Adaptive Refinement of the Flow Map Using Sparse Samples

Samer S. Barakat and Xavier Tricoche, *Member, IEEE*

Abstract— We present a new efficient and scalable method for the high quality reconstruction of the flow map from sparse samples. The flow map is a fundamental concept in the analysis of flow phenomena and all so-called Lagrangian flow visualization techniques require its approximation. Specifically, the flow map is generally obtained by integrating a dense 2D or 3D set of particles across the domain of definition of the flow. Despite its embarrassingly parallel nature, this computation creates a performance bottleneck that existing adaptive techniques alleviate only partially. Our iterative approximation method significantly improves upon the state of the art by precisely modeling the flow behavior around automatically detected geometric structures embedded in the flow, thus effectively restricting the sampling effort to interesting regions. Our data reconstruction is based on a modified version of Sibson's scattered data interpolation, which allows us at each step to offer an intermediate dense approximation of the flow map and to seamlessly integrate regions that will be further refined in subsequent steps. We present quantitative and qualitative results for our method on different types of flow data sets and provide a comparison with existing techniques.

1 Introduction

Fluid flows are part of countless physical and biological phenomena. Their precise understanding is therefore important to many endeavors in science, engineering, and medicine. Computational fluid dynamics (CFD) simulations have become an essential means to investigate and predict a broad range of flow behaviors. They produce increasingly large datasets that require massive storage resources and, yet more significantly, computational resources for the analysis of this data.

Scientific visualization has contributed a wide range of powerful techniques to depict the salient properties of fluid flows datasets [19, 26]. In particular, the characterization and extraction of remarkable structures has been the focus of an intense research effort in topological [20] and feature-based [27] methods. In recent years, the notion of Lagrangian coherent structure (LCS) has emerged as a compelling alternative to study the qualitative behavior of transient flows. At the core of the LCS definition lies the concept of flow map, which is key to the Lagrangian analysis of any fluid flow and its associated transport properties. Unfortunately, the estimation of the flow map poses a fundamental computational challenge, namely it requires to integrate the trajectories of massless particles from a dense set of spatial locations. In unsteady flows, this advection implies that multiple time steps be fetched into memory thus increasing the I/O transfer time. In addition, due to the nature of the flow structures, this computation needs to be carried on along different spatial and temporal scales.

Several adaptive refinement techniques have been proposed to approximate the flow map or quantities derived from it. Yet, despite these contributions, characterizing and extracting structural features from large-scale CFD datasets remains very difficult. In particular, existing approximation schemes yield poor results when applied to the nonlinear and multi-scale characterization of salient structures from the flow map.

In this work, we propose a massively parallel and adaptive technique for the sampling and reconstruction of the flow map from a *sparse* set of trajectories. Specifically, this work makes two main contributions. First, we propose a new model to explicitly detect and describe sharp features in the flow map, and we introduce a reconstruction technique based on an extension of Sibson's interpolation to account for the steep changes associated with these features. Existing methods for signal approximation are highly dependent on the structure of the sample points and they typically lead to poor approximation quality in the vicinity of sharp features. As a consequence, flow structures extracted from an approximated flow map signal (e.g. LCS) exhibit significant errors and artifacts due to their correlation with sharp boundaries in the flow map. Our second contribution corresponds to a new adaptive sampling strategy that consistently reduces reconstruction errors to deliver an approximation quality that clearly outperforms existing refinement techniques using the same number of samples.

• *S. Barakat and X. Tricoche are with Computer Science Department, Purdue University, E-mail: sbarakat@purdue.edu, xmt@purdue.edu.*

2 Related Work

Different topological and geometrical approaches for flow visualization have been explored by the visualization community [21]. While the Eulerian perspective underlying topology is primarily relevant to the characterization of instantaneous flow features [38, 39], Lagrangian techniques advect particles along the flow to identify transient patterns. One category of these techniques requires the advection of a spatially limited set of particles (seeding set) and the computation of intermediate locations across time such as stream lines/surfaces [**?**] and path surfaces [**?**, **?**]. If only a sparse set of the particles is advected and used to approximate the flow map at the intermediate time steps, this can cut on the overall cost of the advection and the associated velocity data transfers.

Another category of Lagrangian techniques include LCS based methods which require particle advections covering the entire spatial domain of the analysis. Lagrangian coherent structures have attracted significant attention in both fluid dynamics [10, 35], and visualization communities [31, 8, 32, 18]. Haller has shown in his pioneering work in 2001 [13] that LCS can be obtained as the ridges of the Finite-Time Lyapunov Exponent (FTLE), a scalar field that characterizes the amount of stretching about the trajectory of a point over a finite time interval. This research was followed by Haller's work on the variational characterization of LCS [14] which provides a more sound and accurate definition of flow separation structures. Both definitions rely on the computation of the flow map.

Garth *et al.* [9] proposed a method for the acceleration of FTLE computation for 2D flows by mapping the particle advections to the GPU. The approximation of the flow map was addressed by Garth *et al.* [8] using an adaptive regular refinement technique. They used two discretization sequences at consecutive resolution levels to assess the quality of the approximation. New samples are acquired at sequence locations of unsatisfactory approximation quality. Agranovsky *et al.* [1] proposed a method to compute the FTLE field from scattered flow map samples using Moving Least Squares (MLS). They suggested using staggered sampling to eliminate the possibility of a skewed MLS neighborhood. Both regular refinement and MLS fitting neighborhood requirements poses a lower bound on the number of samples that could be used. Brunton *et al.* [5] proposed a technique that exploits the similarities between trajectories of a sequence of flow maps over time to speed up the computation. A germane idea was applied by Hlawatsch *et al.* [15] who introduced a new hierarchical computation scheme for integral curves and described a GPU implementation. A sequence of regular grids is used to approximate the flow map over small time intervals. Both approaches trade computational

complexity for increased memory footprint, which is tied to the spatial and temporal resolution. Investigating fine scales can become an expensive task especially for time-dependent data sets.

For FTLE approximation, Sadlo and Peikert [30] proposed a method to adaptively refine grid cells based on the presence of FTLE height ridges. Least squares is used to approximate the gradient and Hessian at the sample points. Continuous changes in points neighborhoods due to consecutive small refinements at the boundary of the ridges forces excessive re-evaluation of the ridge criteria. This introduces an inter-point dependency that limits efficiency on parallel architectures. Barakat *et al.* [4] discussed a view-dependent method for the interactive refinement of FTLE. The visual feedback drives a dynamic hierarchical refinement strategy that optimizes the use of memory and computational resources. Also, Chen *et al.* [7] attempted to reduce the computation time of the flow map by identifying appropriate minimal integration durations.

The flow map contains a complex set of edge structures making its approximation difficult. Unfortunately, it is at these sharp features that the accurate reconstruction of the flow map is needed most. This is expected since sharp boundaries highlight significant differences in the behavior of the trajectories of neighboring particles. The detection of jump discontinuities has been discussed before in the literature of signal approximation. A variety of approaches based on radial basis functions [17], variations cubic splines [37], wavelets [29] and explicit representation of the value transition structures [3] [2] have been proposed. A reconstruction of the flow map from data obtained through a coarse sampling strategy must avoid introducing approximation artifacts or distorting the continuity of these boundaries as they reflect on the quality of structures extracted using the flow map in a later stage. In this paper, we introduce a new method that could be considered as an extension of explicit techniques. Our method however models discontinuities differently to ensure smoothness for both 2D and 3D signals.

3 Flow Map Computation and Interpolation

In this section, we briefly introduce the basic concepts that are necessary to understand the steps involved in the computation of the flow map and its Jacobian. We also introduce the smooth C^1 interpolant used by our method for the approximation of the flow map.

3.1 Flow Map

Let $\mathbf{v} : (I \subset \mathbb{R}) \times (D \subset \mathbb{R}^3) \rightarrow \mathbb{R}^3$ be a smooth time-dependent three-dimensional vector field defined over a spatial domain D and a time interval I describing the velocity of a fluid flow. The corresponding dynamical system describes the motion of massless particles along the flow:

$$\begin{cases} \dot{\mathbf{x}}(t, t_0, \mathbf{x_0}) &= \mathbf{v}(t, \mathbf{x}(t, t_0, \mathbf{x_0})) \\ \mathbf{x}(t_0, t_0, \mathbf{x_0}) &= \mathbf{x_0}. \end{cases} \tag{1}$$

The map $\mathbf{x}(\cdot, t_0, \mathbf{x_0}) : t \mapsto \mathbf{x}(t, t_0, \mathbf{x_0})$ describes a particle *trajectory*. The map $\mathbf{x_t} := \mathbf{x}(t, t_0, \cdot)$ is called *flow map* $\mathbf{x_t}(\mathbf{x_0})$: it indicates the position reached at time t by a particle released at $\mathbf{x_0}$ at time t_0.

With previous notations, one considers the spatial variations of the flow map $\mathbf{x}_t$, whereby $t = t_0 + \tau$ and τ is finite. The variations of this flow map around a given position $\mathbf{x_0}$ are locally determined by its spatial derivative, the matrix $\mathbf{J_x}(t, t_0, \mathbf{x_0}) := \nabla_{\mathbf{x_0}} \mathbf{x}(t, t_0, \mathbf{x_0})$ at $\mathbf{x_0}$. However, it is also possible to compute the flow map Jacobian using a single advection trajectory [34]. Since

$$\frac{\partial \mathbf{x_t}(\mathbf{x_0})}{\partial t} = \mathbf{v}(t, \mathbf{x_t}(\mathbf{x_0}))$$

the derivative with respect to the initial start position $\mathbf{x_0}$ can be formulated as

$$\frac{\partial}{\partial \mathbf{x_0}} \frac{\partial \mathbf{x_t}(\mathbf{x_0})}{\partial t} = \frac{\partial}{\partial \mathbf{x_0}} \mathbf{v}(t, \mathbf{x_t}(\mathbf{x_0}))$$

by applying the chain rule and changing the order of the derivation on the left hand side we get

$$\frac{\partial}{\partial t} \frac{\partial \mathbf{x_t}(\mathbf{x_0})}{\partial \mathbf{x_0}} = \frac{\partial}{\partial \mathbf{x_t}(\mathbf{x_0})} \mathbf{v}(t, \mathbf{x_t}(\mathbf{x_0})) \frac{\partial \mathbf{x_t}(\mathbf{x_0})}{\partial \mathbf{x_0}}$$

$$= \nabla_{\mathbf{x_t}(\mathbf{x_0})} \mathbf{v}(t, \mathbf{x_t}(\mathbf{x_0})) \frac{\partial \mathbf{x_t}(\mathbf{x_0})}{\partial \mathbf{x_0}}.$$

where $\nabla_{\mathbf{x_t}(\mathbf{x_0})} \mathbf{v}(t, \mathbf{x_t}(\mathbf{x_0}))$ is the spatial derivative of the vector field. Hence,

$$\frac{\partial}{\partial t} \mathbf{J_x}(t, t_0, \mathbf{x_0}) = \nabla_{\mathbf{x_t}(\mathbf{x_0})} \mathbf{v}(t, \mathbf{x_t}(\mathbf{x_0})) \mathbf{J_x}(t, t_0, \mathbf{x_0}) \tag{2}$$

The ordinary differential equation (ODE) in equation 2 can be solved alongside the ODE in equation 1 by integrating a higher-dimensional ODE of both the location and the Jacobian entries for each particle advection.

3.2 Sibson's C^1 Continuous Interpolant

The natural neighbor interpolation was first discussed by Sibson [36] to provide a C^1 interpolation of multivariate scattered samples. The method has the advantage of creating a smooth approximation that can reproduce spherical quadrics using a Voronoi tessellation of the scattered sites. The natural neighbor coordinates of a point are defined from the Voronoi diagram of the data sites. The natural coordinate of a point $\mathbf{x}$ with respect to a data site $\mathbf{p_i}$ corresponds to the volume taken by the Voronoi cell of the point $\mathbf{x}$ from the volume of the Voronoi cell of the data site $\mathbf{p_i}$ if the point $\mathbf{x}$ is to be inserted in the Voronoi diagram. This volume is referred to as the natural coordinate $\lambda_i(\mathbf{x})$.

Sibson [36] proposed a C^1 continuous interpolation method that relies on both the function value and its gradient for all sample points $\mathbf{p_i} \in P$. The method can exactly reproduce spherical quadrics of the form $\Phi(x) = a + b^t \mathbf{x} + \gamma \mathbf{x}^t \mathbf{x}$. The proof assumes that the gradient of Φ at the data points is known or can be approximated from the function values.

Sibson's Z^1 interpolant is a combination of the linear interpolant $Z^0 = \sum_i \lambda_i(\mathbf{x}) z_i$, where z_i are the input values, and an interpolant ξ which is the weighted sum of the first degree functions:

$$\xi_i(\mathbf{x}) = z_i + \mathbf{g_i}^t(\mathbf{x} - \mathbf{p_i}),\ \xi(\mathbf{x}) = \frac{\sum_i \frac{\lambda_i(\mathbf{x})}{||\mathbf{x}-\mathbf{p_i}||} \xi_i(\mathbf{x})}{\sum_i \frac{\lambda_i(\mathbf{x})}{||\mathbf{x}-\mathbf{p_i}||}} \tag{3}$$

where g_i is the input gradient at $\mathbf{p_i}$. Sibson observed that the combination of Z^0 and ξ reconstructs exactly a spherical quadric if they are mixed as follows:

$$Z^1(\mathbf{x}) = \frac{\alpha(\mathbf{x}) Z^0(\mathbf{x}) + \beta(\mathbf{x}) \xi(\mathbf{x})}{\alpha(\mathbf{x}) + \beta(\mathbf{x})}$$

where

$$\alpha(\mathbf{x}) = \frac{\sum_i \lambda_i(\mathbf{x}) \frac{||\mathbf{x}-\mathbf{p_i}||^2}{f(||\mathbf{x}-\mathbf{p_i}||)}}{\sum_i \frac{\lambda_i(\mathbf{x})}{f(||\mathbf{x}-\mathbf{p_i}||)}} \ \textit{and}\ \ \beta(\mathbf{x}) = \sum_i \lambda_i(\mathbf{x}) ||\mathbf{x} - \mathbf{p_i}||^2$$

and $f(||\mathbf{x} - \mathbf{p_i}||) = ||\mathbf{x} - \mathbf{p_i}||^2$. This first order interpolant can take advantage of both values and derivatives of the flow map defined based on sparse samples. The method can also take advantage of parallel architectures as will be discussed in section 6.

4 Adaptive Refinement

In this section we discuss a new refinement strategy designed based on the C^1 continuous Sibson's interpolation method discussed in section 3.2. The authors of this paper are not aware of any previous work that attempts to refine the sampling of fields constructed through Sibson's interpolation. There are no constraints on the locations of the samples and no need to compute additional samples only to preserve

a certain organization. The refinement strategy we propose consists of two steps. In the first step, we compute an indicator of the local error everywhere in the domain. In the second step, we add a fixed number of new samples guided by the error measure. This number is user selected based on the computational and memory resources available for the flow map computation.

4.1 Estimating the Approximation Error

In this section we discuss how an error indicator can be computed using the formula defining the Sibson's interpolation. As described in section 3.2, the C^1 Sibson's interpolation is a weighted combination of the linear interpolation Z^0 and a weighted sum of the first degree function defined in equation (3). The C^1 continuity of the Sibson's interpolation is due to the Z^1 term $\xi(\mathbf{x})$. This term can be viewed as the weighted sum of different value suggestions made by the natural neighbors of the reconstruction site $\mathbf{x}$. The weight of each suggestion is directly proportional to the natural coordinate and inversely proportional to the distance between the natural neighbor and the reconstruction site in consideration. The first order suggestion made by each of the data sites $\xi_i(\mathbf{x})$ assumes a constant gradient in the neighborhood of the data sites. Obviously, this assumption is particularly invalid close to regions of high frequency details where the higher order terms are significant. Therefore, it is expected to see that value suggestions made by the different first order approximations have larger discrepancies close to these regions where details are present.

In order to evaluate the discrepancies between the first order suggestions, we use their weighted variance. Hence our indicator can be formulated as:

$$\varepsilon(\mathbf{x}) = log(S\frac{\sum_i \frac{\lambda_i(\mathbf{x})}{||\mathbf{x}-\mathbf{p_i}||}(\xi_i(\mathbf{x})-\xi(\mathbf{x}))^2}{\sum_i \frac{\lambda_i(\mathbf{x})}{||\mathbf{x}-\mathbf{p_i}||}})$$

We use the factor S to scale the weighted variance values to the range between zero and one. The weights ensure that the effect of a certain data site on the variance is equivalent to its effect on the value of $\xi(\mathbf{x})$. Hence, a data site that is further away from the reconstruction point $\mathbf{x}$ compared to other data sites will have a smaller effect on the variance. A consequence of this weighting is to create a tendency for $\varepsilon(\mathbf{x})$ to increase away from any of the data sites since if the distance between $\mathbf{x}$ and any of the sites is particularly small this will reduce the variance as well. This in turn implies that using the described weighted variance as an oracle for the refinement will automatically avoid oversampling since $\varepsilon(\mathbf{x})$ has a local minimum at each existing site. The weighted variance can be computed alongside with the Sibson's interpolation. The strong skewness of the weighted variance can be rectified through logarithmic damping. Finally, the function $\varepsilon(\mathbf{x})$ computed for different components of the flow map can be aggregated to a single function using a max operator.

4.2 Sampling Distribution

The locations of the new samples are identified by mapping the value $\varepsilon(\mathbf{x})$ at each reconstruction point $\mathbf{x}$ to a probability distribution $P(\mathbf{x})$ that indicates the possibility of selecting it as a new sampling location. $P(\mathbf{x})$ is computed as the probability density function of an exponential distribution with the random variable $(-\varepsilon(\mathbf{x}))$:

$$P(\mathbf{x}) = \lambda e^{-\lambda.K.(-\varepsilon(\mathbf{x}))}$$

The term K is simply a constant scaling factor. The mean of the exponential distribution λ is a user controlled parameter that indicates how the new samples are distributed in the domain between regions of different approximation errors. A low value for λ allows for more samples to be allocated to regions of relatively low approximation error. This permits the discovery of any missing details that were not captured by existing samples so far. A relatively high value for λ attempts to restrict the sampling to regions of high approximation errors. It might be useful to use a low value for λ during the first few iterations. In figure 1(e) and 1(f), we show the effect of different values for λ on the distribution of the new samples.

For efficiency, binning is used with the inverse transform sampling from the exponential distribution. A bin is first selected then a point from the bin is selected assuming all points in the each bin have equal probabilities. We impose an additional constraint on the addition of new samples to avoid the selection of several nearby points with high approximation errors simultaneously. This is done by limiting the number of new samples added to each Voronoi cell. If a new sample is to be added to a cell that already contains k new samples than the additional sample will be retained with a probability $1/(k+1)$ otherwise it will be discarded. Hence, we add a new point at a time in each cell to avoid computing samples that might only have a small effect on the approximation error given other surrounding points have been added already.

5 Flow Map Reconstruction

Sibson's interpolant can be used for the reconstruction of the flow map fields from the scattered data sites and can also be used to compute the field $\varepsilon(\mathbf{x})$ needed for the refinement. Each of the scattered data samples correspond to a single particle trajectory, and is associated with the flow map value and its Jacobian computed as discussed in section 3.1. However, the regular Sibson's interpolation suffers from a critical drawback. The interpolation is sensitive to the distribution of the data sites especially close to regions of sharp gradients. This sensitivity takes the form of artifacts that distorts the shape of the edge features. Also, the first order functions when combined with the high gradient magnitude often found at the surrounding data sites will create a set of min and max critical points not originally present in the scalar topology of the flow map fields. These points will in turn lead to artifacts in the FTLE field computed from the flow map. In this section, we demonstrate how both the approximation error and the structures' artifacts can be reduced through a modified version of Sibson's interpolation that takes advantage of an explicit model of the edge features in the data. In this paper, we model these features as smooth open or closed manifolds.

Our solution for the adaptive refinement and reconstruction consists of multiple phases as shown in figure 3. Initially, we use a coarse set of regular samples for the reconstruction of the flow map using Sibson's method. Edge detection can then take place preceding our edge-based reconstruction that will be described in this section. The reconstruction method passes feedback to the refinement through the indicator of the approximation error ε as discussed in section 4. After the new samples are added through refinement, the process can be repeated again. It is also possible to use our edge-based reconstruction in an exploration scenario to give the user an intermediate visualization free of artifacts before more samples are computed. Above all, this reconstruction can save on the total number of samples needed in order to achieve a certain approximation quality. We begin our explanation by pointing out how the sharp features are detected from the sparse samples and how the corresponding surfaces are modeled. We move next to explaining our modified Sibson's method that takes advantage of the discovered edge features.

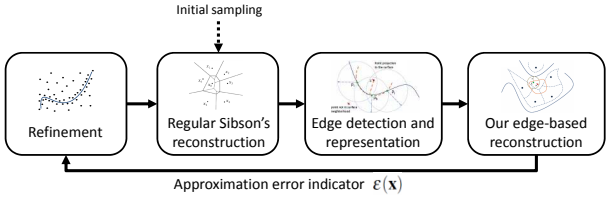
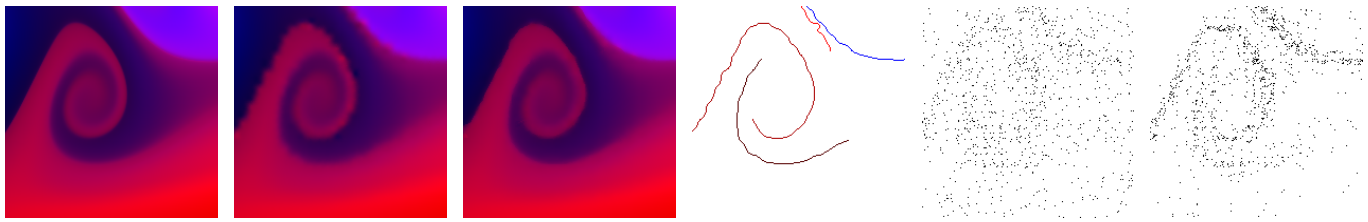


Fig. 3. Diagram showing the different components of our solution.

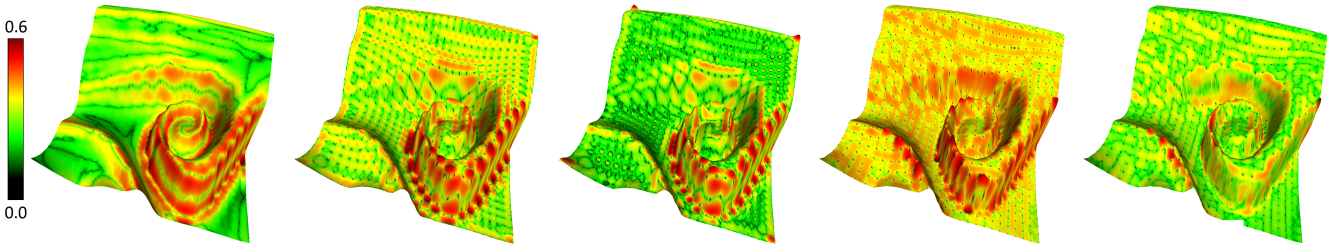
5.1 Sharp Features Detection and Modeling

The edge features can be found by applying a standard edge detection technique to the flow map fields constructed using the regular Sibson's interpolant. We have chosen to apply Canny's edge detection filter [6] for its simplicity in addition to its good localization and robustness.



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

Fig. 1. *2D flow map*: (a) Original high resolution flow map fields, (b) Regular Sibson's reconstruction (MSE is 2.2×10^{-5} and 0.8×10^{-5}), (c) Our feature based reconstruction (MSE is 1.2×10^{-5} and 0.3×10^{-5}), (d) Edge features, (e) Refinement new samples with $\lambda = 2.0$, and (f) Refinement new samples with $\lambda = 4.0$.



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

Fig. 2. The approximation errors for different interpolation techniques: (a) Moving least-squares fit, (b) Multiquadircs radial basis function, (c) Thinplate spline radial basis function, (d) Regular Sibson's interpolation, and (e) Our edge based reconstruction.

The localization property means that identified edges are the closest possible to the real edges in the 3D image. Canny's edge detection filter applies hysteresis thresholding to the gradient magnitude in order to prevent the 3D image noise from creating false edges. We estimate the values of the upper and lower thresholds through a heuristic that maps each to a percentage of the voxels in the data. These percentages are user specified and should reflect the density of edge structures in the data. Binning, based on the gradient magnitude, is used to match the percentages to the threshold values.

The steep changes in the flow map have the shape of smooth surfaces that capture the separation of neighboring particles due to their advection. However, the edges discovered are likely to be non-smooth and noisy due to undersampling as seen in figures 1(d) for a magnified region selected from a 2D turbulent flow. Our goal is to build a smooth surface representation of the edge features through fitting. It is not computationally efficient to use all the edge points for the fitting. It is also not desirable because a smooth reconstruction of the feature surface will become a much harder task in the presence of possibly a large number of noisy edge points. We therefore identify the subset of edge points where each fall closest to at least one site compared to all other edge points. We pair each of these edge points to its closest data site. For each connected edge, this subset of edge points are used for the fitting of a single surface that topologically approximates the feature.

To create a surface representation of the edge features while preserving the smoothness property, we use a variation of the moving least-squares (MLS) method. Specifically, we use the Algebraic Point Set Surfaces (APSS) method described by Guennebaud and Gross [12]. Our main motivation here is to use a normal constrained smooth fitting of the sharp features in the data. The normals are computed as the normalized gradients at the fitting edge points. The value transition itself is tangent to the level sets in its infinitesimal close neighborhood. These level sets are by definition orthogonal to the gradient of the field. By constraining the shape of the value transition to be orthogonal to the gradient, we basically increase its consistency with the known values and gradients at the samples in its neighborhood.

The APSS fitting defines a *potential function* corresponding to the signed distance to the surface. The domain of this function is bounded to the surface neighborhood defined through a set of spheres centered at the fitted edge points. Hence, each edge point is equipped with a fitting radius. To find the potential at a certain query point all edge points with spheres containing the point in question will be used for the fitting. This radius needs to be large enough to ensure the smoothness of the surface but not large to the point where the topology is affected by the excessive smoothing. Therefore, we set the radius to twice the distance between the edge point and its furthest natural neighbor as computed from the preceding regular Sibson's reconstruction. Thus, the radius is directly dependent on the density of the samples in its neighborhood. One problem arises however when computing the potential function at the boundaries of an open surface. An additional check is needed to ensure that a point is projected orthogonal to the surface and within its boundaries before the potential function is assumed valid. In figure 4, we show the surface fitting at a point x that is located within the radius of three surface edge points. The distance from the point to the surface can be computed as the distance to its projection on the fitting circle. The Eigen algebraic sphere fitting used by the APSS method can be efficiently computed for a large number of points while maintaining a fitting quality close to that achieved by geometric sphere fitting.

5.2 Modified Discrete Sibson's Interpolation

In this section, we explain the two steps of the modified Sibson's interpolation we propose. The first step is the computation of the natural neighbors in the presence of the feature surfaces. The second step consists of adapting the Sibson's interpolant to integrate information derived from the surfaces for the reconstruction.

5.2.1 Natural Neighbors

The Voronoi diagrams of continuous data such as lines and curves in the 2D plane or surfaces in the 3D space have been discussed before in the context of transfinite Sibson's interpolation [11]. It was shown that a continuous Voronoi diagram can be computed for a set of line and/or curve segments in the plane. The natural coordinates of a query point $\mathbf{x}$ remains defined as the volume the potential Voronoi cell of $\mathbf{x}$ steals from existing cells when inserted in the diagram (see figure 5).

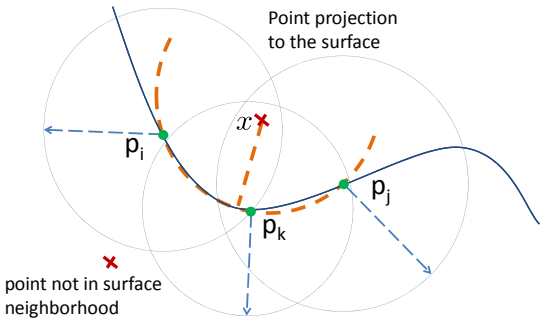


Fig. 4. Sphere fitting of the surface at x using three edge points.

The only difference is that an existing Voronoi cell might belong to a surface instead of a point. The computation of the continuous Voronoi cells is a challenging task however especially with the presence of complex surfaces. We therefore propose an easy extension to the discrete Sibson's interpolation discussed by Park *et al.* [23] to compute the natural neighbors of the reconstruction points in the presence of surfaces.

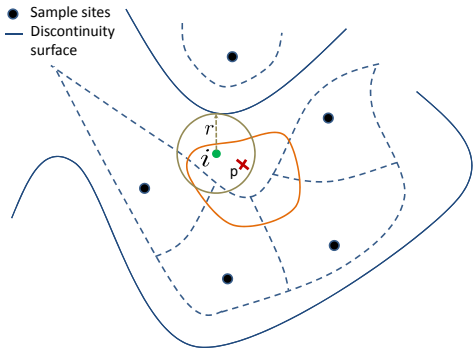


Fig. 5. The Voronoi tessellation in the presence of both curves and data sites. A simulated insertion of a new sample at $\mathbf{p}$ will create a new Voronoi cell taking from the area of the surrounding cells. The area of the pixel $\mathbf{i}$ in the discrete sense will belong to the Voronoi cell of $\mathbf{p}$ because the position $\mathbf{p}$ is inside a circle of radius r around $\mathbf{i}$. Where r is the distance between the point $\mathbf{i}$ and the curve of its cell.

The main idea behind the efficient computation of the natural neighbors as described by Park *et al.* [23] is to scatter the value $V_c(\mathbf{i})$ at point $\mathbf{i}$ to all output raster positions $\mathbf{p}$ that are inside the sphere centered at $\mathbf{i}$ and having a radius of $d(\mathbf{i},V_c(\mathbf{i}))$. The value $V_c(\mathbf{i})$ is the closest data site to the point $\mathbf{i}$. The radius $d(\mathbf{i},V_c(\mathbf{i}))$ is the distance between the point $\mathbf{i}$ and the data site $V_c(\mathbf{i})$). In the presence of surfaces, $V_c(\mathbf{i})$ can be either a data site or a surface. When $V_c(\mathbf{i})$ is a surface the function $d(\mathbf{i},V_c(\mathbf{i}))$ is the distance between the point $\mathbf{i}$ and the surface (the absolute potential function). For the regular discrete Sibson's interpolation, the closest site $V_c(\mathbf{i})$ can be found using a kd-tree containing all the sites. In our case, we also need to find the distance between $\mathbf{i}$ and each surface. Computing the potential function with the associated MLS fitting at each raster position $\mathbf{i}$ for all surfaces would be a time consuming task especially for large volumes. We can instead use the natural neighbors from the regular Sibson's interpolation to identify if the point $\mathbf{i}$ is close to any surface. If none of the natural neighbors of the point from the regular Sibson's method is contributing to a particular surface (not paired with any fitting edge point) then the potential function does not need to be evaluated. This is true because the surface can't be a natural neighbor of the point if none of its original natural neighbors is among the surface sites. The computation of the natural neighbors is summarized in algorithm 1.

Algorithm 1: The computation of the natural neighbors

```
Input: The natural neighbors from the regular Sibson's
       interpolation method oc
Input: Map from sites to surfaces Q
Output: The normalized natural neighbors coordinates c
Construct a kd-tree form m sites.
foreach output raster position p do
    Initialize c(p) = φ.
end
Execute loop in parallel for different values of i
foreach raster position i do
    Find the closest site p_n in the kd-tree.
    F := φ.
    foreach site s in oc(i) do
        F := F ∪ Q(s).
    end
    foreach surface f in F do
        if d(i, p_n) > d(i, f) then
            p_n := f.
        end
    end
    Calculate r = d(i, p_n).
    foreach raster position p inside a d-dimensional sphere of
    radius r around i do
        if V_c(i) not in c(p) then
            Acquire the lock for p
            if V_c(i) not in c(p) then
                Add V_c(i) to c(p).
            end
            Release the lock for p
        end
        Next statement is atomic
        Increment c(p, V_c(i)) by one.
    end
end
Execute loop in parallel for different values of p
foreach output raster position p do
    Sum all natural coordinates weights to compute n(p).
    Set c(p) = c(p)/n(p).
end
```

5.2.2 Modified Interpolant

Unlike the natural neighbors that correspond to sample sites, a surface natural neighbor does not have a flow map value and gradient assigned to it. However, we still need to compute both $z_i(\mathbf{x})$ and $\xi_i(\mathbf{x})$ for a surface natural neighbor i with natural coordinate λ_i in order to perform the Sibson's interpolation described in section 3.2. The computation of z_i for a query point $\mathbf{x}$ can be done using the data sites surrounding the surface and discussed in section 5.1. These sites provide information about the field in the neighborhood of the surface. We assume that the field dynamics in that neighborhood can be expressed in term of the distance to the surface. Recall that the surface itself is orthogonal to the gradient and hence to the level sets in its infinitesimal neighborhood.

In order to compute the value of z_i, we attempt to fit the values at the data sites relative to the surface. As we compute the potential $v(\mathbf{x})$ for the point $\mathbf{x}$, we gain information about all the data sites that belong to the feature surface (paired with the fitting edge points) and that fall in the neighborhood of the point in question. For the computation of z_i through the fitting of the data relative to the surface, we project these sites to the surface and compute the corresponding potential values. Hence, for a site i located at $\mathbf{p}_i$, we have the value y_i, and the potential v_i. For each site i, we have the following fitting constraint $f(v_i) = y_i$, with weight $r_i = ||\mathbf{x} - \mathbf{p}_i||$. The distance based weights are used to penalize further data sites compared those closer to $\mathbf{x}$.

We use a local circle algebraic fit [28] in the 2D domain of y_i and v_i in order to compute the corresponding value for the potential $v(\mathbf{x})$.

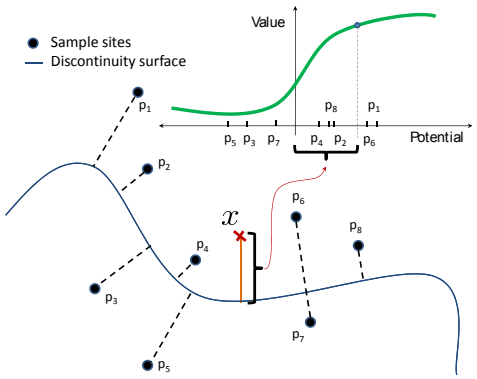


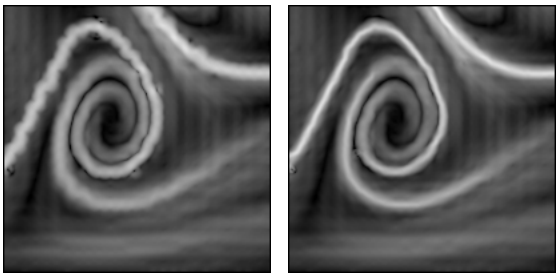
Fig. 6. The edge data sites near x and its natural neighbors are used for the 1D value fit.

This is illustrated in figure 6. We have chosen to use the algebraic fit because of its flexibility in representing the shape of the smooth value transition in addition to its efficiency. It is worth noting that the sample sites of the surfaces are only used for this 1D fitting as they are discarded from the Voronoi diagram. This is because of the large gradient magnitude at these sites that tends to negatively affect the Sibson's interpolation. In addition, these gradients close to the steep transitions in value are often more susceptible to noise. Finally, z_i can be computed as:

$$z_i = f(v(\mathbf{x}))$$

Since $z_i(\mathbf{x})$ provides an estimation of the value exactly at the location $\mathbf{x}$, we do not need the gradient for the first order function $\xi_i(\mathbf{x})$ and we can simply set $\xi_i(\mathbf{x}) = z_i(\mathbf{x})$.

In figure 1, we show a side by side comparison of the 2D flow map approximated using the regular and our modified Sibson's interpolation. The red and blue color channels correspond to the first and second components of the flow map respectively. The difference in approximation quality is particularly clear close to the edge features. The FTLE fields computed from the approximated flow maps is shown in figure 7. The ridges of FTLE which coincide with the edges of the flow map are much clearer and less distorted using our edge-based reconstruction. In figure 2, we demonstrate the approximation quality of our method in comparison to various other existing approximation techniques.



(a) (b)

Fig. 7. *2D FTLE field*: FTLE computed from (a) the regular Sibson's interpolation reconstruction of the flow map, and (b) from our feature based reconstruction.

6 Scalable Implementation

Designing an algorithm that scales to massively parallel architectures is imperative due to the huge sizes of the flow data produced through simulations. Both the flow map reconstruction and the adaptive refinement procedures described in this paper were designed with scalability in mind. For the reconstruction, the investigation and detection of the edges using Canny's filter is done in parallel [22]. The reconstruction time is dominated by the computation of the natural neighbors for both the regular discrete Sibson's interpolation and our method which also requires computing the potential function with respect to the surfaces. A kd-tree data structure is used to keep track of the data samples and to locate the site $V_c((i))$ closest to any raster position $\mathbf{i}$ in parallel as described in algorithm 1. The loop over all raster positions is executed in parallel. However, there are two operations that might lead to a race condition. The first consists of inserting a new natural neighbor for a position $\mathbf{p}$. We therefore have a lock assigned to the natural neighbor list at each position $\mathbf{p}$. The second operation is incrementing the weight of the natural neighbors $n(\mathbf{p}, V_c(\mathbf{i}))$. This operation has to be atomic. This has negligible effect on performance.

The natural neighbor list itself is an unsorted vector where all items' locations in memory remain valid after their insertion despite following insertions. This alleviates the need for read locks that require synchronization between processors causing larger delays compared to a linear search of the list. As more samples are inserted due to refinement, the radius r for raster positions $\mathbf{i}$ decreases and consequently reduces the rasterization cost. The computation of the reconstructed value using the natural neighbors is inherently parallel for all output positions. It is worth noting however, that computing the potential function for the data sites in order to perform the 1D function fitting is a costly operation. Instead, we pre-compute the potential function for all data sites with respect to all surfaces before applying the interpolation. For the adaptive refinement, the integration of different particles tangent the flow velocity is an inherently parallel process. The selection of the new samples locations as described in section 4.2 can be executed in parallel. However, care must be taken while updating the count of new samples in each existing Voronoi cell.

Notice here that the computation of the natural coordinates at any point poses no significant dependency that would limit parallelism. Hence, a large group of samples can be added everywhere in the domain simultaneously through refinement before rasterization is performed in parallel. This is in contrast to approaches that require incremental local reconstruction through MLS fitting intertwined with the repeated computation of FTLE and the corresponding ridge extraction [30] [1]. To achieve a smooth approximation through MLS a large 3D fitting neighborhood is needed in order to avoid skewed situations especially due to the FTLE complexity. As a local small set of new samples are added in a region growing fashion, the MLS fitting and structure investigation need to be repeated for all nodes in the mentioned neighborhood. The fitting might be performed dozens of times at a single node for each new sample added to its domain before the region is considered at sufficient approximation accuracy. These methods also fail to provide an intermediate view of all the structures in the domain and might miss small features. It is worth mentioning that in our case, the limited 1D MLS fitting is only needed in the close neighborhood of the surfaces. This neighborhood is consistently shrinking as new samples are added. Hence, the main difference is that adding new samples reduces the fitting computations rather than increase it.

7 Experiments and Evaluation

A natural and compelling application of our technique is the efficient and scalable computation of Lagrangian coherent structures. These structures can create a portrait of transient flows. The most popular characterization of these structures is based on the concept of FTLE. Note that alternative definitions of LCS that apply a variational approach to their characterization [14] will benefit from our method even more given the high smoothness that they require from the flow map. It is as important as the high-quality reconstruction of the flow map to provide also a high-quality reconstruction of its derivatives since those are directly relevant in the characterization of important flow behaviors.

We have tested our adaptive refinement and reconstruction on five datasets as listed in table 1. The computation of the flow map is performed for a regular grid using a velocity field defined on an unstructured grid. For the steady and analytical test cases, the computation was run on an NVIDIA GTX 590 GPU using CUDA for speed. For

the Delta Wing, the computation was performed using the CPU on a shared memory multiprocessor machine with 64 cores. The same machine was used to document performance numbers for our method. For all datasets used, we compare our refinement and approximation to the four-point adaptive scheme described by Garth *et al.* [8]. The method provides a scalable algorithm for the approximation of the flow map using a four-point subdivision sampling scheme combined with smooth reconstruction kernel. However, the method cannot use gradient information provided at the data sites. We also compare our edge-based approximation to that achieved by the Sibson's interpolation that was adopted to use our refinement strategy. The λ parameter of the refinement has been fixed to 4.0 for all datasets. The computation of the hysteresis thresholds of Canny's edge detection is done based on the percentages listed in table 1 (described in section 5.1).

The first dataset we consider is the standard analytical ABC (Arnold-Beltrami-Childress) flow. The ABC flow is a canonical test case, exhibiting complex turbulent structures [13]. In figure 11, we provide a visual comparison of the FTLE ridges extracted at two different refinement stages of the flow map. Using only 1.2% of the full resolution, our edge-based method was able to extract a smooth surface for the FTLE ridges. The adaptive four-point scheme was less successful in providing an accurate visualization of these structures. The regular hierarchical organization of the samples has a significant effect on the approximation of the flow map gradient close to the surfaces. Our adaptive version of the Sibson's interpolation also suffered from artifacts due to the coarse sampling. The first order term of the Sibson's interpolant combined with the large gradients close to the edge features leads to a high sensitivity to the site locations. At a following refinement stage, we see a considerable improvement in the results achieved by the adaptive four-point scheme and our adaptive version of Sibson. However, the improvement remains behind that achieved by the method presented in this paper. The mean square distance (MSD) relative to the full resolution flow map, shown in figure 10(a), clearly indicates a significant gap between the methods compared in favor of our technique. The refinement method described also leads to a steeper decline in the error especially at the early refinement stages.

The Delta Wing dataset is concerned with the study of the transient flow above a delta shaped wing moving at low speed and an increasing angle of attack. Scientists are mainly interested in the formation of vortices on both sides of the wing and their breakdown. In figure 13, we demonstrate the FTLE ridges extracted using around 4.2% of the full resolution samples. Our method is particularly superior showing structures near the shear surface at the wing front, and near the turbulent vortex in the back. The mean square distance between the original and the approximated flow maps indicates a significant decay in error using the refinement method discussed for the Sibson's based methods. The Delta Wing dataset contains a dense set of structures and the availability of gradient information to the Sibson's methods helped improve the convergence. For the same reason the Sibson's based methods performed poorly when less than 4% of the samples are used. This is because the first order term of the Sibson's interpolant is likely to perform poorly when insufficient samples are available to capture any structures in the flow.

The Ellipsoid dataset is produced through a flow simulation around an ellipsoid. Vortex shedding can be observed as several layers of flow structures at the boundary and above the ellipsoid. The FTLE ridges extracted from the flow map at different refinement stages is shown in figure 12. Our reconstruction leads to smoother surfaces at all refinement stages particularly when only 2% of the full resolution samples are used. In figure 10(c), we can clearly observe a smaller approximation error achieved by our method especially when less than 4% of the full resolution samples are used. Looking to the mean square errors (MSE) of the FTLE field in figure 10(d), the differences between the approximation methods is more obvious (also see figure 14). It is worth noting that even the small differences between the curves in this graph are significant because they mainly correspond to differences around the structures which are only a small percentage of the entire volume.

The Hill's Vortex dataset corresponds to a simple analytic vortex ring model based on Hill's spherical vortex [24]. In figure 10(e), we show a comparison of the different flow map approximations based on the mean square distance. Our edge-based reconstruction is clearly superior. We also notice that the adaptive four-point scheme has a smaller difference to the ground truth than the adaptive Sibson's method for the early iterations. However, the four-point scheme introduces systematic errors due to its inherent sample organization. These errors lead to significant artifacts in the FTLE field as seen in figure 10(f). This is further demonstrated visually in figure 8 (figure 14 for the Ellipsoid dataset). Similar ridge extraction parameters are applied to all test cases. However, we tuned the parameters in favor of the four-point scheme results. Changes to the parameters might reduce the artifacts only at the expense of loosing portions of the structures. It is important to notice, that our approximation explicitly attempts to preserve the gradient directions orthogonal to the surface through the fitting. Hence, the gradients for the flow map fields in our case are better preserved. Since the FTLE computation relies on the Jacobian of the flow map, it is then expected to see differences for ridges extracted from FTLE.

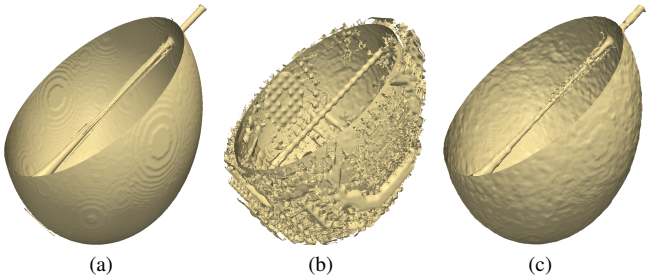


Fig. 8. The FTLE ridges from: (a) The full resolution flow map, (b) adaptive four-point approximation, and (c) our edge-based approximation. The number samples used is 369098 (2.2% of the full resolution).

Another analytical dataset we consider, is the Double Gyre dataset [25]. The reconstruction times for the adaptive Sibson's method and our method is less than three seconds. In figure 9, we compare the FTLE field computed based on the various approximations. We clearly notice the artifacts introduced by the adaptive four-point scheme. Our method is superior especially close to the ridges of FTLE.

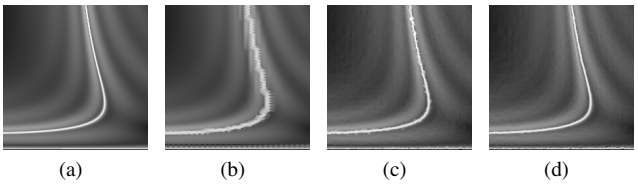


Fig. 9. The FTLE field from: (a) The full resolution flow map, (b) adaptive four-point approximation (FTLE MSE is 2.3×10^{-2}), (c) the adaptive Sibson's approximation (FTLE MSE is 1.2×10^{-2}), and (d) our method (FTLE MSE is 9.2×10^{-3}). The number samples used is 27262 (5.2% of the full resolution).

The computation time needed by the adaptive Sibson's approximation for all datasets is documented in figure 10(h). The cost declines as the number of samples increases due to the decreasing rasterization radius for the points in the refined regions. The computation time of our method is dominated by the need to compute the projection of all data sites to all surfaces. We believe that this cost can be reduced significantly if this projection was limited to those sites close to the structures. This however requires further investigation on the ideal representation to keep track of the domain. Due to the poor edge detection in early refinement stages, the number of surfaces could be

highly affected by fragmented surfaces and leading to high computation times. This cost decreases as the refinement progresses until the number of surfaces is fixed (see figure 10(h)). The computation times for the adaptive four-point method are indicated in table 1. Each refinement iteration for the four-point subdivision scheme requires eight times the time and space of the preceding iteration. We therefore limit the number of iterations to three, and experiment with the threshold parameter until we achieve the needed number of samples. This is impractical in reality since the refinement should be driven by the quality of the approximation making the number of iterations unknown and possibly significantly larger.

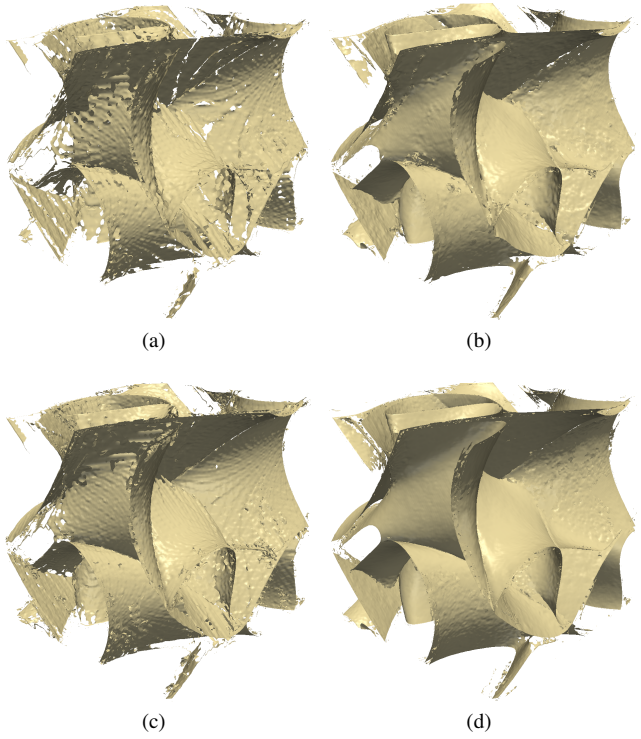


Fig. 11. FTLE ridges for the ABC dataset extracted from the approximated flow map: (a) Adaptive four-point scheme using 201326 samples (1.2% of full resolution), (b) Our method using 201326 samples (1.2% of full resolution), (c) Adaptive four-point scheme using 704643 samples (4.2% of full resolution), (d) Our method with using 704643 samples (4.2% of full resolution).

8 Conclusion and Future Work

In this paper, we proposed to reduce the cost of the flow map computation through approximation. We discussed how to efficiently reconstruct the flow map at a higher resolution using a small fraction of the original samples. This was possible through a model that attempts to explicitly capture edge structures in the flow map and use it to steer the reconstruction. We demonstrated that the careful handling of the Sibson's interpolation near the flow structures can in fact improve the approximation both quantitatively and qualitatively. We have also discussed a new method for the incremental refinement of the approximation using a criteria derived from the interpolation model. The experiments performed on 3D flow datasets demonstrated superiority in preserving the visual quality of the extracted LCS structures based on the approximated flow map compared to the state of the art methods. The results obtained also show that the presented method can consistently reduce the approximation error through refinement and essentially achieve smaller error than existing methods using the same number of samples.

There are several ideas to extend this work in the future. We have attempted to reduce the number of samples without consideration to the possible coherence between the trajectories of certain groups of nearby samples. It might be advantageous to explore such coherency to reduce the cost of the samples used or to improve the quality of the approximation. Another improvement avenue concerns making the detection of the edge features and the modification of their representations applied on the fly locally as new samples are added. We intend to explore other alternatives for the 1D fitting of the values as a function of the potentials at the data sites. Some fitting strategies lead to considerably better results than the ones discussed in this paper at the expense of a much higher computational cost. Also, we would like to extend the method to approximate a sequence of flow map computations corresponding to multiple integration durations. Finally, we want to investigate the use of the method described to provide an interactive visualization of the flow structures, where our reconstruction is applied in a view dependent manner.

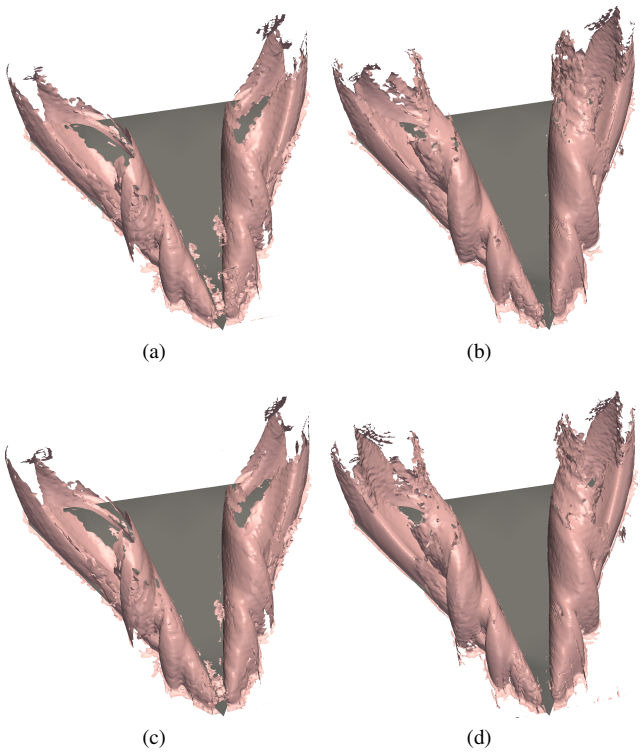


Fig. 13. FTLE ridges for the Delta Wing dataset extracted from the approximated flow map: (a) Adaptive four-point scheme using 940584 samples (4.2% of full resolution), (b) Our method using 940584 samples (4.2% of full resolution), (c) Adaptive four-point scheme using 2284277 samples (10.2% of full resolution), (d) Our method using 2284277 samples (10.2% of full resolution).

Acknowledgments

We thank Christoph Garth at the university of Kaiserslautern for providing the implementation of his method [8]. This paper is based in part upon work supported by the National Science Foundation under Grants No. OCI-1150000 and CMMI-1030326 and by a gift by Intel Corporation. We thank the providers of the libraries: Teem http://teem.sf.net, ITK http://www.itk.org, and CGAL http://www.cgal.org for making these tools available to researchers.

References

[1] A. Agranovsky, C. Garth, and K. Joy. Extracting flow structures using sparse particles. In *Vision, Modeling, and Visualization (2011)*, pages 153–160. The Eurographics Association, 2011.

[2] G. Allasia, R. Besenghi, and R. Cavoretto. Accurate approximation of unknown fault lines from scattered data, mem. *Accad. Sci. Torino Cl. Sci. Fis. Mat. Natur*, 33:3–26, 2009.

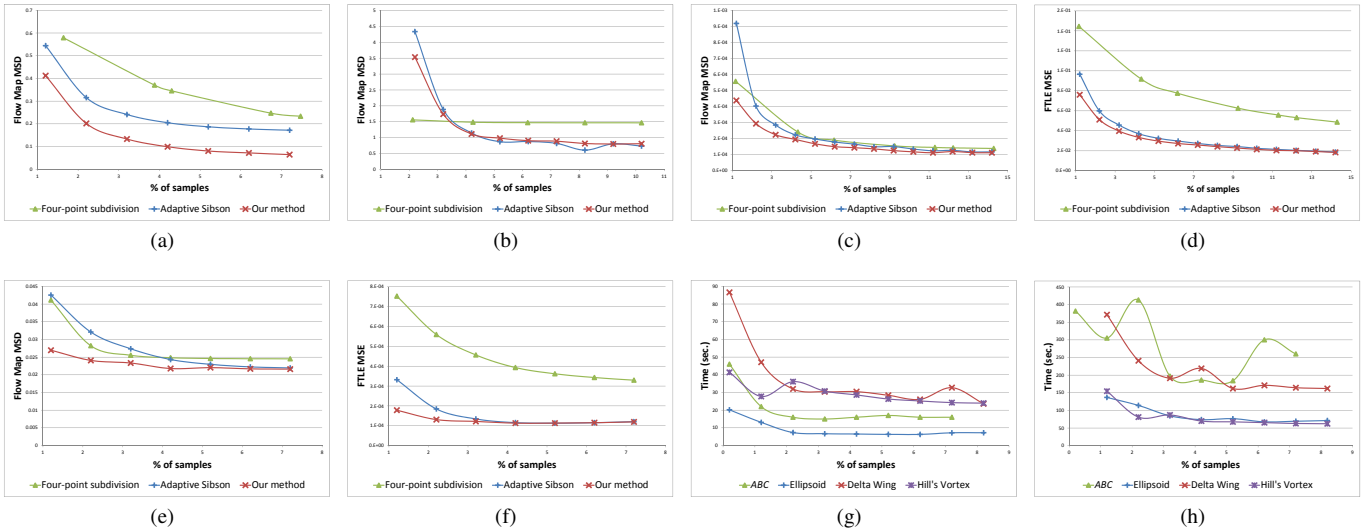
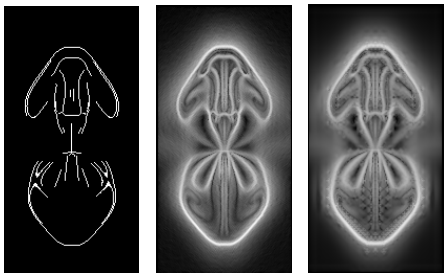


Fig. 10. The approximation errors for the ABC (a), Delta Wing (b), Ellipsoid (c)(d), and Hill's Vortex datasets (e) (f). In (g) and (h), we show the reconstruction times for both the adaptive Sibson's method and our method.

Table 1. Dataset Information and performance numbers. τ refers to the integration duration.

Dataset	Dimensions	Flow map computation time (hour)	τ	Time steps	Edge detection percentage parameter	Refinement Step (sec.)	λ	Adaptive four-point scheme (sec.)
ABC	256x256x256	4.83	5	analytical	0.06, 0.06, 0.06	4.17	4.0	38
Ellipsoid	256x128x224	2.32	1.0	steady	0.05, 0.035, 0.05	1.83	4.0	54
Delta Wing	288x324x240	14.26	1.5	20	0.04, 0.04, 0.04	6.13	4.0	120
Hill's Vortex	256x256x256	1.85	10	analytical	0.015, 0.015, 0.02	3.12	4.0	40
Double Gyre	1024x512	0.07	10	analytical	0.04, 0.04, 0.04	0.08	4.0	27



(a) (b) (c)

Fig. 14. A cross section of the FTLE for the Ellipsoid dataset computed using our approximation of the flow map (b) and using the adaptive four-point approximation (c). (a) The edge features used for our reconstruction.

[3] E. Arge and M. Floater. Approximating scattered data with discontinuities. *Numerical Algorithms*, 8(2):149–166, 1994.

[4] S. Barakat, C. Garth, and X. Tricoche. Interactive computation and rendering of finite-time Lyapunov exponent fields. *Visualization and Computer Graphics, IEEE Transactions on*, 18(8):1368–1380, 2012.

[5] S. Brunton and C. Rowley. Fast computation of finite-time Lyapunov exponent fields for unsteady flows. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(1):017503–017503, 2010.

[6] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[7] G. Chen, Q. Deng, A. Szymczak, R. S. Laramee, and E. Zhang. Morse set classification and hierarchical refinement using conley index. *IEEE Transactions on Visualization and Computer Graphics*, 18(5):767–782, 2012.

[8] 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(6):1464–1471, 2007.

[9] C. Garth, G. Li, X. Tricoche, C. Hansen, and H. Hagen. Visualization of coherent structures in transient 2d flows. *Topology-Based Methods in Visualization II*, pages 1–13, 2009.

[10] M. Green, C. Rowley, and G. Haller. Detection of Lagrangian coherent structures in three-dimensional turbulence. *Journal of Fluid Mechanics*, 572(1):111–120, 2007.

[11] L. Gross and G. Farin. A transfinite form of Sibson's interpolant. *Discrete Applied Mathematics*, 93(1):33–50, 1999.

[12] G. Guennebaud and M. Gross. Algebraic point set surfaces. In *ACM Transactions on Graphics (TOG)*, volume 26, page 23. ACM, 2007.

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

[14] G. Haller. A variational theory of hyperbolic Lagrangian coherent structures. *Physica D: Nonlinear Phenomena*, 240(7):574–598, 2011.

[15] M. Hlawatsch, F. Sadlo, and D. Weiskopf. Hierarchical line integration. *Visualization and Computer Graphics, IEEE Transactions on*, 17(8):1148–1163, 2011.

[16] J. Hultquist. Constructing stream surfaces in steady 3d vector fields. In *Proceedings of the 3rd conference on Visualization'92*, pages 171–178. IEEE Computer Society Press, 1992.

[17] J. Jung and V. Durante. An iterative adaptive multiquadric radial basis function method for the detection of local jump discontinuities. *Applied Numerical Mathematics*, 59(7):1449–1466, 2009.

[18] J. Kasten, C. Petz, I. Hotz, B. Noack, and H. Hege. Localized finite-time Lyapunov exponent for unsteady flow analysis. In *Vision Modeling and Visualization*, volume 1, pages 265–274. Citeseer, 2009.

[19] R. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. Post, and D. Weiskopf.

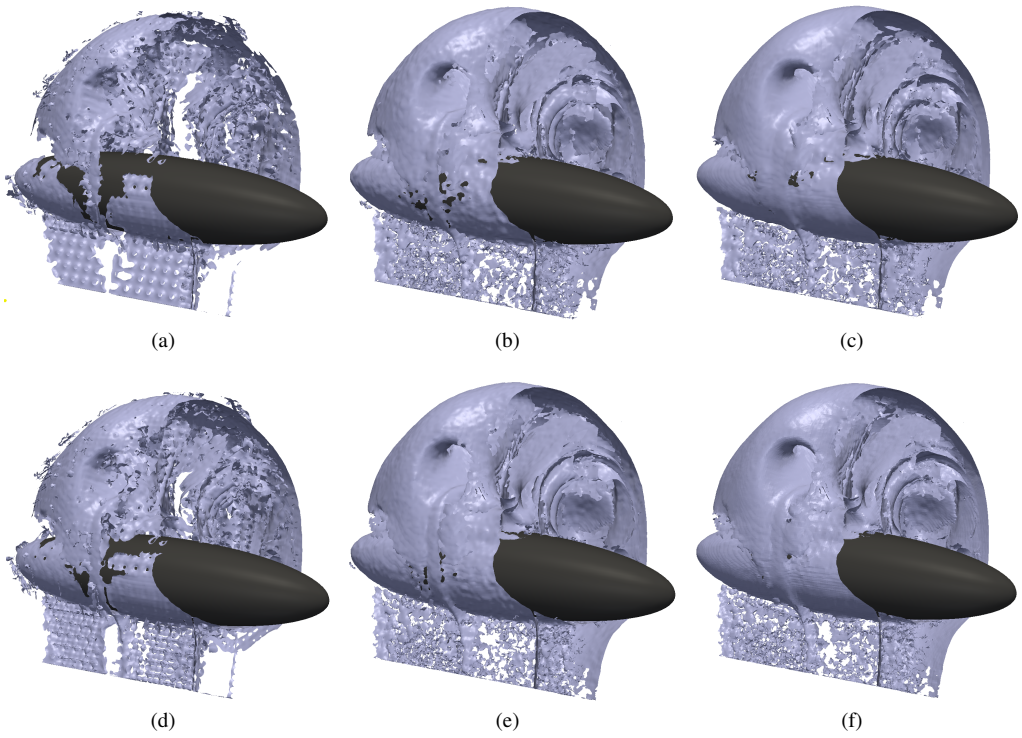


Fig. 12. FTLE ridges for the Ellipsoid dataset extracted from the approximated flow map: (a) Adaptive four-point scheme using 161480 samples (2.2% of full resolution), (b) Adaptive Sibson's method using 161480 samples (2.2% of full resolution), (c) Our method using 161480 samples (2.2% of full resolution), (d) Adaptive four-point scheme using 308281 samples (4.2% of full resolution), (e) Adaptive Sibson's method using 308281 samples (4.2% of full resolution), (f) Our method using 308281 samples (4.2% of full resolution).

The state of the art in flow visualization: Dense and texture-based techniques. In *Computer Graphics Forum*, volume 23, pages 203–221. Wiley Online Library, 2004.

[20] R. Laramee, H. Hauser, L. Zhao, and F. Post. Topology-based flow visualization, the state of the art. *Topology-based methods in visualization*, pages 1–19, 2007.

[21] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010.

[22] S. Niu, J. Yang, S. Wang, and G. Chen. Improvement and parallel implementation of canny edge detection algorithm based on GPU. In *IEEE 9th International Conference on ASIC (ASICON)*, pages 641–644, 2011.

[23] S. W. Park, L. Linsen, O. Kreylos, J. D. Owens, and B. Hamann. Discrete Sibson interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):243–253, 2006.

[24] R. Peikert and F. Sadlo. Flow topology beyond skeletons: Visualization of features in recirculating flow. In *Topology-Based Methods in Visualization II*, pages 145–160. Springer, 2009.

[25] A. Poje and G. Haller. Geometry of cross-stream mixing in a double-gyre ocean model. *Journal of physical oceanography*, 29(8):1649–1665, 1999.

[26] F. Post, B. Vrolijk, H. Hauser, R. Laramee, and H. Doleisch. Feature extraction and visualization of flow fields. *Eurographics 2002 State-of-the-Art Reports*, pages 69–100, 2002.

[27] F. Post, B. Vrolijk, H. Hauser, R. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. In *Computer Graphics Forum*, volume 22, pages 775–792. Wiley Online Library, 2003.

[28] V. Pratt. Direct least-squares fitting of algebraic surfaces. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 145–152. ACM, 1987.

[29] M. Rossini. 2d-discontinuity detection from scattered data. *Computing*, 61(3):215–234, 1998.

[30] F. Sadlo and R. Peikert. Efficient visualization of Lagrangian coherent structures by filtered amr ridge extraction. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1456–1463, 2007.

[31] F. Sadlo and R. Peikert. Visualizing Lagrangian coherent structures and comparison to vector field topology. *Topology-Based Methods in Visualization II*, pages 15–29, 2009.

[32] F. Sadlo, A. Rigazzi, and R. Peikert. Time-dependent visualization of Lagrangian coherent structures by grid advection. *Topological Methods in Data Analysis and Visualization*, pages 151–165, 2011.

[33] 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. ACM, 2007.

[34] D. Schneider, A. Wiebel, and G. Scheuermann. Smooth stream surfaces of fourth order precision. In *Proceedings of the 11th Eurographics / IEEE - VGTC conference on Visualization*, EuroVis'09, pages 871–878, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[35] S. Shadden, F. Lekien, and J. Marsden. Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212(3):271–304, 2005.

[36] R. Sibson. A brief description of natural neighbour interpolation. *Interpreting multivariate data*, volume 21, 1981.

[37] S. Sinha and B. Schunck. A two-stage algorithm for discontinuity-preserving surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):36–55, 1992.

[38] H. Theisel and H. Seidel. Feature flow fields. In *Proceedings of the symposium on Data visualisation 2003*, pages 141–148. Eurographics Association, 2003.

[39] X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen. Topology tracking for the visualization of time-dependent two-dimensional flows. *Computers & Graphics*, 26(2):249–257, 2002.