DOI: 10.1111/cgf.15203

COMPUTER GRAPHICS *forum*
Volume 44 (2025), number 1, e15203

A Particle-Based Approach to Extract Dynamic 3D FTLE Ridge Geometry

Daniel Stelter, Thomas Wilde, Christian Rössl and Holger Theisel

University of Magdeburg, Magdeburg, Germany
daniel@isg.cs.ovgu.de, {thomas, roessl}@isg.cs.uni-magdeburg.de, theisel@ovgu.de

Abstract

Lagrangian coherent structures (LCS) is an important concept for the visualization of unsteady flows. They describe the boundaries of regions for which material transport stays mostly coherent over time which can help for a better understanding of dynamical systems. One of the most common techniques for their computation is the extraction of ridges from the finite-time Lyapunov exponent (FTLE) field. FTLE ridges are challenging to extract, both in terms of accuracy and performance, because they expose strong gradients of the underlying field, tend to come close to each other and are dynamic with respect to different time parameters. We present a new method for extracting FTLE ridges for series of integration times which is able to show how coherent regions and their borders evolve over time. Our techniques mainly build on a particle system which is used for sampling the ridges uniformly. This system is highly optimized for the challenges of FTLE ridge extraction. Further, it is able to take advantage of the continuous evolvement of the ridges which makes their sampling for multiple integration times much faster. We test our method on multiple 3D datasets and compare it to the standard Marching Ridges technique. For the extraction examples our method is 13 to over 300 times faster, suggesting a significant advantage.

Keywords: visualization, flow visualization, scientific visualization

CCS Concepts: • Human-centred computing → Scientific visualization; • Computing methodologies → Computer graphics

1. Introduction

Flow visualization has established itself in different modern fields as it can assist in understanding natural phenomena and optimizing modern constructions. Examples are the investigation of air flow on the Hong Kong International Airport [TCH10], the analysis of air flow around revolving doors [SFB*12] or tracking the movement of plastic pollution in coastal oceans [LDK*19]. One of the most famous techniques for steady (*i.e.* time-independent) flows is the vector field topology [HH91]. The output is structures which separate regions of coherent flow behaviour. This means that integration lines inside the flow on different sides of a separatrix repel from each other. Finding and interpreting such separatrices and regions with similar advection behaviour greatly assist in understanding a flow and its dynamics.

This classical approach of vector field topology considers *steady* flows. In most applications, however, flows are *unsteady*, *i.e.* time-dependent. To deal with this, the concept of Lagrangian coherent structures (LCS) was introduced [HY00]. Its underlying idea is the same as before: LCS are separating structures and thus define regions for which material transport is coherent. They are connected to the finite-time Lyapunov exponent (FTLE) which is a scalar field measuring the stretching of local integration lines [Hal01]. Further, LCS can be extracted as the ridges of FTLE fields [SLM05].

FTLE ridges are line structures in 2D and surface structures in 3D flows. While there is vast literature to compute FTLE fields in an accurate, adaptive and fast way, only little work has been spent on extracting the ridge geometry from FTLE fields. In 2D, this is not a serious limitation, as the FTLE field can simply be visualized by a height field or a colour coding, leaving the location of the ridges to the human visual system. For 3D flows, however, this is hardly possible due to the clutter present in 3D scalar fields. We argue that especially for 3D flows, it is necessary to extract FTLE ridge geometry in addition to the FTLE scalar field to get a comprehensive visual analysis of LCS.

Extracting ridges from a scalar field is a standard problem in image processing, graphics and visualization. However, FTLE ridges

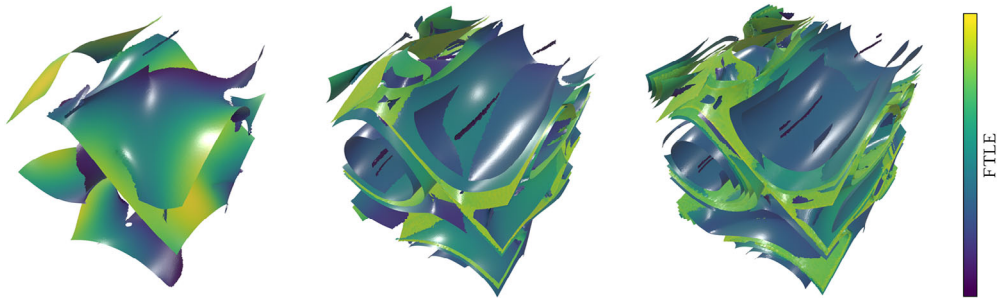


Figure 1: *Ridges extracted from the FTLE of the ABC flow at integration times (from left) $\tau = 1$, $\tau = 3$ and $\tau = 5$. The ridges are sampled by a population of oriented particles. The underlying particle system allows for efficient sampling of ridges for a sequence of multiple integration times. Visualizing how FTLE ridges emerge over time gives new insights into short- and long-term flow behaviour.*

possess multiple properties which makes their extraction more complicated. In particular, FTLE gradients increase and distances between ridges decrease exponentially with increasing integration times [SJS20]. These problems hinder the efficient application of standard approaches for FTLE ridges. This leads to the observation that only few papers demonstrate the extraction of 3D FTLE ridge geometry. To the best of our knowledge, all of them use or adapt a standard technique called Marching Ridges [FP01].

FTLE ridges evolve over time: they may move their locations, become sharper and get closer to each other with increasing integration time. Doing so, moving FTLE ridges reveal temporal coherence. The analysis of the evolvement of FTLE ridges is also part of the visual LCS analysis, making it necessary to extract ridges not only for a single FTLE field, but for a sequence of FTLE fields with increasing integration times.

In this paper, we introduce a new particle-based approach to extract sequences of FTLE ridges for increasing integration times. It is inspired by the work of Kindlmann *et al.* [KESW09] for 3D ridge extraction in the context of medical imaging but has significant differences. First, Kindlmann *et al.* [KESW09] use a scale space approach, making a repeated smoothing of the input field necessary. Due to the nature of the FTLE fields, this is replaced by the consideration of FTLE fields with lower integration times. Second, our approach uses temporal coherence to compute sequences of FTLE ridges for increasing integration times. Third, we propose a multitude of optimizations for decreasing the runtime and improving the accuracy for the specific properties of FTLE ridges. We compare our approach with standard Marching Ridges for FTLE ridge extraction and show a similar accuracy, but a significantly improved performance for the computation of sequences of FTLE ridges with speedup factors between 13 and more than 300, averaging at over 100.

2. Related Work

This section explains relevant concepts and previous work about LCS and their extraction as ridges of FTLE fields. In addition, we briefly review particle systems for sampling implicit surfaces.

LCS: As mentioned, LCS are a famous concept for visualizing dynamical systems. However, there is a variety of challenges, and numerous publications focus on different aspects. Reviews on typical extraction techniques have been provided by Shadden [Sha11] and Haller [Hal15]. A well-established technique for the extraction of FTLE ridges in 2D and 3D is the application of Marching Ridges [FP01] which adapts the well-known Marching Cubes method [LC87]. The approach uses a rectilinear grid of a scalar field (*e.g.* a FTLE field) and searches for positions on the edges which intersect ridge structures. Sadlo and Peikert [SP07] improved this technique with an adaptive refinement of the FTLE sampling grid. This makes the basic approach significantly more efficient. Also, other methods for the adaptive sampling of FTLE fields have been proposed [BT13, GGTH07, HYX*20]. Another adaptation of Marching Ridges by Schindler *et al.* [SFB*12] uses the Fast Fourier Transformation for a scale-space approach. Sadlo *et al.* [SRP11] consider the movement of FTLE ridges for increasing start times in pre-defined boundary regions. They use grid advection to calculate FTLE only in required regions. Lipinski and Mohseni [LM10] take advantage of spatial and temporal coherence for tracing LCS, but only treat 2D flows.

However, computational efficiency is not the only challenge. Tang *et al.* [TCH10] tackled the problem of spatially limited domains which can cause difficulties because of pathlines leaving the domain over time. Wilde *et al.* [WRT18] focus on very long integration times with the challenges of extremely sharp and closely located ridges in 2D. They also present the concept of ridge statistics which can be interpreted to make quantified statements about a dataset. Guo *et al.* [GHP*16] extend the idea of FTLE and LCS to uncertain flows. Nguyen *et al.* [NWMC21] provide a strategy to find large-scale coherent structures in flows with turbulence, which would normally lead to results that are hard to interpret or do not provide useful information at all. With ongoing integration time, FTLE ridges become sharp, their number increases and the distance between two ridges decreases exponentially, as shown by Kuhn *et al.* [KRWT12] and Wilde *et al.* [WRT18]. Therefore, it is a challenging task to extract the actual ridge geometry from FTLE fields. Nonetheless, they aim for an extraction of the complete ridge geometry. The sharpness of ridges and their small distance is the limiting factor in their approach. Furthermore, they stick to the 2D case and completely omit 3D flows. Sadlo and Peikert [SP07] utilize the Marching Ridges algorithm in 3D to compute a set of triangles representing the ridges. In a post-processing step, a consistent orientation of

these triangles should be established. This step failed in some cases which led to non-orientable manifolds and therefore error-prone geometry. Schindler *et al.* [SFB*12] use different ridge concepts to compute LCS surfaces in FTLE fields and especially take care of a finite domain. They utilize the Marching Cubes algorithm to extract the ridge surfaces and extend their approach to analyse the airflow in revolving doors [SPFT12]. As with other approaches, they face the problem of non-orientable surfaces and therefore have to cut back their visualization techniques. Günther *et al.* [GKT16] as well as Rojo *et al.* [RGG20] use a Monte Carlo approach for the computation of ground truth FTLE ridges. Hofmann and Sadlo [HS21] apply streak-based topology which can find FTLE ridges independent of their distance to each other, with the drawback of false negatives. Recently, Xi *et al.* [XLT23] incorporated a neural network to control the FTLE sampling and significantly reduce the computation time. Monte Carlo approaches give impressive results for a qualitative visualization of FTLE ridges—unfortunately, the actual ridge geometries are ignored.

Particle systems: The usage of particle systems for surface representation is a well-studied area of computer graphics. One of the earliest contributions was made by Szeliski and Tonnesen [ST92], who introduced a system of oriented and interacting particles to define new types of surfaces. This work laid the foundation for further surface re-construction methods, such as those used for visualizing implicit surfaces [WH94, Hec97]. Meyer *et al.* proposed methods for sampling surfaces depending on the local curvature [MGW05] and for sampling isosurfaces with high-order finite elements [MNKW07]. Another adaptation was made by Kindlmann *et al.* [KESW09] for finding ridge and valley structures, also using a scale-space approach for finding structures of different scales. This approach gained prominence in visualizing medical scan data, leading to further adaptations and refinements [NJCBP*18, LVHV*23].

3. Theoretical Background

In this section, we briefly review the relevant concepts which are required for our methods. This covers the definition of FTLE and ridges as well as particle systems for sampling and re-constructing surfaces. For more detailed information, we refer to our references in Section 2.

3.1. FTLE ridges

We will extract LCS as the ridges of the FTLE for which we propose a brief introduction. An unsteady n-dimensional flow is given by a vector field $\mathbf{v}(\mathbf{x}, t)$ which represents the velocity of a flow for any position $\mathbf{x} \in \mathcal{D} \subset \mathbb{R}^n$ and time point $t \in \mathcal{T} \subset \mathbb{R}$. Using this, we define the flow map $\Phi_{t_0}^{\tau}(\mathbf{x})$ which maps a position $\mathbf{x}$, a start time t_0 and an integration time τ to the resulting position of the integration of $\mathbf{v}$ for a period τ. In other words, it describes to which position a massless particle seeded at $(\mathbf{x}, t_0)$ has been advected by the integration. Using the flow map, the scalar FTLE field $\xi_{t_0}^{\tau}(\mathbf{x})$ can now be defined as

$$\xi_{t_0}^{\tau}(\mathbf{x}) = \frac{1}{|\tau|} \ln \sqrt{\lambda_{\max}(\nabla^\top \nabla)} \quad \text{with} \quad \nabla := \nabla \Phi_{t_0}^{\tau}(\mathbf{x}) \tag{1}$$

for $\tau \neq 0$. This definition uses the flow map gradient which is computed numerically from finite differences of $\mathbf{x}$. It calculates the maximum stretching of the gradient, *i.e.* the maximum separation of the flow, which is scaled afterwards to counter exponential growth. For positive integration times, large FTLE values indicate strong repelling behaviour, for negative integration times, they indicate attraction.

One way to extract LCS from FTLE is to extract its ridges. There exist different kinds of ridge definitions [Ebe96]. For this work, we use the definition as generalizations of local maxima. Given a scalar field $s(\mathbf{x})$ with $\mathbf{x} \in \mathcal{D} \subset \mathbb{R}^n$, eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$ and corresponding eigenvectors $\mathbf{c}_1, \ldots, \mathbf{c}_n$ of the Hessian of s, then $\mathbf{x}$ is a d-dimensional ridge point if

$$\left[\mathbf{c}_1, \ldots, \mathbf{c}_{n-d}\right]^\top \nabla s(\mathbf{x}) = \mathbf{0}, \quad \lambda_{n-d} < 0 \tag{2}$$

holds, where $0 \leq d < n$. In this work, we focus on the case $d = n - 1$ which coincides to ridge lines for $n = 2$ and to surfaces for $n = 3$. This reduces the conditions of Equation (2) to the dot product $\langle \mathbf{c}_1, \nabla s(\mathbf{x}) \rangle = 0$ and $\lambda_1 < 0$, which means that $\mathbf{x}$ is a maximum of s in direction of the largest negative curvature. Beyond that, it is possible to filter minor ridges by introducing a threshold $\lambda_{\max} < 0$.

3.2. Particle systems

Particle systems build on the idea of sampling surfaces by a set of moving particles. We explain the concepts of constraining as well as updating, creating and deleting them with the goal of an accurate and uniform distribution. For this, we focus on sampling of ridge structures.

The first requirement is that particles can be constrained to correctly sample the surface. For the sake of determining the direction to a closely located ridge, Kindlmann *et al.* [KESW09] use the gradient $\mathbf{g}$ and Hessian $\mathbf{H}$ of the scalar field s at position $\mathbf{x}$. A spectral decomposition of $\mathbf{H}$ yields its eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$ and corresponding orthogonal eigenvectors $\mathbf{c}_1, \ldots, \mathbf{c}_n$ with unit length. For finding d-dimensional ridges, the approximate ridge tangent $\mathbf{T}$ is obtained as the sum of the projection matrices of the d largest eigenvectors. These matrices are computed using the outer product $\otimes$:

$$\mathbf{T} = \sum_i \mathbf{c}_i \otimes \mathbf{c}_i \quad \text{with} \quad i \in \{n - d + 1, \ldots, n\}. \tag{3}$$

The direction $\mathbf{d}$, which points towards the ridge, is calculated as the projection of the gradient:

$$\mathbf{d} = (\mathbf{I} - \mathbf{T})\mathbf{g}. \tag{4}$$

A typical way to use this for constraining a particle to the surface is to iteratively update its position using an adaptive step size until the magnitude of $\mathbf{d}$ falls below a threshold.

Constraining a set of densely seeded particles is already able to locate ridge structures inside the domain. However, it is desirable that surfaces are uniformly sampled to accurately represent the structures. This is typically accomplished by formulating the particle system as a minimization task of a system energy E:

$$E = \sum_{i=1}^{N} E_i \quad \text{with} \quad E_i = \sum_{j=1, i \neq j}^{N} E_{ij}. \tag{5}$$

The global energy E is the sum of the energies E_i of all single particles. Further, E_i is the sum of all inter-particle energies E_{ij} between the particle itself and all other particles in the system. Defining the inter-particle energy is crucial for the desired behaviour of the system, and thus, has to be designed carefully. We will use the tunable smooth energy profile of Kindlmann *et al.* [KESW09] which has multiple desirable properties for our use case. It is defined as

$$\phi(r) = \begin{cases} 1 + \frac{3(d-1)|r|}{w} - \frac{3(d-1)|r|^2}{w^2} + \frac{(d-1)|r|^3}{w^3} & 0 \leq |r| < w, \\ d - \frac{2d(|r|-w)^3}{(w-1)^3} - \frac{3d(|r|-w)^2}{(w-1)^2} & w \leq |r| < 1, \\ 0 & 1 \leq |r| \end{cases} \quad (6)$$

with standard parameters $w = 0.6$ and $d = -0.002$. The function goes through the point (w, d) which defines its minimum. As particles seek to decrease their energy, this position significantly influences the most desirable distance between two particles. This energy potential function can be used to define E_{ij}, *e.g.* as

$$E_{ij} = \phi(r_{ij}) = \phi\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma}\right), \quad (7)$$

which uses the scaled Euclidean distance between two particles. This leads to positive energy for particles that are near each other, negative energy when a neighbour is located a little further away, and no energy at all if the distance exceeds the scaling parameter σ. As particles seek the position of most negative energy, this leads to attracting and repelling forces towards the optimal position. An exception is particles with a distance larger than σ which do neither impact each other nor the global energy at all. Therefore, we will refer to σ as the *local support range*. Its application strongly simplifies the computation of the global energy as particles outside this range do not have to be considered.

Given that a set of particles is already located on a surface, the next step is to improve their distribution. As mentioned before, this is done in terms of minimizing the system energy E. A particle is updated by advecting it in a direction which reduces its energy while keeping it on the surface. Thus, its negative energy gradient $-\nabla E_i$ is calculated and projected onto the ridge tangent $\mathbf{T}$ with

$$\nabla E_i = \frac{dE_i}{d\mathbf{x}_i} = \sum_{j=1, i\neq j}^{N} \frac{dE_{ij}}{d\mathbf{x}_i} = \sum_{j=1, i\neq j}^{N} \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} \frac{1}{\sigma} \phi'(r_{ij}). \quad (8)$$

The particle is updated to a position along the direction $-\mathbf{T}\nabla E_i$ which has a lower energy. Iteratively updating all particles finally results in a uniform distribution.

Above that, population control, which consists of creating and removing particles, is also able decrease the system energy. Since the energy profile ϕ also maps to negative values, adding more particles can trivially lower the system energy—under the restriction of staying on the surface. Oversampled areas, on the other hand, partially contain redundant particles which creates computational overhead. The energy profile addresses this by positive values for nearby particles. Therefore, deleting particles is also capable of decreasing the energy. Iteratively applying particle updates and population control increases the sampling step by step and results in a well-distributed sampling.

4. Methods

We start with an overview of our method in order to give an outline of the algorithm. Its main steps are then discussed in detail in the subsequent parts of this section.

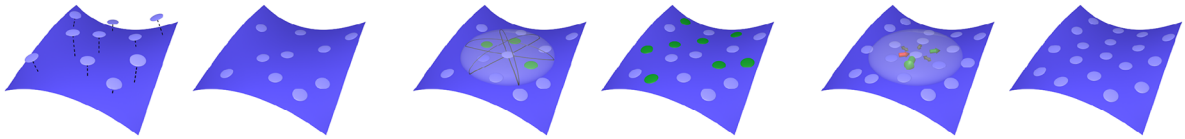
4.1. Overview

We construct a particle system for efficient sampling of FTLE ridges for sequences of integration times. There are two main challenges: First, the extraction takes more time as we extract results for sequences of multiple integration times. Second, accurately sampling all ridges requires a large number of particles. In some cases, the resulting population consists of several hundred thousand particles (see Section 5.2). We use two strategies for increasing the efficiency: First, we reduce the typical steps of a particle system by taking advantage of reusing results from previous integration times. Second, we introduce an adapted distance measure for the energy definition to significantly reduce the amount of required particles (see Section 4.3).

Our particle system uses a uniform grid of FTLE values as input. We apply multi-linear interpolation for re-constructing a continuous FTLE field. This way, the FTLE values need to be calculated only once as a pre-process, avoiding expensive calculations of the flow map during the execution of our ridge re-construction algorithm. We remark that using discrete FTLE fields is a common practice, and we refer to other approaches listed in Section 2. In the following, we will note this as the *FTLE grid*.

Besides the FTLE grid, we additionally introduce the *population grid* and the *initialization grid*. Both may have a smaller resolution than the FTLE grid. The initialization grid is described in Section 4.4. The population grid is utilized for efficiently saving and accessing nearby particles in a specific location. We use a technique similar to Kindlmann *et al.* [KESW09]. Each particle is associated with exactly one (*voxel*) cell of the grid. Due to the local support range σ of the energy profile from Equation (6), all potential neighbouring particles can be found in voxels with a maximum distance of the requested radius. This prevents redundant comparisons of particles in different parts of the domain.

Algorithm 1 describes our particle system. The main steps are illustrated in Figure 2. For each integration time step, we require the corresponding FTLE grid. Every *initPeriod* steps we execute a dense initialization of new particles and add them to the current population (Section 4.4). Then, the whole population is constrained to the FTLE ridge surfaces. We present our constraining method in Section 4.2, its concrete application is explained in Section 4.4. After that, undersampled regions are extended by creating new neighbouring particles, and the population is updated using the system energy (Section 4.3). Finally, the particles can be visualized, which also requires the normal of the ridge surfaces at the particle position. The normals are used to render oriented splats. For a sufficiently large radius of these splats, they overlap which visually 'closes gaps' in the point sets that represents the surfaces. We visualize the strength of the separation using a colour coding of the FTLE values for each individual splat. We emphasize that in all of these steps, particles act autonomously. This enables a



(a) Constraining (b) Creating neighbors (c) Updating

Figure 2: *Sampling a ridge surface (blue) in 3D space using particles (light blue). First, particles are constrained to the ridge (a). Then the constrained particles create neighbours (green) in slices of their support range, which do not contain any particles (b). Last, particles update their position based on attraction and repulsion forces of neighbouring particles (c).*

Algorithm 1. Particle system for FTLE ridge extraction.

Input:
itSteps: Number of integration time steps
initPeriod: Periodicity of dense initialization of new particles
Prerequisites:
FTLE grids for all integration time steps

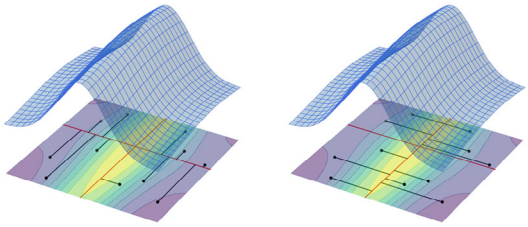
$\mathcal{P} \leftarrow \emptyset$
for $step \in \{1, \ldots, itSteps\}$ **do**
  `loadFTLE`($step$)
  if $step \equiv 1 \bmod initPeriod \vee |\mathcal{P}| = 0$ **then**
    $\mathcal{I} \leftarrow$ `initialize()` ▷ Section 4.4
    $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{I}$
  end if
  $\mathcal{P} \leftarrow$ `constrain`($\mathcal{P}$) ▷ Sections 4.2 and 4.4
  $\mathcal{P} \leftarrow$ `createNeighbors`($\mathcal{P}$) ▷ Section 4.3
  $\mathcal{P} \leftarrow$ `updateByEnergy`($\mathcal{P}$) ▷ Section 4.3
  $\mathcal{N} \leftarrow$ `normals`($\mathcal{P}$)
  $\mathcal{F} \leftarrow$ `ftle`($\mathcal{P}$)
  `visualize`($\mathcal{P}, \mathcal{N}, \mathcal{F}$)
end for

straightforward parallel implementation which significantly accelerates computation.

4.2. Constraining particles to FTLE ridges

Section 3.2 explains how Kindlmann *et al.* [KESW09] use a particle system for sampling d-dimensional ridge structures. Some parts of these methods are not practical for the search of FTLE ridges. The main problem is their constraining approach. In Equation (4), the approximate ridge tangent $\mathbf{T}$ is used, which is created by the eigenvectors of the d largest eigenvalues. In case of ridge surfaces in three-dimensional space, this means that a particle always moves in the direction of the smallest curvature. This works fine for ridges with a limited curvature when particles are already located nearby to the ridge. In general, however, the same direction exhibits a positive curvature further from the actual ridge, thus it might move in parallel to the ridge. For very sharp ridges—as in the case of FTLE—the region for selecting the 'correct' eigenvectors is extremely thin. Figure 3(a) illustrates this problem. In addition, their constraining uses an adaptive step size which showed to be inappropriate for the extreme gradients of FTLE fields (see Section 5.1).

For these reasons, we introduce a different approximation for the ridge tangent. We order the eigenvalues $|\hat{\lambda}_1| \leq \cdots \leq |\hat{\lambda}_n|$ with corresponding eigenvectors $\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_n$ of $\mathbf{H}$ by their *absolute* instead of their *total* value. Then the new approximated tangent $\hat{\mathbf{T}}$ is calculated using the d eigenvectors of smallest absolute curvature:



(a) Eq. (4) (b) Eq. (10)

Figure 3: *Comparison of constraining methods in a 2D scenario. (a) depicts the approach of Kindlmann* et al. *[KESW09] and (b) our own method. Red lines indicate ridge lines of the scalar field. Black dots represent particles and black lines their trajectory by iterating application of the respective direction. In (a), most particles move parallel to the stronger ridge. This is prevented in (b).*

$$\hat{\mathbf{T}} = \sum_i \hat{\mathbf{c}}_i \otimes \hat{\mathbf{c}}_i \quad \text{with} \quad i \in \{1, \ldots, d\} \tag{9}$$

$$\hat{\mathbf{d}} = (\mathbf{I} - \hat{\mathbf{T}})\mathbf{g}. \tag{10}$$

This changes particles to move along eigenvectors of curvature with largest magnitudes, oriented to increasing direction of the FTLE field. The new behaviour is depicted in Figure 3(b).

To actually locate the ridges, we take advantage of the discretization and interpolation of the FTLE field. The distance δ_ξ between two samples of the FTLE grid also limits the minimum distance of ridges that can be distinguished and represented. Therefore, our method sets the length of $\hat{\mathbf{d}}$ to δ_ξ. Particles are moved step by step until their direction flips. This can be checked by saving the last two directions $\hat{\mathbf{d}}$. If their dot product is negative, then a ridge must exist between the last two positions of the particle. A bisection in this segment is used to constrain the ridge location. The search can be done by iteratively determining the gradient $\mathbf{g}_\mathrm{m}$ at the centre point $\mathbf{m}$ of the current segment and checking the sign of the dot product $\langle \hat{\mathbf{d}}, \mathbf{g}_\mathrm{m} \rangle$. This refines the segment up to an arbitrary precision. Because of the multi-linear FTLE interpolation, this method safely and accurately locates the nearest ridge.

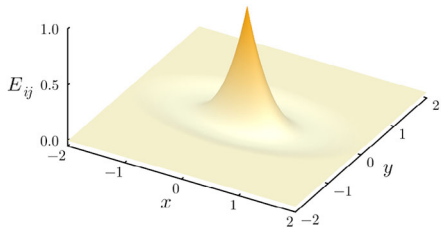


Figure 4: *2D example of inter-particle energy E_{ij} from Equation (14) for $s = 2$, $\sigma = 1$ and $\mathbf{x}_i = (0, 0)$ with ridge normal $\mathbf{c}_1 = (0, 1)$. Particle position $\mathbf{x}_j$ is represented by the x and y axes.*

4.3. Energy definition and population mechanics

As explained in Section 4.1, our system produces a large number of particles. The main reason for this is that the local support range σ has to be chosen such that particles of neighbouring ridges do not interact with each other. FTLE ridges can be—and are also expected to be—located very close to each other, thus σ must not be selected larger than the cell size δ_ξ of the FTLE grid. For this, we adapt the definition of the inter-particle energy E_{ij}. Our idea is to stretch the Euclidean distance in direction of the ridge tangent by a factor $s > 1$ which results in a modified metric with elliptic unit balls. Its definition depends on the dimension d of the ridge structure. For $d = (n - 1)$, *i.e.* a 2D ridge surface in 3D or a 1D ridge in 2D, let $\mathbf{x}_i$ and $\mathbf{x}_j$ be the particle positions on a ridge, $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ their difference and $\mathbf{H}$ the Hessian of the FTLE field at $\mathbf{x}_i$. Further, $\mathbf{c}_1$ is the unit length eigenvector of largest negative curvature of $\mathbf{H}$. This eigenvector corresponds to the normal of the ridge structure at $\mathbf{x}_i$. Let α denote the angle in Euclidean space enclosed by $\mathbf{c}_1$ and $\mathbf{r}_{ij}$. We define our adapted distance γ_{ij} and its gradient as

$$\gamma_{ij} = \|\mathbf{r}_{ij}\| \cdot \sqrt{\cos^2\alpha + \frac{\sin^2\alpha}{s^2}} \tag{11}$$

$$= \|\mathbf{r}_{ij}\| \cdot \sqrt{\frac{\langle \mathbf{c}_i, \mathbf{r}_{ij}\rangle^2}{\|\mathbf{r}_{ij}\|^2} + \left(1 - \frac{\langle \mathbf{c}_i, \mathbf{r}_{ij}\rangle^2}{\|\mathbf{r}_{ij}\|^2}\right) \cdot s^{-2}} \tag{12}$$

$$\nabla\gamma_{ij} = \frac{(s^2 - 1) \cdot \langle \mathbf{c}_i, \mathbf{r}_{ij}\rangle \cdot \mathbf{c}_i + \mathbf{r}_{ij}}{s \cdot \sqrt{\|\mathbf{r}_{ij}\|^2 + (s^2 - 1) \cdot \langle \mathbf{c}_i, \mathbf{r}_{ij}\rangle^2}} \tag{13}$$

We use this modified local distance measurement for a new inter-particle energy E_{ij} and the particle energy gradient ∇E_i:

$$E_{ij} = \phi\left(\frac{\gamma_{ij}}{\sigma}\right) \tag{14}$$

$$\nabla E_i = \sum_{j=1, i\neq j}^{N} \frac{1}{\sigma} \cdot \nabla\gamma_{ij} \cdot \phi'\left(\frac{\gamma_{ij}}{\sigma}\right). \tag{15}$$

Figure 4 shows a plot of the inter-particle energy E_{ij}. For our experiments, we used the stretching factor $s = 4$. This change reduces the number of particles drastically as larger neighbourhoods can be achieved while still keeping the minimum distance δ_ξ to possible neighbouring ridges.

Besides updating particles by their energy, a dense sampling requires creating new particles for undersampled regions in the neighbourhood of existing ones. For a good, uniform isotropic sampling, we assume that a particle ideally has up to six neighbours. Thus, the support region $\gamma_{ij} < \sigma$ of a particle i is partitioned into six equally sized slices. Each of these slices can be checked for already existing particles. In case of an empty slice, a new particle can be added temporarily to the population. First, it has to be constrained to the ridge structure. If this step already fails, the particle can immediately be dropped. Otherwise, it is updated several times using its energy gradient so that it adapts its position to the surrounding neighbours. The particle is permanently added to the population only if $E_i \leq 0$ holds, which indicates an improved sampling quality. After that, the newly created particle can check for new neighbours itself. The procedure stops as soon as each particle was handled once, meaning that no more new particles are added. Figure 2(b) illustrates this step. We use the same energy check for the population constraining. As particles are never inserted if they would decrease the system energy, this already prevents oversampling. Therefore, we can avoid a deletion procedure for removing particles in oversampled regions, further reducing the computational effort.

4.4. Initialization and iteration

The overall goal is to sample all FTLE ridges for a sequence of integration times. We start with the smallest integration time τ_1 and iterate over all remaining ones. The rationale for initializing particles across the domain is to place at least one particle on every structure which is not sampled already. For this, we use a regular initialization grid. For each cell, a new particle is seeded in its centre, which can be constrained using the steps from Section 4.2. Adding a small random offset to the initial position has shown to contribute to a better sampling distribution on the ridge structures.

This initialization must be performed for the first iteration. Additionally, new ridges emerge for increasing integration times, which means that we also need to repeat this process in subsequent steps. However, we can take advantage of several properties of how FTLE ridges change for increasing integration time. First, they move continuously and slowly. Given that the difference between the current and last integration times $|\tau_i - \tau_{i-1}|$ is sufficiently small, the sampling for τ_{i-1} can already deliver many well-distributed particles in vicinity of the ridges for τ_i. Thus, constraining the previous population can already provide a good sampling which prevents extensive creation of new neighbours. Second, ridges become sharper and move closer to other ridges with increasing integration time. This also means that they are weak and have a comparably large distance to other ridges when they evolve, which makes them easier to find. As sharp ridges can easily be sampled using the previous population, initialization is only required to find these new emerging ridges. Therefore, the resolution of the initialization grid can be coarser than for an extensive search for all ridges so that less particles have to be constrained. Additionally, for most values τ_i, no new ridges develop. When a ridge emerges, on the other hand, it is weak and small in size, *i.e.* less relevant. Depending on $|\tau_i - \tau_{i-1}|$, it is reasonable to apply the initialization procedure only every few steps which again saves execution time. In Algorithm 1, this is depicted by the parameter *initPeriod*.

When constraining the population (including particles from the last iteration and initialization) to the ridges for τ_i, they are inserted into a new population to prevent interference of constrained and unconstrained particles. For newly initialized particles, we track their trajectory. If a particle moves more than two cells apart of its origin cell, the constraining is aborted. This prevents particles from searching far-off ridges which will be found by closer particles. After successful constraining, we apply the same steps as for temporarily created neighbours (Section 4.3), which means that we update its position by its energy multiple times and only insert it to the population if $E_i \leq 0$ holds. We have found empirically that tuning the energy profile (see Equation 6) with the energy minimum $d = -0.02$ is beneficial. This increases the influence of negative energies which prevents overly strict denials of new particles.

5. Results

In this section, we compare our method to the standard Marching Ridges [FP01] technique. Additionally, Section 5.1 features a comparison to the particle system of Kindlmann *et al.* [KESW09] without a scale space. Ridges are extracted from FTLE fields that have been sampled on a regular grid as a pre-process. We remark that Sadlo and Peikert [SP07] present an application of Adaptive Mesh Refinement (AMR) to Marching Ridges. It locates ridges for a common maximum cell resolution at a higher efficiency. We acknowledge and discuss potential advantages of their adaptive method in Section 5.6. However, such an extension with similar benefits may be possible in the future (see Section 5.7). Therefore, we defer the comparison of respective adaptive methods to future work and instead focus our analysis on the underlying extraction methods, specifically using regular FTLE grids.

We implemented all methods in C++ which are executed on a system with 32 GB memory and an Intel Core i7-7700K CPU with four physical cores (4.2 GHz each). The code used in this work is available on GitHub at https://github.com/Daniel-Stelter/FTLE-Ridge-Extraction. All methods take advantage of parallel processing: For Marching Ridges, each cell can autonomously search for intersections with the surface and create the corresponding triangles. For the particle systems, each particle can operate autonomously. We test the techniques on multiple datasets and compare their results with respect to runtimes, accuracy, visual quality and extraction for different sampling densities. For our method, we choose the local support range as the distance between two FTLE samples, *i.e.* $\sigma = \delta_\xi$, and the stretching factor $s = 4$ (see Section 4.2). In all experiments, the initialization and population grid resolutions both are set to $1/8$ of the FTLE grid resolution, and we use $initPerid = 5$.

5.1. Accuracy

For testing the accuracy of the ridge extraction methods, we constructed a ground truth dataset with concentric ridges (see Figure 5). We applied Marching Ridges as well as Kindlmann's and our particle systems to the dataset with 100 time steps and a spatial resolution of 200^3. For the 3D application, we added a third dimension with constant values. Table 1 presents runtimes and error measures over all time steps, and Figure 6 shows the extraction of the three methods for the last step. Both Marching Ridges and our particle system reliably find all ridges, with a bit smaller errors for our system. Besides outliers, larger errors appear for the smaller rings with larger curvature and smaller ridge strength. Our method is much faster, which we trace back to the fact that our particle system takes advantage of temporal coherence, as well as a sparser sampling. In fact, for the last step, our method extracts 48, 142 particles, Marching Ridges on the other hand creates 571, 728 triangles with 288, 308 vertices. Kindlmann *et al.* produce very large errors *and* runtimes. Reasons for the long runtimes have been discussed in Section 4 where we explained our design decisions. This shows that their system is not designed for such large population sizes (811, 098 particles for the last iteration). The large errors come due to bad constraining, even leading to false-positives for sharp ridges.

In Figure 7, we further investigate the constraining steps of both particle systems on the 2D dataset. Kindlmann's method suffers from multiple drawbacks. First, particles in vicinity of sharp ridges

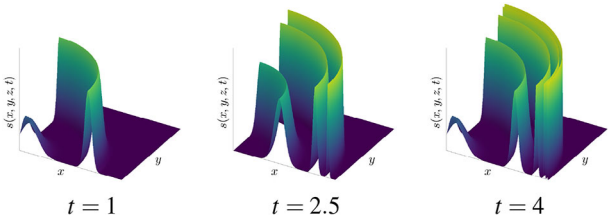


Figure 5: *Plots of the synthetic time-dependent 2D scalar field which is used for testing the accuracy of the extraction methods. The scalar field shows ridges with exponentially decreasing distance and exponentially increasing sharpness over time.*

Table 1: *Runtimes and distance errors to closest ridge for Kindlmann* et al.*, Marching Ridges and our system for the test dataset with 100 time steps (see Figure 5).*

	Runtime [min]	Max error	Mean error
Kindlmann *et al.*	2873	1.62e-2	8.77e-4
Marching Ridges	565	1.25e-2	6.92e-4
Ours	7	3.93e-3	6.82e-4

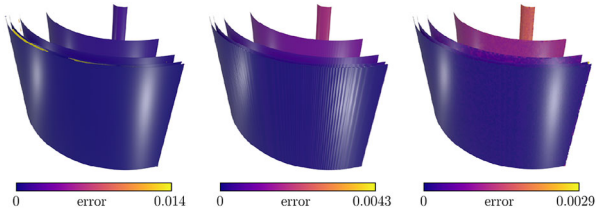


Figure 6: *Ridge extractions for Kindlmann* et al. *(a), Marching Ridges (b) and our method (c). The colour mapping represents the distance error to the closest ridge. Largest errors for Kindlmann* et al. *come due to false-positives. For Marching Ridges and our method the largest errors are few outliers on the boundary edges of the domain.*

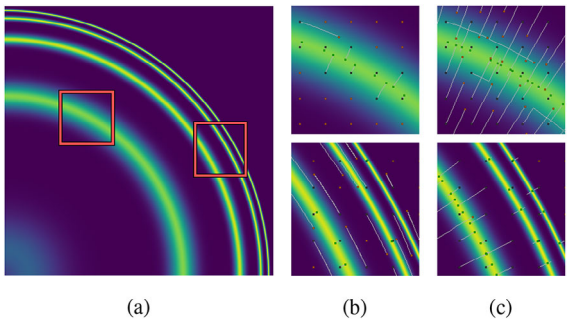


Figure 7: *Comparison of constrainings of Kindlmann's (b) and our (c) methods. Black points represent start positions of particles, grey lines are their trajectories through constraining and green/orange points are their destinations (accepted/rejected).*

move in parallel to the ridges, and thus, fail to find the ridge in most cases. Second, we were unable to find a configuration for the adaptive step size control described by Kindlmann *et al.* [KESW09] which delivers reliable results for sharp ridges due to extreme changes of the gradient. At the same time, many particles further away do not even move towards the ridges as the gradient is too weak. Third, the usage of the adaptive step size also results in a large number of steps, increasing the computational effort. Our system, on the other hand, reliably finds all ridges. Particles move towards the ridges even if located further away. The only drawback is that our particles also might move in parallel for less sharp ridges. However, these ridges are still found by particles which are initialized closer to these ridges.

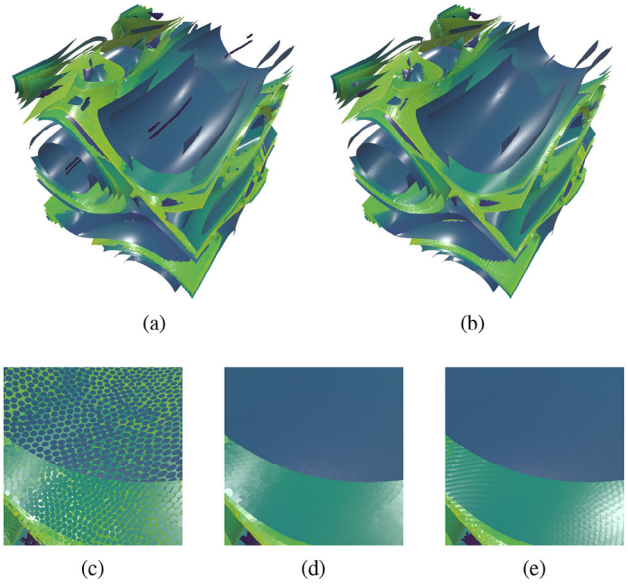


Figure 8: *FTLE ridges for $\tau = 5$ for the ABC flow extracted by our particle system (a) and Marching Ridges (b). The extractions for both approaches produce similar results in terms of number and locations of the sampled ridges. Visualizing the extracted particle population with small splats (c) reveals a uniform distribution across the sampled structures. Increasing the size of the splats leads to the effect of smooth, visually continuous surfaces (d). Marching Ridges extracts the same ridge surfaces with a triangulation (e), however, in some locations, one can see step-like artifacts.*

5.2. Datasets

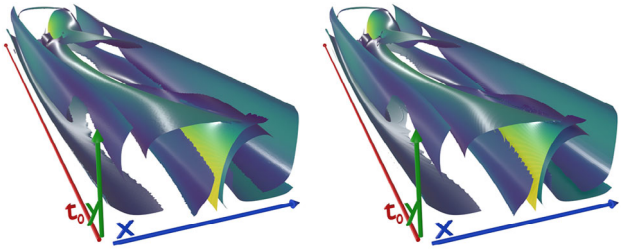
In the following, we applied our particle system and Marching Ridges to FTLE field series of several flow datasets. Table 2 summarizes setups and runtimes for all datasets.

ABC flow: The ABC flow [Hal01] is an analytical three-dimensional flow with turbulent behaviour. For our experiments, we used parameters $A = 3$, $B = 2$ and $C = 1$. Figure 1 shows results of our particle system for three values of τ. For a comprehensive visualization of all time steps, please see the accompanying video. Figure 8 compares the results of the extraction of the approaches for $\tau_{\max}$. In Table 2, one can see that the runtime of Marching Ridges was approximately 30 times longer than for our method. For Marching Ridges, we additionally observed that each iteration approximately had the same execution time, *i.e.* its performance is nearly independent of the integration time. Therefore and due to time limitations, for the following experiments, we applied Marching Ridges only for $\tau_{\max}$. This single iteration can give an estimate for the total execution time for all integration times. At the same time, we can still compare the final results of both approaches qualitatively.

Double Gyre: The Double Gyre [SLM05] is one of the most prominent benchmarks for testing FTLE 2D ridge extraction. It is an analytical dataset describing two gyres with an unsteady, periodic flow behaviour in the domain $[0,\ 2] \times [0,\ 1]$. For our experiments, we extend the dataset with a third dimension that contains different

Table 2: *Setups for all flow datasets, together with runtimes for the pre-computation of the FTLE grid as well as for the ridge extraction using our particle system (PS) and the Marching Ridges (MR) approach. Timings marked with a star (*) have been approximated (see Section 5.2).*

	Setup					Runtimes (min)			
	t_0	$\tau_{\max}$	*itSteps*	FTLE resolution	Domain	FTLE	PS	MR	Speedup
ABC	0	5	250	$256 \times 256 \times 256$	$[0,\ 2\pi] \times [0,\ 2\pi] \times [0,\ 2\pi]$	9	58	1 741	30
Double Gyre	[0,10]	5	100	$256 \times 128 \times 1280$	$[0,\ 2] \times [0,\ 1] \times [0,\ 10]$	8	17	1 692*	99.5
Half Cylinder	13	2	100	$1152 \times 384 \times 192$	$[-0.4,\ 5] \times [-1.4,\ 1.4] \times [-0.45,\ 0.45]$	77	11	3 622*	329.3
CTBL	0	0.5	50	$256 \times 256 \times 768$	$[4.5,\ 5.5] \times [4.5,\ 5.5] \times [0,\ 3.2]$	7	101	1 318*	13



(a) (b)

Figure 9: *Ridge extraction for the 2D Double Gyre dataset. The additional third dimension represents the start time t_0. This enables the user to interpret results for a varying start and integration time simultaneously. For the animation of the integration time, we refer to the accompanying video. Both our particle system (a) and Marching Ridges (b) show results of similar quality.*

start times t_0 in the interval $[0,\ 10]$:

$$\mathbf{v}((x,y,t_0),\tau) = \begin{pmatrix} -\pi\ A \sin(\pi f(x,t_0+\tau))\cos(\pi\ y) \\ \pi\ A \cos(\pi f(x,t_0+\tau))\sin(\pi\ y)\frac{df}{dx}(x,t_0+\tau) \\ 0 \end{pmatrix},$$

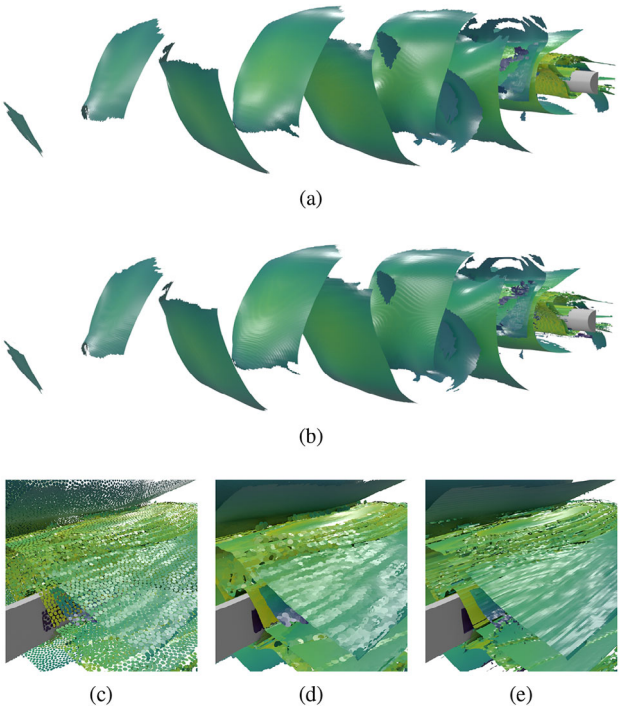
$$f(x,t) = a(t)x^2 + b(t)x,$$

$$a(t) = \varepsilon\ \sin(\omega\, t),\quad b(t) = 1 - 2\varepsilon\ \sin(\omega\, t).$$

We use the parameters $A = 0.1$, $\omega = 0.2\pi$ and $\varepsilon = 0.25$. Due to this adapted definition, we can visualize the dynamics of both the start and the integration time. Figure 9 shows the results of the particle-based ridge extraction (left) and Marching Ridges (right). Please note that the third dimension depicts the varying start time for the original 2D dataset. Both results are similar in quality—neither method shows any significant advantage over the other one. However, we estimate that our method is approximately 100 times faster than Marching Ridges.

Half Cylinder: This dataset stems from a numerical simulation of an incompressible 3D flow around a half cylinder. The simulation was executed with the *Gerris Flow Solver* [Pop04, BRG19]. As shown in the accompanying video, ridge surfaces evolve early around the half cylinder itself. The larger ridge surfaces only evolve after some time, which is one motivation for inspecting sequences of integration times: We can observe after which time regions of particles separate. Figure 10 presents the visual extractions, which partly shows artifacts due to insufficient FTLE resolution for both approaches. The typical solution for this would be to increase the resolution of the FTLE grid.

Simulated Cloud-Topped Boundary Layer (CTBL): The CTBL dataset is the result of a cloud-resolving boundary layer simulation. It is courtesy of the *German Climate Computing Center* (DKRZ) and the *Max Planck Institute for Meteorology* (MPI-M). The version we use in this paper is provided by Günther [GKT16] and contains the simulation result re-sampled on a regular grid. The dataset describes a cumulus cloud convection simulation that used the *UCLA-Large-Eddy Simulation* (UCLA-LES) [Ste13]. UCLA-LES model solves specific equations for the prognostic variables of wind, liquid water potential temperature, total water mixing ratio, rain mass mixing ratio and rain number mixing ratio. The model focuses on the study of detailed cloud dynamics on high spatial and temporal resolutions. It is used for climate simulations, among other things. The simulation domain is $[0,\ 10] \times [0,\ 10] \times [0,\ 3.2]$. The dataset is challenging because it contains strong turbulence and chaotic behaviour, especially in the centre of the domain. We use a cutout and restrict the xy-domain to $[4.5,\ 5.5]^2$ which enables an analysis of this interesting part. The result of the last iteration for our particle system is shown in Figure 11.



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

Figure 10: *FTLE ridges for the Half Cylinder extracted by our particle system (a) and Marching Ridges (b). In (b), large ridge structures in the centre show similar step-like artifacts as for the ABC flow in Figure 8(e). We compare closeups for the region around the half cylinder. (c) and (d) refer to our particle system with small and large splats, respectively. (e) shows the result of Marching Ridges. In the front region, both approaches produce surfaces without gaps. However, the ridge structure is not entirely flat, which disturbs the visualization for both methods. Directly above the half cylinder, multiple ridges are close to each other, resulting in artifacts of flipping particles and triangles.*

5.3. Sampling densities

For the ridge extractions, we rely on the following two samplings: a spatial sampling for the FTLE grids and a temporal sampling for the integration time series. We analyse the extractions for both Marching Ridges [FP01] and our method with different resolutions, based on the ABC dataset. Especially, the spatial tests enable to estimate potential behaviours of respective adaptive versions (see Section 5.6).

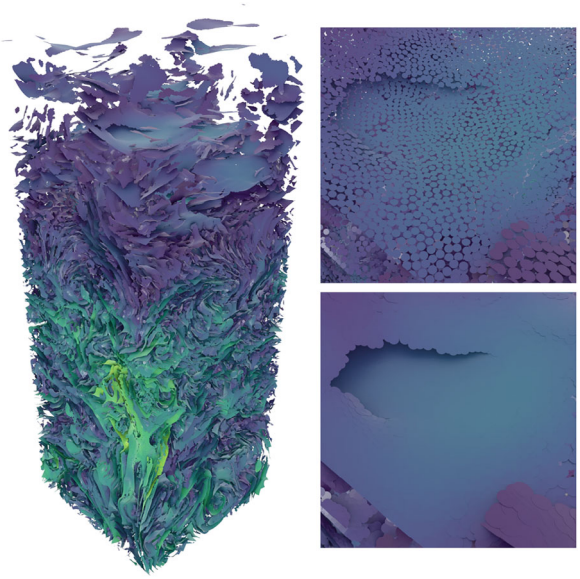


Figure 11: *Result of our particle system for the CTBL dataset. A comparison to Marching Ridges can be found in the accompanying video. We achieve a well-distributed sampling for the densely distributed ridges, even in locations of strong ridges. Although the domain contains a dense set of ridges, which counteracts meaningful definitions of coherent regions, one can still interpret the flow behaviour. For example, in the top regions, there are fewer and flatter ridges,* i.e. *weaker separation. Additionally, most separations in this section are horizontal. At the bottom, one can identify locations of the largest FTLE values, and thus, the strongest separation, as well as multiple vertical ridges.*

Table 3: *Runtimes for the ABC flow with different FTLE field sampling resolutions. Last runtime of Marching Ridges is approximated (see Section 5.2).*

	Runtimes (min)		
Resolution	Ours	Marching Ridges	Speedup
64^3	3	32	10.7
128^3	14	242	17.3
256^3	58	1 741	30
512^3	294	14 527*	49.4

Table 3 and Figure 12 compare the runtimes and visualizations of both methods for different spatial resolutions. For increasing resolutions, the runtime of Marching Ridges grows at a faster rate. This might indicate that the particle system can take even more advantage when implementing an adaptive version, with the ultimate goal of extracting ridges with larger grid resolutions. As already observed in Section 5.2, the extraction qualities of both methods are very similar, even for varying spatial resolutions. Thus, we conclude that the particle system shows the potential for applying adaptive FTLE grids in a manner similar to AMR Marching Ridges [SP07].

For analysing the temporal resolution, we applied our method to the ABC flow with five resolutions: 10, 20, 50, 100 and 250 (with

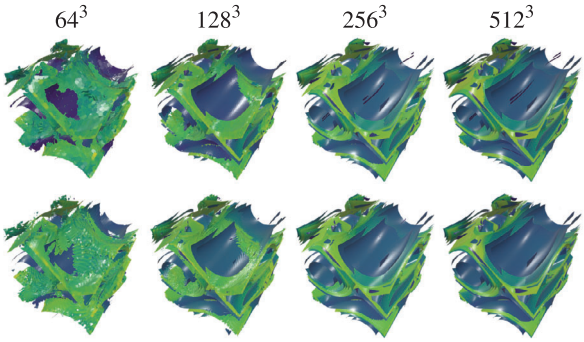


Figure 12: *Extractions of our method (top) and Marching Ridges (bottom) for different resolutions of the FTLE grid.*

accordingly adapted values for *initPeriod*). Visual extractions and numbers can be found in the accompanying video. The population size for the final integration time stayed stable for all extractions, actually with the most particles for the lowest resolution. Thus, we conclude that our system reliably extracts all ridges, even when using lower resolutions for the integration time. However, this has the following two drawbacks: First, low resolutions trivially provide fewer information and less fluid animations; second, the quality of the particle distribution is more uniform for larger resolutions as particles attract and repel each other more often.

5.4. Visualization and sampling quality

Our particle system and Marching Ridges differ in various aspects including their visualization techniques. Marching Ridges extracts a set of triangles which can be visualized directly. Our method generates a set of particles, which we currently render as surface splats. In regions of low surface curvature, this gives good visual results. In some cases, this can even lead to a smoother surface visualization than for Marching Ridges, *e.g.* for the large surfaces in the centre of Figures 10(a) and 10(b). Our system successfully achieves a dense and uniform sampling of the surfaces, see Figures 8(c), 10(c) and 11. However, in regions of high curvature or ridges that are located close to each other, the visualized surface may appear discontinuous, *i.e.* individual particles can be identified due to 'gaps' in-between. Marching Ridges similarly suffers from artifacts of nearby ridges, which can cause jittering triangles. Figures 10(d) and 10(e) show these problems. These artifacts are due to undersampling and a general problem of FTLE ridge extraction as the extraction significantly depends on the resolution of the FTLE field. For a fixed grid resolution, ridges eventually come so close that they cannot be distinguished anymore. Additionally, the particle system requires a few time steps to achieve a uniform distribution. Thus, lower temporal resolutions lead to a little worse results.

5.5. Performance

One key advantage of our approach is its efficiency. In all of our experiments, we clearly outperform Marching Ridges (see Section 5.2). Table 2 shows runtimes and speedups for the flow datasets. We gained the largest performance advantage for the Half Cylinder

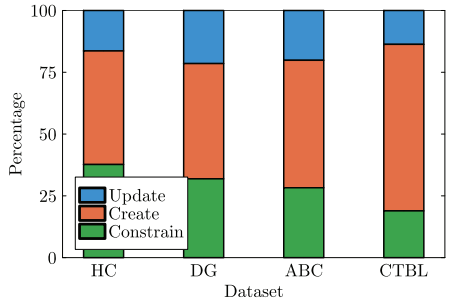


Figure 13: *Relative runtimes for the stages of our particle system for the Half Cylinder (HC), Double Gyre (DG), ABC and CTBL flows. Datasets are ordered from small (left) to large (right) final population size, which correlates to the runtime (see Section 5.2).*

flow with a speedup factor of over 300. The smallest advantage was obtained for the CTBL dataset, with a factor of 13. Note that these are also the datasets with smallest and largest densities of ridges, respectively. Figure 13 gives evidence that with increasing density of ridges in the domain, the runtime proportions of population constraining shrink and of neighbour creation grow. In comparison to Marching Ridges, we observed a mean speedup factor of over 110. As shown in Section 5.3, increasing the resolution of the FTLE grid increases our runtime advantage even further.

5.6. Comparison to AMR Marching Ridges

As mentioned before, Sadlo and Peikert [SP07] extend standard Marching Ridges with AMR. Albeit not comparing to their method in detail in this work, we propose a theoretical comparison. AMR Marching Ridges applies standard Marching Ridges iteratively to a fixed integration time. In each iteration, the resolution of the FTLE grid is locally increased in regions where ridges were found. The procedure stops as soon as a pre-defined maximum resolution is achieved. This prevents expensive FTLE computations in regions where probably no ridges exist while achieving similar results to standard Marching Ridges (with same maximum resolution). Thus, in terms of visual quality and accuracy, our prior findings do also apply to AMR Marching Ridges.

The runtime of AMR Marching Ridges can differ significantly to standard Marching Ridges which mainly scales with the selected grid resolutions. The AMR method, on the other hand, highly depends on the existing FTLE ridges inside the domain. It can take most advantage in cases with many empty regions as large parts of the domain will only be sampled sparsely. Contrary, standard Marching Ridges and our method always pre-compute dense FTLE grids for all time steps. This advantage of the AMR method can certainly achieve comparable or even better runtimes than our particle system, *e.g.* for the Half Cylinder dataset. However, the usage of AMR also creates an overhead as it uses multiple iterations, each testing for ridges and controlling further sub-divisions. Beyond that, AMR also computes samples of the FTLE field on runtime. Since their method has to extract ridges for each integration time step independently, common parts of pathlines may be computed several times. This is computationally expensive and further increases the total runtime. Thus, AMR shows conceptual disadvantages for an application to series of integration times. For flows with many FTLE ridges (*e.g.* the CTBL dataset), AMR might even be a disadvantage.

As described, the AMR application in Sadlo and Peikert [SP07] is not ideal for iterating over multiple time steps. A meaningful application requires additional mechanics for dynamically increasing or decreasing grid resolutions depending on ridge movements over multiple iterations. The general idea of adaptive FTLE sampling in regions where ridges exist can also be transferred to our particles. We see such advancements as a challenging, but solvable problem for future work.

5.7. Limitations and future work

The main limitation for our approach is the same as for many approaches to the extraction of FTLE ridges: It strongly depends on the discretization of the FTLE grid. In Section 5.4, we discuss the existence of artifacts for regions with closely located ridges. Previous work has shown that implementing adaptive ridge extraction can greatly improve accuracy and decrease runtimes [SP07, BT13]. As discussed in Sections 5.3 and 5.6, we are convinced that our method can also be extended to incorporate AMR. However, this extension is not trivial as our constraining method not only depends on the FTLE grid resolution but also on the temporal one: Adaptation requires a strategy for refining the FTLE grid over increasing integration time. This poses a reasonable challenge.

Other possible areas of improvement include visualization and ridge filtering. We see potential to improve the sampling quality for newly emerging ridges. However, one has to find a balance between quality and performance. Another idea is to apply clustering for extracting not only a population of particles, but a set of distinct ridges. This offers multiple chances, *e.g.* filtering and outlier detection, which can improve the results by removing less relevant ridges or artifacts. Another idea is replacing direct particle visualization with splats by surface re-construction that generates a triangulation. This may help to prevent problems as shown in Figure 10(d).

6. Conclusions

We presented a new method for the extraction of FTLE ridge geometry in three-dimensional space, *i.e.* we generally re-construct two-manifolds in 3D. Our approach utilizes a particle system and allows for efficient re-construction of ridge surfaces at sequences of multiple integration times. Visualizing the evolution of ridges over time reveals more of the flow dynamics and gives a better insight into the data and their interpretation and understanding. As our method 'reuses' information collected at earlier integration times, it is efficient for generating time series of ridges. This is the main advantage over the standard Marching Ridges [FP01] techniques: Our experiments show that our method performed significantly and consistently faster. We see various directions for future work like improving the visual representation, *e.g.* by using appropriate clustering methods, or increasing accuracy as well as performance by extending the particle system towards an adaptive sampling that is, *e.g.* guided by local curvature or distance to other ridges.

Acknowledgements

This work was partially supported by DFG Grant TH 692/21-1.

Open access funding enabled and organized by Projekt DEAL.

Conflicts of Interest

All authors declare that they have no conflict of interest.

References

[BRG19] Baeza Rojo I., Günther T.: Vector field topology of time-dependent flows in a steady reference frame. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2019), 280–290.

[BT13] Barakat S. S., Tricoche X.: Adaptive refinement of the flow map using sparse samples. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2753–2762.

[Ebe96] Eberly D.: *Ridges in Image and Data Analysis* (vol. *7*). Springer Science & Business Media, Dordrecht, 1996.

[FP01] Furst J. D., Pizer S. M.: Marching ridges. In *Signal and Image Processing* (2001).

[GGTH07] Garth C., Gerhardt F., Tricoche X., Hans H.: Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE TVCG 13*, 6 (2007), 1464–1471.

[GHP*16] Guo H., He W., Peterka T., Shen H.-W., Collis S. M., Helmus J. J.: Finite-time Lyapunov exponents and Lagrangian coherent structures in uncertain unsteady flows. *IEEE Transactions on Visualization and Computer Graphics 22*, 6 (2016), 1672–1682.

[GKT16] Günther T., Kuhn A., Theisel H.: MCFTLE: Monte Carlo Rendering of Finite-Time Lyapunov Exponent Fields. *Computer Graphics Forum 35*, 3 (2016), 381–390.

[Hal01] Haller G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D: Nonlinear Phenomena 149*, 4 (2001), 248–277.

[Hal15] Haller G.: Lagrangian coherent structures. *Annual Review of Fluid Mechanics 47* (2015), 137–162.

[Hec97] Heckbert P.: Fast surface particle repulsion. In *SIGGRAPH'97, New Frontiers in Modeling and Texturing Course* (1997), pp. 95–114.

[HH91] Helman J. L., Hesselink L.: Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications 11*, 3 (1991), 36–46.

[HS21] Hofmann L., Sadlo F.: Local extraction of 3D time-dependent vector field topology. *Computer Graphics Forum 40* (2021), 111–122.

[HY00] Haller G., Yuan G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D: Nonlinear Phenomena 147*, 3-4 (2000), 352–370.

[HYX*20] Hang H., Yu B., Xiang Y., Zhang B., Liu H.: An objective-adaptive refinement criterion based on modified ridge extraction method for finite-time Lyapunov exponent (FTLE) calculation. *Journal of Visualization 23* (2020), 81–95.

[KESW09] Kindlmann G. L., Estépar R. S. J., Smith S. M., Westin C.-F.: Sampling and visualizing creases with scale-space particles. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1415–1424.

[KRWT12] Kuhn A., Rössl C., Weinkauf T., Theisel H.: A benchmark for evaluating FTLE computations. In *2012 IEEE Pacific Visualization Symposium* (2012), pp. 121–128.

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics 21*, 4 (1987), 163–169.

[LDK*19] Lermusiaux P. F., Doshi M., Kulkarni C. S., Gupta A., Haley P., Mirabito C., Trotta F., Levang S., Flierl G., Marshall J., Peacock T., Noble C.: Plastic pollution in the coastal oceans: Characterization and modeling. In *Oceans 2019 MTS/IEEE Seattle* (2019), IEEE, pp. 1–10.

[LM10] Lipinski D., Mohseni K.: A ridge tracking algorithm and error estimate for efficient computation of Lagrangian coherent structures. *Chaos: An Interdisciplinary Journal of Nonlinear Science 20*, 1 (2010), 017504.

[LVHV*23] Lopes P., Van Herck P., Verhoelst E., Wirix-Speetjens R., Sijbers J., Bosmans J., Vander Sloten J.: Using particle systems for mitral valve segmentation from 3D transoesophageal echocardiography (3D toe)—a proof of concept. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization 11*, 1 (2023), 112–120.

[MGW05] Meyer M. D., Georgel P., Whitaker R. T.: Robust particle systems for curvature dependent sampling of implicit surfaces. In *SMI'05: International Conference on Shape Modeling and Applications 2005* (2005), IEEE, pp. 124–133.

[MNKW07] Meyer M., Nelson B., Kirby R., Whitaker R.: Particle systems for efficient and accurate high-order finite element visualization. *IEEE TVCG 13*, 5 (2007), 1015–1026.

[NJCBP*18] Nardelli P., Jimenez-Carretero D., Bermejo-Pelaez D., Washko G. R., Rahaghi F. N., Ledesma-Carbayo M. J., Estépar R. S. J.: Pulmonary artery–vein classification in CT images using deep learning. *IEEE Transactions on Medical Imaging 37*, 11 (2018), 2428–2440.

[NWMC21] Nguyen D., Wu P., Monico R. O., Chen G.: Dynamic mode decomposition for large-scale coherent structure extraction in shear flows. In *IEEE TVCG* (2021).

[Pop04] POPINET S.: Free computational fluid dynamics, vol. *2*. ClusterWorld. http://gfs.sf.net/ (2004). Accessed 27 September 2023

[RGG20] ROJO I. B., GROSS M., GÜNTHER T.: Accelerated Monte Carlo rendering of finite-time Lyapunov exponents. *IEEE TVCG 26*, 1 (2020), 708–718.

[SFB*12] SCHINDLER B., FUCHS R., BARP S., WASER J., POBITZER A., CARNECKY R., MATKOVIĆ K., PEIKERT R.: Lagrangian coherent structures for design analysis of revolving doors. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2159–2168.

[Sha11] SHADDEN S. C.: Lagrangian coherent structures. In *Transport and Mixing in Laminar Flows: From Microfluidics to Oceanic Currents*. Wiley-VCH, Weinheim (2011), pp. 59–89.

[SJS20] SAGRISTÀ A., JORDAN S., SADLO F.: Visual analysis of the finite-time Lyapunov exponent. *Computer Graphics Forum 39* (2020), 331–342.

[SLM05] SHADDEN S. C., LEKIEN F., MARSDEN J. E.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena 212*, 3-4 (2005), 271–304.

[SP07] SADLO F., PEIKERT R.: Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. *IEEE TVCG 13*, 6 (2007), 1456–1463.

[SPFT12] SCHINDLER B., PEIKERT R., FUCHS R., THEISEL H.: Ridge concepts for the visualization of Lagrangian coherent structures. In *Topological Methods in Data Analysis and Visualization II*. Mathematics and Visualization. Springer, Heidelberg (2012), pp. 221–235.

[SRP11] SADLO F., RIGAZZI A., PEIKERT R.: Time-dependent visualization of Lagrangian coherent structures by grid advection. In *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*. Springer, Berlin (2011), pp. 151–165.

[ST92] SZELISKI R., TONNESEN D.: Surface modeling with oriented particle systems. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (1992), pp. 185–194.

[Ste13] STEVENS B.: Introduction to UCLA-LES. Version 3.2.1. 2013.

[TCH10] TANG W., CHAN P. W., HALLER G.: Accurate extraction of Lagrangian coherent structures over finite domains with application to flight data analysis over Hong Kong international airport. *Chaos: An Interdisciplinary Journal of Nonlinear Science 20*, 1 (2010), 017502.

[WH94] WITKIN A. P., HECKBERT P. S.: Using particles to sample and control implicit surfaces. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (1994), pp. 269–277.

[WRT18] WILDE T., RÖSSL C., THEISEL H.: FTLE ridge lines for long integration times. In *2018 IEEE Scientific Visualization Conference (SciVis)* (2018), IEEE Computer Society, pp. 57–61.

[XLT23] XI Y., LUAN W., TAO J.: Neural Monte Carlo rendering of finite-time Lyapunov exponent fields. *Visual Intelligence 1* (June 2023), 10.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Supplemental Video 1