Eurographics Conference on Visualization (EuroVis) 2012
S. Bruckner, S. Miksch, and H. Pfister
(Guest Editors)

Volume 31 (2012), Number 3

A Quantized Boundary Representation of 2D Flows

J. A. Levine[1], S. Jadhav[1], H. Bhatia[1,2], V. Pascucci[1], and P.-T. Bremer[1,2]

[1]Scientific Computing and Imaging Institute, University of Utah, USA
[2]Lawrence Livermore National Laboratory, USA

Abstract

Analysis and visualization of complex vector fields remain major challenges when studying large scale simulation of physical phenomena. The primary reason is the gap between the concepts of smooth vector field theory and their computational realization. In practice, researchers must choose between either numerical techniques, with limited or no guarantees on how they preserve fundamental invariants, or discrete techniques which limit the precision at which the vector field can be represented. We propose a new representation of vector fields that combines the advantages of both approaches. In particular, we represent a subset of possible streamlines by storing their paths as they traverse the edges of a triangulation. Using only a finite set of streamlines creates a fully discrete version of a vector field that nevertheless approximates the smooth flow up to a user controlled error bound. The discrete nature of our representation enables us to directly compute and classify analogues of critical points, closed orbits, and other common topological structures. Further, by varying the number of divisions (quantizations) used per edge, we vary the resolution used to represent the field, allowing for controlled precision. This representation is compact in memory and supports standard vector field operations.

Categories and Subject Descriptors (according to ACM CCS): 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

1. Introduction

Vector fields are widely used in science and engineering applications to model flow in domains as varied as the simulation of combustion, aerodynamics, climate, and high energy physics. Regardless of the application a robust analysis and faithful visualization are crucial to understand, manipulate, and describe complex flow structures. Currently, there exist two broad classes of techniques for visual analysis of vector fields. One class of approaches derives various scalar quantities from the flow, e.g. vorticity [SPP04, HRAW07] or finite time Lyapunov exponents [Hal01, SP09]. The resulting fields can then be analyzed using well known scalar-based techniques. The main drawback is that during the transformation some information is lost and many features of interest such as closed orbits or stable and unstable manifolds are difficult or impossible to extract from the scalar reduction.

Alternatively, topology-based approaches have been developed that aim to describe the global flow structure through the behavior of its streamlines. The limit sets of streamlines—critical points and closed orbits—are of particular interest as is the *topological skeleton* of the flow which segments the domain into regions of uniform flow. Topological approaches are attractive as they produce an abstract yet complete description of the global flow behavior and thus form an ideal starting point for analysis as well as visualization. The theoretical foundations of this approach are well understood and have been a longstanding focus of a vibrant community [GLL91, HH89].

However, the application of these ideas to practical problems has proven challenging. The majority of theoretical results are based on smooth vector fields defined on smooth manifolds. In practice, one typically deals with sampled vector fields extended to a domain through interpolation, which are rarely smooth. Furthermore, to extract the flow, i.e. the streamlines, from a given vector field numerical integration techniques are used which can be highly accurate but are hardly ever exact. As a result, the flow constructed in this manner no longer fulfills the assumptions of the theory and can violate even basic invariants such as the fact that streamlines should never cross. Consequently, the remaining theory is no longer applicable preventing downstream analysis and severely restricting the effectiveness of such techniques.

This work is based on the insight that the challenge of constructing theory compliant flow can be split into two orthogonal problems. First, sampled vector fields have a limited smoothness based on interpolant (e.g. piecewise linear flow is only C^1 continuous), and thus the flow they define is not smooth. Second, the flow extracted from these vector fields uses numerical techniques that introduce various approximation errors. We address the second problem by in-

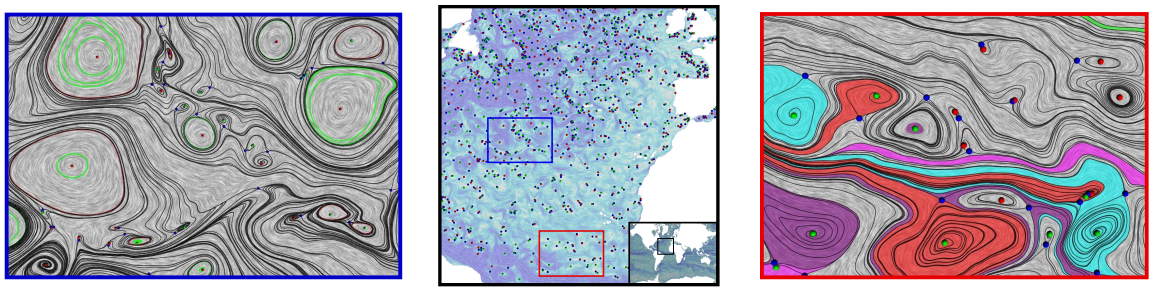


Figure 1: *Oceanic currents of the North Atlantic. We show a* 600×600 *tile from the larger simulation (center). Each image on the side is a zoomed view visualizing the topology for the tile. (Left) Black lines are separatrices grown from all saddles (blue balls), and green curves show detected closed orbits. (Right) Different colored regions are unstable manifolds grown from all the sources (green balls).*

troducing a new representation that directly encodes the flow by efficiently storing a large but finite set of streamlines. Once constructed, this eliminates the need for any numerical integration, allowing for algorithms that guarantee the theoretical consistency of all streamlines. Given a consistent set of streamlines, the first problem of smoothness can be overcome by a careful adaptation of theory in the spirit of recent scalar field approaches addressing the corresponding problem [GNP*06, LBM*06]. The resulting vector field representation is guaranteed to be consistent with the theory and supports the combinatorial computation of all critical points, closed orbits, and the topological skeleton.

Contributions We propose a new representation, called *quantized flow*, that supports discrete computations like the approaches discussed above while enabling a multi-resolution approximation of a given piecewise linear (PL) vector field. Instead of encoding the vector field and extracting its streamlines, the flow is quantized directly by efficiently storing a large but finite number of streamlines (our implementation scales up to 2^{32} streamlines passing through every edge). We accomplish this using a structure similar to the edge maps of [BJB*12, JBB*12] where only the intersections of streamlines with mesh edges are represented. However, unlike [BJB*12] quantized flow does not rely on numerical critical point detection and classification nor does it employ a floating point mapping between edges which is susceptible to round-off errors. Specifically, quantized flow:

A. Provides a combinatorial representation of flow that still maintains a geometric fidelity to an input field;
B. Is constructed with an embarrassingly parallelizable algorithm using any given streamline computation;
C. Supports user-defined levels of approximation which control the level of geometric fit; and
D. Utilizes a compact, memory efficient data structure.

These properties enable a unified setting for:

1. Computing consistent streamlines combinatorially;
2. Detecting and classifying all critical points using a discrete equivalent of the Poincaré index;
3. Progressively detecting and classifying closed orbits; and
4. Extracting and visualizing the entire topological skeleton.

2. Related Work

Starting with Helman and Hesselink's seminal paper [HH89], vector field topology has been a staple of vector field visualization. Helman and Hesselink define a two-dimensional vector field's *topological skeleton* as a graph constructed using a special set of streamlines, i.e. *separatrices* that connect the critical points of the field. Separatrices are the four streamlines that (asymptotically) travel to and from each saddle point and a large number of approaches exist to approximate such structures in both two- and three-dimensional vector fields using numerical integration [GLL91, TSH01, TWHS03]. Similar techniques have been extended to multi-resolution representations as well as time-dependent flows [GTS04, TWHS05].

The topological skeleton described above provides only a partial picture of a flow's topological behavior. In particular, there can exist periodic orbits not connected to saddles which cannot be detected by following separatrices. Notable exceptions that augment the skeleton with periodic orbits have been proposed in two [LI07, TWHS04, WS01] and three [WS02] dimensions. The topological skeleton as well as the orbits describe the boundaries between regions of uniform flow and thus, by definition, consist of the most unstable streamlines. Consequently, it is well known that computing a valid topological skeleton can be numerically unstable due to errors inherent in the integration and potential inconsistencies among neighboring triangles [CMLZ08].

To address the problems introduced through numerical integration, several schemes have been developed to discretize vector fields in various ways. Reininghaus et al. [RLH11] convert PL vector fields into discrete vector fields according to Forman's definition [For98]. They represent flow as pairs of mesh elements, e.g. vertex-edge or edge-triangle pairs. Topological structures can be efficiently extracted through graph traversals. However, discrete vector fields no longer correspond to traditional interpolation and may lose much of their geometric information. Furthermore, certain elements such as rotating critical points cannot be represented directly. Thus, the benefit of combinatorial robustness comes at a high cost. The geometric resolution of a discrete vector field is limited by the mesh resolution, and it is unclear how well a given PL vector field can be approximated.

Chen et al. [CMLZ08] create a triangle level graph by determining how the image of a triangle is advected and deformed by the flow. While the graph is computed using the underlying PL vector field, the accuracy of this technique remains limited by the triangle resolution. However, their technique enables classification of Morse sets via Conley indices [CDS*12], which provides meaningful information, albeit at a coarse granularity. Szymczak and Zhang [SZ12] focus on piecewise constant (PC) vector fields, and extract a similar structure, the *transition graph* that subsequently can be used to extract topological structures. The transition graph can be seen as a PC equivalent of the link graph used in Section 6.1 to extract orbits from PL vector fields. The graph provides a more generic refinement mechanism. Either the mesh may be refined (similar to Chen et al. [CMLZ08]), or the graph may be refined by splitting nodes. What is missing is a notion of when split nodes converge to an input field in a geometric sense. Szymczak later extends this work to allow the representation of sets of vectors for each triangle [Szy11] and model the stability of the computation.

Our goal is to have the benefits of the combinatorial approaches described above, but allow for a structure that captures a geometric approximation of the flow. We accomplish this by relying on the edge maps of Bhatia et al. [BJB*12, JBB*12] which stores the intersections of streamlines with mesh edges. While edge maps discretize the mapping between edges, they do not discretize the flow at a streamline level. As a result, it is difficult to compute edge maps without a strong set of assumptions on the input. Moreover, it is challenging to compute closed orbits robustly with edge maps alone, as numerical instabilities and finite precision arithmetic can still cause inconsistencies. However, Bhatia et al. describe how this structure may be used for bounded error refinement [BJB*12] with respect to an input PL flow—a feature that quantized flow also provides.

3. Vector Fields as Quantized Boundary Maps

Consider a 2-dimensional vector field $\vec{V}\colon \mathbb{M} \to \mathbb{R}^2$. For simplicity, we assume $\mathbb{M} \subset \mathbb{R}^2$ but all concepts extend to embedded, two-dimensional manifolds. Interpreting $\vec{V}$ as a steady velocity field, it implicitly describes the parametric path that massless particles travel. The *flow*, $\Phi(x,t)$, is given by the solution of the differential equation $\frac{d\Phi(x,t)}{dt} = \vec{V}(x)$ with the initial condition $\Phi(x,0) = x_0$. Fixing a point x_0 and varying t, we call the set of points $\Phi(x_0,t)$ a *streamline*.

Typically, a subset of $\mathbb{R}^2$ is represented by a triangulation $T = (V,E,C)$ with the set of vertices V, edges E, and triangles C. There currently exist two popular approaches to define a vector field on T. Either a set of sample vectors, e.g. $\vec{V}(v_i) = (x_i, y_i)$, $v_i \in V$, is provided and extended by (generally, PL) interpolation to $\mathbb{M}$. Alternatively, a discrete vector field can be defined as a set of simplex pairs, e.g. (v_i, e_j) and (e_k, c_l), where each pair represents "flow" from the lower dimensional to the higher dimensional simplex. Instead, we follow a third convention based on encoding the flow Φ of $\vec{V}$ directly rather than representing $\vec{V}$.

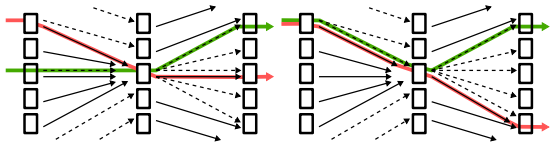


Figure 2: *Quantized streamlines using ξ^+ indicated by the solid and ξ^- by the dashed arrows. ξ^+ and determine the quantized source–destination pairs. (a) The red streamline (using ξ^+) intersects the green streamline (using ξ^-). (b) Adjusted $\xi^\pm$ pairing picks the rightmost source/destination wrt the tracing direction. The forward and backward streamlines share a bin but do not intersect.*

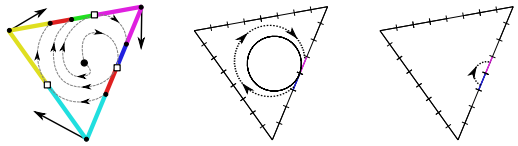
3.1. Sampling Flow

Given a sampled vector field the flow is typically constructed by numerically integrating streamlines. Conceptually, this approach describes infinitely many streamlines for a triangle. Instead, we encode a subset of representative streamlines. These *quantized streamlines* encode an approximation of the infinite continuum of flow. Quantized streamlines are defined based on a discrete set of sample points $\{p_0, p_1, \ldots, p_n\}$ on each edge of the triangulation. Let us assume that at each of these samples $\vec{V}$ is not parallel to the corresponding edge, i.e. each sample can be classified into inflow or outflow with respect to a given triangle. Section 4.1 will discuss enforcing this in practice. The streamline S_i for any inflow sample p_i either leaves the triangle at an outflow point o or approaches a critical point c on the interior. A quantized streamline disregards the path of S_i and instead stores a pair (p_i, p_j) or (p_i, c) for S_i, where p_j is the sample point closest to o. We call these *source* and *destination* pairs.

3.2. Quantized Streamlines

Given a source and destination for each sample we define a *quantized streamline* as the sequence of samples reached by following the destination maps forward and the source maps backward. However, in order to use quantized streamlines as a stand-in for smooth streamlines we must ensure that they observe the same set of invariants most importantly the fact that streamlines do not intersect. Using only the above definition of source and destination does not maintain this invariant. While each sample has exactly one source and one destination, multiple samples can have the same source/destination and the source-destination maps are not necessarily inverse to each other. Therefore, in areas of sufficient divergence, quantized forward streamlines can cross backward lines creating an inconsistent flow (Figure 2(a)).

To avoid this problem we modify the mapping slightly and define the forward map $\xi^+(p)$ of a sample p by considering all pairs (p, q_i). We define $\xi^+(p)$ as the rightmost (relative to $\vec{V}$) sample q_i. Symmetrically, we define the backward map $\xi^-(p)$ as the leftmost sample among all pairs (q_i, p), see Figure 2(b). Using the modified maps ensures that quantized streamlines may share samples (equivalent to becoming infinitely close) but never cross. This builds quantized streamlines that are a provably consistent approximation of Φ.



(a) (b) (c)

Figure 3: *(a) Edge map of a spiraling source. Transition points (white squares) and their images (black dots) form the colored intervals. (b) An interior orbit touching an edge creates an ITP whose boundary flow (c) appears as an ETP.*

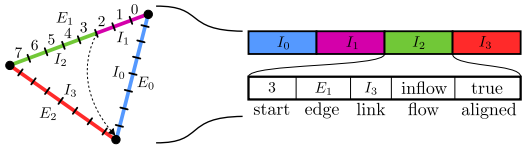


Figure 4: *The map of a triangle (CCW orientation) is stored as a linked list of intervals. Each interval stores a k-bit value ($k = 3$ here) for the beginning bin of the interval and a two byte descriptor.*

4. Vector Field Conversion

Using quantized streamlines to approximate a vector field has several advantages in terms of consistency and ease of use. However, storing the $\xi^{\pm}$ maps naively quickly becomes infeasible for all but the coarsest representations. We introduce a simple data structure to efficiently approximate a large number of quantized streamlines and a straightforward and parallelizable algorithm to convert PL vector fields into the new representation.

4.1. Quantized Edge Maps

To encode quantized streamlines efficiently we exploit the fact that the $\xi^{\pm}$ maps represent an approximation of Φ. In particular, the $\xi^{\pm}$'s can be seen as a quantized version of the edge maps recently introduced by Bhatia et al. [BJB*12]. They show that the boundary map induced by Φ is continuous almost everywhere and naturally partitions triangle boundaries into open *intervals* that map to either interior critical points or other intervals. The boundaries of intervals are either *transition points* at which the flow is parallel to an edge, sources/destinations of transition points (*image points*), or intersections with separatrices called *sepx points*, see Figure 3(a). Furthermore, they classify transition points into *internal* and *external* transition points (ITPs and ETPs) based on the local flow geometry.

However, edge maps rely on some inconvenient assumptions such as the fact that all critical points of $\vec{V}$ must lie on the interior of triangles. These are difficult to enforce in practice and prevent the modeling of higher order critical points. Furthermore, edge maps use barycentric coordinates to encode the geometry of intervals and linear interpolation for the mapping. Both operations introduce round off errors making it difficult to ensure consistency. We introduce *quantized edge maps* to adapt the concepts of edge maps for use with quantized flow. Quantized edge maps eliminate the need for most assumptions and numerical computations.

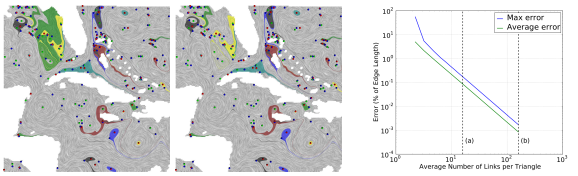
To discretize an edge we explicitly quantize it into $n = 2^k$ bins, with $k = 32$ unless otherwise noted. Each bin corresponds to the sample point at its center and in the remainder of the paper we use bins and samples interchangeably. To ensure that each bin can be classified into inflow and outflow we only require that no edge is entirely parallel to $\vec{V}$ or conversely that all transition points are isolated. Note, that this condition is weaker than those of the original edge maps and allows the existence of critical points on both edges and vertices. Given that each bin represents either inflow or outflow all transition points exist on the boundary of bins. Consequently, their sources and destinations must also lie on the boundary of bins. Furthermore, the construction algorithm of Section 4.2 guarantees that a sepx point can also exist only between bins. This allows the $\xi^{\pm}$ maps to be stored efficiently by encoding the mapping between intervals.

A sequence of bins representing an interval is uniquely identified through two integers, its edge number and its first bin. Since intervals will be stored contiguously, their extent can be inferred by their neighbor. Mapping an interval with m bins onto an interval with n bins can be solved by any rasterization procedure, in our implementation we use Bresenham's algorithm [Bre65]. Given a bin $i < m$, we compute its destination by computing $n * (i/m)$ to the nearest integer. This computation can potentially produce a round off in which case we output $n * (i/m) + 1$. Note that while this rasterization emulates a linear interpolation, it is an entirely integer procedure. Additionally, the rasterization naturally creates consistent $\xi^{\pm}$'s in the sense of Section 3.2.

Finally, one can classify transition points into external and internal by analyzing the local maps. At an ETP the two bins on either side will be mapped to each other while at an ITP both bins will map to disparate locations. Note this classification disregards the paths streamlines would take. If there exists a closed orbit in the interior of a triangle which touches an edge, the corresponding point is an ITP by the original edge maps (Figure 3(b)). However, according to the local flow behavior, it is classified as an ETP. This discrepancy reflects the fact that quantized edge maps only store the flow behavior as it pertains to the boundary. Nevertheless, either classification is consistent with the given boundary map.

Data Structure For each triangle we store a circular linked list of intervals around the boundary. Each interval stores a k-bit integer as the index of the first bin for the interval. Additionally, two bytes per interval are bit-packed with the following information: 2-bits for the edge number within the face since bin numbering is only unique per edge; a 1-bit flag indicating inflow or outflow; 1-bit indicating the orientation of the interval, relative to the bin numbering; and 12-bits for the index of the paired source/destination interval.

Figure 4 shows a schematic of our quantized edge map datastructure. Given this information a streamline is traced through the triangle by finding its source/destination interval followed by a rasterization step to map the corresponding



(a) (b) (c)

Figure 5: *Map refinement at maximal error (wrt the edge length) of (a)* 0.18%*; and (b)* 0.023%*. The initial refinement causes a large change in the unstable region to the west of Florida (top-left of dataset). (c) Total mapping error as a function of average number of links shows well-behaved convergence as more links are used.*

bin. Across edges there exists a trivial map between bins and we find the corresponding interval through a binary search.

4.2. Converting Piecewise Linear Fields

One of the disadvantages of the original edge map representation is that it relies on constructing non-degenerate local flow numerically. Furthermore, edge maps are constructed by tracing transition points and separatrices, by definition the most unstable structures in the flow. Any numerical instabilities encountered during the construction can produce inconsistencies causing the conversion to fail. Instead, the quantized edge map construction is designed to first and foremost produce a consistent map independent of any errors in tracing streamlines or classifying inflow outflow points. Furthermore, contrary to the edge map algorithm the structure of the mapping, e.g. the number and connection of intervals, is computed using the most stable aspects of the flow and thus typically produces a more accurate map as well.

Quantized edge maps are constructed using a divide-and-conquer approach for each triangle. For each edge of the triangle, we first determine the connected intervals of inflow to outflow bins. Next, given these intervals we find the transition points on vertices and edges and classify these into internal and external. Note that, since a triangle is convex all transition points on vertices must be external. Transition points on edges are classified by examining the local flow direction. Flow switching from inflow to outflow in a counterclockwise traversal indicates an ITP while a switch from out- to inflow indicates an ETP. Furthermore, any ITP is immediately classified as ETP in its neighboring triangle since by definition a dual ITP classification is inconsistent with a PL flow. Unlike edge maps this construction allows for critical points on edges, e.g., a transition point classified as ETP on both sides naturally produces a rotating critical point.

Our method is to divide edges into intervals where the flow maps continuously and pair the resulting intervals to form *links*. Starting from the initial inflow/outflow intervals, we select the largest interval (outflow or inflow) and cast a streamline (numerically) from its center bin to identify its corresponding source/destination. This divides the flow within the triangle into two subregions.

Subsequently, we recursively handle each region separately until we reach one of the three possible final configurations: (a) A region without transition points; (b) A single ITP; or (c) A single ETP with only two intervals. In (a) we either have only two intervals which must be linked or multiple intervals that must contain sepx points. For the latter case we numerically identify candidates for separatrices and split the intervals accordingly. For (b) we determine the forward and backward image of the ITP and split the intervals. Finally, for (c) we link the two intervals around the ETP. By the time the algorithm resorts to numerically tracing potentially unstable streamlines the basic structure of the maps has been established. Thus, the tracing simply computes the best geometric fit to a known flow structure. In a last cleanup step we merge intervals not separated by transition, image, or sepx points to construct the final quantized edge map.

Consistency The construction algorithm relies on two types of numerical computations: the identification of transition points and the tracing of streamlines. We compute the former by solving a linear system of equations and the latter through exact streamline computation [NJ99]. Both techniques can produce inconsistent results and numerical errors which in fact is one of the primary motivation for constructing quantized edge maps. The algorithm described above, nevertheless, guarantees consistency by explicitly storing any numerical decision in form of the region decomposition. Any streamline that is traced divides the triangle into subregions and no subsequent streamline can leave its subregion. This naturally prevents any streamline crossings guaranteeing a consistent representation. Note this approach does not prevent numerical errors but instead forces any numerical decision to defer to all previous decisions.

4.3. Approximation Error

Quantized edge maps incur two types of approximation error; caused by (a) the quantization, and (b) the linear mapping between intervals. The quantization error is controlled by the bin size and determines how many unique streamlines are used to represent a flow. The geometric location of all landmarks (critical points, transition points, images points, etc.) can be off by at most half the bin size. In practice, we use 2^{32} bins, which makes the precision of our representation higher than possible with a 32-bit floating representation. Also, our bins are uniformly distributed, unlike the IEEE floating point standard. Thus, when compared to vertex-sampled and interpolated flows, the quantization error is negligible especially considering the gained consistency.

The mapping error on the other hand cannot be neglected. Similar to [BJB*12] we consider the mapping error at a given source p in terms of distance between its mapped destination $\xi^+(p)$ and its ground truth destination relative to the edge length. Unfortunately, there exists no closed form solution for this error but experimentally (evaluated using [NJ99]) it appears to be smooth and well-behaved. Thus, we can evaluate the error for each link using a dense sample. We can refine the maps to reduce the mapping error in the same manner as [BJB*12]. Specifically, we select links with

a high error and iteratively split them by the streamline at the sampled point with maximal error. Assuming fewer than 2^{12} intervals per triangle (we use 12 bits to index into the interval array), each split requires only $2(32+16)$ additional bits of storage (the overhead for storing two new intervals). While refinement requires careful sampling, after map refinement the extra intervals only have a small effect on runtime for map queries. Figure 5(c) shows the observed convergence in terms of the maximal and average error for a given number of links. We see this error is well behaved in both a maximal and average sense, with only a small initial dip as flow is refined (mostly near saddles) initially. Figure 5 shows examples of the effect of refinement on the stable and unstable manifolds of an ocean current flow.

5. Quantized Critical Points

In PL vector fields, critical points are typically detected by numerically solving for points with $\vec{V}(x) = \vec{0}$ and classified using the eigenvalues of the Jacobian matrix of $\vec{V}$ [HH89]. However, this classification is limited to non-degenerate, first-order critical points and moreover finding exact zeros is numerically unstable. An alternate classification is to use the Poincaré index [EW10, TSH01]. Given a simple closed curve γ with $\vec{V}(p) \neq \vec{0}$, for all $p \in \gamma$, the Poincaré index ind_γ of $\vec{V}$ with respect to γ is defined as the number of vector rotations of the field as it travels along γ: $ind_\gamma = \frac{1}{2\pi}\oint_\gamma d\theta$, where $\theta = \arctan(\vec{V}_y/\vec{V}_x)$ is the angle coordinate of the vector field $\vec{V}(x) = (\vec{V}_x, \vec{V}_y)$. The Poincaré index reflects the aggregate structure of $\vec{V}$ on the interior of γ. In particular, if γ encloses no critical points then $ind_\gamma = 0$ and if there only exists a single critical point ind_γ is called the index of the critical point. Finally, given a set of critical points c_i with a simple enclosing curve γ then $ind_\gamma = \sum_i index(c_i)$. The Poincaré index can distinguish higher order critical points such as k-saddles with $ind_\gamma = -k$ or k-poles with $ind_\gamma = +k$ when they are isolated in a neighborhood bounded by γ.

Nevertheless, computing the Poincaré index in practice is non-trivial as it becomes difficult to ensure that γ does not contain a critical point. Instead, using quantized flow ind_γ can be computed by a simple count of transition points. In particular, one can imagine transforming $\vec{V}$ in an epsilon neighborhood of γ such that $\vec{V}$ is orthogonal to $d\gamma/dt$ and switches infinitely fast from outflow to inflow at transition points. This concentrates the rotation of $\vec{V}$ at the vertices and transition points and does not affect ind_γ. This can be seen as the equivalent of using exterior angles as discrete curvature measure for polygonal curves [GS08] where all curvature is concentrated at vertices. In particular, for a counterclockwise traversal around a triangle T, each transition point contributions a rotation of π (positive for ITPs and negative for ETPs), while each vertex contributes α, its exterior angle to ind_γ (which sum to 2π for all three vertices). When there is an ETP on a vertex, we can separate the contribution of the transition point from that of the exterior angle:

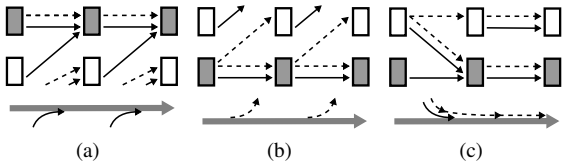


Figure 6: *Classifications of forward stable closed orbit shown as grey bins. Forward maps/streamlines are indicated by solid, and backward by dashed arrows. (a) On the right of the closed orbit streamlines can only converge leading to attracting behavior. (b) If no forward streamlines approach from the left, the closed orbit is repelling from the left. (c) If a single forward streamline merges with the closed orbit it is attracting; all backward lines only touch it.*

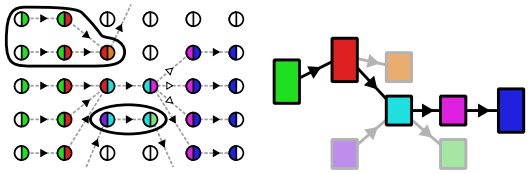
$$ind_{\partial T} = \frac{1}{2\pi}\left(\sum_{\|I\|}\pi + \sum_{\|E\|} -\pi + 2\pi\right) = (\|I\| - \|E\|)/2 + 1$$

where I is the set of ITPs and E is the set of ETPs. Note that this formula conveniently ignores potential critical points on both edges and vertices. As discussed above, the classification into ETPs and ITPs is made based on the local map. As a result, critical points on the boundary of a triangle will be treated as conventional transition points which effectively computes the Poincaré index of the interior of the triangle. Computing the Poincaré index for all triangles detects and classifies all critical points on the interior of triangles.

For critical points on edges one computes ind_γ or the boundary of the quadrilateral formed by the adjacent triangles and subtracts the index of both triangle interiors. This is possible since transition points at quadrilateral corners can still be easily classified into external and internal transition points based on the local maps. Note that while the algorithm of Section 4.2 avoids creating critical points on edges other approaches may not and the representation naturally supports edge criticalities. Finally, the Poincaré index of a vertex is computed using the boundaries of its star and subtracting the triangle and edge indices. Overall, this strategy provides a generic, combinatorial detection and classification of critical points including the often problematic cases of critical points near vertices and edges.

6. Quantized Closed Orbits

An equally important aspect of vector field analysis focuses on other invariant structures than just critical points. A general vector field can contain closed orbits which generically can be attracting, repelling, or both. Closed orbits are particularly difficult to detect with numerical techniques. Instead, a quantized flow is at its core described by a very large graph and thus closed orbits exist explicitly as circular quantized streamlines. Since the forward and backward maps differ, there exist three types of closed orbits, those stable in ξ^+ and those stable in ξ^- and those stable in both directions. These can easily be classified into attracting or repelling by analyzing the mappings around the closed orbit. Furthermore, due to some subtleties in the definition of the $\xi^\pm$s there only exist six different classifications.



(a) (b)

Figure 7: *Exact closed orbit detection. The bin level connectivity (a) defines the initial link graph (b). Extracting MSCCs removes the faded links, and the bin level pruning removes the circled bins.*

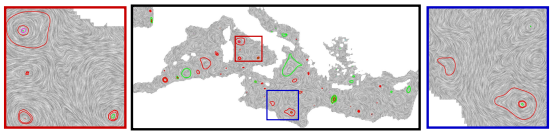


Figure 8: *The classification of closed orbits. Flow is mostly dominated by both-side attracting (red) and both-side repelling (green) closed orbits. Blue closed orbits are neutral on both sides and hence are both forward and backward-stable. Other closed orbit types exist but are too small to be seen.*

Consider the forward-stable closed orbit C shown in Figure 6. Since C is defined by ξ^+ (shown as solid arrows) there cannot exist a streamline diverging from C towards its right (wrt. to the flow direction). Thus, on its right C can either be attracting (Figure 6(a)), if there exists at least one other streamline converging to it or neutral (neither attracting nor repelling). On its left, C can be neutral if no other streamline approaches, attracting if there exists one or more forward streamlines approaching C (Figure 6(c)), or repelling otherwise (Figure 6(b)). The reason for this classification is that if there exists one or more forward streamlines that approach C from the left, then all backward streamlines that share bins with C will ultimately leave C. However, since we consider all streamlines to be distinct entities, which may become infinitely close but never touch, such backward streamlines are never part of C but simply approach and leave its vicinity. As a result a forward-stable closed orbit can be attracting, repelling, or neutral on its left but only attracting or neutral on its right. Symmetrically, a backward closed orbit can be attracting, repelling or neutral on its left but only repelling or neutral on its right. One consequence of these properties is that no quantized closed orbit can switch between attracting and repelling and thus cannot form a saddle-like region in the sense of Conley theory [CMLZ08].

6.1. Exact Closed Orbit Detection

To detect all closed orbits in a data set we use a two-stage process exploiting the fact that a quantized flow represents a multi-resolution graph. In the first stage we create the *link graph* (Figure 7(b)) which contains a node for every link and arcs between links that share at least one bin. The link graph is a conservative description of the flow in the sense that two links sharing a streamline are connected in the link graph but not all connected links share a streamline (Figure 7(a)). Subsequently, we extract all maximally strongly connected components (MSCCs) from this graph using the algorithm of Barnat and Moravec [BM06]. Since usually most triangles are non-critical they contain at most four links and thus the initial link graph typically has around four links per triangle on average. However, since all critical points have already been identified one can avoid creating any nodes whose flow is entirely connected to sinks or sources. This drastically reduces the size of the initial link graph and also splits it into independent components which can be processed separately.

After the first stage each MSCC represents a set of bins forming a region that may contain closed orbits. To identify individual closed orbits we *prune* the graph by geometrically shrinking its nodes. The region is shrunken by removing the bins that either enter or leave the region of interest. Once no more bins can be pruned, the boundary of the MSCC forms a bin level closed orbit. Next, all the bins in the closed orbit are removed, exposing new bins to the pruning. This process continues until all regions are empty. A closed orbit stable in one direction may not be stable in the other. As a result, pruning bins according to ξ^+ may remove parts of a closed orbit in ξ^- and vice versa. Thus, the pruning is performed twice, once to identify all forward and once for all backward closed orbits. Figure 8 shows all closed orbits of an ocean flow around Italy with the color indicating the classification.

6.2. Approximate Closed Orbit Detection

The pruning algorithm is guaranteed to find all closed orbits as a sequence of bins which can subsequently be classified. However, in practice there can exist regions with extremely low divergence, and in many applications vector fields are theoretically divergence-free. Typically, these are not represented exactly and the resulting approximated vector field has near-zero divergence. In such cases a quantized flow may represent billions of closed orbits. Extracting these individually is neither practical nor desirable. Instead, we approximate potential closed orbits up to a user threshold.

To approximate closed orbits we split MSCCs by tracing quantized streamlines. Specifically, we select the MSCC node with the largest width of bins and trace a streamline for its center bin both forward and backward. For all nodes that are part of this streamline we duplicate the corresponding bin, split the node into two disconnected nodes, and update the arcs of either node. The streamline can either: (a) form a closed orbit; (b) leave the MSCC; or (c) connect to a source and/or sink; all of which significantly change the connectivity of the graph. To avoid tracing such a streamline for too long we specify a user determined maximal length after which the tracing is terminated. Note that even a partial streamline splits graph nodes and thus provides new information. Once the graph connectivity has been updated we again look for MSCCs. We iterate this process until the cross-section of all remaining regions falls below a threshold of minimal number of bins as shown in Figure 10. These regions become the current approximation of potential closed

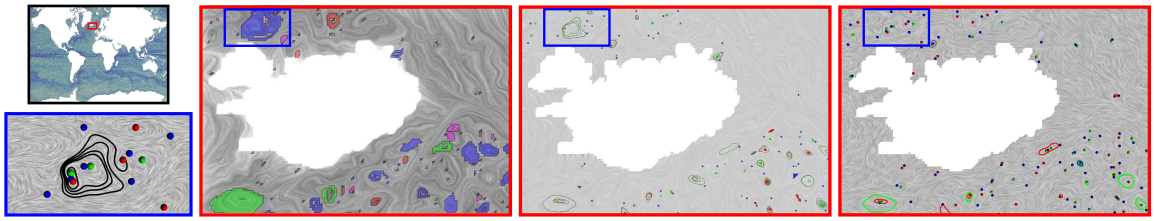


Figure 9: *Comparison of closed orbit detection with Morse sets (left, Chen et al. [CDS*12]) and refined PC Morse sets (middle, Szymczak and Zhang [SZ12]). Both Morse techniques compute cyclic regions in the blue boxed region, but our approach (right) identifies none, as the bottom left inset indicates a streamline spans the region.*

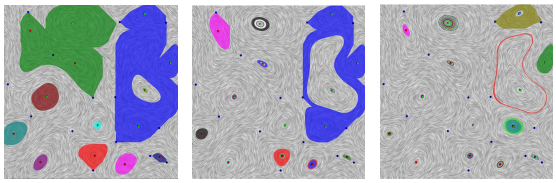


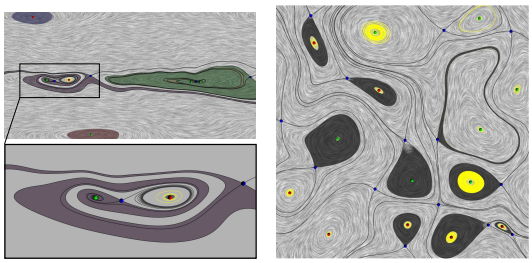
Figure 10: *Graph-splitting progressively convergence to closed orbits. Left-to-right: a gradual reduction in the size of regions.*

orbits and their aggregated flow behavior can be determined by examining the maps along their boundary.

6.3. Comparison

Figure 9 presents a comparison of our closed orbit detection technique with that of Chen et al. [CDS*12] and Szymczak and Zhang [SZ12]. We use a small tile around Iceland containing 23655 triangles. This figure highlights differences between our technique and the other two, in part because of the instabilities and very small scale orbits that are contained within. The triangle-based approach of Chen et al. identifies 160 Morse sets. Szymczak and Zhang's PC approach detects 297 Morse sets, of which 113 are trivial. In this example, they have refined their transition graph with 11 iterations of refinement, subdividing far beyond the Morse sets seen by Chen et al. Our approach detects 122 closed orbits and 166 critical points (45 sinks, 38 sources, and 83 saddles) after refining to a threshold of 0.023%.

The limit sets that each technique produces are qualitatively quite similar, as depicted for the larger orbits. This example shows that numerical instabilities are handled differently by each approach. Chen et al. prefer a coarser approximation, covering whole triangles with regions which behave as Morse sets. Szymczak and Zhang find a finer scale approximation by refining the transition graph represented by their PC flow. One difference with our technique is explained in the blue box of Figure 9. Our refinement, instead of equally splitting links as Szymczak and Zhang, splits links based on an error measure which indicates geometric closely to the original PL flow. It turns out that in this region, we find no closed orbits, because the PL flow prescribes a streamline (computed using [NJ99]) that crosses over the closed orbit regions detected by the other two techniques. This leads to an interesting contrast between the three discrete approaches. All three approaches make some sacrifice when computing the input PL data to a combinatorial representation. However, our approach strives to capture as much geometric information as possible by encoding an approximation of the streamlines prescribed by the input PL field.



(a) (b)

Figure 11: *(a) Stable and unstable manifolds for an ocean wind dataset, where a separatrix converges to a closed orbit. (b) Separatrices and closed orbits on the nearly divergence free dataset.*

7. Topological Decomposition

Each bin in a quantized flow is either part of a closed orbit or connects a source with a sink (either of which could be a closed orbit or boundary point). In many applications it is important to understand the region of influence of a particular source/sink. In particular, we can define the unstable/stable manifold of a source/sink as the region in which all streamlines start/end at the corresponding critical point. The sets of stable/unstable manifolds partition the domain and their intersection forms the topological skeleton.

Computing stable/unstable manifolds in a quantized flow is straightforward. Starting, for example, at a pointwise source in the interior of a triangle, we add all outflow bins on the triangle boundary into a moving front. We then propagate this front by determining connected sets of bins that are part of forward streamlines originating at the current front and whose backward streamlines will end at the original source. Note that the front is grown by advecting entire intervals of bins rather than individual bins and thus avoids performance issues due to the large number of bins. Figure 11(a) shows exemplary stable and unstable manifolds for two attracting spirals (red balls) and four repelling spirals (green balls). To unclutter the visualization we do not show the manifolds corresponding to boundary inflow or outflow.

A key piece of the topological skeleton are separatrices—streamlines separating stable/unstable manifolds. One

Dataset (Figure)	# Tri.	# (S^+; S^-; S^*)	Intervals per tri.	Maps		# Closed Orbits (Time)	
				Memory	Gen. time	Approx.	Exact
Iceland currents (9)	23,655	45; 38; 83	4.73	0.9	0:05	74 (0:02)	96 (0:02)
Italy currents (8)	56,071	128; 111; 236	4.79	2.18	0:10	217 (0:14)	434 (0:17)
Cuba currents (5)	62,882	56; 63; 117	4.47	2.32	0:10	149 (0:13)	399 (0:15)
Ocean winds (11(a))	328,328	4; 2; 4	4.05	11.37	0:20	16 (0:14)	35 (0:12)
North Atlantic (1)	580,084	391; 386; 772	4.46	21.4	0:45	983 (1:24)	2000 (1:48)
HCCI (10, 11(b))	816,642	5; 9; 14	4.05	28.3	01:35	376 (05:38)	5018 (78:12)

Table 1: *Performance details per dataset: # (S^+; S^-; S^*) indicates the number of sources, sinks, and saddles. Memory and time are given in MB and Min:Sec, respectively. The approximate orbits converge to roughly 1 million bins (2e-4 of an edge).*

method to compute them is to extract them from the segmentation. However, often the separatrices are visualized without the stable/unstable manifolds, and it can be easier (and more efficient) to extract them directly. Since all bins are classified according to their corresponding stable/unstable manifold it is clear that separatrices are represented by bin boundaries, specifically, by the leftmost bin in the sepx point. Starting at each saddle we trace the streamline (either forward or backward) corresponding to the bin just left (with respect to the tracing direction) of the separatrix. By construction the maps defining streamlines will always pick the rightmost bin and thus the right side boundary of this streamline defines the separatrix. Separatrices and closed orbits together form the topological skeleton of a vector field, as shown in Figure 11(b). We found 1547 closed orbits (yellow) around 14 critical points (red and green). Of the separatrices (black) from 14 saddles, 9 converge to closed orbits.

8. Computational Details

Table 1 shows a list of data sets used, their size, number of topological features, as well as the time taken for conversion, and closed orbit detection, as well as the memory footprint for representing quantized flows. Given the exceedingly large number of streamlines that are explicitly encoded and the amount of detail preserved, e.g. in Figure 1, the memory footprint remains surprisingly competitive. While our average of about 24-28 bytes per triangle is certainly larger than a grid with two floating point values per vertex a typical half-edge data structure may easily store a similar amount of data just to maintain the connectivity information. The conversion times are far from interactive in our prototype. However, this one-time cost could easily be parallelized since each triangle can be processed independently. In many cases computing streamlines is actually faster than traditional integration since each triangle requires only a rasterization step versus likely many Runge-Kutta integrations.

Unsurprisingly, the time for an exact closed orbit detection is highly data dependent. In compressible flows with non-trivial divergence detecting all individual closed orbits is reasonably fast. For data sets with extremely low divergence and/or a large number of closed orbits the computational cost increases significantly. Theoretically, each edge could represent as many as 2^{32} crossings of the same streamline all of which must be traversed sequentially. However, the approximate closed orbit detection is fast so in practice we recommend a combination of both techniques.

9. Discussion

Quantized flows represent an important step towards a new theory of vector fields by combining discrete algorithms with the approximation power of interpolated fields. This representation enables graph-based algorithms which extract typical flow structures consistently and with geometric fidelity. Consequently, they have the potential to form the basis of a new set of analysis techniques that repeat the recent success of discrete scalar field topology.

While we have focused on PL vector fields and triangulated surfaces, one of the advantages of our new representation is the fact that is easily extends to different mesh types and interpolation schemes. By reserving more bits to encode the edge number on an interval we believe the existing data structure can encode flow on any polygonal mesh. Furthermore, the flow is queried in just two clearly defined ways: to find zero-crossings and to find exit points of streamlines. Thus, the input flow representation could be a black box solver making it amenable to, for example, higher order interpolation schemes. The main adjustment necessary would be a more general conversion algorithm as some of the convenient assumptions, e.g. the limited number of transition points per edge, would no longer hold. In this manner, quantized flow forms a unifying representation able to separate the flow encoding used by a simulation from the analysis within a strict and well-known approximation error. A complete understanding of the theoretical nature of these approximation errors remains future work. Finally, since the edge map representation can encode uncertainty information as well as time-dependent vector fields [BJB*12], both concepts can be extend to the quantized version.

Acknowledgements This work is supported in part by NSF awards IIS-1045032, OCI-0904631, OCI-0906379 and CCF-0702817, and by KAUST Award KUS-C1-016-04. This work was performed under the auspices of the U.S. DOE by the Univ. of Utah under contracts DE-SC0001922, DE-AC52-07NA27344, and DE-FC02-06ER25781, and LLNL under contract DE-AC52-07NA27344. We thank Guoning Chen, Eugene Zhang, and Andrzej Szymczak for helping us generate Fig. 9. We are grateful for data from Jackie Chen (Figs. 10 and 11(b)), Han-Wei Shen (Fig. 11(a)), and Mathew Maltrud from the Climate, Ocean and Sea Ice Modelling program at LANL and the BER Office of Science UV-CDAT team (Figs. 1, 5, 8, 9). LLNL-CONF-548652.

References

[BJB*12] BHATIA H., JADHAV S., BREMER P.-T., CHEN G., LEVINE J. A., NONATO L. G., PASCUCCI V.: Flow visualization with quantified spatial and temporal errors using edge maps. *IEEE Trans. Vis. Comp. Grap.* (2012). To appear. 2, 3, 4, 5, 9

[BM06] BARNAT J., MORAVEC P.: Parallel algorithms for finding SCCs in implicitly given graphs. In *FMICS/PDMC* (2006), pp. 316–330. 7

[Bre65] BRESENHAM J.: Algorithm for computer control of a digital plotter. *IBM Systems Journal 4*, 1 (1965), 25–30. 4

[CDS*12] CHEN G., DENG Q., SZYMCZAK A., LARAMEE R. S., ZHANG E.: Morse set classification and hierarchical refinement using Conley index. *IEEE Trans. Vis. Comp. Grap.* (2012). To Appear. 3, 8

[CMLZ08] CHEN G., MISCHAIKOW K., LARAMEE R. S., ZHANG E.: Efficient Morse decompositions of vector fields. *IEEE Trans. Vis. Comp. Grap. 14*, 4 (2008), 848–862. 2, 3, 7

[EW10] EFFENBERGER F., WEISKOPF D.: Finding and classifying critical points of 2d vector fields: a cell-oriented approach using group theory. *Computing and Visualization in Science 13* (2010), 377–396. 6

[For98] FORMAN R.: Combinatorial vector fields and dynamical systems. *Math. Z. 228*, 4 (1998), 629–681. 2

[GLL91] GLOBUS A., LEVIT C., LASINSKI T.: A tool for visualizing the topology of three-dimensional vector fields. In *IEEE Visualization* (1991), pp. 33–41. 1, 2

[GNP*06] GYULASSY A., NATARAJAN V., PASCUCCI V., BREMER P.-T., HAMANN B.: A topological approach to simplification of three-dimensional scalar functions. *IEEE Trans. Vis. Comp. Grap. 12*, 4 (2006), 474–484. 2

[GS08] GRINSPUN E., SECORD A.: Introduction to discrete differential geometry: the geometry of plane curves. In *ACM SIGGRAPH ASIA 2008 courses* (2008), ACM, pp. 9:1–9:4. 6

[GTS04] GARTH C., TRICOCHE X., SCHEUERMANN G.: Tracking of vector field singularities in unstructured 3d time-dependent datasets. In *IEEE Visualization* (2004), pp. 329–336. 2

[Hal01] HALLER G.: Lagrangian coherent structures and the rate of strain in two-dimensional turbulence. *Phys. Fluids A 13* (2001), 3365–3385. 1

[HH89] HELMAN J., HESSELINK L.: Representation and display of vector field topology in fluid flow data sets. *IEEE Computer 22*, 8 (1989), 27–36. 1, 2, 6

[HRAW07] HELGELAND A., REIF B., ANDREASSEN Ø., WASBERG C.: Visualization of vorticity and vortices in wall-bounded turbulent flows. *IEEE Trans. Vis. Comp. Grap. 13*, 5 (2007), 1055–1067. 1

[JBB*12] JADHAV S., BHATIA H., BREMER P.-T., LEVINE J. A., NONATO L. G., PASCUCCI V.: Consistent approximation of local flow behavior for 2D vector fields using edge maps. In *Topological Methods in Data Analysis and Visualization II* (2012), Peikert R., Hauser H., Carr H., Fuchs R., (Eds.), Springer, pp. 141–160. 2, 3

[LBM*06] LANEY D., BREMER P. T., MASCARENHAS A., MILLER P., PASCUCCI V.: Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Trans. Vis. Comp. Grap. 12*, 5 (2006), 1053–1060. 2

[LI07] LIU Z., II R. J. M.: Robust loop detection for interactively placing evenly spaced streamlines. *Computing in Science and Engineering 9*, 4 (2007), 86–91. 2

[NJ99] NIELSON G. M., JUNG I.-H.: Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE Trans. Vis. Comp. Grap. 5*, 4 (1999), 360–372. 5, 8

[RLH11] REININGHAUS J., LÖWEN C., HOTZ I.: Fast combinatorial vector field topology. *IEEE Trans. Vis. Comput. Graph. 17*, 10 (2011), 1433–1443. 2

[SP09] SADLO F., PEIKERT R.: Visualizing Lagrangian coherent structures and comparison to vector field topology. In *Topology-Based Methods in Visualization II* (2009), Hege H.-C., Polthier K., Scheuermann G., (Eds.), Springer, pp. 15–30. 1

[SPP04] SADLO F., PEIKERT R., PARKINSON E.: Vorticity based flow analysis and visualization for pelton turbine design optimization. In *IEEE Visualization* (2004), pp. 179–186. 1

[SZ12] SZYMCZAK A., ZHANG E.: Robust Morse decompositions of piecewise constant vector fields. *IEEE Trans. Vis. Comp. Grap.* (2012). To Appear. 3, 8

[Szy11] SZYMCZAK A.: Stable Morse decompositions for piecewise constant vector fields on surfaces. *Comp. Grap. Forum 30*, 3 (2011), 851–860. 3

[TSH01] TRICOCHE X., SCHEUERMANN G., HAGEN H.: Continuous topology simplification of planar vector fields. In *Proc. of IEEE Visualization '01* (2001), pp. 159–166. 2, 6

[TWHS03] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Saddle connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In *Proc. of IEEE Visualization '03* (2003), pp. 225–232. 2

[TWHS04] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Grid-independent detection of closed stream lines in 2D vector fields. In *Proceedings of the Vision, Modeling, and Visualization Conference 2004 (VMV)* (2004), pp. 421–428. 2

[TWHS05] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Topological methods for 2D time-dependent vector fields based on stream lines and path lines. *IEEE Trans. Vis. Comp. Grap. 11*, 4 (2005), 383–394. 2

[WS01] WISCHGOLL T., SCHEUERMANN G.: Detection and visualization of closed streamlines in planar flows. *IEEE Trans. Vis. Comp. Grap. 7*, 2 (2001), 165–172. 2

[WS02] WISCHGOLL T., SCHEUERMANN G.: Locating closed streamlines in 3D vector fields. In *Symposium on Data Visualisation* (2002), VISSYM '02, Eurographics, pp. 227–232. 2