Appeared in: IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 13, NO. 6, pp. 1456 - 1463, 2007

# Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction

Filip Sadlo, Ronald Peikert, *Member, IEEE*

**Abstract**—This paper presents a method for filtered ridge extraction based on adaptive mesh refinement. It is applicable in situations where the underlying scalar field can be refined during ridge extraction. This requirement is met by the concept of Lagrangian coherent structures which is based on trajectories started at arbitrary sampling grids that are independent of the underlying vector field. The Lagrangian coherent structures are extracted as ridges in finite Lyapunov exponent fields computed from these grids of trajectories. The method is applied to several variants of finite Lyapunov exponents, one of which is newly introduced. High computation time due to the high number of required trajectories is a main drawback when computing Lyapunov exponents of 3-dimensional vector fields. The presented method allows a substantial speed-up by avoiding the seeding of trajectories in regions where no ridges are present or do not satisfy the prescribed filter criteria such as a minimum finite Lyapunov exponent.

**Index Terms**—Ridge extraction, flow visualization, coherent structures, vector field topology, unsteady vector fields.

## 1 Introduction

There are many applications in science and industry where visualization by isosurfaces is not feasible e.g. because the feature of interest is superimposed by a field that decays along its desired isosurface. However, sometimes it is possible to address these visualizations by ridge extraction. In short, ridges are lower-dimensional (elongated) regions of relatively high values.

The extraction of 1-dimensional ridges in $n$-space is easily accomplished by the *Parallel Vectors* method [23]. One of the advantages of this method is that explicit computation of eigenvectors is avoided and therefore the computational costs of the extraction are alleviated. However, the extraction of $n$-dimensional ridges with $n > 1$ can not be addressed by Parallel Vectors. These are the cases where the *Marching Ridges* method [6] is appropriate. Extracting ridges by field lines of the *Feature Flow Field* as done in [26] is a global operation and therefore not used here. This paper presents a method based on Marching Ridges that is applicable in situations where the underlying scalar field can be sampled during ridge extraction. Its strength shows up especially in cases where finely resolved ridges are desired, large regions do not exhibit ridges, or where the sampling of the scalar field is expensive.

The concept of Lagrangian coherent structures (LCS) is widely and increasingly used in fluid dynamics and in the analysis of the phase space of dynamical systems. There is no consensus what is meant by coherent structures, leading to different definitions. Some are restricted to vorticity such as that by Hussain [14], others are more general such as the one by Robinson [24] where coherent motion is defined as "a region over which at least one fundamental flow variable exhibits significant correlation with itself or with another variable over a range of space and/or time that is significantly larger than the smallest local scales". These definitions are referring to 3-dimensional regions. According to Haller [9], attracting and repelling Lagrangian coherent structures in (transient) vector fields are of lower dimension and tend to be the equivalent to unstable and stable manifolds (separatrices) in vector field topology [12, 1]: they separate regions of qualitatively different behavior and are also involved in mixing processes. Opposed to Newtonian coherent criteria such as the $Q$-criterion [13], the $\Delta$-criterion [4], and the $\lambda_2$-criterion [15], which are derived from the velocity gradient and therefore Galilean invariant, Lagrangian coherent structures are based on trajectories and even invariant to rotation of the frame of reference (they are objective). Because of their foundation on trajectories, LCS are insensitive to short-term perturbations or anomalies. In 2001 Haller has shown [9, 10] that LCS can be obtained as ridges in the largest direct Lyapunov exponent (DLE), also called largest finite-time Lyapunov exponent (FTLE). FTLE was defined by Lorenz in 1965 [20] and e.g. by Goldhirsch et al. in 1987 [8] for measuring predictability, see Yoden et al. [30] for details. For steady vector fields LCS is comparable to vector field topology [9], although it tends to convey more information [25]. However, LCS is still well defined and interpretable for unsteady vector fields due to its Lagrangian definition, whereas classical vector field topology is only able to give an instantaneous view, except for the approaches by Theisel et al. [29] and Shi et al. [27] based on path lines. LCS move and deform over time as the starting time of their trajectories is modified. They behave as material surfaces that get advected with the flow in a fluid dynamics view. LCS have played only a minor role in the field of scientific visualization until now. Two examples are direct visualization of 2-dimensional FTLE by Garth et al. [7] and the visualization of ridges in 3-dimensional FTLE by the authors [25].

Section 2 gives some background on ridge extraction and ridge filtering. Section 3 introduces the filtered adaptive mesh refinement (AMR) ridge extraction. Section 4 gives some background on (finite) Lyapunov exponents and builds on the approach of Haller [9] to identify LCS by extraction of ridges in FTLE, where FTLE is computed from the flow map as described in Section 4.1. It also presents a method to compute the finite-size Lyapunov exponent, and introduces a new finite Lyapunov exponent variant as well as the way how to compute it. In Section 5 the filtered AMR ridge extraction is applied to the different finite Lyapunov exponents for vector fields from practical CFD simulations, and interpretations are given. Finally, Section 6 concludes the work.

## 2 Ridges

Haralick [11], Eberly [5], and Lindeberg [18] proposed closely related definitions for $k$-dimensional *Height Ridges* in $n$-space. There are also other ridge concepts such as *Profile Ridges* and *Second Derivative Ridges* [21]. However, there are usually only minor differences in the results. Height Ridges can be seen as the most natural concept in most applications and therefore it is widely used such as in the case of Parallel Vectors [23] and Marching Ridges [6]. This is also the cause why it has been chosen for the filtered AMR ridge extraction (of finite Lyapunov exponents) in this paper.

Height Ridges can be seen as local maxima in a relaxed sense. They reside at locations where the scalar field $s$ exhibits a maximum in at least one direction. Generally, Height Ridges are $d$-dimensional man-

• *F. Sadlo and R. Peikert are with ETH Zurich, Switzerland. E-mail: {sadlo, peikert}@inf.ethz.ch.*

*Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.*

ifolds in $n$-dimensional space with $n > d \geq 0$. The constituent criteria can be formulated using the gradient and the Hessian of $s$. By definition, the eigenvectors $\boldsymbol{\varepsilon}_i$ corresponding to the $d$ largest eigenvalues $\lambda_i$ $(i = 1,\ldots,d)$ of the Hessian point along the ridge whereas the eigenvectors $\boldsymbol{\varepsilon}_j$ corresponding to the $(n-d)$ smallest eigenvalues $\lambda_j$ $(j = d+1,\ldots,n)$ are perpendicular to the ridge. According to the local maximum property, one necessary condition for a ridge is that the directional derivatives in $\boldsymbol{\varepsilon}_j$ directions are zero, formulated as

$$\boldsymbol{\varepsilon}_j \cdot \nabla s = 0. \tag{1}$$

The second condition for a local maximum and hence a Height Ridge is that the second directional derivatives in $\boldsymbol{\varepsilon}_j$ directions are negative, formulated as

$$\lambda_j < 0. \tag{2}$$

The same concept can be used to compute *Valley Lines* which is the opposite of Height Ridges. They can be obtained by extracting Height Ridges of the field $-s$. The reader is referred to the work of Haralick, Eberly, Lindeberg, and Majer [21] for further details.

### 2.1 Ridges in Discrete Data

As already mentioned, 1-dimensional ridges are preferably extracted from discrete data using the Parallel Vectors method, whereas $n$-dimensional ridges with $n > 1$ are preferably extracted using the Marching Ridges algorithm. Marching Ridges is similar to the family of *Marching Cubes* [19] algorithms. Marching Cubes is a cell-wise algorithm for generating isosurfaces, it generates a set of triangles for each cell of the grid. The so-called edge intersections are the positions on the edges of the cell where the scalar field has the desired value. The triangles are generated according to the edge intersections using a look-up table for signs of the values at the nodes. Criterion (1) is suited for being addressed by Marching Cubes. However, the fact that eigenvectors lack an orientation impedes a direct application, meaning that the evaluated directional derivatives can not be assumed to be consistent. Marching Ridges (and the presented algorithm) solves the problem by making the eigenvectors of a cell consistent using *Principal Component Analysis*. Once the orientations are made consistent, criterion (2) can be applied. If an edge intersection violates it, we do not generate the corresponding triangles. Another issue is the orientation of the resulting triangles. Kindlmann et al. [17] orientate the triangles in a post-processing pass. However, we experienced non-orientable manifolds in some cases of finite Lyapunov exponent ridges. This problem is addressed by appropriate rendering techniques such as two-sided normals.

### 2.2 Ridge Filtering

Here some filtering criteria for ridges are recapitulated, and discussed in the context of adaptive ridge extraction, that have already been presented in [25] for ridge extraction in FTLE fields.

Because ridges are extracted in this work using the Hessian of the scalar field, noise amplification can become an issue. A common way to handle it, is to apply smoothing prior to the evaluation of derivatives. One has to keep in mind however, that smoothing can deform the ridges, which can lead to e.g. instantaneous finite Lyapunov exponent ridges that are permeated by trajectories in contradiction to the theory. In our case, the gradient at a given node (of the possibly unstructured sampling grid) is computed by fitting a linear field to its neighboring nodes in a *Least Squares* sense. This allows for incorporating the smoothing into gradient computation by simply increasing the neighborhood range by a user-defined integer value. A value of 2 was used for the results in Section 5, which increased perceptibility and only led to negligible deviations.

Even with smoothing, the Marching Ridges method from Section 2.1 often yields more ridge regions than desired. This can be addressed by feature filtering.

One natural criterion for filtering ridge regions is to prescribe a minimum height of the ridge:

$$s \geq s_{min}. \tag{3}$$

In the case of finite Lyapunov exponent ridges, this equals to the prescription of a minimum separation and is therefore a straightforward choice. This way, ridges with low separation property are suppressed, leading to significant, consistent, and reliable visualizations. It is therefore our preferred method for filtering finite Lyapunov exponent ridges. The reader is referred to [25] for further details on the influence of this filtering criterion.

Another natural criterion for filtering ridge regions is to prescribe a maximum for the second derivative $\lambda_n$ across the ridge, which results in suppressing regions with too "flat" ridge property:

$$\lambda_n \leq \lambda_{max}. \tag{4}$$

In the case of finite Lyapunov exponent ridges, this is not as meaningful as criterion (3). Furthermore, it depends on the sampling of the scalar field, making it less suitable for guiding the subdivision of Section 3, but useful for post-processing, e.g. for further reducing the amount of ridges and hence occlusion.

Since ridge extraction often delivers small ridges which might be regarded as noise as long as the other filtering criteria did not disrupt the ridges due to low tolerance, another criterion is to prescribe a minimum size of the connected components of the resulting mesh. However, because this is a global criterion, it can not be used for the subdivision of Section 3, only for post-processing, mainly for reducing occlusion.

Finally, for the case of scalar fields derived from trajectories, such as Lyapunov exponents, another filtering criterion is to prescribe a minimum integration time for the trajectories that lead to a given ridge region. This allows to suppress ridge parts that are generated due to trajectories reaching the domain boundary. However, it was not needed for the results in Section 5.

The point-wise filtering conditions of this section are tested at the vertices of each generated triangle and it is rejected if at least one of the conditions is violated for at least one of its vertices. Triangle trimming was not implemented but would be a way to reduce zigzag ridge borders.

## 3 Filtered AMR Ridge Extraction

This section describes the method for filtered ridge extraction based on adaptive mesh refinement.

Sampling a scalar field at a given resolution uniformly, generating ridge elements from that field, and rejecting many of these elements in the end by filtering wastes a lot of computation time and space, especially if there are large regions that would exhibit no ridges at all even if no filtering was applied. One would like to sample the scalar field only in regions that exhibit ridges in the end, especially if the sampling of the scalar field is as costly as in the case of Lyapunov exponent computation. This would allow to end up with finer resolved ridges in the regions of interest with the same computational cost or even with lower cost. This constitutes the main motivation for incorporating adaptive refinement of the scalar mesh into the ridge extraction procedure.

The main goal of this method and a requirement for successful application in engineering is to obtain results that are identical to those obtained from uniform sampling at the prescribed finest subdivision resolution. This is especially valid for finite Lyapunov exponents since they depend on the sampling resolution (section 2.3 of [9] and Section 3.2 of this work). Although complete ridges may get missed in the presented method (because of too coarse initial sampling, because (4) was used instead of (3), or due to reasons mentioned in Section 3.2), the obtained ones are identical to those from uniform sampling due to the "ridge growing" presented in the next section. This is the reason why no measurements of accuracy are presented although they have been performed extensively during testing, where ridges were rarely missed. Part of the problem is solved by the "look ahead" procedure presented in the next section. After the algorithm has terminated and the result is obtained, it is possible to continue with the look ahead as a background process in order to guarantee that all ridges are captured.

Assuming sufficiently fine initial grids, a ridge does not move substantially during refinement, therefore we can start with a relatively

coarse mesh and use the ridges therein for the refinement process. The refinement is performed in an iterative manner. At each iteration, all cells that contain, are expected to contain, or adjoin to ridges are subdivided, but not more than to the current subdivision level. The current subdivision level starts with 1 and is increased with each iteration. This procedure makes sure that all cells at and in the neighborhood of ridges are subdivided to the same level. This has the advantage that the resulting grid has uniform subdivision level in ridge regions and therefore the Marching Ridges algorithm can be applied without the risk of producing cracks in the meshes, which would usually occur if a ridge extends over regions of different subdivision level. The next section describes the algorithm in more detail.

### 3.1 Algorithm

The following functions will be used in Algorithm 1:

- `detectRidgeCells(candidateCells, filter)`: Gradient, Hessian, and the eigenvalues and eigenvectors of the Hessian are computed at the nodes of the candidate cells. Then the intersected edges of `candidateCells` are determined in the sense of Marching Ridges, according to the Height Ridge criteria (1) and (2). If `filter` is *true*, the point-wise filtering criteria from Section 2.2 are also applied to the edge intersections. A cell is a *ridge cell* and returned in the set of ridge cells if at least one of the edges of the cell remains intersected.[1]

- `getNeighboringCells(cells, range, level)`: This function returns the neighboring cells that lie at least partially within the node-connected[2] `range` around `cells`, whereas `range` is measured at subdivision `level`. Cells with lower subdivision level $\sigma$ contribute a range of $2^{(\texttt{level}-\sigma)}$.

- `cellsForLevelDiff1(cells)`: This function returns cells that need to get subdivided additionally to `cells` so that neighboring cells differ at most in one subdivision level.

- `subdivideCells(cells)`: This function subdivides `cells`, adapts their subdivision level $\sigma$, and returns newly generated nodes and newly generated cells.

The explanations for Algorithm 1 are as follows:

i) **iterations** (Line 1): The total number of subdivision levels to be performed by the loop at line 12.

ii) **range** (Line 2): The neighborhood range around the ridge cells. Please refer to the discussions of (v) and (vi) below for further details.

iii) **laCellCnt** (Line 3): See look ahead step (viii) explained below.

iv) **laCriterion()** (Line 4): One of the point-wise filtering criteria of Section 2.2 has to be chosen for the look ahead procedure (viii). This method returns the value of the quantity tested by the corresponding criterion. If the ridge height criterion (3) is chosen, it returns the maximum of the scalar values at the nodes of the cell. If the second derivative criterion (4) is chosen, it returns the negative of the minimum $\lambda_n$ at the edge intersections of the cell.

v) **Add ridge cell neighbors** (Line 15): This maintains a uniformly subdivided band of width *range* around the filtered ridge cells which is needed for gradient and Hessian computation that is not affected by AMR (neighboring cells with lower subdivision level). See also Figure 1 (left). For results that are identical to those based on a uniformly subdivided grid, *range* has to be set to 2 times the gradient fitting neighborhood range of Section 2.2. However, experience has shown that it can often be set to much smaller values, leading to negligible deviations. This allows for a more efficient but approximate ridge extraction because it reduces the number of performed field evaluations. If the scalar field itself requires gradient computation, the corresponding gradient fitting neighborhood range should also be added to *range* (see Section 3.2 for details).

vi) **Add lower-level neighbors** (Line 26): Schedules lower subdivision-level neighbors of ridge cells for subdivision (see also Figure 1 (center)). This maintains the band of uniform subdivision level around the ridges as in (v) above and also allows the ridges to grow. The fact that not only neighbors of the new ridge cells $\mathscr{P}$ are tested but also neighbors of already existing ridge cells at finest subdivision level $\mathscr{Q}$, accounts for the case where *range* has been chosen smaller than required for exact results. In this case subdivided cells may get ridge cells only after subdivision of nearby cells.

**Algorithm 1** Filtered AMR Ridge Extraction

```
 1: iterations: number of subdivision iterations to perform
 2: range: user-defined integer neighborhood range
 3: laCellCnt: number of cell-subdivisions to look ahead
 4: laCriterion(): filter criterion used for look ahead
 5:
 6: Initialization: A coarse sampling grid is supplied by the user and
    the scalar field is sampled on this grid.
 7: for all cells c of sampling grid do
 8:    σ_c ← 0  // σ_c: subdivision level of cell c
 9: end for
10: ℛ ← detectRidgeCells(allCells,true)
11:
12: // iterate over subdivision levels
13: for it = 0 to iterations − 1 do
14:    𝒮 ← ℛ
15:    𝒮 ← 𝒮 ∪ getNeighboringCells(ℛ, range, it + 1)
16:    ℛ ← ∅
17:
18:    // subdivide / let ridges grow
19:    subIter ← 0
20:    while 𝒮 ≠ ∅ ∨ subIter = 0 do
21:       𝒮 ← 𝒮 ∪ cellsForLevelDiff1(𝒮)
22:       𝒩, 𝒞 ← subdivideCells(𝒮)
23:       𝒮 ← ∅
24:       sample scalar field at new nodes 𝒩
25:
26:       // add lower-level neighbors of ridge cells
27:       𝒫 ← detectRidgeCells(𝒞, true)
28:       𝒯 ← cells with σ = it + 1
29:       𝒬 ← detectRidgeCells(𝒯 \ 𝒞, true)
30:       ℳ ← getNeighboringCells(𝒫 ∪ 𝒬, range, it + 1)
31:       𝒮 ← 𝒮 ∪ cells in ℳ with σ < it + 1
32:       ℛ ← ℛ ∪ cells in (𝒫 ∪ 𝒬) with σ = it + 1
33:
34:       // test lower-level cells
35:       𝒯 ← cells with σ < it + 1
36:       𝒮 ← 𝒮 ∪ detectRidgeCells(𝒯, true)
37:
38:       // look ahead
39:       if laCellCnt > 0 ∧ (𝒫 ≠ ∅ ∨ subIter = 0) then
40:          𝒯 ← cells with σ < it + 1
41:          𝒫 ← detectRidgeCells(𝒯, false)
42:          R ← sort cells in 𝒫 by laCriterion()
43:          𝒮 ← 𝒮 ∪ laCellCnt highest cells in R
44:       end if
45:       subIter ← subIter + 1
46:    end while
47: end for
48:
49: generate ridge triangles from ℛ according to Section 2
```

[1] Defining a ridge cell by the presence of a generated triangle is not feasible because no triangles are generated if an edge intersection violates e.g. (2) which is often the case at the fronts of the current ridges, and this would not allow the ridges to grow during the refinement process.

[2] It has been chosen to use node-connected neighborhoods instead of face-connected neighborhoods because it is a better approximation to the support radius of second derivatives (Hessian) even for a `range` of 1.

vii) **Test lower-level cells** (Line 34): This tests cells that may have got ridge cells because of subdivision of nearby cells.

viii) **Look ahead** (Line 38): We look ahead for the cases where ridge components would get missed completely because they are too small or too faint, or because the initial sampling grid was chosen too coarse.[3] This can happen if no cell of a ridge satisfies the point-wise filtering criteria of Section 2.2 at a low subdivision level, but would satisfy them after further subdivision(s). The parameter *laCellCnt* prescribes the number of unfiltered ridge cells with maximum value of $laCriterion()$ that are subdivided (looked ahead) at each iteration even if they do not satisfy the filtering criteria. The parameter *laCellCnt* can be set to 0 if the user is only interested in the most prominent ridges.

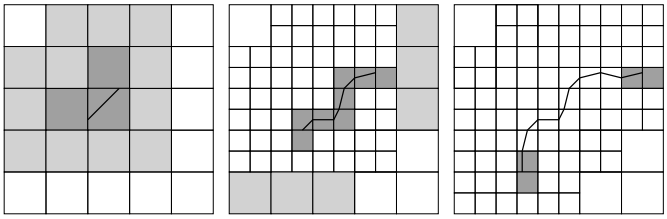


Fig. 1. Refinement iteration. Left: 3 ridge cells (dark gray) and their 12 neighbors (light gray) scheduled for subdivision. Center: 8 ridge cells (dark gray) and their 6 lower-level neighbors scheduled for subdivision (light gray). Right: 4 new ridge cells. No new lower-level neighbors, iteration proceeds to next subdivision level (left figure). Ridge is not computed during refinement, only drawn for illustration of edge intersections. Neighborhood *range* is 2.

### 3.2 Implications for Fields Based on Local Operators

Some issues arise if filtered AMR ridge extraction is applied to scalar fields that can not be evaluated in a strictly point-wise manner. One class of such fields are finite Lyapunov exponents where the computation is based on gradients of a map (see Section 4.1). A natural approach to the computation of such fields is to sample the underlying field (the map) at a grid that is identical to that of the scalar field. In that case, there are several implications. One aspect is that the support radius of that gradient has to be added to the *range* parameter of Section 3.1 for obtaining exact results as in the discussion of (v) in Section 3.1.

Another issue is that the scalar value may depend on the sampling, because e.g. gradients of non-linear fields vary with sampling resolution. It is therefore not possible to estimate the value at a finer resolution from the values at a lower resolution without any assumptions on the field. In the case of finite Lyapunov exponents, no satisfying assumption can be made. The variation of the value during refinement has implications on the ridge height criterion (3). There are two approaches to handle the problem: either use a lower threshold and apply the desired threshold as post-processing, or increase *laCellCnt*. In the worst case complete ridges may get missed, but thanks to the "ridge growing" procedure, the obtained ridges will be identical to those from a uniform sampling at finest subdivision level.

A further issue is that there might be nodes in the grid that are invalid. In the case of the map, this can be because the positions are outside the domain. At such nodes no gradient is computed and furthermore they are not used for gradient computation at neighboring nodes. It might also happen that there are not enough neighbors for computing gradients due to the restriction of the node neighborhood. This case has also to be handled appropriately. If the gradient can not be computed at a node for the mentioned reasons, the node of the scalar field is marked accordingly. During application of the filtered AMR ridge extraction method, cells containing such nodes are rejected from the ridge extraction process because they can not be handled by the Marching Cubes look-up table.

[3] In theory a minimum of one trajectory per region of different behavior is required, although small regions are often detected during subdivision due to the subdivision band around detected ridges.

## 4 Lyapunov Exponent

The Lyapunov exponent (LE), also called Lyapunov characteristic exponent, measures the exponential growth of an infinitesimal perturbation. It is often used to analyze the predictability of continuous dynamical systems or their sensitivity to initial conditions. An $n$-dimensional system, or vector field, has $n$ Lyapunov exponents and the largest Lyapunov exponent $\sigma_1(\mathbf{x})$ measures the maximum possible divergence of two nearby trajectories, starting in the neighborhood of $\mathbf{x}$. If it is positive, the trajectory is part of the unpredictable (chaotic) regime of the system, otherwise it belongs to a predictable region.

The largest Lyapunov exponent at position $\mathbf{x}$ is defined as

$$\sigma_1(\mathbf{x}) = \lim_{T\to\infty} \lim_{\|\boldsymbol{\delta}(t_0)\|\to 0} \frac{1}{|T|} \ln \frac{\|\boldsymbol{\delta}(t_0+T)\|}{\|\boldsymbol{\delta}(t_0)\|} \tag{5}$$

where $\boldsymbol{\delta}(t)$ is the perturbation at position $\mathbf{x}$ and time $t$. The initial perturbation has to be oriented in direction of maximum expansion.

The subsequent sections describe finite LE variants with implications regarding filtered AMR ridge extraction, additional to those mentioned in Section 3.2. Section 4.1 describes the FTLE variant and its computation according to Haller [9]. In Section 4.2 the FSLE variant is described and a method for computing it is presented that builds on the one used for FTLE. Finally, Section 4.3 proposes a new variant called FTLE Maximum and describes its computation.

### 4.1 Finite-Time Lyapunov Exponent

Equation (5) is an asymptotic measure in time and therefore a global quantity. However, there are several reasons for the need of a more local measure. One reason is that many vector fields have domain boundaries and therefore do not allow for infinite advection time. Besides computability another reason is that the LE is constant along a trajectory even if the local expansion rate varies along it.

According to Nese [22] the local divergence rate at time $t_i$ and for time step $\Delta t$ can be expressed as

$$\frac{1}{|\Delta t|} \ln \frac{\|\boldsymbol{\delta}(t_i+\Delta t)\|}{\|\boldsymbol{\delta}(t_i)\|}, \tag{6}$$

and the time average of (6) is the largest Lyapunov exponent

$$\sigma_1(\mathbf{x}) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n} \frac{1}{|\Delta t|} \ln \frac{\|\boldsymbol{\delta}(t_0+(k+1)\Delta t)\|}{\|\boldsymbol{\delta}(t_0+k\Delta t)\|}, \tag{7}$$

provided that $\boldsymbol{\delta}(t_0)$ is oriented in direction of maximum expansion. The local divergence rate (6) represents the largest finite-time Lyapunov exponent (FTLE) for $i=0$ and $\Delta t = T$. Nese was concerned with the information theory aspect and therefore used $\log_2$ instead of ln. He also mentioned that in practice, renormalization [3] has to be performed frequently along the trajectories. This addresses the fact that one has to make sure that the trajectories do not separate too much, otherwise they can not measure the expansion rates around either of the trajectories. In the case of renormalization, (7) follows only one of the two trajectories. This is numerically achieved by regular renormalization of the perturbation $\boldsymbol{\delta}(t_i)$ to the length $\|\boldsymbol{\delta}(t_0)\|$ but preserving its orientation. Haller [9] addresses the renormalization issue by stating that only finite-time phenomena are measured and hence requiring a dense enough sampling grid solves the problem. However, there are cases where even arbitrarily fine sampling is not able to produce the same results as with renormalization, e.g. in a flow that splits without shear. On the other hand, the objective of this paper is coherent structures and not predictability, and even coarse samplings are able to capture the large-scale behavior of a vector field.

Haller [9] proposed to base FTLE computation on the *flow map*. The flow map $\boldsymbol{\phi}_{t_0}^{t_0+T}(\mathbf{x})$ maps a sample point $\mathbf{x}$ to its advected position and is obtained by defining a sampling grid, seeding a particle at each node of the grid at time $t_0$, advecting the particles for time $T$, and storing the resulting positions at the nodes of the grid. We stop the integration of a trajectory if it reaches a domain boundary. One could

determine the direction of maximum expansion as the eigenvector belonging to the largest eigenvalue of

$$\Delta_{t_0}^{T}(\mathbf{x}) = (\nabla \boldsymbol{\phi}_{t_0}^{t_0+T}(\mathbf{x}))^{\top} \cdot \nabla \boldsymbol{\phi}_{t_0}^{t_0+T}(\mathbf{x}).$$

According to Haller [9] the maximum stretching factor can be obtained as the spectral norm of $\nabla \boldsymbol{\phi}_{t_0}^{t_0+T}(\mathbf{x})$, defined as the square root of the largest eigenvalue of $\Delta_{t_0}^{T}(\mathbf{x})$:

$$\sqrt{\lambda_{max}(\Delta_{t_0}^{T}(\mathbf{x}))} \tag{8}$$

and the largest FTLE is computed from (8) as follows:

$$\sigma_{t_o}^{T}(\mathbf{x}) = \frac{1}{|T|} \ln \sqrt{\lambda_{max}(\Delta_{t_0}^{T}(\mathbf{x}))}. \tag{9}$$

The reader is referred to [9] for further information on LCS and FTLE.

Some additional issues arise if the filtered AMR ridge extraction technique from Section 3 is to be used for ridge extraction in finite LE variants of unsteady vector fields. Because the subdivision procedure is based on the values of these fields, the complete (incremental) procedure of their evaluation has to be performed for all nodes at a given subdivision (and ridge-growth) iteration before being able to proceed to the next level. One implication is that this makes a path line integration method necessary, that is even efficient for depth-first integration of path lines. Instead of loading complete time steps, we reorganize the vector field data in a preprocessing step. A single file is generated that stores for each node the vectors of its time steps consecutively. The access to the data is performed by *mmap()*: the file is mapped to an address space of equal size and the paging subsystem makes sure that for each memory access the corresponding memory page is loaded, with possible read-ahead. This allows to exploit temporal coherency because a trajectory usually traverses multiple time steps before having passed a complete cell, and because several trajectories are likely to pass a given cell at different times when computing flow maps. If the file size exceeds the available address space in case of 32-bit systems, several files are generated instead, each storing only a fixed number of time steps. In this case, the high-level access routine used for integration has to make sure that the file containing the necessary time is mapped before accessing the data. This is easily accomplished and the overall performance does not noticeably degrade compared to the single-file approach because the corresponding parts of the files are cached by the soft disk cache of the operating system.

### 4.2 Finite-Size Lyapunov Exponent

Introduced by Aurell et al. [2], the finite-size Lyapunov exponent (FSLE) measures the shortest necessary time it takes for two infinitesimally close particles to separate by a given factor $s$. The motivation was to make the measure independent of the advection time $T$ because different regions of a system often require different choices of $T$. Aurell et al. computed the FSLE by advection of differently oriented particle pairs.

Here a formulation of the FSLE $\sigma_{t_o}^{s}(\mathbf{x})$ is presented that is based on the FTLE formulation of Haller (9):

$$\sigma_{t_o}^{s}(\mathbf{x}) = \frac{1}{|T_s|} \ln s,$$

with minimal $|T_s|$ such that

$$\sqrt{\lambda_{max}(\Delta_{t_0}^{T_s}(\mathbf{x}))} = s. \tag{10}$$

As with the formulation (9) for FTLE, this has the advantages that trajectories have to be integrated only for each node of the grid, and that the computed quantities do not substantially depend on the orientation of the seeding of the trajectories. However, in practice there is often no simple analytic solution to determine the $T_s$ that is necessary to achieve separation $s$, because $\Delta_{t_0}^{T_s}(\mathbf{x})$ is based on trajectories computed from discrete fields. Therefore it was chosen to compute FSLE in an incremental manner. This is achieved by increasing $T_s$ from 0 to a user-defined upper limit $T_{max}$ by $n$ time steps $\Delta t = T_{max}/n$, and each time computing the left-hand side of (10). The trajectories are computed incrementally. If the value gets larger than $s$, the corresponding advection time $T_s$, with linear interpolation in the last time step, is used for FSLE computation. The corresponding trajectories are not stopped however, because they are likely to be needed for the gradient computation at nearby trajectories later on.

Computation of the trajectories is the most expensive part in the computation of the left-hand side of (10). The computation of the flow map gradient, the eigenvalues, and the square root is much less expensive. Therefore one can afford to use a sufficiently high number of steps $n$ and hence perform these operations $n$ times.

Another issue shows up if filtered AMR ridge extraction is applied to finite LE variants that are computed incrementally such as FSLE and FTLEM (see section below). These variants are based on the incremental computation of the flow map and its gradient. As already mentioned at the end of Section 4.1, the procedure has to be repeated for each subdivision (and ridge-growth) iteration. For each step in the incremental computation, the flow map consists of intermediate positions of trajectories computed at previous subdivision levels, and the end points of the newly computed trajectories. Therefore all trajectories and not just the flow map have to be stored during filtered AMR ridge extraction.

### 4.3 Finite-Time Lyapunov Exponent Maximum

To further reduce the dependency on parameters, we propose a new variant of FTLE, the finite-time Lyapunov exponent maximum (FTLEM):

$$\hat{\sigma}_{t_0}^{T,n}(\mathbf{x}) = \max_{k=1,\ldots,n} \frac{1}{|k\Delta t|} \ln \frac{\|\boldsymbol{\delta}(t_0 + k\Delta t)\|}{\|\boldsymbol{\delta}(t_0)\|} \tag{11}$$

with $\Delta t = T/n$, computed in an incremental way, similarly to the FSLE computation of Section 4.2. The FTLE $\sigma_{t_0}^{k\Delta t}(\mathbf{x})$ is computed at each of the $n$ time steps according to (9) and its maximum is taken. The motivation for doing so is to avoid parameters that may require different choices for different regions of the vector field (which is to some extent still the case for FSLE), and to capture high expansions along the trajectory instead of only analyzing the final flow map. Taking the maximum of the largest FTLE is a quite straightforward decision: it still measures the maximum expansion. This FTLE variant comes in two flavors: with or without normalization. Equation (11) is the case with normalization and represents the true maximum of the FTLE along the trajectory. If the normalization $1/|k\Delta t|$ is omitted, it measures the (logarithm of) maximum stretching factor along the trajectory, which avoids high values of (11) at the beginning to dominate the result due to the normalization. However, this differs only from ridges of FTLE if trajectories converge later on.

### 4.4 Separation and Attachment Lines

Separation and attachment lines can be extracted locally as proposed by Kenwright et al. [16] but also globally using methods from vector field topology. However, both methods are only able to give an instantaneous view to separation and attachment processes. Furthermore, both are restricted to near-wall flow. One common way to inspect the interrelations with the interior flow is to seed stream surfaces from the separation lines or lines of attachment.

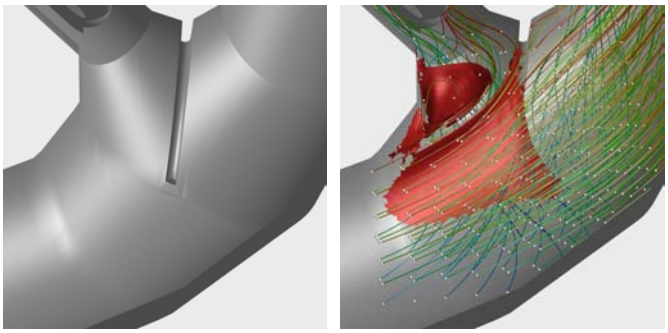
Due to their analogy to vector field topology, Lagrangian coherent structures based on ridges in finite Lyapunov exponents are suited to visualize both separation and attachment phenomena, and their interrelations with the internal flow in a time-dependent way. Attachment lines can be indicated by the curves where ridges of positive-time finite Lyapunov exponents attach to a boundary. Separation lines can be obtained the same way using negative-time finite Lyapunov exponents. If such a situation is observed, the ridges in question convey information about the process of attachment or separation.

## 5 RESULTS

This section presents some results obtained by filtered AMR ridge extraction from finite Lyapunov exponents. Several datasets are examined and the different finite LE variants are compared. A steady CFD simulation inside the distributor ring of a Pelton water turbine is examined and used for the comparison of the finite LE variants in Section 5.1. A unsteady CFD simulation of the intake of a hydropower plant is the target in Section 5.2, and the unsteady CFD simulation inside the diffusor of a Francis water turbine is examined in Section 5.3. We verified by also doing the non-adaptive ridge extraction that no filtered ridges were missed in the present examples, even without using the "look ahead". The obtained ridges deviate only slightly from those of uniform sampling because a *range* parameter of 2 was used instead of 5 (see discussion of choice of *range* in Section 3).

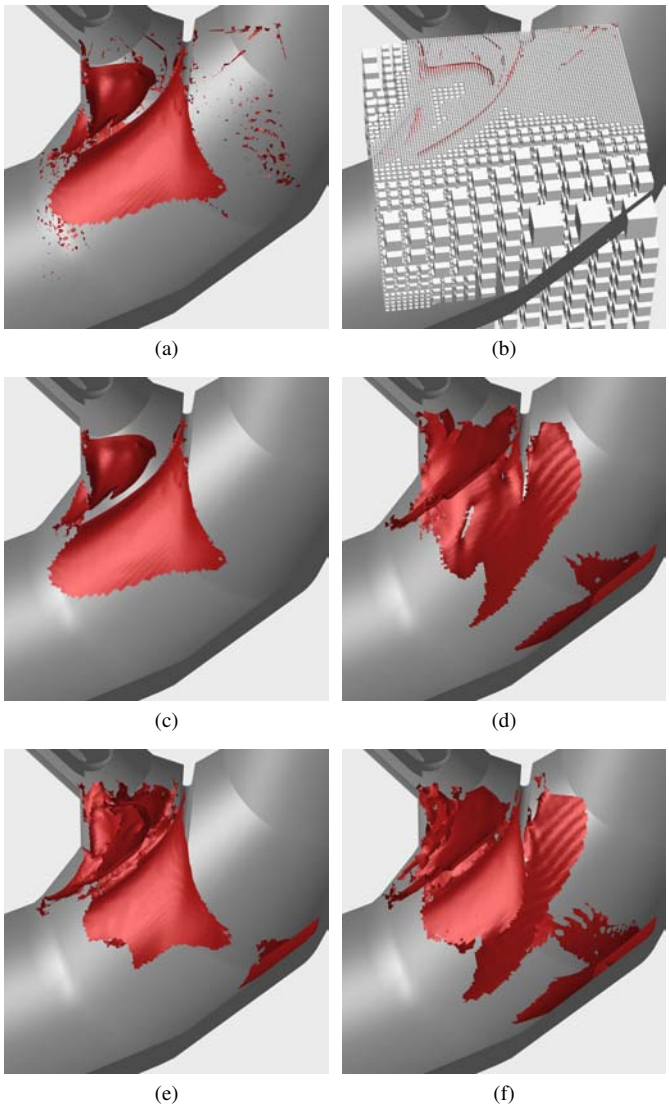
### 5.1 Pelton Turbine

Filtered AMR ridge extraction was applied to the different finite LE variants of the steady flow inside the distributor ring of a Pelton water turbine. The inspected region is in front of one of the constructs called sickle that bifurcate the flow into the injectors. The injectors produce the water jets that impel the runner of the turbine. Figure 2(a) shows the geometry at the second injector and Figure 2(b) shows the ridge resulting from the FTLE with some trajectories. The flow comes from the bottom left and continues to the upper right while part of it is bifurcated to the upper left. Figure 3 compares the results of the different positive-time finite LE variants and Table 1 gives some extraction details.



(a) (b)

Fig. 2. Pelton dataset at second injector of distributor ring. (a) Geometry. (b) Filtered AMR ridge extraction from FTLE (red, same as Figure 3(c)) with some trajectories (colored lines). Seeding points of trajectories visualized by white spheres.



(a) (b) (c) (d) (e) (f)

Fig. 3. Pelton dataset at second injector. Filtered AMR ridge extraction from different finite LE variants. See Table 1 for extraction details. (a) FTLE ridge without post-processing. (b) same as (a) with mesh. (c) FTLE ridge with post-processing. (d) FSLE with separation factor 1.5. (e) FSLE with separation factor 4. (f) FTLEM (similar to (d) and (e)).

Figure 3(a) shows ridges of FTLE without post-processing and Figure 3(b) shows additionally the corresponding mesh (the cells have been shrunk for visualization). The result has been post-processed by suppressing ridges that were smaller than 2000 triangles, visualized in Figure 3(c). The FTLE ridges visualize the bifurcation at the sickle and the recirculation zone at the top (see also Figure 2(b)). Regarding FSLE (Figures 3(d)–3(e)), one can see that the results depend on the choice of the prescribed separation factor (compare also Table 1). The ridges of the low-separation FSLE in Figure 3(d) visualize the near-wall flow whereas the ridges of the high-separation FSLE in Figure 3(e) mainly show the bifurcation at the sickle. The recirculation region is captured by both FSLE examples, but not as well as by the FTLE example of Figure 3(c). Figure 3(f) shows the advantage of FTLEM over FSLE: it is capable of visualizing both the phenomena of Figure 3(d) and Figure 3(e) without the need for finding appropriate FSLE separation factors.

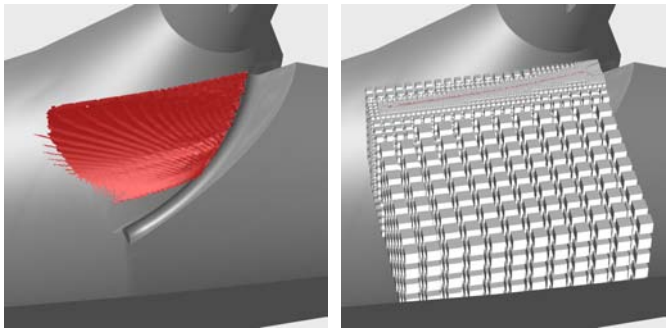
The efficiency gain of filtered AMR ridge extraction over direct ridge extraction from a uniformly sampled field at corresponding resolution was only about 1.4 for the FTLE in the inspected region. This is because the result contains a lot of ridge regions in the region of interest. However, many applications do not exhibit that dense Lagrangian coherent structures. In order to show the efficiency gain, we examined a region in front of the first injector of the distributor ring (see Figure 4 and Table 2). There is only a single filtered ridge in this region and therefore the efficiency gain is $> 4$ at the fourth subdivision level. The efficiency gain would further increase with an increase of the number of subdivision iterations or with longer advection times (this test was not performed because the FTLE computation on the uniform grid already took a considerable amount of time).

### 5.2 Hydropower Plant Intake

The underlying data of this section is a unsteady CFD simulation in the intake of a hydropower plant [28] (Figure 5(a)). Filtered AMR ridge extraction was applied to negative-time FTLE in a region of interest. Ridge regions with FTLE $< 0.02$ or Hessian $\lambda_n > -10$ were suppressed and small ridge components were rejected, see Figure 6. One can see FTLE ridges that wind around the vortex core lines, both in water and in the air. Another FTLE ridge is consistent with the water/air interface.

Table 1. Extraction details for Pelton example. The parameters below the second horizontal line were not used for mesh refinement, only for post-processing.

| | FTLE | FSLE | FSLE | FTLEM |
|---|---|---|---|---|
| initial grid (nodes) | 7x7x5 | 7x7x5 | 7x7x5 | 7x7x5 |
| final grid (nodes) | 157204 | 239551 | 215491 | 244651 |
| integration time [s] | 0.1 | 0.1 | 0.1 | 0.1 |
| sep. factor $s$ | – | 1.5 | 4 | – |
| $1/\|T\|$ | yes | yes | yes | yes |
| min. scalar $s_{min}$ | 23 | 50 | 14 | 22 |
| gradient range | 2 | 2 | 2 | 2 |
| ridge `range` | 2 | 2 | 2 | 2 |
| iterations | 4 | 4 | 4 | 4 |
| Hess. $\lambda_{max}$ | 0 | -150000 | -30000 | -50000 |
| min. triangle cnt. | 2000 | 2000 | 2000 | 2000 |
| Figure | 3(c) | 3(d) | 3(e) | 3(f) |



(a) (b)

Fig. 4. Pelton dataset at first injector, see Table 2 for performance details. (a) Filtered AMR ridge extraction for FTLE. The ripples are due to high FTLE values along the ridge and its approximate alignment with the sampling grid. This is an inherent problem with discretized FTLE computation. (b) Together with adaptive mesh.

### 5.3 Francis Turbine

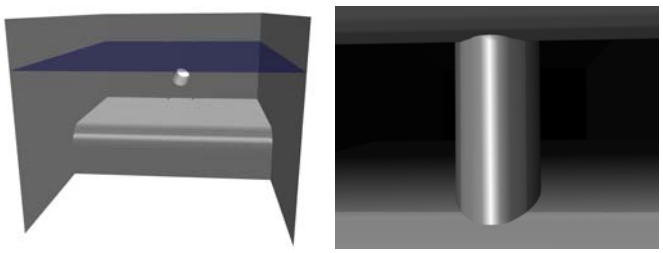
This section examines the flow of a unsteady CFD simulation in the draft tube of a Francis water turbine. More precisely, the region in front of a construct named divider, that bifurcates the flow into the two channels (Figure 5(b)). Filtered AMR ridge extraction of positive-time FTLE resulted in several ridges, some of them interacting with the vortices over time. Ridge regions with FTLE $< 10$ were rejected and only the large ridge components were used for visualization (Figure 7). One FTLE ridge winds around the vortex. This is consistent with the notion that vortices are coherent structures. The same ridge also visualizes the bifurcation at the top of the divider.

## 6 Conclusion

Filtered AMR ridge extraction was introduced as a means of efficient ridge extraction in situations where the underlying scalar field exhibits only few relevant ridges or can be sampled during ridge extraction. Although rarely some ridges may be missed, the obtained ridges are identical to those from a uniform sampling at finest resolution level. The method has been applied to different variants of finite Lyapunov exponents in vector fields from CFD simulations. A method for computing the existing FSLE variant was proposed and a new Lyapunov exponent variant was introduced. A goal of studying this new variant was to reduce the dependency on parameters such as the integration time or a prescribed separation factor. Based on our comparison, we believe that the FTLEM variant can be an interesting alternative to the existing FTLE or FSLE concepts.

Table 2. Performance analysis for Pelton dataset at first injector. Four iterations of filtered AMR ridge extraction from FTLE compared to direct computation on corresponding uniform grid. Achieved speed-up factor is $> 4$. See also Figure 4.

| | direct | adaptive |
|---|---|---|
| initial grid | 193x193x97 (3613153 nodes) | 13x13x7 (1183 nodes) |
| final grid | 193x193x97 (3613153 nodes) | 298964 nodes |
| flow map [s] | 19953.51 | 2350.21 |
| FTLE [s] | 10.73 | 30.73 |
| ridge extr. [s] | 278.46 | 2337.16 |
| total [s] | 20242.74 | 4930.72 |



(a) (b)

Fig. 5. (a) Hydropower plant intake. Water enters the region from the bottom and flows to the intake of the power plant visible as hole in the back wall. Air enters the region at the top front and leaves the region at the top. Water/air interface by isosurface (transparent blue). (b) Draft tube of Francis turbine in front of divider. The flow is bifurcated at the divider into the two channels.

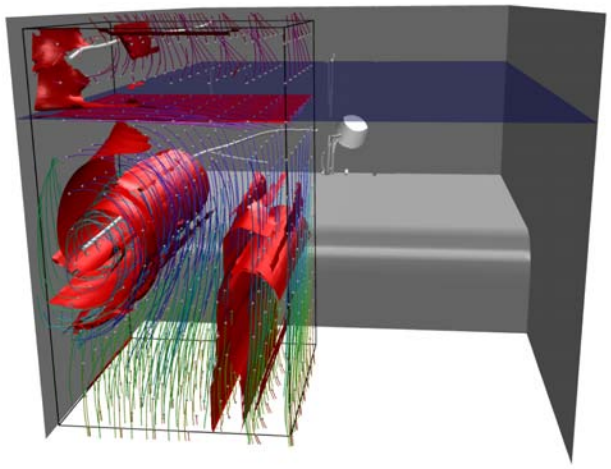


Fig. 6. Hydropower plant intake. FTLE ridges (red) with some upstream trajectories (colored lines) and vortex core lines (white). Seeding points of trajectories visualized by spheres (white), and water/air interface by isosurface (transparent blue).

## Acknowledgements

The authors would like to thank VATECH HYDRO for the CFD simulations of the Pelton and Francis turbine and France Suerich-Gulick for the CFD simulation of the hydropower plant intake. This work was funded by Swiss Commission for Technology and Innovation grant 7338.2 ESPP-ES.

## References

[1] D. Asimov. Notes on the topology of vector fields and flows. Technical Report RNR-93-003, NASA Ames Research Center, 1993.

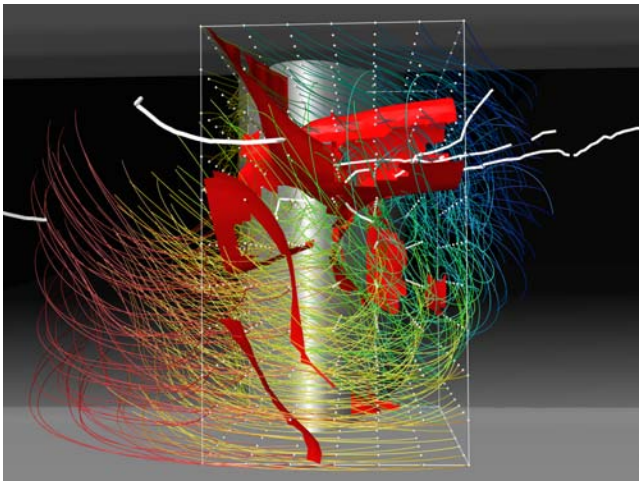


Fig. 7. Filtered AMR ridge extraction from FTLE (red) in draft tube of Francis turbine, some trajectories (colored lines), their seeds (white spheres), and vortex core lines (white tubes). One of the ridges winds around vortex in front of the divider.


[2] E. Aurell, G. Boffetta, A. Crisanti, G. Paladin, and A. Vulpiani. Predictability in the large: an extension of the concept of lyapunov exponent. *J. Phys. A: Math. Gen*, 30:1–26, 1997.

[3] G. Benettin, L. Galgani, A. Giorgilli, and J. M. Strelcyn. Lyapunov characteristic exponent for smooth dynamical systems and hamiltonian systems; a method for computing all of them. *Mechanica*, 15:9–20, 1980.

[4] M. Chong, A. Perry, and B. Cantwell. A general classification of three-dimensional flow fields. *Phys. Fluids*, 1990.

[5] D. Eberly. *Ridges in Image and Data Analysis. Computational Imaging and Vision.* Kluwer Academic Publishers, 1996.

[6] J. D. Furst and S. M. Pizer. Marching ridges. In *2001 IASTED International Conference on Signal and Image Processing*, 2001.

[7] C. Garth, G.-S. Li, X. Tricoche, and C. Hansen. Interactive visualization of coherent structures in transient flows. *Topology-based Methods in Mathematics + Visualization*, accepted for publication, 2007.

[8] I. Goldhirsch, P.-L. Sulem, and S. a. Orszag. Stability and lyapunov stability of dynamical systems: A differential approach and a numerical method. *Physica D*, 27:311–337, 1987.

[9] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D*, 149:248–277, 2001.

[10] G. Haller. Lagrangian coherent structures from approximate velocity data. *Phys. Fluids*, 2002.

[11] R. Haralick. Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing*, 22(10):28–38, 2001.

[12] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, 1989.

[13] J. Hunt, A. Wray, and P. Moin. Eddies, stream, and convergence zones in turbulent flows. Technical Report CTR-S88, Center for Turbulence Research, NASA Ames Research Center and Stanford University, 1988.

[14] F. Hussain. Coherent structures and turbulence. *Journal of Fluid Mechanics*, 173:303–356, 1986.

[15] J. Jeong and F. Hussain. On the identification of a vortex. *Journal of Fluid Mechanics*, 285(69):69–94, 1995.

[16] D. N. Kenwright, C. Henze, and C. Levit. Feature extraction of separation and attachment lines. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):135–144, 1999.

[17] G. Kindlmann, X. Tricoche, and C.-F. Westin. Anisotropy creases delineate white matter structure in diffusion tensor MRI. In *Ninth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'06)*, Lecture Notes in Computer Science 3749, Copenhagen, Denmark, October 2006.

[18] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *Intern. Journal of Computer Vision*, 30(2):77–116, 1996.

[19] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, volume 21, pages 163–169, July 1987.

[20] E. N. Lorenz. A study of the predictability if a 28-variable atmospheric model. *Tellus*, 17:321–333, 1965.

[21] P. Majer. *A Statistical Approach to Feature Detection and Scale Selection in Images*. PhD thesis, University of Goettingen, 2000.

[22] J. M. Nese. Quantifying local predictability in phase space. *Physica D*, 35:237–250, 1989.

[23] R. Peikert and M. Roth. The parallel vectors operator: a vector field visualization primitive. In *Proceedings of IEEE Visualization*, pages 263–270, 1999.

[24] S. K. Robinson. Coherent motions in the turbulent boundary layer. *Annu. Rev. Fluid Mech.*, 23:601–639, 1991.

[25] F. Sadlo and R. Peikert. Visualizing lagrangian coherent structures and comparison to vector field topology. *Topology-based Methods in Mathematics + Visualization*, accepted for publication, 2007.

[26] J. Sahner, T. Weinkauf, and H.-C. Hege. Galilean invariant extraction and iconic representation of vortex core lines. In *Proc. Symposium on Visualization (EuroVis '05)*, pages 151–160, 2005.

[27] K. Shi, H. Theisel, T. Weinkauf, H. Hauser, H.-C. Hege, and H.-P. Seidel. Path line oriented topology for periodic 2D time-dependent vector fields. In *Proc. Symposium on Visualization (EuroVis '06)*, pages 139–146, 2006.

[28] F. Suerich-Gulick, S. Gaskin, M. Villeneuve, G. Holder, and E. Parkinson. Experimental and numerical analysis of free surface vortices at a hydropower intake. In *Proceedings of the 7th International Conference on Hydroscience and Engineering*, pages 1–11, September 2006.

[29] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Stream line and path line oriented topology for 2d time-dependent vector fields. In *IEEE Visualization*, pages 321–328, 2004.

[30] S. Yoden and M. Nomura. Finite-time lyapuov stability analysis and its application to atmospheric predictability. *Journal of the Atmospheric Sciences*, 50(11):1531–1543, 1993.