Robust Morse Decompositions of Piecewise Constant Vector Fields

Andrzej Szymczak, *Member*, *IEEE*, and Eugene Zhang, *Senior Member*, *IEEE*

Abstract—In this paper, we introduce a new approach to computing a Morse decomposition of a vector field on a triangulated manifold surface. The basic idea is to convert the input vector field to a piecewise constant (PC) vector field, whose trajectories can be computed using simple geometric rules. To overcome the intrinsic difficulty in PC vector fields (in particular, discontinuity along mesh edges), we borrow results from the theory of differential inclusions. The input vector field and its PC variant have similar Morse decompositions. We introduce a robust and efficient algorithm to compute Morse decompositions of a PC vector field. Our approach provides subtriangle precision for Morse sets. In addition, we describe a Morse set classification framework which we use to color code the Morse sets in order to enhance the visualization. We demonstrate the benefits of our approach with three well-known simulation data sets, for which our method has produced Morse decompositions that are similar to or finer than those obtained using existing techniques, and is over an order of magnitude faster.

Index Terms—Morse decomposition, vector field topology.

1 Introduction

Vector field visualization has a wide range of application in dynamic systems, fluid nd solid mechanics, electromagnetism [3], computer vision [25], population theory, and economics. Vector field topology can provide a compact representation of essential structures in a vector field, and has gained much attention from the visualization community since its introduction to the community by Helmann and Hesselink [12].

Most of the current approaches to vector field topology rely on the ability to accurately compute trajectories such as periodic orbits and separatrices. Such approaches are prone to errors resulting from inaccuracy of numerical integration. Morse decomposition is a relatively new tool in vector field visualization, which has been used to define and extract vector field topology in a numerically stable and rigorous fashion in [6]. Morse decomposition consists of a finite number of disjoint Morse sets that contain all recurrent dynamics of the flow, in particular periodic orbits and stationary points. Morse sets are typically classified according to their *Conley index* [7]. In particular, the Conley index allows one to distinguish Morse sets similar to stationary points and periodic orbits. Morse decompositions naturally support multiscale analysis. Roughly speaking, a coarser one can be obtained by replacing two neighboring Morse sets with the union of both of them and their connecting trajectories [7] (where neighbors are defined by means of the flow). The work [5] uses these properties to build a unified framework for extracting vector field features and to design a vector field simplification algorithm.

There are a number of reasons for our interest in Morse decompositions and Conley index theory. Morse decomposition provides a unified framework for recognizing and extracting recurrent vector field features. In particular, standard features such as periodic orbits or stationary points can be interpreted as the same object: Morse set. However, Morse sets can also be more complicated. This is important since the success of traditional approaches relies on the ability to find a *finite* number of nondegenerate isolated stationary points and periodic orbits. They may fail for vector fields that have infinite number of such features. In the 2D case, this could be a vector field that contains a ring consisting of periodic orbits (which can still be a valid Morse set). In 3D case, a typical chaotic attractor (such as the Lorenz attractor [22]) contains an infinite number of *hyperbolic* periodic orbits. While attempting to compute all of them is pointless, methods based on Conley index theory have been successfully used in rigorous computer-assisted analysis of chaotic attractors [26], in particular to prove the existence of infinite number of periodic trajectories [27]. Similar issues may arise in vector fields that are not known exactly, but only up to an error. Such vector fields are ubiquitous in science and engineering, where inaccuracies may arise from numerical or measurement errors. One can think of a trajectory of such vector field as a curve whose velocity is within the error bound from the input vector field. In particular, this means that a generic periodic orbit is not isolated: one can perturb the velocity vector at any of its points and then steer it back to the same point (staying within the error bound) so that it stays periodic and has a similar period. Thus, a natural way to study the topology of such systems in terms of *sets of trajectories* rather than individual trajectories. In particular, the approach of [30] to a similar problem based on a probabilistic model of error uses density distributions (that can be thought of as sets with fuzzy membership function) to model topological

- *A. Szymczak is with the Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401-1887. E-mail: aszymcza@mines.edu.*
- *E. Zhang is with the School of Electrical Engineering and Computer Science, Oregon State University, 2111 Kelley Engineering Center, Corvallis, OR 97331. E-mail: zhange@eecs.oregonstate.edu.*

Manuscript received 28 June 2010; revised 26 Jan. 2011; accepted 23 Mar. 2011; published online 5 May 2011.
Recommended for acceptance by G. Scheuermann.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-06-0128.
Digital Object Identifier no. 10.1109/TVCG.2011.88.

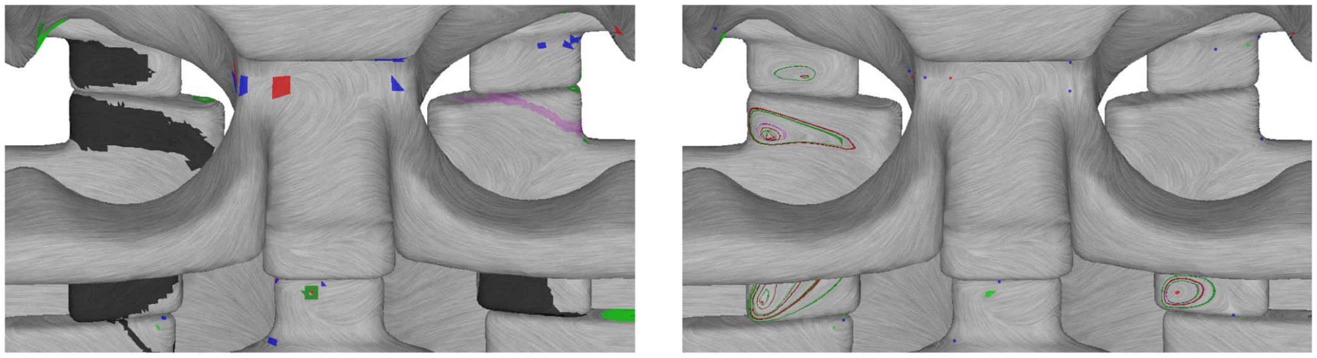
Fig. 1. Morse sets obtained using the approach of [4] (left) and the algorithm described in this paper (right), color-coded by type (red: classified as equivalent to a repelling fixed point or periodic orbit, green: attracting fixed point or periodic orbit, blue: saddle; Morse sets shown in magenta are trivial, i.e., contain no features or features that cancel each other; all other—*complex*—Morse sets are shown in black). Our algorithm produces a more detailed result, in terms of both a fineness of the Morse decomposition (topological) and the precision of individual Morse sets (geometric). In particular, it leads to Morse sets that can be interpreted as simple flow features (stationary points or periodic orbits). Because of the inherent low precision of the approach of [4], [6], the resulting Morse sets are larger. Some of them contain many flow features, and therefore are classified as complex. Consequently, they are hard to interpret. Moreover, the classification scheme described in [4] is not guaranteed to be accurate and therefore colors of some Morse sets shown on the left may not be correct. For the results shown here, the parameters of the algorithm of [4] were selected so that it requires roughly an order of magnitude more time than the algorithm introduced in this paper. More precisely, the running times were 435 and 25 seconds, respectively.

features. Generalizations of Morse decompositions and Conley index that are suitable for the deterministic error model have been developed in [23] and [29]. An extension of the PC framework introduced in this paper in this direction is proposed in [34].

The standard approach to compute Morse decompositions [4], [6] relies on numerical integration of trajectories of a large number of particles to generate the graph representation that is needed to determine the decomposition. This process is expensive and can still suffer from numerical integration errors. In addition, the precision of the resulting Morse decomposition is restricted by the underlying mesh resolution, since Morse sets are unions of triangles. These challenges have greatly limited the potential of using Morse decompositions to describe and study vector field topology as they are often too coarse and too computationally expensive to be practical. In this paper, we introduce a method to compute a Morse decomposition of a vector field on a triangulated manifold surface that addresses these issues. The key idea behind our approach is to convert the input vector field (typically, *vertex-based*, i.e., defined by vector values at mesh vertices) to a piecewise constant (PC) one. A PC vector field is constant in the interior of a triangle. Its trajectories can be defined using simple geometric rules. If the mesh is fine, the input vector field and its PC variant have similar Morse decompositions. Although our algorithm does require numerical calculations, they are simpler and more efficient than standard numerical integration. Furthermore, our approach allows Morse sets to have subtriangle precision. High precision Morse decompositions are usually easier to understand (Fig. 1), since their Morse sets tend to correspond to stationary points or periodic orbits.

There are some fundamental difficulties associated with topological analysis of PC vector fields, such as the discontinuity along the edges in the mesh (the standard theory of ordinary differential equations no longer applies) and the ambiguity in the definition of trajectories (there can be multiple trajectories emanating from a single point). We address both difficulties by developing a multivalued flow framework based on the theory of differential inclusions. We provide analysis for the correctness and efficiency of this framework. We show that trajectories of the PC vector field constructed using our algorithm converge to the trajectories of the underlying smooth vector field as the mesh gets finer and closer to the domain of that vector field (Appendix A in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.88). Since Morse decompositions are known to be robust under perturbation [6], [7], one can expect that a vector field and its PC variant have similar Morse decompositions if the mesh is sufficiently fine. Experiments that support this claim are described in Section 7.2. We also prove that our algorithm produces a correct Morse decomposition for the PC vector field (Appendix C, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.88). This is important since it guarantees the integrity of the result.

To the best of our knowledge, this is the first time a rigorous topological analysis framework is proposed for general, i.e., not necessarily gradient, PC vector fields. We also introduce a new classification scheme for Morse sets based on fixed point index and stability, i.e., categorization as attracting, repelling or neither attracting nor repelling. This scheme yields information similar to the Conley index and allows one to distinguish Morse sets corresponding to sinks, sources, saddles, and attracting or repelling periodic orbits.

The paper is organized as follows: in Section 2, we include a brief discussion of the related work on vector field visualization and vector field topology. Section 3 describes PC vector fields on a triangulated manifold surface. A transition graph, a finite representation of all trajectories of a PC vector field, is described in Section 4. The procedure to compute the Morse decomposition from a transition graph and to classify its Morse sets is described in Section 5. Section 6 discusses the complexity of our algorithm. Experimental results are presented in Section 7. Finally, Section 8 discusses future research directions that may be motivated by this work.

2 Prior Work

Vector field visualization has been an active research topic during the past two decades. It is beyond the scope of this paper to review all research related to vector field visualization and analysis. Thus, we will only review the most relevant work, namely, topology-driven vector field visualization, and refer the readers to the surveys [17], [18], [24] for more thorough reviews of vector field visualization.

Considerable amount of work on extracting vector field topology has been done in recent years. In most cases, the focus was on computing basic features such as stationary points, periodic orbits, and separatrices. Stationary points and separatrices can be found using the technique of [12]. Periodic orbits can be computed by following trajectories until they converge to a limit cycle [40]. An approach based on a geometric interpretation of periodic trajectories as intersections of stream surfaces of the flow 'lifted' to the 3D space has been proposed in [36].

In [5], Morse decomposition is used to identify stationary points and periodic orbits. They are incorporated into a topological graph called the Entity Connection Graph (ECG), which extends the original definition of vector field topology of [13]. Numerical instability intrinsically associated with vector field topology defined in terms of individual trajectories is discussed in [6]. One can use Morse decomposition to obtain a more robust representation of vector field topology. An algorithm to compute a Morse decomposition and the Morse Connection Graph (MCG), that is similar to ECG but represents connections between Morse sets rather than periodic orbits and stationary points, is described in [6]. Generally, the MCG is less detailed than the ECG but is more stable and less dependent on numerical integration method. An adaptive refinement scheme for Morse decompositions that can lead to more efficient and more precise analysis was recently introduced in [4].

In contrast to [4], [6], our approach has subtriangle precision (i.e., produces Morse sets that are not necessarily unions of mesh triangles). PC vector fields also support a simple method to accurately classify the Morse sets (classification in [4] is based on an upper bound on the Conley index and is not guaranteed to be accurate). Finally, analysis of a PC vector field is over an order of magnitude faster than analysis performed using the approach of [4], [6] (Section 7). Piecewise constant vector fields have been used as a tool to study separation and attachment line features (similar to the exploding and imploding edges, Section 3.1) in [37].

Let us stress that high performance of our algorithm and high precision of its output are achieved at the cost of approximation accuracy. More precisely, in our approach, trajectories of the input vector field are generally not approximated as well as by one of the standard numerical integration methods. Therefore, the approach of [4], [6] may still be a better choice if high accuracy is desired. While the PC framework can be used to obtain Morse decompositions guaranteed to be correct for a given continuous vector field (as discussed in the forthcoming paper [34]), this significantly increases the computational cost and memory usage and leads to a coarser result.

An approach based on discrete vector fields motivated by Forman's discrete Morse theory [10] is proposed in [32].

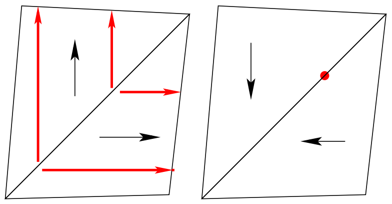
Fig. 2. Two examples illustrating the fundamental issues of the naive definition of PC vector fields. Black arrows represent vectors associated with the triangles that contain them. Left: Trajectories (red lines) starting at points arbitrarily close to the edge separating the two triangles, but on different sides, diverge when traced forward in time. The flow is discontinuous. Right: Trajectories cannot be traced forward in time from the red point because of the contradictory velocity constraints in both triangles.

Trajectories of a discrete vector field can only move along edges of the dual graph and, in general, do not converge to the trajectories of the original vector field as triangle sizes go to zero. For a PC vector field, the trajectories are also restricted by the mesh—they have to follow straight lines inside triangles—but convergence to the original vector field can be established under mild assumptions (Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.88).

3 Piecewise Constant Vector Fields

A planar PC vector field is constant and equal to $f(\Delta)$ in the interior of a triangle Δ of a triangulated domain D. Naively, a PC vector field can be defined by $g(x) := f(\Delta)$ for any $x \in \Delta$, where Δ is a mesh triangle. However, this definition is ambiguous for points on edges or at the vertices of the triangulation. No matter how these ambiguities are resolved, the resulting vector field is generally not continuous at these points. Therefore, the existence of its trajectories (i.e., solutions of an initial value problem $\dot{x} = g(x)$, $x(0) = x_0$) is not guaranteed by the general theory of ordinary differential equations. Even though one could attempt to trace the trajectories numerically, the resulting flow would be discontinuous, making reliable interpretation of results problematic. Examples demonstrating nonexistence of trajectories and discontinuity of the flow are shown in Fig. 2.

In this section, we describe a solution of these fundamental problems based on multivalued flows.

3.1 Definition

Let M be a manifold triangle mesh embedded in the 3D space. For each triangle Δ, let $f(\Delta)$ be a nonzero vector parallel to Δ, but not to any edge of Δ.

An edge e of a triangle Δ *attracts the flow in* Δ if and only if the vector $f(\Delta)$ points toward the edge e. Analytically, $\mathrm{e} = \bar{ab}$ attracts the flow in $\Delta = \Delta abc$ if and only if

$$\mathrm{sign}(((\vec{ab} \times \vec{ac}) \times \vec{ab}) \cdot f(\Delta)) \neq \mathrm{sign}(((\vec{ab} \times \vec{ac}) \times \vec{ab}) \cdot \vec{ac}),$$

e *repels the flow in* Δ if it does not attract the flow in Δ. An *imploding* (*exploding*) edge is an edge that attracts (respectively, repels) the flow in both of its incident triangles. Fig. 2 shows an exploding edge (left) and an imploding edge (right). A *crossing* edge attracts the flow in one of its incident

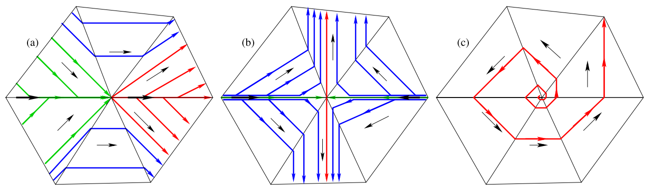

Fig. 3. Three examples of flow defined by a PC vector field near a vertex. Trajectories arriving to the vertex are shown in green. Trajectories leaving the vertex are shown in red. Trajectories in the hyperbolic sectors (two for the vertex on the left and four for the vertex on the right) are shown in blue.

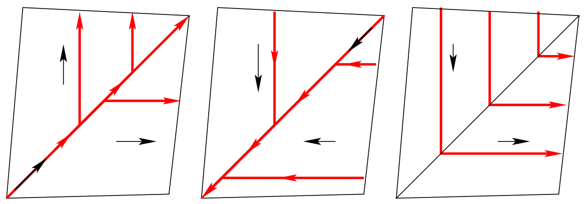

Fig. 4. Trajectories (red lines) in two adjacent triangle in the mesh. Black arrows represent the vectors assigned by f to these triangles and to the edge they share, if it is exploding or imploding. Left: an exploding edge. Middle: an imploding edge. Right: a crossing edge. Note that exploding and imploding edges are related to separation and attachment lines studied in fluid flow analysis [31], [37].

triangles and repels in the other. A PC vector field is defined by

- A function f that assigns a nonzero vector to any mesh triangle Δ and any exploding or imploding mesh edge $\mathbf{e}$. $f(\Delta)$ is required to be parallel to Δ, but not to any Δ's edge. $f(\mathbf{e})$ is required to be parallel to $\mathbf{e}$.
- A set of stationary points $\mathcal{S}$, consisting of mesh vertices.

3.2 Trajectories and Multivalued Flow

The goal of this section is to discuss the multivalued flow induced by a PC vector field. First (Section 3.2.1), we define the multivalued unstable vector field F_* that assigns a set of vectors $F_*(x)$ to every point $x \in M$. F_* is defined in terms of the assignment f and the set of stationary points $\mathcal{S}$ described in Section 3.1. F_* is used to define trajectories in Section 3.2.2. Section 3.2.3 describes the flow near a spiral sink or source. Finally, in Section 3.2.4 we discuss conditions that the flow needs to fulfill in order to ensure the applicability of topological analysis.

3.2.1 Unstable Vector Field

For a point $x \in M$, let the set of vectors $F_0(x)$ consist of 1) vectors $f(\Delta)$ for all triangles Δ containing x, 2) vectors $f(\mathbf{e})$ for all edges containing x, and 3) the zero vector if x is a vertex of M and $x \in \mathcal{S}$.

We say that a vector $w \in F_0(x)$ is an *unstable direction* at x if and only if $x + tw \in M$ and $w \in F_0(x + tw)$ for all sufficiently small positive t values. Intuitively, by moving from x in the unstable direction by a small amount we reach a point at which one of the vectors assigned by F_0 points *away* from x. The *unstable vector field* F_* assigns the set of all unstable directions at x to every $x \in M$.

It turns out that $F_*(x)$ is easy to determine.

If x is in the interior of a triangle Δ then $F_*(x) = \{f(\Delta)\}$.

Let x be a point that is not a mesh vertex, belonging to an edge $\mathbf{e}$ with incident triangles Δ_1 and Δ_2. If $\mathbf{e}$ is exploding, $F_*(x) = \{f(\Delta_1), f(\Delta_2), f(\mathbf{e})\}$. If $\mathbf{e}$ is imploding, $F_*(x) = \{f(\mathbf{e})\}$. If $\mathbf{e}$ is a crossing edge, $F_*(x) = \{f(\Delta_i)\}$, where $i \in \{1, 2\}$ is such that $\mathbf{e}$ repels the flow in Δ_i.

For a mesh vertex v, $F_*(v)$ consists of

- $f(\mathbf{e})$ for any edge $\mathbf{e} = \bar{v}\bar{w}$ incident to v such that $f(\mathbf{e}) \cdot \vec{vw} \geq 0$.
- $f(\Delta)$ for any triangle Δ incident upon v such that both edges of Δ incident upon v repel the flow in Δ.
- zero vector if $v \in \mathcal{S}$.

For example, for the vertex shown in Fig. 3a, $F_*(v)$ contains the vectors assigned by f to the horizontal edge to the right of v and both of its incident triangles, and possibly the zero vector if $v \in \mathcal{S}$. For the vertex shown in Fig. 3b, it contains vertical vectors assigned to the top and bottom triangles and the zero vector if $v \in \mathcal{S}$. For the vertex shown in Fig. 3c, $F_*(v)$ can only contain the zero vector. Vertices shown in (b) and (c) have to be stationary by the requirements discussed in Section 3.2.4 (see also Section 3.3.4).

Stable directions are defined in a similar manner. A vector w is a *stable direction* at $x \in M$ if $x + tw \in M$ and $-w \in F_0(x + tw)$ for small positive t values. Intuitively, trajectories can leave x along unstable directions. They arrive at x from stable directions. Trajectories are discussed in the next section.

3.2.2 Trajectories and Multivalued Flow

Trajectories of a PC vector field can be obtained by solving the *differential inclusion* $\dot{x} \in F_*(x)$, instead of the differential equation $\dot{x} = g(x)$ traditionally used for a single-valued vector field g. In the theory of differential inclusions [11], solutions are defined not as continuously differentiable functions, but as functions of a wider class of functions in order to guarantee desirable properties of the solution set. In the setting of this paper, the solutions are continuous piecewise linear paths in M, with knots at the vertices of the mesh or on its edges, possibly with infinite number of linear segments. They can be built by following a simple set of rules described below.

When a trajectory enters the interior of a triangle Δ, it moves along a straight line, with velocity $f(\Delta)$, until it hits Δ's boundary.

From a point on a mesh edge $\mathbf{e}$ (but not at a vertex), a trajectory can move along this edge if it is exploding (Fig. 4, left) or imploding (Fig. 4, middle). If $\mathbf{e}$ is exploding, the trajectory can leave $\mathbf{e}$ at any point, along a direction assigned to one of its incident triangles. If $\mathbf{e}$ is a crossing edge (Fig. 4, right) the trajectory has to immediately enter its incident triangle Δ, in which $\mathbf{e}$ repels the flow.

For a vertex v of the mesh, a trajectory can stay at v for some time, possibly forever, if v is stationary. Otherwise, it has to leave v immediately, moving along any vector in $F_*(v)$ (however, spiral sinks are an exception to this rule—Section 3.2.3). Note that in order to ensure the correctness of our algorithm, requirements discussed in Section 3.2.4 need to be satisfied, in particular, $F_*(v) \neq \emptyset$ for any vertex v. This means that a trajectory can be continued indefinitely. A number of examples of trajectories near a vertex are shown in Fig. 3.

Trajectories leaving a point are not unique, for example, for any point on an exploding edge or at any vertex with

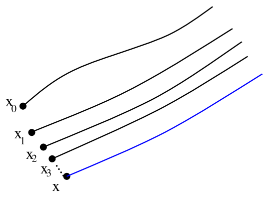

Fig. 5. Upper semicontinuity. The convergent sequence of trajectories starting at x_i defined for time values in $[0, t]$ (black) is required to converge to a trajectory starting at $x_* = \lim x_i$ (blue). Note that in the multivalued case, trajectories out of each of the points are not unique.

more than one unstable direction. Tracking a particle x forward along all trajectories starting at x over time t leads to a *set of locations*, which we denote by $\Phi(x, t)$. Formally, $\Phi(x, t)$ is defined as the set of all endpoints of trajectory segments $\sigma : [0, t] \rightarrow D$ starting at x. Φ is the *multivalued flow* of the PC vector field.

3.2.3 Spiral Sinks and Sources

Spiral sinks and sources in PC vector fields behave in a slightly different way than in the standard smooth vector field setting. Note that they are stationary points by the requirements discussed in Sections 3.2.4 (see also Section 3.3.4). First, let us look at a spiral sink shown in Fig. 7b. Trajectories approaching the vertex v in the middle are polygonal logarithmic spirals and therefore have finite length. Since their velocity does not decrease as they get closer to v, they arrive at v in *finite* time, despite spiraling around it infinitely many times and hence intersecting mesh edges *infinitely* many times before reaching v. After a trajectory hits v, it stays at v. Trajectories starting at a spiral source vertex v (Fig. 3c) are identical to trajectories arriving at a spiral sink with time reversed. One of them stays at v all the time. Others leave v along polygonal logarithmic spirals at some time t. Note that the spirals traced by these trajectories are similar: one can be obtained from any other by means of a uniform scale with center at v.

3.2.4 Requirements

Our algorithm is based on topological analysis of the multivalued flow Φ. To ensure the applicability of topological tools to the flow, we assume that it is *admissible* [11]. There are two properties that need to be satisfied in order to ensure admissibility of Φ. First, the flow is required to be *upper semicontinuous*. Upper semicontinuity is a generalization of continuity to the multivalued case. It means that the limit of a convergent sequence of trajectory segments traced over time interval $[0, t]$, is also a trajectory segment (Fig. 5). Second, there must exist a positive number h such that the set $S(x_0, h)$ of trajectories starting at any point $x_0 \in M$ defined on time interval $[0, h]$ is a nonempty *acyclic set*, i.e., has trivial reduced homology [33]. Acyclic sets have to be simply connected (and therefore also connected). Also, any contractible set is acyclic.

Theoretical results in [11], [14], and [29] guarantee that the algebraic topological tools such as the fixed point index or Conley index are applicable to admissible flows. In Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.88,we prove that the multivalued flow induced by a PC vector field constructed as described in Section 3.3 is admissible.

3.3 Construction

The goal of this section is to construct a PC vector field that induces an admissible multivalued flow on M and is close to an input vector field defined by vector values at the mesh vertices. Our construction consists of the following simple steps:

- Determine $f(\Delta)$ for each triangle Δ.
- Determine $f(\mathbf{e})$ for any exploding or imploding edge $\mathbf{e}$.
- Determine $\mathcal{S}$: mesh vertices need to be stationary in order to ensure admissibility of the flow.

3.3.1 Computing $f(\Delta)$

Simulation vector fields used in Section 7 are defined by vector values given at mesh vertices rather than mesh triangles. For such vector fields, we define $f(\Delta)$ as the perpendicular projection of the average of vector values at the vertices of Δ to the Δ's plane. Note that the algorithm can be used with other choices of $f(\Delta)$.

3.3.2 Flow along an Exploding or Imploding Edge

At this point, the definition given in Section 3.1 can be used to classify mesh edges as exploding, imploding, or crossing.

To each imploding or exploding edge $\mathbf{e} = \overline{ab}$, we assign the vector $f(\mathbf{e})$ specifying the direction of the flow along e. $f(\mathbf{e})$ is the perpendicular projection of a weighted average $w_0 f(\Delta_0) + w_1 f(\Delta_1)$ onto $\mathbf{e}$, where Δ_0 and Δ_1 are $\mathbf{e}$'s incident triangles. We use $w_i = \alpha_i / (\alpha_0 + \alpha_1)$, where

$$\alpha_i = \sqrt{f(\Delta_{1-i})^2 - (f(\Delta_{1-i}) \cdot \vec{e})^2} \; (i \in \{0, 1\})$$

and $\vec{e}$ is a unit vector parallel to $\mathbf{e}$.

The weights are designed so that if Δ_0 and Δ_1 are coplanar, the weighted average is parallel to $\mathbf{e}$, and therefore $f(\mathbf{e})$ belongs to the convex hull of $f(\Delta_0)$ and $f(\Delta_1)$. This is motivated by the theory of differential inclusions [11]. Solution sets of differential inclusions tend to be more regular for convex-valued vector fields. This has been confirmed by our early experiments, that were originally based on weights $w_0 = w_1 = 0.5$. This choice typically leads to slightly larger transition graphs and slightly higher number of Morse sets.

In principle, the construction can be carried over with $f(\mathbf{e})$ defined as *any* nonzero vector pointing along the edge, although we generally recommend to use the weighting scheme described above for cleaner results. The approximation result in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.88 holds if $f(\mathbf{e})$ is a weighted average of $f(\Delta_0)$ and $f(\Delta_1)$. Also, the algorithm is insensitive to the magnitude of the vector $f(\mathbf{e})$: the output depends only on its direction.

3.3.3 Degenerate Cases

A few types of degeneracies can arise in our construction.

First, $f(\Delta)$ computed as described in Section 3.3.1 can be zero. If this is the case, we treat $f(\Delta)$ as an infinitesimal vector pointing in an arbitrary direction parallel to Δ.

Second, $f(\Delta)$ can be parallel to one of Δ's edges, $\mathbf{e}$. In this case, we simulate an infinitesimal perturbation of that vector to make it not parallel to $\mathbf{e}$. In our implementation,

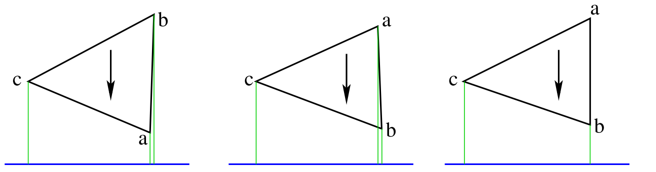

Fig. 6. Projection along $f(\Delta)$ at or near degeneracy. Green lines show the numerically computed projections of Δ's vertices. In the left and center figures, projection of a is strictly between the projections of b and c. In the case shown on the left, edges $\bar{ab}$ and $\bar{ac}$ have to be treated as attracting the flow and in the case shown in the center—as repelling the flow. In the case shown in the right, numerically computed projections of a and b are the same, so one can choose the status of $\bar{ab}$ (i.e., whether it attracts or repels the flow) arbitrarily to simulate an infinitesimal perturbation of $f(\Delta)$.

this boils down to treating $\mathbf{e}$ as either attracting or repelling the flow in Δ. We make sure that this choice is consistent with the projection along $f(\Delta)$, used in the transition graph refinement (Section 4.2.2). If the *numerically computed* projection of a vertex v of Δ is strictly between projections of the other two, then *both* edges incident upon v have to either attract or repel the flow in Δ (Fig. 6). Our implementation uses the projection to determine which edges attract and which repel the flow to enforce consistency for all triangles.

Finally, $f(\mathbf{e})$ can be zero or undefined, which happens if $\alpha_0 = \alpha_1 = 0$. Then, we treat $f(\mathbf{e})$ as an infinitesimal nonzero vector pointing in arbitrarily chosen direction along $\mathbf{e}$.

3.3.4 Stationary Vertices

At this point, we know all nonzero vectors in $F_*(x)$ for any point $x \in M$ (Section 3.2.1). The only component of the definition of a PC vector field that has not been determined yet is the set $\mathcal{S}$ of stationary vertices.

The decision whether a vertex v should be included in $\mathcal{S}$ is based on the sector structure of v. Its objective is to ensure admissibility of the flow. To define the sector structure of a vertex in a PC vector field, one can use a simple variation of the definition in [38] and [39]. Pick a small neighborhood U of v. Hyperbolic sectors in the vicinity of v are formed by trajectory segments contained in U that both start and end on the boundary of U and do not pass through v. Elliptic sectors consist of trajectory segments contained in U that both start and end at v. Unstable parabolic sectors are unions of trajectory segments that start at v and end on the boundary of U. Stable parabolic sectors are unions of trajectories that start on the boundary of U and end at v.

A number of examples are shown in Figs. 3 and 7, where trajectories in hyperbolic, elliptic, stable parabolic, and unstable parabolic sectors are shown in blue, brown, green, and red, respectively. Note that in some cases, parabolic sectors degenerate to a single line. For example, the vertex shown in Fig. 7d has two degenerate stable sectors and one degenerate unstable sector.

Sector structure of a vertex v in a PC vector field can be defined and analyzed using the approach of [38], [39], as described in Section 3.3.5. It turns out that a vertex v needs to be declared stationary if

- v has at least one elliptic sector, or
- the number of v's unstable parabolic sectors is other than 1 or the number of its stable parabolic sectors is other than 1. This case includes all sources and sinks, also of spiral type.

We include a formal proof of admissibility of the resulting flow in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.88. Here, we illustrate the argument on a number of examples as shown in Fig. 7

a. A vertex with an elliptic sector. There are periodic trajectories, shown in brown, that are arbitrarily close to the vertex. Since they accumulate at the vertex, it has to be stationary by upper semicontinuity of the flow.
b. A spiral sink has to be stationary, since otherwise no trajectory would start at it.
c. A source also needs to be declared stationary. To see why, assume that it is not. Consider the set $\Phi(v, t)$ for a small positive t. It is a polygonal loop around the vertex (shown in magenta), since the trajectories are not allowed to stay at the vertex for any positive time. Since there is no imploding edge incident to a vertex, for each point on the polygonal loop there is a unique trajectory in $S(v, t)$ ending at that point. This one-to-one correspondence can be used to argue that $S(v, t)$ is a topological circle and therefore is not acyclic. A spiral source (Fig. 3c) also has to be stationary by the same argument.
d. A saddle-like vertex with two unstable sectors (red; one extends along the edge $\mathbf{e}$ to the right of the vertex). If trajectories are not allowed to stay at v, $\Phi(v, t)$ is disconnected: it consists of the polygonal line to the left of v and a single point on $\mathbf{e}$, both shown in magenta. Thus, $S(v, t)$ can be split into two closed sets, consisting of trajectories ending in the same component of $\Phi(v, t)$: it is not connected and hence also not acyclic.
e. A vertex with one stable, one unstable, and no elliptic sectors. This one can be treated as nonstationary without breaking the desirable properties of the flow. For each point of $\Phi(v, t)$, the magenta line,

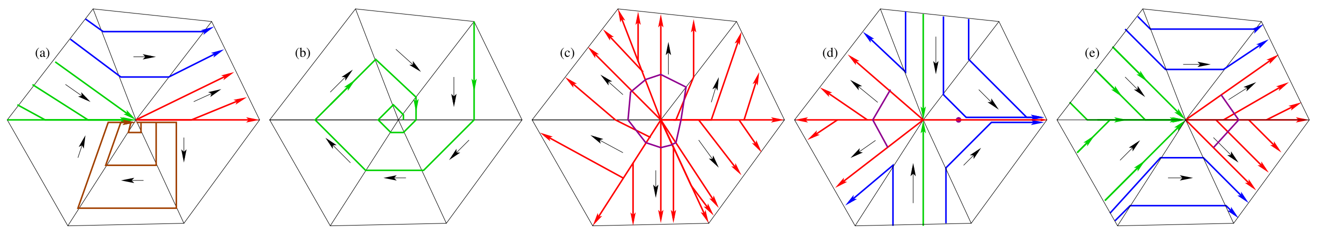

Fig. 7. Should a vertex be declared stationary? It should if it has precisely one stable and one unstable parabolic sector and no elliptic sectors.

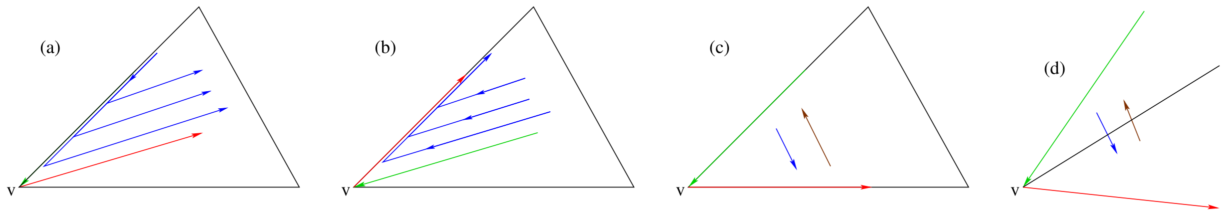

Fig. 8. Examples illustrating how hyperbolic and elliptic sectors can be distinguished. In (a) and (b), one of the sector boundaries points into the interior of a triangle and the other along an edge of that triangle. Such sectors are always hyperbolic. If the sector contains exactly one triangle (c), the sector type is determined based on the vector field in that triangle. More precisely, the sector is hyperbolic if the edge containing the unstable direction attracts the flow (for example, vector field inside the triangle is consistent with the blue arrow). Otherwise, the sector is elliptic (brown arrow). In any other case, there is at least one edge within the sector and it has to be a crossing edge (otherwise, the stable and unstable directions would not be consecutive) as shown in (d). In these cases, sector type is determined based on the direction in which the flow crosses that edge.

there is unique trajectory ending at that point. This correspondence can be used to argue that $S(x_0, t)$ is homeomorphic to a closed interval and therefore is contractible.

3.3.5 Structure of the Flow Near a Vertex

The procedure described in Section 3.3.4 is based on sector structure of a vertex. We perform the sector analysis using the method of [38], [39]. However, our setting is slightly different. First, we work in the multivalued PC vector field setting. Second, the analysis of a vertex v needs to be done *before* $F_*(v)$ is fully determined, i.e., without knowing if v is stationary. Therefore, we include a brief description of the sector analysis procedure in this section. Note that our algorithm requires us only to *count* the number of stable parabolic, unstable parabolic, hyperbolic, and elliptic sectors of each vertex.

First, we determine the stable and unstable directions of v. The set of unstable directions has already been described previously (Section 3.2.1). A similar procedure is used to determine stable directions of v.

If a vertex has no stable or unstable directions, it is a spiral sink, spiral source or possibly a center in the singular case. Our algorithm does not require more detailed analysis of these vertices. They are treated as stationary points.

In what follows, we assume there is at least one stable or unstable direction. We scan the stable and unstable directions in counterclockwise order around the vertex. Consecutive sequences of stable (respectively, unstable) directions define stable (unstable) parabolic sectors. A number of examples can be seen in shown in Figs. 3 and 7, where the stable sectors are shown in green and unstable sectors—in red. Note that, in some cases, the stable or unstable parabolic sectors reduce to a line. Pairs of consecutive directions of distinct types (one stable, one unstable) define boundaries of a hyperbolic or elliptic sector. Such sectors are shown in blue or brown in Figs. 3 and 7. In a hyperbolic sector, the flow moves from the stable to the unstable sector boundary (as seen by an observer at v). In an elliptic sector, it moves the other way. All cases that can be encountered when distinguishing hyperbolic and elliptic sectors are shown in Fig. 8.

4 Transition Graph

A PC vector field F_* on a manifold mesh M can be represented by a finite directed *transition graph* defined in this section. The abstract definition of the transition graph is given in Section 4.1. An algorithm for constructing the transition graph is described in Section 4.2.

4.1 Preliminaries

By an *edge piece*, we mean a closed line segment contained in an edge of M. We say that a finite set of edge pieces P forms a subdivision if the following two conditions are satisfied:

1. The union of edge pieces in P is the same as the union of all edges of M (denoted by M_1).
2. Any two edge pieces in P are either disjoint or intersect at a single point.

The nodes of a transition graph $\mathcal{G}$ are the elements of $V \cup P$, where V is the set of vertices of M and P is a set of edge pieces that form a subdivision. Thus, a node is of one of two types: it either corresponds to a vertex of M or to an edge piece in P. In what follows, we call elements of $V \cup P$, i.e., vertices or edge pieces, *n-sets* for brevity.

We require $\mathcal{G}$ to represent all trajectories of the vector field in the following sense. For any trajectory, let us record the consecutive n-sets visited by it, giving priority to vertices over edge pieces, i.e., at the moment a trajectory hits a vertex, recording that vertex but not any of the edge pieces that meet at it. The resulting sequence of n-sets has to be a path in $\mathcal{G}$.

The above requirement is guaranteed to hold if the arc $\mathbf{a} \rightarrow \mathbf{b}$ belongs to $\mathcal{G}$ for any pair of distinct nodes $\mathbf{a}, \mathbf{b}$ whose corresponding n-sets (also denoted by $\mathbf{a}$, $\mathbf{b}$) are *connected by a simple trajectory segment* defined on nonzero-length time interval. By a simple trajectory segment, we mean a trajectory segment $\sigma : [0, t] \rightarrow M$, contained in a single mesh triangle that has constant velocity $\dot{\sigma}$. In particular, such segments stay at a stationary point or move along a straight line. σ connects $\mathbf{a}$ to $\mathbf{b}$ if $\mathbf{a}$ is a minimal n-set containing $\sigma(0)$, $\mathbf{b}$ is a minimal n-set containing $\sigma(t)$ and any point on σ contained in M_1 (the union of all edges of M) is also in $\mathbf{a}$ or $\mathbf{b}$ ($\sigma([0, t]) \cap M_1 \subset \mathbf{a} \cup \mathbf{b}$). A minimal n-set containing a point p is an n-set containing p such that no other n-set contained in it contains p. Thus, if p is at a mesh vertex, the minimal n-set containing p is the vertex itself. If p is not at a vertex but belongs to an edge, any edge piece containing p is a minimal n-set containing it. Finally, there is no minimal n-set containing a point in the interior of a mesh triangle.

4.2 Construction of Transition Graph

We store the transition graph in a standard directed graph data structure. Each node contains a pointer to the corresponding n-set as well as separate lists of arcs out of and into the node.

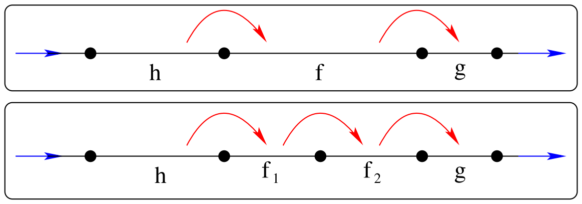

Fig. 9. Type E arcs generated as a result of refinement. The edge piece $\mathbf{f}$ on an imploding or exploding edge is split into edge pieces $\mathbf{f}_1$ and $\mathbf{f}_2$. The vector field points to the right (blue arrow). The arcs of $\mathcal{G}$ into and out of $\mathbf{f}$ (top) and arcs of $\mathcal{G}'$ into and out of the two new nodes (bottom) are shown in red. Note that there is a trajectory moving along the edge from left to right, which is reflected by the arcs of the graph both before and after refinement.

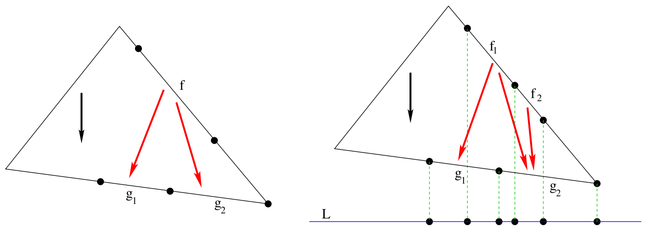

Fig. 10. Refining type T arcs. Left: arcs of $\mathcal{G}$ connecting three edge pieces are shown in red. The vector field inside the triangle points down. Right: refinement of $\mathbf{f}$. The green lines show the parallel projection transformation P in the direction of the vector field. Since $P(\mathbf{f}_1)$ intersects both $P(\mathbf{g}_1)$ and $P(\mathbf{g}_2)$, arcs $\mathbf{f}_1 \to \mathbf{g}_1$ and $\mathbf{f}_1 \to \mathbf{g}_2$ are added to $\mathcal{G}'$. For a similar reason, so is $\mathbf{f}_2 \to \mathbf{g}_2$. Note that this is consistent with the definition of the transition graph given in Section 4.1: these arcs have to be in $\mathcal{G}'$ since their starting and end edge pieces are connected by simple trajectory segments running through the triangle.

Our algorithm first builds the coarse graph (Section 4.2.1) and then refines it using refinement operations (Section 4.2.2). In this section, we focus on describing the refinement operation itself. An example of an adaptive refinement strategy is described in Section 5.3. The coarse transition graph has the properties outlined in Section 4.1. The transition graph refinement procedure is designed to preserve them.

4.2.1 Initialization: Coarsest Level

On the coarsest level, $\mathcal{G}$ is based on the coarsest possible subdivision P, whose edge pieces are the mesh edges. Thus, nodes of $\mathcal{G}$ correspond to edges and vertices of M. $\mathcal{G}$ is built as described below.

First, for each imploding or exploding edge $\mathbf{e} = \bar{u}\bar{v}$ of the mesh we add arcs $u \to \mathbf{e}$ and $\mathbf{e} \to v$ if $f(\mathbf{e})$ points from u to v and arcs $v \to \mathbf{e}$ and $\mathbf{e} \to u$ if it points the other way. In what follows, we call the arcs created this way *type E* arcs. With each such arc, we keep a pointer to the edge that gave rise to it (i.e., the edge $\mathbf{e}$).

Then we add arcs that link pairs of nodes corresponding to edges and vertices of a triangle that are connected by trajectory segments moving through its interior. These arcs are called *type T* arcs later on. For any triangle Δuvw, either one or two edges of Δ attract the flow in Δ. In the first case, we add arcs from the nodes corresponding to the two edges that repel the flow and the vertex between them to the node corresponding to the edge that attracts the flow (for example, if $\bar{v}\bar{w}$ is the edge that attracts the flow, $u \to \bar{v}\bar{w}$, $\bar{u}\bar{v} \to \bar{v}\bar{w}$, and $\bar{u}\bar{w} \to \bar{v}\bar{w}$). In the second case, we add arcs from the edge that repels the flow to each of the two edges that attract the flow and the vertex between them. With each type T arc, we keep a pointer the triangle Δ which was used to generate that arc.

It is convenient to include stationary point information in the transition graph. For each stationary vertex v, we add the *type S* arc $v \to v$, connecting v to itself. By doing this, we ensure that stationary points are contained in strongly connected components of the transition graph and therefore require no special treatment in Section 5.

Finally, our implementation removes nodes corresponding to mesh vertices with no stable or unstable directions (spiral sinks, spiral sources, or centers, Section 3.3.5). These vertices form isolated connected components of $\mathcal{G}$. Morse sets containing them are detected and classified using the general approach described in Section 5, since spiraling flow causes edge pieces incident upon them to form loops in $\mathcal{G}$.

4.2.2 Refinement

A local refinement operation corresponds to splitting one of the edge pieces $\mathbf{f}$ in the subdivision associated with $\mathcal{G}$ into two, $\mathbf{f}_1$ and $\mathbf{f}_2$. The node $\mathbf{f}$ is removed from $\mathcal{G}$ (together with all arcs into and out of it) and replaced by the nodes $\mathbf{f}_1$ and $\mathbf{f}_2$ with a set of new incident arcs computed as described below. In what follows, $\mathcal{G}$ and $\mathcal{G}'$ denote the transition graph before and after refinement, respectively. It remains to describe the arcs in $\mathcal{G}'$ into and out of the new nodes, $\mathbf{f}_1$ and $\mathbf{f}_2$.

To construct arcs out of each of the new nodes, we scan arcs out of $\mathbf{f}$ in $\mathcal{G}$. Each such arc $\vec{a} = \mathbf{f} \to \mathbf{g}$ will induce a number of arcs in $\mathcal{G}'$, all of them of the same type (E or T) and with the same associated mesh element as $\vec{a}$.

If $\vec{a}$ is of type E, the new arcs are generated as follows: if $\mathbf{g}$ and $\mathbf{f}_1$ intersect, we include arcs $\mathbf{f}_2 \to \mathbf{f}_1$ and $\mathbf{f}_1 \to \mathbf{g}$ in $\mathcal{G}'$. Similarly, if $\mathbf{g}$ and $\mathbf{f}_2$ intersect, we add arcs $\mathbf{f}_1 \to \mathbf{f}_2$ and $\mathbf{f}_2 \to \mathbf{g}$ to $\mathcal{G}'$. This case is illustrated in Fig. 9. Note that the figure also shows an arc *into* one of the new nodes, described later in this section.

Now, assume $\vec{a}$ is of type T and Δ is its associated triangle. Let $P : \Delta \to L$ be a parallel projection transformation that projects Δ to a line perpendicular to $f(\Delta)$, with projection direction $f(\Delta)$. If $P(\mathbf{f}_i)$ $(i \in \{1,2\})$ intersects $P(\mathbf{g})$, we include the arc $\mathbf{f}_i \to \mathbf{g}$ in $\mathcal{G}'$. An example illustrating type T arc refinement is shown in Fig. 10.

A similar procedure is applied to generate arcs into the new nodes. We scan all arcs $\vec{a} = \mathbf{h} \to \mathbf{f}$ into $\mathbf{f}$. If $\vec{a}$ is of type E, the arc $\mathbf{h} \to \mathbf{f}_i$ is included in $\mathcal{G}'$ if $\mathbf{h}$ and $\mathbf{f}_i$ intersect (for $i \in \{1,2\}$). If $\vec{a}$ is of type T the arc $\mathbf{h} \to \mathbf{f}_i$ $(i \in \{1,2\})$ is added to $\mathcal{G}'$ if $P(\mathbf{f}_i)$ and $P(\mathbf{h})$ intersect.

Type S arcs are not affected by the refinement since they start and end at a node corresponding to a mesh vertex. Such nodes are never refined.

5 Morse Decomposition

In this section, we describe the algorithm for computing a Morse decomposition and classifying the Morse sets.

We use the following variant of definition of a Morse decomposition [7]. We say that a trajectory $\sigma : (-\infty, \infty) \to M$ *links* a set $C \subset M$ to a set $C' \subset M$ if and only if it converges to C if followed backward and to C' if followed forward, i.e.,

$$\lim_{t \to -\infty} \operatorname{dist}(\sigma(t), C) = \lim_{t \to \infty} \operatorname{dist}(\sigma(t), C') = 0.$$

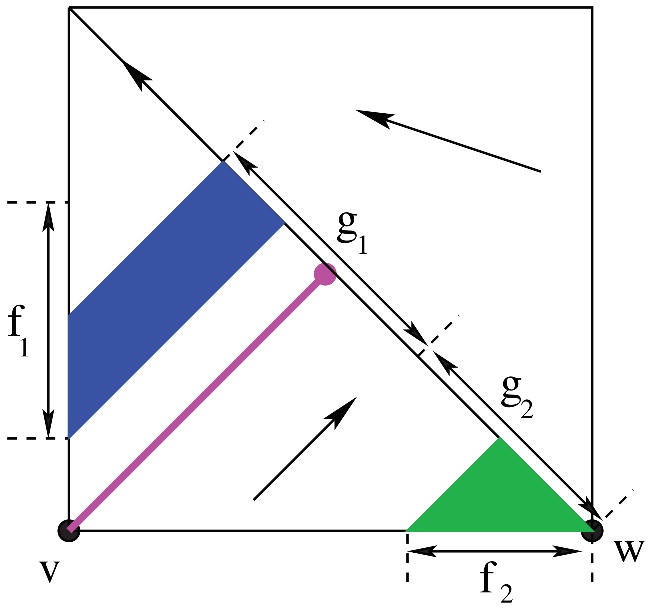

Fig. 11. Examples of sets represented by an arc in $\mathcal{G}$. The arc $\mathbf{f}_1 \to \mathbf{g}_1$ represents the blue quadrilateral, $\mathbf{v} \to \mathbf{g}_1$—the magenta line and $\mathbf{f}_2 \to \mathbf{g}_2$—the green triangle. The arc $\mathbf{w} \to \mathbf{g}_2$ (note that the edge containing $\mathbf{g}_1$ and $\mathbf{g}_2$ is imploding) represents the union of the segment $\mathbf{g}_2$ and the vertex $\mathbf{w}$, i.e., $\mathbf{g}_2$. The arc $\mathbf{g}_2 \to \mathbf{g}_1$ represents $\mathbf{g}_1 \cup \mathbf{g}_2$.

A family $\mathcal{C}$ of disjoint closed subsets of M forms a Morse decomposition if and only if 1) any trajectory passing through a point outside the union of all sets in $\mathcal{C}$ links two different sets in $\mathcal{C}$, and 2) $\mathcal{C}$'s *linkage graph* is acyclic. The nodes of the linkage graph are the sets in $\mathcal{C}$. An arc $C_1 \to C_2$ belongs to the linkage graph if and only if there is a trajectory that links C_1 to C_2.

Acyclicity of the linkage graph forces the dynamics outside the union of Morse sets to be free of recurrence. Each periodic orbit and stationary point is contained in one of the Morse sets by condition 1.

5.1 Morse and Pseudo-Morse Sets

Strongly connected components of the transition graph can be computed using the algorithm of [35]. They define Morse sets for a PC vector field. The precise argument is outlined in Appendix C, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.88. To our best knowledge, our setting has not yet been described in the mathematical literature.

Since the definition of Morse sets in terms of strongly connected components of the transition graph is complicated and they would be expensive to compute exactly, we use simpler *supersets* of Morse sets (that we call *pseudo-Morse sets*) for visualization purposes. In this section, we describe the construction of pseudo-Morse sets. Note that pseudo-Morse sets are not guaranteed to be disjoint, but they have disjoint interiors (Appendix D, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.88) and their size tends to quickly decrease as the transition graph is refined (Section 7), so they generally give a good idea of the spatial distribution of features described by the Morse decomposition.

For an arc $\vec{a} = \mathbf{f} \to \mathbf{g}$ of $\mathcal{G}$, where $\mathbf{f}$ and $\mathbf{g}$ are n-sets, the subset of M represented by $\vec{a}$ is the union of all simple trajectory segments (Section 4.1) $\sigma : [0, t] \to M$, that start in $\mathbf{f}$ and end in $\mathbf{g}$. Examples are shown in Fig. 11. Generally, a set represented by an arc can be a triangle, a quadrilateral, a line segment or a mesh vertex (if it is a stationary point).

A set of nodes A of $\mathcal{G}$ represents the union of all sets represented by arcs that both start and end in A and all edge pieces and vertices corresponding to nodes in A. The subset represented by A is denoted by $\mathcal{R}(A)$.

Pseudo-Morse sets are sets represented by strongly connected components of the transition graph.

5.2 Classification

In order to classify a Morse set C defined by a strongly connected component A of the transition graph, we first compute its fixed point index with respect to the translation by a small time t along the flow. The index is the sum of Poincaré indices of the stationary points in C (by additivity property of the index, [11]). In the PC case, the stationary points only occur at mesh vertices. The index of a stationary point is equal to $1 + \frac{e-h}{2}$, where h and e are the numbers of its hyperbolic and elliptic sectors [8]. The index of C is equal to the sum of 1) indices of nodes in A that correspond to stationary vertices of the PC vector field, and 2) indices of vertices with no stable or unstable directions (centers, spiral sinks, and saddles) whose incident edge pieces are in A. Recall that our implementation removes nodes corresponding to such vertices from the graph (Section 4.2.1).

Then, we determine if C is attracting, repelling or neither. C is attracting if and only if there are no arcs in the transition graph from a node in C to a node outside C and therefore flow cannot leave C. Similarly, C is repelling if and only if there are no arcs from a node outside C to a node in C and therefore no trajectory can enter C from the outside.

We say that a Morse set C whose index is i is of type $(i, +)$, $(i, -)$, or $(i, 0)$ if it is repelling, attracting or neither, respectively. This simple classification scheme is surprisingly powerful. In particular, it allows one to distinguish Morse sets that enclose different kinds of basic flow features since they are of distinct types. Namely, sinks are of type $(1, -)$, sources—of type $(1, +)$, saddles—of type $(-1, 0)$, and periodic orbits—of type $(0, +)$ if repelling and $(0, -)$ if attracting. In what follows, we call Morse sets of these types *simple*.

Conversely, Morse sets of type $(0, +)$ or $(0, -)$ are guaranteed to contain a periodic orbit if they do not contain a stationary point by the Poincaré-Bendixon theory [15]. Morse sets of nonzero index (in particular, of types $(1, -)$, $(1, +)$ $(-1, 0)$) must contain a stationary point. Morse sets of type $(0, 0)$ are *trivial*: they contain features that cancel each other or no features at all. We call nontrivial Morse sets that are not of a simple type *complex*.

The Morse set type described above carries information equivalent to its Conley index [7] under certain technical assumptions (existence of a connected index pair for the Morse set) by the results of [28].

5.3 Adaptive Transition Graph

To obtain Morse decompositions of increasing precision, we adaptively refine the transition graph. First, we compute the coarse transition graph $\mathcal{G}_0$ as described in Section 4.2.1.

Given a graph $\mathcal{G}_i$, we compute its strongly connected components. An optional *cleanup step* removes all nodes of

$\mathcal{G}_i$ that are not connected to a node in a strongly connected component (by an arc directed in any way), together with their incident edges. Then, we apply the *refinement step* to $\mathcal{G}_i$, i.e., refine every node in a strongly connected component that corresponds to an edge piece by splitting this edge piece into two of equal length. The refinement step yields the transition graph $\mathcal{G}_{i+1}$.

A simple way to obtain a Morse decomposition is to compute it from the transition graph $\mathcal{G}_N$ for a prescribed number of refinement iterations N. Intermediate results can be used to produce results for any smaller number of refinement iterations with little overhead. N can be viewed as a natural parameter controlling the precision of the output Morse decomposition. Clearly, other refinement criteria can easily be used with our approach. For example, if the goal is to describe the vector field in terms of its basic features, the Morse sets could be refined until all of them are of simple types. Experimental results, described in Section 7, indicate that Morse sets of nonsimple types tend to disappear after a small number of refinement operations. Refinement could focus on large and complex Morse sets (as in [4]) in hope of obtaining smaller and simpler Morse sets. In an interactive system, one can let the user select Morse sets to be refined. Potentially, one might hope that edge pieces can be split into nonequal parts for more optimal results. We leave these issues for future investigation.

For any refinement strategy, any Morse set obtained from a finer transition graph is contained in a Morse set obtained using a coarser graph. Furthermore, Morse sets defined by strongly connected components whose nodes are not refined stay the same. Pseudo-Morse sets have the same properties. A proof is included in Appendix E, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.88.

By restricting refinement steps to strongly connected components, we slow the growth of the transition graph's size and therefore speed up the algorithm and reduce its memory requirements. Note that the cleanup stage is designed to leave enough information in the transition graph to enable one to classify the Morse sets correctly (Section 5.2). While the cleanup stage can make the size of the transition graph much smaller, in some cases it is not desirable. For example, it discards information about connecting trajectories between Morse sets, that are represented by paths in $\mathcal{G}$ connecting different strongly connected components.

6 Complexity Analysis

In this section, we analyze the complexity of our algorithm.

Clearly, the assignment f (Sections 3.3.1 and 3.3.2) can be computed in linear time. Local analysis of the flow near a vertex v (Section 3.3.5) can be implemented in linear time in the degree of v. This is because the total number of stable and unstable directions at v cannot be higher than twice the degree of v (there is at most one direction pointing into each incident triangle and at most one pointing along any incident edge). The directions can be generated in order around v using the mesh triangle and edge incidence information, so that no sorting is necessary. Time needed to build the coarse transition graph (Section 4.2.1) is also linear.

Now, we argue that the total running time of the ith refinement step (Section 5.3) is linear in the size of the transition graph $\mathcal{G}_{i-1}$, i.e., the transition graph at the beginning of that refinement step. The strongly connected components can be computed in linear time [35]. Clearly, the cleanup stage can be implemented in linear time as well. Refining a node (Section 4.2.2) requires time linear in the degree of that node. The sum of all degrees is equal to twice the number of arcs. Still, there is a technical issue to overcome: refinement of neighbors of a node may raise its degree before it is refined. However, one can argue that the degree cannot increase by a factor more than two in the refinement scenario of Section 5.3. This is because refinement replaces a node $\mathbf{f}$ with two nodes; therefore, it can replace an arc connecting another node to $\mathbf{f}$ with at most two new arcs. Therefore, the total degree of all nodes at the time of refinement is linear in the size of $\mathcal{G}_{i-1}$, and so is the total running time of the refinement stage.

The growth of the size of $\mathcal{G}_{i-1}$ as a function of i depends on the vector field. In the worst case (if $\mathcal{G}_{i-1}$ is strongly connected), all edges are refined and the graph size can grow by close to a factor of 2 for large i. In practice, Morse sets get smaller as the graph is refined and the growth of the graph size is much slower.

7 Experimental Results

In this section, we describe Morse decompositions obtained using our algorithm for three simulation data sets obtained by extrapolating velocity data from a 3D fluid flow simulation to the boundary of the model [16], [20] and gradient vector fields derived from scalar fields on triangle meshes. Section 7.1 discusses results for the simulation data sets. We compare results obtained using our approach to results obtained using other methods that perform analysis of vertex-based vector fields in Section 7.2. Finally, in Section 7.3 we discuss results for gradient vector fields.

All images shown in this section except for Figs. 15(left) and 16, are obtained using the image-based LIC visualization algorithm of [19], applied directly to the PC vector field. Therefore, in some images, the flow has a polygonal look. We render the pseudo-Morse (Section 5.1) sets to visualize the Morse sets. Trivial Morse sets (of type $(0,0)$) are shown in magenta. Repelling Morse sets of types $(1,+)$ and $(0,+)$ are shown in red (those of type $(1,+)$ are slightly brighter). Attracting Morse sets of type $(1,-)$ and $(0,-)$ are shown in green (also in this case, type $(1,-)$ sets are slightly brighter). Morse sets of type $(-1,0)$, that in the generic case contain saddles, are shown in blue. Complex Morse sets are shown in black. Note that after a large number of refinement steps, some Morse sets may become small and hard to see. Morse sets consisting of a single vertex are rendered as antialiased points (small discs).

7.1 Simulation Data Sets

Experiments on the simulation data sets demonstrate both precision and efficiency of our approach.

First, the sizes of the Morse sets rapidly decrease with the number of refinement iterations (Figs. 12, 13, and 14).

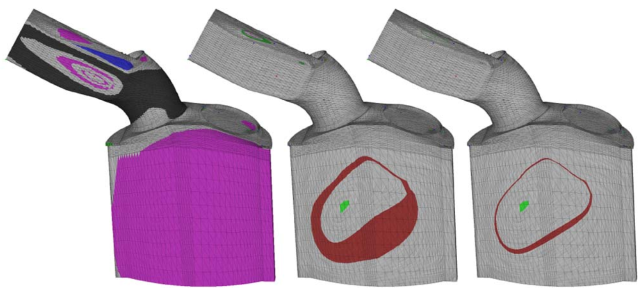
Fig. 12. Results for the gas engine data set for 1, 5, and 9 refinement steps, showing the mesh edges. The large periodic orbit has been localized very accurately in the image on the right.

Morse sets obtained from a fine transition graph provide a precise bound on stationary points or periodic orbits that they enclose.

Second, for any of the three data sets, our algorithm is able to produce Morse decompositions that do not contain a complex Morse set. In fact, complex Morse sets disappear after a small number of refinement steps (the "complex" column in Tables 1, 2, and 3). This behavior is highly desirable, since complex Morse sets are harder to understand. Moreover, the number of Morse sets of each of the simple types tends to stabilize as refinement steps are applied. After a certain number of iterations, few, if any, new flow features tend to be discovered. Additional iterations only decrease the size of the Morse sets that contain the already discovered ones.

Trivial Morse sets generally contain no features or features that cancel each other. Their number as a function of the number of refinement iterations varies in a less predictable way. As refinement iterations are applied, they often appear near periodic orbits or stationary points that are weakly attracting or repelling (Fig. 13, right—see the closeup of the feature in front of the jacket head). They could also appear near a saddle that is close to having a homoclinic orbit (Fig. 14). Generally, after a few refinement steps, the trivial Morse sets seem to indicate *almost recurrent* dynamics:

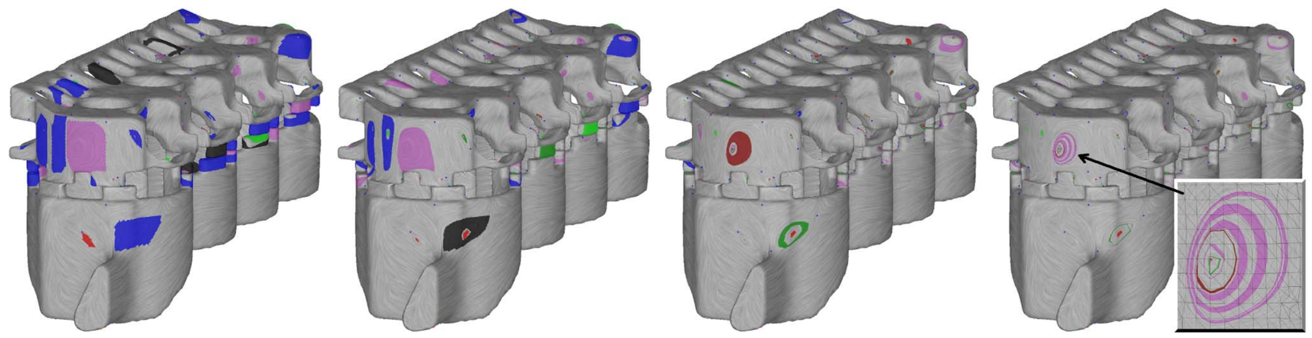
Fig. 13. Results for the cooling jacket data sets for 1, 3, 5, and 7 refinement steps.

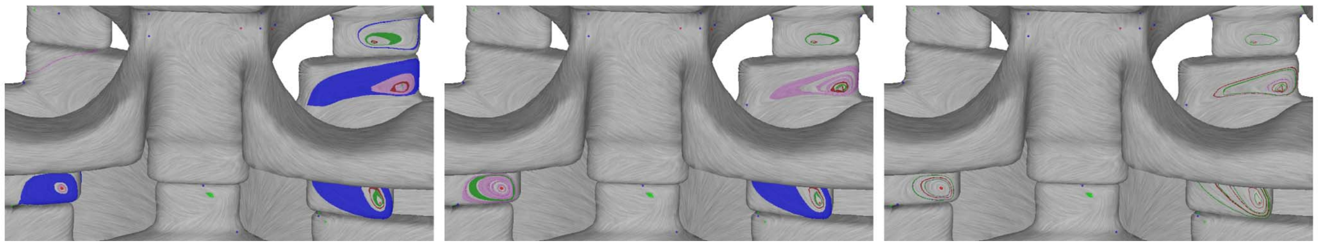
Fig. 14. A closeup view of the Morse decomposition for the cooling jacket data sets (5, 6, and 10 refinement steps). The finest decomposition (right) shows a number of Morse sets enclosing periodic orbits. The coarsest decomposition (left) contains a number of Morse sets of type $(-1, 0)$ (blue) that are topological rings. One refinement step causes a saddle to split off from three of them, leading to creation of a number of Morse sets of zero index. The thin blue loop Morse set in the top right corner (left image) is refined to a single saddle. Note that blue Morse sets that are topological rings typically arise from loops in the graph that start and end at a saddle and therefore they indicate the existence of homoclinic orbits for a small perturbation of the vector field.

TABLE 1
Statistics for the Gas Engine Data Sets

	with cleanup			without cleanup			Morse sets						
refinement	total	Graph size (last step)		total	Graph size (last step)								
steps	time [s]	nodes	arcs	time [s]	nodes	arcs	trivial	$(1,-)$	$(1,+)$	$(-1,0)$	$(0,-)$	$(0,+)$	complex
0	0.61	52,593	82,380	0.64	52,593	82,380	1	11	12	13	0	0	3
1	0.76	42,571	73,851	0.78	70,391	118,599	6	13	12	20	0	0	1
2	0.92	43,922	80,949	0.92	89,899	158,331	4	13	12	23	0	0	0
3	1.09	47,194	90,716	1.12	111,712	202,953	2	15	13	26	1	1	0
4	1.23	35,382	70,026	1.33	128,548	237,512	13	15	13	26	2	1	0
5	1.38	47,250	93,911	1.52	151,219	283,887	4	14	14	26	4	1	0
6	1.49	19,998	39,872	1.68	160,670	303,422	5	14	14	26	4	1	0
7	1.61	17,659	35,794	1.85	169,001	320,942	3	14	14	26	4	1	0
8	1.71	22,761	47,609	2.05	179,903	344,370	2	14	14	26	4	1	0
9	1.82	29,205	63,591	2.31	194,028	375,785	3	14	14	26	4	1	0

The mesh has 26,298 triangles and 13,151 vertices and genus 0.

TABLE 2
Statistics for the Diesel Engine Data Sets (221,574 Triangles, 110,789 Vertices, Genus 0)

refinement	with cleanup			without cleanup			Morse sets						
	total	Graph size (last step)		total	Graph size (last step)								
steps	time [s]	nodes	arcs	time [s]	nodes	arcs	trivial	$(1,-)$	$(1,+)$	$(-1,0)$	$(0,-)$	$(0,+)$	comlpex
0	5.02	443,149	672,614	4.89	443,149	672,614	40	10	9	14	0	0	1
1	5.48	51,081	86,202	5.41	462,142	710,851	2	9	10	17	1	0	0
2	5.52	10,772	19,806	5.69	466,868	720,487	5	9	10	17	1	0	0
3	5.65	17,282	32,708	6.13	474,700	736,443	0	9	10	17	1	0	0
4	5.71	22,681	44,648	6.46	485,534	758,527	0	9	10	17	1	0	0
5	5.72	23,682	47,142	6.92	496,954	781,863	0	9	10	17	1	0	0
6	5.81	26,225	52,820	7.29	509,672	808,017	0	9	10	17	1	0	0
7	6.00	32,494	66,183	7.84	525,556	840,868	0	9	10	17	1	0	0
8	6.16	45,611	93,113	8.32	547,992	887,168	0	9	10	17	1	0	0
9	6.38	62,713	127,915	8.80	578,716	950,664	0	9	10	17	1	0	0

TABLE 3
Statistics for the Cooling Jacket Data Sets (227,868 Triangles, 113,868 Vertices, Genus 34)

refinement	with cleanup			without cleanup			Morse sets						
	total	Graph size (last step)		total	Graph size (last step)								
steps	time [s]	nodes	arcs	time [s]	nodes	arcs	trivial	$(1,-)$	$(1,+)$	$(-1,0)$	$(0,-)$	$(0,+)$	complex
0	5.39	455,625	706,940	5.41	455,625	706,940	51	73	61	152	2	0	18
1	6.61	305,282	525,460	6.55	582,176	963,939	42	92	65	206	3	0	7
2	8.19	412,726	765,326	8.25	768,065	1,341,520	58	110	65	233	6	1	4
3	10.32	589,002	1,134,190	10.49	1,043,954	1,902,405	69	121	73	252	18	7	4
4	12.45	554,545	1,089,972	12.67	1,307,952	2,441,227	61	121	82	267	36	13	1
5	14.75	522,849	1,043,296	15.16	1,559,374	2,957,098	86	113	90	269	47	17	0
6	17.08	502,597	1,016,924	17.83	1,802,188	3,459,898	126	111	92	269	57	24	0
7	19.34	491,237	1,007,724	20.55	2,039,231	3,957,571	159	110	93	269	63	28	0
8	21.83	510,227	1,073,214	23.85	2,284,582	4,486,752	137	110	93	269	68	34	0
9	24.55	575,151	1,264,461	27.18	2,562,734	5,111,623	126	110	93	269	71	37	0

trajectories that tend to form tight spirals. Therefore, they may potentially be used as flow complexity indicators.

Finally, our algorithm is efficient. The runtime statistics for the three data sets are shown in Tables 1, 2, and 3. The tables show total running times (in seconds) of the adaptive refinement algorithm described in Section 5.3, both with and without the cleanup stage, on an Intel Q6600 machine with 4 GB of RAM. Note that initialization (building the mesh data structure, analysis of sector structure of vertices—Section 3.3.5 and building the coarse transition graph—Section 4.2.1) typically takes considerably longer than any of the first 10 refinement iterations (Section 5.3). Since the initialization time is included in all running times reported here, the running time for 0 refinement iterations is relatively high. As the tables show, cleanup decreases the size of the transition graph and therefore also time needed for a refinement iteration.

The running times reported in [4] for the gas engine, diesel engine, and cooling jacket data sets are 65, 96, and 435 seconds, respectively (for $\tau_{\max} = 0.4$). The results do not look more detailed than ours for about $3 - 5$ refinement iterations. The precision of their results can be increased by using $\tau_{\max} = \infty$. However, this makes the output less reliable and increases the running times to 213, 1,012, and 4,524 seconds, respectively.

7.2 PC and Vertex-Based Vector Field Features

Fig. 15 shows Morse decompositions of comparable precision computed using the approach of [6] (based on the parallel transport interpolation scheme [41]) and the method presented in this paper. Morse sets obtained using both methods are similar, even though our algorithm analyzes the PC variant of the original vector field. A Morse decomposition obtained using the approach of [4] is compared to a high precision one obtained using our algorithm in Fig. 1. Note that in this case, Morse sets of our decomposition (right) appear to be contained in the Morse sets of the other one (left), possibly up to a small error. This

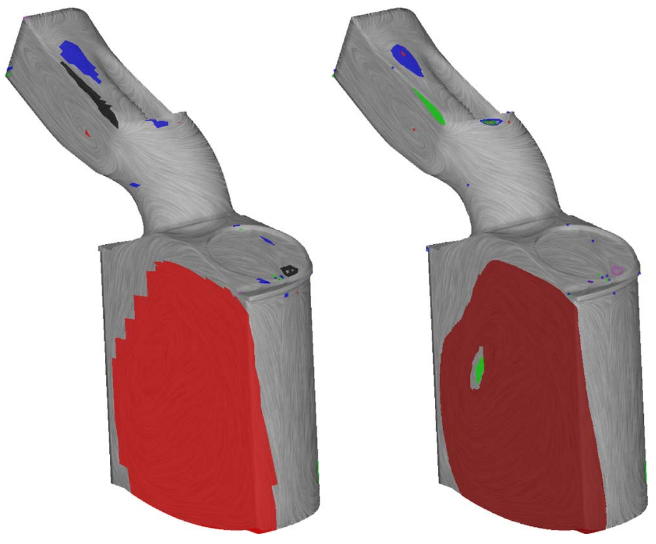
Fig. 15. Left: morse sets of the vertex-based vector field extracted using the τ-map approach of [6], classified using the upper bound on the Conley index [4]. Right: morse sets for the PC variant of the same vector field extracted using our method (three refinement iterations). Note that the colors of similar Morse sets often do not match, for two reasons. First, they are not exactly the same. In particular, they may contain different flow features and hence be of different types. Second, the classification method of [4] may not be accurate and therefore errors in Morse set coloring are possible. However, the Morse sets themselves are similar.

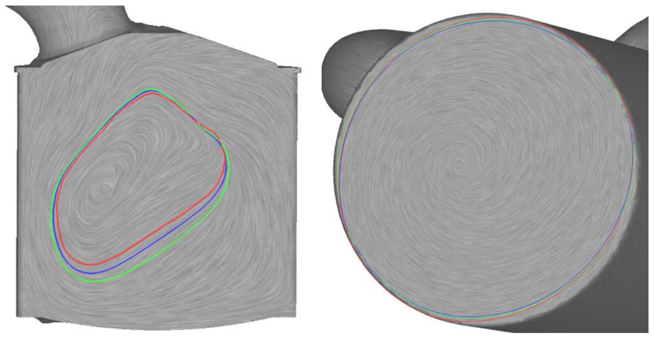
Fig. 16. Periodic orbits of two vertex-based vector fields computed using the algorithm of [5] with the Runge-Kutta method of second (blue) and fourth (green) order. The Morse set containing a similar periodic orbit (for the PC vector field) computed using our method (14 refinement iterations) is shown in red.

means that our Morse decomposition can be viewed as a refinement of that of [4].

Periodic orbits are known to be sensitive to the numerical method used to approximate them as well as perturbation of the vector field [6]. Fig. 16 compares large periodic orbits obtained using two different integration schemes and the Morse set containing a similar periodic orbit for the PC vector field, showing that they are geometrically close.

7.3 Gradient Vector Fields

A natural class of PC vector fields are gradients of piecewise linear scalar functions. Fig. 17 shows Morse sets and connecting regions for a height field derived from the Puget Sound data sets [21]. We smoothed the height field to obtain a smaller number of Morse sets and cleaner looking connecting regions. Note that also in this case, our procedure leads to expected results, generating sinks at peaks and strings of sources and saddles along valley bottoms. Connecting regions contain the separatrices (edges of the Morse complex of the underlying scalar function [2], [9]). They are represented by arcs on paths in $\mathcal{G}$ and on paths in $\mathcal{G}^T$ ($\mathcal{G}$ with arcs reversed) that start in a Morse set of type $(-1, 0)$ (saddle), obtained using a simple algorithm that **1)** computes the paths using the depth-first search algorithm, and **2)** refines nodes on the paths that correspond to edge pieces by splitting each of them into two of equal length. The two steps are repeated a prescribed number of times to increase the precision of the result.

8 Conclusion and Future Work

We introduced an efficient and robust algorithm to compute a Morse decomposition of a vector field on a triangulated manifold surface and accurately classify its Morse sets. For all test data sets, our approach has been able to produce high precision Morse decompositions, whose Morse sets tend to correspond to stationary points and periodic orbits and therefore are easy to interpret. Finally, it is easy to use since it depends on just one parameter that controls the precision of the results. There are several research directions that could potentially arise from this work.

It would be interesting to extend the PC vector field formulation to the three (and, potentially, higher) dimensional cases. We believe that the PC vector field framework is promising for higher dimensions because of its high efficiency, robustness, and relative simplicity.

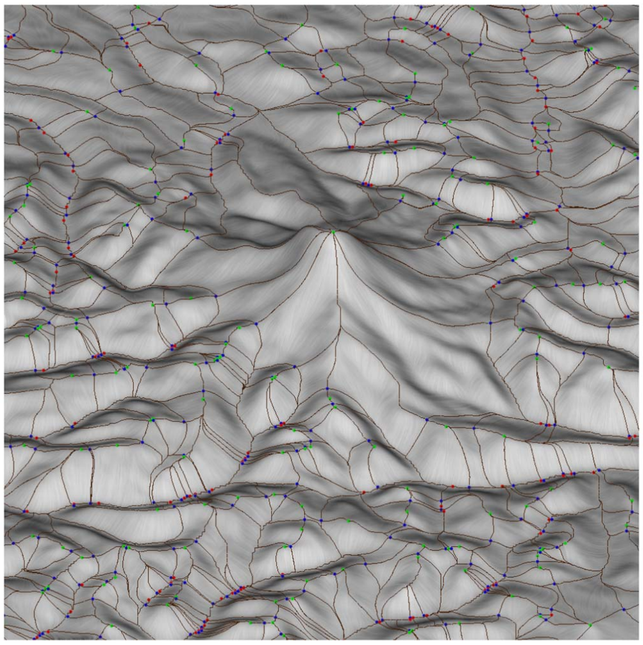
Fig. 17. Morse sets and connecting regions for the terrain model.

It would also be interesting to extend the transition graph approach to vector fields defined by standard interpolation schemes. This will probably require some form of integration (either numerical or analytical) of the flow, but the need to follow trajectory segments across several triangles (required in [6]) could possibly be avoided.

We would like to develop an algorithmic framework for hierarchical Morse decompositions. A natural theoretical basis for such framework is provided by the theory of differential inclusions: increasing the values of a multivalued vector field yields a flow with a richer trajectory structure and therefore produces a coarser Morse decomposition.

Finally, it would be interesting to study theoretical properties of the concepts introduced in this paper. The mathematical literature focuses on different discretizations of the flow: either similar to the τ-map idea of [6] or built upon triangulations whose triangles stretch in the direction of the flow and whose edges are transverse to it [1]. In both cases, the results are developed for single-valued flows. It appears that theoretical study of the relationship of the transition graph and the underlying PC flow may require new technical tools. In particular, it would be interesting to investigate if the Morse sets (or perhaps even pseudo-Morse sets) forming an arbitrarily close upper bound of all recurrent dynamics of the PC flow can be obtained from a fine enough transition graph.

Acknowledgments

The authors wish to thank Robert S. Laramee and Guoning Chen for data sets used in Section 7. We are also grateful to Guoning Chen and Qingqing Deng for help with Figs. 15 and 16. Eugene Zhang was partially supported by US National Science Foundation (NSF) IIS-0546881 and NSF CCF-0830808.

REFERENCES

[1] E. Boczko, W.D. Kalies, and K. Mischaikow, "Polygonal Approximation of Flows," *Topology and Its Applications,* vol. 154, no. 13, pp. 2501-2520, 2007.

[2] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci, "A Multi-Resolution Data Structure for 2-Dimensional Morse Functions," *Proc. IEEE 14th Visualization (VIS '03),* pp. 139-146, 2003.

[3] J. Chai, Y. Zhao, W. Guo, and Z. Tang, "A Texture Method for Visualization of Electromagnetic Vector Field," *Proc. Sixth Int'l Conf. Electrical Machines and Systems (ICEMS),* vol. 2, pp. 805-808, 2003.

[4] G. Chen, Q. Deng, A. Szymczak, R.S. Laramee, and E. Zhang, "Morse Set Classification and Hierarchical Refinement Using Conley Index," *IEEE Trans. Visualization and Computer Graphics,* vol. 18, no. 5, pp. 767-782, May 2012.

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

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

[7] C. Conley, *Isolated Invariant Sets and Morse Index.* American Mathematical Soc., 1978.

[8] E. Early, "On the Euler Characteristic," *The MIT Undergraduate J. Math.,* vol. 1, pp. 37-48, 1999.

[9] H. Edelsbrunner, J. Harer, and A. Zomorodian, "Hierarchical Morse Complexes for Piecewise Linear 2-Manifolds," *Proc. 17th Ann. Symp. Computational Geometry (SCG '01),* pp. 70-79, 2001.

[10] R. Forman, "Combinatorial Vector Fields and Dynamical Systems," *Mathematische Zeitschrift,* vol. 228, pp. 629-681, 1998.

[11] L. Gorniewicz, *Topological Fixed Point Theory of Multivalued Mappings: Volume 4 of Topological Fixed Point Theory and Its Applications,* second ed. Springer, 2006.

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

[13] J.L. Helman and L. Hesselink, "Visualizing Vector Field Topology in Fluid Flows," *IEEE Computer Graphics and Applications,* vol. 11, no. 3, pp. 36-46, May 1991.

[14] T. Kaczynski and M. Mrozek, "Conley Index for Discrete Multivalued Dynamical Systems," *Topology and Its Applications,* vol. 65, pp. 83-96, 1997.

[15] A. Katok and B. Hasselblatt, *Introduction to the Modern Theory of Dynamical Systems: Encyclopedia of Mathematics and Its Applications.* Cambridge Univ. Press, 1995.

[16] R.S. Laramee, C. Garth, H. Doleisch, J. Schneider, H. Hauser, and H. Hagen, "Visual Analysis and Exploration of Fluid Flow in a Cooling Jacket," *Proc. IEEE Visualization,* pp. 623-630, 2005.

[17] R.S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F.H. Post, and D. Weiskopf, "The State of the Art in Flow Visualization: Dense and Texture-Based Techniques," *Computer Graphics Forum,* vol. 23, no. 2, pp. 203-221, 2004.

[18] R.S. Laramee, H. Hauser, L. Zhao, and F.H. Post, "Topology-Based Flow Visualization, the State of the Art," *Proc. Topology-Based Methods in Visualization (TopoInVis '05),* pp. 1-19, 2007.

[19] R.S. Laramee, J.J. van Wijk, B. Jobard, and H. Hauser, "ISA and IBFVS: Image Space-Based Visualization of Flow on Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 10, no. 6, pp. 637-648, Nov./Dec. 2004.

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

[21] P. Lindstrom and V. Pascucci, "Visualization of Large Terrains Made Easy," *Proc. IEEE Visualization,* pp. 363-371, 2001.

[22] E.N. Lorenz, "Deterministic Non-Periodic Flow," *J. Atmospheric Science,* vol. 20, pp. 130-141, 1963.

[23] R.P. McGehee and T. Wiandt, "Conley Decomposition for Closed Relations," *J. Difference Equations and Applications,* vol. 12, no. 1, pp. 1-47, 2006.

[24] T. McLoughlin, R.S. Laramee, R. Peikert, F.H. Post, and M. Chen, "Over Two Decades of Integration-Based, Geometric Flow Visualization," *Proc. Eurographics,* pp. 73-92, 2009.

[25] R. Mehran, B.E. Moore, and M. Shah, "A Streakline Representation of Flow in Crowded Scenes," *Proc. 11th European Conf. Computer Vision (ECCV '10),* pp. 439-452, 2010.

[26] K. Mischaikow and M. Mrozek, "Chaos in the Lorenz Equations: A Computer-Assisted Proof," *Bull. Am. Math. Soc.,* vol. 32, pp. 66-72, 1995.

[27] K. Mischaikow, M. Mrozek, and A. Szymczak, "Chaos in the Lorenz Equations: A Computer Assisted Proof. Part III: The Classical Parameter Values," *J. Differential Equations,* vol. 169, no. 1, pp. 17-56, 2001.

[28] M. Mrozek, "Index Pairs and the Fixed Point Index for Semidynamical Systems with Discrete Time," *Fundamental Mathematicae,* vol. 133, pp. 177-192, 1989.

[29] M. Mrozek, "A Cohomological Index of Conley Type for Multivalued Admissible Flows," *J. Differential Equations,* vol. 84, pp. 15-51, 1990.

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

[31] R. Panton, *Incompressible Flow.* John Wiley & Sons, 1984.

[32] J. Reininghaus, C. Lowen, and I. Hotz, "Fast Combinatorial Vector Field Topology," *IEEE Trans. Visualization and Computer Graphics,* vol. pp, no. 99, p. 1, 2010.

[33] E.H. Spanier, *Algebraic Topology.* Springer, 1966.

[34] A. Szymczak, "Stable Morse Decompositions for Piecewise Constant Vector Fields on Surfaces," *Computer Graphics Forum,* to appear, 2011.

[35] R. Tarjan, "Depth-First Search and Linear Graph Algorithms," *SIAM J. Computing,* vol. 1, no. 2, pp. 146-160, 1972.

[36] H. Theisel and T. Weinkauf, "Grid-Independent Detection of Closed Stream Lines in 2D Vector Fields," *Proc. Conf. Vision, Modeling and Visualization (VMV '04),* pp. 421-428, 2004.

[37] X. Tricoche, C. Garth, and G. Scheuermann, "Fast and Robust Extraction of Separation Line Features," *Proc. Scientific Visualization: The Visual Extraction of Knowledge from Data,* pp. 249-263, 2006.

[38] X. Tricoche, G. Scheuermann, and H. Hagen, "Higher Order Singularities in Piecewise Linear Vector Fields," *Proc. IMA Conf. Math. Surfaces,* pp. 99-113, 2000.

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

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

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

Andrzej Szymczak received the MS degree in mathematics from the University of Gdańsk, Poland in 1994 and the PhD degree in mathematics and the MS degree in computer science from the Georgia Institute of Technology in 1999. Currently, he is an assistant professor in the Department of Mathematical and Computer Sciences at the Colorado School of Mines. His research interests include scientific visualization, computational topology, medical image analysis, and computer graphics. He is a member of the IEEE.

Eugene Zhang received the PhD degree in computer science in 2004 from Georgia Institute of Technology. Currently, he is an associate professor at Oregon State University, where he is a member of the School of Electrical Engineering and Computer Science. His research interests include computer graphics, scientific visualization, and geometric modeling. He received a National Science Foundation (NSF) CAREER award in 2006. He is a senior member of the IEEE and the ACM.