Interactive Separating Streak Surfaces

Florian Ferstl, Kai Bürger, Holger Theisel, and Rüdiger Westermann

Fig. 1. Our method generates streak surfaces revealing separating structures in unsteady 3D flow interactively on the GPU. Red surfaces in the figures depict generalized streak surfaces emanating from 1D FTLE ridges on a planar seeding probe. Green particles show points on time surfaces, which are additionally released from the seeding plane into the flow and serve as context information.

Abstract—Streak surfaces are among the most important features to support 3D unsteady flow exploration, but they are also among the computationally most demanding. Furthermore, to enable a feature driven analysis of the flow, one is mainly interested in streak surfaces that show separation profiles and thus detect unstable manifolds in the flow. The computation of such separation surfaces requires to place seeding structures at the separation locations and to let the structures move correspondingly to these locations in the unsteady flow. Since only little knowledge exists about the time evolution of separating streak surfaces, at this time, an automated exploration of 3D unsteady flows using such surfaces is not feasible. Therefore, in this paper we present an interactive approach for the visual analysis of separating streak surfaces. Our method draws upon recent work on the extraction of Lagrangian coherent structures (LCS) and the real-time visualization of streak surfaces on the GPU. We propose an interactive technique for computing ridges in the finite time Lyapunov exponent (FTLE) field at each time step, and we use these ridges as seeding structures to track streak surfaces in the time-varying flow. By showing separation surfaces in combination with particle trajectories, and by letting the user interactively change seeding parameters such as particle density and position, visually guided exploration of separation profiles in 3D is provided. To the best of our knowledge, this is the first time that the reconstruction and display of semantic separable surfaces in 3D unsteady flows can be performed interactively, giving rise to new possibilities for gaining insight into complex flow phenomena.

Index Terms—Unsteady flow visualization, feature extraction, streak surface generation, GPUs.

1 Introduction

For the visual analysis of flow data, feature extraction methods are a well-established class of techniques because the extraction of features offers insight into different flow phenomena while reducing the amount of data to be processed. Another important and well-established class of visualization algorithms are real-time interactive exploration approaches, such as interactively seeding and tracking particles, characteristic lines, or integral surfaces. The increasing amount and complexity of flow data brings limitations to both classes of techniques: the features themselves may become so complex that their visual representation becomes challenging. On the other hand, interactive exploration techniques suffer from the danger that important phenomena are missed because certain areas are not explored. A solution for this is a combination of feature extraction and interactive exploration: either the extracted complex features are visualized by appropriate real-time flow exploration tools, or the seeding in interactive flow exploration is controlled by a feature extraction approach. In this paper, we propose such a combination for particular features (ridges of FTLE fields) and interactive exploration tools (generalized streak surfaces).

Ridges of finite time Lyapunov exponent (FTLE) fields are well-established features for computing separating structures in time-dependent flows. While their definition is well-understood, for 3D time-dependent flows their visualization is complicated because the ridges of interest are 3D hypervolumes in the 4D space-time domain, i.e., surface structures changing their shapes and appearance over time. Because of this, existing algorithms in 3D carefully focus on particular times and locations to show ridge surfaces, making a systematic exploration of all FTLE ridges in 3D time-dependent flows a time-consuming process.

Streak surface extraction is a prominent tool for interactive flow exploration. Since for every location in the space-time domain there is a one-parametric family of streak lines passing through, the sheer amount of existing streak lines (and therefore streak surfaces as well) leave the chance of missing interesting and important streak surfaces.

The approach presented in this paper aims in overcoming the drawbacks of both FTLE ridges and interactive streak surface exploration. It is justified by the following observation: FTLE ridges are approximately material structures [11, 44] and can therefore be interpreted as generalized streak surfaces (for 2D flows, [33] exploit this fact by considering the intersection of forward and reverse-time FTLE LCS).

We use this for the following algorithm: given a 3D time-dependent flow field, we interactively place and move a planar seeding structure $\mathbf{s}$ (usually a rectangle) in the flow domain at a certain time t. Note that moving the seeding structure $\mathbf{s}$ is possible both in space and time. We consider the restriction of the FTLE field on $\mathbf{s}$ (i.e., a 2D field), either by computing the FTLE values on $\mathbf{s}$ in-turn or by computing the entire FTLE field (i.e., a 4D scalar field) in a preprocess and resampling the values onto $\mathbf{s}$ via interpolation. We then extract ridge structures in the

- *F. Ferstl (E-mail: ferstlf@in.tum.de), K. Bürger (E-mail: buerger@tum.de) and R. Westermann (E-mail: westermann@tum.de) are with the Computer Graphics & Visualization group, Technische Universität München.*
- *Holger Theisel (E-mail: theisel@isg.cs.uni-magdeburg.de) is with the Visual Computing group, University of Magdeburg.*

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

FTLE field on **s** in real-time, and employ them as seeding structures for a streak surface integration. Since the ridges on **s** change their shape by moving **s** in space and time, the surfaces generated this way are generalized streak surfaces (an extension of the concept of generalized streak lines [52]). The streak surfaces are shown only for the integration time which was used for computing the FTLE, since only for this integration time a separation was detected. As our choice of seeding locations for streak surface integration aims to uncover separation structures, we will refer to them as separating streak surfaces in the following.

Our method exploits the fact that 2D FTLE ridges (LCS) in 3-space are advected in a similar way to streak surfaces seeded at 1D FTLE ridges on a 2D manifold in 3-space. This statement is based on two facts: firstly, the 1D FTLE ridges on the seeding plane are approximately on 2D ridges in 3-space as long as the seeding plane is approximately perpendicular to the flow (this was exploited in [7], where ridges on cutting planes are considered instead of 2D ridges). Secondly, FTLE ridges are approximately material structures and do in fact converge to exact material structures if the integration time goes to infinity [44].

In [32] and [23] this temporal coherence of LCS was exploited to efficiently compute time series of FTLE ridges via simultaneous advection of a sampling grid and incremental 1D ridge tracking, respectively. Since a finite integration time is used in our work, the generalized streak surfaces we extract do not coincide with 2D FTLE ridges in general. However, we will demonstrate in this work that these surfaces resemble the 2D ridges at high fidelity and can be computed very efficiently. Furthermore, since generalized streak surfaces move according to the flow they provide a more intuitive flow exploration metaphor than 2D FTLE ridges. Notably, no visual information will be generated in regions where many 2D ridges exist but for none of them an approximating streak surface has its origin on the selected seeding structure. This allows using our approach as an effective technique to focus on particular flow structures in space and time.

Our contributions: In this work we present the first approach to construct separating streak surfaces in 3D unsteady flows at interactive rates. This enables visually guided 3D flow exploration based on the concept of LCS. Our approach distinguishes from previous approaches in that it avoids computing LCS in 3D. Instead, the computation is restricted to a 2D manifold and streak surfaces are constructed at significantly less computational effort. All processing stages of the proposed algorithm are realized on the GPU, including FTLE computation, ridge extraction, streak surface reconstruction, and surface rendering. The specific contributions of our work are:

- A navigation tool that allows placing a 2D sampling grid in space-time and computing FTLE values on it in an interactive way.
- A new ridge extraction method that is specifically tailored to the GPU and produces ridges well-suited as seeding structures.
- A new method for the reconstruction and rendering of high-quality streak surfaces emanating from 1D FTLE ridges.

The remainder of this paper is organized as follows: After reviewing previous work that is related to ours, Section 3 is dedicated to the spatial selection of separating streak surfaces based on the FTLE. In Section 4 we introduce our new ridge extraction algorithm. The reconstruction of streak surfaces from extracted 1D FTLE ridges is described in Section 5. Section 6 presents a detailed analysis of our approach with respect to performance and quality. We conclude the paper with some thoughts on future work.

2 Related work

Our approach is based on a number of established techniques in visualization, namely FTLE computation, ridge extraction, streak surface integration, and interactive flow exploration. A thorough overview of feature extraction techniques in flow visualization and geometric flow visualization techniques can be found in [28] and [24], respectively.

FTLE

Lagrangian coherent structures as ridges of FTLE fields were introduced by Haller [11, 13] and experienced an intensive research since then [21, 12, 45, 51]. Shadden [44] has shown that ridges of FTLE are approximate material structures, i.e., they converge to material structures for increasing integration times. This fact was used in [33] to extract topology-like structures and in [23] and [30, 7] to accelerate the FTLE computation in 2D and 3D flows.

In the visualization community, different approaches have been proposed to increase the performance, accuracy and usefulness of FTLE as a visualization tool. For example, volume rendering and slicing techniques were used for a visual analysis of 3D FTLE fields in [9, 7], [31] proposed to extract LCS as filtered height ridges and compared LCS- and topology-based flow visualizations and [3, 7] considered the FTLE to control the visualization of particles to show divergent regions in 3D flows. None of them is designed for a real-time exploration of the separating structures in 3D space and time.

Ridge Extraction

To extract ridge structures, a variety of different approaches has been proposed in the literature. We mention local conditions by relaxing conditions of extremal structures [5, 22], topological/watershed approaches [34], definitions based on extremal curvature structures, adaptive methods [30], or particle based methods [17]. [27, 43] focus on the extraction of ridge surfaces in 3D fields. To the best of our knowledge, none of these approaches aims in a real-time extraction of ridge structures in time-varying fields.

Streak Surfaces

Streak surfaces have recently moved into the focus of flow visualization. [50] introduced non-adaptive smoke surfaces in an attempt to mimic the appearance of real-world flow structures. [19] presented remeshing techniques and conditions to get high-quality streak surfaces for flow fields on irregular grids without real-time constraints. [2] proposed interactive GPU based visualizations of streak surfaces for flow fields on regular grids. [52] introduced generalized streak lines which are obtained by moving the seeding point according to the zeros of the flow on a boundary surface. In addition, there exists a variety of approaches for the integration of stream and path surfaces [15, 48, 38, 49, 42, 36, 8].

Interactive Flow Exploration

Interactive flow exploration heavily relies on interactive frame rates and is therefore often concerned with efficient solutions on graphics hardware. On recent GPUs it is now possible to interactively trace millions of particles in Cartesian grids using higher order integration schemes [40, 46, 20, 4], and to instantly render these particles using a multitude of different options including oriented point sprites, lines, and more complex geometric representations like bands and tubes [10, 41, 18, 25]. For a detailed description of an efficient GPU implementation of particle tracing in tetrahedral meshes we refer the reader to the work by Schirski and Kuhlen [39].

3 FTLE

Our approach for visualizing separating streak surfaces is based on seeding particles along Lagrangian coherent structures (LCS) in a 3D unsteady flow field. Since LCS are formed by ridges in the finite time Lyapunov exponent (FTLE) field, the FTLE first has to be computed before meaningful seeding structures can be found.

The FTLE is a scalar quantity that measures the stretching induced by the flow. Let $\phi_{t_0}^T(x)$ denote the flow map that defines the mapping of particles at position $\mathbf{x}$ in space and t_0 in time via path line integration over the time interval T. According to [11] the FTLE is then defined as

$$\sigma_{t_0}^T(x) = \frac{1}{|T|} \ln \sqrt{\lambda_{max}\left((\nabla \phi_{t_0}^T(x))^\top \cdot \nabla \phi_{t_0}^T(x)\right)}$$

where λ_{max} is the largest eigenvalue of the right Cauchy-Green deformation tensor $(\nabla \phi_{t_0}^T(x))^\top \cdot \nabla \phi_{t_0}^T(x)$ of the flow map. Intuitively the

FTLE can also be seen as a value derived from the spectral norm of the flow map gradient, describing the maximum rate of separation of infinitesimally closely seeded particles.

In this work we compute the flow map, and the FTLE derived thereof, by sampling particles on a planar seeding structure $\mathbf{s}$, which is discretized by a uniform 2D sampling grid. For estimating the flow deformation in the vicinity of one of these particles, however, we consider additional particles that are seeded within an ε-region around it. Following [16], we have chosen ε according to the grid spacing in all of our examples.

The FTLE computation, and thus the following ridge extraction, is restricted to a sub-domain of the 3D flow domain. The user is provided a navigation tool to place $\mathbf{s}$ in the 3D field. Both the size and the resolution of the sampling grid can be set by the user. At the center of each grid cell the FTLE value is computed as described above. Specifically, if a center is at position (x,y,z), the trajectories of 6 particles seeded at positions $(x\pm\varepsilon, y\pm\varepsilon, z\pm\varepsilon)$ are traced and the deformation gradient is computed from the particle destinations. Particle tracing and FTLE computation is entirely performed on the GPU, and the resulting values are written into a 2D texture.

It should be noted that the FTLE computation we perform can generate less reliable results, since the particles can separate significantly during path line integration. Even though there exist approaches to overcome this problem, e.g. by a FTLE redefinition to local criteria on the center trajectory [16] or renormalization of the particle neighborhood [1], we have not yet integrated these approaches into our method.

Figure 2 shows two snapshots of an exploration session in which FTLE values have been computed on different sampling grids. In both cases the computation was performed at a grid size of 256×256 with an integration time of 0.15 s (100 Runge-Kutta 4th order integration steps, requiring 20 time steps of the unsteady flow field). Since the computation is performed on the GPU, interactive update rates of less than 150 ms are achieved as long as all time steps can be stored in the GPU memory.

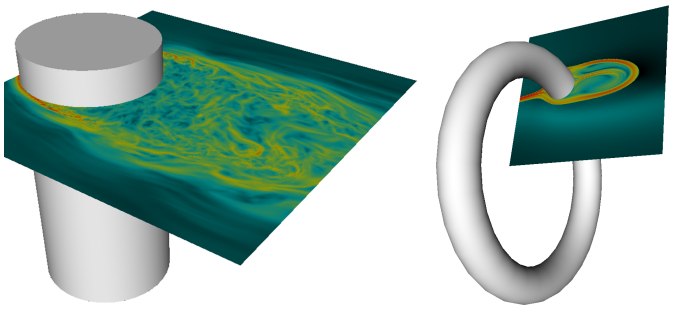


Fig. 2. Two FTLE fields on a planar probe at grid size 256×256. The parallelized FTLE computation on the GPU yields interactive update rates (less than 150 ms for the given probe texture resolution).

This indicates that often it is not necessary to pre-compute the 3D FTLE field. The values can be updated in-turn once the user moves $\mathbf{s}$ or changes any of the parameters the FTLE depends upon, like the start time t_0, the integration time T, the spatial sampling distance ε, or the size and resolution of the sampling grid. However, in scenarios where the time-resolved flow field sequence does not fit into GPU memory, pre-computing the time-dependent FTLE should be preferred. In this case the pre-computed FTLE values can simply be interpolated at the grid cell centers. Note that in this case the FTLE parameters are fixed and, therefore, ε might significantly differ from the grid spacing on $\mathbf{s}$.

4 FTLE Ridge Extraction

In the following we describe our novel extraction technique for 1D ridges in 2D FTLE fields. Ridge extraction techniques—also in the context of LCS extraction—have been studied extensively over the last years. There is a vast body of literature related to this field and a comprehensive review is beyond the scope of this paper. However, Eberly [5], Haralick [14], and Peikert and Sadlo [27] discuss the basic principles underlying such techniques as well as the different ridge types that exist, and they provide many useful algorithmic and implementation specific details.

Our ridge extraction technique builds upon the concepts of *height ridges* and *watersheds*. The definition of height ridges involves pointwise evaluations of algebraic equations based on geometric ridge properties, which are expressed via first-order derivatives and derivatives into the main curvature directions, i.e., the (transversal) ridge directions [5]. Let $f(\mathbf{p})$ denote the FTLE value at a point $\mathbf{p}$ on $\mathbf{s}$, and let H and $\mathbf{g}$ denote the Hessian matrix and the gradient of f, respectively. According to [27] the height ridges are a subset of the zero-contour of $\det(H\mathbf{g},\mathbf{g})=0$, which can be extracted in 2D using the marching squares algorithm.

In general, *unfiltered* height ridges do not provide suitable seeding regions for streak surfaces. Even though height ridges cannot really branch as shown in [43], they tend to appear as branched structures at larger scales. Highly branched and fragmented structures, however, result in many separate and even non-manifold surface parts. From a visualization point of view, the streak surfaces constructed from such ridges do not allow any intuitive interpretation due to their complex topology and visual clutter thereof. Figure 3 shows a set of unfiltered FTLE height ridges (top) and compares them to the ridges we are interested in (bottom). Even though height ridges can be post-processed to eliminate undesirable ridge parts and thus to yield simpler seeding structures [30], it seems to be difficult to implement this process efficiently on the GPU.

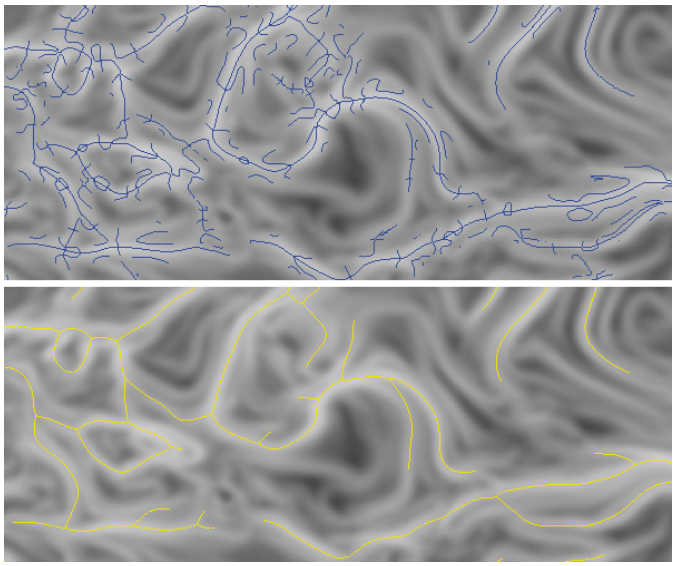


Fig. 3. Top: unfiltered height ridges; Bottom: ridges extracted by our approach.

Watersheds [29] are another popular approach for ridge extraction in 2D. It is based on the topology of the underlying 2D scalar field and aims at extracting slopelines separating hills and basins. In this definition a ridge is considered a slopeline going from one maximum to another maximum through a single saddle point. Even though the topology of watershed ridges is often much simpler than that of height ridges, they nevertheless fail to focus on the main axis of hills of the height field. Using a general watershed approach can also lead to rather cumbersome special cases in which significant ridges are missed because they do not separate different minima correctly.

To overcome the limitations of height ridges and watersheds we introduce a novel ridge extraction algorithm. Generally speaking, a ridge is a graph $G=(V,E)$ consisting of a set V of vertices and a set E of edges. Vertices $\mathbf{v}\in V$ can be end points ($deg(\mathbf{v})=1$), line points ($deg(\mathbf{v})=2$) or crossings ($deg(\mathbf{v})>2$). With respect to this definition, for our purpose the specific goals are a) to minimize the number of crossings per ridge, and thus to avoid non-manifold surfaces, and b) to maximize the ridge length, i.e. to connect as many vertices as possible, and thus to prevent the streak surfaces from falling into many parts.

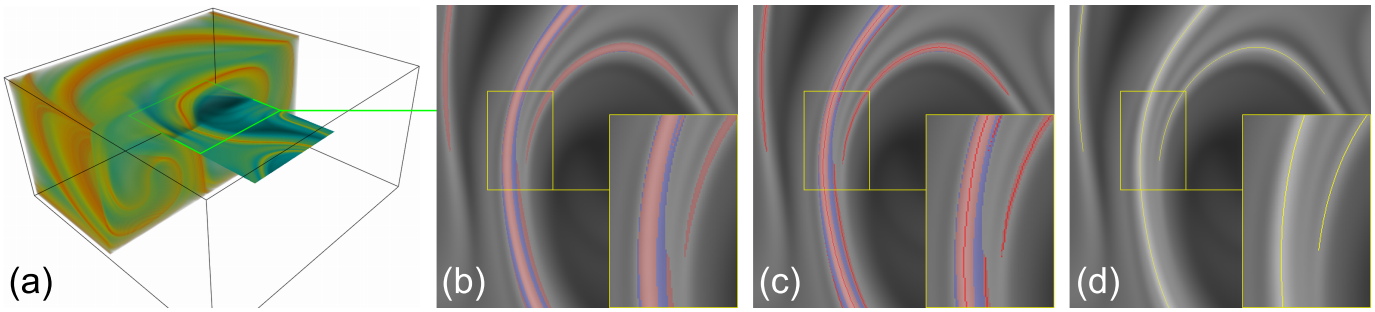


Fig. 4. Steps of the ridge extraction algorithm. (a) A planar probe positioned in the flow domain, and the corresponding color coded FTLE scalar field. (b) Threshold regions. (c) Thinning yields the pixel-accurate ridge skeleton. (d) Extracted ridge line segments at sub-pixel accuracy.

The basic idea underlying our algorithm is to separate the extraction of the ridge topology from the computation of the exact ridge locations, similar to the concept proposed in [37]. Starting with the FTLE field in a 2D texture in GPU memory, the texture is first filtered via a gaussian kernel of size 5×5 to smooth high-frequency FTLE regions (typically, $5-10$ smoothing iterations are performed). Then, the texture is processed to classify the FTLE values and build threshold regions. These regions are successively thinned to compute a pixel-accurate ridge skeleton, from which ridge line segments are extracted at sub-pixel accuracy. The different steps of this algorithm are illustrated in Figure 4.

The result of our technique are continuous ridges with a simple topology. They consist of points that are local maxima into the direction of the local ridge normals, similar to the concept of watersheds. These ridges are returned as a common graph structure G with uniform vertex spacing.

4.1 Ridge Topology

The extraction of the ridge topology is performed by first classifying the discrete set of FTLE values on the sampling grid based on the height and the local curvature of this field, and then by shrinking the resulting regions towards the ridge skeletons. If a sufficient symmetry of the hills in the height field along their main axis can be assumed, the skeleton will roughly coincide with valid ridge locations.

4.1.1 Classification

Performing the classification of FTLE values based on a height threshold is not sufficient in general, since ridges can be of different heights. This classification also fails to segment regions of nearby ridges that are separated by valleys of insufficient depth. To solve this problem we introduce an additional threshold that is used to separate convex and non-convex FTLE regions.

Let $f_{\mathbf{ww}}$ be the second order directional derivative of $f(\mathbf{p})$ into direction $\mathbf{w}$ at a fixed position $\mathbf{p}$. Moreover let λ_1, λ_2 (with $\lambda_1 \leq \lambda_2$) be the eigenvalues of H at $\mathbf{p}$. The point $\mathbf{p}$ is a convex point if every second order directional derivative is non-positive: $\forall \mathbf{w} \neq 0 : f_{\mathbf{ww}} \leq 0$. Since $\lambda_1 \leq f_{\mathbf{ww}} \leq \lambda_2$ holds for any arbitrary $|\mathbf{w}| = 1$, $\mathbf{p}$ is convex if and only if H is negative semi-definite. This results in the following condition to be fulfilled by every ridge point:

$$\lambda_2 \leq 0$$

Applying this criterion to every sample on the seeding structure $\mathbf{s}$ gives the desired classification into points belonging to convex regions and points belonging to non-convex regions. For this we calculate the Hessian pixel-wise using discrete filters on the smoothed FTLE field f on $\mathbf{s}$.

The used criterion, on the other hand, is rather sensitive against small but random fluctuations and thus leads to a rather noisy classification in approximately planar regions. This misclassification, which results in unfeasible skeletons, is resolved by allowing small positive values of λ_2, i.e., a curvature threshold $\kappa > 0$. Combined with a height threshold h to exclude ridges at locations where the FTLE value is too low we arrive at the condition

$$\lambda_2 \leq \kappa \quad \wedge \quad f(\mathbf{p}) \geq h \,. \tag{1}$$

The reason for making the condition dependent on λ_2 rather than λ_1 is that in practice this condition turns out to be much more capable of reducing the ridges' fuzziness.

Figure 5 shows FTLE classifications using the different criteria with varying threshold values. As can be seen, vastly different results are obtained, ranging from rather fuzzy to well-defined and smooth threshold regions. According to our experience, choosing κ one or two orders smaller than the largest occurring curvatures on $\mathbf{s}$ provides the best results, i.e. $\kappa \in [0.01; 0.1] \cdot \max_{\mathbf{s}}\{|\lambda_1|, |\lambda_2|\}$. For the minimal ridge height h, reasonable values are between 50% and 80% of the maximum FTLE value.

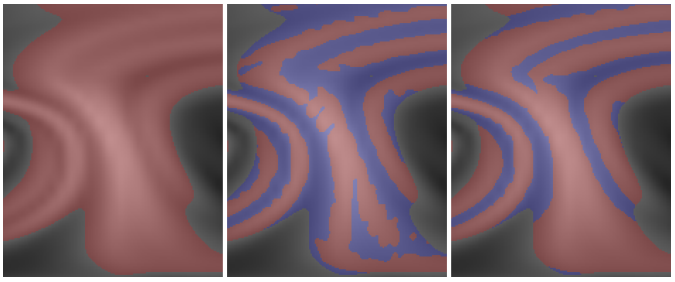
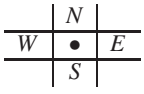


Fig. 5. Classification of FTLE values into convex (red) and non-convex (blue) regions, using a) height threshold, b) height and curvature threshold with $\kappa = 0$ and c) with $\kappa = 10^{-3}$.

4.1.2 Skeletonization via Curve-Thinning

Applying the convexity test (1) to the FTLE field results in a binary threshold image. We assume that pixels labeled 1 passed the test, while all others are labeled 0. We are now seeking for the topological skeletons of those regions consisting of pixels that passed the test, i.e., the skeletons of the convex regions.

To compute these skeletons efficiently on the GPU we employ a 2D version of the region thinning algorithm proposed by [26]. The algorithm is very robust against noise at the region boundaries, and since it performs purely local computations at every pixel it can be parallelized effectively. Furthermore, it directly generates the inner skeleton of a region, meaning that the algorithm avoids branches touching the region contour. At every pixel the algorithm considers the 4-neighborhood to classify this pixel, i.e., a pixel is classified as N- (or W-, E-, S-) border-pixel if it has value 1 and its neighbor in the respective direction has value 0:



In every iteration, at every pixel four sub-iterations are performed to remove certain border pixels. The first sub-iteration NW removes

N- and W- border-pixels that match at least one of the following three adjacency templates:

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline x & 1 & x \\ \hline x & 1 & x \\ \hline \end{array} \quad \vee \quad \begin{array}{|c|c|c|}\hline 0 & x & x \\ \hline 0 & 1 & 1 \\ \hline 0 & x & x \\ \hline \end{array} \quad \vee \quad \begin{array}{|c|c|c|}\hline 0 & 0 & \cdot \\ \hline 0 & 1 & 1 \\ \hline \cdot & 1 & \cdot \\ \hline \end{array}$$

Here '0' and '1' mark pixels that have to be exactly matched. Of the neighbors marked 'x', per template at least one has to be '1' while those marked with '$\cdot$' are irrelevant for the evaluation of the respective template. Upon finishing this sub-iteration, the algorithm proceeds with sub-iterations SE, NE, and SW in exactly this order. The templates for these sub-iterations are derived by rotating the templates used in the first sub-iteration accordingly. The thinning process is performed in as many iterations as are required until no more pixels are removed from the input image (typically a maximum of 20 iterations is sufficient).

4.2 Sub-pixel Ridge Refinement

Given the set of skeleton pixels that is output by the thinning algorithm, we construct a graph representation of the skeletons by connecting neighboring pixels. For every skeleton pixel with at least one neighbor a vertex at its center is created. Two vertices are connected via an edge if they belong to horizontally or vertically adjacent pixels (N,W,S, or E template positions) or if they belong to diagonally adjacent pixels which do not share a common neighbor. Note that this implies $deg(\mathbf{v}) \leq 4$ for all ridge vertices $\mathbf{v}$.

The graph is stored on the GPU as a linear array of vertex primitives, each carrying a pixel coordinate $\mathbf{p_v}$ in the sampling grid $\mathbf{s}$ and an adjacency list $U_\mathbf{v} \subset V$ with $1 \leq |U_\mathbf{v}| \leq 4$ implemented as pointers (indices) to up to 4 neighbors. The array is created by invoking a geometry shader for every texel in the 2D texture storing the FTLE values. Using the stream output functionality of current GPUs, we can generate primitives solely for the pixels who are part of the skeleton. To establish the connectivity between these vertex primitives, in a second rendering pass, we scatter their array indices back into an index texture of the same dimensions as the initial texture. In a third pass every vertex finally determines the connectivity information U by a lookup into the index texture.

The ridge graph is then refined iteratively at sub-pixel precision. Underlying the refinement process is the condition that every ridge vertex $\mathbf{v}$ should be lying on a maximum of the image function f into the direction of the local ridge normal $\mathbf{n_v}$. Consequently, the ridge vertices have to be moved upwards the FTLE field until they reach such a maximum. Since moving vertices along the image gradient $\mathbf{g}$ would cause the ridge graph G to be heavily distorted or even collapse at the absolute maxima of f, we restrict the movement to the $\mathbf{n_v}$-direction by projecting $\mathbf{g}$ onto $\mathbf{n_v}$. This also ensures the convergence of the refinement process under the assumption of a reasonable initial guess produced by the skeletonization. To avoid degenerate cases and to additionally obtain evenly spaced vertices, we incorporate some smoothing into each refinement step by interpolating vertex positions along ridge lines. Specifically, the position $\mathbf{p_v}$ of a vertex $\mathbf{v}$ is updated according to

$$\mathbf{p_v}' = \begin{cases} \mathbf{p_v} + \delta\, \mathbf{r_v} & , \text{ if } |U_\mathbf{v}| = 1 \\ (1-\sigma)\, \mathbf{p_v} + \frac{\sigma}{|U_\mathbf{v}|} \left(\sum_{u \in U_\mathbf{v}} \mathbf{p_u} \right) + \delta\, \mathbf{r_v} & , \text{ else} \end{cases} \tag{2}$$

with

$$\mathbf{r_v} = \begin{cases} \langle (\mathbf{p_{u_1}} - \mathbf{p_{u_2}})^\perp, \mathbf{g} \rangle \cdot (\mathbf{p_{u_1}} - \mathbf{p_{u_2}})^\perp & , \text{ if } U_\mathbf{v} = \{\mathbf{u_1}, \mathbf{u_2}\} \\ \sum\limits_{\mathbf{u} \in U_\mathbf{v}} \langle (\mathbf{p_u} - \mathbf{p_v})^\perp, \mathbf{g} \rangle \cdot (\mathbf{p_u} - \mathbf{p_v})^\perp & , \text{ else} \end{cases}$$

Here, δ is the step-size along the gradient, σ is the amount of line smoothing, and $\mathbf{w}^\perp$ denotes a unit-length vector perpendicular to $\mathbf{w}$. In order to allow for the procedure to converge, we choose the largest δ for which the step size $|\delta\, \mathbf{r_v}|$ is smaller than one pixel. σ was set to 0.25.

We perform a fixed number of iteration steps (typically 50), which are computed for every vertex in parallel on the GPU. In a final post-process the graph G is modified by removing vertices that have moved more than a user-specified distance threshold d_{max} during the refinement (by default we set d_{max} to the length of 5 pixels). In this way we eliminate skeletons that were too far from ridges after the initial skeletonization. Therefore, each vertex $\mathbf{v}$ gets assigned an additional attribute $d_\mathbf{v}$, which stores the distance a vertex has been moved. Similar to $\mathbf{p_v}$, $d_\mathbf{v}$ is smoothed along ridge lines to prevent oscillation artifacts caused by thresholding:

$$d'_\mathbf{v} = \begin{cases} (1-\omega)\, d_\mathbf{v} + \frac{\omega}{|U_\mathbf{v}|} (\sum\limits_{u \in U_\mathbf{v}} d_\mathbf{u}) + |\mathbf{p_v}' - \mathbf{p_v}| & , \text{ if } |U_\mathbf{v}| \leq 2 \\ d_\mathbf{v} + |\mathbf{p_v}' - \mathbf{p_v}| & , \text{ else} \end{cases} \tag{3}$$

Compared to σ, ω should be chosen significantly smaller. For instance, $\omega = 0.05$ was used throughout all of our experiments. Vertices with $d_v > d_{max}$ are marked invalid. Finally this mark is propagated through G in k_{cut} iterations, marking all vertices invalid from which an invalid vertex can be reached in k_{cut} steps. k_{cut} was set to 5 throughout all experiments. In Figure 6 extracted ridges before (left) and after the sub-pixel refinement stage are shown.

The result of the ridge extraction stage is the array of ridge vertices containing both the invalid and the valid ridge vertices $V^+ \subseteq V$. From these vertices the set of valid edges $E^+ \subseteq V^+ \times V^+$ is derived.

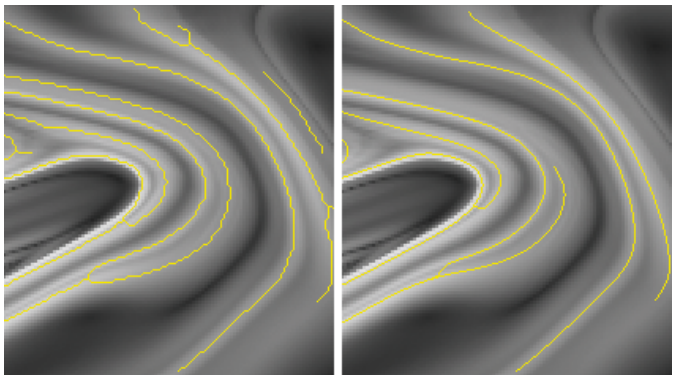


Fig. 6. Ridges extracted with our approach. a) Ridges obtained by connecting adjacent vertices. b) Sub-pixel precise ridges after the refinement and post-processing stage.

5 Separating Streak-Surface Visualization

Our ridge extraction technique yields a set of ridge structures for a given point in time. These structures are used as seeding curves for streak surfaces. Since the seeding structures change over time, the surfaces generated this way are generalized streak surfaces.

The ridges are provided as a set of uniformly distributed line segments E^+ consisting of a discrete set of control vertices V^+. To construct separation surfaces, we repeatedly release particles from the set of seed points V^+ into the flow and compute their trajectories in the unsteady 3D flow. All particles released at a given point in time are then integrated and rendered. Since the FTLE values have been computed by integrating particles over a specific time interval, the life time of the particles seeded at the FTLE ridges is restricted to the same interval.

Figures 7 and 9 (c) show separating streak surfaces that were visualized by rendering the set of particles as individual spherical point sprites. As proposed by Sigg et al. [47], an analytic ray/sphere intersection is performed in the pixel shader stage to determine correct depth values on a per fragment basis. Numerical particle integration on the GPU is performed as described in [20].

Red particles in the images correspond to control points on a separating streak surface. We optionally adapt the opacity of particles representing a separating streak surface according to the scalar FTLE value at their current position in space and time, thus, fading out primitives that are passing through regions of low FTLE. Green particles

represent control points of time surfaces. Those time surfaces are aligned parallel to the planar probe and released into the flow at a fixed frequency to serve as additional context information, emphasizing the separating nature of the extracted streak surfaces. Approximating the surface through a set of individual samples allows us to use large sets of particles at real-time performance. However, as particles start to diverge, missing connectivity between surface samples and the omission of an adaptive refinement make it difficult to identify the separating surface.

Fig. 7. Particle based surface visualization. Red particles correspond to points on the separating surface. Green particles serve as context information. They correspond to points on time surfaces, which are released from the planar probe at a fixed frequency.

In [2] two different approaches for the visualization of closed streak surfaces were proposed, with the focus on the efficient construction of such surfaces on the GPU. The first approach represents the surface as a set of separate quad-patches, which deform under the influence of the flow. Each patch is traced separately through the flow, and it is adaptively refined into a set of sub-patches if the stretching becomes too severe. The refinement process introduces new vertices that are not shared by adjacent patches, and, thus, successive integration can lead to holes in the surface representation. The second approach generates a closed surface by repeatedly releasing time lines from a single static seeding structure and triangulating adjacent (adaptively refined) timelines.

Unfortunately, the latter approach can not easily be applied in our scenario since the seeding curve changes permanently. This makes it difficult to establish particle connectivity and construct a consistent surface triangulation. Especially since the topology of the seeding curves changes from time step to time step, we would first have to determine matching curve segments in successive time steps to build a triangulation. Finding these matchings is a rather time consuming task and can not efficiently be mapped to the GPU. For this reason we adopt a variant of the patch-based approach from [2] in this work.

In each time step t_i and for every edge $e \in E^+$ we construct a zero area quad, and we release two control vertices of each patch into the flow. Before releasing the remaining two vertices in time step t_{i+1}, the ridge line segments E^+ extracted in t_i are traced along the gradient of the 2D FTLE field at t_{i+1} as described in section 4.2. Thus, the vertices are moved according to the movement of the ridge structure from one time step to the next. In this way we employ the temporal coherence of FTLE ridges to find for each ridge vertex a corresponding vertex in the next time step. The remaining two patch vertices are then released into the flow from the new positions on the seeding plane.

As the adjusted edges E_i^+ (extracted at t_i) do not exactly match the edges E_{i+1}^+, subsequent integration can lead to holes in the surface representation. This is fixed to a certain degree by using point splatting, which renders a slightly enlarged footprint to smear out holes between adjacent patches. Figure 8 sketches the construction of surface patches for a seeding structure that moves over time. Adaptive patch refinement is performed as proposed in [2]. Figures 1 and 12 show separating streak surfaces that have been constructed and visualized using our approach. In Figures 11 and 9 (a,b) depth peeling was applied to create a semi transparent visualization of the separating streak surfaces.

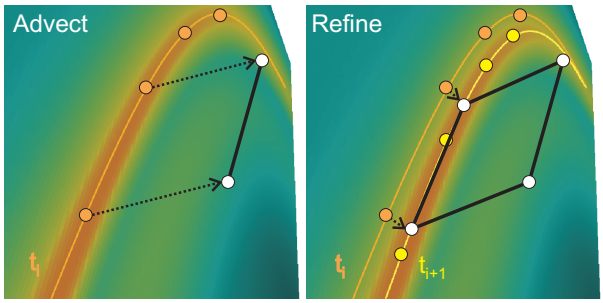


Fig. 8. Patch-based surface visualization. Before releasing the second pair of vertices at time t_{i+1}, the line segment of the corresponding ridge structure extracted at time t_i is traced along the FTLE gradient field $\mathbf{g}_{i+1}$. The four white vertices (right) depict the control points of the resulting quadrilateral.

6 Results and Discussion

To validate the effectiveness of the proposed techniques, we have conducted a number of experiments on different data sets given on 3D Cartesian grids. Performance statistics were measured on a 2.83 GHz Core 2 Quad processor, equipped with a NVIDIA Quadro FX5800 with 4 GB local video memory. Results were rendered into a viewport at FullHD resolution (1920×1080). The following data sets were used:

- *3D double gyre:* A 3D extension of the synthetic, periodic 2D double gyre [44], sampled at a spatial resolution of $256 \times 128 \times 256$ and a temporal resolution of 10 for one period:

$$\mathbf{gyre}(x,y,z,t) = (-\pi A \cdot \sin(\pi f(x,t+5z)) \cdot cos(\pi y), \\ \pi A \cdot \cos(\pi f(x,t+5z)) \cdot sin(\pi y) \cdot \frac{df}{dx}, 0) \; with$$

$$f(x,t) = a(t)x^2 + b(t)\,x,\; a(t) = \varepsilon sin(\omega t),\; b(t) = 1 - 2\varepsilon sin(\omega t),$$
$$A = 0.1,\; \varepsilon = 0.25,\; \omega = \tfrac{2\pi}{10} \; and$$
$$(x,y,z,t) \in [0;2] \times [0;1] \times [0;2] \times [0;10)$$

- *Square cylinder:* A DNS simulation of the three-dimensional flow around a square cylinder between parallel walls [35]. The vector field was resampled onto a uniform grid at resolution $192 \times 64 \times 48$. 102 time-steps were used. The scalar FTLE fields were pre-computed at fourfold the spatial and eightfold the temporal resolution.

- *Flow around a cylinder*: A large eddy simulation of an incompressible unsteady turbulent flow around a wall-mounted cylinder [6]. 22 time-steps were simulated. The size of the data grid is $256 \times 128 \times 128$. Pre-computed FTLE fields were generated at twice the spatial, and fourfold the temporal resolution.

- *LBM Flow:* A Lattice-Boltzmann simulation of the flow around a donut-shaped obstacle. The spatial resolution of the simulation domain is $128 \times 64 \times 64$. The FTLE fields on the 2D sampling grid were computed on the fly during flow exploration. In every time step, 10 vector fields were used in advance to compute the FTLE values.

6.1 Visual Exploration

3D double gyre: Placing the seeding plane parallel to the (x,y) plane essentially means to compute the FTLE ridges on the classical 2D double gyre. Our extraction shows that the obtained ridges agree with expected ridges known from the literature, with the main difference that our extraction works in real-time. Seeding streak surfaces from the

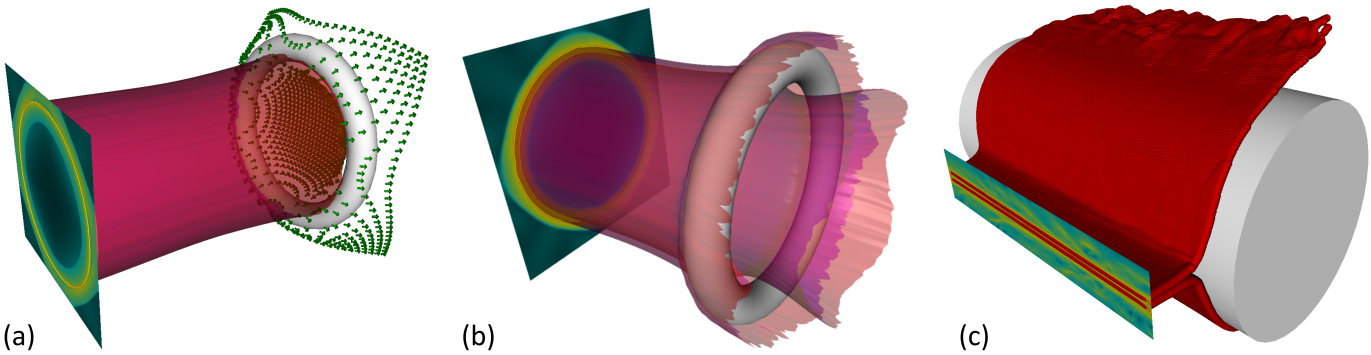


Fig. 9. (a) A tube shaped transparent streak surface around a torus, separating the flow passing through the hole and around the obstacle. (b) Moving the planar probe closer to the object reveals two individual surfaces denoting a separation behind it. (c) Placing the planar probe perpendicular to the inflow in front of the cylinder reveals two separating surfaces enclosing the object. Note that the rendering artifacts in (c) are a result of the particle based visualization approach.

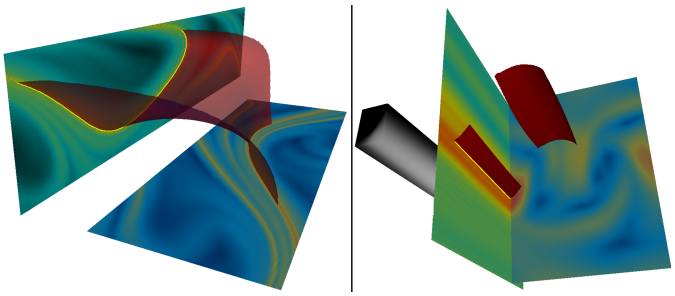


Fig. 10. Correspondence between separating streak surfaces and FTLE ridges. Besides the seeding plane, a second plane is placed such that it intersects the surface and FTLE values are visualized on it. Left: The separating surface stays on the 2D FTLE ridges. Right: In general, since the integration time used to compute the FTLE is finite, the surface keeps staying in regions of high FTLE but does not stay on the 2D FTLE ridges any more.

ridges confirms that the ridges are approximate material structures: the generalized streak surfaces and the ridges show a good coincidence as depicted in Figure 10 (left).

Placing the seeding plane parallel to the (x, z) plane reveals approximate sine shaped ridge structures (see Figure 1 (right)). Note that this curve does not coincide with moving saddle of the vector field which is a well-known characteristic property of the data set [44].

Furthermore, the plane can interactively be moved, yielding separating generalized streak surfaces (see accompanying video).

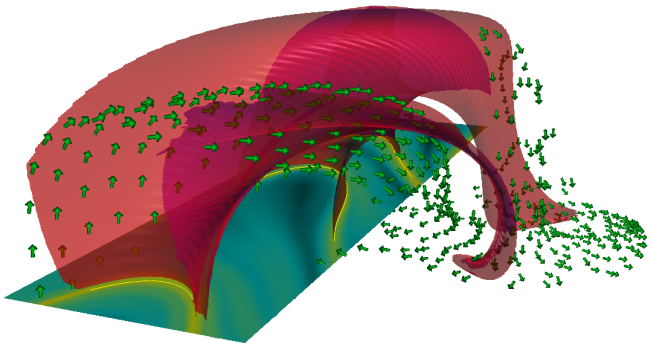


Fig. 11. Separating streak surfaces in the double gyre data set. Individual surface layers are extracted via depth peeling. Green arrows show the velocity direction on time surfaces that are additionally released from the planar probe.

Square cylinder: Seeding from a plane in front of the cylinder with a distance according to the integration time to compute the FTLE field reveals one distinct streak surface separating the flow passing above and below the cylinder (see Figure 12). Moving the seeding plane closer towards the cylinder makes more separation surfaces parallel to the first one appear. They denote a separation occurring behind the cylinder.

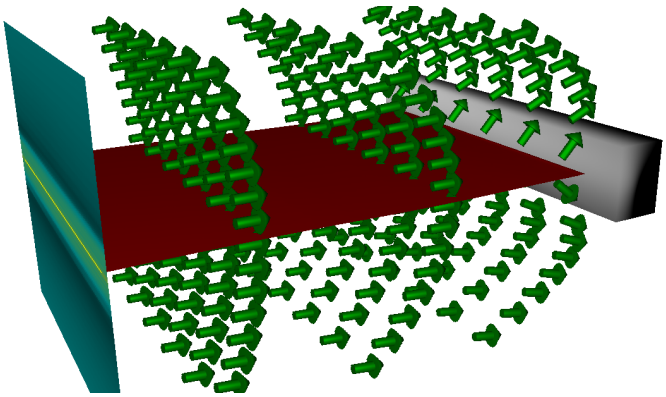


Fig. 12. A separating streak surface (red) in the square cylinder data set, visualized using a patch-based approach.

Placing the seeding plane behind the cylinder perpendicular to the main flow direction shows periodically appearing and disappearing streak surfaces which alternate in moving upward and downward. This confirms the appearance of the well-known von Karman vortex street behind the cylinder. In order to show that the streak surfaces are indeed separating structures, we release time surfaces from the seeding plane at times when streak surfaces are released. The time surfaces get advected and clearly get distorted mostly around the intersections with the streak surfaces. As can be seen in Figures 1 (middle) and 7, this shows the separating structure of our streak surfaces.

LBM Flow: A tube shaped generalized streak surface is revealed by placing the seeding plane in front of the torus, separating the flow passing through the hole and around the toroidal obstacle.

Moving **s** closer towards the obstacle creates two surfaces parallel to the first one, indicating the occurrence of a separation behind the torus (see Figure 9 (b)).

Flow around a cylinder: Two streak surfaces enclosing the cylindrical obstacle are revealed by placing the seeding probe perpendicular to the inflow in front of the object. The extracted surfaces emanating from 1D FTLE ridges on the planar probe—as shown in Figures 1 (left) and 9 (c) — closely resemble the 2D FTLE ridges obtained by incremental ridge tracking in a similar data set [32]. Notably the separating streak surface is in strong accordance with the LCS in this data set.

6.2 Performance

We applied an explicit fourth-order Runge-Kutta scheme at single floating point precision for numerical particle integration during FTLE (pre-)computation as well as for the integration of streak surfaces.

Performance measures for FTLE pre-computation are presented in Table 2. Representative timings in hours (h) are given in column *Time*

Seed Interval	Sampling texture resolution	FTLE Setup	FTLE Time	Ridge extraction	# Quads + Particles	Adv + Vis per Frame	FPS
25ms	250 × 250	-	-	6.9ms	108k	13.2ms •	32.0
50ms	500 × 500	-	-	15.2ms	639k	80.0ms □	7.3
50ms	400 × 800	-	-	25.7ms	100k	10.7ms •	24.4
100ms	250 × 250	10s in 50 steps	58.6ms	10.5ms	57k	6.7ms •	18.1
100ms	250 × 250	15s in 100 steps	120ms	9.2ms	42k	11ms □	3.3
100ms	400 × 800	10s in 50 steps	277ms	23.0ms	200k	16.7ms •	1.6

Table 1. Performance statistics for the extraction and visualization of separating streak surfaces. The surfaces were visualized using either the particle based approach (entries marked •), or the patch-based approach (marked □) including adaptive surface refinement.

for varying temporal (*Timesteps*) and spatial (*SpatialRes*) FTLE resolutions. Column *Integration* contains the integration time T and the amount of integration steps.

Timesteps	SpatialRes	Integration	Time
80	256 × 256 × 128	8s in 50 steps	0.8h
576	384 × 128 × 96	10s in 100 steps	5.0h
576	768 × 256 × 192	10s in 100 steps	43h

Table 2. Performance statistics for GPU-based FTLE computation.

Timings for the ridge extraction on the planar probe as well as for the streak surface generation and visualization are given in Table 1. We extract seeding structures (FTLE calculations and ridge extraction) at a fixed temporal frequency (*Seed Interval*) on the planar probe **s** with varying texture resolutions (*Sampling texture resolution*). In cases where the FTLE was calculated interactively, columns *FTLE Setup* and *FTLE Time* show the used parameters and the respective calculation time, whereas resampling the precalculated FTLE using trilinear interpolation comes at negligible cost. Timings for FTLE thresholding, curve thinning and ridge refinement are summarized in column *Ridge extraction*. Column *#Quads+Particles* shows the average amount of primitives employed to visualize the separating streak surfaces and additional context information, column *Adv+Vis* the time spent for respective particle integration and rendering. Column *FPS* contains the average achieved frame rate during the interactive flow exploration sessions.

6.3 Limitations

For the visual exploration of turbulent flows the proposed technique seems problematic. As can be seen in Figure 13, when placing **s** in turbulent flow regions the FTLE exhibits rather fuzzy ridge structures undergoing frequent topology changes. Hence, many small, disconnected, and strongly moving surface parts will be generated, leading to visual clutter. Reducing the FTLE integration time T as proposed by [33] to simplify the "Lagrangian skeleton" can only be done to a certain extent, as the surface integration time is restricted to T.

The application of the curvature criterion (1) followed by skeletonization especially aims to simplify the extracted ridges. It is clear, on the other hand, that this can change the ridge topology, e.g. by removing non-shallow saddles, or lead to slightly misplaced ridges, i.e., at crossings. Therefore, care has to be taken to not "misinterpret" the resulting ridges.

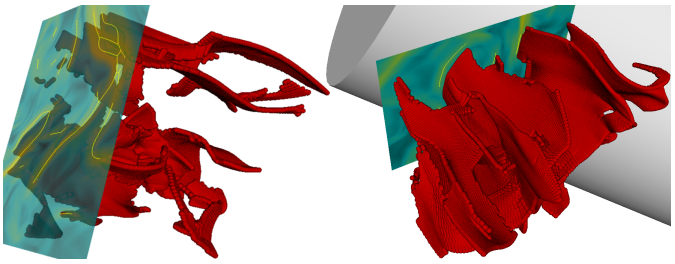


Fig. 13. Placing the seeding probe in turbulent regions. Frequent movements and topology changes of ridges result in highly fragmented surface parts and visual clutter thereof.

6.4 Conclusion

In this paper we have presented a real-time technique for the extraction of FTLE ridges on a 2D planar seeding structure in unsteady 3D flows. As we employ ridges as seeding structures for generalized streak surface integration, we focused on the extraction of a subset of all valid ridges. Whereas we aimed to a) obtain predominant features, i.e., long continuous ridge lines and b) to remove unwanted ridges such as discontinuous structures and crossings to avoid visual clutter while rendering the separating streak surface.

The GPU-based framework allows users to experience a visually guided exploration of semantic separating surfaces by moving the probe in space and/or time and changing parameters steering the ridge extraction and streak surface construction process interactively. To the best of our knowledge, this is the first time that the reconstruction and display of semantic separable surfaces in 3D unsteady flows can be performed at interactive rates, giving rise to new possibilities for gaining insight into complex 3D flow phenomena.

In fact, the interactive treatment of LCS allows insight not only into the locations of the separating structures but also into their temporal evolution including changing shapes, appearance and disappearance. Moreover, regions of interest can be determined interactively by moving around the seeding plane. This way, a fast visual impression of the "big picture" of the flow as well an in-depth analysis of relevant parts (both in space and time) becomes possible.

In the future we will pursue research into the following two directions: Firstly, we will perform a detailed analysis of the similarities and differences between generalized streak surfaces and LCS in 3D flows. Secondly, adaptive meshing techniques for constructing high-quality polygonal generalized streak surfaces will be examined. In this respect it will be worthwhile to investigate ridge extraction techniques that are specifically tailored to the intended application and can monitor topological changes and degeneracies.

Acknowledgments

The authors wish to thank Simone Camarri et al. for providing the square cylinder data set, and Tino Weinkauf for resampling the data onto a cartesian grid. Furthermore we wish to thank Octavian Frederich and co-workers for providing the large eddy simulation results.

References

[1] 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(1):9–20, 1980.

[2] K. Bürger, F. Ferstl, H. Theisel, and R. Westermann. Interactive Streak Surface Visualization on the GPU. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1259–1266, November-December 2009.

[3] K. Bürger, P. Kondratieva, J. Krüger, and R. Westermann. Importance-Driven Particle Techniques for Flow Visualization. In *Proceedings of IEEE VGTC Pacific Visualization Symposium 2008*, pages 71–78, 2008.

[4] K. Bürger, J. Schneider, P. Kondratieva, J. Krüger, and R. Westermann. Interactive Visual Exploration of Instationary 3D-Flows. In *Eurographics/IEEE VGTC Symposium on Visualization (EuroVis)*, pages 251–258, 2007.

[5] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for image analysis. *J. Math. Imaging Vis.*, 4(4):353–373, 1994.

[6] O. Frederich, E. Wassen, and F. Thiele. Flow Simulation around a Finite Cylinder on Massively Parallel Computer Architecture. In *International Conference on Parallel Computational Fluid Dynamics*, pages 85–93, 2005.

[7] C. Garth, F. Gerhardt, X. Tricoche, and H. Hagen. Efficient Computation and Visualization of Coherent Structures in Fluid Flow Applications. *IEEE Transactions on Visualization and Computer Graphics*, 13:1464–1471, 2007.

[8] C. Garth, H. Krishnan, X. Tricoche, T. Bobach, and K. I. Joy. Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1404–1411, 2008.

[9] C. Garth, G. Li, X. Tricoche, C. Hansen, and H. Hagen. Visualization of Coherent Structures in Transient 2D Flows. In *Topology-Based Methods in Visualization II (Proceedings of TopoInVis 2007)*, pages 1–14, March 2009.

[10] S. Guthe, S. Gumhold, and W. Strasser. Interactive visualization of volumetric vector fields using texture based particles. In *Proceedings of WSCG*, volume 10, pages 33–41, 2002.

[11] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Phys. D*, 149(4):248–277, 2001.

[12] G. Haller. Lagrangian coherent structures from approximate velocity data. *Physics of Fluids*, 14(6):1851–1861, 2002.

[13] G. Haller and G. Yuan. Lagrangian coherent structures and mixing in two-dimensional turbulence. *Phys. D*, 147(3-4):352–370, 2000.

[14] R. M. Haralick. Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing*, 22(1):28–38, 1983.

[15] J. P. M. Hultquist. Constructing stream surfaces in steady 3D vector fields. pages 171–178, 1992.

[16] J. Kasten, C. Petz, I. Hotz, B. Noack, and H.-C. Hege. Localized finite-time lyapunov exponent for unsteady flow analysis. In M. Magnor, B. Rosenhahn, and H. Theisel, editors, *Vision Modeling and Visualization*, volume 1, pages 265–274. Universität Magdeburg, Inst. f. Simulation u. Graph., 2009.

[17] G. L. Kindlmann, R. S. J. Estépar, S. M. Smith, and C.-F. Westin. Sampling and visualizing creases with scale-space particles. *IEEE Trans. Visualization and Computer Graphics*, 15(6):1415–1424, Nov/Dec 2009.

[18] P. Kondratieva, J. Krüger, and R. Westermann. The Application of GPU Particle Tracing to Diffusion Tensor Field Visualization. 0:10, 2005.

[19] H. Krishnan, C. Garth, and K. Joy. Time and Streak Surfaces for Flow Visualization in Large Time-Varying Data Sets. *IEEE Transactions on Visualization and Computer Graphics*, 15:1267–1274, 2009.

[20] J. Krüger, P. Kipfer, P. Kondratieva, and R. Westermann. A Particle System for Interactive Visualization of 3D Flows. *IEEE Transactions on Visualization and Computer Graphics*, 11(6):744–756, 2005.

[21] F. Lekien, C. Coulliette, A. J. Mariano, E. H. Ryan, L. K. Shay, G. Haller, and J. Marsden. Pollution release tied to invariant manifolds: A case study for the coast of florida. *Phys. D*, 210(1), 2005.

[22] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *Int. J. Comput. Vision*, 30(2):117–156, 1998.

[23] D. Lipinski and K. Mohseni. A ridge tracking algorithm and error estimate for efficient computation of lagrangian coherent structures. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(1):017504, 2010.

[24] T. McLoughlin, R. Laramee, R. Peikert, F. Post, and M. Chen. Over Two Decades of Integration-Based, Geometric Flow Visualization. In *Computer Graphics Forum, (to appear)*, 2010.

[25] D. Merhof, M. Sonntag, F. Enders, C. Nimsky, and G. Greiner. Hybrid visualization for white matter tracts using triangle strips and point sprites. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1181–1188, 2006.

[26] K. Palgyi and A. Kuba. A parallel 3d 12-subiteration thinning algorithm. *Graphical Models and Image Processing*, 61(4):199 – 221, 1999.

[27] R. Peikert and F. Sadlo. Height Ridge Computation and Filtering for Visualization. In *Proceedings of IEEE VGTC Pacific Visualization Symposium 2008*, pages 119–126, 2008.

[28] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

[29] M. Roerdink. The watershed transform: definitions, algorithms, and parallellization strategies. *Fundamenta Informaticae*, 41(1):187–228, 2000.

[30] F. Sadlo and R. Peikert. Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1456–1463, 2007.

[31] F. Sadlo and R. Peikert. Visualizing lagrangian coherent structures and comparison to vector field topology. In *Topology-Based Methods in Visualization II (Proceedings of TopoInVis 2007)*, pages 15–30, March 2009.

[32] F. Sadlo, A. Rigazzi, and R. Peikert. Time-Dependent Visualization of Lagrangian Coherent Structures by Grid Advection. In *Proceedings of TopoInVis 2009 (to appear)*. Springer, 2009.

[33] F. Sadlo and D. Weiskopf. Time-Dependent 2D Vector Field Topology: An Approach Inspired by Lagrangian Coherent Structures. *Computer Graphics Forum*, 29(1):88–100, 2010.

[34] J. Sahner, T. Weinkauf, N. Teuber, and H.-C. Hege. Vortex and Strain Skeletons in Eulerian and Lagrangian Frames. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):980–990, September - October 2007.

[35] S.Camarri, M. Salvetti, M. Buffoni, and A.Iollo. Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers. In *Proceedings of XVII Congresso di Meccanica Teorica ed Applicata*, 2005.

[36] T. Schafhitzel, E. Tejada, D. Weiskopf, and T. Ertl. Point-based Stream Surfaces and Path Surfaces. In *Proceedings of Graphics Interface 2007*, pages 289–296, 2007.

[37] T. Schafhitzel, J. E. Vollrath, J. P. Gois, D. Weiskopf, A. Castelo, and T. Ertl. Topology-preserving λ_2-based vortex core line detection for flow visualization. *Computer Graphics Forum*, 27(3):1023–1030, 2008.

[38] G. Scheuermann, T. Bobach, H. H. K. Mahrous, B. Hamann, K. Joy, and W. Kollmann. A Tetrahedra-based Stream Surface Algorithm. pages 151–158, 2001.

[39] M. Schirski, C. Bischof, and T. Kuhlen. Interactive particle tracing on tetrahedral grids using the GPU. In *Vision Modeling and Visualization*, 2006.

[40] M. Schirski, A. Gerndt, T. van Reimersdahl, T. Kuhlen, P. Adomeit, O. Lang, S. Pischinger, and C. H. Bischof. ViSTA FlowLib: A Framework for Interactive Visualization and Exploration of Unsteady Flows in Virtual Environments. In *7th International Workshop on Immersive Projection Technology, 9th Eurographics Workshop on Virtual Enviroments*, pages 77–86, 2003.

[41] M. Schirski, T. Kuhlen, M. Hopp, P. Adomeit, S. Pischinger, and C. Bischof. Efficient visualization of large amounts of particle trajectories in virtual environments using virtual tubelets. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 141–147, 2004.

[42] D. Schneider, A. Wiebel, and G. Scheuermann. Smooth Stream Surfaces of Fourth Order Precision. In *Eurographics/IEEE VGTC Symposium on Visualization (EuroVis)*, pages 871–878, 2009.

[43] T. Schultz, H. Theisel, and H.-P. Seidel. Crease Surfaces: From Theory to Extraction and Application to Diffusion Tensor MRI. *IEEE Transactions on Visualization and Computer Graphics*, 16:109–119, 2010.

[44] S. C. Shadden, F. Lekien, and J. E. Marsden. Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. *Phys. D*, 212(7):271–304, 2005.

[45] S. C. Shadden, F. Lekien, J. D. Paduan, F. P. Chavez, and J. E. Marsden. The correlation between surface drifters and coherent structures based on high-frequency radar data in monterey bay. *Deep Sea Research Part II: Topical Studies in Oceanography*, 56(3-5):161 – 172, 2009. AOSN II: The Science and Technology of an Autonomous Ocean Sampling Network.

[46] H.-W. Shen, G.-S. Li, and U. D. Bordoloi. Interactive Visualization of Three-Dimensional Vector Fields with Flexible Appearance Control. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):434–445, 2004.

[47] C. Sigg, T. Weyrich, M. Botsch, and M. Gross. GPU-Based Ray Casting of Quadratic Surfaces. In *Proceedings of the Eurographics/IEEE VGTC Symposium on Point-Based Graphics*, pages 59–65, 2006.

[48] D. Stalling. *Fast Texture-based Algorithms for Vector Field Visualization*. PhD thesis, FU Berlin, Department of Mathematics and Computer Science, 1998.

[49] J. J. van Wijk. Implicit Stream Surfaces. pages 245–252, 1993.

[50] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke Surfaces: An Interactive Flow Visualization Technique Inspired by Real-World Flow Experiments. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1396–1403, 2008.

[51] M. Weldon, T. Peacock, G. B. Jacobs, M. Helu, and G. Haller. Experimental and numerical investigation of the kinematic theory of unsteady separation. *Journal of Fluid Mechanics*, 611:1–11, 2008.

[52] A. Wiebel, D. Schneider, H. Jaenicke, X. Tricoche, and G. Scheuermann. Generalized Streak Lines: Analysis and Visualization of Boundary Induced Vortices. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1735–1742, 2007.