Hierarchy of Stable Morse Decompositions

Andrzej Szymczak, *Member*, *IEEE*

Abstract—We introduce an algorithm for construction of the *Morse hierarchy*, i.e., a hierarchy of Morse decompositions of a piecewise constant vector field on a surface driven by stability of the Morse sets with respect to perturbation of the vector field. Our approach builds upon earlier work on stable Morse decompositions, which can be used to obtain Morse sets of user-prescribed stability. More stable Morse decompositions are coarser, i.e., they consist of larger Morse sets. In this work, we develop an algorithm for tracking the growth of Morse sets and topological events (mergers) that they undergo as their stability is gradually increased. The resulting Morse hierarchy can be explored interactively. We provide examples demonstrating that it can provide a useful coarse overview of the vector field topology.

Index Terms—Morse decomposition, persistence, vector field

1 Introduction

Virtually no interesting data set can be assumed to be known precisely because of noise, measurement error, or numerical error. Features extracted from an imprecisely known data set need to be taken with a grain of salt. Some features may be artifacts of the error present in the data. Other features may be missing because of the error. Error may change characteristics of features and hence may cause misclassification. These problems could hamper one's ability to produce insightful visualizations and hence understand the structure of the data.

These issues have motivated numerous research efforts, targeting different types of data sets and different kinds of features. An approach that is frequently used in recent years is to filter features based on their stability with respect to perturbation of the data. Imprecisely known data sets can be thought of as small perturbations of the exact data. Conversely, the unknown precise data can be considered a small perturbation of the known imprecise data. Therefore, features observed in the imprecise data that are more stable under perturbation are more likely to be correct for the exact data. Hence, measures of stability of features under perturbation can be used to *filter out* weak features that are less likely to be valid for the exact data. Stability measures are also used to *reduce the complexity* of the set of features. More stable features are typically less numerous and are often easier to understand. They can be used to provide the user with a coarse overview of the structure of the data.

Stability measures of topological features have proven particularly useful in scalar field analysis, where they have been used to remove the topological noise or reduce the complexity of the feature set (for example, see [4], [10], [15]). The theoretical foundations behind the stability measures have been provided by [11]. In the scalar field domain, the features of interest are typically critical points and cells of the Morse complex (which can be interpreted as sets of trajectories of the gradient vector field connecting pairs of critical points).

In this paper, we propose a method for building a *Morse hierarchy*, i.e., a hierarchy of stable Morse decompositions of varying stability under perturbation. Our algorithm is developed for general (not necessarily gradient) piecewise constant (PC) vector fields. The Morse hierarchy is based on the convex-valued piecewise constant (CVPC) vector field framework of Szymczak [27]. Stabler Morse sets obtained using the technique of Szymczak [27] are larger than the less stable ones. In other words, Morse sets grow in size as the stability parameter is increased. Our algorithm essentially logs their growth and topological events (mergers) that they undergo throughout the process. We also propose a simple and compact representation of the Morse hierarchy. In particular, it allows for interactive exploration of Morse decompositions of varying stability. We apply our procedure to a number of well-known fluid simulation data sets and provide examples demonstrating that coarse Morse decompositions can provide useful high-level summary of the flow structure.

The paper is organized as follows: First, we discuss the prior work on vector field topology (Section 2). Then, we review the necessary background on piecewise constant vector fields and convex-valued piecewise constant vector fields in Section 3. Note that this review is by no means complete: we still assume that the reader is familiar with the work [27], [29]. Section 4 describes our sweep algorithm for computing the Morse hierarchy. Section 5 discusses the visualization of the Morse hierarchy. Experimental results are discussed in Section 6. Finally, a discussion of possible future work is included in Section 7.

2 Prior Work

Vector field visualization has received a significant amount of attention in recent years. It is beyond the scope of this paper to discuss all vector field visualization techniques. We refer the reader to survey papers [20], [21], [24], [25] and focus on discussion of vector field topology below.

Most of the work on vector field topology focuses on finding special kinds of trajectories such as stationary points, periodic orbits, and separatrices. Examples of such techniques developed for continuous vector fields include

• *The author is with the Department of Electrical Engineering and Computer Science, Colorado School of Mines, Golden, Colorado 80401-1887. E-mail: aszymcza@mines.edu.*

Manuscript received 9 Oct. 2011; revised 6 Mar. 2012; accepted 4 June 2012; published online 21 June 2012.
Recommended for acceptance by D. Weiskopf.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-10-0246.
Digital Object Identifier no. 10.1109/TVCG.2012.147.

[17], [31], [33]. A problem that often arises in these approaches is that trajectories are numerically approximated rather than followed exactly. This means that correctness or consistency of the output is generally not guaranteed. These issues have only recently started to be addressed, for example, using a combination of algebraic and topological techniques [13].

The combinatorial vector field framework of Reininghaus et al. [26] builds upon the mathematical results [14] to provide a method for computing a representation of vector field topology *without* numerical integration. It also supports a natural way to simplify the topology by allowing one to restrict the number of critical points. However, the problem of this approach is low accuracy: trajectories of a combinatorial vector field are strongly restricted by the directions of mesh edges which makes them relatively poor approximation of the true trajectories of the underlying continuous vector field.

The edge map representation [2], [3] provides an alternative to the standard numerical integration that avoids the associated consistency issues. It supports a refinement scheme that can be used to approximate the trajectories of a piecewise linear vector field with arbitrary accuracy. It also allows one to follow "fuzzy" trajectories, i.e., construct sets that can be reached from a given point by following trajectories of vector fields within an error bound from the input one. Building upon this idea, fuzzy variants of stable and unstable sets of stationary points and periodic trajectories (or, more generally, Morse sets) can be constructed. Intersecting stable and unstable sets can potentially be used to construct stable Morse sets. However, this approach does not by itself provide a complete Morse decomposition or a hierarchy thereof.

In [6], a 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 [18]. In [7], Morse decomposition is proposed as a coarser, but also more stable alternative to the standard vector field topology. The same paper introduces an algorithm for computing a Morse decomposition and a Morse Connection Graph (MCG), that encodes connections between Morse sets. An extension of this approach that leads to a hierarchy of Morse sets and a simple scheme for Morse set classification is introduced in [5]. However, let us stress that the hierarchy of Morse decompositions constructed in [5] is not based on a Morse set stability measure, but is driven by the refinement algorithm and its arbitrarily chosen parameters. Also, all of the techniques discussed in this paragraph are still based on numerical integration and therefore the output, despite being more robust, is still not guaranteed to be topologically consistent.

An efficient and robust algorithm for computing Morse decompositions for piecewise constant vector fields is described in [29]. In [27], an extension of this algorithm that supports construction of Morse decompositions of a user-specified stability under perturbation is introduced. However, the algorithm of Szymczak [27] is relatively slow and leads to very large geometric models of Morse sets. Therefore, exploration of Morse decompositions of varying stability under perturbation based on the algorithm of Szymczak [27] alone is arduous and time consuming. The ability to explore stable Morse decompositions in an interactive fashion is important since the value of the stability parameter that leads to an insightful Morse decomposition usually has to be found by trial and error. This is the main motivation for this paper.

Our main contribution is a sweep algorithm for computing a Morse hierarchy, i.e., hierarchy of stable Morse decompositions. Our algorithm tracks the changes to the Morse decomposition as the stability parameter, representing the maximum perturbation of the vector field for which the Morse decomposition is guaranteed to be valid, is gradually increased. The Morse sets gradually increase and undergo topological events (mergers). Our algorithm produces the *Morse merge tree*, a tree representation of all mergers, which can be used to estimate stability of any Morse set in the hierarchy. The tree can also be used to explore the topological events or all possible structures of stable Morse decompositions and can therefore simplify Morse hierarchy exploration. We also propose a compact geometric representation of the hierarchy which can be used for fast extraction of a geometric model of the Morse decomposition of any stability from the hierarchy.

Cancelation or combination of topological features in nongradient like vector fields has previously been discussed in [6], [32], [34]. These operations are similar to Morse set mergers used in this paper. However, we use a framework entirely based on Morse decompositions and therefore we are not limited to simple features such as stationary points or periodic orbits.

3 Background

In this section, we briefly review the results related to piecewise constant and convex valued piecewise constant vector fields that this work builds upon. For a complete, in-depth discussion, the reader is referred to [27], [29].

3.1 Morse Decompositions

Let σ be a trajectory and A and B be compact sets. We say that σ *links* A *to* B if $\lim_{t\to-\infty} \mathrm{dist}(\sigma(t), A) = 0$ and $\lim_{t\to\infty} \mathrm{dist}(\sigma(t), B) = 0$. By a Morse decomposition of a vector field, we mean a finite family of mutually disjoint compact sets $\mathcal{M}$ satisfying the following conditions:

1. Any trajectory passing through a point outside the union of all sets in $\mathcal{M}$ links two different sets in $\mathcal{M}$.
2. The *linkage graph* of $\mathcal{M}$ is acyclic. The nodes of the linkage graph are sets in $\mathcal{M}$. An arc $A \to B$ exists in the graph if and only if there is a trajectory that links A to B.

If $\mathcal{M}$ is a Morse decomposition, then sets in $\mathcal{M}$ are called *Morse sets*. Any stationary point or periodic orbit (and, more generally, any chain recurrent point [8]) is contained in one of the Morse sets for any Morse decomposition. On the complement of the union of Morse sets, the vector field is gradient-like, i.e., free of circulation [8]. An example of a Morse decomposition and its linkage graph is shown in Fig. 1. Note that the same vector field generally has more than one Morse decomposition. For the example shown in Fig. 1, one can obtain a coarser Morse decompositon by replacing $A1$, $A2$, and S with a single Morse set containing them and trajectories that connect S to $A1$ and $A2$.

In [27], [29], fixed point index and stability information (i.e., categorization as attracting, repelling or neither attracting nor repelling, Fig. 1), is used to classify Morse sets. The fixed point index of a Morse set can be defined as follows: Take a small neighborhood of the Morse set in question.

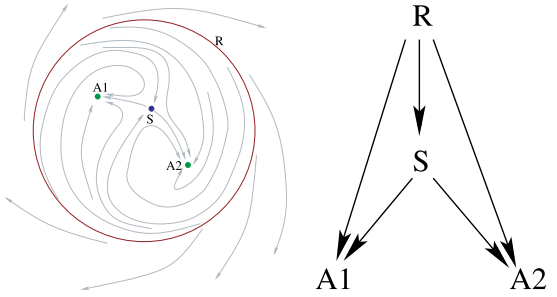

Fig. 1. An example of a Morse decomposition and its linkage graph. $A1$ and $A2$ are attracting (trajectories followed forward from a point near Ai converge to Ai) and R is repelling (trajectories followed backward from a point near R converge to R). S is neither attracting nor repelling.

Apply a small perturbation to the vector field to make the number of stationary points in that neighborhood finite. In order to obtain the index of the Morse set, add the indices of the stationary points in the neighborhood. The index of a stationary point is equal to $1+\frac{e-h}{2}$, where e and h is its number of hyperbolic and elliptic sectors (respectively).

A Morse set is referred to as *of type* $(i,-)$, $(i,+)$, *or* $(i,0)$ if it is of index i and is attracting, repelling, or neither attracting nor repelling (respectively). This scheme allows one to distinguish the basic features, i.e., sinks, sources, saddles (which are of types $(1,-)$, $(1,+)$, and $(-1,0)$) as well as attracting or repelling periodic orbits (of types $(0,-)$ and $(0,+)$—note their index is zero since they do not contain any stationary points). A Morse set of type $(0,0)$ is called *trivial*. Trivial Morse sets contain features that cancel each other, or contain no features at all.

3.2 CVPC Vector Field Analysis

A piecewise constant vector field [29] is a function that assigns a vector $f(\Delta)$, parallel to Δ, to any triangle Δ of the mesh. By a CVPC vector field on a triangulated manifold surface M, we mean a function F that assigns a convex set of vectors parallel to the Δ's plane for each triangle Δ. A PC vector field can be viewed as a CVPC vector field for which $F(\Delta)$ consists of exactly one vector.

For each mesh edge e, we define the set of vectors $F(e)$ parallel to e as follows: Let Δ_0 and Δ_1 be triangles incident to e. Rotate Δ_0 (together with the set of vectors assigned to it by F) around e so that it becomes coplanar with Δ_1. $F(e)$ is defined as the set of all vectors parallel to e contained in the convex hull of $F(\Delta_0)\cup F(\Delta_1)$. Note that $F(e)$ can possibly be empty if $F(\Delta_0)$ and $F(\Delta_1)$, after the rotation, point to the same side of e.

By a *stationary triangle*, we mean a triangle Δ such that $F(\Delta)$ contains the zero vector. Similarly, an edge e is stationary if and only if $F(e)$ contains the zero vector. Any mesh vertex v that belongs to a stationary edge or triangle is also referred to as stationary. A vertex that does not belong to a stationary edge or triangle is stationary if it has an elliptic sector, its number of unstable parabolic sectors is other than one, or its number of stable parabolic sectors is other than one [27]. Below, we review the definitions and algorithm for sector analysis and discuss monotonicity properties of sectors that play an important role in our algorithm. Let us stress that sector analysis is performed *only* for vertices that do not have an incident stationary edge or triangle.

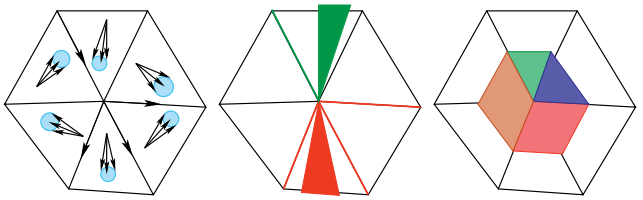

Fig. 2. Left: a CVPC vector field F near a vertex v. Blue discs indicate the convex sets of vectors assigned to each triangle. F assigns nonempty sets of vectors to 4 of the edges incident to v. Black arrows pointing along edges indicate the direction of vectors in these sets. Middle: stable (green) and unstable (red) bundles. Right: sectors (red-unstable parabolic, green-stable parabolic, blue hyperbolic, brown elliptic). Hyperbolic and elliptic sectors can be determined based on the vector field directions, as seen from v. For example, the vectors in the brown elliptic sector point clockwise around v, from the unstable bundle toward the stable bundle bounding that sector. In the hyperbolic sector, the vectors point from the stable bundle toward the unstable bundle.

For a triangle Δ incident upon a vertex v that does not have an incident stationary edge or triangle, by O_Δ we mean the set of all positive multiples of vectors that start at v and end at a point of Δ other than v. Similarly, for an edge e by O_e, we mean the set of all positive multiples of vectors starting at v and ending at a point on e other than v. By $\bar{F}(\Delta)$ and $\bar{F}(e)$, we mean the set of all positive multiples of vectors in $F(\Delta)$ and $F(e)$ (respectively). The nonempty sets of the form $O_\Delta\cap\bar{F}(\Delta)$ or $O_e\cap\bar{F}(e)$ are called *unstable bundles* at v. Note that in some cases, an unstable bundle associated with an edge incident to v may be contained in an unstable bundle of an incident triangle. Analogously, nonempty sets of the form $O_\Delta\cap-\bar{F}(\Delta)$ or $O_e\cap-\bar{F}(e)$ are called *stable bundles* at v.

In [27], sector analysis of v is performed by scanning all bundles (stable and unstable) in the counterclockwise order around v. A consecutive sequence of stable bundles defines a stable parabolic sector of v. A consecutive sequence of unstable bundles defines an unstable parabolic sector of v. Hyperbolic and elliptic sectors are bounded by a stable bundle on one side and an unstable one on the other. In a hyperbolic sector, the trajectories move from the stable bundle to the unstable bundle (as seen from v). In an elliptic sector, they move from the unstable bundle to the stable bundle. An example showing sectors of different types is shown in Fig. 2.

Notice that the stable and unstable bundles and parabolic sectors are monotone increasing with respect to the vector field. More precisely, let F_1 and F_2 be any CVPC vector fields with no stationary edges or triangles incident to v and such that $F_1(\Delta)\subset F_2(\Delta)$ for any triangle Δ. Then, these bundles and sectors are larger for F_2 than for F_1. Hence, hyperbolic and elliptic sectors are monotone decreasing: any such sector for F_2 is contained in a sector of the same type for F_1. Conversely, each hyperbolic or elliptic sector of F_1 contains a sector of the same type of F_2 (otherwise, intersecting unstable and stable bundles of F_2 would exist, which would mean that v has an incident stationary edge or triangle for F_2). One can conclude that a vertex v with no incident stationary triangle or edge (in both F_1 and F_2) is stationary for F_1 if and only if it is stationary for F_2. This observation will help us keep track of stationary vertices in the sweep algorithm described later on.

Topological features of a CVPC vector field are determined from its *supertransition graph*. Nodes of that graph are mesh vertices and *edge pieces*, i.e., line segments obtained

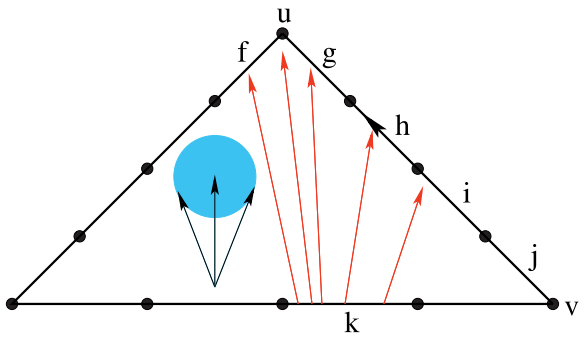

Fig. 3. A mesh triangle Δ and its associated set of vectors $F(\Delta)$ (blue disk). We assume that 1) all vertices of Δ are nonstationary, 2) the set of vectors $F(e)$ assigned to the edge $e = u\bar{v}$ contains a vector pointing from v to u but no vector pointing in the opposite direction, and 3) the sets of vectors assigned to the other two edges are empty. The red lines represent feasible segments moving through Δ, since each extends in the directions of a vector in $F(\Delta)$. N-set pairs they connect define arcs in the graph, $k \to f$, $k \to g$, $k \to h$, $k \to i$, and $k \to u$. There is also a feasible segment following the edge e from v to u. Hence, the graph contains the path $v \to j \to i \to h \to g \to u$.

from mesh edges by means of subdivision. The sets corresponding to nodes of the graph are called *node sets* (n-sets for brevity). Thus, an n-set is either a mesh vertex or an edge piece. Arcs of the graph connect n-sets that are connected by a *feasible segment* defined on a time interval of positive length. Roughly speaking, a feasible segment is a constant velocity path that is contained in a single mesh triangle and follows a direction given by F. More precisely, feasible segments can

- Move through the triangle Δ with constant velocity chosen from $F(\Delta)$.
- Move along an edge e with constant velocity in $F(e)$. Such feasible segments exist only if $F(e) \neq \emptyset$.
- Stay at a stationary vertex for some amount of time.

Examples of feasible segments and the respective arcs of a supertransition graph are discussed in Fig. 3. As shown in [27], strongly connected components of a supertransition graph for F define Morse sets that are valid for any *feasible PC vector field*, i.e., any PC vector field f with $f(\Delta) \in F(\Delta)$. Consistently with [29], by the *transition graph* of a PC vector field f we mean the supertransition graph of the corresponding CVPC vector field F_f, defined by $F_f(\Delta) = f(\Delta)$.

4 Algorithm

The input to our algorithm is a piecewise constant vector field f on a triangulated surface S and a nonnegative integer l that controls the subdivision depth for the transition graph. First (Section 4.1), we build the transition graph $\mathcal{G}$ for f based on uniform subdivision of every mesh edge into 2^l equal length edge pieces. We compute the strongly connected components of the resulting graph. In what follows, $\mathcal{C}$ denotes the family of all strongly connected components of this initial transition graph $\mathcal{G}$ that define nontrivial Morse sets. Note that the node sets in $\mathcal{C}$ stay fixed throughout the algorithm. We chose to ignore trivial Morse sets (i.e., not to include them in $\mathcal{C}$) because such sets may contain no topological features, or, if they do, these features cancel each other. The underlying theory does not provide a reliable way to decide which of the above two alternatives is true for a particular trivial Morse set. Ignoring trivial Morse sets is akin to applying a small perturbation to the vector field to cancel their features, if they exist, in the spirit of simulation of simplicity technique [12].

We compute the *discrete stable and unstable sets* of all sets in $\mathcal{C}$. By the discrete stable set of a set A of nodes of a (super-) transition graph $\mathcal{G}$, we mean the union of all paths in $\mathcal{G}^T$, i.e., $\mathcal{G}$ with arcs reversed, starting at a node in A. The discrete unstable set of A is the union of all paths in $\mathcal{G}$ starting at a node in A. The algorithm inserts new arcs into $\mathcal{G}$ and maintains the discrete stable and unstable sets of all sets in $\mathcal{C}$, defined as above.

In [27], Morse decompositions that are stable under any perturbation of f of magnitude no more than E are computed from the supertransition graph of the convex-valued piecewise constant vector field $F_E(\Delta) := B_\Delta(f(\Delta), E)$, where by $B_\Delta(\vec{v}, r)$ we denote the disk of vectors parallel to Δ's plane, centered at $\vec{v}$ and of radius r. The idea behind the main loop of the algorithm (Section 4.3) is to maintain the supertransition graph and the discrete stable and unstable set information (for node sets in $\mathcal{C}$) for F_E as the stability parameter E is gradually increased. As E is increased, arcs need to be inserted into the supertransition graph. As a result, the discrete stable and unstable sets grow in size. The strongly connected components of $\mathcal{G}$ corresponding to the initial ones (sets in $\mathcal{C}$) also grow in size. These components are intersections of the discrete stable and unstable set of the same set $C \in \mathcal{C}$. They also undergo topological events (merge into larger strongly connected components). We record these events in a tree data structure. Overall, the algorithm can be viewed as a sweep like one, where the sweep is controlled by the increasing stability parameter E.

Note that by the monotonicity property of sectors (Section 3.2), it is enough to determine stationary vertices at startup, for the PC vector field $F_0 = f$. These vertices stay stationary throughout the sweep. New stationary vertices appear only as a result of triangles or edges becoming stationary because of the increase in E.

The supertransition graphs used to obtain stable Morse decompositions can get quite large, especially when computing Morse decompositions of high stability [27]. Therefore, in this paper, we use a compact representation of the supertransition graph to make the memory requirements smaller. Roughly, the idea is to represent "fans" of arcs using just the extreme (i.e., clockwise-most and counterclockwise-most) vectors in a fan. Details are given in Section 4.2. Note that this representation also simplifies event handling, particularly for the expansion events (Section 4.3.3), which move the extreme vectors. The same compact representation is used to represent the graph $\mathcal{G}^T$, which is identical to the supertransition graph of the CVPC vector field $-F_E$ opposite to F_E, defined by $-F_E(\Delta) := B_\Delta(-f(\Delta), E)$.

4.1 Initialization

We use the algorithm of Szymczak and Zhang [29] to construct the initial graph $\mathcal{G}$. First, we construct a coarse graph whose nodes correspond to the mesh edges and vertices. Then, all nodes corresponding to edge pieces are subdivided, each into two edge pieces of equal length. This subdivision step is repeated l times. The resulting transition graph $\mathcal{G}$ and its transpose $\mathcal{G}^T$ are then converted to the compact representation described in Section 4.2.

Strongly connected components of $\mathcal{G}$ are computed using the Tarjan's algorithm [9], [30]. By $\mathcal{C}$, we denote the family

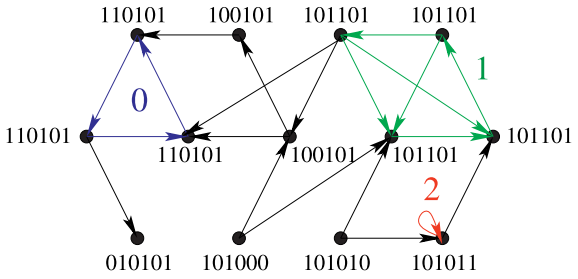

Fig. 4. An example of a graph with three strongly connected components, containing the blue (component 0), green (component 1), and red (component 2) edges. Next to each node, we display its associated bitmap (with the zeroth bit coming first). For example, the bit map of the node on the bottom left is 010101, since there is a path connecting any strongly connected component to that node (which means that it is in the discrete unstable set of any component), but there is no path connecting that node to any of the components (i.e., it is not in the discrete stable set of any component).

of all strongly connected components of $\mathcal{G}$ that define nontrivial Morse sets at this stage of the algorithm. For any component $C \in \mathcal{C}$, we compute its discrete stable and unstable sets by starting depth first search (of $\mathcal{G}^T$ and $\mathcal{G}$, respectively) from any node in C. All nodes visited during search are in the stable (respectively, unstable) set.

For each node, we keep its stable and unstable set membership information in a bit map. More precisely, for the ith strongly connected component $C_i \in \mathcal{C}$ and a node n, the $2i$th and $(2i+1)$th bits of the bitmap $\mathcal{B}_n$ associated with n are set to 1 if and only if n is in the discrete stable (respectively, unstable) set of C_i. At this stage of the algorithm, a node n is in C_i if and only if *both* the $2i$th and $(2i+1)$th bit of $\mathcal{B}_n$ are set. An example illustrating the bitmaps is shown in Fig. 4.

4.2 Compact Supertransition Graph

In what follows, feasible segments moving through the interior of a mesh triangle Δ are called *feasible segments in* Δ. Arcs of a supertransition graph representing such feasible segments are be called *type T arcs in* Δ. Let Δ be a mesh triangle and n be an n-set (edge piece or vertex) contained in its boundary $\partial\Delta$. If type T arcs in Δ out of n exist, they form a fan as shown in Figs. 5 and 6.

In our transition graph representation, for a node n we keep only the endpoints of the extreme, i.e., counterclockwise-most and clockwise-most, arcs for each fan of type T arcs in Δ out of n. In what follows, these endpoints will be called the left and the right (n, Δ)-endpoints (respectively). Arcs of types other than T (i.e., arcs representing stationary vertices or trajectories moving along mesh edges) are stored in an explicit manner. Information described above is sufficient to determine all arcs out of any node n. In particular, all endpoints of type T arcs out of n in a triangle Δ containing n in its boundary can be obtained by walking around the boundary of Δ counterclockwise, starting from the right (n, Δ)-endpoint, until the left (n, Δ)-endpoint is reached.

Along with $\mathcal{G}$, our implementation maintains the graph $\mathcal{G}^T$ represented in the same way. $\mathcal{G}^T$ is identical to the supertransition graph for the opposite CVPC vector field $-F_E$. The initial graph $\mathcal{G}'$ is the same as the transition graph for the PC vector field $-f$. For any fan of type T arcs out of a node n representing feasible segments of $-F_E$ in a triangle Δ, the compact representation of $\mathcal{G}^T$ keeps the counterclockwise-most and clockwise-most arcs, which we refer to as the *left and right* (n, Δ)*-endpoints in* $\mathcal{G}^T$.

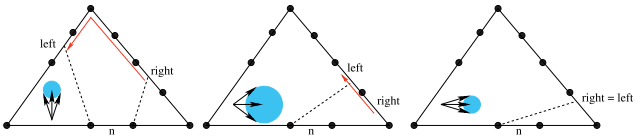

Fig. 5. An edge piece n on the boundary of a triangle Δ. Type T arcs out of n represent feasible segments traversing the interior of Δ. Such segments have constant velocity in $F_E(\Delta)$ (blue disc). In the figure on the left, the points in Δ that can be reached by such segments are located between the two dashed lines. For the figure in the center and on the right, they are between the dashed line and the bottom edge of Δ. The figures show the left and right (n, Δ)-endpoints in each case. In the case shown on the right, the left and right (n, Δ)-endpoints are the same. The general rule is to define the left and right endpoints so that all endpoints of type T arcs representing feasible segments in Δ can be obtained by following the boundary of Δ counterclockwise from the right to the left endpoint (as shown by the red arrows).

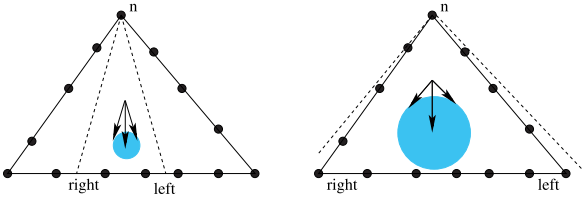

Fig. 6. Left and right (n, Δ)-endpoints for a vertex node n in $\partial\Delta$. All points of the triangle that can be reached from n by a feasible segment traversing the interior of Δ are located between the two dashed lines, extending along the clockwise- and counterclockwise-most vector in $F(\Delta)$. Endpoints of type T arcs out of n representing such segments are all edge pieces on the edge opposite to n that intersect the area between the dashed lines.

4.3 The Sweep

The main loop of the algorithm is driven by a priority queue that holds *events*, representing changes to $\mathcal{G}$ resulting from gradual increase in the stability parameter E. Until the priority queue is empty, we extract the lowest cost event from the queue. The event results in change to the supertransition graph $\mathcal{G}$ (in most cases, addition of one or more arcs). Any new arc could increase the discrete stable and unstable sets. This means that some bits in the bitmaps described in Section 4.1 may need to be set. The process of updating the bitmaps as a result of graph changes is described in Section 4.3.1. Topological changes to the Morse sets are discussed in Section 4.3.2. Types of events and their corresponding actions are discussed in the subsequent Sections (4.3.3, 4.3.4, 4.3.5, and 4.3.6). In a section devoted to an event type, we first describe the event. Then, we discuss its cost and how events of that type are generated at startup. Finally, we describe the action (status change) resulting from the event.

The actions resulting from events are designed so that the bitmaps represent the discrete stable and unstable sets of strongly connected components of the initial transition graph (i.e., elements of $\mathcal{C}$). More precisely, the discrete stable set of the ith set in $\mathcal{C}$ consists of all nodes n whose bitmap has the $2i$th bit set. Similarly, the ith unstable set of the ith set in $\mathcal{C}$ consists of all nodes whose bitmap's $(2i+1)$th bit is

```
Procedure PropagateFrom ( node n, bitmap B, bitmap M, graph F )
  NewBitmap := Bitmap(n) OR (B AND M)
  if NewBitmap≠Bitmap(n) then
    Bitmap(n):=NewBitmap
    for each arc n → m out of n in F do
      PropagateFrom(m,B,M,F)
```

Fig. 7. Pseudocode of the key component of the bitmap update algorithm, based on the depth-first search of $\mathcal{G}$. Addition of the arc $a \to b$ to $\mathcal{G}$ results in the calls PropagateFrom(b,$\mathcal{B}_a$,U,$\mathcal{G}$) and PropagateFrom(a,$\mathcal{B}_b$,S,$\mathcal{G}^T$). U is the unstable mask, i.e., bitmap with all and only odd bits set. S is the stable mask, with only even bits set.

set (cf Section 4.1). In addition, the compact representation of $\mathcal{G}^T$ is kept consistent with $\mathcal{G}$.

4.3.1 Bitmap Update

Whenever an arc is added to $\mathcal{G}$, bitmaps of nodes are updated. The goal of the update is to set some of the bits in the bitmaps to maintain the discrete stable and unstable sets of strongly connected components of the initial transition graph (i.e., sets in $\mathcal{C}$).

Let $a \to b$ be the new arc added to $\mathcal{G}$. If a is in a discrete unstable set A, then any node that can be reached from b by following a path in $\mathcal{G}$ needs to be included in A as well. Similarly, if b is in a stable set B then any node that can be reached from a by a path in $\mathcal{G}^T$ needs to be included in B.

This naturally leads to an algorithm shown in Fig. 7. The idea is to search $\mathcal{G}$ starting at b and set all odd bits of the visited nodes that are set in $\mathcal{B}_a$, and then search $\mathcal{G}^T$ starting at a and set all even bits of the visited nodes that are set in $\mathcal{B}_b$. Each of the searches can be terminated at nodes whose bitmaps are not changed.

4.3.2 Trackable Components and Morse Sets

Recall that our goal is to track the growth and mergers of strongly connected components as new arcs are inserted into the graph $\mathcal{G}$. For reasons discussed in the beginning of Section 4, we focus on strongly connected components that define nontrivial Morse sets. Any such component must be *trackable*, i.e., must contain a set belonging to $\mathcal{C}$. Intuitively, the initial strongly connected components that define nontrivial Morse sets (i.e., sets in $\mathcal{C}$) grow and merge with each other as $\mathcal{G}$ gains new arcs. Trackable components are strongly connected components that result from this process. Components that are not trackable are disjoint with the union of sets in $\mathcal{C}$ and therefore Morse sets defined by them are trivial, since they cannot contain any complete trajectory of f. Let us note that trackable components can be trivial. Such components result from mergers representing Morse set cancelations.

Let M_i be the trackable component containing the ith set in $\mathcal{C}$. It is easy to see that a node n is in M_i if and only if bits $2i$ and $2i+1$ of $\mathcal{B}_n$ are set. Two trackable components M_i and M_j with $i \neq j$ merge at the moment a node n belonging to both M_i and M_j appears. Such nodes are detected when bitmaps are updated as described in Section 4.3.1.

Trackable components define Morse sets of stability at least E. We record trackable component mergers in the *Morse merge tree*, whose leaves correspond to sets in $\mathcal{C}$ (or to the initial Morse sets). Whenever trackable components M_i and M_j merge, we create a new node of the tree (representing the component resulting from the merger) and make the nodes corresponding to M_i and M_j its children. The *height* of the new node of the tree is the value of the stability parameter E at which the merger took place. The height of the leaf nodes is zero.

Recall that determining the Morse set type information boils down to fixed point index computation and categorization as attracting, repelling, or neither attracting nor repelling (Section 3.1). The fixed point index is the sum of indices of all stationary vertices in the trackable component [27], [29]. When a trackable component M_i is merged with M_j, the index of the resulting one is the sum of indices of M_i and M_j. A trackable component M is attracting (repelling) if and only if the number of nodes in M is the same as the number of nodes in its combinatorial unstable (respectively, stable) set. Hence, in our implementation, we maintain the number of nodes in each trackable component and its combinatorial stable and unstable sets. This provides type information for all Morse sets at any stage of the algorithm.

Additionally, we store information describing trackable component changes in a compact *journal*, consisting of a list of *span changes* for each mesh edge. The span of a trackable component C in a mesh edge e is defined as the minimal subsegment of e containing all n-sets contained in e and belonging to C. Note that the span is empty if C contains no n-sets contained in e. It can also consist of an endpoint of e if the only such n-set is an e's endpoint. Whenever the span of a component C in e changes, we add an entry to e's span change list. The entry contains the value of E for which the change took place, ID of the trackable component C and the updated span of that component in e. The journal is used to explore the Morse hierarchy (Section 5).

4.3.3 Expansion Events

The idea behind expansion events is shown in Fig. 8 (left). As E increases, the fans of type T arcs out of a node n in $\partial\Delta$ in a triangle Δ expand, invading new n-sets. An expansion event can either

- move right (n,Δ)-endpoint to the next clockwise n-set b around $\partial\Delta$; in this case, the event is called the *right* $n \to b$ *expansion event*, or
- move the left (n,Δ)-endpoint to the next counterclockwise n-set b around $\partial\Delta$ (*left* $n \to b$ *expansion event*).

In what follows, for an n-set a in $\partial\Delta$, $CW_\Delta(a)$, and $CCW_\Delta(a)$ will denote the clockwise (respectively, counterclockwise) neighbor of a around $\partial\Delta$.

Cost. An $n \to b$ expansion event is *valid* if n and b are not contained in the same mesh edge and $f(\Delta) \cdot \vec{pq} \geq 0$, where p and q are (respectively)

- the counterclockwise-most endpoints of n and b for a right expansion event, or,
- the clockwise-most endpoint of n and b for a left expansion event.

The cost of a valid expansion event is given by $\sqrt{|\, f(\Delta)\,|^2 - (f(\Delta) \cdot \frac{\vec{pq}}{|\vec{pq}|})^2}$. This is illustrated in Fig. 8 (right). For an n-set that is a mesh vertex, both clockwise- and counterclockwise-most endpoints are equal to the vertex itself.

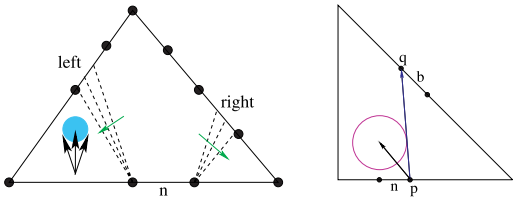

Fig. 8. Expansion events. Left: as the E increases ($F_E(\Delta)$ is the blue disc), the set of points that can be reached from n by following a feasible segment in Δ (located between the dashed lines) widens as indicated by the green arrows. At some point, the left and right (n, Δ)-endpoints move: the left one to the counterclockwise neighbor in $\partial\Delta$ and the right one—to the clockwise neighbor. The cost of the expansion event is equal to the radius that causes the endpoint of the dashed line opposite to n to move to a different n-set. Right: the cost of a right $n \to b$ expansion event is equal to the minimum radius E for which the disk of vectors (magenta circle) centered around $f(\Delta)$ (shown as black arrow) intersects the half-line extending in the direction of $\vec{pq}$ (blue arrow).

Initialization. For any node n and its right (n, Δ)-endpoint b in the initial graph, the right $n \to CW_\Delta(b)$ expansion event is placed on the queue if it is valid. Similarly, for any left (n, Δ)-endpoint b in the initial graph, the left $n \to CCW_\Delta(b)$ expansion event is inserted into the queue if it is valid.

Action. An $n \to b$ expansion event (left or right) results in adding the arc $n \to b$ to $\mathcal{G}$ and requires a respective update of the bit maps (Section 4.3.1).

For the right event, the right (n, Δ)-endpoint moves to b. The right $n \to CW_\Delta(b)$ expansion event is added to the queue if it is valid to allow the fan to continue to expand. The left (b, Δ)-endpoint in $\mathcal{G}^T$ moves to n (Fig. 9). However, in some cases there is no fan of arcs out of b in $\mathcal{G}^T$ representing feasible segments of $-F_E$ in Δ, and hence neither left nor right (b, Δ)-endpoints is defined (Fig. 10). In this case, we create a fan of arcs in $\mathcal{G}^T$ as described above with both the left and the right (b, Δ)-endpoints set to n.

Analogously, for the left $n \to b$ expansion event, the left (n, Δ)-endpoint moves to b. The left $n \to CCW_\Delta(b)$ expansion event is added to the queue if it is valid. In the compact representation of $\mathcal{G}^T$, the right (b, Δ)-endpoint moves to n if it is exists. Otherwise, left and right (b, Δ)-endpoints are created and set to n.

4.3.4 Edge Events

Edge events correspond to directed edges $p \to q$ of the mesh. A $p \to q$ edge event takes place when constant velocity paths moving from p to q become feasible as a result of increase in E, i.e., when $F_E(\bar{pq})$ starts to contain nonnegative multiples of $\vec{pq}$.

Cost. The cost $c_{p\to q}$ of a $p \to q$ edge event is defined in a way consistent with the way in which $F_E(\bar{pq})$ is determined [27]. First, we make the two triangles Δ_1 and Δ_2 incident upon $e = \bar{pq}$ coplanar by rotating one of them around e (together with the vector assigned to it by f). The cost $c_{p\to q}$ is equal to the distance between the interval connecting $f(\Delta_1)$ and $f(\Delta_2)$ and the half-line consisting of nonnegative multiples of the vector $\vec{pq}$. This is illustrated in Fig. 11. In practice, only events of strictly positive costs matter: if $c_{p\to q} = 0$, then $F_0(\bar{pq})$ contains a nonnegative multiple of $\vec{pq}$, which is reflected in the initial transition graph.

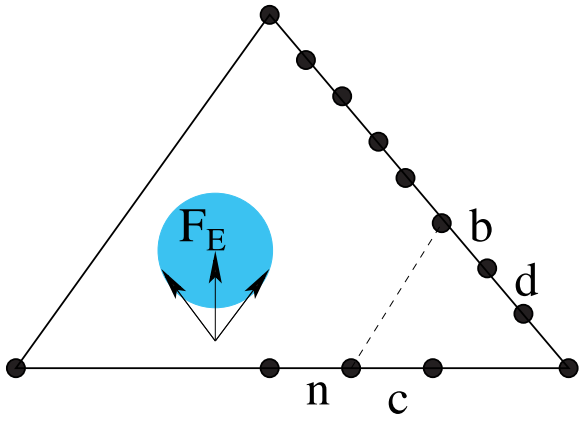

Fig. 9. A mesh triangle Δ and $F_E(\Delta)$ at the moment of an $n \to b$ expansion event. The dashed line is parallel to the clockwise-most vector in $F_E(\Delta)$. As a result of the event, b becomes the right (n, Δ)-endpoint. Just before the event takes place, c is the right (b, Δ)-endpoint in $\mathcal{G}^T$. This is because, there is a feasible segment of $-F_E$ connecting b to c, but there is no feasible segment connecting b and n. Since the event causes the arc $n \to b$ to be added to $\mathcal{G}$, the arc $b \to n$ needs to be inserted into $\mathcal{G}^T$. Hence, the right (b, Δ)-endpoint in $\mathcal{G}^T$ needs to be moved from c to n.

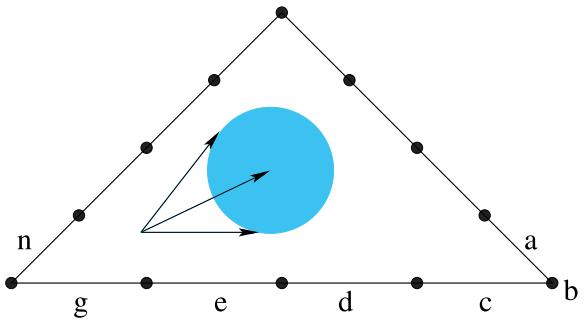

Fig. 10. A mesh triangle Δ with the vector $f(\Delta)$ pointing away from the bottom edge. Let E_0 be the smallest value of E for which $F_E(\Delta)$ (the blue disc) contains a vector parallel to the bottom edge. For E barely smaller than E_0, the right (n, Δ) endpoint is a, since there is a feasible segment in Δ connecting n to a but there is no such feasible segment connecting a point on n to the bottom edge. E moving past E_0 triggers a series of right expansion events that move the right (n, Δ)-endpoint to g. However, notice that there are no feasible segments for $-F_E$ in Δ starting on the bottom edge if $E < E_0$. Hence, there are no fans of arcs out of nodes b, c, d, e, or g in $\mathcal{G}^T$ representing feasible segments traversing Δ. Such segments and fans appear as E moves past E_0. Therefore, our algorithm creates such fan for each of these nodes.

Initialization. All edge events are computed and placed on the queue at startup.

Action. A $p \to q$ edge event results in insertion of $2^l + 1$ arcs connecting consecutive pairs of n-sets along e (in order from p to q) into the graph $\mathcal{G}$. Reverse arcs are added to $\mathcal{G}^T$. These arcs represent feasible segments following a mesh edge and therefore they are stored explicitly for both $\mathcal{G}$ and $\mathcal{G}^T$. The bitmaps are updated as described in Section 4.3.1, for each of the added arcs. No new events are added to the queue.

4.3.5 Triangle Events

A triangle event E_Δ arises when a mesh triangle Δ becomes stationary as a result of increase in the stability parameter. Recall that this means that the zero vector becomes an element of $F_E(\Delta)$.

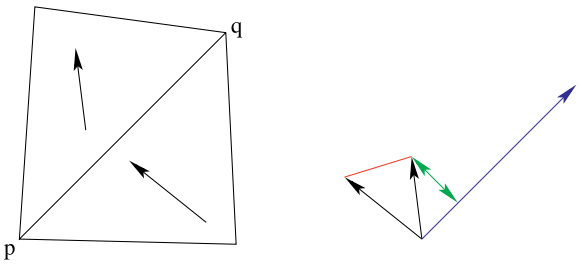

Fig. 11. Determining the cost of an edge event corresponding to $p \rightarrow q$. First, the two incident triangles are made coplanar (left). The figure on the right shows the vectors assigned to each of the triangles and the vector $\vec{pq}$ anchored at the same point. The red interval connects the endpoints of the former two vectors. The cost of the event is equal to the distance between the red interval and the half-line extending in the direction of $\vec{pq}$ (blue line). In this case, it is equal to the length of the green interval.

Cost. The cost of a triangle event E_Δ is equal to the magnitude of $f(\Delta)$. We treat the cost of a triangle event as slightly greater than the cost of any edge event corresponding to a directed edge of Δ. Thus, all edge events for directed edges of Δ (the cost of some of them is also equal to the magnitude of $f(\Delta)$) are processed *before* the triangle event corresponding to Δ. This means that all edges of Δ become stationary before Δ does.

Initialization. All triangle events are computed and placed on the queue at startup.

Action. An event E_Δ makes Δ stationary. Since all edges of Δ are stationary by then, all n-sets on the boundary of Δ form a loop in the supertransition graph. This means that no type T arcs representing trajectories through Δ are necessary during subsequent bitmap updates. Thus, we remove all fans of type T arcs in Δ from $\mathcal{G}$ and $\mathcal{G}^T$. No new events are generated.

4.3.6 Tangency Events

Let Δ be a triangle incident to an edge e, with $f(\Delta)$ pointing toward e. The (e, Δ) tangency event is triggered when a vector parallel to e finds its way to $F_E(\Delta)$. A tangency event causes feasible segments in Δ and starting at e to appear, and therefore requires creation of new fans of type T arcs starting at n-sets in e.

Cost. Let $\vec{e}$ be a unit vector parallel to e. The cost of the (e, Δ) tangency event is equal to $\sqrt{|f(\Delta)|^2 - (f(\Delta) \cdot \vec{e})^2}$, i.e., the distance between $f(\Delta)$ and the line parallel to e.

Initialization. All tangency events are computed and placed on the queue at startup.

Action. Let p and q be endpoints of e such that $f(\Delta) \cdot \vec{pq} \geq 0$. Arcs connecting any n-set on e except for q to the edge piece n in $\partial\Delta$ incident to q but not contained in e (Fig. 12) are inserted to $\mathcal{G}$. For compact graph representations used here, this means

- Making n the left and the right (m, Δ)-endpoint for any n-set m contained in e other than q; this is equivalent to adding the single-arc fan of type T arcs in Δ out of m containing only $m \rightarrow n$.
- The consistent update to $\mathcal{G}^T$, i.e., making o, the edge piece in e incident to q, the right (n, Δ)-endpoint in $\mathcal{G}^T$ if the vector $\vec{pq}$ points in the clockwise direction along the boundary of Δ (as shown in Fig. 12) or making o the left (n, Δ)-endpoint in $\mathcal{G}^T$ otherwise.

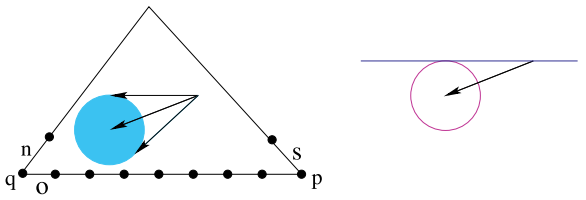

Fig. 12. Left: a tangency event corresponding to an edge $e = \bar{pq}$. The vector assigned to a triangle Δ points toward e. Immediately prior to the event, there are no feasible segments in Δ starting on the edge $\bar{pq}$. In particular, n-sets contained in $\bar{pq}$ do not have right or left Δ-endpoints. After the event, n becomes the right and left Δ-endpoint for all n-sets contained in $\bar{pq}$ except for q. Also, before the event, s is the right (n, Δ) endpoint for n in $\mathcal{G}^T$, since there is a feasible segment of $-F_E$ connecting n to s (here, it is close to being horizontal), but no feasible segment connecting n to p. When the event is processed, arcs connecting any n-set contained in the edge $\bar{pq}$ except for q to n are added to $\mathcal{G}$. The reverse arcs (all starting at n) need to be added to $\mathcal{G}^T$. This is equivalent to moving the right (n, Δ)-endpoint from s to o. Right: the cost of the event is equal to the radius of the magenta circle, centered at the endpoint of $f(\Delta)$ (black vector) and tangent to the linear space parallel to e (blue line).

Finally, we generate new expansion events. For any n-set m contained in e other than q, the right $m \rightarrow CW_\Delta(n)$ and the left $m \rightarrow CCW_\Delta(n)$ expansion event are added to the queue if they are valid. The costs of these events are determined as described in Section 4.3.3.

4.4 Computational Complexity

The worst case running time of the algorithm described in this section is $O(Em + Vm^2)$, where E is the maximum number of arcs in the supertransition graph, V is the number of its nodes, and m is the number of Morse sets determined from the initial supertransition graph for F_0 (i.e., the transition graph of f). This is because, each of the arcs of the supertransition graph $\mathcal{G}$ or its transpose $\mathcal{G}^T$ is traversed no more than m times during the bitmap updates (since there are m discrete stable sets and m discrete unstable sets to be maintained). Also, each of the V bitmaps is updated at most $2m$ times, and each update takes time proportional to m. Analysis of the result of each bitmap update (i.e., detecting Morse set mergers) can be implemented in time proportional to m.

5 Visualization

To extract the geometric model of the stable Morse decomposition in the hierarchy corresponding to stability parameter value E_0, we use the journal described in Section 4.3.2. For each edge e, we determine the span of every trackable component in e by scanning the e's span change list until an entry corresponding to $E > E_0$ is encountered. For any trackable component C, the last entry in the list associated with C contains the span information we need. If there is no such entry, the span of C in e is empty.

Having determined spans of a trackable component C in each edge, we build a geometric model of the corresponding Morse set. In this paper, we use a fast method that is not based on the pseudo-Morse sets used in [27], [29]. Our model of the Morse set corresponding to C depends *only* on the span of C in all mesh edges. It consists of no more than

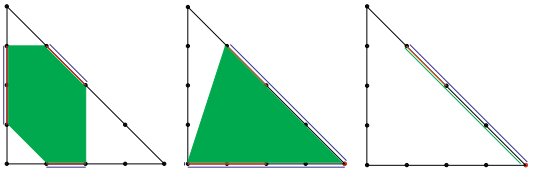

Fig. 13. Sets C_Δ contributing to a geometric model of a Morse set. N-sets corresponding to nodes in C are shown in red. Spans of C in all edges are shown in blue. The set C_Δ is shown in green. Note that in the case shown on the right C_Δ is degenerate, so it does not contribute to the geometric model of the Morse set.

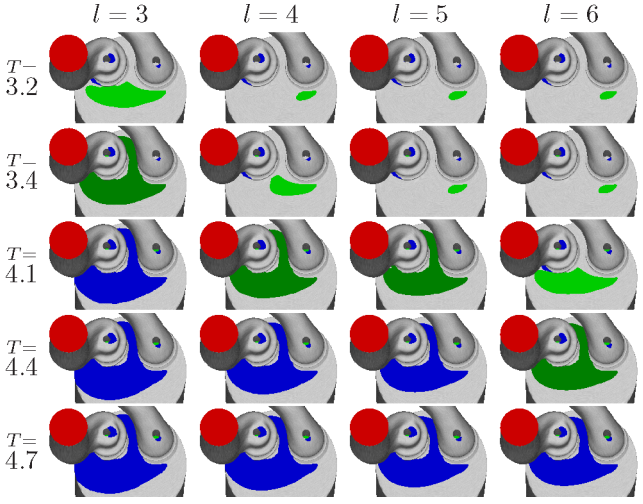

Fig. 14. A matrix of images of stable Morse decompositions for the diesel engine data set, with columns corresponding to different subdivision depths and rows—to different values of the stability parameter T. Note that while the results for any fixed value of T depend on l, for any value of l they undergo similar evolution as T is increased. In particular, notice that the dark green Morse set of type $(0,-)$ appears for any value of l shown here, albeit with different T.

one polygon (with at most six edges) for each triangle of the mesh and no more line segments and points than mesh edges. In contrast, in order to build a geometric model of a pseudo-Morse set, one needs to generate a polygon, line segment or point for each arc connecting nodes in C. Since the number of arcs of the transition graph is typically significantly higher than the number of nodes, the method proposed here is much faster and leads to a smaller number of geometric primitives.

Let C be a trackable component of the supertransition graph corresponding to $E = E_0$. For a triangle Δ, let C_Δ be the convex hull of the union of all spans of C in edges of Δ. A number of examples are shown in Fig. 13. The geometric model of the Morse set corresponding to C consists of nonempty spans of C in mesh edges and all sets C_Δ that are nondegenerate polygons. Note that the span of C in a mesh edge e, if it is nonempty, is a line segment contained in e or an endpoint of e (Section 4.3.2). Let us stress that, like pseudo-Morse sets, the geometric models of Morse sets are *supersets* of the true Morse sets defined by the trackable components.

6 Experimental Results

In this section, we discuss experimental results obtained for three vector fields derived from 3D fluid simulations by extrapolating the velocity field to the boundary of the model [19], [23]. All reported running times were measured on a laptop with an i7-2630QM processor, an NVIDIA GTX460M graphics card and 16 GB of RAM.

TABLE 1
Comparison of Running Times of the Journal-Based Morse Decomposition Computation and the Algorithm of Szymczak [27]

Dataset	stability parameter, E	[27]	Journal
diesel engine	4.03 (5.7%, 15.8%)	88.9	0.55
gas engine	5.5 (9.3%, 33.8%)	39.2	0.23
gas engine	7 (11.9%, 43.0%)	49.4	0.24
cooling jacket	1 (3.1%, 34.2%)	157.5	1.7
cooling jacket	1.5 (4.6%, 51.1%)	191.8	2.0
cooling jacket	2 (6.1%, 68.1%)	170.6	2.3

All reported running times are expressed in seconds. With each value of E, we provide the percentage of maximum vector magnitude and the mean vector magnitude that it represents.

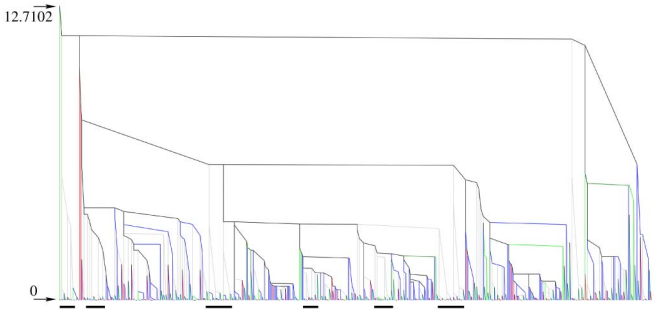

Fig. 15. The Morse merge tree for the cooling jacket data set. The tree has 536 leaf nodes (this is the number of nontrivial Morse sets of f determined from the initial transition graph). Note that that there is a large number of low stability Morse sets that undergo cancelations as E is increased. Examples include leaf nodes above the thick black lines. The height of the top vertex is 12.7102.

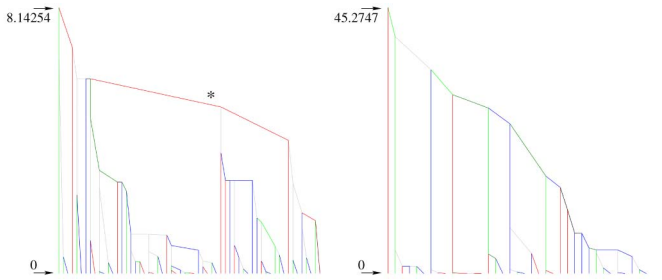

Fig. 16. Morse merge trees for the gas engine (left) and for the diesel engine (right).

To visualize the Morse sets, we use the coloring scheme of [27]. Trivial Morse sets of type $(0,0)$ are shown in gray. Repelling Morse sets of types $(1,+)$ and $(0,+)$ are shown in red (the ones of the former type slightly brighter). Similarly, attracting Morse sets of types $(1,-)$ and $(0,-)$ are shown in green (slightly brighter for type $(1,-)$). Sets of type $(-1,0)$, similar to saddles, are shown in blue. All other Morse sets are shown in dark red if repelling, dark green if attracting, and black if neither attracting nor repelling. The same color coding is applied when drawing the Morse merge trees. Colors of the tree edges represent types of the corresponding Morse sets.

The numbers of triangles and vertices in the data sets used in this section are 26,298 and 13,151 (gas engine), 221,574 and 110,789 (diesel engine), and 227,868 and 113,868 (cooling jacket). For subdivision depth $l = 5$ (which we focus on in this section), our implementation generated Morse hierarchies in 6.5, 104, and 154 minutes (respectively) for these data sets. Generally, increasing (decreasing) l by 1

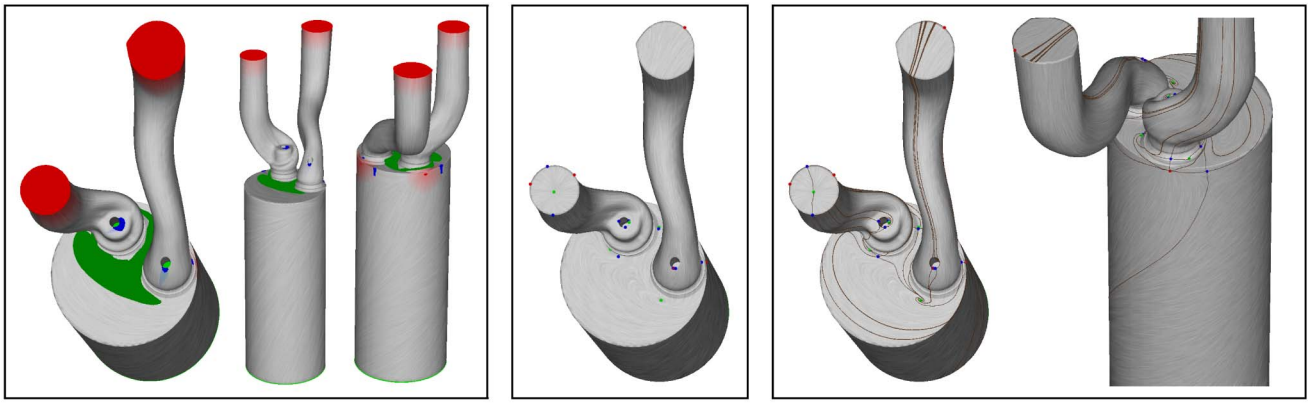

Fig. 17. Left box: an overview of the boundary topology provided by Morse decomposition of stability $E = 4.03$. The image combines image-based LIC visualization [22] with Morse set advection along the flow. The ends of intake tubes become repelling Morse sets of type $(1, +)$. There are attracting Morse sets of type $(1, -)$ on each of the valves (which can be seen through the holes in the tubes in the left image). The Morse sets of type $(-1, 0)$ (similar to saddles) on the rims of the two holes indicate that the flow moving along boundaries of the tubes splits, partly moving toward the valves and partly toward the cylinder (color bleeding clearly shows the flow descending toward the cylinder in the left image). The large green Morse set on top of the cylinder is of type $(0, -)$, i.e., has structure similar to an attracting periodic orbit. One can interpret it as an indication of an onset of the swirling motion [23]. Since the set is attracting, one may expect that the flow near that set is pushed out toward the bottom of the cylinder and gradually smoothed to form the swirling motion. Other features that can easily be observed are two repelling Morse sets of type $(1, +)$ on the side of the cylinder, which correspond to places where the flow dispersed at the valves hits the cylinder walls. The bottom of the cylinder is a Morse set of type $(1, -)$. For comparison, we provide an image showing a fine Morse decomposition extracted using the method of Szymczak and Zhang [29] in the middle box and image of connecting regions (similar to separatrices) computed using the technique described in [28] in the right box. Note that other standard field topology algorithms would lead to a similar set of features (stationary points and separatrices). The physical meaning of the features is not immediately clear. Of course, the fine features *can* be merged into coarser features similar to those in the left box, but this requires some mental effort that our approach is aiming to reduce.

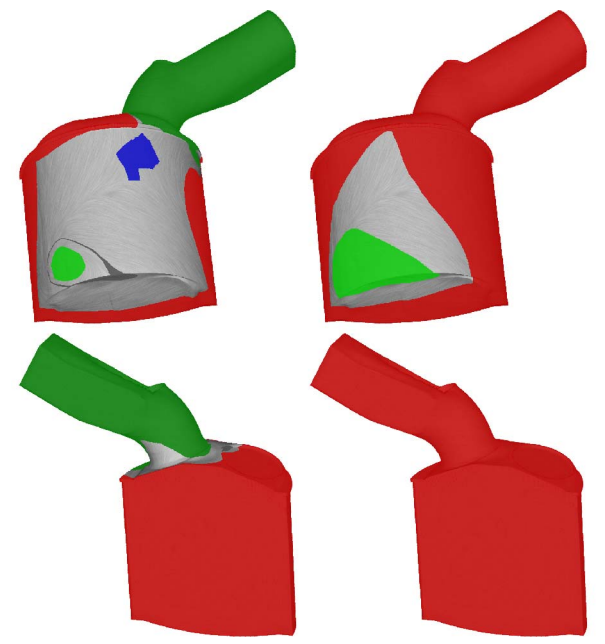

Fig. 18. Two Morse decompositions for the gas engine, both of high stability (left: $E = 5.5$, right: $E = 7$). In the case shown on the right, there are just two Morse sets, of types $(1, +)$ and $(1, -)$. The dominant flow along the boundary is from the red set to the green set. The images on the left show that the intake is covered by an *attracting* Morse set of index $(0, -)$. This indicates that near the model's surface the flow generally moves toward the intake. This could be because of the tumble motion as well as because the fuel mixture is directed in an off-tangent direction by the valve: particles hit the wall and disperse causing the flow to move toward the intake near the boundary. Note that there is a *repelling* Morse set of index $(1, +)$ on the engine's valve (which cannot be seen here; it corresponds to the long red branch of the Morse merge tree marked by a star in Fig. 16(left)).

increases (respectively, decreases) these running times by a factor of about 4. This behavior is expected since the number of arcs in the supertransition graph depends on l in the same way. Although the hierarchies for different values of l are not guaranteed to be the same, they tend to be similar in practice for close values of l. More precisely, one can usually find a Morse decompositions with the same structure in the hierarchies built for a similar values of l. An example is shown in Fig. 14. Movies showing the growth and mergers of Morse sets during the sweep algorithm for $l = 5$ are included in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2012.147.

Geometric models of Morse sets can be generated at least close to two orders of magnitude faster using the journal (Section 5) than from scratch, using the algorithm of Szymczak [27]. We provide a comparison of running times for stability parameter values used in Figs. 17, 18, and 19 in Table 1. Note that the reported running times include time needed for construction of the display list containing geometric models of Morse sets. For data set of sizes and complexities comparable to the three data sets discussed here, the journal-based algorithm provides an attractive way to explore the Morse hierarchy in an interactive manner. We believe this is important since stability parameter values leading to most insightful Morse decompositions are typically unknown.

The cooling jacket data set is the most complicated of the three data sets: it gives rise to a large number of Morse sets and a large number of complex Morse sets (nontrivial ones of types other than $(0, \pm)$, $(1, \pm)$, and $(-1, 0)$) arise in the Morse hierarchy. In contrast, the gas engine and the diesel engine data are relatively simple. The Morse merge trees for all three data sets are shown in Figs. 15 and 16.

Stable Morse decompositions can provide a useful high-level overview of the flow. An example of this for the diesel

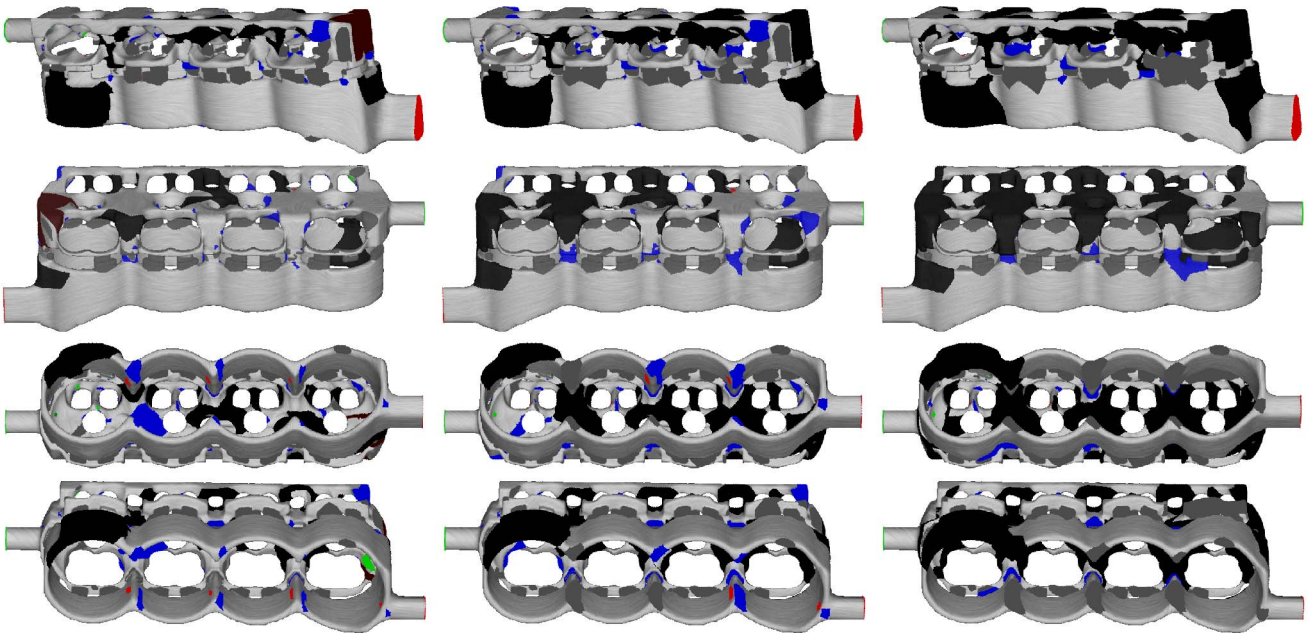

Fig. 19. Four views of the results for the cooling jacket data set, corresponding to $E = 1$ (left), $E = 1.5$ (center), and $E = 2$ (right). The inlet and the outlet are covered by repelling and attracting Morse sets of types $(1, +)$ and $(1, -)$. There are no complex attracting or repelling Morse sets for $E = 1.5$ or $E = 2$. There is one for $E = 1$ (the dark red set above the inlet, upper right in the top left image). Such Morse sets indicate areas of close to recurrent (circulating) flow near the surface which could negatively impact the heat transfer, leading to hot spots [19]. All attracting or repelling Morse sets for $E = 1.5, 2$ are relatively small and of a simple type $(1, \pm)$. Some are hidden in fine features of the geometry and likely represent strong flow directed away from or toward the wall, or are artifacts of the extrapolation process. Also, notice the regular pattern of pairs of Morse sets of types $(1, +)$ (red) and $(-1, 0)$ (blue) on the bottom of the cooling jacket (two bottom rows) that gradually cancel (i.e., merge into trivial, gray, Morse sets) as E is increased, in order from the outlet to the inlet. The red sets are repelling and represent the spots on the cooling jacket wall into which the flow is directed because of the jacket's geometry. The cancelation order can be explained by generally decreasing trend in the flow velocity as distance from the intake increases. The black Morse sets are complex, and may also be used as indicators of low velocity or close to recurrent flow near the wall (potential hot spots). However, these sets are neither repelling nor attracting (which means that fluid flows both into them and out of them), so they are less likely to point to a severe problem.

engine data set is shown in Fig. 17. Fig. 18 shows the output for the gas engine data set.

Morse decompositions of relatively high stability for the cooling jacket data set are shown in Fig. 19. While these decompositions are still complex, their number of Morse sets is not as overwhelming as the number of features obtained using the standard vector field topology [19] or by using Morse decompositions without stability requirement [29]. This makes certain features, patterns, and flow properties easier to see.

7 Conclusion

We introduced an algorithm for building the Morse hierarchy, a hierarchy of Morse decompositions driven by stability of Morse sets under perturbation. Our method is developed for piecewise constant vector fields on triangulated manifold surfaces. The resulting hierarchy can be explored in an interactive fashion. We also introduce the Morse merge tree, a representation of all mergers that Morse sets undergo as they grow as a result of a growing stability parameter. The Morse set mergers can be viewed as a counterpart of critical point cancelation operations in the scalar field case. However, in our case, the result of a merger may potentially be a nontrivial Morse set. The Morse merge tree can potentially be used to estimate the stability of any Morse set. Stability can be defined as the smallest value of E for which that Morse set undergoes a nonnegligible merger event (i.e., merger with a Morse set that is not trivial). This stability measure could be used to design Morse decomposition simplification schemes that would be driven by other objectives (e.g., result in Morse decompositions with no complex Morse sets).

Clearly, it would be interesting to improve the running time of the algorithm (both in the worst case and in practice). One potentially interesting research direction would be to investigate the relationship between the supertransition graph and the Filippov-Ważewski relaxation theorem and related results in differential inclusions and control theory [1]. Roughly speaking, these results show that if a point b can be reached from a point a by following trajectories of a differential inclusion $\dot{x} \in F(x)$ with $F(x)$ convex everywhere, then it can also be reached from a by a trajectory that moves only in the direction of extreme vectors of $F(x)$. This could mean that in the PC setting, just the extreme vectors of a CVPC vector field (i.e., the clockwise- and counterclockwise-most vectors in $F(\Delta)$ for each Δ) determine the topological structure of the feasible vector fields. If this is indeed the case, the size of the supertransition graph could be substantially reduced, by restricting the arcs to represent only the extreme directions.

It would be interesting to explore more informative ways to visualize the Morse decomposition, or Morse sets themselves (especially the complex ones). Diffusion of Morse sets (Fig. 17) seems to enhance the visualization, but only for relatively simple Morse decomposition and surface geometry. Examples of potential enhancements could include displaying the connection regions between Morse sets [7], exit sets for the Morse sets [8], adding information on the topology of the Morse sets, or techniques

for combining geometric models for several stability parameter values in a single image.

Extending our technique to three dimensions will be challenging in the near future because of much higher memory requirements. Improvements based on the Filippov-Ważewski theorem (which is dimension independent) could make this extension possible. Still, it is an interesting question if stable Morse decompositions and Morse hierarchy have potential to enhance slice-based 3D flow analysis, or if surface flows obtained by methods other than extrapolation to the boundary can lead to useful insights into the structure of 3D flows.

Finally, it would be interesting to examine the applicability of recent incremental algorithms for strongly connected components, such as [16], to our setting. The best of these algorithms run in $O(E^{3/2})$ or $O(V^{5/2})$ time and hence they may be competitive to our implementation if the number of Morse sets is relatively large.

REFERENCES

[1] J.-P. Aubin and A. Cellina, *Differential Inclusions: Set-Valued Maps and Viability Theory.* Springer, 1984.

[2] H. Bhatia, S. Jadhav, P.-T. Bremer, G. Chen, J.A. Levine, L.G. Nonato, and V. Pascucci, "Edge Maps: Representing Flow with Bounded Error," *Proc. IEEE Pacific Visualization Symp. (PacificVis' 11),* pp. 75-82, 2011.

[3] H. Bhatia, S. Jadhav, P.-T. Bremer, G. Chen, J.A. Levine, L.G. Nonato, and V. Pascucci, "Flow Visualization with Quantified Spatial and Temporal Errors Using Edge Maps," *IEEE Trans. Visualization and Computer Graphics,* vol. 18, no. 9, pp. 1383-1396, Sept. 2012.

[4] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci, "A Multi-Resolution Data Structure for 2-Dimensional Morse Functions," *Proc. IEEE Visualization,* pp. 139-146, 2003.

[5] 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.

[6] 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 2007.

[7] 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.

[8] C. Conley, *Isolated Invariant Sets and Morse Index,* no. 38. Am. Math. Soc., 1978.

[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms,* second ed. MIT Press, 2001.

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

[11] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological Persistence and Simplification," *Discrete Computational Geometry,* vol. 28, pp. 511-533, 2002.

[12] H. Edelsbrunner and E.P. Mücke, "Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms," *ACM Trans. Graphics,* vol. 9, no. 1, pp. 66-104, 1990.

[13] F. Effenberger and D. Weiskopf, "Finding and Classifying Critical Points of 2D Vector Fields: A Cell-Oriented Approach Using Group Theory," *Computing and Visualization in Science,* vol. 13, pp. 377-396, 2010.

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

[15] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann, "A Topological Approach to Simplification of Three-Dimensional Scalar Functions," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 4, pp. 474-484, July/Aug. 2006.

[16] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R.E. Tarjan, "Incremental Cycle Detection, Topological Ordering, and Strong Component Maintenance," *ACM Trans. Algorithms,* vol. 8, article 3, 2011.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] R.S. Laramee, H. Hauser, L. Zhao, and F.H. Post, "Topology-Based Flow Visualization, the State of the Art," *Topology-Based Methods in Visualization,* H. Hauser, H. Hagen, and H. Theisel, eds., pp. 1-19, Springer-Verlag, 2007.

[22] 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.

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

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

[25] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matkovic, and H. Hauser, "On the Way Towards Topology-Based Visualization of Unsteady Flow - The State of the Art," *Proc. EuroGraphics Conf.,* pp. 137-154, 2010.

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

[27] A. Szymczak, "Stable Morse Decompositions for Piecewise Constant Vector Fields on Surfaces," *Computer Graphics Forum,* vol. 30, no. 3, pp. 851-860, 2011.

[28] A. Szymczak, "Morse Connection Graphs for Piecewise Constant Vector Fields on Surfaces," *Computer Aided Geometric Design,* in press, 2012.

[29] A. Szymczak and E. Zhang, "Robust Morse Decompositions of Piecewise Constant Vector Fields," *IEEE Trans. Visualization and Computer Graphics,* vol. 18, no. 6, pp. 938-951, June 2012.

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

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

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

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

[34] 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 Gdansk, Poland in 1994 and the PhD degree in mathematics and the MS degree in computer science in 1999 from the Georgia Institute of Technology. Currently, he is an assistant professor in the Department of Electrical Engineering and Computer Science, 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.