Efficient Computation of Combinatorial Feature Flow Fields

Jan Reininghaus, Jens Kasten, Tino Weinkauf, and Ingrid Hotz

Abstract—We propose a combinatorial algorithm to track critical points of 2D time-dependent scalar fields. Existing tracking algorithms such as Feature Flow Fields apply numerical schemes utilizing derivatives of the data, which makes them prone to noise and involve a large number of computational parameters. In contrast, our method is robust against noise since it does not require derivatives, interpolation, and numerical integration. Furthermore, we propose an importance measure that combines the spatial persistence of a critical point with its temporal evolution. This leads to a time-aware feature hierarchy, which allows us to discriminate important from spurious features. Our method requires only a single, easy-to-tune computational parameter and is naturally formulated in an out-of-core fashion, which enables the analysis of large data sets. We apply our method to synthetic data and data sets from computational fluid dynamics and compare it to the stabilized continuous Feature Flow Field tracking algorithm.

Index Terms—9.VI.IX.II Flow Visualization, 7.II.II.I Graph algorithms

1 Introduction

Time-dependent 2D scalar data arises in many scientific disciplines. For the analysis of such data, the extraction of minima, saddles, and maxima of each individual time step has been proven useful. These point features of the data are often called critical points. To understand the dynamic behavior of time-dependent data, it can be beneficial to analyze the temporal evolution of these critical points.

To enable an efficient quantification of the temporal evolution of the critical points, we can track them over time. In this paper, we call such a tracked critical point a critical line of the data. Many different algorithms that extract critical lines have been proposed, see Section 2 for a small overview.

For smooth functions, the Feature Flow Field method [1] provides a particularly sound mathematical foundation. Given a smooth time-dependent scalar field, the critical lines are implicitly defined by streamlines in a higher dimensional derived vector field.

While this method works well for smooth functions, its application to functions that are only continuous is problematic as derivatives have to be computed. To circumvent this problem, derivative free algorithms employing concepts from algebraic topology have been developed recently, see Section 2.

The main remaining weakness of the available algorithms is their inability to handle noisy data in a meaningful way. Such data usually contains an overwhelming number of critical lines that hinder meaningful visual data analysis. To reduce the number of critical lines, one typically smooths the data or discards short critical lines. Both approaches can be problematic. A simple smoothing of the data may remove important critical lines and affect the spatial position of the critical lines, see Figure 6 for an example. Discarding short critical lines may remove an important and stable, but short lived feature. See Figure 7 for an example of such a short but important critical line.

This paper proposes a combinatorial algorithm that is able to track critical points in noisy data. This robustness is achieved by combining Forman's notion of a combinatorial gradient field [2] with the notion of Persistence proposed by Edelsbrunner et al. [3]. Persistence is a well founded importance measure for critical points. Together, these concepts enable a robust and consistent combinatorial representation of the gradient of a scalar function. Both fundamental concepts will be briefly introduced in a graph theoretic formulation in Section 3.1.

A definition for a critical line of a sequence of combinatorial gradient fields was recently proposed by King et al. [4]. The basic idea is similar to the continuous Feature Flow Field method – a higher dimensional field is constructed in which the critical lines are given by combinatorial streamlines. We therefore refer to the higher dimensional field as a Combinatorial Feature Flow Field in this paper. We formulate King's definition of critical lines in combinatorial gradient fields using a graph theoretic formulation in Section 3.2.

- *J. Reininghaus, J. Kasten and I. Hotz are with Zuse Institute Berlin, E-mail: reininghaus@zib.de, kasten@zib.de and hotz@zib.de.*
- *T. Weinkauf is with Max Planck Institute for Informatics, Saarbrücken, Germany, E-mail: weinkauf@mpi-inf.mpg.de*

Our main contribution is the introduction of the first efficient algorithm that extracts the critical minima, saddle, and maxima lines in 2D discrete scalar fields using Combinatorial Feature Flow Fields.

The efficiency of our algorithm is achieved by pointing out that, at least in 2D, the combinatorial critical lines as defined in [4] can in fact be computed without considering the higher dimensional combinatorial feature flow field – we only need to consider the sequence of combinatorial gradient fields. We provide an informal proof of the equivalence of the original definition of combinatorial critical lines to our simplified definition in Section 4.2.

The proposed algorithm has many valuable properties. It has a reasonable running time (see Table 1) and is naturally formulated in an out-of-core fashion enabling the analysis of large data sets as only two subsequent time steps have to be kept in memory. The input consists of a regular cell complex, so the algorithm can deal with many widely used representations of discrete data like triangulations, quadrangulations, or a mixture of these. It contains only one easily-tuned computational parameter, the persistence threshold σ, used to construct the combinatorial representation of the gradient fields.

Due to the combinatorial nature of our algorithm, we can formulate a natural spatio-temporal importance measure for the resulting critical lines called Integrated Persistence (see Section 3.3).

2 Related Work

Many algorithms that track features in time-dependent data have been proposed in many different scientific communities. A lot of this work has been partially inspired by object tracking methods in the area of computer vision, see [5] for a survey. In the context of visual data analysis, tracking approaches can roughly be categorized into three classes depending on the treatment of the temporal dimension [6].

The first class considers feature tracking as a two-step process: feature extraction for each time slice and subsequent feature matching solving a correspondence. Such methods do not rely on a temporal interpolation. Event analysis mostly happens implicitly during tracking defined by event functions. Common tracked features are volumes or areas, boundaries or contours and points. Correspondence criteria use distance metrics of the domain and the attribute space, which are in general based on application specific heuristics. Typical attributes comprise feature size, shape descriptors or also texture characteristics [7]. Features are linked, if their distance falls below a given threshold [8], [9], [10], [11]. Improvements using feature overlap instead of Euclidean distances are used in [12]. A more global approach is followed in [13] employing a best matching algorithm. Improved tracking can be achieved by utilizing additional information for motion prediction [14]. In [15], an algorithm is proposed that progressively tracks isosurfaces using the isosurface at time t as an initial guess for the next time-step $t+1$. An extension to tracking of the entire contour tree using volume overlap has been proposed in [16].

The second class of algorithms considers time as additional dimension, treated equally to spatial dimensions. Features are extracted from space-time directly, thus increasing the dimension of the domain and the features by one. Tracking is accurate with respect to the chosen temporal interpolation. No explicit distance metrics for features are needed. Event analysis is mostly a subsequent step after tracking and is based on well-founded theory. Methods extracting isosurfaces in space-time have been proposed in [17], [18]. A topological event analysis based on the Reeb-Graph of the surface resulting from sweeping contours has been performed in [19], [20]. The development of topological structures in 2D and 3D flow fields has been analyzed in [21], [22]. These algorithms consider vector fields composed of space-time cells with linear interpolation, for which events are restricted to cell boundaries. Critical point tracking thus reduces to the computation of entry and exit points for each cell. Similarly, [23] introduces an algorithm to track vortex core lines over time and scale space searching for features, represented as parallel vectors, on all boundary cells of the space-time cell. While giving accurate results, these methods are sensitive to noisy data and a high feature density. To reduce the number of extracted features and events, a common practice is to delete short living features. A combinatorial approach to track critical points is based on the definition of Jacobi sets [24]. It consists of Jacobi edges, which are extracted from a spatial-temporal simplicial complex assuming a linear interpolant. The decision whether an edge belongs to the Jacobi curve involves the topological analysis of the lower link of vertices and edges of the simplicial complex. While providing a nice theoretical framework, the resulting Jacobi curves of real data sets are often very complex and hard to analyze. Based on this work it is also possible to track the evolution of the Reeb-graph of a scalar function [25].

The third class of algorithm combines aspects of both above-mentioned types. They represent the dynamic behavior of features implicitly as streamlines of a higher dimensional derived vector field in space-time. Critical points can then be tracked by computing certain streamlines in this vector field, referred to as a Feature Flow Field [1]. Recently, a combinatorial version of the Feature Flow Field method has been proposed [4]. This method is discussed in detail in Section 3.2 and provides the mathematical foundation for our novel tracking algorithm presented in Section 4.

3 FUNDAMENTAL CONCEPTS

The purpose of this section is to introduce the reader to the main concepts that build the mathematical foundation for our combinatorial tracking algorithm described in detail in Section 4. We first introduce the reader to the well known concept of combinatorial gradient vector fields (CGF) in Section 3.1 using a graph theoretic formulation. Using this concept we can define the notion of a combinatorial feature flow field (CFFF) in Section 3.2. We conclude this section with a definition of a time-aware importance measure for the tracked critical points that is based on the notion of persistence.

3.1 Combinatorial Gradient Fields

For simplicity, we restrict ourselves to 2D manifolds while the mathematical theory for combinatorial gradient fields is defined in a far more general setting [2]. Let C denote a finite regular cell complex of a 2D manifold. Examples of such cell complexes are triangulations or quadrangular meshes. Given C, we first define its cell graph $G_C = (S, L)$ that encodes the combinatorial information contained in C in a graph theoretic setting.

The nodes S of the graph consist of the cells C of the complex and each node u^p is labeled with the dimension p of the cell it represents. For a triangulation, the nodes of the cell graph therefore consist of the vertices (0-cells), edges (1-cells), and triangles (2-cells).

The links L of the graph encode the neighborhood relation of the cells in C: if the cell represented by node u^p is in the boundary of the cell represented by node w^{p+1} then $\ell^p = \{u^p, w^{p+1}\}$ is a link in the graph. Note that we label each link with the dimension of its lower dimensional node.

A matching of a graph is defined as a subset of links such that no two links are adjacent. Using these definitions, a *combinatorial vector field* V on a regular cell complex C can be defined as a matching of the cell graph G_C (see Figure 1, middle). An arrow representation of this combinatorial field as used in [2] is shown in Figure 1 left.

The nodes of the graph that are not covered by V are called critical points. If u^p is a critical point of V, we say that the critical point has index p. A critical point of index p is called sink $(p = 0)$, saddle $(p = 1)$, or source $(p = 2)$ (see Figure 1, right).

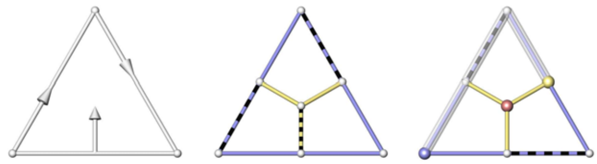

Fig. 1. Combinatorial gradient fields (CGF) – basic definitions. Left: arrow representation of a CGF on a single triangle. Middle: the same CGF represented as a matching (dashed links) of a cell graph consisting of 0-links (blue) and 1-links (yellow). Right: topological features of a CGF – a minimum (blue), a saddle (yellow), a maximum (red), and a separatrix (transparent).

A combinatorial p-streamline is a path in the graph whose links alternate between V and the complement of V and the dimension of the links equals p. A p-streamline connecting two critical points is called a separatrix (see Figure 1, right). If a p-streamline is closed, we call it either an attracting periodic orbit $(p = 0)$ or a repelling periodic orbit $(p = 1)$.

As shown in [26], a *combinatorial gradient field* (CGF) V can be defined as a combinatorial vector field that contains no periodic orbits. In the context of CGF we refer to a critical point u^p as a minimum $(p = 0)$, saddle $(p = 1)$, or maximum $(p = 2)$. For the computation of a CGF that represents the input data we refer to [27], [28], [29].

When we deal with noisy data, the corresponding CGF contains a huge number of minima, saddles, and maxima. Fortunately, the theoretical foundation of CGF [30] allows for a consistent removal of these spurious features. Suppose there is a unique separatrix connecting a saddle to a maximum or minimum. Reversing this separatrix results in a CGF without this pair of critical points, see [31] for an explanation of this property. When we simplify a CGF using this idea we have to decide on the order of the simplifications. A well founded order is given by persistence [3].

Note that the need for a simplified CGF implies that one cannot make direct use of the algorithm proposed in [32] since it computes a simplified Morse-Smale complex and not a simplified CGF. Reconstructing a CGF from a Morse-Smale complex is a nontrivial problem, especially in higher dimensions.

To track critical points in noisy data, we compute a CGF with a given persistence threshold σ. For example, if the noise is in the range $[-\epsilon, \epsilon]$, then it suffices to compute its CGF with a persistence threshold of $\sigma = 2\epsilon$ to remove all noise induced critical points. For more information on the connection between discrete Morse theory and persistence simplification we refer the interested reader to [28].

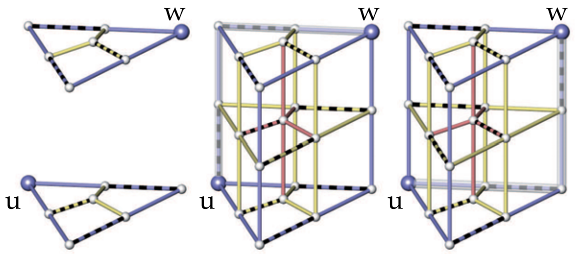

Fig. 2. Combinatorial feature flow fields – basic definitions. Left: Two subsequent combinatorial gradient fields V_0 and V_1. Middle: Forward tracking field $V_{[0,1]}$. Right: Backward tracking field $V_{[1,0]}$. The minima (blue) u in V_0 and w in V_1 are tracked as there is a combinatorial 0-streamline (transparent) in $V_{[0,1]}$ and a combinatorial 0-streamline in $V_{[1,0]}$ that connect u and w.

3.2 Combinatorial Feature Flow Fields

Using the combinatorial representation of the gradient fields defined above, we will now describe the combinatorial feature flow field concept introduced in [4] that allows us to track critical points in our graph theoretical framework. This formulation enables an efficient and simple implementation described in Section 4.

Given a sequence of combinatorial gradient fields $(V_t)_{t=0,1,2,\ldots,T}$ on a cell complex C of a 2D manifold we now define the notion of a combinatorial feature flow field (CFFF) that allows us to track the critical points in (V_t). For simplicity, we assume $T = 1$ as the general case follows easily. We first construct the cell graph of $C \times [0,1]$ using the graph theoretic formulation introduced in Section 3.1.

For a depiction of a simple example of the rather technical construction that follows, we refer to Figure 2. We start the construction of $G_{C\times[0,1]}$ with three copies G_C^1, G_C^2, G_C^3 of the cell graph G_C. We then add links to this graph that connect the corresponding nodes of G_C^1 with G_C^2 and G_C^2 with G_C^3. The label p of each node in G_C^2 is then increased by one. For example, if u^p is a node of the second copy that corresponds to the node w^2 of the first copy, then $p = 3$.

We can now define the forward tracking field $V_{[0,1]}$, a CGF of $G_{C\times[0,1]}$. We first use the matching V_0 to define a matching in G_C^1 and G_C^2 (see Figure 2, middle). For G_C^3 we use the matching V_1. We then add all links to the matching of $G_{C\times[0,1]}$ that connect a critical point of V_0 with a node of G_C^2.

Constructing a forward tracking field $V_{[0,T]}$ for the whole sequence of combinatorial gradient fields (V_t) can be done iteratively: if we have a forward tracking field for $V_{[0,k]}$, we get $V_{[0,k+1]}$ as the union of $V_{[0,k]}$ and $V_{[k,k+1]}$. The backward tracking field $V_{[T,0]}$ can be defined by reversing the order of the sequence (V_t). As proven in [4], the forward tracking field defined above is indeed a combinatorial gradient field as it does not contain any periodic orbits. Also, the only critical cells of this CGF are the cells that are critical in V_T.

We are now in a position to give a precise definition of the space-time relation of critical points in this combinatorial setting. Let u^p and w^p denote critical points in (V_t). We say u^p and w^p are connected if and only if there is a combinatorial p-streamline connecting u^p with w^p within $V_{[0,T]}$ and a combinatorial p-streamline connecting u^p with w^p within $V_{[T,0]}$. For future reference, we call the set of lines that connect the critical points of (V_t) the critical lines of (V_t). Note that in principal this definition allows for splitting and merging critical saddle lines. While our implementation allows for this behavior we have not observed any such critical saddle lines in our numerical experiments.

The presented approach is related to the continuous Feature Flow Field method [1] – both approaches for the tracking of critical points define a higher dimensional field where the critical points can be tracked by streamlines. We therefore refer to the approach presented in this section as the Combinatorial Feature Flow Field method (CFFF).

3.3 Integrated Persistence

This section proposes an importance measure for the critical lines of a sequence of T scalar fields (f_t) defined on a 2D manifold as introduced in Section 3.2. To incorporate the spatial importance of the critical points that make up the critical line we can make use of the notion of persistence [33]. Loosely speaking, persistence measures the stability of the critical points with respect to perturbations of the data values.

For simplicity we restrict the definition of persistence to the 1D case and refer the interested reader to [33] for the general case. For a 1D function f this measure can be defined by considering the number of components of the sublevel sets $F_r = f^{-1}((-\infty, r])$. As r increases the number of components in F_r changes: when r passes the value of a local minimum a component is born, while two components merge when r passes the value of a local maximum. A maximum is then paired with the larger minimum of the two merged components. The persistence of the paired critical points is now defined as the difference of their function values.

We now define an importance measure for a critical line L as the sum of the persistence values of the critical points that make up the line divided by the total number of time steps in the data set T. For future reference we refer to this measure as Integrated Persistence.

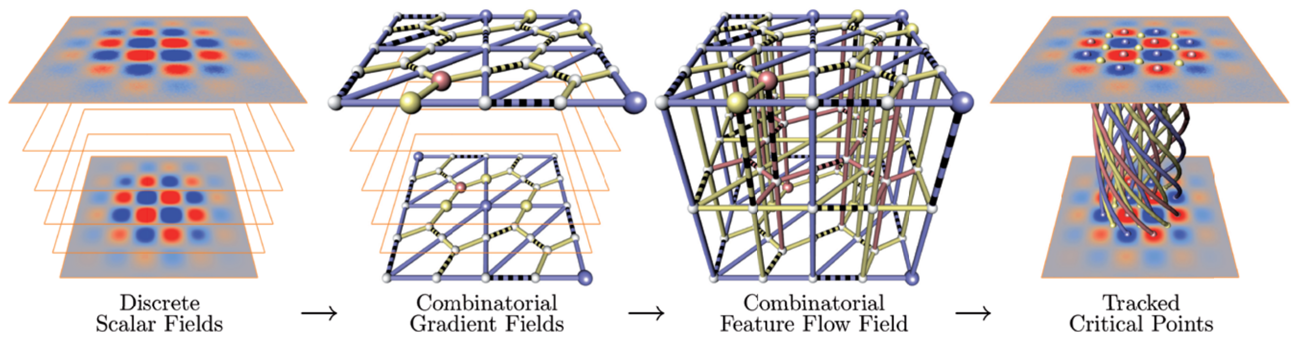

Fig. 3. Computational pipeline of the algorithm described in Section 4.

Note that in some sense Integrated Persistence is a spatio-temporal importance measure. A short, but spatially persistent critical line, is considered as important as a long critical line with low spatial persistence. Figures 9 and 7 demonstrate the physical relevance of Integrated Persistence.

4 Algorithm

In this section, we will describe our combinatorial tracking algorithm in detail. We will first give an overview of the algorithmic pipeline in Section 4.1, describing the input, output and out-of-core approach. Section 4.2 describes how we can efficiently track critical points. We will finish this section with a detailed description of our algorithm including pseudo-code to ensure a good reproducibility of the results presented in Section 5.

4.1 Overview

The input of our algorithm consists of a regular cell complex C of a 2D manifold and a sequence of scalar fields (f_t) defined on the 0-cells of C. A simple example of such input data is a triangulation or a quadrangular mesh with a sequence of scalar values defined on each vertex. We then compute a sequence of combinatorial gradient fields (V_t) with persistence threshold σ that represents the gradient of the input data in a discrete fashion. To deduce an importance measure for our result we will also require the persistence values of the critical points contained in (V_t).

A closer inspection of the definition given in Section 3.2 reveals that we can compute all critical lines contained in (V_t) in a streaming fashion – it is sufficient to compute the critical lines of each consecutive pair of the sequence (V_t). Due to the combinatorial nature of the critical lines they can easily be merged afterwards to get the result for the complete data set. The importance measure for a critical line introduced in Section 3.2 can be computed by adding the persistence values of the critical points contained in the line. See Figure 3 for an overview of the overall algorithm.

4.2 Efficient Extraction of Critical Lines in CFFF

As described above, it suffices to track the critical points for each consecutive pair (V_k, V_{k+1}) of the sequence of combinatorial gradient fields (V_t). As defined in Section 3.2, a critical point of V_k is connected to a critical point of V_{k+1} if and only if there is a combinatorial streamline in the forward tracking field $V_{[k,k+1]}$ and a combinatorial streamline in the backward tracking field $V_{[k+1,k]}$ connecting the two points.

The goal of this section is to give a simple algorithm that finds all pairs of critical points that satisfy this condition. It will be shown that we actually do not need to construct the higher dimensional cell graph $G_{C\times[0,1]}$. This significantly reduces the runtime, memory consumption, and greatly simplifies the implementation of our algorithm.

For a depiction of the following argument, we refer to Figure 2. We start with the minima. Let u denote a minimum in V_k. When we iterate the combinatorial 0-streamlines of the forward tracking field that start in u we see that there is only a single streamline that ends in a minimum of V_{k+1}. This is due to two reasons. First, the structure of the forward tracking field implies that the only way to reach V_{k+1} is to start with the matched link adjacent to u. Second, a combinatorial streamline whose first node is not a 1-cell and whose first link is matched, is uniquely defined as it cannot split. For an explanation of this basic property of combinatorial streamlines we refer the reader to [31]. The same arguments can be employed to show that there is only a single streamline connecting a minimum of V_{k+1} to a minimum of V_k in the backward tracking field.

Tracking minima is therefore a rather simple procedure. Given a minimum u in V_k we find its only possible partner w in V_{k+1} by computing the unique streamline in the forward tracking field that starts in u with a matched link. We then compute the unique streamline in the backward tracking field that starts in w with a matched link. If this streamline ends in u, then u and w are connected in the sense of the

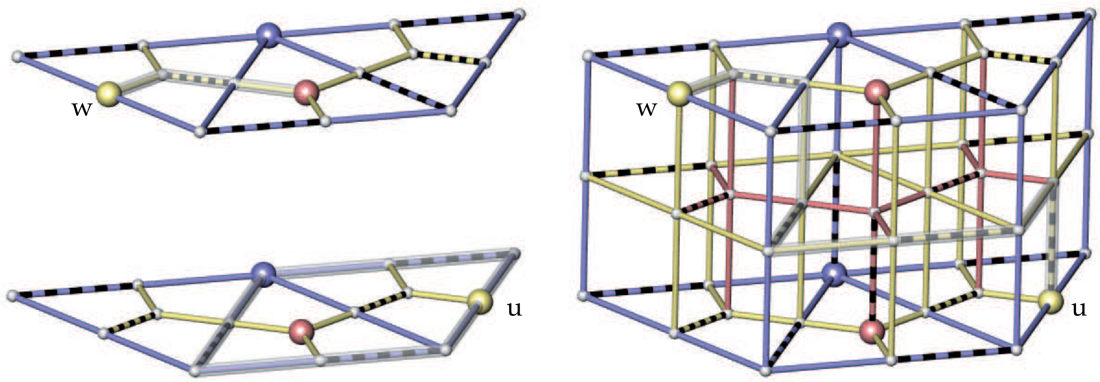

Fig. 4. Left: Two subsequent combinatorial gradient fields V_0 and V_1 on three triangles. Right: Forward tracking field $V_{[0,1]}$. The saddles (yellow) u in V_0 and w in V_1 are connected in $V_{[0,1]}$ by a combinatorial 1-streamline (transparent) that connects the corresponding nodes. Note that the minima lines (transparent) of the saddle of V_0 (bottom-left) intersect the maxima lines (transparent) of the saddle in V_1 (top-left).

definition given in Section 3.2.

Note that we do not actually need to construct the forward and backward tracking fields to compute these combinatorial streamlines. It suffices to trace them in the given pair of CGFs V_k and V_{k+1} as can be seen in Figure 2.

The maxima can be tracked in the same way, we only have to switch forward and backward tracking fields: the maxima of V_{k+1} have only a single partner in V_k in the forward tracking field, and the maxima of V_k have only a single partner in V_{k+1} in the backward tracking field.

While tracking minima and maxima has been proven to be rather simple, tracking of saddles seems to be a very daunting task as the combinatorial 1-streamlines in the higher dimensional tracking fields may merge and split (see Figure 4). On first sight, it seems that the only way to compute the critical saddle lines is a brute-force depth-first-search in the tracking fields. However, a close inspection of the structure of the tracking fields reveals that this is not actually necessary.

Consider the 1-streamlines of the forward tracking field that start in a saddle u of V_k and end in a saddle w of V_{k+1}. If we think of the graph of the forward tracking field as consisting of three layers (the three copies of G_C), we can observe three properties of these streamlines (see Figure 4 for an example):

1) The layer of the nodes of the streamlines only increases and the only node of the bottom layer is the node in which we start.
2) The section of these streamlines that runs through the second layer follows the 0-streamlines of V_k that start in u.
3) The section of these streamlines that runs through the third layer follows the 1-streamlines of V_{k+1} and ends in w.

These properties imply that there is a combinatorial 1-streamline in the forward tracking field that connects u with w if and only if the 0-streamlines of V_k that start in u intersect the 1-streamlines of V_{k+1} that end in w. Similarly, there is a combinatorial 1-streamline in the backward tracking field that connects w with u if and only if the 0-streamlines of V_{k+1} that start in w intersect the 1-streamlines of V_k that end in u.

Instead of a brute-force search in the higher dimensional cell graph $G_{C\times[0,1]}$, it therefore suffices to intersect the separatrices of u defined by V_k with the separatrices of w defined by V_{k+1} in the low dimensional cell graph G_C. This simplifies the following tracking algorithm significantly.

4.3 Implementation

The main algorithm that tracks the critical points of a sequence of discrete scalar fields (f_t) defined on the 0-cells of a cell complex C is given in Algorithm 1. Line 1 constructs the cell graph G_C of the cell complex C as defined in Section 3.1. The CGF subfunction called in Lines 3 and 4 computes a combinatorial gradient field with a persistence threshold σ.

To do this, we first compute an initial CGF using the algorithm *ProcessLowerStar* proposed in [34]. The basic idea of this algorithm is to apply simple homotopic expansions in the lower star of each 0-cell. The algorithm results in a combinatorial gradient field V whose critical nodes coincide with the changes of the topology of the sub-level complexes of C. For more algorithmic details and the proof of this property, we refer the interested reader to [34].

We can thereby state an estimate for the precision of our method. Suppose that we sample an analytic function with a uniform grid whose cells have size h. Furthermore, assume that h is sufficiently small

Algorithm 1 Main CFFF algorithm

Input: C, (f_t), T, σ
Output: All critical lines in $V_{[0,T]}$
1: $G_C \leftarrow$ constructCellGraph(C)
2: **for** $k = 0$ to $T - 1$ **do**
3: $V_k \leftarrow$ CGF(G_C, f_k, σ)
4: $V_{k+1} \leftarrow$ CGF(G_C, f_{k+1}, σ)
5: lines $\leftarrow$ lines $\cup$ trackMinMax(G_C, V_k, V_{k+1})
6: lines $\leftarrow$ lines $\cup$ trackSaddles(G_C, V_k, V_{k+1})

Algorithm 2 Combinatorial Streamline Integrator: *traceLine(...)*

Input: $G_C = (S, L)$, $V_k \subset L$, $u \in S$, $p = 0, 1$
Output: Combinatorial p-streamline that starts in u
1: **loop**
2: Line.append(u)
3: **if** there exists w: $\{u, w\} = \ell^p \in V_k$ **then**
4: u $\leftarrow$ w
5: $V_k \leftarrow V_k^c$
6: **else**
7: **return**

Algorithm 3 Min and max tracking algorithm: *track-MinMax(...)*

Input: $G_C = (S, L)$, $V_k \subset L$, $V_{k+1} \subset L$
Output: All critical min/max lines in $V_{(k,k+1)}$
1: **for all** $u^p \notin S(V_k)$ and $p \neq 1$ **do**
2: $p \leftarrow max(0, p - 1)$
3: Line $\leftarrow$ traceLine(G_C, V_{k+1}, u, p)
4: $w \leftarrow$ Line.last()
5: Line $\leftarrow$ traceLine(G_C, V_k, w, p)
6: **if** Line.last() = u **then**
7: MinMaxCritLines.append($\{u, w\}$)

Algorithm 4 Separatrix Integrator: *traceSeps(...)*

Input: $G_C = (S, L)$, $V_k \subset L$, $u \in S$, $p = 0, 1$
Output: All combinatorial p-streamlines that start in saddle u
1: **for all** $\{u, w\} = \ell^p \in L$ **do**
2: Line $\leftarrow$ traceLine(G_C, V_k, w, p)
3: Separatrices.append(Line)

so that all critical points of the analytic function are represented in the discrete representation. Then the distance between the exact critical points and the critical points in the CGF is smaller than h.

To compute the simplified CGF we follow the approach presented in [29]. We thereby compute the whole sequence of simplified CGFs, which has the advantage of allowing the user to quickly select the appropriate simplification threshold σ in a post processing step.

For the persistence values, we employ the matrix reduction algorithm presented in [35].

Line 5 and 6 extract the critical minima, maxima and saddle lines of the current pair of CGFs as defined in Section 3.2 using the simplified but equivalent definition presented in Section 4.2.

To compute the critical lines we need to compute a lot of combinatorial p-streamlines in a given CGF V_k. The pseudo-code for such a combinatorial streamline integrator is given in Algorithm 2. Almost all computational time of the main Algorithm 1 is spent integrating such lines which makes the performance of this algorithm crucial for the overall runtime. Note that due to the structure of the cell graph G_C and the matching property of V_k, there cannot exist multiple links that fulfill the condition in Line 3. Of course, an actual implementation would not take the complement of the matching in each iteration (Line 5). One would rather simply switch the if condition in Line 3.

Using Algorithm 2, we can compute the critical minima and maxima lines as shown in Algorithm 3. For each minimum or maximum u (Line 1, $S(V_k)$ denotes the nodes S contained in the set of matching links V_k) of the first CGF V_k, we integrate the corresponding combinatorial p-streamline in V_{k+1} (Line 3). We now take the last point w of this streamline as the start point of a streamline in V_k (Lines 4, 5). If this streamline comes back to u (Line 6), we found a critical line in $V_{(k,k+1)}$ and append the pair $\{u, w\}$ to the result.

To compute the critical saddle lines, we need to compute the separatrices of the saddles. A simple method that returns the separatrices of type p of the saddle u in the CGF V_k is given in Algorithm 4. We iterate over all adjacent links ℓ of the saddle u of the given type p (Line 1) and integrate the combinatorial p-streamline that starts in the end point w of ℓ (Line 2). This line is then appended to the separatrices (Line 3).

Using Algorithm 4, we can now trace the critical saddle lines as shown in Algorithm 5. Each saddle is appended to the nodes covered by its separatrices of type 0 (Lines 1-8). We then iterate over each saddle u of V_k (Line 9). The possible saddle partners for u in V_{k+1} are then given as the union of the saddles in V_{k+1} whose separatrices of type 0 are intersected by the separatrices of type 1 of u (Lines 10-13). For each such partner w we then iterate its partners in V_k (Lines 14-18). If this set of saddles contains u, we have found a critical saddle line in $V_{(k,k+1)}$ and append the pair $\{u, w\}$ to the result.

5 RESULTS

In this section, we will evaluate the algorithm presented in Section 4. We show its robustness with respect to noise in Section 5.1, compare it to the continuous Feature Flow Field tracking algorithm in Section 5.2, and apply it to a real-world data set from computational fluid dynamics in Section 5.3.

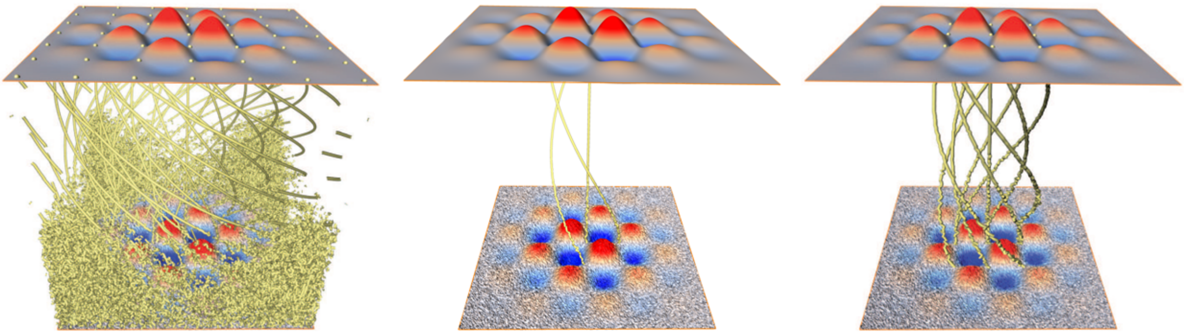

Fig. 5. Evaluation of noise robustness on a synthetic data set – time is represented using the z-coordinate. Left: critical saddle lines extracted with the Stable Feature Flow Field method. Middle: critical saddle lines of the Stable Feature Flow Field method filtered by line length. Right: critical saddle lines extracted by our method using an appropriate persistence threshold σ.

Algorithm 5 Saddle tracking algorithm: *trackSaddles(...)*

Input: $G_C = (S, L)$, $V_k \subset L$, $V_{k+1} \subset L$
Output: All critical saddle lines in $V_{(k,k+1)}$

```
for all u^1 ∉ S(V_k) do
    minLines_u ← traceSeps(G_C, V_k, u, 0)
    for all w ∈ minLines_u do
        saddles_k[w].append(u)
for all u^1 ∉ S(V_{k+1}) do
    minLines_u ← traceSeps(G_C, V_{k+1}, u, 0)
    for all w ∈ minLines_u do
        saddles_{k+1}[w].append(u)
for all u^1 ∉ S(V_k) do
    maxLines_u ← traceSeps(G_C, V_k, u, 1)
    partners_u ← ∅
    for all w ∈ maxLines_u do
        partners_u ← partners_u ∪ saddles_{k+1}[w]
    for all w ∈ partners_u do
        maxLines_w ← traceSeps(G_C, V_{k+1}, w, 1)
        partners_w ← ∅
        for all v ∈ maxLines_w do
            partners_w ← partners_w ∪ saddles_k[v]
        if u ∈ partners_w then
            SaddleCritLines.append( {u, w} )
```

We conclude the evaluation of our algorithm with a performance analysis in Section 5.4.

5.1 Robustness

To demonstrate the ability of our algorithm to deal with noisy data we consider a synthetic data set. The data values are given by a 2D analytic function sampled on a uniform 256×256 mesh. A height field visualization of this function is shown in Figure 5. This data is then rotated to generate a sequence of 256 scalar fields (f_t) on the uniform mesh. To show the influence of the noise on the extraction methods we added an increasing amount of noise to the second half of the sequence (f_t).

We then applied the algorithm presented in Section 4 and the stabilized continuous Feature Flow Field method [36] to this data set. Figure 5 shows the critical saddle lines extracted by these two algorithms. Due to the presence of noise, the continuous extraction method results in an overwhelming number of critical saddle lines. Note that some important lines are interrupted which implies that they are removed early when we filter the result by line length. In contrast, our combinatorial algorithm is able to extract all dominant critical saddle lines of this time-dependent data set using a persistence threshold σ slightly above the range of the noise .

5.2 Comparison

We compare our algorithm to the stabilized continuous Feature Flow Field method [36] using a data set from computational fluid dynamics [37]. The data set consists of a simulation of the time-dependent flow behind a cylinder. The data set is given on an adaptive mesh with $108k$ vertices and 320 time steps. We analyze the scalar quantity acceleration, a measure for vortex activity in fluid flows [38] depicted in Figure 9. For the combinatorial method we set the persistence threshold σ for the computation of the combinatorial gradient fields to about one percent of the data range.

Figure 6 shows the critical lines extracted by both methods in a small subregion of the data set. The continuous results are shown in the left (original data set) and middle column (a smoothed version of the data) while the combinatorial results are shown in the right column. The three rows show the critical minima lines (top), saddle lines (middle), and maxima lines (bottom).

In general, both methods extract the correct critical lines in the right half of the depicted subregion of

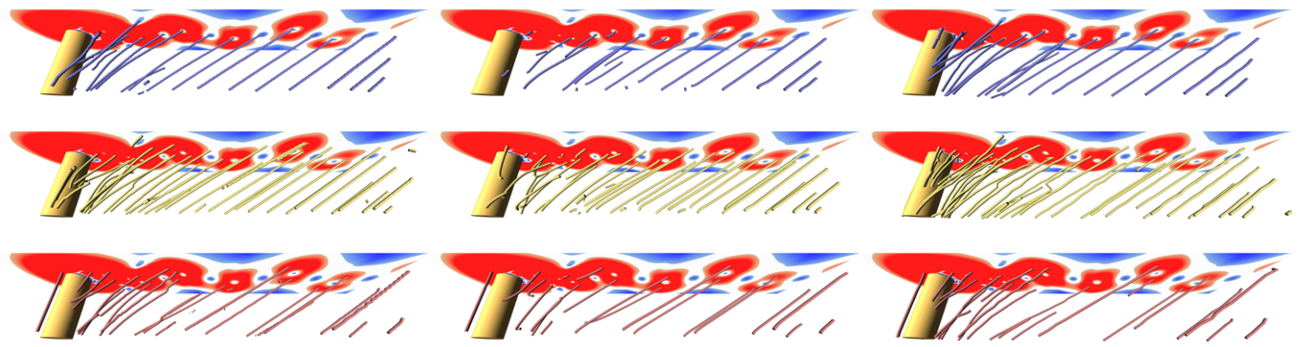

Fig. 6. Comparison between the Stable Feature Flow Field method (left column) and our combinatorial method (right column) on a subset of the cylinder flow data set. The middle column depicts the result of the Stable Feature Flow Field method applied to a pre-smoothed version of the data set. Top row: critical minima lines. Middle row: critical saddle lines. Bottom row: critical maxima lines.

the data set. Some lines extracted by the continuous method do contain a very high oscillation. Applying the continuous method to the smoothed version of the data set removes these oscillations, but also removes some important critical lines. Note that our combinatorial method deals directly with the original data without any modifications.

5.3 Application

We applied our method to a scalar data set derived from a flow simulation [39]. The simulation shows the flow over a cavity from left to right. Due to the cavity, there is a dominant vortex that separates from the wall after some time and moves through the cavity to the right side, where it hits the other wall.

As an indicator for time-dependent vortex structures, we used the acceleration, a scalar quantity whose dominant minima indicate vortex activity [38], [40]. Note that in contrast to the zeros of the velocity field, the minima of the acceleration do not depend on the chosen frame of reference. We computed the acceleration on the adaptive mesh that was used during the simulation of the flow consisting of $26k$ nodes for each of the 690 time steps. For the combinatorial computation of the critical lines, the persistence threshold σ was set to about one percent of the data range. Since we are only interested in the minima of the acceleration, we only show the critical minima lines in Figure 7. To demonstrate the physical significance of the importance measure introduced in Section 3.3, the thickness of the lines is determined by Integrated Persistence. The dominant vortices that pass through the cavity have a high Integrated Persistence. This can be visualized by seeding path lines in the vicinity of the lines with high Integrated Persistence.

Note that our algorithm has found one critical minima line that is difficult to observe manually (see zoom-in in Figure 7 and consider the color map therein). This short critical line has a higher Integrated Persistence than most other critical lines in the data set. By seeding path lines in its vicinity we observe that this line corresponds to strong vortex activity.

This example shows that it is in general problematic to use line length as an importance measure for critical lines.

5.4 Performance

The performance of our implementation was calculated for all three data sets used in this section. Table 1 contains the running times for a standard workstation containing two Intel Xeon E5620 CPUs. The table shows the number of data values given at the vertices of the grid and the number of slices T for which the critical lines were computed. Tracking the critical points in the computed CGFs is very fast – for a mesh with approximately one hundred thousand vertices, only 38 milliseconds are required for each time step.

For comparison, we have also measured the running time of the stable Feature Flow Field method. Computing the critical lines for the synthetic data set shown in Figure 5 with this method takes 333 seconds compared to the total running time of 149s using our method. Comparing the running time for the other data sets is problematic, since they are defined on an adaptive mesh and the implementation of the Feature Flow Field that is available to us can only be applied to uniform grids.

Note that the timings for the CGF computation represent the computation of the full hierarchy of CGFs, see [29]. The user can thereby quickly select an appropriate persistence threshold σ in a post processing step.

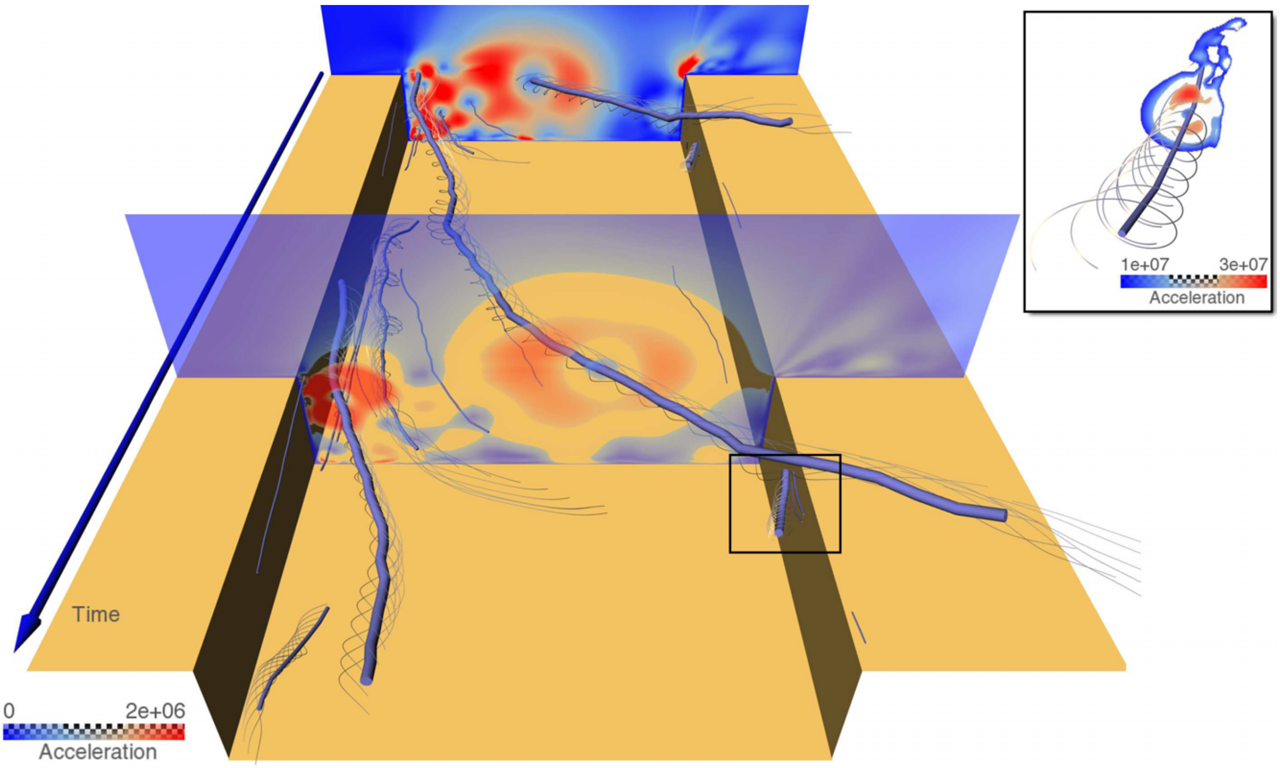

Fig. 7. Application of our method to a real-world data set from computational fluid dynamics. The data set is the simulation of the flow over a cavity. The dominant minima of the acceleration of the flow describe the vortex activity. We extracted the critical minima lines of this data set using our method. The thickness of these lines is defined by our novel importance measure integrated persistence. To demonstrate the physical relevance of this importance measure, path lines are seeded in the vicinity of the most important lines. Note that there is a short, but important critical line in this data set (see close-up). This shows that the length of a critical line by itself is not a good importance measure in general.

Dataset	#vertices	T	CGF	Pers.	CFFF	Total
Synthetic	65k	256	77s	67s	5s	149s
Cavity	26k	690	186s	110s	4s	300s
Cylinder	108k	320	368s	96s	12s	476s

TABLE 1
Performance analysis of our method. For each data set shown in the paper we measured the running time for the computation of the combinatorial gradient fields (CGF), the computation of the persistence values (Pers.), and the tracking of the critical points in the resulting sequence (CFFF).

6 Discussion and Future Work

As shown in Section 5, our novel combinatorial algorithm to extract critical lines of discrete scalar data works very well in practice:

- It effectively handles noisy data (see Figure 5).
- It allows for a physically relevant importance measure for the tracked critical points (see Figures 9 and 7).
- Its extracted features correspond to the results of the Feature Flow Field method for a smooth data set (see Figure 6).
- It has a practical running time (see Table 1).

The robustness of our algorithm with respect to noise is mainly due to the notion of persistence which allows for a robust computation of a CGF. Unfortunately, using persistence can be problematic if the data contains outliers. To efficiently deal with such data, an importance measure for critical points would need to be developed that can address outliers in a sensible way.

Many of the existing tracking algorithms mentioned in Section 2 extract bifurcation points, i.e. the points where a pair of critical points appears or disappears. The spatial importance of such critical points becomes arbitrarily small as they approach a bifurcation point, see Figure 8. Due to our focus on noise resilient extraction of critical lines, we do not aim at a precise computation of bifurcation points in this work. Note that critical points of course can appear or disappear in our method – we start tracking them as soon as their spatial importance is high enough to differentiate them from noise induced critical points.

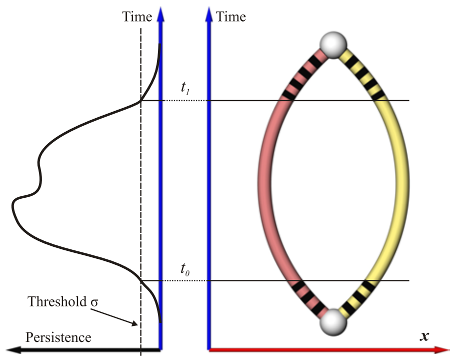

Fig. 8. Bifurcation handling in CFFF. Right: a maximum-saddle pair evolving over time. Left: spatial importance of this pair over time. The pair is only tracked while the spatial importance is above the threshold σ. For $t < t_0$ and $t_1 < t$ the critical points are considered as noise.

An extension of the presented algorithm to 3D may be quite feasible. The mathematical foundation for our algorithm presented in Section 3 easily extends to 3D. A close inspection of the definition of combinatorial critical minima and maxima lines in 3D reveals that they have the same combinatorial structure as in 2D. Given an algorithm that can compute a combinatorial gradient field in 3D we can therefore directly use our algorithm to track the minima and maxima of 3D time-dependent data.

Acknowledgments

This work was supported by the German Science Foundation (DFG) Emmy-Noether research program. Tino Weinkauf acknowledges support of the Max-Planck-Center for Visual Computing. The authors would like to thank Gerd Mutschke for providing the cylinder data set, Mo Samimy for providing the cavity data set, and David Günther for many fruitful discussions on this topic. All visualizations have been created using Amira – a system for advanced visual data analysis (http://amira.zib.de).

References

[1] H. Theisel and H.-P. Seidel, "Feature flow fields," in *VisSym '03: Proceedings of the symposium on Data Visualization 2003*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 141–148.

[2] R. Forman, "Combinatorial vector fields and dynamical systems," *Mathematische Zeitschrift*, vol. 228, no. 4, pp. 629–681, August 1998.

[3] H. Edelsbrunner, J. Harer, and A. Zomorodian, "Hierarchical morse complexes for piecewise linear 2-manifolds," in *Proc. of the 17th Symposium on Computational Geometry*, 2001, pp. 70–79.

[4] H. King, K. Knudson, and N. Mramor, "Birth and death in discrete morse theory," *arXiv:0808.0051v1*, 2008. [Online]. Available: http://adsabs.harvard.edu/abs/2008arXiv0808.0051K

[5] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, p. 13, 2006.

[6] F. H. Post, "The state of the art in flow visualization: Feature extraction and tracking," in *Computer Graphics Forum*, D. Duke and R. Scopigno, Eds. Oxford, UK and Boston, USA: Blackwell Publishing Inc, Jun. 2003, vol. 22(4), pp. 775–792. [Online]. Available: http://www.vrvis.at/scivis/star/index.html

[7] J. Caban, A. Joshi, and P. Rheingans, "Texture-based feature tracking for effective time-varying data visualizations," *IEEE Transactions on Visualization and Computer Graphics (Vis07)*, vol. 13, no. 6, pp. 1472–1479, 2007.

[8] R. Samtaney, D. Silver, N. Zabusky, and J. Cao, "Visualizing features and tracking their evolution," *Computer*, vol. 27, no. 7, pp. 20–27, 1994.

[9] F. Reinders, F. Post, and H. Spoelder, "Attribute-based feature tracking," in *Proceedings of EG - IEEE TCVG Symposium on Visualization '99*, 1999.

[10] D. Laney, P. T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci, "Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1053–1060, 2006.

[11] W. de Leeuw and R. van Liere, "Chromatin decondensation: A case study of tracking features in confocal data," in *VIS '01: Proceedings of the conference on Visualization '01*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 441–444.

[12] D. Silver and X. Wang, "Tracking and visualizing turbulent 3d features," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 129–141, 1997. [Online]. Available: www.caip.rutgers.edu/vislab.html

[13] G. Ji, "Feature tracking and viewing for time-varying data sets," Ph.D. dissertation, Ohio State University, Columbus, OH, USA, 2006, advised by Han-Wei Shen.

[14] F. Reinders, I. A. Sadarjoen, B. Vrolijk, and F. H. Post, "Vortex tracking and visualisation in a flow past a tapered cylinder," *Computer Graphics Forum*, vol. 21, no. 4, pp. 675–682, 2002.

[15] C. Bajaj, A. Shamir, and B.-S. Sohn, "Progressive tracking of isosurfaces in time-varying scalar fields," Department of Computer Sciences & TICAM, University of Texas Austin, Tech. Rep. TR-02-4, CS & TICAM, 2002.

[16] B.-S. Sohn and C. Bajaj, "Time-varying contour topology," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 1, pp. 14–25, 2006.

[17] C. Weigle and D. C. Banks, "Extracting iso-valued features in 4-dimensional scalar fields," in *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*. New York, NY, USA: ACM, 1998, pp. 103–110.

[18] G. Ji, H.-W. Shen, and R. Wenger, "Volume tracking using higher dimensional isosurfacing," in *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 209–216.

[19] G. H. Weber, P.-T. Bremer, M. S. Day, J. B. Bell, and V. Pascucci, "Feature tracking using Reeb graphs," in *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*, V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, Eds. Springer Verlag, 2011, pp. 241–253.

[20] P.-T. Bremer, G. H. Weber, V. Pascucci, M. Day, and J. B. Bell, "Analyzing and tracking burning structures in lean premixed hydrogen flames," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 2, pp. 248–260, 2010.

[21] X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen, "Topology tracking for the visualization of time-dependent two-dimensional flows," *Computer & Graphics*, vol. 26, pp. 249–257, 2002.

[22] C. Garth, X. Tricoche, and G. Scheuermann, "Tracking of vector field singularities in unstructured 3d time-dependent datasets," in *VIS '04: Proceedings of the conference on Visualization '04*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 329–336.

[23] D. Bauer and R. Peikert, "Vortex tracking in scale space," in *Joint Eurographics — IEEE TCVG Symposium on Visualization*, May 2002, pp. 140–147. [Online]. Available: citeseer.csail.mit.edu/bauer02vortex.html

[24] H. Edelsbrunner and J. Harer, "Jacobi sets of multiple morse functions," in *Foundations of Computational Mathematics, Minneapolis 2002*, F. Cucker, R. DeVore, P. Olver, and E. Sueli, Eds. Cambridge Universtiy Press, 2004, pp. 37–57.

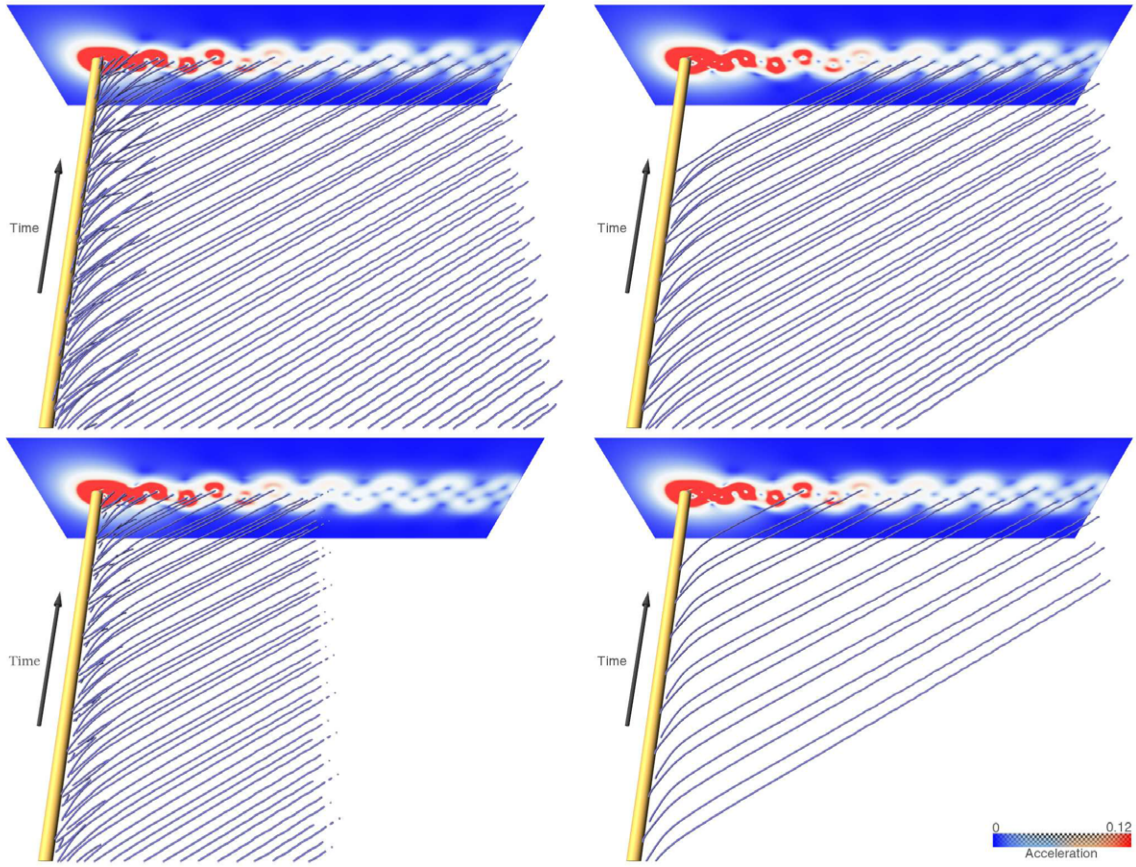

Fig. 9. Evaluation of different filter criteria for critical minima lines (blue) of the acceleration of a flow dataset. The dominant minima of the acceleration describe vortex activity of the flow. Top-left: all extracted critical minima lines computed by Algorithm 1 without any post processing. Top-right: lines filtered by length. Bottom-left: lines filtered by spatial persistence. Bottom-right: lines filtered by our novel importance measure integrated persistence. The lines with high integrated persistence correspond to the dominant vortex activity of this data set as shown in [41].

[25] H. Edelsbrunner, J. Harer, A. Mascarenhas, J. Snoeyink, and V. Pascucci, "Time-varying Reeb graphs for continuous space-time data," *Computation Geometry: Theory and Applications*, vol. 41, no. 3, pp. 149–166, 2008.

[26] M. K. Chari, "On discrete morse functions and combinatorial decompositions," *Discrete Math.*, vol. 217, no. 1-3, pp. 101–113, 2000.

[27] T. Lewiner, "Geometric discrete Morse complexes," Ph.D. dissertation, Department of Mathematics, PUC-Rio, 2005, advised by Hélio Lopes and Geovan Tavares. [Online]. Available: http://www.matmidia.mat.puc-rio.br/~tomlew/phd_thesis_puc_uk.pdf

[28] U. Bauer, C. Lange, and M. Wardetzky, "Optimal topological simplification of discrete functions on surfaces," *arXiv:1001.1269v2*, 2010.

[29] J. Reininghaus, D. Günther, I. Hotz, S. Prohaska, and H.-C. Hege, "TADD: A computational framework for data analysis using discrete morse theory," in *Proc. ICMS 2010*, 2010, (in press).

[30] R. Forman, "Morse theory for cell complexes," *Advances in Mathematics*, vol. 134, pp. 90–145, 1998.

[31] ——, "A user's guide to discrete Morse theory," in *Seminaire Lotharingien de Combinatoire*, vol. B48c, 2002, pp. 1–35.

[32] A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci, "A practical approach to Morse-Smale complex computation: scalability and generality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1619–1626, 2008.

[33] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," *Discrete Comput. Geom.*, vol. 28, pp. 511–533, 2002.

[34] V. Robins, P. Wood, and A. Sheppard, "Discrete morse theory for grayscale digital images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, to appear.

[35] D. Cohen-Steiner, H. Edelsbrunner, and D. Morozov, "Vines and vineyards by updating persistence in linear time," in *Proceedings of the twenty-second annual symposium on Computational geometry*, ser. SCG '06. New York, NY, USA: ACM, 2006, pp. 119–126. [Online]. Available: http://doi.acm.org/10.1145/1137856.1137877

[36] T. Weinkauf, H. Theisel, A. V. Gelder, and A. Pang, "Stable feature flow fields," *IEEE Transactions on Visualization and Computer Graphics*, 2010, accepted. [Online]. Available: http://tinoweinkauf.net/

[37] B. R. Noack, M. Schlegel, B. Ahlborn, G. Mutschke, M. Morzyński, P. Comte, and G. Tadmor, "A finite-time thermodynamics of unsteady fluid flows," *J. Non-Equilibr. Thermodyn.*, vol. 33, no. 2, pp. 103–148, 2008.

[38] J. Kasten, I. Hotz, B. Noack, and H.-C. Hege, "On the extraction of long-living features in unsteady fluid flows," in *Topological Methods in Data Analysis and Visualization. Theory, Algorithms, and Applications.*, V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, Eds. Springer, 2010, pp. 115–126.

[39] E. Caraballo, M. Samimy, and J. DeBonis, "Low dimensional modeling of flow for closed-loop flow control," *AIAA Paper*, vol. 59, 2003.

[40] R. Fuchs, J. Kemmler, B. Schindler, F. Sadlo, H. Hauser, and R. Peikert, "Toward a Lagrangian Vector Field Topology," *Computer Graphics Forum*, vol. 29, no. 3, pp. 1163–1172, 2010.

[41] T. Weinkauf, J. Sahner, H. Theisel, and H.-C. Hege, "Cores of swirling particle motion in unsteady flows," *IEEE Transactions*

on Visualization and Computer Graphics (Proceedings Visualization 2007), vol. 13, no. 6, pp. 1759–1766, November – December 2007.

Jan Reininghaus received the M.S. degree in Mathematics from the Humboldt University of Berlin, Germany, for a thesis on the numerical treatment of Maxwell's equations. He is a main author of OpenFFW, an open source finite element framework written in Matlab. Currently he is with the Scientific Visualization Department of Zuse Institute Berlin (ZIB) where he is working on his PhD thesis. His current research interests include Hodge theory, volume rendering, finite element exterior calculus and discrete Morse theory.

Jens Kasten is a junior researcher at the Zuse Institute Berlin. His main field of research is the analysis of time-dependent flow fields including the extraction of Lagrangian features. He studied mathematics and computer science at the University of Münster, Germany, and received his M.S. degree in mathematics in 2008.

Tino Weinkauf studied computer science with the focus on computer graphics at the University of Rostock, Germany, where he received his M.S. degree in 2000. Based on his research carried out at the Scientific Visualization department of Zuse Institute Berlin (ZIB) on feature-based analysis and comparison techniques for flow fields he received his awarded PhD in computer science from the University of Magdeburg in 2008. He spent two years as a post-doc at the Courant Institute of Mathematical Sciences, New York University, based on a Feodor Lynen research fellowship of the Alexander von Humboldt foundation. Since 2011 he is a senior researcher at the Max-Planck-Institute for Informatics in Saarbrücken, Germany, and the head of the research group "Feature-Based Data Analysis for Computer Graphics and Visualization." His current research interests focus on flow and tensor analysis, geometric modeling, and information visualization.

Ingrid Hotz received the M.S. degree in theoretical Physics from the Ludwig Maximilian University in Munich Germany and the PhD degree from the Computer Science Department at the University of Kaiserslautern, Germany. During 2003 – 2006 she worked as a postdoctoral researcher at the Institute for Data Analysis and Visualization (IDAV) at the University of California. Currently she is the leader of a junior research group at the Zuse Institute in Berlin Germany. Her research interests are in the area of data analysis and scientific visualization with focus on tensor and vector fields.