):165–172, 2001.
[32] E. Zhang, K. Mischaikow, and G. Turk. Vector field design on surfaces. *ACM Trans. Graph.*, 25(4):1294–1326, 2006.

APPENDIX

A MAPPING ERROR

In [17], the motion of a particle under a linear vector field $\vec{V}(x) = Ax + o$ is defined as

$$x(t) = e^{(t-t_0)A}x_0 + (e^{(t-t_0)A} - Id)A^{-1}o$$

where, x_0, t_0 give the particle's initial position and time, and $x(t)$ gives the position after time t. A is a 2×2 matrix, o is the translation vector, and Id is the identity matrix.

Consider a link, in which origin interval (a,b) flows to destination interval (c,d). Point a flows to point c in time $t(a)$. Similarly, point d is calculated as the true destination of point b, reached in time $t(d)$.

$$c(a,t(a)) = e^{t(a)A}a + (e^{t(a)A} - Id)A^{-1}o$$
$$d(b,t(b)) = e^{t(b)A}b + (e^{t(b)A} - Id)A^{-1}o$$

Now, consider a point x on the src interval, whose true destination is given by y.

$$y(x,t(x)) = e^{t(x)A}x + (e^{t(x)A} - Id)A^{-1}o$$

Suppose, $x = \lambda a + (1-\lambda)b$. We can interchangebly use $y(x)$ and $y(\lambda)$, and $t(x)$ and $t(\lambda)$.

The map gives x's destination as y', and the mapping error is calculated as $e(\lambda) = y'(\lambda) - y(\lambda)$. The map interpolates the destination interval to approximate the destination of x.

$$\begin{aligned} y'(\lambda) &= \lambda c + (1-\lambda)d \\ &= \lambda e^{t(a)A}(a + A^{-1}o) + (1-\lambda)e^{t(b)A}(b + A^{-1}o) - Id\ A^{-1}o \end{aligned}$$

Calculating the deviation between the two,

$$\begin{aligned} e(\lambda) &= y'(\lambda) - y(\lambda) \\ &= \lambda e^{t(a)A}(a + A^{-1}o) + (1-\lambda)e^{t(b)A}(b + A^{-1}o) - Id\ A^{-1}o \\ &\quad - \{e^{t(x)A}x + (e^{t(x)A} - Id)A^{-1}o\} \\ &= \lambda e^{t(a)A}(a + A^{-1}o) + (1-\lambda)e^{t(b)A}(b + A^{-1}o) \\ &\quad - e^{t(x)A}(x + A^{-1}o) \\ &= \lambda e^{t(a)A}(a + A^{-1}o) + (1-\lambda)e^{t(b)A}(b + A^{-1}o) \\ &\quad - e^{t(\lambda)A}(\lambda a + (1-\lambda)b + A^{-1}o) \end{aligned} \quad (1)$$

The maximum length of this deviation of $e(\lambda)$ over every link is assigned as the mapping error of the link

$$\varepsilon = max_{0\le\lambda\le 1}\ ||e(\lambda)||_2$$