Table of Contents

Combinatorial 2D Vector Field Topology Extraction and Simplification

Jan Reininghaus[1], Ingrid Hotz[2]

[1] Zuse Institute Berlin (ZIB), Germany. reininghaus@zib.de
[2] Zuse Institute Berlin (ZIB), Germany. hotz@zib.de

Summary: This paper investigates a combinatorial approach to vector field topology. The theoretical basis is given by Robin Forman's work on a combinatorial Morse theory for dynamical systems defined on general simplicial complexes. We formulate Forman's theory in a graph theoretic setting and provide a simple algorithm for the construction and topological simplification of combinatorial vector fields on 2D manifolds. Given a combinatorial vector field we are able to extract its topological skeleton including all periodic orbits. Due to the solid theoretical foundation we know that the resulting structure is always topologically consistent. We explore the applicability and limitations of this combinatorial approach with several examples and determine its robustness with respect to noise.

1 Introduction

Topological methods have developed into an integral part in data analysis and visualization, for scalar as well as for vector valued data. While the results of scalar and vector field topology coincide when dealing with gradient vector fields, the respective mathematical approach originated in different fields.

Vector field topology is usually motivated and defined as a clustering of streamlines of the flow generated by the vector field. This clustering has a theoretical sound foundation in dynamical systems theory. One first defines limit sets of the streamlines. Features of the vector field are then defined as intersections of these limit sets. The extracted structure of the vector field is therefore connected to the equivalence classes generated by the flow and is called topology of the vector field. The extraction of the topological skeleton builds on a continuously defined vector field using numerical methods.

Scalar field topology on the other hand is usually defined through Morse theory. Given a non degenerate function $f : M \to \mathbb{R}$ this theory defines a structure M^*, called Morse complex. This structure consists of minima, maxima, saddle points and their connectivity. A main result of Morse theory is that M^* is isomorphic to the singular homology of the domain of the function M. As the singular homology is a rather fine grained topological invariant of the domain M, this structure is also called the topology of the scalar function f. In general the topology is extracted using a combinatorial approach building on the data given in discrete points.

In this paper we approach vector field topology from a Morse theoretic viewpoint. Forman describes a discrete Morse theory for general vector fields in [4], our theoretical foundation. Lewiner has successfully applied Forman's discrete Morse theory for gradient vector fields [5] in [11] which motivated our investigation.

Forman's definitions lead to a structure M^* that contains all critical points, their connectivity and all closed streamlines. If M^* contains no closed streamlines, then this structure is isomorphic to the singular homology of the domain M. Morse theoretic vector field topology therefore degenerates to scalar field topology when dealing with gradient vector fields. Should M^* contain closed streamlines, then there are still many topological invariants of M that are contained in M^*, e.g. there are the strong Morse inequalities which guarantee that the topological complexity of M is bounded by the complexity of M^*.

The proof of these inequalities provides a certain inherent consistency to this vector field topology. For example, a vector field on a sphere always contains at least one critical point, a vector field on a torus with no critical points contains at least two periodic orbits. These kinds of topological constraints are always reflected in M^*. Note that this consistency is preserved in practice, as we can compute M^* exactly because of the combinatorial nature of the definitions.

2 Related Work

The use of topological methods for scalar and vector field visualization developed almost independently over the last two decades. Both areas build on solid mathematical foundations. While methods for vector fields mostly refer back to Poincaré Index Theory, topological scalar field analysis is based on Morse Theory. For a basic overview over these theories and their relation we refer to [13].

Vector field topology was introduced to visualization by Helman and Hesselink [4]. They defined the concept of a topological skeleton consisting of critical points and connecting separatrices to segment the field into regions of topologically equivalent streamline behavior. An algorithm to extract periodic orbits, completing the topological structure, was first proposed by Wischgoll et al. [19] based on the analysis of cell cycles. Later Theisel et al. [15] suggested an algorithm to detect isolated closed streamlines in 2D vector fields by intersecting certain stream surfaces in 3D. Following this work the use of topological methods has been further advanced according to many aspects like topology tracking, extraction of boundary topology and extensions to 3D.

In the following we only point out some specific publications dealing with topological simplification, which is also a central question in our work. For a more complete overview we refer to the survey paper [6] and the references therein.

A major obstacle in the application of vector field topology is the high feature density in complex data sets. Therefore much attention has been payed to scaling and simplification of the topological skeleton. Solutions have been proposed by Tricoche et al. following two different strategies. A scaling approach determines groups of close singularities according to a given measure. These are replaced consistently by fewer structures consisting of higher order singularities [17]. The main target of this method is the removal of visual clutter - the structural influence of singularities is ignored allowing for possibly important flow features to be removed. The second approach focuses on pair annihilations leading to a progressive simplification of the topological graph. The pairs are sorted according to a relevance measure, e.g. the Euclidean distance of

the critical points [16]. In both cases the vector field is changed to fit the simplified structure.

Similar to the scaling approach Weinkauf et al. [18] cluster close critical points in 3D fields and replace them by one critical point of higher order represented by a specific icon. Klein et al. [9] apply a scale space technique to track critical points over multiple spatial scales in order to assess the importance of a critical point to the overall behavior of the underlying flow field using Gaussian filter kernels. All this work is based on numerical analysis of continuous vector fields resulting from interpolated data values given at single vertices.

Scalar field topology developed mostly independently from vector field topology. The main application areas in visualization include segmentation, transfer function design, and ridge extraction. A variety of algorithms for its extraction have been introduced, some using the corresponding gradient vector field, others suggesting combinatorial approaches using Morse-Smale theory. The Morse-Smale complex is a topological structure partitioning the domain into regions of uniform gradient flow. We would like to point out a few publications here, which motivated our work. Edelsbrunner et al. [3] define Morse-Smale complexes for piecewise linear data resulting in a combinatorial algorithm for its extraction. To reduce the often very complex structure a controlled simplification is suggested based on the general concept of persistence introduced in [2]. These ideas have been applied for visualization of 3D data in [10].

Forman has chosen a slightly different approach by developing a discrete Morse theory [6] for scalar fields defined on cell complexes. Rather than choosing a suitable class of continuous functions on the space, he assigns a single number to each cell of the complex and all further steps are combinatorial. Lewiner et al. [12] has applied this theory successfully to scalar fields on triangulated manifolds.

Motivated by the success of these combinatorial algorithms for scalar field topology, recently first steps have been taken towards a combinatorial vector field topology. Chen [1] et al. have developed a combinatorial topology extraction method based on Conley Index Theory. Contrary to our approach they however make use of continuous numerical methods for the subsequent simplification.

3 Basic Ideas and Definitions

In this section we present Forman's combinatorial Morse theory for vector fields [4] in graph theoretic terms. For simplicity we restrict ourselves to triangulated 2D manifolds while Forman's theory is defined in a far more general setting. We refer to graph edges as links to avoid confusion with the edges of the triangulation.

Given a triangulation of a manifold M we first define its simplicial graph (see Figure 1a). The nodes of the graph consist of the vertices, edges and triangles of the triangulation and each node α^p is labeled with the dimension p of the geometric simplex it represents. The links of the graph encode the neighborhood relation of the triangulation: If the simplex represented by the node α^p is in the boundary of the simplex represented by

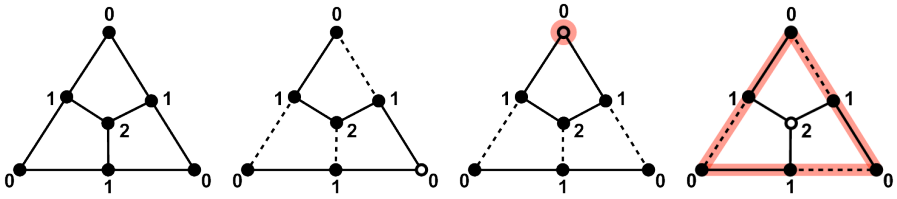


Fig. 1. Basic definitions, from left to right: a) simplicial graph of a single triangle; b) combinatorial vector field (dashed); c) critical point of index 2 (red); d) attracting periodic orbit (red)

the node β^{p+1} then $\{\alpha^p, \beta^{p+1}\}$ is a link in the graph. Note that only simplices whose dimension differs by one are linkable.

A matching of a graph is defined as a subset of links such that no two links are adjacent. We are now ready for the central definition of this paper: A *combinatorial vector field V* is a matching of a simplicial graph (see Figure 1b).

The nodes of the graph that are not covered by V are called critical points (see Figure 1c). If α^p is a critical point of V we say that α^p has index p. A critical point of index p is called sink $(p=0)$, saddle $(p=1)$ or source $(p=2)$. Because of the matching property non critical nodes always appear in pairs whose associated dimension p differs by exactly one. Therefore one can immediately conclude the combinatorial Poincaré-Hopf formula (see [4])

$$\begin{aligned}\chi(M) &= \sum_{p=0}^{2} (-1)^p \{\text{number of } p\text{-simplices}\} \\ &= \sum_{p=0}^{2} (-1)^p \{\text{number of critical points with index } p\},\end{aligned} \tag{1}$$

which already hints at the inherent topological consistency provided by Forman's combinatorial approach.

A combinatorial p-streamline is a path in the graph whose links alternate between V and the complement of V and the dimension of the nodes of the path alternates between p and $p+1$. A p-streamline connecting two critical points is called a separatrix. If a p-streamline is closed we call it either an attracting periodic orbit $(p=0)$ (see Figure 1d) or a repelling periodic orbit $(p=1)$. When a combinatorial vector field contains no periodic orbits we call it a discrete gradient vector field (see Lewiner).

As the relation of the critical points to the topology of the manifold is quite hard to state in general combinatorial vector fields, we restrict ourselves to combinatorial Morse-Smale vector fields, i.e. combinatorial vector fields whose periodic orbits are pairwise disjoint. Note that combinatorial vector fields of simplicial graphs of 2D manifolds are always of type Morse-Smale due to the simple graph structure: Each 1-simplex has exactly two 0-simplices in its boundary and is itself in the boundary of at most two 2-simplices. Therefore p-streamlines can merge but not split, which implies that periodic orbits are pairwise disjoint.

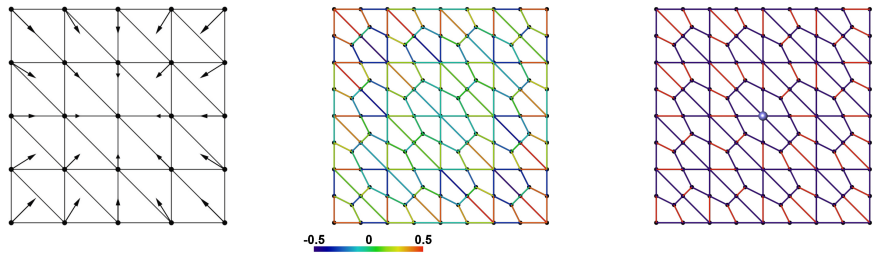


Fig. 2. Algorithmic pipeline, from left to right: a) input triangulation with vectors on vertices; b) link weighted simplicial graph; c) computed combinatorial vector field (red) with a sink in the center (blue). Color plate C. 1, page 253.

Let c_p denote the number of critical points with index p, A_p the number of closed p-streamlines and b_p the p-th Betti number of M. Forman has proven in [4] that for any k there holds the strong Morse inequality

$$A_k + c_k - c_{k-1} + \ldots \pm c_0 \geq b_k - b_{k-1} + \ldots \pm b_0. \tag{2}$$

This result shows that the critical points and periodic orbits carry the topology of the triangulated manifold. As the topological features (critical points, separatrices and periodic orbits) can be computed exactly in a combinatorial vector field due to the finite nature of the definitions we always get a skeleton which is consistent with the topology of the underlying manifold.

Note that the separatrices do not take part in (2). However, there is an intricate topological invariant, called Reidemeister torsion, whose computation involves the separatrices (see [4]). Therefore we consider the separatrices to be part of the topological skeleton of a vector field.

4 Algorithm

In this section we present the algorithm we developed to construct a combinatorial vector field and extract its topological skeleton. Furthermore we show how the same algorithm that is used for the construction of the field can be employed to consistently simplify its topological skeleton. The input of the algorithmic pipeline is assumed to consist of a triangulation with vectors given on the vertices (see Figure 2).

We begin by constructing the simplicial graph $G = (S, L)$ (see Section 3) of the given triangulation. We add all simplices (vertices, edges and triangles) to the graph and then encode the connectivity information by adding the corresponding links to the graph (see Figure 1a).

To represent the given vector field data we compute link weights $w : L \to \mathbb{R}$ in the simplicial graph. Let $c : S \to \mathbb{R}^3$ denote the coordinates of the midpoints of the simplices represented in the graph. To calculate the auxiliary data values $f : S \to \mathbb{R}^3$ on the nodes of the graph we average the data values given on the vertices of the triangulation. The weight of a link $e = \{n_1^{p+1}, n_2^p\} \in L$ is then given by

$$w(e) = \frac{1}{2}\left(f(n_1^{p+1}) + f\left(n_2^p\right)\right) \cdot \left(c(n_1^{p+1}) - c(n_2^p)\right), \tag{3}$$

where the dot denotes the scalar product. This term corresponds to the tangential component of the given data value scaled with the geometric length of the link. Note that the link weighted graph contains enough information to reconstruct the tangential vector field induced by the given data.

Let $\mathscr{M}$ denote the set of all matchings of the simplicial graph G and let $w(M) = \sum_{e \in M} w(e)$ denote the weight of the matching M. Then a combinatorial vector field V representing the given data can be constructed by computing

$$V = \arg\max_{M \in \mathscr{M}} w(M), \tag{4}$$

i.e. by finding the heaviest matching of the link weighted simplicial graph.

Note that the graph $G = (S, L)$ is bipartite, as the links only connect odd with even dimensional simplices. This greatly simplifies the algorithm needed to solve (4). The graph problem (4) has been thoroughly studied, usually called maximum weighted bipartite matching. For this work we chose an exact approach based on the Hungarian method with a computational complexity of $O(|S|^2 \log(|S|))$. This approach iteratively calculates the sequence of matchings

$$M_k := \arg\max_{M \in \mathscr{M}, |M| = k} w(M). \tag{5}$$

As this sequence does not contain several isolated maxima we can stop the computation once $w(M_{k+1}) < w(M_k)$ and return the combinatorial vector field $V := M_k$ satisfying (4). For more details, a correctness proof and the computational complexity analysis we refer to [14].

Let $s(M_k) \subset S$ denote the critical points of the combinatorial vector field M_k. Then $s(M_{k+1}) \subset s(M_k)$ and $|s(M_k)| - |s(M_{k+1})| = 2$. The vector fields $M_{k+\ell}$ can therefore be interpreted as topologically simplified versions of the vector field $V = M_k$.

Given a combinatorial vector field we can easily extract its topological skeleton, i.e. its critical points, separatrices and periodic orbits. To find the critical points of the vector field we only have to find the nodes of the graph that are not covered by the matching. For the classification into sinks, sources and saddles we just have to look at the label p. Overall all critical points can be found and classified in $O(|S|)$.

To extract all periodic orbits we make use of a depth first search through the graph with the side constraints of a p-streamline. Because p-streamlines of simplicial graphs of 2D manifolds can merge but not split (see section 3) we can extract all periodic orbits in $O(|S|)$. The classification into attracting and repelling orbits is again given by the label p of the participating nodes.

The separatrices can be found analogously with the constrained depth first search mentioned above with a computational complexity $O(|S|)$ for the extraction of all separatrices.

5 Results

The goal of this section is to explore the properties of the combinatorial vector fields constructed in Section 3. The robustness of the algorithm when dealing with noisy data is analyzed in the first example. The second example verifies the topological consistency guaranteed in Section 3. Finally we present a real world data set to examine the potential of this method for multi-scale topological vector field analysis.

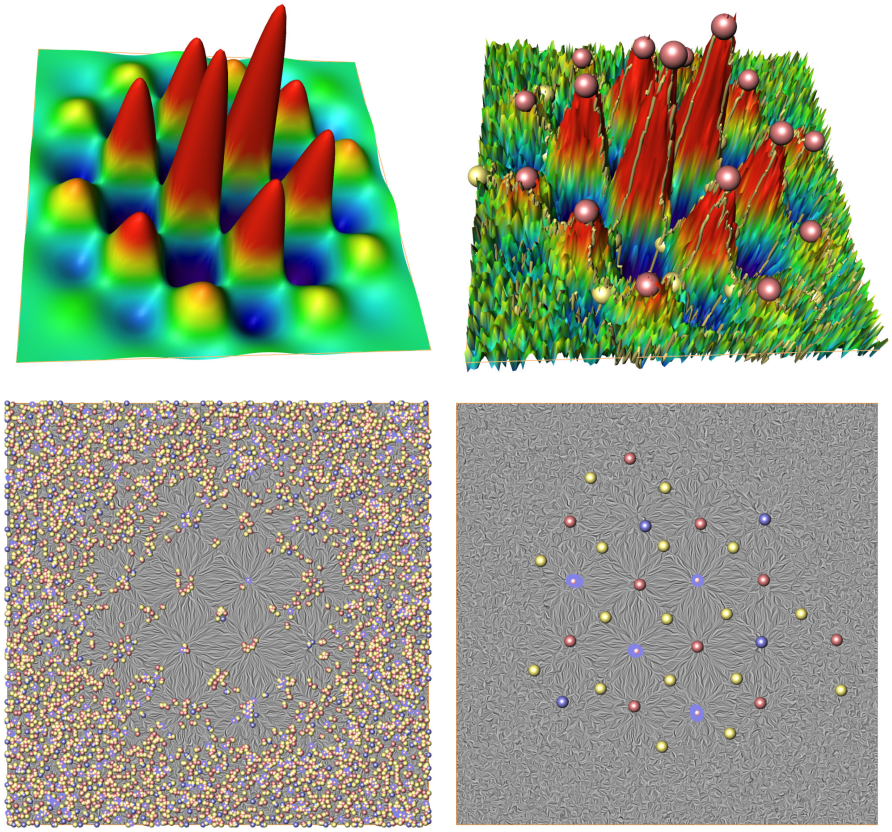


Fig. 3. Top-left: height field of unperturbed data set; top-right: height field of perturbed data set with simplified topological skeleton; bottom-left: sources (red), sinks (blue), and saddles (yellow) of initial combinatorial vector field on planar LIC of the gradient vector field; bottom-right: critical points of simplified topological skeleton - the blue circles represent tiny attracting periodic orbits (color plate C. 3, page 254).

Example 1 The purpose of this data set is to show the connection of our simplification algorithm with the concept of persistency from scalar field topology. To do this, we

analyzed the gradient field of the scalar function

$$f(x,y) = \sin(10x)\sin(10y)\exp^{-3(x^2+y^2)}$$

on the domain $[-1,1]^2$ (see Figure 3 top-left) discretized with a uniform triangulation consisting of $16,000$ vertices. To determine the robustness of our algorithm with respect to noise we added random numbers in the range of $[-0.1,0.1]$ to the scalar values of the mesh before taking the gradient (see Figure 3 top-right). This results in a perturbed gradient vector field containing many critical points (see Figure 3 bottom-left). Our topological simplification behaves like a persistence based approach: the extrema of the gradient field are generally canceled in the order given by the absolute value of the difference of the corresponding scalar values (see Figure 3 bottom-right). The data set is designed such that this order is reflected in the distance to the origin.

Our simplification algorithm recovers the topology of the original unperturbed field. The presence of noise is indicated by the graph shown in Figure 4. The x-axis represents the simplification threshold, i.e. we stop the simplification process when $w(M_{k+\ell}) - w(M_{k+\ell+1})$ is larger than the given threshold. The y-axis represents the number of the critical points of the corresponding combinatorial vector field. The sharp drop in the red graph at a simplification threshold of 0.1 indicates the presence of noise in the range of $[-0.1,0.1]$. The symmetry of the unperturbed data f can be seen in the blue graph as there are eight cancellations with the same simplification threshold of about 0.47.

Note that the combinatorial vector fields computed in this example are not always discrete gradient vector fields, i.e. they may contain periodic orbits. The gradient field property is not representable by the point set of vectors that is given for the construction of the vector field. However, most of the time when we sample a gradient vector field on the nodes of a triangulation and compute the combinatorial vector field, we get a discrete gradient vector field containing no periodic orbits.

Example 2 This data set consists of a random vector field defined on a torus. We created a random vector field on a $5 \times 5 \times 5$ grid and then sampled the trilinear interpolant on the triangulation of a torus with $16,000$ vertices (see Figure 5). As we know the Betti numbers of a torus ($b_0 = 1, b_1 = 2, b_2 = 1$) we can verify that the initial and all simplified topological skeletons satisfy the strong Morse inequalities (2) as expected (see Table 1).

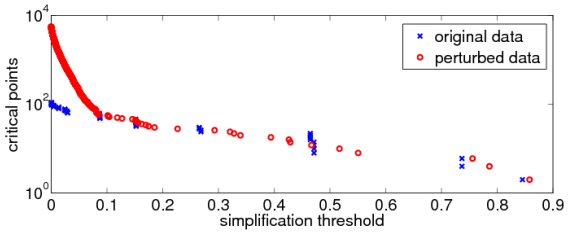


Fig. 4. Simplification graph of example 1. The sharp drop of the red graph at 0.1 indicates the presence of noise in the range of $[-0.1,0.1]$

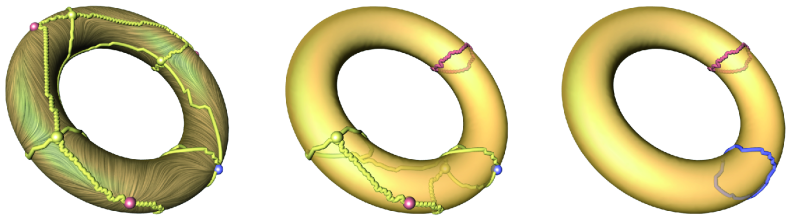


Fig. 5. Random vector field on a torus with three levels of topological simplification (color plate C. 3, page 254).

Example 3 This real world example depicts the surface velocity field of a simulation of blood flow through a cerebral aneurysm done by the Biofluid Mechanics Lab of the Charité - Universitätsmedizin Berlin. The simplicial graph of the triangulation consists of 60,000 nodes. The computation time for the construction, simplification and topology extraction is 29 minutes on a 3 GHz CPU with our current implementation. Almost all time is spent in the maximum weighted bipartite matching code that solves (4).

The critical points in this vector field are stagnation points and thus of interest for the flow analysis. While standard vector field topology is able to reliably extract these critical points too, our algorithm delivers a hierarchy of topological skeletons which captures the dominant nature of the flow (see Figure 6 bottom-left). The blood enters the aneurysm at the bottom, and leaves it horizontally. This behavior is found by our algorithm and the global separation on the surface is extracted. This reduced flow structure may serve as a basis when comparing different cerebral aneurysms.

6 Discussion and Future Work

This first approach to combinatorial vector field topology based on Forman's discrete Morse theory looks promising. The extracted topological skeleton is always consistent, the multi scale applicability has been demonstrated, and our algorithm is able to reliably

ℓ	sinks	saddles	sources	repelling orbits	attracting orbits
0	2	7	5	0	0
1	2	6	4	0	0
2	2	5	3	0	0
3	1	4	3	0	0
4	1	3	2	0	0
5	1	2	1	0	1
6	0	1	1	1	1
7	0	0	0	1	1

Table 1. Topological signature of a sequence of combinatorial vector fields $M_{k+\ell}$ on the torus, see Example 2. The strong Morse inequalities (2) hold in each case.

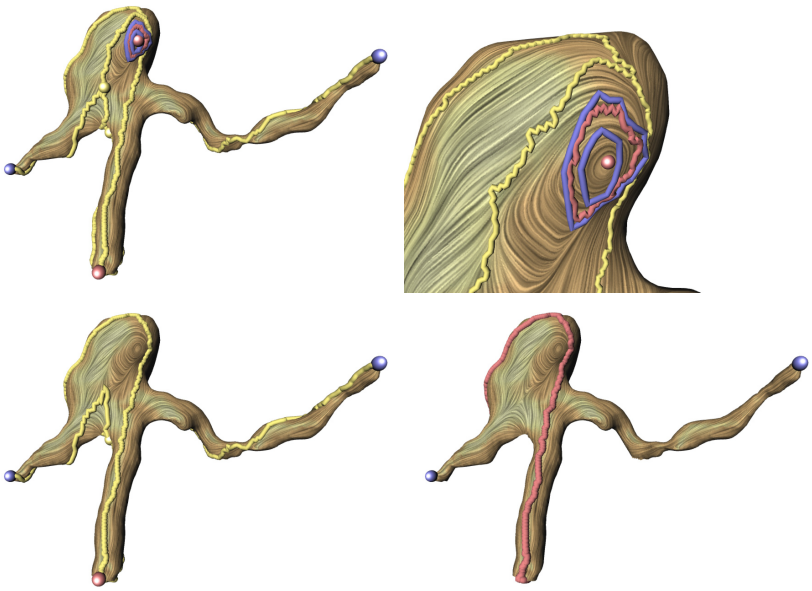


Fig. 6. Top-left: topological skeleton of a vector field on a cerebral aneurysm given by a blood flow simulation; top-right: zoom in of an area exhibiting recurrent flow behavior indicated by attracting periodic orbits (red) and repelling periodic orbits (blue); bottom-left: simplified topological skeleton with 1 saddle, 2 sinks and 1 source; bottom-right: simplified topological skeleton with 2 sinks and 1 repelling orbit (color plate C. 4, page 255).

deal with noisy data (see Section 5). We have identified three areas where the algorithm could be further improved:

The quadratic runtime stemming from the maximum weighted matching problem (4) reduces the applicability of our algorithm to real world data sets. A greedy approximation algorithm for (4) should alleviate this problem. Note that the consistency of the topological skeleton would not be affected by such an approximation as it is induced by the matching property.

While continuous vector field topology is able to compute the position of the separatrices accurately, the separatrices computed by our combinatorial approach do not coincide with the exact separatrices of a given continuous vector field - regardless of the resolution of the mesh employed used to sample the continuous vector field (see Figure 6 top-right). The same problem is exhibited by periodic orbits. This behavior might be corrected by a more sophisticated definition of link weights of the graph (3).

Consider a vector field on a torus with two attracting and two repelling orbits. In principle it is possible to simplify this topology. However, our algorithm is unable to do so. The simplification is based on matchings and always cancels pairs of critical points. If there are no critical points in the vector field, then the algorithm cannot simplify its topology.

In conclusion, the presented algorithm delivers a completely combinatorial approach to vector field topology on triangulated manifolds that is always consistent. The formulation of the algorithm in standard graph terminology allows for a straight forward implementation using a graph library and the only parameter present is the level of topological detail we want to extract.

Acknowledgements

We would like to thank David Günther and Tino Weinkauf for many fruitful discussions on this topic and Christian Löwen for his great implementational efforts. This work was funded by the DFG Emmy-Noether research programm. All visualizations in this paper have been created using `AMIRA` - a system for advanced visual data analysis (see http://amira.zib.de/).

References

1. G. Chen, K. Mischaikow, R.S. Laramee, P. Pilarczyk, and E. Zhang. Vector field editing and periodic orbit extraction using morse decomposition. *IEEE Transactions in Visualization and Computer Graphics*, 13:769–785, 2007.
2. Herbert Edelsbrunner and J. Harer. Persistent homology — a survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry: Twenty Years Later*, volume 458, pages 257–282. AMS Bookstore, 2008.
3. Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Morse-smale complexes for piecewise linear 3-manifolds. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 361–370, New York, NY, USA, 2003. ACM.
4. Robin Forman. Combinatorial vector fields and dynamical systems. *Mathematische Zeitschrift*, 228:629–681, 1998.
5. Robin Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998.
6. Robin Forman. A user's guide to discrete morse theory. In *Proceedings of the 2001 Internat. Conf. on Formal Power Series and Algebraic Combinatorics*, Advances in Applied Mathematics, 2001.
7. Attila Gyulassy, Vijay Natarajan, Valerio Pascucci, Peer-Timo Bremer, and Bernd Hamann. A topological approach to simplification of three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):474–484, 2006.
8. J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *Computer*, 22(8):27–36, August 1989.
9. Thomas Klein and Thomas Ertl. Scale-space tracking of critical points in 3d vector fields. In Hans Hagen Helwig Hauser and Holger Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 35–49. Springer Berlin Heidelberg, May 2007.
10. Robert S. Laramee, Helwig Hauser, Lingxiao Zhao, and Frits H. Post. Topology-based flow visualization, the state of the art. In Hans Hagen Helwig Hauser and Holger Theisel, editors, *Topology-based Methods in Visualization*, Mathematics and Visualization, pages 1–19. Springer Berlin Heidelberg, May 2007.
11. Thomas Lewiner. *Geometric discrete Morse complexes*. PhD thesis, Department of Mathematics, PUC-Rio, 2005. Advised by Hlio Lopes and Geovan Tavares.

12. Thomas Lewiner, Helio Lopes, and Geovan Tavares. Applications of forman's discrete morse theory to topology visualization and mesh compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):499–508, 2004.
13. J.J. Sanchez-Gabites. Dynamical systems and shapes. *RACSAM: Geometry and Topology*, 102:127–159, 2008.
14. Alexander Schrijver. *Combinatorial Optimization*. Springer, 2003.
15. Holger Theisel, Tino Weinkauf, Hans-Christian Hege, and Hans-Peter Seidel. Grid-independent detection of closed stream lines in 2d vector fields. In *Proceedings of the VMV Conference 2004*, page 665, Stanford, USA, November 2004.
16. Xavier Tricoche, Gerik Scheuermann, and Hans Hagen. Continuous topology simplification of planar vector fields. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 159–166, Washington, DC, USA, 2001. IEEE Computer Society.
17. Xavier Tricoche, Gerik Scheuermann, Hans Hagen, and Stefan Clauss. Vector and tensor field topology simplification on irregular grids. In D. Ebert, J. M. Favre, and R. Peikert, editors, *VisSym '01: Proceedings of the symposium on Data Visualization 2001*, pages 107–116, Wien, Austria, May 28–30 2001. Springer-Verlag.
18. Tino Weinkauf, Holger Theisel, K. Shi, Hans-Christian Hege, and Hans-Peter Seidel. Extracting higher order critical points and topological simplification of 3D vector fields. In *Proc. IEEE Visualization 2005*, pages 559–566, Minneapolis, U.S.A., October 2005.
19. Thomas Wischgoll and Gerik Scheuermann. Detection and visualization of closed streamlines in planar flows. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):165–172, 2001.

Simplification of Jacobi Sets

Suthambhara N[1] and Vijay Natarajan[1,2]

[1] Department of Computer Science and Automation
[2] Supercomputer Education and Research Centre
Indian Institute of Science, Bangalore, India
email: {suthambhara, vijayn}@csa.iisc.ernet.in

Abstract. The Jacobi set of two Morse functions defined on a 2-manifold is the collection of points where the gradients of the functions align with each other or where one of the gradients vanish. It describes the relationship between functions defined on the same domain, and hence plays an important role in multi-field visualization. The Jacobi set of two piecewise linear functions may contain several components indicative of noisy or a feature-rich dataset. We pose the problem of simplification as the extraction of level sets and offset contours and describe an algorithm to compute and simplify Jacobi sets in a robust manner.

1 Introduction

Motivation. The Jacobi set extends the notion of critical points to multiple functions and helps describe the relationship between multiple scalar functions. Edelsbrunner et al. [5] have shown that the Jacobi sets can be used to compute a comparison measure between two scalar functions. Bennett et al. [1] have used the Jacobi set to represent tunnels and the silhouette of a mesh, both of which are subsequently used to compute a cross parameterization. Jacobi sets have also been used to track features of time-varying events such as molecular interactions, combustion simulation, etc. [2]. All the above applications face a common challenge, namely the presence of degenerate regions and noise in the data. The number of components of the Jacobi set is often more than what can be visually comprehended. So, it is necessary to simplify the Jacobi set. The simplification can be accomplished either using the notion of persistence [6], or otherwise.

Prior work and our approach. In their paper, Bremer et al. [2] have described a method to remove noise in the Jacobi set for time varying data. The persistence of a component of the Jacobi set is the time interval between its birth and death. This measure has been used to remove components that are either noise in the data or unimportant features. Extending this for general functions is nontrivial and hence a more complete approach with guaranteed error bounds is required. We pose the problem of computing Jacobi sets as the computation of a level set of a function defined on the input manifold. Jacobi set simplification now simplifies the level set. We also ensure that the change in relationship between the functions due to simplification does not exceed a given input threshold.

2 Background

Morse Theory. Morse theory studies the relationship between functions and domains. Let $\mathbb{M}$ be a smooth Riemannian 2-manifold. Let f be a smooth function defined on $\mathbb{M}$ and (x_1, x_2) be a local coordinate system such that the unit tangent vectors $(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2})$ form an orthonormal basis with respect to a Riemannian metric. The gradient of f at x is defined as the vector $\nabla f(x) = (\frac{\partial f}{\partial x_1}(x), \frac{\partial f}{\partial x_2}(x))$

A point x is a critical point of f if $\nabla f(x)$ is the zero vector. The function f is called a *Morse function* if the Hessian

$$\mathscr{H}_f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) \end{bmatrix}$$

is non-singular at all critical points.

Level sets and Reeb graphs. The *level set* at c is defined as the set of all points where f attains the value $c : f^{-1}(c) = \{x \in \mathbb{M} \mid f(x) = c\}$. The *Reeb graph* of f is obtained by contracting connected level set components to points. Nodes in a Reeb graph correspond to critical points of f, see Fig.1. The level sets *sweep* the domain as we increase c over the range of the function f. During a sweep over the domain, the topology of the level set changes at critical points of f. If the sweep is in the direction of increasing function value, level set components are created at minima, they merge or split at saddles, and are destroyed at maxima. Given a sweep direction, saddles may be classified as split or merge saddles depending on the change in the topology of level sets at these points.

Jacobi Sets. The *Jacobi set* of two Morse functions f and g defined on a 2- manifold $\mathbb{M}$ is the collection of points or where the gradients of the functions align with each other or one of the gradients vanish. Alternately, the Jacobi set can be described as the collection of critical points of the family of functions $f + \lambda g, \lambda \in \mathbb{R}$:

$$\mathbb{J} = \{x \in \mathbb{M} \mid \text{x is a critical point of } f + \lambda g \text{ or of } \lambda f + g\}$$

Note that the Jacobi set contains critical points of f and g. Edelsbrunner and Harer [4] used this alternate description to compute Jacobi sets of piecewise linear functions. They also showed that the Jacobi set of two Morse functions is a smoothly embedded 1-manifold in $\mathbb{M}$.

3 Simplification

We prefer to use the description of the Jacobi set as the level set of a gradient-based comparison measure [5] because it leads us to a natural algorithm for computing Jacobi sets. Let $\mathbb{M}$ be a 2-manifold smoothly embedded in $\mathbb{R}^3$. The *comparison measure* at a point $x \in \mathbb{M}$ for two Morse functions f and g is defined as $\kappa_x = \|\nabla f(x) \times \nabla g(x)\|$. Assuming $\mathbb{M}$ is orientable, we define the *sign extended comparison measure*, κ_x^S, at the point x with unit normal $\hat{n}$ as $\kappa_x^S(f,g) = (\nabla f \times \nabla g) \cdot \hat{n}$. The sign extended comparison measure is a function defined on the manifold $\mathbb{M}$ and the Jacobi set can be described

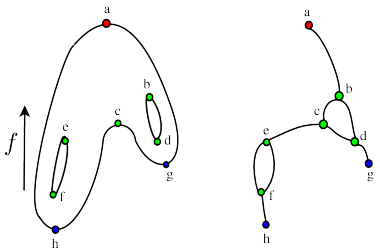


Fig. 1. Left: A two-holed 2-manifold and the height function defined on it. Points in blue, green, and red correspond to minima, saddle, and maxima of the function, respectively. **Right:** The Reeb graph of the height function. Loops in the Reeb graph correspond to holes in the manifold.

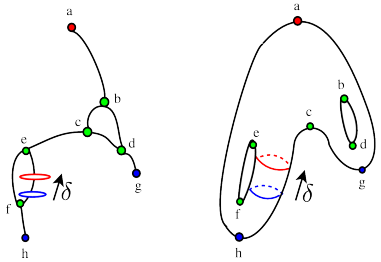


Fig. 2. Offsetting a level set component. **Left:** Level set components on the manifold. **Right:** Offsetting a level set component (blue) to another component (red) along an edge of the Reeb graph (color plate C. 6, page 255).

as the set of points where κ_x^S equals zero, i.e. the zero level set of κ_x^S, $\mathbb{J} = \kappa_x^{-1}(0) = \kappa_x^{S^{-1}}(0)$.

The Jacobi set contains spurious loops because of noise and degeneracies in the data. Simplification of the Jacobi set refers to the reduction in number of components of $\mathbb{J}$ with minimal change to the relationship between the two input functions.

The relationship between the functions is quantified by the *global comparison measure* κ, which is equal to the comparison measure integrated over the manifold and normalized by the total area [5].

$$\kappa = \frac{1}{Area(\mathbb{M})} \int_{x \in \mathbb{M}} \kappa_x dA_x,$$

where dA_x is the area element at x.

Offsetting components. The Jacobi set components are altered by computing offset level set components. Let p and p' be two level set components such that their corresponding points on the Reeb graph are connected by a monotone path. The level set component p is said to be offset to p' if it is replaced by the component p'. The cost of an offset operation is given by the hypervolume, which is computed as an integral over the swept region R of the domain:

$$H = \frac{1}{Area(\mathbb{M})} \int_{x \in R} \kappa_x dA_x. \tag{1}$$

Figure 2 shows a level set component offset upwards by a hypervolume δ. The direction of offset is upward if the function value increases and downward otherwise. We simplify the Jacobi set by computing offsets in an appropriate direction.

The following basic offset operations are used in the simplification process.

Merge : Two components whose edges share a common saddle are offset to the saddle so that they merge. The merged component is further offset by a small value resulting in a single component.

Split : A component is offset to a saddle and is further offset by a small value resulting in a split.

Purge : A component is offset to a local maximum or minimum. A further offset by a small value removes the component.

Create : A component is created at a local maximum or minimum and offset by a small value.

Figure 3 illustrates the basic offset operations, using the Reeb graph. The Reeb graph is naturally suited to represent the offsets because it traces the connected components of the level sets. Only two operations result in a reduction in the number of components. Temporary splits may be required to obtain a small number of components. We ensure that the number of splitting operations is lower than the number of component merging operations. We show in the next section that twice the total hypervolume swept during the operations is an upper bound over the total change in relationship between the functions.

The first step in the simplification procedure is the computation of the Reeb graph for κ_x^S. Arcs in the Reeb graph that contain the zero level set are also identified.

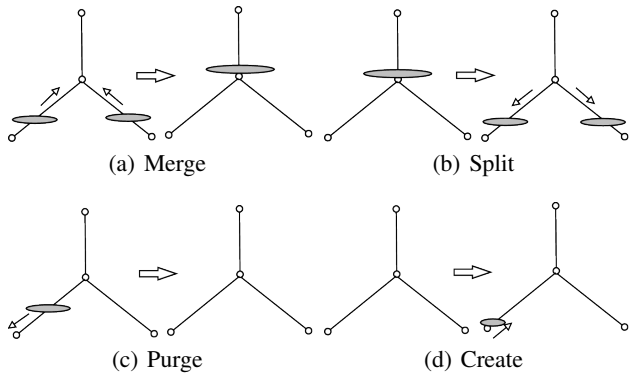


Fig. 3. Different offset operations used during simplification. All offsets are shown against the Reeb graph of κ_x^S.

Algorithm. The required simplification is specified as a percentage of the global comparison measure. The corresponding hypervolume threshold, i.e., the total hypervolume allowed for the operations is calculated next. Since each simplification operation involves exactly one critical point, we can represent an offset by a critical point. We first augment the Reeb graph by adding dummy nodes at level zero. This augmented graph is transformed into a directed graph by replacing each arc uv with a directed arc uv (arc from u towards v), if $|\kappa_v^S| > |\kappa_u^S|$, see Fig. 4 . Each vertex is then assigned a profit $P(v)$

given by

$$P(v) = \begin{cases} 1 & \text{if } v \text{ is a dummy vertex} \\ in(v) - out(v) & \text{otherwise,} \end{cases}$$

where $in(v)$ and $out(v)$ represent the indegree and outdegree of v in the directed graph. The profit for a non dummy node signifies the reduction in number of Jacobi set components if the operation corresponding to the node is chosen. The optimal simplification can now be formulated as an integer linear program (ILP) that maximizes profit. The variables in the ILP correspond to nodes of the directed Reeb graph.

$$max \sum P(v)x_v$$

subject to constraints

$$\begin{aligned} &\sum C(v)x_v \leq T \\ &x_v - x_u \leq 0 \quad \text{for a directed arc } uv \\ &x_u + x_v \leq 1 \quad u,v \text{ adjacent to a common dummy node} \\ &x_u, x_v \in \{0,1\} \end{aligned}$$

The cost $C(v)$ for each simplification operation is the sum of hypervolumes of the incoming arcs. T is the threshold given as input. A simplification operation is performed on a node if the corresponding variable in the ILP is set to one. The first constraint bounds the total hypervolume for the simplification. The second constraint enforces a dependency between variables corresponding to a directed arc uv. This dependency captures the fact that a simplification operation at v can be performed only after a level set component has been offset through the node u. At dummy vertices, there is a choice to perform an offset in either of the directions but not both. This choice is modeled in the third constraint. The ILP is a variant of the knapsack problem with dependencies among objects. Though a solution to the above ILP corresponds to the optimal simplification, the computation is slow in practice. So, we resort to a greedy strategy that chooses the least cost offset operation at every step until the threshold is reached. The greedy strategy has an additional advantage-it enables the creation of a multi-resolution representation of the Jacobi set.

The greedy algorithm requires all nodes to be stored in a priority queue. The priority queue is initialized with all possible simplification operations and updated with new operations that may become valid after an offset is performed. We define a node of the directed Reeb graph as *unreachable* if it cannot be reached by a path from a dummy node and *reachable* otherwise. Unreachable nodes may become obstacles that prevent offset operations. For example, a saddle with an incoming arc from an unreachable node prevents a merge operation, see Fig.5. Let G denote the directed Reeb graph and H denote the subgraph of G containing all unreachable vertices. A component J of H is a connected component in the undirected version of H. The cost of removing J is the sum of the cost of all edges of G that have at least one end point in J. If the algorithm is not able to proceed due to some obstacles, then least cost components of unreachable vertices are removed from G until a valid operation is identified. Finally, we extract offset components using seed sets stored in the Reeb graph [7]. We also

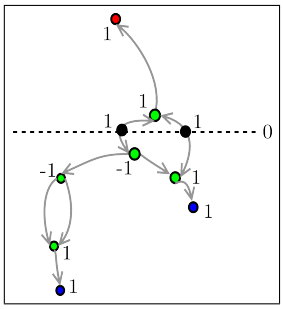


Fig. 4. Directed Reeb graph. The dotted line in the figure shows level 0. The dummy vertices are shown in black on the zero line. The profit for each node is also shown.

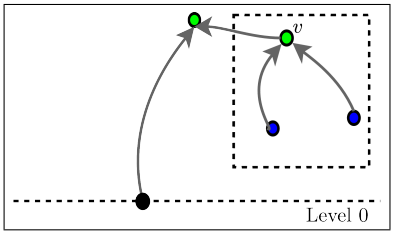


Fig. 5. A Section of a Reeb graph with unreachable vertices shown in the boxed rectangle. The unreachable component prevents the algorithm to proceed beyond the merge saddle v.

ensure that the number of simplification operations with negative profits is smaller than a constant fraction of the operations with positive profits. This ensures that the number of components decreases as a result of simplification.

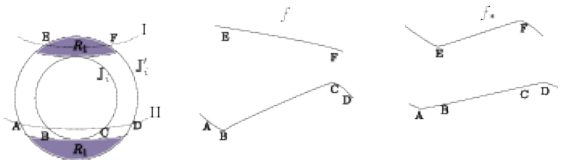
4 Analysis

In this section we show that twice the hypervolume swept during a simplification operation is an upper bound over the change in the relationship between the input functions.

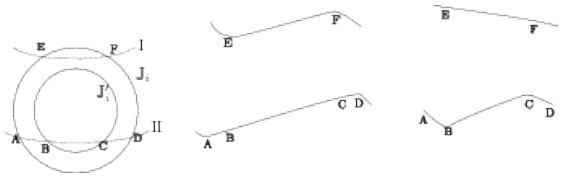
Simplifying the input function. We do not change the function values in our experiments. However, we now compute changes to the function f caused by a small offset in order to obtain the upper bound. Figures 6a and 6b depict the changes to the function f after offsets in the up and down directions respectively. An upward offset introduces critical points at E and F of f restricted to level set I of g. To accomplish this, the function values at E and F can be interchanged to become $f(F)$ and $f(E)$ respectively. Within level set II of g, the critical points of f move from B and C to A and D respectively. The function f restricted to level set II between A and D is made monotone to achieve this movement of critical points. The function values at A and D do not change and therefore the new pair have a reduced persistence. Downward offset destroys the critical point pair E and F and the restricted function f between E and F is made monotone. The function values at E and F are interchanged to become $f(F)$ and $f(E)$ respectively. Within level set II of g, critical points move from A and D to B and C respectively.

Effect on global comparison measure. As shown by Edelsbrunner et al. [5], the global comparison measure is given by

$$\kappa = \frac{2}{Area(\mathbb{M})} \int_{v \in \mathbb{J}} sign(v) f(v) dg,$$



(a) Upward Offset



(b) Downward Offset

Fig. 6. Simplifying the input function. The left column shows a Jacobi set component $\mathbb{J}_i$ and its offset version $\mathbb{J}'_i$. The dashed lines are level sets of the function g. The center column shows f restricted to the level sets I and II. The right column shows the simplified function f_* that corresponds to the offset Jacobi set component $\mathbb{J}'_i$.

where $sign(v)$ is defined as

$$sign(v) = \begin{cases} +1 \text{ if } v \text{ is a maximum of } f_{|g^{-1}(g(v))} \\ -1 \text{ otherwise.} \end{cases}$$

Let $\mathbb{J}_i$ denote the i^{th} component of the Jacobi set. Define

$$\kappa_i = \frac{2}{Area(\mathbb{M})} \int_{v \in \mathbb{J}_i} sign(v) f(v) dg.$$

κ_i can be interpreted as the contribution of $\mathbb{J}_i$ to the global comparison measure, $\kappa = \sum_i \kappa_i$.

Since the change to the function f corresponding to an offset is local to the region of the component, we will now compute the change in κ_i corresponding to an upward offset. If f_* is the modified function, the change in κ_i is given by

$$|\delta \kappa_i| = \frac{2}{Area(\mathbb{M})} \Big| \int_{v \in \mathbb{J}'_i} sign(v) f_*(v) dg - \int_{v \in \mathbb{J}_i} sign(v) f(v) dg \Big|.$$

Let R be the region of $\mathbb{M}$ swept during the offset and R_1 be the region where the level sets of g do not intersect $\mathbb{J}_i$ (shaded region in Fig.6a). The integral over $\mathbb{J}'_i$ can be rewritten as a sum of integrals over two regions:

$$\begin{aligned} |\delta \kappa_i| = \frac{2}{Area(\mathbb{M})} \Big| & \int_{v \in \mathbb{J}'_i \cap R_1} sign(v) f_*(v) dg - \int_{v \in \mathbb{J}_i} sign(v) f(v) dg \\ & + \int_{v \in \mathbb{J}'_i \cap (R - R_1)} sign(v) f_*(v) dg \Big|. \end{aligned} \tag{2}$$

Consider the level sets I in Fig.6a. The difference between function values at E and F can be written as

$$f_*(F) - f_*(E) = f(E) - f(F) = \int_F^E \|\nabla f_t(x)\| dl.$$

Here, $\nabla f_t(x)$ represents the tangential component of $\nabla f(x)$ along the level sets and dl is the length element along the level set. The integral of $sign(v)f_*(v)$ over $\mathbb{J}_i' \cap R_1$ can be rewritten as an integral over R_1 using the above expression,

$$\int_{v \in \mathbb{J}_i' \cap R_1} sign(v) f_*(v) dg = \iint_{x \in R_1} \|\nabla f_t(x)\| dl dg.$$

Let du be the length element orthogonal to the level set. The area element is given by $dldu$. Using the fact that $dg = \|\nabla g(x)\| du$,

$$\begin{aligned} \int_{v \in \mathbb{J}_i' \cap R_1} sign(v) f_*(v) dg = \iint_{x \in R_1} \|\nabla f_t(x)\| \|\nabla g(x)\| dl du &= \int_{x \in R_1} \|\nabla f(x) \times \nabla g(x)\| dA_x \\ &= \int_{x \in R_1} \kappa_x dA_x. \end{aligned} \tag{3}$$

Consider the level set II of g in Fig.6a:

$$f_*(A) = f(A) = f(B) + \int_B^A \|\nabla f_t(x)\| dl$$

and

$$f_*(D) = f(D) = f(C) - \int_D^C \|\nabla f_t(x)\| dl.$$

Combining the above two equations,

$$\begin{aligned} (f(C) - f(B)) - (f_*(D) - f_*(A)) &= (f_*(A) - f(B)) + (f(C) - f_*(D)) \\ &= \int_B^A \|\nabla f_t(x)\| dl + \int_D^C \|\nabla f_t(x)\| dl. \end{aligned}$$

All pairs of points $A, D \in \mathbb{J}_i' \cap (R - R_1)$ have a corresponding pair $B, C \in \mathbb{J}_i$. So, we have

$$\begin{aligned} \Big| \int_{v \in \mathbb{J}_i' \cap (R - R_1)} sign(v) f_*(v) dg - \int_{v \in \mathbb{J}_i} sign(v) f(v) dg \Big| &= \iint_{x \in (R - R_1)} \|\nabla f_t(x)\| \|\nabla g(x)\| dl du \\ &= \int_{x \in (R - R_1)} \kappa_x dA_x. \end{aligned} \tag{4}$$

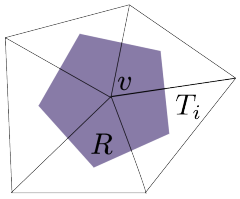


Fig. 7. Consider the vertex v and its adjacent vertices as a point set. The neighborhood R of a vertex v on a piecewise linear surface is represented by the Voronoi region of v.

Substituting (4) and (3) in (2) and using the triangle inequality,

$$|\delta k_i| \leq \frac{2}{Area(\mathbb{M})} \int_R \kappa_x dA_x = 2H.$$

The above inequality can be similarly derived for the downward offset. Thus, the hypervolume is a conservative estimate of the change in relationship between f and g caused by an offset.

5 Implementation for Piecewise Linear Functions

Scalar scientific data is typically represented by piecewise linear functions on triangle meshes, where the gradient and hence κ_x^S is not defined at vertices of the mesh. Given a vertex v of the triangle mesh, its neighborhood is the Voronoi region as shown in Fig. 7. Meyer et al. [9] used the Voronoi region to define discrete differential operators with minimal numerical error for triangulated surfaces. Let $T_1, T_2, \cdots, T_t$ be triangles that intersect the neighborhood R of v. The sign extended comparison measure κ^S is constant within each of the regions $T_i \cap R$. We follow Meyer et al. to define κ_v^S as the average value of the sign extended measure over R;

$$\kappa_v^S = \frac{1}{Area(R)} \sum_{i=1}^{t} \kappa_i^S Area(T_i \cap R),$$

where κ_i^S is the value of the sign extended comparison measure at a point that lies in the interior of T_i. Note that the gradients of f and g are constant in the interior of a triangle and hence κ_x^S is also constant within a triangle. The sign extended comparison measure is stored at vertices and a linear approximation is used within the edges and triangles. This approximation does not introduce significant artifacts in practice. The zero level set can be extracted using a marching triangles algorithm or from seed sets computed using a Reeb graph of κ_x^S.

6 Applications

We demonstrate the usefulness of the simplified Jacobi set using two different applications. Our approach to the definition and simplification of Jacobi sets is particularly useful when studying the relationship between two functions using their gradients.

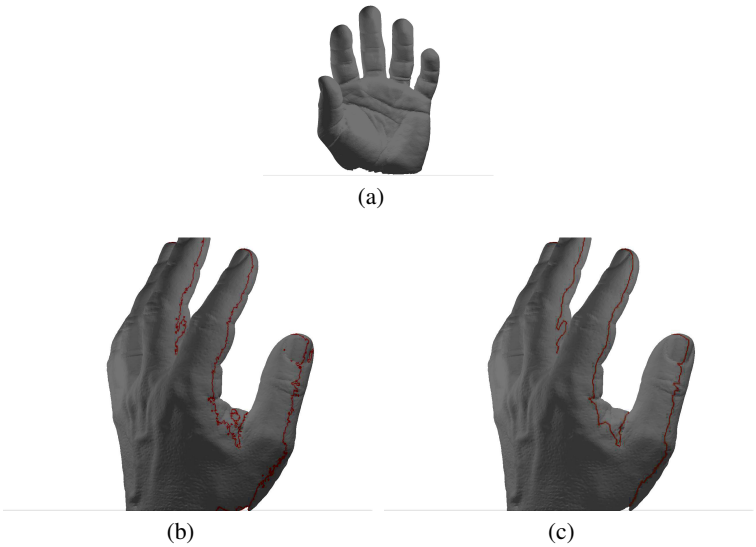
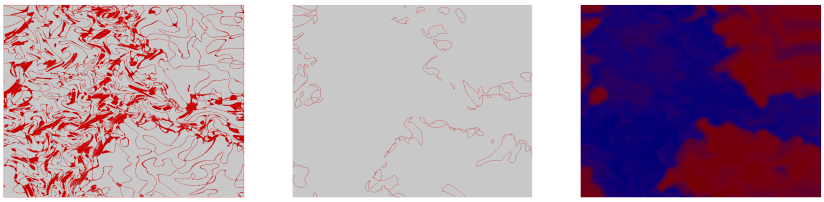


Fig. 8. Silhouettes. (a) Model of a hand in its original orientation. (b) Silhouette when viewed from a different angle. (c) Simplified silhouette (color plate C. 7, page 256).

Visualizing Silhouettes. Given a view direction d in $\mathbb{R}^3$ and a 2-manifold $\mathbb{M}$ embedded smoothly in $\mathbb{R}^3$, the silhouette is the set of points in $\mathbb{M}$ where the tangent plane is parallel to d. Consider a Cartesian coordinate system with the z-axis along the view direction d. The Jacobi set of the two scalar fields $f(x,y,z) = x$ and $g(x,y,z) = y$ is the required silhouette.The silhouette of a model of the hand is shown in Fig.8(c). The model is shown in the original orientation in Fig.8(a). The view direction is perpendicular to the plane of paper. The orientation of the model has been changed for a better view of the computed silhouette in Figs.8(b) and 8(c). As seen from the figure, the silhouette has many components that are unimportant and the silhouette itself appears to contain noise. The simplification process removes small components because their removal does not adversely affect the relationship between the fields f and g used to compute the silhouette. We found that simplification using 2% threshold removed all noise. The Jacobi set was simplified using the greedy algorithm.

Combustion. We apply our algorithm to study a time varying dataset from the simulation of a combustion process. This application demonstrates the use of simplification when handling degenerate data. Degeneracies occur when κ_x is zero within a region, resulting in the Jacobi set containing higher dimensional parts. During simplification, Jacobi set components within degenerate regions are automatically removed because they do not contribute to κ.

The dataset consists of the concentrations of H_2(fuel) and O_2(air) defined on a 600x600 grid for 67 time steps. We compute and simplify the Jacobi set for H_2 and O_2 at different time steps to identify the front of combustion. Combustion begins at



(a) (b) (c)

Fig. 9. Combustion. (a) Jacobi set of H_2 and O_2 in the 64th time step. (b) Simplified Jacobi set. (c) Concentration of O_2 (color plate C. 8, page 256).

regions where the fuel-air mixture is appropriate for ignition. The data is degenerate away from the front, thereby introducing noise in the Jacobi set.
Figure 9 shows the results for the 64th time step when the combustion is in its final stage. The simplified Jacobi set again appears at the front. Figure 9(c) shows the O_2 concentration. Blue signifies a low function value and red signifies a high function value.The front consists of the boundary of red regions, which is also traced by the simplified Jacobi set.

7 Conclusions

We have described a robust algorithm for simplifying the Jacobi set of two Morse functions. Our algorithm ensures minimal change to the relationship between the two functions. Future work includes extending the algorithm to multiple functions and higher dimensions.

Acknowledgments

This work was supported by the Department of Science and Technology, India under grant SR/S3/EECE/048/2007. The combusion data was provided by Jackie Chen and Valerio Pascucci. We also thank the anonymous reviewers for their feedback.

References

1. J. Bennett, V. Pascucci, and K. Joy. Genus oblivious cross parameterization: Robust topological management of inter-surface maps. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 238–247, Washington, DC, USA, 2007. IEEE Computer Society.
2. P-T Bremer, E. M. Bringa, M. A. Duchaineau, A. G. Gyulassy, D. Laney, A. Mascarenhas, and V. Pascucci. Topological feature extraction and tracking. *Journal of Physics: Conference Series*, 78:012007 (5pp), 2007.
3. T. Echekki and J. H. Chen. Direct numerical simulation of auto-ignition in inhomogeneous hydrogen-air mixtures. In *Proceedings of the 2nd Joint Meeting U.S. Sections Combustion Institute*, 2001.

4. H. Edelsbrunner and J. Harer. Jacobi set of multiple morse funtions. In *Foundations of Computational Mathematics, Minneapolis, 2002*, pages 37–57. Cambridge Univ. Press, 2004.
5. H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Local and global comparison of continuous functions. In *Proceedings of the conference on Visualization '04*, pages 275–280, Washington, DC, USA, 2004. IEEE Computer Society.
6. H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 454, Washington, DC, USA, 2000. IEEE Computer Society.
7. M. Kreveld, R. Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 212–220, New York, NY, USA, 1997. ACM.
8. Y. Matsumoto. *Introduction to Morse Theory*. Translated from Japanese by K. Hudson and M. Saito. Amer. Math. Soc., 2002.
9. M. Meyer, M. Desbrun, P. schroder, and A. Barr. Discrete differential geometry operators for triangulated 2-manifolds. *VisMath.*, 2002.
10. G. Reeb. Sur les points singuliers d'une forme de pfaff complèment intégrable ou d'une fonction numérique. *Computes Rendus de L'Académie des Séances*, 222:847–849, 1946.

Reconstructing Cell Complexes From Cross-sections

Scott E. Dillard[1,2], Dan Thoma[1] and Bernd Hamann[2]

[1] Materials Design Institute, Los Alamos National Laboratory
[2] Institute for Data Analysis and Visualization, Department of Computer Science, University of California, Davis

Abstract. Many interesting segmentations take the form of cell complexes. We present a method to infer a 3D cell complex from of a series of 2D cross-sections. We restrict our attention to the class of complexes whose duals resemble triangulations. This class includes microstructures of polycrystalline materials, as well as other cellular structures found in nature. Given a prescribed matching of 2D cells in adjacent cross-sections we produce a 3D complex spanning these sections such that matched 2-cells are contained in the interior of the same 3-cell. The reconstruction method considers only the topological structure of the input. After an initial 3D complex is recovered, the structure is altered to accommodate geometric properties of the dataset. We evaluate the method using ideal, synthetic datasets as well as serial-sectioned micrographs from a sample of tantalum metal.

1 Introduction

Cross-section imaging is a common technique used to study and analyze the structure of materials. A series of planar cross-sections through a specimen is generated and the cross-sections are used to reconstruct a 3D representation. A related, well-studied problem in computer graphics and visualization asks to construct a surface from a set of curves lying in the cross-sections. There have been many solutions suggested for the *two-phase* version of this problem, in which the reconstructed surface divides space into two "phases" and hence can be a manifold. For example, the zero-surface of a continuous scalar function divides the domain into two phases: points with negative values and points with non-negative values. In this paper we consider the *multiphase* generalization of this problem, in which the non-manifold separating surface divides space into multiple phases. This distinction in illustrated in Figure 1.

Solutions to the problem of constructing a surface from cross-sections can be divided into two types: *mesh-based* ones, operating on an irregular collection of vertices, edges and faces, and *voxel-based* ones, operating on a 3D array of sample points. See the papers by Nonato *et al.* [6] and Braude *et al.* [1] and the references therein for examples of mesh and voxel-based solutions, respectively. Previous solutions to the multiphase problem have all been of the voxel-based type. See the paper by Dillard *et al.* [3] and references therein for examples.

We present a novel mesh-based solution to the multiphase segmentation reconstruction problem. In voxel-based methods, topological and geometric properties are conflated, which is beneficial if one wants to optimize some geometric criterion (e.g., smoothness) without being hindered by topological constraints (e.g., genus). In the multiphase problem, however, direct control over reconstructed topology can be important.

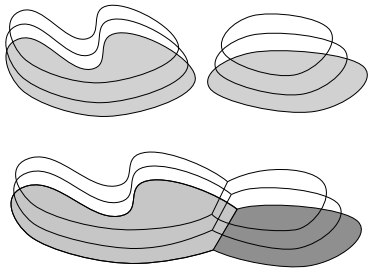


Fig. 1. In a two-phase segmentation (top) every point is incident on at most two regions. In a mult-phase segmentation (bottom) there are "triple points."

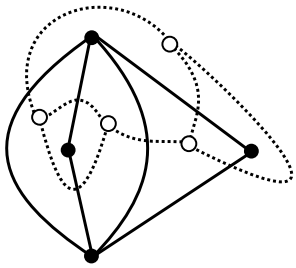


Fig. 2. A Δ-complex, in solid dots and lines, and its dual Δ^*-complex in hollow dots and dashed lines.

An example of this kind of control is given by Nonato *et al*. [6] in their mesh-based β-connection method, a solution to the two-phase reconstruction problem. Their method provides a user with control over the reconstruction topology through a parameter β. Higher values of β favor reconstructed surfaces of higher genus. Our method supports a similar type of control in the multiphase setting: The user can prescribe that two regions in two adjacent cross-sections should be path-connected, or not. A significant challenge in the mesh-based setting is the avoidance of self-intersections in the constructed surface, a problem that most voxel-based methods avoid by design.

The data we consider are serial-sectioned micrographs of polycrystalline material. Each phase represents a grain, a region of uniform crystal structure. The ability to prescribe connections between 2D cross-section regions is important when the imaging mode provides more information than just region boundary geometry. In the case of metal micrography, electron back-scatter diffraction (EBSD) measures crystallographic orientations, and we therefore prescribe a connection between two cross-section regions if their orientations are similar and they are relatively close to each other.

2 Definitions

Many of the terms are familiar from simplicial complexes, with subtle yet important differences in definition. A d-simplex $\Delta^d \subset \mathbb{R}^d$ is the convex hull of $d+1$ affinely independent vertices. A face of a simplex is the $(d-1)$-simplex obtained by removing a vertex. Let $\partial\Delta^d$ be the union of all faces of Δ^d, and $\mathring{\Delta}^d = \Delta^d \setminus \partial\Delta^d$. Let the sequence $(v_0 \dots v_d)$ be an ordering of the vertices of Δ^d, then the ordering of the ith face is $(-1)^i(v_0 \dots \hat{v}_i \dots v_d)$ where the "hat" indicates that v_i is removed and a negative coefficient swaps the first two vertices of the sequence. Let X be a topological space and let $\sigma : \partial\Delta^d \to X$ be a continuous map. A new space $Y = X \cup_\sigma \Delta^d$ is obtained from X by *attaching a cell* to X, giving Y the quotient topology of $X \cup \Delta^d/\sim$, where $y \sim \sigma(y)$ for all $y \in Y$. A d-dimensional *cell complex*, or just *d-complex*, is a topological space constructed by attaching cells of dimension no greater than d. A **Δ-complex** is a collection of maps $\sigma_i : \Delta^d \to X$ such that:

1. The restriction $\sigma_i|\mathring{\Delta}^d$ is injective, and each point of X is in the image of exactly one such restriction. We say that σ_i is a *d-cell*, and let $\mathring{\sigma}_i$ denote the restriction. Referring to σ_i in the context of a set refers to its range.
2. Each restriction of σ_i to one face of Δ^d is one of the maps $\sigma_j : \Delta^{d-1} \to X$, identifying the face of Δ^d with Δ^{d-1} by the linear homeomorphism that preserves the ordering of vertices. We say that σ_i is *attached* to σ_j.
3. The restriction $\sigma_i|\partial\Delta^d$ is injective.
4. A set $A \subset X$ is open iff $\sigma_i^{-1}(A)$ is open in Δ^d for each i.

Conditions 1, 2 and 4 follow Hatcher [4]. We impose condition 3 to prevent degeneracies such as loops. This class of complexes contains simplicial complexes, but is broader. For instance, we may have multiple 1-cells each incident on the same pair of 0-cells. Informally, one may think of a Δ-complex as a triangulation with curved edges and faces.

Two cells σ_i and σ_j are *incident* on each other if $\sigma_i \cap \sigma_j \neq \emptyset$. Two d-cells are *adjacent* if there exists a $(d-1)$-cell and a $(d+1)$-cell on which they are both incident. Let all 0-cells be incident on a single (-1)-cell, and if σ_i and σ_j are of maximal dimension, let them be adjacent only if they are incident on a common $(d-1)$ cell. Since we are concerned here only with 2-complexes and 3-complexes we make the following abbreviations: A *vertex* is a 0-cell. An *edge* is a 1-cell. A *face* is a 2-cell, and henceforth *only* a 2-cell. A *tetrahedron* is a 3-cell of a Δ-complex. A 2-complex that is also a Δ-complex is a 2Δ-complex, and a 3Δ-complex is defined analogously. The *valence* of a vertex is the number of edges that are incident on it.

Let $\mathscr{C}$ be a cell complex. $\mathscr{C}'$ is a *subcomplex* of $\mathscr{C}$ if $\mathscr{C}'$ is a cell complex and $\mathscr{C}' \subseteq \mathscr{C}$. The *closure* of a set of cells is the smallest subcomplex containing it. Two d-complexes $\mathscr{C}$ and $\mathscr{C}'$ are *isomorphic* if there exist bijections $M_k : \mathscr{C}_k \to \mathscr{C}'_k$, $0 \leq k \leq d$, such that if cells $a, b \in \mathscr{C}$ are adjacent then so are $M_k(a)$ and $M_k(b)$. Two d-complexes $\mathscr{C}$ and $\mathscr{C}^*$, are *duals* if there exist bijections $D_k : \mathscr{C}_k \to \mathscr{C}_{d-k}$, $0 \leq k \leq d$ such that if $a, b \in \mathscr{C}$ are adjacent then so are $D_k(a)$ and $D_k(b)$. For example, in a pair of dual 3-complexes, tetrahedra of one complex are mapped to vertices of the other, and faces to edges. Our method is restricted to a certain class of segmentations because it exploits the structure of the dual complex. In particular, the dual must be a Δ-complex. Correspondingly, we call the complex that represents the segmentation a Δ^*-complex. An example of dual Δ and Δ^*-complexes is shown in Figure 2.

A bijection f is a *homeomorphism* if both f and f^{-1} are continuous. If such f exists, its domain and range are *homeomorphic*. A Δ-complex is a *manifold* if for every vertex v, the union of the interiors of cells incident on v (the *star* of v) is homeomorphic to the open unit ball $\{x : \|x\| < 1\} \subset \mathbb{R}^d$. The relevant implication is that every $(d-1)$-cell in a manifold d-complex is incident on two d-cells. A complex is a *manifold with boundary* if the star of every vertex is homeomorphic to the open unit ball or the half-ball $\{x : x_1 \geq 0, \|x\| < 1\} \subset \mathbb{R}^d$, and its *boundary* is the closure of those $(d-1)$-cells that are incident on only one d-cell. A Δ-complex is a *sphere* if it is homeomorphic to the standard sphere $\{x : \|x\| = 1\} \subset \mathbb{R}^d$, and a *ball* is homeomorphic to $\{x : \|x\| \leq 1\}$.

A *homotopy* between two continuous functions $f, g : X \to Y$ is a continuous function $h : [0,1] \times X \to Y$, such that $h(0,x) = f(x)$ and $h(1,y) = g(y)$. Denote the existence of h by $f \simeq g$. X and Y are *homotopy equivalent* if $f \circ g \simeq \mathrm{id}_Y$ and $g \circ f \simeq \mathrm{id}_X$, where

$\mathrm{id}_X(x) = x \in X$. A space is *contractible* if it is homotopy equivalent to a point. Crucially, if $\mathscr{S}$ is a contractible subcomplex of $\mathscr{C}$, then the complex $\mathscr{C}'$ obtained by replacing $\mathscr{S}$ with a vertex is homotopy equivalent to $\mathscr{C}$ [4]. Because homotopy equivalence is transitive, we may also replace $\mathscr{S}$ by any other contractible complex.

2.1 Edge Flip

An important operator for 2Δ-complexes is the *edge flip*, which modifies a complex by replacing two adjacent faces. Let face f be attached to edges s, r and q, spanning vertices a, d and c. Similarly, face g is attached to edges r, t and p spanning vertices a, c and b. This is shown in Figure 3(a). We flip the edge r by first removing f and g, then removing r, then reattaching r' to b and d, then reattaching faces f' and g' to edges r', p, q and r', s, t, as shown in Figure 3(b). The flip *consumes* f and *produces* f'.

In a 2D simplicial complex, a flip must not be performed if the vertices to be connected by the new edge are already connected. Doing so collapses the space because every simplex is uniquely determined by its vertices. This constraint is overly restrictive because a Δ-complex admits multiple edges between the same pair of vertices. Let the *apex* of a triangular face with respect to an edge e be the vertex of that face which is not incident on e. In a manifold 2Δ-complex, we allow an edge e to flip if the apexes of its two incident faces, f and g, are not the same vertex. If this is the case, then the closure of $\{f,g\}$ is contractible and the complex resulting from the flip is homotopy equivalent to the original. If the two apexes of f,g are the same, then flipping e creates a loop. While not immediately changing the topology of the complex, loops complicate further operations such as edge contractions, so we avoid them altogether. An example of a non-flippable edge is shown in Figure 3(c), labeled r.

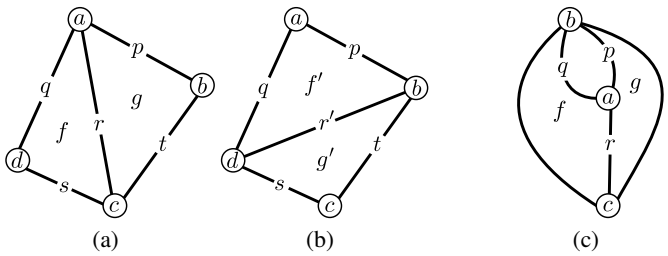


Fig. 3. (a) and (b) show the process of flipping edge r. In (c), edge r is not flippable. The apexes of f and g with respect to r are both b, so flipping r would create a loop.

2.2 Edge Contraction

An *edge contraction* modifies a 3Δ-complex by merging two adjacent vertices. We decompose the edge contraction into three operations that contract a 1-cell, some 2-cells and some 3-cells. Collapsing the initial edge removes the edge e and one vertex. The resulting complex is not a Δ-complex: every face that was incident on e is now a

2-cell incident on only two edges. To restore the Δ-complex property we contract these 2-cells, but this still does not create a Δ-complex: every tetrahedron that was incident on e is now a 3-cell incident on only two faces, resembling a triangular "pillow." We finally contract these 3-cells to restore the Δ-complex property. This process is shown in Figure 4.

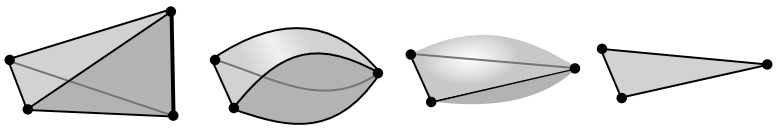


Fig. 4. An edge contraction.

This operation is decidedly different from the edge contraction operation for simplicial complexes. For that operation, the condition to preserve the topology of the complex is much stricter [2]. In Δ-complexes the condition is more relaxed: As with edge flipping, we forbid edge contractions that create loops. If vertices u and v are both incident on the same pair of edges, then neither of those edges may be contracted. As a consequence of condition 3 in the definition of a Δ-complex, the closure of every cell is a contractible subcomplex, and thus the intermediate complexes during the edge contraction process are all homotopy equivalent. Avoiding the creation of loops suffices to preserve condition 3. To see that the result is a Δ-complex, note that every tetrahedron incident on the contracted edge is turned into a valid face, and every incident d-cell, $d < 3$, is removed. This holds only as long as $\mathscr{C}'$ has enough vertices to satisfy the definition of a Δ-complex.

3 Algorithm

The problem is defined as follows: Let K_1 and K_2 be two 2-sphere Δ-complexes, and let P be a prescribed matching between their vertices, such that each vertex is matched with at most one other vertex, and that vertex is in the other complex. These complexes are the duals of cross-sections of a segmentation, and their vertices represent the regions or "phases" of that segmentation. The output is a 3-sphere Δ-complex $\mathscr{C}$ that contains subcomplexes $\mathscr{C}_1$ and $\mathscr{C}_2$ isomorphic to K_1 and K_2, respectively. Additionally, for vertices u and v, $u \in K_1$, $v \in K_2$, if $(u,v) \in P$, then the isomorphisms $M_i : K_i \to \mathscr{C}$ map u and v to the same $v' \in \mathscr{C}$. In other words, $\mathscr{C}$ connects K_1 to K_2 and unifies matched vertices.

K_1 and K_2 are first combined to create a single Δ-complex K. To do so, nest K_1 inside K_2, then find two faces, f_1 and f_2, from K_1 and K_2, respectively, and connect them with a ball 3Δ-complex containing f_1 and f_2 on its boundary, like a triangular prism. Call this prism F. If possible, F should be chosen so that its three pairs of vertices each match under P. Next, two new vertices are created, v_1 and v_2, and each v_i is connected to K_i by a cone V_i, which is a ball 3Δ-complex containing v_i as a vertex and K_i as its boundary. The cone V_i lies on the side of K_i *opposite* F. The result, $V_1 \cup V_2 \cup F = B$ is a

3-ball. The boundary of this ball, A, is a 2-sphere formed by faces of K_i and F. Figure 5 shows a cut-away diagram of this construction.

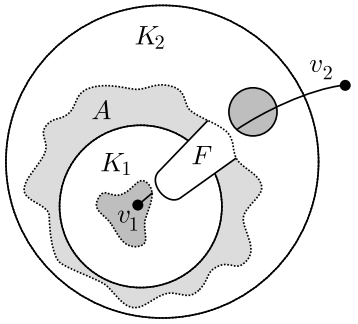
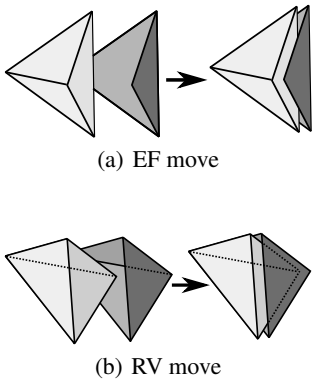


Fig. 5. The initial Δ-complex. Nested spheres K_1 and K_2 are connected by a solid prism F. Vertex v_1 is connected to the interior of K_1 by solid cone, and likewise v_2 is connected to the exterior of K_2. The union of these cones and F forms a ball, and the complement of this is a sphere bounded by the surface A.



(a) EF move

(b) RV move

Fig. 6. The two boundary operations used to modify the active surface. The EF move effectively flips an edge, and the RV move removes a vertex.

The algorithm fills in the space bounded by A, turning B into a 3-sphere. This is accomplished by attaching tetrahedra to A in one of two ways. In each case, B is extended by one additional tetrahedron, and A is updated to track the boundary of B. A loses vertices in the process, until it eventually reduces to one of two 4-vertex configurations, at which point the algorithm terminates. This is ensured because A remains a 2-sphere throughout the process. All operators take place on the surface A so one can think of A as the "active surface." The two operations on A are the following: An *EF* operation adds a tetrahedron c to B by attaching it to two adjacent faces in A. After updating A to track the new boundary, we see that the edge between these faces is flipped. An *RV* operation attaches c to three mutually adjacent faces of A, effectively removing a vertex from the boundary of B. These operations are illustrated in Figure 6.

Edge flips, vertex removals and edge contractions are applied in a goal-directed way. We call a vertex who has no matching a "loner," and likewise, two matched vertices "mates." The goal is to remove every loner and contract an edge between every mated pair. To remove a loner w, we perform EF moves until the valence of w is three, then remove w by an RV move. If u and v are a mated pair then we use EF moves to flip all edges along a shortest path from u to v through the faces of A. Doing so is always possible, and causes u and v to become adjacent via a shared edge which is subsequently contracted, producing a new loner vertex.

Let a *face path* of a 2Δ-complex be a path between vertices u and v consisting of a sequence of faces f_i, $0 \leq i \leq n$, such that u is incident on f_0 and v is incident on f_n, and each f_i is adjacent to f_{i+1}. A face path is *shortest* if no other face path between u and v has fewer faces. When we refer to "the edges of a face path" we mean a sequence of edges e_i such that f_i and f_{i+1} are both incident on e_i, i.e., the edges one crosses

when walking along the path. By the *length* of a face path we mean the number of edges, which is one less than the number of faces. In the proofs below, all complexes are assumed to be 2-spheres without loops.

Lemma 1. *The first edge of a shortest face path between distinct, non-adjacent vertices u and v is not incident on u.*

Proof. Let u, w_1, w_2 be the vertices of f_0. If $e_0 = uw_1$, then u is also incident on f_1 and the path is not shortest. By the same logic, $e_0 \neq uw_2$, leaving only $e_0 = w_1w_2$. □

Lemma 2. *The first edge of a shortest face path between distinct, non-adjacent vertices is always flippable.*

Proof. By Lemma 1, u is incident on f_0 and not incident on e_0, so u is the apex of f_0 with respect to e_0. If e_0 is not flippable, then u is also the apex of f_1. If u is then incident on both faces, and no shortest face path from u to any other vertex passes through e_0. □

We call a face path *flippable* if the first edge is flippable, and after flipping it the remaining path is flippable or empty. The previous lemma implies that shortest face paths are flippable.

Lemma 3. *Let e_i, $0 \leq i \leq n$, $n \geq 1$, be the edges of a flippable face path between distinct, non-adjacent vertices u and v. Flipping e_i in order of increasing i results in a Δ-complex in which u and v are adjacent.*

Proof. Let $n = 1$, then flipping e_0 immediately connects u and v. Now let e_i, $0 \leq i \leq k$ be the edges of a face path between x and u of length k. Assume that by flipping each e_i in sequence, a face s is created with u and x incident on s. Let w be the apex of s with respect to edge ux, let t be the other face on edge ux, and let y be the apex of t with respect to ux. (By assumption $y \neq u$.) After flipping edge ux, y is made adjacent to u. The length of the face path between y and u was $k+1$, so the claim follows from induction on k. □

Lemma 4. *Any sequence of l flips, transforming 2Δ-complex $\mathscr{C}_0$ to $\mathscr{C}_l$, and causing vertices u, v not adjacent in $\mathscr{C}_0$ to become adjacent in $\mathscr{C}_l$, defines a face path between u, v in $\mathscr{C}_0$ of length at most l.*

Proof. Let S_i be a sequence of sets of faces, where S_l contains a face of C_l incident on u and v. Construct S_{i-1} from S_i as follows: Remove from S_i the faces produced by flip i. There are three cases where zero, one or two faces are removed. If one face r is removed, add to S_{i-1} the two faces p, q that are consumed by flip i. If S_i was a connected face path, then so is S_{i-1} because any face s adjacent to r in C_i is adjacent to one of p or q (which are themselves adjacent) in C_{i-1}, or s is consumed by the flip. If two faces are removed, S_i remains connected for the same reason. Thus, if S_l contains a single face, then all S_i are connected face paths. Vertices u and v are each incident on faces of S_i for all i, and therefore the path in S_0 connects them. At each iteration, the cardinality of S is increased by at most 1, so the cardinality of S_0 is at most $l+1$. □

Corollary 1. *If the shortest face path between u and v has length l, then no sequence of fewer than l flips can make u and v adjacent.*

The previous lemma and corollary allow us to guarantee that no mated pair of vertices become inadvertently connected by multiple edges, so that when the time comes to merge this pair the edge contraction is always possible. We define the cost of removing a vertex to be the number of EF moves required to remove it from the active surface. When removing a loner v the cost is $|3 - \text{valence } v|$, because the valence of v must first become three using EF moves before v can be removed with an RV move. When merging a pair of mated vertices u and v, the cost is the length of the face path between u and v.

Lemma 5. *If every mated pair of vertices is either non-adjacent or adjacent by one edge before an iteration, and the iteration performs the cheapest vertex removal, then the same is true after an iteration.*

Proof. Let c be the number of edge flips needed to remove the vertex. No face path of length $c-1$ or less exists between any pair of mated vertices, and thus by Corollary 1, performing $c-1$ flips does not cause any mated pair to become adjacent. After performing $c-1$ of the c flips needed to remove the vertex, it is still true that no other mated pair is adjacent. Flipping a single edge can only add one edge between any pair of non-adjacent vertices, implying that after the final flip every mated pair is still connected by at most one edge. □

If A starts with n vertices, then we perform exactly $n-4$ iterations. In each iteration we perform the cheapest move sequence to merge a mated pair or remove a loner. After $n-4$ iterations, what results is a Δ-complex in one of two configurations: either a tetrahedron or the complex shown in Figure 2, which is a tetrahedron that has had one edge flipped.

Lemma 6. *Any sphere 2Δ-complex with four vertices is isomorphic to either the boundary of a tetrahedron or the complex shown in Figure 2.*

Proof. If each edge is attached to a different pair of vertices, then the edges form a complete graph. The other possibility is that two edges, e and f, are attached to the same pair of vertices, u and v. Because the complex is a 2-sphere, the path from u along e to v and back along f to u separates the complex into two balls which must each contain one of the remaining vertices. □

The algorithm is summarized as follows:

1. Connect the two input spheres with a prism, and fill in the interior of each sphere with a cone. Initialize A to the boundary of the space between the spheres.
2. If A has four vertices, stop. If A is isomorphic to the complex in Figure 2, apply an EF operation to transform it into a tetrahedron.
3. Determine the cheapest sequence of EF and RV moves that either removes a loner or connects a mated pair.
   (a) If the cheapest operation is to remove a loner, then perform EF moves until valence is three, then perform an RV move.

(b) If the cheapest operation is to connected a mated pair of vertices, then find a shortest face path between them. Flip those edges in sequence, then contract the resulting edge between the pair. Mark the newly merged vertex as a loner.

4. Update A and goto 2.

Main Theorem. The 3-sphere Δ-complex constructed by this algorithm contains subcomplexes isomorphic to K_1 and K_2, and this isomorphism identifies vertices of K_1 and K_2 that match under P.

Proof. After first constructing A, $\mathscr{C}$ already contains K_1 and K_2 as subcomplexes. No edge between two vertices of K_1 or two vertices of K_2 is ever contracted, nor any such face removed, so the output complex still contains these subcomplexes. Every matched pair of vertices from K_1 and K_2 is connected by flipping the edges of the shortest face path between them (Lemma 3.) Doing so does not create loops (Lemma 2) nor does it create multiple connections between any mated pairs (Lemma 5) and so the edge between the pair can always be contracted. Every iteration of the algorithm removes one vertex from A while maintaining that A is a 2-sphere Δ-complex, so when the algorithm terminates (Lemma 6) the complex is a 3-sphere.

4 Discussion

An initial 2-sphere can be constructed from a segmented planar image by adding an additional region containing everything "outside" the image. The algorithm can be extended to multiple input cross-sections by turning each one into a 2-sphere and sequentially nesting these spheres like the layers of an onion, then performing the algorithm on the cavity between each pair of consecutive layers. After the algorithm finishes, the cells representing the "outside" are removed, leaving a 3-ball to be embedded in $\mathbb{R}^3$. Most of this embedding is prescribed by the boundary curves of the cross-sections, but the vertices created by the algorithm (dual to tetrahedra of the Δ-complex) need to be placed. We place them by iteratively moving them toward the average position of nearby vertices. Placing them without self-intersections is a challenge.

A straightforward implementation of the algorithm has a run-time complexity bounded by $O(n^3)$ in the worst case, where n is the number of vertices in the initial complex. There are exactly $n-4$ iterations, each iteration having to determine the cheapest move. The cost of mating a pair is the length of the shortest face path between vertices, which can be determined by an $O(n)$ breadth-first search. There can be as many as $O(n)$ mated pairs, thus an upper bound on the running time of $O(n^3)$. In practice, the lengths of shortest face paths are much less than $O(n)$. There is usually a path of nearly constant length close to the interface between K_1 and K_2 in A. Further, if we have already found a shortest path of length l, we can cut future searches short. These two facts make the practical running time nearly linear for well-behaved inputs.

5 Results

To evaluate the 3D models produced by the algorithm, we have reconstructed two types of synthetic ideal cellular structures. Figure 7 shows the reconstruction of a polycrystal

microstructure that has been simulated using the Potts model [5]. The algorithm was run on three different sets of cross-sections of the simulation lattice. One containing every plane of lattice sites, one containing every third plane, and one containing every fifth plane. The cross-sections were extracted from the lattice using marching-triangles, and subsequently simplified using an area-preserving polyline simplification method.

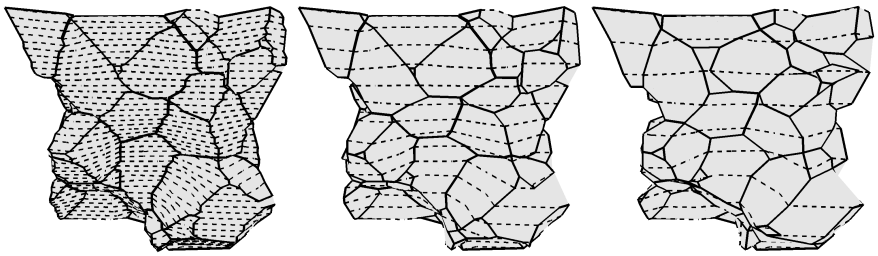


Fig. 7. Three reconstructions of a simulated microstructure on 42^3 cubic lattice. Some exterior grains have been removed for illustration. The solid lines are reconstructed grain edges and the dashed lines are input cross-sections. From left to right the distance between cross-sections is 1, 3 and 5 voxels.

Figure 8 shows the reconstruction of a group of cells called *truncated octahedra*. Each cell is bounded 14 faces, six squares and eight hexagons. Figure 8(a) shows the initial output of the algorithm, in which the reconstructed faces are not all squares and hexagons, a consequence of ambiguity. However this is a good starting point for further refinement, as the faces contain multiple polylines from adjacent cross-sections, which allows the estimation of a plane fitting the face, or a local region of it. Using this information we have modified the reconstructed complex to achieve the correct cell structure, shown in Figure 8(b).

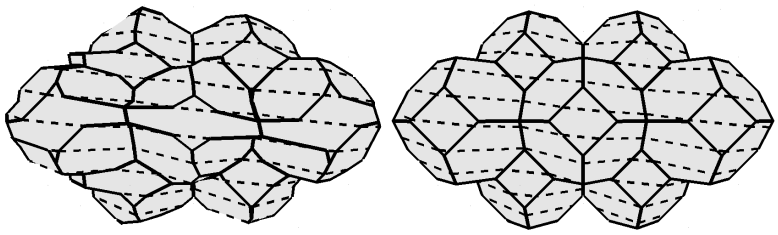


Fig. 8. Two reconstructions of a group of truncated octahedra. Reconstructed edges of the complex are drawn with solid lines and the input cross-sections are drawn with dashed lines. The left image shows the output of the algorithm as presented. The right image shows the complex after local topological modifications guided by the cell face geometry.

This modification process is the subject of ongoing research, so we only sketch the method here. Notice in Figure 8(b) that not every cell edge intersects a cross-section. We group such ambiguous edges into clusters. In the dual, this edge cluster is a cluster of Δ-tetrahedra bounded by non-ambiguous Δ-faces. The interior of the cluster is deleted and then re-tessellated using the following heuristic: Non-ambiguous Δ-faces are associated with lines defined by the triple points from consecutive cross-sections, and during re-tessellation we favor creating Δ-tetrahedra whose faces are associated with lines that come closest to intersecting. Not every edge of the Δ^*-complex intersects enough cross-sections to define a line, but if there are a sufficient number of non-ambiguous edges and faces then their gemoetry can be used to resolve topological ambiguities.

Figure 8(b) was created by placing the unconstrained vertices—those not lying in a cross-section—to minimize the sum of squared distances to the planes of incident faces. This strategy produces accurate reconstructions if ambiguities are correctly resolved, but can cause severe self-intersections if they are not.

Figure 9 shows the reconstruction of a sample of tantalum. The sample was subjected to impact and consequently it exhibits deformed polycrystal grains and a number of small and large voids. There are 1976 cells in total. The running time of the main portion of the algorithm was approximately 30 seconds on a computer with a 2.1 GHz processor. The cross-sections were observed using an EBSD microscope, at a spacing of $25\mu m$. A larger portion of this same dataset has been previously reconstructed using a voxel-based method [3]. The cross-sections in Figure 9 are sparse, and the planar boundary curves are complex, so there are some self-intersections.

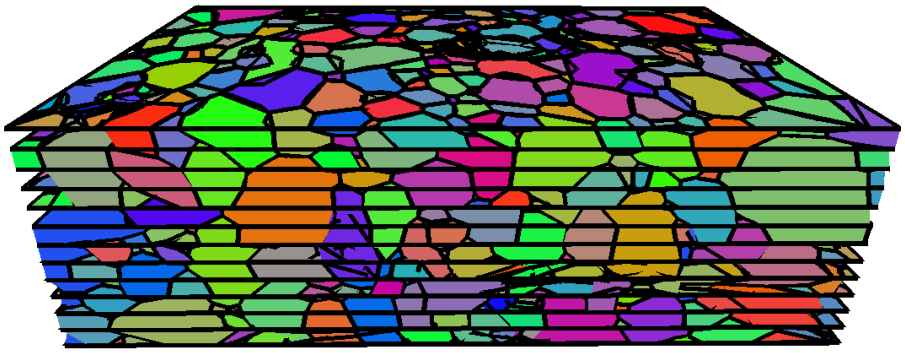


Fig. 9. A portion of a reconstruction of 13 cross-sections of a sample of shocked tantalum. The input cross-sections are indicated by horizontal lines (color plage C. 9, page 256).

Conclusion

We have presented an algorithm for reconstructing a 3D cell complex from a series of 2D cross-sections. Given any matching of regions between cross-sections, the algorithm produces a cell complex that connects matched regions. Self-intersections remain

an outstanding issue. The constructed complex is guaranteed to have simple topology, but embedding it in $\mathbb{R}^3$ may not be straightforward. Indeed since any matching between cross-sections is permitted, it is quite easy to produce intractably tangled cell complexes. The embedding the reconstructed complex in $\mathbb{R}^3$ remains an area of future research.

References

1. I. Braude, J. Marker, K. Museth, J. Nissanov, and D. Breen. Contour-based surface reconstruction using MPU implicit models. *Graphical Models*, 69(2):139–157, 2007.
2. T. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge contraction. *Publ. Inst. Math.(Beograd)(NS)*, 66(80):23–45, 1999.
3. S. Dillard, J. Bingert, D. Thoma, and B. Hamann. Construction of Simplified Boundary Surfaces from Serial-sectioned Metal Micrographs. *IEEE Transactions on Visualization and Computer Graphics*, pages 1528–1535, 2007.
4. A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.
5. E. Holm and C. Battaile. The computer simulation of microstructural evolution. *JOM*, 53(9):20–23, 2001.
6. L. Nonato, A. Cuadros-Vargas, R. Minghim, and M. De Oliveira. Beta-connection: Generating a family of models from planar cross sections. *ACM Transactions on Graphics (TOG)*, 24(4):1239–1258, 2005.

Modeling and Simplifying Morse Complexes in Arbitrary Dimensions

Lidija Čomić[1] and Leila De Floriani[2]

[1] Faculty of Engineering, University of Novi Sad, Novi Sad (Serbia)
comic@uns.ac.rs,
[2] Department of Computer Science, University of Genova, Genova (Italy)
deflo@disi.unige.it

Abstract. Ascending and descending Morse complexes, defined by a scalar function f over a manifold domain M, decompose M into regions of influence of the critical points of f, thus representing the morphology of the scalar function f over M in a compact way. Here, we introduce two simplification operators on Morse complexes which work in arbitrary dimensions and we discuss their interpretation as n-dimensional Euler operators. We consider a dual representation of the two Morse complexes in terms of an incidence graph and we describe how our simplification operators affect the graph representation. This provides the basis for defining a multi-scale graph-based model of Morse complexes in arbitrary dimensions.

1 Introduction

The problem of representing morphological information extracted from discrete scalar fields is a relevant issue in several applications, such as terrain modeling and volume data analysis and visualization. The increasing availability of time-varying volume data sets and the need of extracting knowledge from such data sets makes it important to have a morphological representation of such fields. Time-varying volume data sets are often viewed as four-dimensional scalar fields and 4D models of such data sets based on hypercubic or simplicial meshes have been developed in the literature [3,27]. Morse theory offers a natural and intuitive way of analyzing the structure of a scalar field as well as of compactly representing a decomposition of its domain into meaningful regions associated with critical points of the field.

Discrete scalar fields are defined by a finite set of points in a domain D in $\mathbb{R}^n$. At each of these points a value of a scalar function f is given. Traditionally, discrete scalar fields are described by decomposing their domain into cells, on which an interpolating function is defined based on discrete function values given at the vertices of the cells. This geometry-based description provides an accurate representation of a scalar field, but it fails in capturing the morphological structure of the field, which is defined by its critical points and integral lines.

Based on indexMorse Theory Morse theory, subdivisions of a manifold M, induced by a function f defined over it, have been defined as suitable representations for analyzing the topology of M and the behavior of f over M. The ascending and descending

Morse complexes are defined by considering the integral lines emanating from, or converging to, the critical points of f. The Morse-Smale complex describes the subdivision of M into parts, characterized by uniform flow of the gradient between two critical points of f. Here, we consider a representation, the incidence graph, that encodes both the ascending and descending Morse complexes which is dimension-independent, is based on encoding the incidence relations of the cells of the two complexes, and exploits the duality between the two complexes. The incidence graph can be effectively combined with a representation of the simplicial decomposition of the underlying domain M, in the discrete case.

Structural problems in Morse and Morse-Smale complexes, like over-segmentation in the presence of noise, or efficiency issues arising due to the very large size of the input data sets, can be faced and solved by defining simplification operators on those complexes and on their morphological representations. Morse and Morse-Smale complexes can be simplified by applying an operator called (general) *cancellation* of critical points, which eliminates pairs of critical points of f with consecutive index. A cancellation which does not involve a maximum or a minimum of f increases the number of pairs of cells in the Morse complexes which become incident to each other, and, counter-intuitively, it introduces new cells in the Morse-Smale complex. This motivated us to introduce two simplification operators on n-dimensional Morse complexes by imposing some constraints on a general cancellation. These operators do not enlarge the incidence relation on the Morse complexes, they do not introduce new cells in the Morse-Smale complex, and can be viewed as merging of cells in the Morse complexes. We show that the two simplification operators can be interpreted as Euler operators, and we show through an example how the general cancellation operator can be realized as a sequence of our elementary simplification operators. We discuss the effect of the two simplification operators on the incidence-based representation of the two Morse complexes. These simplification operators, together with their inverse refinement operators, are the basis for generating a dimension-independent multi-scale representation of Morse complexes, based on a hierarchy of incidence graphs. The inverse operators are not discussed here for brevity.

The remainder of the paper is organized as follows. In Section 2, we review some basic notions on indexMorse Theory Morse theory. In Section 3, we discuss some related work. In Section 4, we briefly describe the dual representation of the two Morse complexes. In Section 5, we define two simplification operators on Morse complexes, and in Section 6, we describe the effect of these operators on their incidence-based representation. Finally, in Section 7, we draw some concluding remarks.

2 Morse Theory

We review here some basic notions of indexMorse Theory Morse theory. For more details, see [20].

Let f be a C^2 real-valued function defined over a closed compact n-manifold M. A point p is a *critical point* of f if and only if the gradient $\nabla f = (\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n})$ (in some local coordinate system around p) of f vanishes at p. Function f is a *Morse function* if all its critical points are non-degenerate (the Hessian matrix $Hess_p f$ of the second

derivatives of f at p is non-singular). The number of negative eigenvalues of $Hess_p f$ is called the *index* of critical point p, and p is called an *i-saddle*. A 0-saddle, or an n-saddle, is also called a *minimum*, or a *maximum*, respectively. An *integral line* of f is a maximal path which is everywhere tangent to the gradient of f. Each integral line connects two critical points of f, called its *origin* and *destination*. If m_i is the number of i-saddles of a Morse function f, and $\chi(M)$ is the Euler characteristic of M, then $\chi(M) = \sum_{i=0}^{n} (-1)^i m_i$.

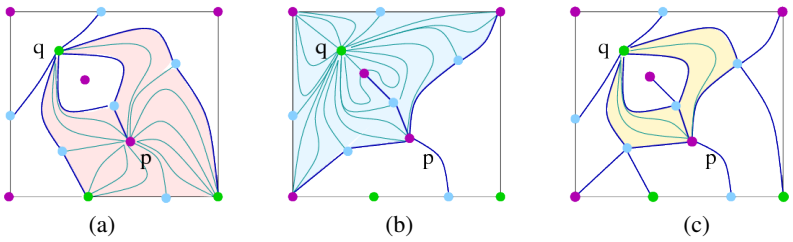


Fig. 1. (a) A 2D ascending Morse complex. The ascending cell of minimum p is shaded. (b) A dual descending Morse complex. The descending cell of maximum q is shaded. (c) Morse-Smale complex. 2-cells related to minimum p and maximum q are shaded (color plate C. 10, page C. 10).

Integral lines that converge to (originate from) a critical point p of index i form an i-cell ($(n-i)$-cell) called a *descending (ascending) cell* of p. The descending (ascending) cells decompose M into a Euclidean cell complex, called a *descending* (*ascending*) *Morse complex*, denoted as Γ_d (Γ_a). A Morse function f is called a *Morse-Smale function* if and only if the descending and the ascending cells intersect transversally. The connected components of the intersection of descending and ascending cells of a Morse-Smale function f decompose M into a *Morse-Smale complex*. We illustrate the correspondence between Morse and Morse-Smale complexes in Figure 1. If f is a Morse-Smale function, then the ascending complex Γ_a of f and the descending complex Γ_d of f are dual to each other. In this work, we restrict our consideration to Morse-Smale functions satisfying an additional condition: in the descending (and ascending) Morse complexes, each i-cell is bounded by at least one $(i-1)$-cell, $1 \leq i \leq n$, and each i-cell bounds at least one $(i+1)$-cell, $0 \leq i \leq n-1$.

3 Related Work

In this Section, we review the state of the art on morphological representation of scalar fields, focusing on algorithms which assume a discretization of the domain of the field as a manifold simplicial complex.

There have been two attempts in the literature to discretize indexMorse Theory Morse theory, either by developing its discrete version, called *Forman theory* [13], by considering functions defined on all cells, and not only on vertices, of a cell complex

Γ, or by representing the combinatorial structure of Morse-Smale complexes in 2D and 3D by *quasi-Morse complexes* [11, 12]. This latter has been the basis for algorithms for computing discrete counterparts of the Morse-Smale complex.

The extraction of critical points of a scalar field f defined on a simplicial mesh has been investigated in 2D [2, 21], and in 3D [11, 14, 24–26], as a basis for computing Morse and Morse-Smale complexes. Algorithms for decomposing the domain D of f into an approximation of a Morse, or of a Morse-Smale complex in 2D can be classified as *boundary-based* [1, 5, 12, 22, 23], or *region-based* [6, 8, 18]. In [11], an algorithm for extracting the Morse-Smale complex from a tetrahedral mesh is proposed. In [8], a region-based dimension-independent algorithm is proposed, which subdivides the domain of f into an approximation of Morse complexes, by processing the points according to sorted function values. For a review of the work in this area, see [4].

One of the major issues that arise when computing a representation of a scalar field as a Morse, or as a Morse-Smale complex is the over-segmentation due to the presence of noise in the data sets. *simplification algorithms* have been developed in order to eliminate less significant features from the Morse-Smale complex. simplification is achieved by applying an operator, called *cancellation* of critical points. In 2D Morse-Smale complexes, cancellation operator has been investigated in [5, 12, 23, 28]. Cancellation operators on Morse and Morse-Smale complexes of a 3D scalar field have been investigated in [7, 15].

4 A Dual Representation for Morse Complexes

In this Section, we briefly describe a dual combinatorial representation for both the ascending and descending Morse complexes based on the *incidence graph* $G = (N, A)$ [10]. The nodes of G are in one-to-one correspondence with the critical points of f. We call a node of G representing an i-saddle of f a node at *level* i. Thus an i-level node in G represents an i-cell of the descending Morse complex Γ_d and an $(n-i)$-cell of the ascending Morse complex Γ_a. Values of the scalar field are attached to the nodes of the incidence graph as well as the level of the node. The direct incidence relations between cells in Γ_d (and thus also in Γ_a) are encoded as arcs. Arcs connect pairs of nodes which differ in level by 1 and represent integral lines connecting the corresponding critical points. An arc exists between an i-level node p and an $(i+1)$-level node q if and only if i-cell p is on the boundary of $(i+1)$-cell q in the descending complex Γ_d (and thus $(n-i)$-cell p is bounded by $(n-i-1)$-cell q in the ascending complex Γ_a). Each arc connecting an i-level node p to an $(i+1)$-level node q is labeled by the number of times the corresponding i-cell p and $(i+1)$-cell q in Γ_d are incident to each other [17]. Note that incidence graph can also be seen as a combinatorial representation of (the 1-skeleton of) a Morse-Smale complex.

In the discrete case, the underlying domain of the field is decomposed into a simplicial mesh, and for this a very compact representation is an indexed data structure with adjacencies, as shown in [9], which encodes only the vertices and the n-simplexes in the n-dimensional simplicial complex and each n-simplex is encoded as the list of the indexes of its $n+1$ vertices. Thus, the storage requirement is equal to n floats for each vertex of the mesh and $n+1$ integers for each n-simplex. We can obtain a com-

bined morphological and geometrical representation by attaching to each n-level node p of the incidence graph, which corresponds to a maximum and to an n-cell p in the descending Morse complex, the set of n-simplexes whose union gives the n-cell p, and, symmetrically, to each 0-level node q of the incidence graph, which corresponds to a minimum and to an n-cell q in the ascending Morse complex, the set of n-simplexes whose union gives the n-cell q. A data structure for 3D Morse-Smale complexes described in [16] represents the connectivity of the complex as an incidence graph, and attaches geometrical information referring to the underlying simplicial decomposition to all nodes of the complex, thus encoding the geometry of 1- 2- and 3-cells in both the ascending and descending Morse complexes. This representation supports efficient traversal, but it is dimension-specific.

5 Removal and Contraction on Morse Complexes

The effect of a cancellation of a pair of critical points, as defined in indexMorse Theory Morse theory, has been investigated in 3D on Morse-Smale [15] and on Morse complexes [7]. An i-saddle p and an $(i+1)$-saddle q can be cancelled if there is a unique integral line connecting them. After a cancellation, each cell r which was on the boundary of $(i+1)$-cell q in Γ_d becomes incident to each cell t which was in the co-boundary of i-cell p in Γ_d. If a cancellation does not involve an extremum, the number of pairs of cells in the Morse complexes which become incident to each other increases, although the number of cells decreases by two, and the number of cells in the Morse-Smale complex increases, although the number of vertices decreases by two. In the current approaches, after a cancellation of a 1-saddle and a 2-saddle in 3D, additional cancellations of maxima and 2-saddles, or minima and 1-saddles, are applied to eliminate the new cells in the Morse-Smale complexes created by the cancellation [15].

Here, we define two operators, which we call *removal* and *contraction*. They are defined in arbitrary dimensions, and are obtained by imposing additional constraints on a cancellation. They do not enlarge the incidence relation on the Morse complexes, they do not introduce new cells in the Morse-Smale complex, and they can be viewed as merging of cells in the Morse complexes.

A *removal* (of index i) of an i-saddle σ_i and an $(i+1)$-saddle σ_{i+1} is defined if σ_i is connected by a unique integral line to

- $(i+1)$-saddle σ_{i+1} and exactly one $(i+1)$-saddle σ'_{i+1} different from σ_{i+1}, or
- exactly one $(i+1)$-saddle σ_{i+1}.

In the first case, a removal of σ_i and σ_{i+1} is denoted as $rem(\sigma_{i+1}, \sigma_i, \sigma'_{i+1})$, and in the second case as $rem(\sigma_{i+1}, \sigma_i, \emptyset)$. In both cases, a removal is specified only by σ_i and σ_{i+1}. We include σ'_{i+1} in the notation to emphasize the condition for the feasibility of the operator, and to highlight the cells merged by it.

After a removal $rem(\sigma_{i+1}, \sigma_i, \sigma'_{i+1})$ of an i-saddle σ_i and an $(i+1)$-saddle σ_{i+1}, i-cell σ_i is deleted in Γ_d, and $(i+1)$-cell σ_{i+1} is merged into $(i+1)$-cell σ'_{i+1}. All cells which were on the boundary of $(i+1)$-cell σ_{i+1} before a removal (except i-cell σ_i) are on the boundary of $(i+1)$-cell σ'_{i+1} after the removal. An example of the effect of a removal $rem(\gamma, \beta, \gamma')$ on a 3D descending Morse complex is illustrated in Figure 2 (a).

After the removal, 1-cell β is deleted, and 2-cell γ is merged with the unique 2-cell γ' incident in β and different from γ.

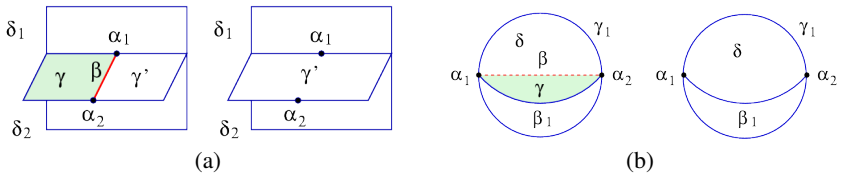


Fig. 2. Part of a 3D descending Morse complex before and after a removal (a) $rem(\gamma,\beta,\gamma')$, and (b) $rem(\gamma,\beta,\emptyset)$ (color plate C. 11, page 257).

After a removal $rem(\sigma_{i+1},\sigma_i,\emptyset)$, i-cell σ_i and $(i+1)$-cell σ_{i+1} are deleted in Γ_d. An example of the effect of a removal $rem(\gamma,\beta,\emptyset)$ on a 3D descending Morse complex is illustrated in Figure 2 (b). 1-cell β is incident to exactly one 2-cell γ. 2-cell γ is bounded by 1-cells β and β_1. 3-cell δ is the only 3-cell in the co-boundary of γ. After the removal, cells β and γ are deleted, and the boundary and the co-boundary of all other cells remain unchanged.

Dually, a *contraction* (of index $i+1$) of an $(i+1)$-saddle σ_{i+1} and an i-saddle σ_i is defined if σ_{i+1} is connected by a unique integral line to

- i-saddle σ_i, and exactly one i-saddle σ_i' different from σ_i, or
- exactly one i-saddle σ_i.

In the first case, a contraction of σ_i and σ_{i+1} is denoted as $con(\sigma_i,\sigma_{i+1},\sigma_i')$, and in the second case as $con(\sigma_i,\sigma_{i+1},\emptyset)$.

The effect of a contraction of index i on Γ_d (Γ_a) is the same as the effect of a removal of index $n-i$ on Γ_a (Γ_d). After a contraction $con(\sigma_i,\sigma_{i+1},\sigma_i')$, $(i+1)$-cell σ_{i+1} in Γ_d is deleted, i-cell σ_i is merged into i-cell σ_i' and all cells which were in the co-boundary of i-cell σ_i before a contraction (except $(i+1)$-cell σ_{i+1}) are in the co-boundary of i-cell σ_i' after the contraction (i-cell σ_i is deleted, and each $(i+1)$-cell in the co-boundary of σ_i is extended to include a copy of $(i+1)$-cell σ_{i+1}). An example of the effect of a contraction $con(\beta,\gamma,\beta')$ on a 3D descending Morse complex is illustrated in Figure 3 (a). The two 1-cells β and β' are merged, and 2-cell γ is deleted.

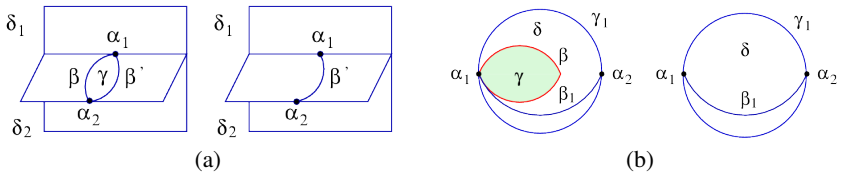


Fig. 3. Part of a 3D descending Morse complex before and after a contraction $con(\beta,\gamma,\beta')$ (a), and $con(\beta,\gamma,\emptyset)$ (b) (color plate C. 12, page 257).

After a contraction $con(\sigma_i, \sigma_{i+1}, \emptyset)$, $(i+1)$-cell σ_{i+1} and i-cell σ_i are deleted in Γ_d. An example of the effect of a contraction $con(\beta, \gamma, \emptyset)$ on a 3D descending Morse complex is illustrated in Figure 3 (b). 1-cell β is the only 1-cell on the boundary of 2-cell γ. The only 3-cell in the co-boundary of 2-cell γ is δ. After the contraction, 1-cell β and 2-cell γ are deleted.

It is possible to give a unifying definition of the two operators as a special case of a cancellation when one of the two canceled critical points satisfies certain valence conditions in the incidence graph. We define the two operators separately, due to the different (dual) geometric effect they have on the Morse complexes.

When $i = 0$ (and dually when $i = n$), a removal $rem(\sigma_1, \sigma_0, \sigma_1')$, which is defined when a 0-cell σ_0 is incident once to exactly two different 1-cells σ_1 and σ_1' in Γ_d, has the same geometric effect as a contraction $con(\sigma_0, \sigma_1, \sigma_0')$, where σ_0' is the other 0-cell on the boundary of 1-cell σ_1. For example, removal $rem(\beta, \alpha, \beta')$ illustrated in Figure 4 (a) can also be viewed as contraction $con(\alpha, \beta, \alpha')$. Note that the result of a removal or contraction may be a complex which does not satisfy the condition stated in Section 2, as illustrated in Figure 4 (b).

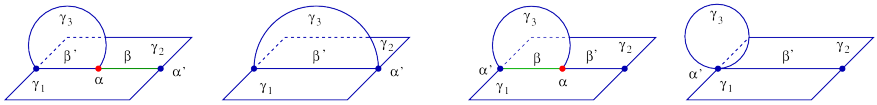


Fig. 4. Part of a 3D descending Morse complex before and after a removal $rem(\beta, \alpha, \beta')$ (or a contraction $con(\alpha, \beta, \alpha')$). γ_3 is a bubble-like 2-cell bounded by one 1-cell. (a) Operator is allowed. (b) Operator is not allowed.

Both removal and contraction can be interpreted as Euler operators, since they cancel a pair of cells of consecutive dimension in the two Morse complexes. Thus, Euler formula is satisfied after each simplification. The two operators are instances of the same Euler operator Kill_i-Cell_and_$(i+1)$-Cell in the descending complex, and to Kill_$(n-i)$-Cell_and_$(n-(i+1))$-Cell in the ascending complex.

We can show that the two operators form a basis of the set of operators for updating Morse complexes using the approach in [19]. Each operator c which simplifies Morse complexes in a topologically consistent manner maintains the Euler formula, and thus can be expressed as a vector $c = (c_0, c_1, .., c_n)$ with positive integer coordinates c_i, $0 \leq i \leq n$, in an n-dimensional discrete subspace V of an $(n+1)$-dimensional discrete space, defined by $c_0 - c_1 + .. + (-1)^n c_n = 0$. The ith coordinate c_i corresponds to the number of i-cells removed by operator c. A removal (and a contraction) b_i of an i-cell p and an $(i+1)$-cell q, $0 \leq i \leq n-1$, can be expressed as a vector $b_i = (a_0, a_1, .., a_n)$ in V, where $a_i = a_{i+1} = 1$, and $a_j = 0$, $j \neq i, i+1$. The set of all such vectors obviously forms a basis of subspace V. The above argument does not provide an algorithm for expressing an arbitrary operator c as a sequence of basis operators. As pointed out in [19], if $c = k_0 b_0 + k_1 b_1 + .. + k_{n-1} b_{n-1}$ then c may be expressed through $K_i \geq k_i$ simplifications b_i and $K_i - k_i$ refinements inverse to b_i. Moreover, the entities (cells in

Morse complexes, or critical points of f) which are introduced by inverse operations are not restricted to belong to a fixed set of entities of the initial full-resolution model.

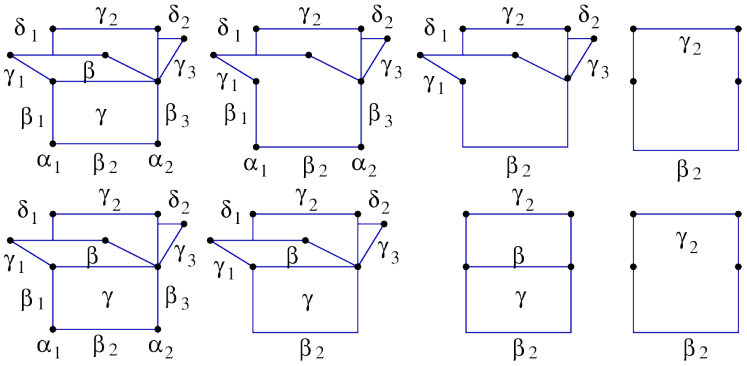


Fig. 5. A sequence of cancellations (top), and of removals and contractions (bottom) on a 3D descending Morse complex, which produce the same simplified Morse complex.

We illustrate the above argument on a simple example. After a cancellation of a 1-cell β and a 2-cell γ in a 3D descending Morse complex Γ_d shown in Figure 5 (top), each 1-cell β_i (and each 0-cell α_j) in Γ_d on the boundary of γ becomes incident to each 2-cell γ_k (and to each 3-cell δ_l) in the co-boundary of β. Each such pair of incident cells (e.g. δ_1 and α_1) induces a new cell in the Morse-Smale complex, which is eliminated by subsequent cancellations of 1-cells and 0-cells (e.g. α_1 and β_1), and of 2-cells and 3-cells (e.g. δ_1 and γ_1) [15]. Such a sequence of cancellations can be expressed as a (simpler) sequence of contractions (of 1-cells and 0-cells) and removals (of 2-cells and 3-cells), which do not have a side-effect of introducing new cells in the Morse-Smale complex, as illustrated in Figure 5 (bottom).

6 Removal and Contraction on the Incidence Graph

In this Section, we illustrate the effect of removals and contractions on the incidence graph and on the corresponding incidence-based representation.

Before a removal $rem(\sigma_{i+1},\sigma_i,\sigma'_{i+1})$, node σ_i in the incidence graph $G=(N,A)$ is connected through an arc in A to exactly two different nodes σ_{i+1} and σ'_{i+1} at level $i+1$ (the label of those arcs is 1), and to an arbitrary number of nodes at level $i-1$. The number of arcs incident to nodes σ_{i+1} or σ'_{i+1} may be arbitrary. For example, before a removal $rem(\gamma,\beta,\gamma')$ illustrated in Figure 6, node β at level 1 is connected to exactly two different nodes γ and γ' at level 2.

In terms of the incidence graph $G=(N,A)$, a removal $rem(\sigma_{i+1},\sigma_i,\sigma'_{i+1})$ can be expressed as the deletion of the arcs connecting σ_i to lower-level nodes and of arcs connecting σ_{i+1} to higher-level nodes, and merging of nodes σ_i and σ_{i+1} into node σ'_{i+1} by contraction of arcs connecting σ_i to σ_{i+1} and to σ'_{i+1}. For each arc connecting

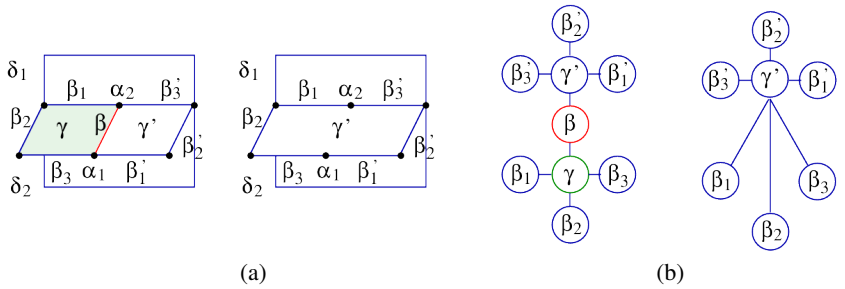


Fig. 6. Effect of a removal $rem(\gamma,\beta,\gamma')$ on a 3D descending Morse complex (a), and on a part of the corresponding incidence graph (b).

$(i+1)$-level node σ_{i+1} to an i-level node $\tau \neq \sigma_i$, such that σ'_{i+1} and τ are connected through an arc in A, the label of the arc connecting σ'_{i+1} and τ is increased by the label of the arc connecting σ_{i+1} and τ.

In Figure 6, after a removal $rem(\gamma,\beta,\gamma')$, arcs connecting node β to 0-level nodes α_1 and α_2 (not illustrated in the Figure) and arcs connecting γ to 3-level nodes δ_1 and δ_2 (not illustrated in the Figure) are deleted. Nodes β and γ are merged into node γ', by contracting arcs connecting β to γ and γ', i.e., arcs connecting γ to β_1, β_2 and β_3 are replaced by arcs connecting γ' to β_1, β_2 and β_3.

In the example shown in Figure 7, before a removal $rem(\gamma,\beta,\gamma')$, 1-cell β_1 is incident to 2-cells γ and γ' once. After the removal, β_1 is incident twice to γ'.

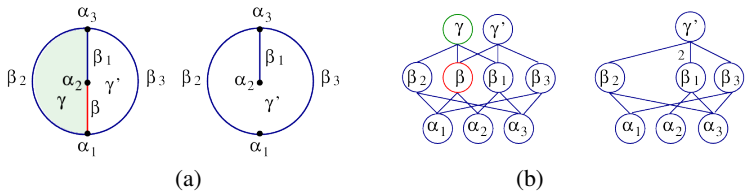


Fig. 7. Effect of a removal $rem(\gamma,\beta,\gamma')$ on a 2D descending Morse complex (a), and on a part of the incidence graph (b). The label of the arc connecting nodes β_1 and γ' is increased after the removal.

Before a removal $rem(\sigma_{i+1},\sigma_i,\emptyset)$ of the second kind, node σ_i at level i in the incidence graph is connected through an arc in A to exactly one node σ_{i+1} at level $i+1$. After the removal, nodes σ_i and σ_{i+1} together with all arcs incident in them are deleted, as illustrated in Figure 8.

A contraction $con(\sigma_i,\sigma_{i+1},\sigma'_i)$ is expressed in terms of the graph in a completely dual fashion. Its description is omitted here for brevity.

The effect on the incidence-based representation, that is on the combination of the incidence graph with the underlying simplicial decomposition of the domain, is restricted to the incidence graph when a simplification operator does not involve an ex-

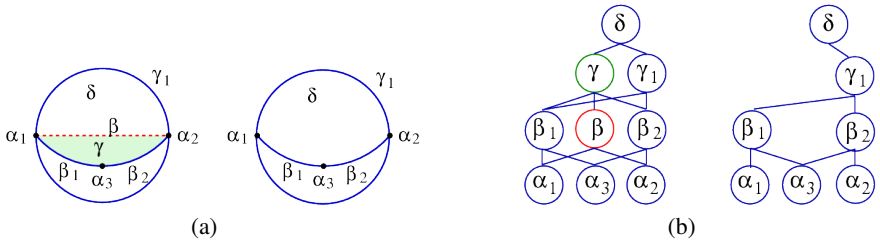


Fig. 8. Effect of a removal $rem(\gamma,\beta,\emptyset)$ on a 3D descending Morse complex (a), and on a part of the corresponding incidence graph (b).

tremum. When we perform a removal $rem(\sigma_n,\sigma_{n-1},\sigma'_n)$, then the partition of the n-simplexes of the underlying mesh into descending cells of maxima is updated by merging the set of n-simplexes forming the descending cell of σ_n into set of n-simplexes forming the descending cell of σ'_n. Dually, a contraction $con(\sigma_0,\sigma_1,\sigma'_0)$ merges n-simplexes of the ascending cell of σ_0 with n-simplexes of the ascending cell of σ'_0.

7 Concluding Remarks

We have defined removal and contraction operators for simplifying n-dimensional Morse complexes. We have expressed them as Euler operators, thus providing a minimal set of atomic operators for updating Morse complexes. We have shown the effect of such simplification operators on the incidence graph representation of the two Morse complexes.

We are currently working on the definition of a multi-scale representation for Morse complexes, which will provide a combinatorial description of the ascending and descending Morse complexes at different levels of abstraction. This will be obtained by defining the inverse operators of removal and contraction operators, a dependency relation between such operators, and by using an incidence graph representation for the complexes. We are also planning to implement the simplification operators and the multi-scale model for 3D scalar fields. Our future work would be in the direction of combining a multi-scale morphological representation with a mesh-based multi-resolution representation of the field.

Acknowledgements

This work has been partially supported by the National Science Foundation under grant CCF-0541032, by the MIUR-FIRB project SHALOM under contract number RBIN04HWR8, and by the Ministry of Science of the Republic of Serbia through Project 23036.

References

1. C. L. Bajaj and D. R. Shikore. Topology Preserving Data Simplification with Error Bounds. *Computers and Graphics*, 22(1):3–12, 1998.

2. T. Banchoff. Critical Points and Curvature for Embedded Polyhedral Surfaces. *American Mathematical Monthly*, 77(5):475–485, 1970.
3. P. Bhaniramka, R. Wenger, and R. Crawfis. Isosurfacing in Higher Dimensions. In *Proceedings IEEE Visualization 2000*, pages 267–273. IEEE Computer Society, Oct. 2000.
4. S. Biasotti, L. De Floriani, B. Falcidieno, and L. Papaleo. Morphological Representations of Scalar Fields. In L. De Floriani and M. Spagnuolo, editors, *Shape Analysis and Structuring*, pages 185–213. Springer Verlag, 2008.
5. P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A Topological Hierarchy for Functions on Triangulated Surfaces. *Transactions on Visualization and Computer Graphics*, 10(4):385–396, July/August 2004.
6. F. Cazals, F. Chazal, and T. Lewiner. Molecular Shape Analysis Based upon the Morse-Smale Complex and the Connolly Function. In *Proceedings of the nineteenth Annual Symposium on Computational Geometry*, pages 351–360, New York, USA, 2003. ACM Press.
7. L. Čomić and L. De Floriani. Cancellation of Critical Points in 2D and 3D Morse and Morse-Smale Complexes. In *Discrete Geometry for Computer Imagery (DGCI), Lecture Notes in Computer Science*, volume 4992, pages 117–128, Lyon, France, Apr 16-18 2008. Springer-Verlag GmbH.
8. E. Danovaro, L. De Floriani, and M. M. Mesmoudi. Topological Analysis and Characterization of Discrete Scalar Fields. In T.Asano, R.Klette, and C.Ronse, editors, *Theoretical Foundations of Computer Vision, Geometry, Morphology, and Computational Imaging*, volume LNCS 2616, pages 386–402. Springer Verlag, 2003.
9. L. De Floriani and A. Hui. Shape Representations Based on Cell and Simplicial Complexes. In *Eurographics 2007, State-of-the-art Report*. September 2007.
10. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer Verlag, Berlin, 1987.
11. H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale Complexes for Piecewise Linear 3-Manifolds. In *Proceedings 19th ACM Symposium on Computational Geometry*, pages 361–370, 2003.
12. H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse Complexes for Piecewise Linear 2-Manifolds. In *Proceedings 17th ACM Symposium on Computational Geometry*, pages 70–79, 2001.
13. R. Forman. Morse Theory for Cell Complexes. *Advances in Mathematics*, 134:90–145, 1998.
14. T. Gerstner and R. Pajarola. Topology Preserving and Controlled Topology Simplifying Multi-Resolution Isosurface Extraction. In *Proceedings IEEE Visualization 2000*, pages 259–266, 2000.
15. A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. Topology-Based Simplification for Feature Extraction from 3D Scalar Fields. In *Proceedings IEEE Visualization'05*, pages 275–280. ACM Press, 2005.
16. A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. A Topological Approach to Simplification of Three-Dimensional Scalar Functions. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):474–484, 2006.
17. T. Illetschko, A. Ion, Y. Haxhimusa, and W. G. Kropatsch. Collapsing 3D Combinatorial Maps. In F. Lenzen, O. Scherzer, and M. Vincze, editors, *Proceedings of the 30th OEAGM Workshop*, pages 85–93, Obergurgl, Austria, 2006. sterreichische Computer Gesellschaft.
18. P. Magillo, E. Danovaro, L. De Floriani, L. Papaleo, and M. Vitali. Extracting Terrain Morphology: A New Algorithm and a Comparative Evaluation. In *2nd International Conference on Computer Graphics Theory and Applications*, pages 13–20, March 8–11 2007.
19. M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, 1988.
20. J. Milnor. *Morse Theory*. Princeton University Press, New Jersey, 1963.

21. X. Ni, M. Garland, and J. C. Hart. Fair Morse Functions for Extracting the Topological Structure of a Surface Mesh. In *International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH*, pages 613–622, 2004.
22. V. Pascucci. Topology Diagrams of Scalar Fields in Scientific Visualization. In S. Rana, editor, *Topological Data Structures for Surfaces*, pages 121–129. John Wiley & Sons Ltd, 2004.
23. S. Takahashi, T. Ikeda, T. L. Kunii, and M. Ueda. Algorithms for Extracting Correct Critical Points and Constructing Topological Graphs from Discrete Geographic Elevation Data. In *Computer Graphics Forum*, volume 14, pages 181–192, 1995.
24. S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological Volume Skeletonization and its Application to Transfer Function Design. *Graphical Models*, 66(1):24–49, 2004.
25. G. H. Weber, G. Schueuermann, H. Hagen, and B. Hamann. Exploring Scalar Fields Using Critical Isovalues. In *Proceedings IEEE Visualization 2002*, pages 171–178. IEEE Computer Society, 2002.
26. G. H. Weber, G. Schueuermann, and B. Hamann. Detecting Critical Regions in Scalar Fields. In G.-P. Bonneau, S. Hahmann, and C. D. Hansen, editors, *Proceedings Data Visualization Symposium*, pages 85–94. ACM Press, New York, 2003.
27. C. Weigle and D.Banks. Extracting Iso-Valued Features in 4-dimensional Scalar Fields. In *Proceedings IEEE Visualization 1998*, pages 103–110. IEEE Computer Society, Oct. 1998.
28. G. W. Wolf. Topographic Surfaces and Surface Networks. In S. Rana, editor, *Topological Data Structures for Surfaces*, pages 15–29. John Wiley & Sons Ltd, 2004.

Geometric Topology & Visualizing 1-Manifolds

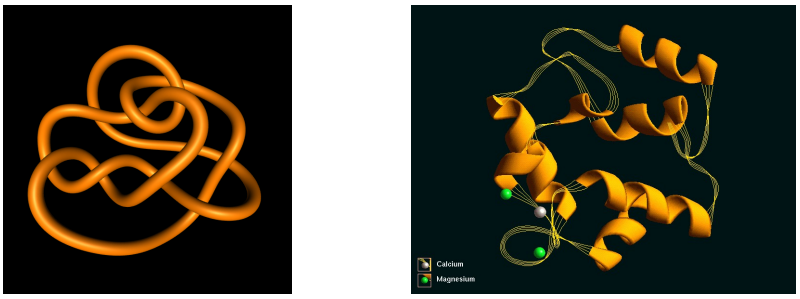
Kirk E. Jordan[1], Lance E. Miller[2], Thomas J. Peters[3], and Alexander C. Russell[4]

[1] IBM Corporation, One Rogers St., Cambridge, MA 02142 kjordan@us.ibm.com
[2] University of Connecticut, Storrs, CT, 06269-2155 lance.e.miller@gmail.com
[3] University of Connecticut, Storrs, CT, 06269-2155 & Kerner Graphics, Inc., San Rafael, CA, 94912 tpeters@cse.uconn.edu
[4] University of Connecticut, Storrs, CT, 06269-2155 acr@cse.uconn.edu

Abstract: Ambient isotopic approximations are fundamental for correct representation of the embedding of geometric objects in $\mathbf{R}^3$, with a detailed geometric construction given here. Using that geometry, an algorithm is presented for efficient update of these isotopic approximations for dynamic visualization with a molecular simulation.

1 Approximation and Topology for Visualization

Figure 1(a) depicts a knot[5] and Figure 1(b) shows a visually similar protein model[6]. prompting two criteria for efficient algorithms for visualization:



(a) (b)

Fig. 1. (a) Complicated Unknot, (b) Protein-enzyme complex Color plate C. 13, page 258.

- a piecewise linear (PL) approximation that preserves model topology,
- preservation of topology during dynamic changes, such as protein unfolding.

The visual comparison from Figure 1 led to invoking knot theory to provide the unifying mathematics. This paper presents a curvature-adaptive, topology preserving approximation for a parametric 1-manifold with the primary result being Theorem 3. The piecewise linear (PL) approximations presented will

[5] Image credit: R. Scharein, www.knotplot.com
[6] Image credit: http://160.114.99.91/astrojan/protein/pictures/parvalb.jpg

(i) be topologically equivalent to the original manifold,
(ii) minimize the number of linear approximants,
(iii) respect user-specified error bounds for distance & curvature.

While many approximation methods fulfill criteria (ii) and (iii), the stipulation of criterion (i) is of recent interest and the methods here are for a rich class of curves, extending related results. The topological equivalence chosen for dynamic molecular visualizations is by ambient isotopy, as this preserves the embedding of the geometric model over time.

Definition 1. *Let X and Y be two subsets of n-dimensional space $\mathbf{R}^n$. An* **ambient isotopy** *is a continuous function $H : \mathbf{R}^n \times [0,1] \to \mathbf{R}^n$ such that*

1. *$H(\cdot,0)$ is the identity function,*
2. *$H(X,1) = Y$, and*
3. *For each t in $[0,1]$, $H(\cdot,t)$ is a homeomorphism on $\mathbf{R}^n$.*

2 Related Work

Preliminary work by some of these authors and collaborators has appeared: for the integration of time and topology in animations and simulations [12] and for isotopic approximations on various classes of spline curves [14–17]. The theory presented here extends to a broad class of parametric curves that properly includes splines. While one approximation method could be applied to general parametric curves [12] the isotopy results within that paper relied upon Bézier geometry.

The proof techniques for isotopy here are a slight variant of the well-known 'push' from geometric topology for a 3-manifold [2]. The importance for applications is the creation of explicit neighborhoods within which the approximant can be perturbed while remaining ambient isotopic. These can serve as the basis for determining efficient updates for these isotopic constraints during dynamic visualizations.

A previous application of these tubular neighborhoods has emphasized visualizing knots undergoing dynamic changes [4]. Another tubular neighborhood algorithm [14] for rational spline curves [19] relies upon specialized numerical solution software, whereas computation with Newton's method has been exhibited on Bézier curves [15,16]. Related treatments to curve approximations are available: within Hilbert spaces [10], with restrictions to planar curves [8] or specialized to spline curves [18]. The approach of approximating with respect to curvature is similar to an approach for 'aesthetic engineering' [20].

This terse summary on curve approximation stresses only the most relevant literature, as any comprehensive survey would be voluminous, with one indication being that a literature search on *curve approximation* resulted in 1.4 million hits[7]. The distinguishing feature, here, is the additional insistence of topological equivalence. This emphasis upon geometric topology is appropriate when a geometric model is present, as for the molecular models discussed, and could prove complementary to other uses of topology in visualization that depend largely upon algebraic topology [6,7].

[7] http://scholar.google.com/

3 Curvature & Topology for Parametric 1-Manifolds

Each curve considered here is assumed to be a compact 1-manifold, thereby excluding self-intersections, where stated differentiability assumptions also preclude wild arcs[8]. Further, each 1-manifold is assumed to have total arc length 1 and is parameterized over the unit[9] interval.

Notation: Let $c : [0,1] \to \mathbf{R}^3$ be a C^3 curve and we denote

$$\mu_c([a,b]) = \int_a^b ||c''(t)||dt,$$

which will be invoked as the basis for our curvature-adaptive approximation.

Theorem 1. *Let $c : [0,1] \to \mathbf{R}^3$ be a C^3 curve. For each $\varepsilon > 0$, there exists a natural number n and a partition $X = \{p_1, \ldots, p_n\} \subset [0,1]$ such that, for $i = 1, \ldots, n$, $p_1 = 0, p_n = 1$; $p_1 < p_2 \ldots < p_{n-1} < p_n$; and*

$$\int_{p_i}^{p_{i+1}} \|c''(z)\|dz = \mu_c([p_i, p_{i+1}]) < \varepsilon, \tag{1}$$

there is a set of compact cylinders $\{C_i\}_{i=1}^n$, such that each C_i has its axis aligned with the tangent at $c(p_i)$ and has radius ε. Furthermore both the polyline formed by consecutively connecting the vertices $\{c(p_i) : i = 1, \ldots, n\}$ and the curve c lie in $\bigcup_{i=1}^n C_i$.

Theorm 1 can be used to create a PL approximation of a curve, but there are no guarantees given that the approximant is topologically equivalent to the original curve. Further constraints must be imposed upon the choice of ε to ensure that the approximant produced is ambient isotopic to the original curve, as will be developed in the rest of this paper. The proof of Theorem 1 follows.

Proof. We construct one cylinder C_i for each point in the set X. Let L_i be the line containing the vector $c'(p_i)$. Consider the plane normal to $c'(p_{i+1})$. This plane intersects L_i at a point, denoted as q. Define C_i to be the cylinder of radius ε whose axis is the portion of L_i connecting p_i and q. By Taylor's Theorem if $t \in [p_i, p_{i+1}]$ then

$$|c(t) - c(p_i) - (t - p_i)c'(p_i)| \leq \int_{p_i}^{t} ||c''(z)||(t-z)dz \leq \int_{p_i}^{p_{i+1}} ||c''(z)||dz < \varepsilon.$$

Hence, $c(t)$ is of distance at most ε from $c(p_i) + (t - p_i)c'(p_i)$. Thus $c(t)$ is at most ε away from L_i, and so $c(t)$ is in the cylinder C_i. The last statement is clear as each cylinder is convex and the endpoints $c(p_i), c(p_{i+1})$ of an approximating segment are contained in the cylinder C_i.

[8] Any C^2 compact 1-manifold is tame [14].

[9] If not of unit length, scale accordingly.

If, for each i, Inequality 1 is modified so that $\mu_c([p_i, p_{i+1}]) = \varepsilon$, then a previously published proof [12] can be applied for an asymptotic limit on the number of segments in the approximation. For $\varepsilon > 0$, denote by $N(\varepsilon)$ the number of cylinders given by the construction in the proof of Theorem 1.

Corollary 1. *If c is also C^3 such that $||c''(t)|| > 0$ for all t, then*

$$\lim_{\varepsilon \to 0} \sqrt{\varepsilon} \cdot N(\varepsilon) = \int_0^1 \sqrt{||c''(u)||} du.$$

We identify important properties studied in geometric topology and prove maintenance of those characteristics for the approximant. For the remainder of the section we will refer to the set $X = \{p_1, \ldots, p_n\}$ where the p_i's are ordered as described, above, and the cylinders C_i are constructed as in Theorem 1. We ensure that the curve is well-behaved within each cylinder through the following lemmas.

Lemma 1. *Let $c : [0,1] \to \mathbf{R}^3$ be a C^3 curve and $r, s \in [p_i, p_{i+1}]$, then*

$$||c'(r) - c'(s)|| \leq \sqrt{3}\varepsilon.$$

Proof. For $t \in [0,1]$ denote by $c'(t)_x$ the x-coordinate for $c'(t)$. We can apply Taylor's Theorem to see for $r, s \in [p_i, p_{i+1}]$,

$$|c'(r)_x - c'(s)_x| \leq \int_r^s |c''(u)_x| du \leq \int_r^s ||c''(u)|| du \leq \varepsilon.$$

The case is similar for the y and z coordinates, and the result follows.

Corollary 2. *Let $c : [0,1] \to \mathbf{R}^3$ be a C^3 space curve, and $\varepsilon < \sqrt{2/3}$. Then for points $r, s \in [p_i, p_{i+1}]$ the tangential deviation between $c'(r)$ and $c'(s)$ is no more than $\pi/2$.*

Proof. Since we are parameterized using arc length, we have $||c'(r)|| = ||c'(s)|| = 1$ for any $r, s \in [p_i, p_{i+1}]$. Further, by Lemma 1,

$$||c'(r) - c'(s)|| < \sqrt{3}\varepsilon.$$

The Law of Cosines shows the angle between them is $\arccos(1 - \frac{3}{2}\varepsilon^2)$. When $\varepsilon < \sqrt{\frac{2}{3}}$, we have $\arccos(1 - \frac{3}{2}\varepsilon^2) < \frac{\pi}{2}$.

So we can explicitly control the tangential deviation within each cylinder C_i. The benefit here is that we can use this information to control intersections within each cylinder.

Lemma 2. *Let $c : [0,1] \to \mathbf{R}^3$ be a C^3 curve and $[a,b] \subset [0,1]$. Assume all of the tangent vectors of the subcurve $c([a,b])$ deviate in angle by no more than $\eta < \frac{\pi}{2}$ from a particular tangent vector $c'(t)$ for $t \in [a,b]$. If Π is a plane with normal $c'(t)$ and $\Pi \cap c([a,b]) \neq \emptyset$, then $|c([a,b]) \cap \Pi| = 1$.*

Proof. Orient the plane Π so that it is parallel with the xy-plane and $c'(t)$ with the positive z-axis. In this orientation if $c'(r)_z = 0$ for any $r \in [a,b]$ then $c'(r)$ is parallel to the plane Π and this would be a contradiction as $c'(t)$ is normal to the plane Π and the angular deviation of tangent vectors is not more than $\frac{\pi}{2}$. For any $s \in [a,b]$ with $s \neq t$, if $c(s)$ lies on the plane Π then $c(s)_z = 0$ and since $c(t)_z = 0$ we have, by the Mean Value Theorem, another point r with $c'(r)_z = 0$, which is a contradiction.

Notation: For the remainder of this paper, let

- $c : [0,1] \to \mathbf{R}^3$ denote a C^3 curve,
- $\kappa = \max ||c''(z)||$, $\alpha = \min\{1, ||c''(z)||\}$,
- X be a set which satisfies the hypothesis of Theorem 1, and
- For $p_i, p_{i+1} \in X$, let $c_i = c([p_i, p_{i+1}])$.

Our next step is to construct sets upon which we can build local isotopies between the curve and the approximant. In order to do this we will make use of Taylor's theorem, as in the following lemma.

Lemma 3. *Let γ be chosen such that $\gamma > 0$, but $\gamma \ll (\kappa/2)(t-p_i)^2$. For $t \in (p_i, p_{i+1}]$, let S_t be the closed ball with center $c(p_i) + (t-p_i)c'(p_i)$ and radius r_t, with $r_t = (\kappa/2)(t-p_i)^2 + \gamma$. Then, $c(t) \in \text{int}(S_t)$.*

Proof. By Taylor's theorem we have

$$\begin{aligned} ||c(t) - c(p_i) - (t-p_i)c'(p_i)|| &\leq \int_{p_i}^{t} ||c''(z)||(t-z)dz \\ &\leq \kappa[t^2 - (1/2)t^2 - (tp_i - (1/2)p_i^2)]. \end{aligned}$$

However $t^2 - (1/2)t^2 - (tp_i - (1/2)p_i^2) = (1/2)(t-p_i)^2$. The use of γ permits the conclusion about containment in the interior.

For each $t \in (p_i, p_{i+1}]$, define a 'snowcone' K_t as the convex hull[10] of S_t and $\{p_i\}$, where the use of the colloquial term 'snowcone' is meant to suggest the compact set created, with a planar cross-section shown in Figure 2. The next lemma shows the relationship between the opening angle of these snowcones[11] for different values t.

Lemma 4. *Choose $\varepsilon \in (0, \alpha/(2\kappa))$. Let $t \in (p_i, p_{i+1}]$, let θ_t be opening angle at $c(p_i)$ for K_t. Then θ_t is an increasing function of t and each snowcone K_t, for $t \in (p_i, p_{i+1}]$, is contained in the snowcone $K_{p_{i+1}}$.*

[10] For any two sets A and B, their convex hull is defined as the smallest convex set that contains $A \cup B$.

[11] In the United States, a frozen desert having a two dimensional profile similar to that of Figure 2 is known as a 'snowcone'.

Proof. Denote by $r_t = (\kappa/2)(t-p_i)^2$ the radius of S_t. Since

$$\int_{p_i}^{p_{i+1}} ||c''(z)||dz < \varepsilon$$

we have that $\alpha(p_{i+1}-p_i) < \varepsilon$ and since $\varepsilon < (2\alpha/\kappa)$ we have that

$$(t-p_i) \leq (p_{i+1}-p_i) \leq \varepsilon/\alpha \leq 2/\kappa.$$

Thus, $(\kappa/2)(t-p_i) \leq 1$ and $(\kappa/2)(t-p_i)^2 \leq (t-p_i)$. Therefore, $c(p_i)$ is not contained in the sphere S_t. Consider a planar cross section of the snowcone K_t and within this planar cross section, choose a tangent to S_t and denote the point of tangency as v. Denote the angle between $c'(p_i)$ and the segment $[c(p_i), v]$ by θ_t. Denote by z the center of S_t. Using the triangle defined by $z, v, c(p_i)$ one can conclude that $\sin\theta_t = r_t/(t-p_i) = (\kappa/2)(t-p_i)$. For reference, consider Figure 2. Since r_t is finite, $\theta_t < \pi/2$. For $t, s \in [p_i, p_{i+1}]$ with $t < s$, $\sin\theta_t < \sin\theta_s$ and so $\theta_t < \theta_s$, as $\theta_s < \pi/2$.

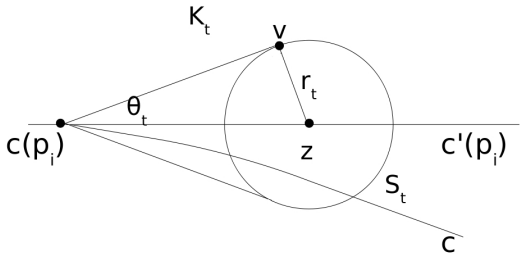


Fig. 2. Cross section of snowcone K_t

Hence, for $t \in (p_i, p_{i+1}]$, $K_t \subset K_{p_{i+1}}$.

Corollary 3. *The snowcone $K_{p_{i+1}}$ contains c_i.*

Proof. First note that $c(p_i) \in K_{p_{i+1}}$ as it is the apex of this snowcone. For each $t \in (p_i, p_{i+1}], c(t) \in K_{p_{i+1}}$ by Lemmas 3 and 4.

Notation: First, modify the existing notation $K_{p_{i+1}}$ to be $K_{i,i+1}$, using the order of these subscripts to express that the snowcone's apex is at p_i and opens towards p_{i+1}. This is to contrast it with a similar snowcone, denoted as $K_{i+1,i}$, which will have its apex as the point $c(p_{i+1})$, its axis as the line containing $c'(p_{i+1})$ *but* will open towards p_i, as indicated in Figure 3.

Lemma 5. *For each $i = 1, \ldots, n-1$, denote the approximating segment connecting the points $c(p_i)$ and $c(p_{i+1})$ by $a_{i,i+1}$. For each cylinder C_i, there is a convex subset of C_i, denoted $V_{i,i+1}$, containing c_i, such that*

$$V_{i,i+1} \cap V_{i+1,i+2} = c(p_{i+1}).$$

Proof. Define $V_{i,i+1} = K_{i,i+1} \cap C_i \cap K_{i+1,i}$. Since each of the intersecting sets is convex, it is clear that $V_{i,i+1}$ is convex. But then, since both $c(p_i)$ and $c(p_{i+1})$ are in $V_{i,i+1}$, it is clear that $a_i \subset V_{i,i+1}$. From Corollary 3, $c_i \subset V_{i,i+1}$.

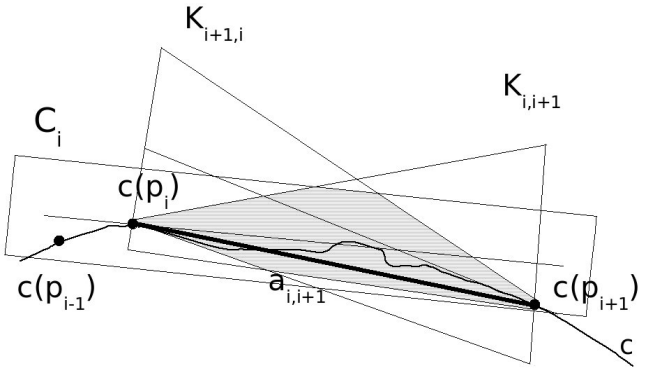


Fig. 3. Cross Section for bounding volume $V_{i,i+1}$

4 Building the Ambient Isotopy

The results so far only provide local views on c by focusing on each c_i independently. It is often easy to build an isotopy locally on one portion of a curve, but considerable subtlety can be required to unify these into an isotopy of the entire curve. The snowcones were defined for use in an iterative algorithm to establish an ambient isotopy for all of c.

Outline of Algorithm for Entire Curve: The geometric objective for the algorithm is to continue to reduce the radius of each cylinder C_i used in defining the sets $V_{i,i+1}$ until the interiors of these $V_{i,i+1}$ are pairwise disjoint, that is, for $i \neq j$,

$$int(V_{i,i+1}) \cap int(V_{j,j+1}) = \emptyset.$$

The algorithm begins with a seed value for δ to construct a set X_1 such that for each $i = 1, \ldots, n$, and $p_i, p_{i+1} \in X_1$, we have $\mu_c([p_i, p_{i+1}]) < \delta$. The algorithm proceeds by replacing the value of δ by $\delta/2$, so that at each successive iteration j, the set X_j has

the above mentioned containment properties relative to the current value of δ. The algorithm proceeds until these bounding volumes are pairwise disjoint in order to form an isotopy of the entire curve. A previously published termination proof [15] can be easily modified for this snowcone geometry. Indeed, the more aggressive containment snowcone geometry used here actually simplifies the previous proof, so that both algorithms terminate in $O(\log \Delta^{-1})$ iterations [15], where Δ is the *minimum separation distance* of the curve. For the curve c, define $d : [0,1] \to \mathbf{R}$ to be the distance function $d : (s,t) \mapsto \|c(s) - c(t)\|$. Then the minimum separation distance is the minimum critical value of d. A useful geometric formulation of this problem is to find all pairs of distinct points at parametric values s and t of c to satisfy the equations [13]:

$$(c(s) - c(t)) \cdot c'(s) = 0 \tag{2}$$

$$(c(s) - c(t)) \cdot c'(t) = 0. \tag{3}$$

Recent approaches to efficiently solve these simultaneous Equations 2 and 3 have appeared [15]. The value of Δ is then the minimum Euclidean distance over all pairs $(c(s), c(t))$ that are solutions to these simultaneous equations. The role of Δ as a stopping criterion can be intutitively expressed as measuring the minimum Euclidean distance between points of c that can be geodesically far apart. The algorithm restricts the size of the snowcones relative to Δ.

We first build a local homeomorphism on c_i and use that as a basis for constructing an ambient isotopy over all of c. For each point $w \in c_i$, let N_w denote the normal plane to c_i at w. Choose ε, in accordance with Corollary 2 to ensure that tangents on c_i deviate by less than $\pi/2$. Define the function

$$h_i : c_i \to a_{i,i+1}$$

by

$$h_i(w) \text{ is the single point in } N_w \cap a_{i,i+1}.$$

Theorem 2. *The function h_i is a homeomorphism that fixes both $c(p_i)$ and $c(p_{i+1})$.*

Proof. We consider the intersection of each N_w and $a_{i,i+1}$.

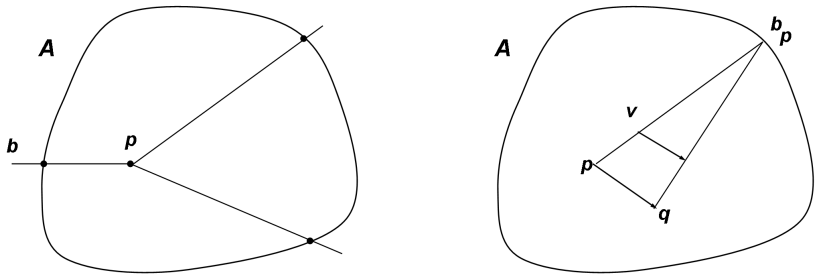
First, suppose that $a_{i,i+1}$ is a subset of N_w, requiring that both end points of $a_{i,i+1}$ were in N_w. But these end points are also points of c_i, which would be a contradiction to Lemma 2. If w is either endpoint of c_i, then it is also a point of $a_{i,i+1}$, but then Lemma 2 provides that this endpoint is the only element of the $N_w \cap a_{i,i+1}$.

So, suppose w is not an endpoint, then $c(p_i)$ cannot also be in N_w, by Lemma 2, and, similarly, $c(p_{i+1})$ cannot be in N_w. But, then, since $\pi/2$ is an upper bound on the tangents on c_i, it is clear that $c(p_i)$ and $c(p_{i+1})$ must be on opposite sides of N_w. So, $a_{i,i+1}$ and N_w cannot be disjoint. But, then, N_w and $a_{i,i+1}$ intersect in a single point, since a plane and a line can intersect only in the line itself or in a single point.

Then h_i is a well-defined function on each c_i, where continuity is obvious. Moreover, note that h_i keeps the end points of c_i fixed. That h_i is onto follows since the endpoints of $a_{i,i+1}$ remain fixed and the image of h_i is a connected subset of $a_{i,i+1}$. That h_i is 1-1 follows, since $a_{i,i+1}$ is within ε of c_i, with ε chosen to be less than $1/(2\kappa)$ (with κ chosen to be the maximum curvature) so, the line segments given by $[w_1, h(w_1)]$ and $[w_2, h(w_2)]$ do not intersect.

Now we are set to define a local isotopy within the bounding volume $V_{i,i+1}$. But first we need to recall some easily proven properties of convex sets of $\mathbf{R}^n$. Let A denote a non-empty, compact, convex subset of $\mathbf{R}^n$, for some positive integer n.

Lemma 6. *For each point $p \in \mathrm{int}(A)$ and $b \in \partial A$, the ray going from p to b only intersects ∂A at b (See Figure 4(a).)*



(a) (b)

Fig. 4. (a) Rays outward. (b) Variant of a push.

Lemma 7. *Let A be a compact convex subset of $\mathbf{R}^2$ with non-empty interior and fix $p \in int(A)$. For each boundary point $b \in \partial A$, denote by $[p,b]$ the line segment from p to b. Then $A = \bigcup_{b\in\partial A}[p,b]$.*

Many of the arguments of the proof of Theorem 2 can be adapted to build the ambient isotopy of c. The construction relies strongly on having a compact, convex set of support, as illustrated in Figure 4(b). The previous attention to convexity of $V_{i,i+1}$ was directed towards this construction.

Corollary 4. *There is an ambient isotopy of c_i, with compact support $V_{i,i+1}$ that takes each point of c_i to $h(c_i)$.*

Proof. A value of $\varepsilon > 0$ should be chosen so that $\varepsilon < \sqrt{2/3}$ to fulfill the conditions of Corollary 2 so that each h_i is a homeomorphism. Simultaneously choose, $\varepsilon < \alpha/(2\kappa)$ to satisfy the hypotheses of Lemma 4 while also constraining $\varepsilon \leq \delta$ to ensure that the interiors of the bounding volumes $V_{i,i+1}$ and $V_{j,j+1}$ are disjoint, with δ being the output of the iterative algorithm outlined at the beginning of this section. It should be clear that bounding volumes $V_{i,i+1}$ and $V_{i+1,i+2}$ intersect only at the point $c(p_{i+1})$.

The proof is a variant of a 'push' [2], where point p and q are in the interior of a non-empty, convex, planar set. As illustrated in Figure 4(b), each point v on a line segment between p and a boundary point b is mapped by linear interpolation onto the segment $[q,b]$, as p is mapped onto q, and then Lemmas 6 and 7 are applied to build the ambient isotopy from the homeomorphism of Theorem 2.

Theorem 3. *There is an ambient isotopy of c onto its PL approximant a and this isotopy has compact support* $\bigcup_i V_{i,i+1}$.

Proof. The snowcone construction leaves bounding volumes $V_{i,i+1}$ and $V_{i+1,i+2}$ intersecting only at the point p_{i+1}, which is fixed under the local isotopy, so a 'pasting lemma' [16] can be applied to complete the proof.

5 Conclusions and Future Work

The primary result of this work is a curvature-adpative, ambient isotopic approximation for a 1-manifold, inclusive of distance and tangency error bounds on the approximant. The bounding volumes also constrain many isotopic movements of the approximant curve.

The image of Figure 1(a) is the first frame of an animation showing that curve deforming under the application of energy described by Gaussian functionals, with guarantees that the embedded topology is preserved during this process. Similarly, the long-term focus here is to produce dynamic visualization of complex molecules undergoing simulated deformations under energy and chemical changes that also preserve the embedded topology. The essential first step is to produce a topology-preserving approximation of the static model. A specific approximated geometry model might be able to have multiple ambient isotopic perturbations within its bounding volume. However, if successive movements have the approximated geometry approaching the boundaries of the bounding volumes, new bounding volumes need to be computed. As long as the curve remains C^2 the existence of these new bounding volumes is known from classical differential topology [11].

However, those existence theorems typically do not provide explicit bounding geometry and they are not necessarily adaptive in the sense shown here. The value provided here is the detailed geometric constructions can be used to create algorithms that will allow for efficient updates of the geometric bounding neighborhoods as fundamental for dynamic visualization. The algorithm outlined in the second paragraph of Section 4 can be refined further, dependant upon the specific models being considered, for even more aggressive bounding volumes, when geometric alternatives to the snowcones used here might be useful on specific data.

In Figure 1(b), there was considerable empty space around the geometric model being unfolded, but Figure 5 depicts a very dense configuration[12].

The control for isotopies can become quite delicate for molecular models. In Figure 5, the close geometric proximity and associated small bounding volumes will require updates after even small movements. This model has been the subject of consideration for dynamic visualization of molecular simulation [12] and is the current test case for algorithmic performance. The geometry in Figure 5 is also multi-dimensional, but the geometric topology used has well-known generalizations beyond dimension one. Guidance for extension to higher dimensions may be gained by considering other papers on topological approximation of 2-manifolds [1, 3, 5, 9]. A particularly relevant

[12] Image credit: http://domino.research.ibm.com/comm/pr.nsf/pages/rscd.bluegene-picaa.html.

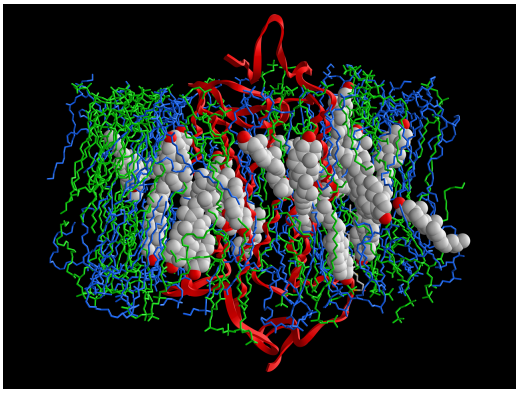


Fig. 5. Dense molecular configuration (color plate C. 14, page 258).

comparison is to recent results focusing on a sufficiently dense set of sample points [5] to give both error bounds and topological guarantees.

6 Acknowledgements

All but the first author gratefully acknowledge partial support from NSF grant CCF 0429477. L. E. Miller & T. J. Peters express appreciation for additional support from IBM Doctoral Fellowships and an IBM Faculty Award during 2005 - 2008. All statements here are the responsibility of the authors, not of these funding sources. We thank the anonymous reviewers for many helpful suggestions.

References

1. N. Amenta, T. J. Peters, and A. C. Russell. Computational topology: ambient isotopic approximation of 2-manifolds. *Theoretical Computer Science*, 305(1-3):3–15, 2003.
2. R. H. Bing. *The Geometric Topology of 3-Manifolds*. American Mathematical Society, Providence, RI, 1983.
3. Jean-Daniel Boissonnat, David Cohen-Steiner, and Gert Vegter. Isotopic implicit surface meshing. In László Babai, editor, *STOC*, pages 301–309. ACM, 2004.
4. Jason Cantarella, Michael Piatek, and Eric Rawdon. Visualizing the tightening of knots. In *VIS '05: Proceedings of the conference on Visualization '05*, pages 575–582, Washington, DC, USA, 2005. IEEE Computer Society.
5. Kenneth L. Clarkson. Building triangulations using ε-nets. In Jon M. Kleinberg, editor, *STOC*, pages 326–335. ACM, 2006.
6. David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.
7. David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. Persistent homology for kernels, images, and cokernels. In Claire Mathieu, editor, *SODA*, pages 1011–1020. SIAM, 2009.

8. Luiz Henrique de Figueiredo, Jorge Stolfi, and Luiz Velho. Approximating parametric curves with strip trees using affine arithmetic. *Comput. Graph. Forum*, 22(2):171–180, 2003.
9. T. K. Dey, H. Edelsbrunner, and S. Guha. Computational topology. In *Advances in Discrete and Computational Geometry (Contemporary Mathematics 223*, pages 109–143. American Mathematical Society, 1999.
10. Daniel Freedman. Combinatorial curve reconstruction in Hilbert spaces: A new sampling theory and an old result revisited. *Comput. Geom.*, 23(2):227–241, 2002.
11. M. W. Hirsch. *Differential Topology*. Springer-Verlag, New York, 1976.
12. K. E. Jordan, L. E. Miller, E. L. F. Moore, T. J. Peters, and A. C. Russell. Modeling time and topology for animation and visualization with examples on parametric geometry. *Theoretical Computer Science*, 405:41–49, 2008.
13. T. Maekawa and N. M. Patrikalakis. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer, New York, 2002.
14. T. Maekawa, N. M. Patrikalakis, T. Sakkalis, and G. Yu. Analysis and applications of pipe surfaces. *Computer Aided Geometric Design*, 15(5):437–458, 1998.
15. L. Miller, E. L. F. Moore, T. J. Peters, and A. C. Russell. Topological neighborhoods for spline curves : Practice & theory. *Lecture Notes in Computer Science: Real Number Algorithms*, 5045:149 – 161, 2008.
16. E. L. F. Moore. *Computational Topology of Spline Curves for Geometric and Molecular Approximations*. PhD thesis, The University of Connecticut, 2006.
17. E. L. F. Moore, T. J. Peters, and J. A. Roulier. Preserving computational topology by subdivision of quadratic and cubic Bézier curves. *Computing*, 79(2):317–323, 2007.
18. J. Peters and X. Wu. On the optimality of piecewise linear max-norm enclosures based on slefes. In *Proceedings of Curves and Surfaces, St Malo 2002*. Vanderbilt Press, 2003.
19. L. Piegl and W. Tiller. *The* NURBS *Book, 2nd Edition*. Springer, New York, NY, 1997.
20. Carlo H. Séquin. CAD tools for aesthetic engineering. *Computer-Aided Design*, 37(7):737–750, 2005.

The Stability of the Apparent Contour of an Orientable 2-Manifold *

Herbert Edelsbrunner[1,2,3], Dmitriy Morozov[4], and Amit Patel[1,2]

[1] Dept. Comput. Sci., Duke Univ., Durham, North Carolina, USA.
[2] IST Austria (Institute of Science and Technology Austria).
[3] Geomagic, Research Triangle Park, North Carolina, USA.
[4] Dept. Comput. Sci. and Math, Stanford Univ., Stanford, California, USA.

Abstract. The *(apparent) contour* of a smooth mapping from a 2-manifold to the plane, $f : \mathbb{M} \to \mathbb{R}^2$, is the set of critical values, that is, the image of the points at which the gradients of the two component functions are linearly dependent. Assuming $\mathbb{M}$ is compact and orientable and measuring difference with the erosion distance, we prove that the contour is stable.

1 Introduction

The most familiar setting of the problem studied in this paper is the view of a three-dimensional, solid body. We only see its surface and only one side at a time, but we get cues about its shape from the curve of points at which the surface normal is orthogonal to the viewing direction [17]. The view is the projection to a plane and its apparent contour is the image of the mentioned curve under the projection. Common roughly synonymous terms are fold, silhouette, outline, and profile. Only the first of these terms has a precise meaning introduced by Whitney [20]. Specifically, he defines *fold points* and *cusp points* that admit parametrizations of the neighborhood such that the mapping can locally be written as $f(x_1,x_2) = (x_1^2,x_2)$ and $f(x_1,x_2) = (x_1(x_1^2 - x_2),x_2)$, and he showed that these are the only kinds of critical points that are stable under infinitesimal perturbations. We will refer to them as *double points* and *triple points* of the mapping. A related concept is the Jacobi curve as introduced in [9]. This is the set of critical points, and its image is the apparent contour. In computer graphics, the contour of the projection of a surface is often used for artistic enhancements of displays [6, 7]. In the typical case, the computational cost of the contour is significantly smaller than that of the entire surface [13, 16]. This motivates its use in efficient rendering; see [15] for a survey of algorithms generating contours. Additional applications are for shadow calculations, occlusion testing, and the simplification of surface models [8, 19].

The main result in this paper is a quantitative contribution to the structural stability of the apparent contour. This study began with Whitney's seminal paper [20] which originated the related fields of catastrophe theory [2] and singularity theory [14]. Looking at smooth mappings of manifolds, these fields focus on the structure of singularities

*This research is partially supported by the Defense Advanced Research Projects Agency (DARPA) under grants HR0011-05-1-0007 and HR0011-05-1-0057.

and their stability under infinitesimal perturbations. In contrast to these studies, we allow for more severe perturbations and we quantify the changes in the contour. In a nutshell, we prove that a small perturbation of the mapping of an orientable 2-manifold to the plane changes the apparent contour only slightly. This seems plausible but is false for naive measurements of the distance between two contours. Indeed, even small perturbations can introduce arbitrarily many creases, each delimited by arbitrarily long contour lines. The crucial insight is that creases are thin, that is, they are delimited by pairs of contour lines that run roughly parallel to each other at close distance. We thus have two cases: creases that are thin in a technical sense that we will make precise shortly, and contour lines that are close to contour lines of the unperturbed mapping. This distinction is crucial in any effort to simplify the contour of a mapping in a way that retains its essential character.

Outline. Section 2 explains the setting for our study. Section 3 introduces important concepts. Section 4 presents our main result, a global statement of stability of the apparent contour. Section 5 contains the proof. Section 6 concludes the paper.

2 The Setting

In this section, we describe the setting, namely generic, smooth mappings from an orientable 2-manifold to the plane.

The apparent contour. Instead of projections of surfaces, we consider the more general setting of mappings $f : \mathbb{M} \to \mathbb{R}^2$, in which $\mathbb{M}$ is a compact, orientable 2-manifold without boundary. It may or may not be embedded in $\mathbb{R}^3$. We assume the mapping is smooth and satisfies a small number of requirements we need in the proof of our result. This includes that for most points $x \in \mathbb{M}$, the derivative of f at x, which we denote as $\mathrm{D}f(x) : \mathbb{R}^2 \to \mathbb{R}^2$, is surjective. We call these the *regular points* of f. All other points of $\mathbb{M}$ are *critical points* of f and their images are *critical values*. A point in $\mathbb{R}^2$ is a *regular*

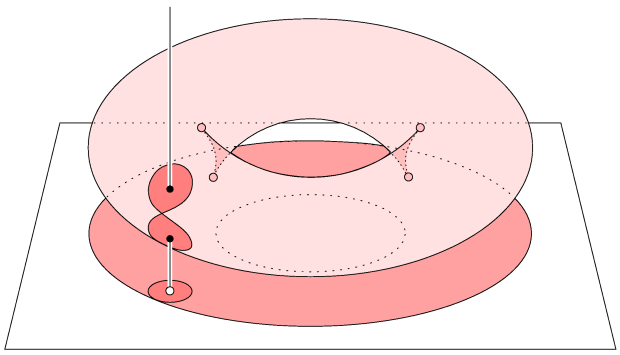


Fig. 1. The projection of the torus to the plane. The distance function defined by the marked value in the plane is illustrated by showing one of its sublevel sets.

value if it is not a critical value or, equivalently, if all its preimages are regular points. We call the set of critical values the *(apparent) contour* of the mapping, denoting it by Contour(f). The adjective serves as a reminder that we are not talking about a structural property of the 2-manifold but rather of its mapping to the plane. We refer to Figure 1 for an example. The vertical projection of the torus to the plane below it has a contour that consists of two concentric circles. In contrast, the projection to the drawing plane is more complicated, with the inner circle twisting up twice, forming a triangle on the left and another one on the right. Two vertices of each triangle are *cusps*, that is, values at which the contour comes to a sudden halt and reverses its direction. The third vertex is a *crossing*, that is, a value at which the contour intersects itself.

Distance functions. Fixing a value $a \in \mathbb{R}^2$, we let $f_a : \mathbb{M} \to \mathbb{R}$ be the *distance function* that maps every point x to the Euclidean distance of its image from a, that is, $f_a(x) = \|f(x) - a\|_2$. In Figure 1, one such distance function is illustrated by showing the value, the two points in its preimage, a disk around the value, and the preimage of the disk. The *sublevel set* of f_a for threshold $r \geq 0$ is the set of points with function value at most r, denoted as $\mathbb{M}_r(a) = f_a^{-1}[0,r]$. Writing $B_r(a)$ for the closed disk with center $a \in \mathbb{R}^2$ and radius r, the sublevel set of f_a is the preimage of this disk, $\mathbb{M}_r(a) = f^{-1}(B_r(a))$. Following Morse theoretical ideas, we increase r and notice that the sublevel set changes its topology only at critical values. To explore this, we apply the chain rule to get the derivative of f_a at a point x as the composition of the derivative of f at x and the scalar product with $u = f(x) - a$. Writing $\mathrm{D}f(x) : \mathbb{R}^2 \to \mathbb{R}^2$ for the former and $\sigma_u : \mathbb{R}^2 \to \mathbb{R}$ for the latter, we have $\mathrm{D}f_a(x) = \sigma_u \circ \mathrm{D}f(x) : \mathbb{R}^2 \to \mathbb{R}$. The point x is regular for f_a if $\mathrm{D}f_a(x)$ is surjective and it is critical if $\mathrm{D}f_a(x)$ is the zero function. If x is regular for f then $\mathrm{D}f(x)$ is surjective, and unless $f(x) = a$, this implies that $\mathrm{D}f_a(x)$ is surjective and hence x is regular for f_a. On the other hand, if x is critical for f then it may or may not be critical for f_a but it will be critical for the distance function defined by another value.

CRITICAL POINT LEMMA. A point $x \in \mathbb{M}$ is critical for f iff there exists a value $a \neq f(x)$ in $\mathbb{R}^2$ such that x is critical for f_a.

Genericity. We aim at limiting the class of mappings to those with manageable properties. For reasons of exposition, we do not strive to make the class as large as possible but rather large enough to be interesting. In particular, we sacrifice some generality to avoid the need to explain homology groups before giving the proofs.

DEFINITION. Let $\mathbb{M}$ be a compact, orientable 2-manifold without boundary. We call a smooth mapping $f : \mathbb{M} \to \mathbb{R}^2$ *generic* if

(I) the distance function, f_a, is tame for every value $a \in \mathbb{R}^2$;
(II) there are no critical points of f beyond double and triple points;
(III) the apparent contour of f has finitely many cusps and crossings.

Condition (I) means that f_a has only a finite number of critical values and every sublevel set consists only of finitely many components with finitely many holes. Condition (II) prohibits critical points other than the two simple types, fold points and cusp points.

Not allowing other, more complicated critical points is convenient and not a serious restriction since the corresponding mappings are dense in the larger class of smooth mappings [20]. Condition (III) implies the existence of a finite, smooth stratification of the plane compatible with the contour. Specifically, with $\mathbb{X}^1 = \text{Contour}(f)$ and $\mathbb{X}^0$ the set of cusps and crossings, we have a decomposition $\emptyset = \mathbb{X}^{-1} \subseteq \mathbb{X}^0 \subseteq \mathbb{X}^1 \subseteq \mathbb{X}^2 = \mathbb{R}^2$ such that each *stratum*, $S^i = \mathbb{X}^i - \mathbb{X}^{i-1}$, is either empty or an i-dimensional manifold. Calling the components of the strata the *pieces* of the stratification, we have only a finite number and each piece is smoothly embedded in $\mathbb{R}^2$. We call the pieces in S^2 the *chambers* of the stratification.

3 The Concepts

In this section, we introduce the main concepts needed to give a precise statement of our result.

Degree. An important step in our construction is the assessment of the shape of a component in the sublevel set of a distance function. For this purpose, we use the standard concept of degree, which we first define for the entire 2-manifold. To begin, we orient all sufficiently small, simple, closed curves in $\mathbb{M}$ in a consistent manner. This is possible because $\mathbb{M}$ is orientable. Let $a \in \mathbb{R}^2$ be a regular value and $f(x) = a$. A sufficiently small, simple, closed curve going around x in $\mathbb{M}$ maps injectively to a closed curve going around a in $\mathbb{R}^2$. We count $+1$ if this curve goes around a in a counterclockwise order and -1 if it goes around a in a clockwise order. Finally, we sum these numbers over all preimages of a and denote the result as $\deg(f,a)$. Extending this definition to critical values, we count 0 for each double point and $+1$ or -1 for each triple point. Moving a continuously from one value to another does not change this number. This is clear if we stay inside the same chamber but also when we cross the contour, picking up or losing two points whose contributions cancel each other. This implies $\deg(f,a) = \deg(f,b)$ for all $a,b \in \mathbb{R}^2$, and it makes sense to call this value the *degree* of f, denoted as $\deg(f)$. Since $\mathbb{M}$ is compact, its image under f does not exhaust $\mathbb{R}^2$. The degree of f at a value outside the image vanishes, which implies $\deg(f) = 0$.

Next, consider the closed disk $B_r(a)$ with center $a \in \mathbb{R}^2$ and radius $r \geq 0$. As mentioned earlier, the preimage of this disk is the sublevel set of f_a for threshold r. Let C be a component of this sublevel set and $f|_C : C \to \mathbb{R}^2$ the restriction of f to C. For each value b in $B_r(a)$, we get $\deg(f|_C, b)$ by summing the contributions over all points in the preimage, ${f|_C}^{-1}(b) = f^{-1}(b) \cap C$. As before, this number is the same at all values in the disk so we can call it the *degree* of $f|_C$. However, the number may change when we leave the disk so the degree is not necessarily zero.

Level sets and well function. We study the contour in terms of the family of preimages of all values. For a given value $a \in \mathbb{R}^2$, we call the preimage the *level set* of f at a. Equivalently, this is the zero set of the corresponding distance function, $f^{-1}(a) = f_a^{-1}(0)$. By Condition (I), this is a finite set of points in $\mathbb{M}$. The central concept in our approach is the health of these points. Considering the sublevel set, $\mathbb{M}_r(a) = f_a^{-1}[0,r]$, we wish to distinguish between components that necessarily map to the entire disk and components that can be pushed off the disk with moderate effort. We call a component C of

$\mathbb{M}_r(a)$ *well* if the degree of $f|_C$ is non-zero and *ill* if the degree is zero. For $r = 0$, each point in the level set forms its own component, which is well if the point is regular or a cusp and ill if it is a double point. As we increase r, we get a nested sequence of sublevel sets, $\mathbb{M}_r(a) \subseteq \mathbb{M}_s(a)$ for all $0 \leq r \leq s < \infty$. If the interval $[r,s]$ does not contain any critical value of f_a then the components of $\mathbb{M}_r(a)$ grow continuously into those of $\mathbb{M}_s(a)$ without changing their degree and status. At a simple critical value of f_a, we either encounter a new component or we merge two old components into one. The newly formed component has vanishing degree and is therefore ill from the start. When we merge two components, we add their degrees. The status of the new component thus depends only on the status of the old components. Specifically, merging a well and an ill component gives a well component, while merging two well or two ill components gives an ill component. There are also non-simple critical values, where we encounter two or more critical points at the same time or we encounter a cusp approaching it from its normal direction. In such a case, the change in the level set can be understood as the composition of a few simple changes as described.

The history of a component in the nested sequence of sublevel sets is therefore straightforward. If the component begins as a regular point or a triple point then it starts out well and falls ill later, at some critical value of f_a. We call this a *terminal critical value* to distinguish it from others at which no component falls ill. If the component begins as a double point then it is ill from the start. Once a component is ill, it does not get well any more (except it can become part of another, well component). It thus makes sense to introduce a function $\varphi : \mathbb{M} \to \mathbb{R}$ defined by mapping x to the terminal critical value of f_a, with $a = f(x)$, at which the component that contains x falls ill. We call φ the *well function* of f and $\varphi(x)$ the *well threshold* of x. We have $\varphi(x) = 0$ iff x is a double point and $\varphi(x) > 0$ if x is a regular point or a triple point.

Well diagrams. Fixing a value $a \in \mathbb{R}^2$, we get a well threshold for each point $x \in f_a^{-1}(0)$. We collect these thresholds to form a multiset of real numbers, called the *well diagram* of f_a and denoted as $\mathrm{Dgm}(f_a)$. For most regular values a of f, this diagram consists of an even number of positive thresholds that come in equal pairs. For each pair, the two contributing points lie on patches facing opposite directions. A threshold in $\mathrm{Dgm}(f_a)$ is *simple* if it is positive and occurs only twice. A non-trivial property of the well diagram is its stability; see the Stability Theorem for Well Diagrams in Section 5 but also [11]. To make this precise, let $a, b \in \mathbb{R}^2$ and let $0 \leq u_1 \leq u_2 \leq \ldots \leq u_l$ and $0 \leq v_1 \leq v_2 \leq \ldots \leq v_l$ be the thresholds in the well diagrams of f_a and f_b, possibly after adding zeros to the shorter sequence so we get the same length for both. The mentioned theorem states that $|u_i - v_i|$ is bounded from above by the L_∞-difference between the two functions. Using the triangle inequality, we get $\|f_a - f_b\|_\infty \leq \|a - b\|_2$ and therefore

$$\max_{1 \leq i \leq l} |u_i - v_i| \quad \leq \quad \|a - b\|_2. \tag{1}$$

In words, corresponding well thresholds change at most by the Euclidean distance between the values in the plane.

Surgery. The stability of the well diagram expressed in (1) provides some hope that the well function defined earlier is continuous. This is indeed the case at points with simple

well thresholds. Similarly, φ is continuous at double points, where it is zero. However, continuity is not guaranteed at points with positive but non-simple well thresholds. Let $a \in \mathbb{R}^2$ be a value and $u > 0$ a threshold that occurs at least four times in $\mathrm{Dgm}(f_a)$. Equivalently, a is equidistant to at least two different values of the contour. Take for

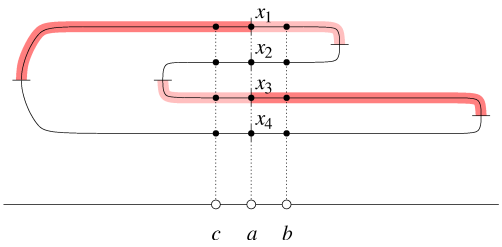


Fig. 2. Schematic cross-section of the twisted triangle in Figure 3. To make the well function continuous, we cut at x_1 and x_3 and reglue the sides as indicated by the light and dark shading.

example a value a somewhere in the interior of the bold, black segment connecting the upper left cusp with the center of the twisted triangle in Figure 3. It has four points in its preimage, x_1, x_2, x_3, x_4, indexed from front to back in the picture. Figure 2 shows a section of the configuration, crossing the black, bold segment at a. These four points have four identical well thresholds. For a value b to the right of a, the top (front) two points in the preimage have a small well threshold, while for a value c to the left of a, the middle two points have a small well threshold. The other preimages of b and c have a large well threshold. As we move from b to c, we observe a jump of φ at x_1 and x_3. The same jump occurs along the entire length of the black, bold segment. We remedy the discontinuity by cutting along the segment and regluing the sides as necessary to get continuity. In particular, the surfaces to the left of x_1 and to the right of x_3 are glued and so are the surfaces to the right of x_1 and to the left of x_3.

Branch points. Even more interesting is what happens at the center, a, of the twisted triangle, the lower right endpoint of the bold segment in Figure 3. It has three closest values on the contour. We study the structure by going around a in a counterclockwise circle and drawing the well diagrams as we go. Each value b on this circle has four preimages, y_1, y_2, y_3, y_4, indexed from front to back, as before. Growing the disk centered at b, we get a tree that describes how the components of the preimage merge until only one component remains. All four components start out well and fall ill in pairs during the process. This is illustrated in Figure 3, where well components are represented by bold branches in the trees and their falling ill is marked by shaded dots. As discussed earlier, there is a switch between y_1 and y_3 when we cross the bold segment. Symmetrically, there is a switch between y_2 and y_4 when we cross the segment connecting the lower cusp with the center of the triangle. The switches imply that we have to go around the circle twice to return to the original configuration. In other words, the surgery along the bold segment creates a *branch point* at the center of the trian-

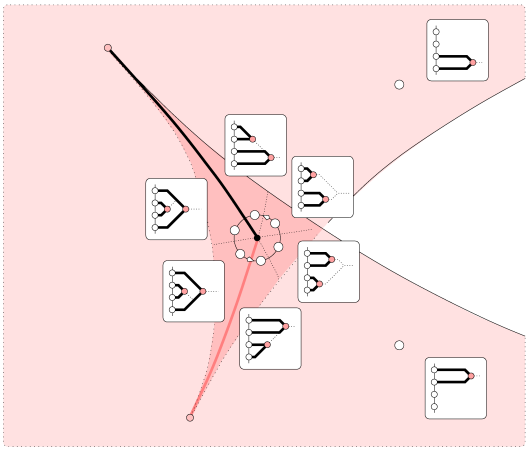


Fig. 3. Enlarged view of the twisted triangle to the left of the hole in the torus in Figure 1. The trees sketch the health histories of the points in the level sets at the marked values.

gle, that is, a point with a disk neighborhood that covers the neighborhood of $a \in \mathbb{R}^2$ twice. Using complex numbers to parametrize the neighborhood of a, the map to the neighborhood of the branch point can locally be written as $z \mapsto z^2/\|z\|$.

We note that the situation leading to the creation of the branch point reminds us of the concept of a ring species in biology; see e.g. [12]. Locally, at a value b, we seem to have two distinct species, y_1 and y_3, which we discover to be the same if we take a more global view of the situation.

Summary. Using the distance function defined for a value $a \in \mathbb{R}^2$, we have defined well thresholds for the points $x \in f^{-1}(a)$, and by exhausting all values in the plane, we have constructed a well function, $\varphi : \mathbb{M} \to \mathbb{R}$. Similar to the elevation function defined in [1], the well function is continuous almost everywhere but not necessarily everywhere. The stability of the well diagram implies that we can do surgery to change $\mathbb{M}$ to a 2-manifold with boundary, Φ, on which the well function is continuous. Specifically, we cut $\mathbb{M}$ along the curve of critical points of f. Doing so, we double every point to form the boundary of the 2-manifold with boundary. In addition, we cut and reglue along select curves originating at triple points. When we cut, we double the points and when we glue, we identify points in pairs. The two operations change the topology but cancel each other's effect on the multiplicity of points in the interior of the cut lines. Each such line starts at the third copy of a triple point (the first two copies are part of the boundary) and either ends at the third copy of another triple point or at a branch point. For reasons that will become clear later, we keep each branch point as two points with indistinguishable neighborhoods. The result is a non-Hausdorff 2-manifold with boundary, Φ, and a continuous well function, $\varphi : \Phi \to \mathbb{R}$. It vanishes along the boundary and is positive everywhere else.

4 The Result

In this section, we compare two mappings of the same 2-manifold and relate the difference between the contours to the difference between the mappings. We have two statements of stability. The first is straightforward and leads up to the second statement, our main result.

Silhouette stability. The *silhouette* of a mapping $f : \mathbb{M} \to \mathbb{R}^2$ is the boundary of the image, $\mathrm{Sil}(f) = \mathrm{bd}\,\mathrm{im}\,f$. Thinking of the image as the foreground and its complement as the background, the silhouette is the subset of the contour that separates foreground from background. To compare the silhouette of f with that of another mapping $g : \mathbb{M} \to \mathbb{R}^2$, we define the *dilation* of a set $A \subseteq \mathbb{R}^2$ by a radius $\varepsilon \geq 0$ as the set of points in $\mathbb{R}^2$ at distance at most ε from some point in A. We denote this set by $A^{+\varepsilon}$. The *Hausdorff* or *dilation distance* between two sets $A, B \subseteq \mathbb{R}^2$ is the infimum of the radii for which each dilated set contains the other, un-dilated set,

$$D(A,B) \quad = \quad \inf\{\varepsilon \mid A \subseteq B^{+\varepsilon} \text{ and } B \subseteq A^{+\varepsilon}\}.$$

Setting $\varepsilon = \max_{x \in \mathbb{M}} \|f(x) - g(x)\|_2$, we can be sure that every value in the image of f has a value in the image of g at distance at most ε. Together with the symmetric relation, this implies our first result.

SILHOUETTE STABILITY LEMMA. The Hausdorff distance between the images of f and g is $D(\mathrm{im}\,f, \mathrm{im}\,g) \leq \max_{x \in \mathbb{M}} \|f(x) - g(x)\|_2$.

This result is nothing short of trivial and allows for easy generalizations to higher dimensions, spaces that are not manifolds, and mappings that are neither generic nor smooth. Note that the small Hausdorff distance between the images does not imply that the two silhouettes are everywhere close. Indeed, it allows for small holes arbitrarily far from the other silhouette.

Erosion distance. When we consider the entire contour then small holes cannot disappear without a trace. To the contrary, little islands may appear or disappear anywhere inside the foreground. This motivates us to define the *erosion* of a set $A \subseteq \mathbb{R}^2$ by a radius $\varepsilon \geq 0$ is obtained by removing all points at distance at most ε from the complement, that is, $A^{-\varepsilon} = \mathbb{R}^2 - (\mathbb{R}^2 - A)^{+\varepsilon}$. The *complementary Hausdorff* or *erosion distance* between two sets $A, B \subseteq \mathbb{R}^2$ is the infimum of the radii for which each eroded set is contained in the other, un-eroded set,

$$E(A,B) \quad = \quad \inf\{\varepsilon \mid A^{-\varepsilon} \subseteq B \text{ and } B^{-\varepsilon} \subseteq A\}.$$

To extend the idea of erosion to the manifold, we note a relation between the well function and the Euclidean distance in the image stated as the Well Function Lemma in Section 5. Specifically, $\varphi(x)$ is the distance between $a = f(x)$ and a locally closest value of $\mathrm{Contour}(f)$. In other words, $\varphi(x)$ measures how far x is from the relevant portion of the boundary of Φ, and this measure is taken in the image rather than on the manifold. Eroding in the plane thus generalizes to taking a superlevel set of the well function,

that is, $\Phi^{-\varepsilon} = \varphi^{-1}[\varepsilon,\infty)$. Letting $g : \mathbb{M} \to \mathbb{R}^2$ be another generic, smooth mapping and $\gamma : \Gamma \to \mathbb{R}$ its well function after surgery, we define $\Gamma^{-\varepsilon} = \gamma^{-1}[\varepsilon,\infty)$. The *erosion distance* between Φ and Γ is then the infimum of the radii $\varepsilon \geq 0$ for which there are injections $\imath_f : \Phi^{-\varepsilon} \to \Gamma$ and $\imath_g : \Gamma^{-\varepsilon} \to \Phi$ such that $f(x) = g \circ \imath_f(x)$ and $g(y) = f \circ \imath_g(y)$ for all points x and y.

Contour stability. We are now ready to state the main result of this paper. It compares the image of the eroded 2-manifold, $\Phi^{-\varepsilon}$, using f, with the image of the un-eroded 2-manifold, Γ, using g, where $\varepsilon = \max_{x\in\mathbb{M}} \|f(x) - g(x)\|_2$, as before. Specifically, it says the second mapping covers every value in $\mathbb{R}^2$ at least as often as the first mapping. The same is true if we exchange f and g.

CONTOUR STABILITY THEOREM. Let $\mathbb{M}$ be a compact, orientable 2-manifold without boundary and $f, g : \mathbb{M} \to \mathbb{R}^2$ two generic, smooth mappings. Then the erosion distance is $E(\Phi,\Gamma) \leq \max_{x\in\mathbb{M}} \|f(x) - g(x)\|_2$.

We illustrate the result in Figure 4, which shows the familiar projection of the torus superimposed on a perturbation of that projection. The perturbed mapping has two extra cusps connected to each other by two contour lines bounding a narrow lip-shaped chamber. Cutting along the corresponding curves of critical points, we get a hole in the surface, which we mend by cutting and regluing along two preimages of the medial line between the two contour lines of the lips; see Figure 4. We get no additional branch point but instead two new components, each covering the lips once.

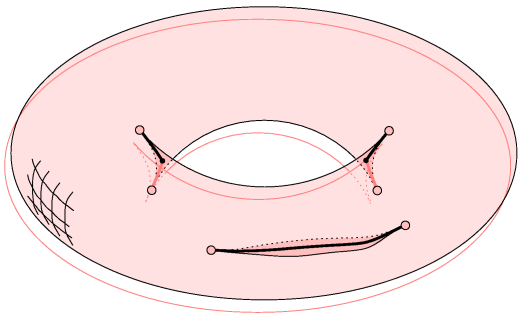


Fig. 4. Superposition of the faint contour of the original mappings of the torus and the clear contour of the perturbed mapping.

At this juncture, we wish to draw attention to the fact we use injections in the definition of the erosion distance. Write $E_{\text{strong}}(\Phi,\Gamma)$ for the strong version in which we require $\imath_f$ and $\imath_g$ be embeddings. Clearly, $E(\Phi,\Gamma) \leq E_{\text{strong}}(\Phi,\Gamma)$ so that substituting the strong for the original version of erosion distance would give a stronger theorem. Our proof does not support this strengthening. Although we currently do not have an example that shows such a strengthening is impossible, we believe such examples exist.

5 The Proof

In this section, we present the proof of our main result, delegating the bulk of the underlying algebraic construction to [11].

From components to homology groups. In lieu of the components in the sublevel set, $\mathbb{M}_r(a)$, we consider the 0-dimensional homology group of that set, which we denote as $\mathsf{F}_r(a) = \mathsf{H}_0(\mathbb{M}_r(a))$. With this formalization, we gain access to the concept of persistence, as introduced in [10]. Particularly important is the stability of the persistence diagram, which was established for tame functions in [5]. To explain this result, we consider again the nested sequence of sublevel sets, $\mathbb{M}_r(a) \subseteq \mathbb{M}_s(a)$ for $0 \leq r \leq s < \infty$. The inclusion between two sublevel sets induces a homomorphism between the corresponding homology groups, giving rise to $0 \to \ldots \to \mathsf{F}_r(a) \to \mathsf{F}_s(a) \to \ldots$, which we call a *filtration*. Within it, a component is *born* at $\mathsf{F}_r(a)$ if the minimum function value of its points is r, and it *dies entering* $\mathsf{F}_s(a)$ if it merges at s with another component born before itself. The component is thus characterized by two numbers, r and s, which we interpret as coordinates of a point in the plane. We set $s = \infty$ if the component never dies, so we need the extended plane, $\bar{\mathbb{R}}^2 = [-\infty, \infty]^2$, to draw the points. Representing each component that ever appears in the filtration, we get a multiset in $\bar{\mathbb{R}}^2$, which we call the *persistence diagram* of f_a, denoted as $\mathrm{Dgm}(f_a)$. For a technical reason that will be clear shortly, we add infinitely many copies of every point on the diagonal to the diagram. Letting $g : \mathbb{M} \to \mathbb{R}^2$ be a second mapping, we get a second distance function and a second persistence diagram, $\mathrm{Dgm}(g_a)$. Using the triangle inequality, it is easy to show that the difference between the distance functions is $\|f_a - g_a\|_\infty \leq \varepsilon$, where $\varepsilon = \max_{x \in \mathbb{M}} \|f(x) - g(x)\|_2$. The mentioned stability result states that the bottleneck distance between the persistence diagrams is bounded by the difference between the functions and therefore by ε, that is,

$$W_\infty(\mathrm{Dgm}(f_a), \mathrm{Dgm}(g_a)) \quad \leq \quad \varepsilon, \tag{2}$$

see [5]. This means there is a perfect matching between the points in the two diagrams such that the L_∞-distance between matched points is at most ε. This result suffices to derive a local statement of contour stability but not the stronger, global statement given in Section 4.

Equivalence of definitions. To go the extra mile, we need to understand the subgroups of the homology groups generated by the well components of the sublevel sets. We refer to these as the *well groups*, $\mathsf{U}_r(a) \subseteq \mathsf{F}_r(a)$. These groups have been studied in [11], where a different, more general definition is used. We reproduce this definition. Letting $f, h : \mathbb{M} \to \mathbb{R}^2$ be two mappings, we call h a *ρ-perturbation* of f if $\max_{x \in \mathbb{M}} \|h(x) - f(x)\|_2 \leq \rho$. Note that the level set of h at a is contained in the sublevel set of f_a for radius ρ, that is, $h^{-1}(0) \subseteq \mathbb{M}_\rho(a)$. Hence, there is a homomorphism $\mathsf{j}_h : \mathsf{H}_0(h^{-1}(a)) \to \mathsf{F}_\rho(a)$. The image of j_h is a subgroup of $\mathsf{F}_\rho(a)$ and so is the common intersection of like images, $\bigcap_h \mathrm{im}\, \mathsf{j}_h \subseteq \mathsf{F}_\rho(a)$, where h ranges over all ρ-perturbations of f. Finally, we set $\rho = r + \delta$ for a sufficiently small $\delta > 0$, and we define $\mathsf{W}_r(a)$ as the largest subgroup of $\mathsf{F}_r(a)$ so its image in $\mathsf{F}_\rho(a)$ is contained in this common inter-

section. The group $\mathsf{W}_r(a)$ is what [11] calls the well group of $\mathbb{M}_r(a)$. Our aim here is to prove that for the setting in this paper, the two definitions give the same groups.

WELL GROUP LEMMA. We have $\mathsf{U}_r(a) = \mathsf{W}_r(a)$ for every $a \in \mathbb{R}^2$ and every $r \geq 0$.

Proof. Fixing $a \in \mathbb{R}^2$, we consider a point $x \in f^{-1}(a)$, and for every $r \geq 0$, we let C_r be the component of $\mathbb{M}_r(a)$ that contains x.

CASE 1: C_r is well. We show that there exists $\delta > 0$ such that $C_{r+\delta} \cap h^{-1}(a) \neq \emptyset$ for every ρ-perturbation h of f, where $\rho < r + \delta$. Specifically, we choose $\delta < \varphi(x) - r$ and note that $C_{r+\delta}$ is well. Consider the homotopy defined by $g_t(x) = (1-t)f(x) + th(x)$. Since the boundary of $C_{r+\delta}$ is too far from the center for its image to reach a, the degree of g_t restricted to $C_{r+\delta}$ at a remains unchanged. This degree is non-zero for $f = g_0$ and therefore also non-zero for $h = g_1$. This implies that $h^{-1}(a)$ has a non-empty intersection with $C_{r+\delta}$, as required.

CASE 2: C_r is ill. We show that for every $\delta > 0$ there exists a $\rho < r + \delta$ and a ρ-perturbation h of f such that $C_r \cap h^{-1}(a) = \emptyset$. We use induction, following the change in the sublevel set as we increase the radius. The first time we have to prove something is when $r = \varphi(x)$. At this radius, two well components merge to form C_r, which is now ill. Let y be a double point at which the two components touch; see Figure 5. The per-

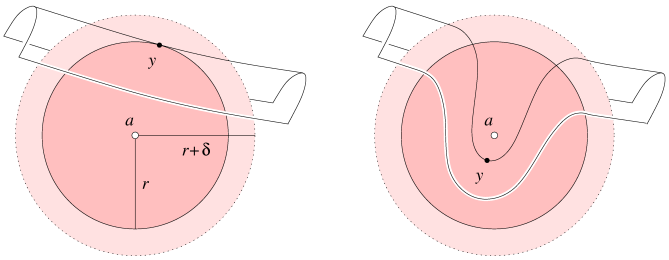


Fig. 5. Left: two well components meeting at y. Right: the locally perturbed mapping in which the two merged components avoid a.

turbation needs to move y beyond a, which it can do without changing f outside $C_{r+\delta}$. If there are two or more such double points, we move all of them beyond a the same way. We choose $\delta < s - \varphi(x)$, where s is the next, larger critical value of f_a, and call the resulting perturbation $h_0 : \mathbb{M} \to \mathbb{R}^2$. It is good for all radii $\varphi(x) \leq r < s$. Now suppose $r = s$ and the growing component merges with another, ill component, forming C_s. Let $h_1 : \mathbb{M} \to \mathbb{R}^2$ be the perturbation we constructed for this other component when it fell ill at $\varphi(x') < s$. Choose t such that $\max\{\varphi(x), \varphi(x')\} < t < s$. The two perturbations differ from each other in two disjoint components of $\mathbb{M}_t$. We can therefore combine them to get a new perturbation $h_{01} : \mathbb{M} \to \mathbb{R}^2$ that agrees with f outside these components, with h_0 inside one, and with h_1 inside the other component. The level set of h_{01} at a has empty intersection with C_r, as required. The claimed relationship follows by induction. □

Stability of diagram. While being more complicated algebraically, the persistence diagram of the well groups is simpler geometrically. Specifically, it is only 1-dimensional, namely precisely the well diagram introduced in Section 3. The complete proof of the stability of the well diagram is beyond the scope of this paper. The main idea is the realization that the well groups for a given value form a zigzag module as defined in [4]. We sketch the construction of this module for the distance function $f_a : \mathbb{M} \to \mathbb{R}$. By definition of a generic, smooth mapping, f_a has only finitely many critical values and therefore only finitely many different homology groups. We index them consecutively as F_i. Let $\mathsf{U}_i \subseteq \mathsf{F}_i$ be the corresponding well groups. A class may fall ill entering U_{i+1} because it dies entering F_{i+1} or because its image in F_{i+1} does not belong to U_{i+1}. To express the two cases algebraically, we let Q_i be the quotient formed by identifying all classes in U_i that differ only by a class that maps to zero in F_{i+1}. Inserting the quotient between the two well groups and connecting it with the obvious forward and backward maps, we get the zigzag module, $\ldots \leftarrow \mathsf{U}_i \to \mathsf{Q}_i \leftarrow \mathsf{U}_{i+1} \to \ldots$. It is characterized by its persistence diagram, like a filtration [4]. By the Well Group Lemma, this diagram is precisely the well diagram described in Section 3. Stability does not follow from general principles known yet but has been established in [11]. We skip the argument and state the result.

STABILITY THEOREM FOR WELL DIAGRAMS. Let $f, g : \mathbb{M} \to \mathbb{R}^2$ be two generic, smooth mappings. Then the bottleneck distance between the well diagrams of the distance functions at any value $a \in \mathbb{R}^2$ is $W_\infty(\mathrm{Dgm}(f_a), \mathrm{Dgm}(g_a)) \leq \|f_a - g_a\|_\infty$.

As mentioned earlier, the difference between the distance functions is bounded from above by $\varepsilon = \max_{x \in \mathbb{M}} \|f(x) - g(x)\|_2$.

Eroding the manifold. The stability of the well diagram justifies the surgery which turns $\mathbb{M}$ into a non-Hausdorff 2-manifold with boundary, Φ, such that the well function, $\varphi : \Phi \to \mathbb{R}$, is continuous. We recall that for each point $x \in \Phi$, the value, $\varphi(x)$, is the well threshold of x, that is, the terminal critical value of f_a, $a = f(x)$, at which the component of x in the sublevel set falls ill. The well threshold has another geometric interpretation. Letting $p : [0,1] \to \Phi$ be a path on the manifold after surgery, we consider its composition with the mapping, $f \circ p : [0,1] \to \mathbb{R}^2$, and write $\ell(p)$ for the length of the image, $f \circ p[0,1]$. Taking the infimum over all paths that start at x and end on the boundary, $\partial\Phi$, we get $\mathrm{dist}(x) = \inf_p \ell(p)$, the *distance* of $a = f(x)$ from the relevant portion of the contour. We note that $\mathrm{dist}(x)$ is not necessarily the distance to the nearest point on the contour but rather to the nearest point that affects the wellness of the component of x in the sublevel set of f_a.

WELL FUNCTION LEMMA. Let $f : \mathbb{M} \to \mathbb{R}^2$ be a generic, smooth mapping and $\varphi : \Phi \to \mathbb{R}$ its well function. Then $\varphi(x) = \mathrm{dist}(x)$ for every point $x \in \Phi$.

PROOF. Let $a = f(x)$. The point x belongs to the zero set of f_a and its component in the sublevel set falls ill at $\mathbb{M}_{\varphi(x)}(a)$. We write $R = \varphi(x)$ for short. The goal is to prove $R = \mathrm{dist}(x)$. It is easy to see that $R \leq \mathrm{dist}(x)$. By the Stability Theorem for Well Diagrams, we have $|\varphi(x) - \varphi(y)| \leq \|f_a - f_b\|_\infty$, and by the triangle inequality in $\mathbb{R}^2$, we have $\|f_a - f_b\|_\infty \leq \|a - b\|_2$, where $b = f(y)$. It follows that $\varphi(y) > 0$ for all points

y with $\|a - f(y)\|_2 < R$. Since φ is zero at the boundary, this implies that all points of $\partial\Phi$ are at Euclidean distance at least R from a.

The more difficult direction is to prove $\text{dist}(x) \leq R$. To get a contradiction, we assume $R < \text{dist}(x)$. Let $q : [0,1] \to \Phi$ be a path starting at $q(0) = x$ with length $\ell(q) = R$, and let $y = q(1)$ be its endpoint. It belongs to the component C of $\mathbb{M}_R(a)$ that contains x. Since $\varphi(y) > 0$, there is a positive radius δ such that the component, C', of $\mathbb{M}_\delta(b)$ that contains y is well, that is, the degree of f restricted to C' is non-zero. Since C and C' overlap, their degrees are the same and we can form the union to get a patch, $C \cup C'$, that has the same degree still. We do the same for all points y reachable from x by paths of length R, choosing $\delta > 0$ smaller than the minimum well threshold of any of these points. The result is a component C'' of $\mathbb{M}_{R+\delta}(a)$ that contains C and the restriction of f to C'' has the same degree as the restriction to C. Hence, C'' is well, contradicting the choice of R as the well value of x. ◻

Similarity of well functions. We have one more hurdle to clear, namely showing that the well functions for similar mappings are similar. Let $\varphi : \Phi \to \mathbb{R}$ and $\gamma : \Gamma \to \mathbb{R}$ be the well functions of the mappings $f, g : \mathbb{M} \to \mathbb{R}^2$. We say the *difference* between them is at most r, denoted as $\|\varphi - \gamma\|_\infty \leq r$, if there are subspaces $\Phi_0 \subseteq \Phi$ and $\Gamma_0 \subseteq \Gamma$ that contain all points with well threshold r or larger and a bijection $\iota : \Phi_0 \to \Gamma_0$ such that $f(x) = g \circ \iota(x)$ for every $x \in \Phi_0$ and $g(y) = f \circ \iota^{-1}(y)$ for every $y \in \Gamma_0$. We derive an upper bound on the difference between the two well functions.

HOMOTOPY LEMMA. Let $f, g : \mathbb{M} \to \mathbb{R}^2$ be two generic, smooth mappings with corresponding well functions $\varphi : \Phi \to \mathbb{R}$ and $\gamma : \Gamma \to \mathbb{R}$. Then the difference between the two well functions is $\|\varphi - \gamma\|_\infty \leq \max_{x \in \mathbb{M}} \|f(x) - g(x)\|_2$.

PROOF. We use the straight-line homotopy between f and g defined by $f_t(x) = (1 - t)f(x) + tg(x)$. All f_t are smooth but not necessarily generic. Nevertheless, the well diagram is defined for each distance function $(f_t)_a$. The Stability Theorem for Well Diagrams holds also for non-generic functions, implying that the points in these diagram vary continuously with a and t. Specifically, the bottleneck distance between the diagrams of $(f_t)_a$ and $(f_{t'})_a$ is bounded from above by $|t - t'|\varepsilon$, where ε is the maximum Euclidean distance between corresponding images, as before.

To relate φ with γ, we pick a point $\varphi(x)$ in the well diagram of $f_a = (f_0)_a$. Initializing the construction of a function $\alpha : [0,1] \to \mathbb{R}$, we set $\alpha(0) = \varphi(x)$. Increasing t, we continuously extend α until we either reach $t = 1$ or α vanishes. Whenever we reach $t = 1$, we get a point $y \in \Gamma$ with $\alpha(1) = \gamma(y)$. Because the slope of α is between $\pm\varepsilon$, we have $|\varphi(x) - \gamma(y)| \leq \varepsilon$. Collecting all pairs (x, y) generated by this process, we get the bijection $\iota : \Phi_0 \to \Gamma_0$ required by the claim. We get $\varphi(x) < \varepsilon$ for all $x \in \Phi - \Phi_0$ because α vanishes before reaching $t = 1$. The construction of the functions α can also be done in the other direction, starting at $t = 1$. Making sure we get the same pairs, we also get $\gamma(y) < \varepsilon$ for all $y \in \Gamma - \Gamma_0$, as required. ◻

Note that the paths connecting points x with y form a homeomorphism between Φ_0 and Γ_0, unless there are branch points in the graph of the homotopy connecting f and g. In the absence of such branch points, we can substitute a homeomorphism for the

bijection in the definition of difference between well functions and embeddings for the injections in the definition of erosion distance.

Finale. We are now in a position to tie up all ends and finish the proof of the Contour Stability Theorem. Let $\varphi : \Phi \to \mathbb{R}$ and $\gamma : \Gamma \to \mathbb{R}$ be the well functions of the mappings f and g. By the Well Function Lemma, eroding the 2-manifolds with boundary is the same as taking superlevel sets of the well functions, $\Phi^{-r} = \varphi^{-1}[r,\infty)$ and $\Gamma^{-r} = \gamma^{-1}[r,\infty)$. By the Homotopy Lemma, $\|\varphi - \gamma\|_\infty \leq \varepsilon$. We recall that this means there is a bijection, $\iota : \Phi_0 \to \Gamma_0$, that is compatible with the two mappings. Here, $\Phi_0 \subseteq \Phi$ and $\Gamma_0 \subseteq \Gamma$ contain all points with well threshold ε or larger, that is,

$$\begin{aligned} \Phi^{-\varepsilon} &= \varphi^{-1}[\varepsilon,\infty) &\subseteq \Phi_0; \\ \Gamma^{-\varepsilon} &= \gamma^{-1}[\varepsilon,\infty) &\subseteq \Gamma_0. \end{aligned}$$

Restricting the bijection to the superlevel set of φ, we get the injection $\iota_f : \Phi^{-\varepsilon} \to \Gamma$ defined by $\iota_f(x) = \iota(x)$. Symmetrically, restricting it to the superlevel set of γ, we get the injection $\iota_g : \Gamma^{-\varepsilon} \to \Phi$ defined by $\iota_g(y) = \iota^{-1}(y)$. By construction, $f(x) = g \circ \iota_f(x)$ for every $x \in \Phi^{-\varepsilon}$ and $g(y) = g \circ \iota_g(y)$ for every $y \in \Gamma^{-\varepsilon}$. It follows that the erosion distance between the 2-manifolds with boundary is $E(\Phi,\Gamma) \leq \varepsilon$, which completes the proof of the Contour Stability Theorem.

6 Discussion

An immediate application of our result is to the artistic representation of shapes using contours. Instead of the entire contour, or perhaps the entire visible contour, we advocate drawing only the portion that remains after a small erosion. A similar strategy may be used to improve the efficacy of shape matching methods that work by comparing contours [18].

The current statement of the Contour Stability Theorem is based on injections in the definition of the erosion distance. It would be nice to replace them by embeddings, but possible branch points in the homotopy as constructed in the proof of the Homotopy Lemma would contradict their existence. Can we find an explicit example in which at least one branch point occurs? Can we substitute piecewise embeddings for the injections?

References

1. P. K. AGARWAL, H. EDELSBRUNNER, J. HARER AND Y. WANG. Extreme elevation on a 2-manifold. *Discrete Comput. Geom.* **36** (2006), 553–572.
2. V. I. ARNOLD. *Catastrophe Theory.* Third edition, Springer-Verlag, Berlin, Germany, 1992.
3. H. BLUM. A transformation for extracting new descriptors of shape. In *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn (ed.), MIT Press, Cambridge, Massachusetts, 362–380, 1967.
4. G. CARLSSON AND V. DE SILVA. Zigzag persistence. Manuscript, Dept. Mathematics, Stanford Univ., Stanford, California, 2008.

5. D. COHEN-STEINER, H. EDELSBRUNNER AND J. HARER. Stability of persistence diagrams. *Discrete Comput. Geom.* **37** (2007), 103–120.
6. F. COLE, A. GOLOVINSKIY, A. LIMPAECHER, H. S. BARROS, A. FINKELSTEIN, T. FUNKHOUSER AND S. RUSINKIEWICZ. Where do people draw lines? SIGGRAPH Conf. Proc., *ACM Trans. Graphics* **27** (2008), 1–11.
7. D. DECARLO, A. FINKELSTEIN, S. RUSINKIEWICZ AND A. SANTELLA. Suggestive contours for conveying shape. *AMC Trans. Graph.* **22** (2003), 848–855.
8. F. DUGUET AND G. DRETTAKIS. Robust epsilon visibility. SIGGRAPH Conf. Proc., *ACM Trans. Graphics* **21** (2002), 567–575.
9. H. EDELSBRUNNER AND J. HARER. Jacobi sets of multiple Morse functions. In F. Cucker, R. Devore and P. Olver (eds.), *Foundations of Computational Mathematics, Minneapolis 2002*, 35–57, Cambridge University Press, 2004.
10. H. EDELSBRUNNER, D. LETSCHER AND A. ZOMORODIAN. Topological persistence and simplification. *Discrete Comput. Geom.* **28** (2002), 511–533.
11. H. EDELSBRUNNER, D. MOROZOV AND A. PATEL. Quantifying transversality by measuring the robustness of intersections. Manuscript, Dept. Comput. Sci., Duke Univ., Durham, North Carolina, 2009.
12. D. J. FUTUYAMA. *Evolutionary Biology.* Third edition, Sinauer Associates, 1998.
13. M. GLISSE AND S. LAZARD. An upper bound on the average size of silhouettes. *Discrete Comput. Geom.* **40** (2008), 241–257.
14. M. GOLUBITSKY AND V. GUILLEMIN. *Stable Mappings and Their Singularities.* Springer-Verlag, New York, 1973.
15. T. ISENBERG, B. FREUDENBERG, N. HALPER, S. SCHLECHTWEG AND T. STROTHOTTE. A developer's guide to silhouette algorithms for polygonal models. *IEEE Comput. Graph. Appl.* **23** (2003), 28–37.
16. L. KETTNER AND E. WELZL. Contour edge analysis for polyhedron projections. In *Geometric Modeling: Theory and Practice*, 379–394, eds. W. Straßer, R. Klein and R. Rau, Springer-Verlag, 1996.
17. J. J. KOENDERINK. What does the occluding contour tell us about solid shape? *Perception* **13** (1984), 321–330.
18. P. MIN, J. CHEN AND T. FUNKHOUSER. A 2D sketch interface for a 3D model search engine. SIGGRAPH Technical Sketches (2002), 138.
19. P. V. SANDER, X. GU, S. J. GORTLER, H. HOPPE AND J. SNYDER. Silhouette clipping. SIGGRAPH Conf. Proc., *Computer Graphics* (2000), 327–334.
20. H. WHITNEY. On singularities of mappings of Euclidean space. I. Mappings of the plane to the plane. *Ann. Math.* **62** (1955), 374–410.

On the Extraction of Long-living Features in Unsteady Fluid Flows

Jens Kasten[1], Ingrid Hotz[1], Bernd R. Noack[2], and Hans-Christian Hege[1]

[1] Zuse Institute Berlin (ZIB), {kasten;hotz;hege}@zib.de
[2] Berlin Institute of Technology MB1, Bernd.R.Noack@tu-berlin.de

Abstract. This paper proposes a Galilean invariant generalization of critical points of vector field topology for $2D$ time-dependent flows. The approach is based upon a Lagrangian consideration of fluid particle motion. It extracts long-living features, like saddles and centers, and filters out short-living local structures. This is well suited for analysis of turbulent flow, where standard snapshot topology yields an unmanageable large number of topological structures that are barely related to the few main long-living features employed in conceptual fluid mechanics models. Results are shown for periodic and chaotic vortex motion.

1 Introduction

With increasing computational power and advancement in experimental techniques, the focus in flow visualization has moved from steady to unsteady fields. The demands for analysis and visualization tools have changed accordingly. Many successful methods for steady fields, such as extraction of vector field topology, only provide an incomplete view on unsteady phenomena.

There are two viewpoints for describing a flow, dependent on the choice of independent variables. The Eulerian view assigns dynamic properties to fixed points in space, while the Lagrangian view assigns these to moving fluid parcels; the dynamic equations describe changes that occur to a fluid particle along its trajectory. This view corresponds to a natural extension of particle mechanics. Both views can be transformed into each other and offer different perspectives onto the flow behavior; for more details see [1]. Especially for unsteady flows, it is important to provide analysis tools offering both perspectives.

Vector field topology has permeated fluid dynamics since entering the scientific field [2]. It has significantly supported the development of conceptual flow models for steady flows or snapshots of time dependent flows. Standard vector field topology is based on streamline behavior and thus is appropriate to capture snapshot features. Pathline or streakline related features are not represented. Furthermore, topological features are not invariant under *Galilean transformations*. These are transformations between two frames of reference that differ by a constant relative motion. The distinguished points in vector field topology are fixed points that exhibit zero velocity. Thus, choosing a suitable Galilean transformation any location can be turned into a feature point. Additional practical limitations result from the complexity of the topological skeleton and its high feature density. Distinguishing long-living structures from short-living incoherent structures is not easy.

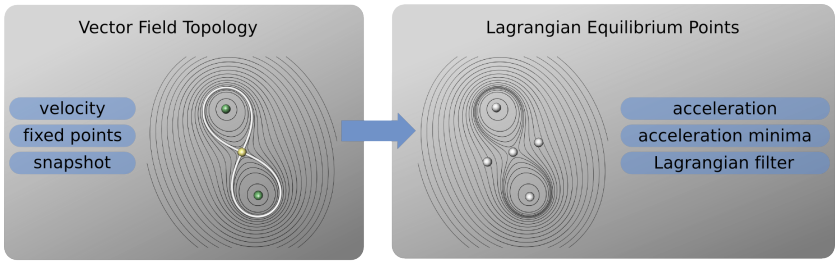


Fig. 1. Correspondences between critical points from standard vector field topology and Lagrangian equilibrium points. Displayed are the respective distinguished points with streamlines providing context.

Some of these problems have been tackled by the introduction of Lagrangian coherent structures. These long-living features can be extracted in a number of ways, for example by analysis of the *Finite Time Lyapunov Exponent* field [3]. This measure characterizes the separation of particles over time, providing a Galilean invariant Lagrangian view. Coherent features are depicted as ridges of high separation. Center or vortex-like features are not considered by this approach, since nearly no separation can be observed here.

This paper follows a different approach to Lagrangian coherent structures, motivated by two points: 1) The concept of critical points is successful, but its applicability to unsteady flow fields is limited. We introduce an unsteady analogon to the zero velocity definition of critical points. In the steady case, particles at fixed points have zero acceleration, which is a Galilean invariant property. Generalizing this behavior to unsteady fields, particles with low acceleration compared to their neighbors become particles of interest. We call these features *Lagrangian equilibrium points* (LEP). 2) Fluid flow researchers are mainly interested in the dominant structures that mainly influence the flow behavior. These interesting and influential features usually exist over some period of time. The time a particle exhibits a given property becomes a basic component of the analysis. In the proposed approach short-living structures are filtered out by a lifetime parameter leaving only salient features. We consider the extracted features to be a first building block of a *Finite Time Topology* (FTT). We verify the significance of this approach by applying it to basic well-known flow structures such as a mix of Oseen vortices (Sec. 5). We limit our considerations to unsteady 2D fields and leave an extension to 3D fields for future work.

2 Related Work

Streamline topology was introduced by Tobak and Peake [2] to the flow community and by Helman and Hesselink [4, 5] to the visualization community a few years later. In this early work, they define the concept of fixed points and integral curves connecting these, thereby building a topological skeleton. Afterwards, many extensions like simplifications and tracking algorithms have been published. Wee refer to the survey paper [6] and the references therein.

Appropriate vortex definitions and extraction methods are crucial for the understanding of complex flows. A variety of scalar quantities have been introduced to extract vortex regions. Two of the most commonly used vortex identifiers are Q [7] and λ_2 [8]. Both measures are based on the Jacobian matrix of the flow field and are therefore Galilean invariant. Geometrical approaches using streamline curvature for locating vortices [9] are not Galilean invariant. Other methods employ the parallel vectors operator [10] for computing global line-like features, such as vortex cores.

Most of the methods mentioned above are based either on streamlines or the Jacobian matrix. They are well suited for analysis of single time-slices but not for characteristics of unsteady flow fields. Thus, more attention has been paid to methods based on pathline analysis, representing the Lagrangian point of view. Theisel et al. [11] have presented an extension of streamline topology to pathlines. Weinkauf et al. [12] have generalized the parallel vectors method to detect cores of swirling motion. Fuchs et al. [13] accumulate Eulerian quantities along pathlines to add a Lagrangian view. Similarly, Shi et al. [14] explore the dynamical process of a flow by averaging the kinetic energy and momentum along pathlines.

Other features have also been identified as interesting by fluid flow researchers and have subsequently found their way into flow visualization. Haller, for instance, has introduced an analytical criterion for finite-time attracting and repelling material surfaces [15]. A further advancement has been the introduction of the Finite Time Lyapunov Exponent (FTLE) [3], which is a scalar quantity indicating the separation rate of infinitesimal close particles. Using ridge extraction [16], Lagrangian coherent structures can be captured. Garth et al. [17] have presented a computational less expensive, adaptive method to extract FTLE ridges.

3 Motivation

3.1 Acceleration and Lagrangian Equilibrium Points

The goal of this section is to explain the considerations that finally leads to the definition of *Lagrangian equilibrium points*. Vector field topology defines critical points as fixed points, as shown on the left hand side of Fig. 1. This definition does not satisfy the requirement of Galilean invariance. This fact motivates the investigation of alternative concepts for "distinguished points". In the following, let $\mathbf{v}$ be a flow field. As a technical requirement it is assumed that its spatial and time derivatives are bounded. Our starting point is the observation that for steady fields the particle acceleration at fixed points is equal to zero. Furthermore, particle acceleration is a Galilean invariant entity, see Appendix. The particle acceleration $\mathbf{a}$ is the material derivative of the field $\mathbf{v}$, i.e. the acceleration in a space-time point $(\mathbf{x},t)$ is given by

$$\mathbf{a}(\mathbf{x},t) = D\mathbf{v}/Dt = \partial_t \mathbf{v}(\mathbf{x},t) + (\mathbf{v}(\mathbf{x},t)\cdot\nabla)\,\mathbf{v}(\mathbf{x},t), \tag{1}$$

where ∂_t is the partial derivative with respect to t and ∇ the spatial gradient, i.e. (∂_x, ∂_y) for the two-dimensional case. The squared magnitude of the acceleration $\mathbf{a}$ defines a scalar field in the space-time domain by

$$\|\mathbf{a}\|^2 = \|\partial_t \mathbf{v}\|^2 + 2\,\partial_t \mathbf{v}\cdot((\mathbf{v}\cdot\nabla)\,\mathbf{v}) + \|(\mathbf{v}\cdot\nabla)\,\mathbf{v}\|^2. \tag{2}$$

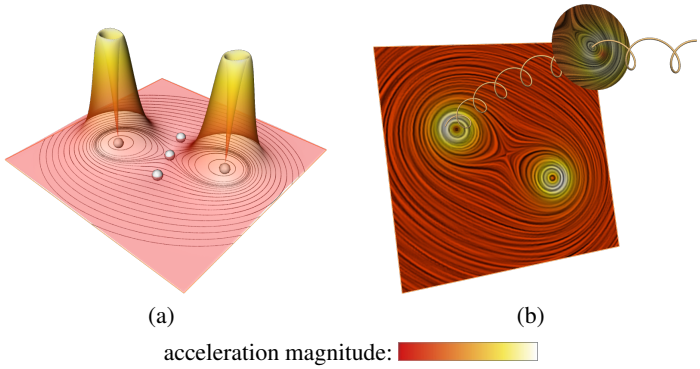


Fig. 2. Visualization of two co-rotating Oseen vortices in two dimensions (a) The acceleration magnitude of one time-slice is shown as a color-coded heightfield. The extracted points are the acceleration minima – which include fixed points. (b) The pathline of a particle seeded in one time-slice is displayed. The time is depicted as third dimension. The relative behavior of the flow field in the neighborhood of this particle at a later point in time is displayed on a surrounding circular disk. The acceleration magnitude is color coded in the line integral convolution (LIC) images of the flow field (color plate C. 15, page 259).

For steady fields the partial time derivative vanishes, $\partial_t \mathbf{v}(\mathbf{x},t) = 0$, resulting in

$$\mathbf{a}(\mathbf{x}) = D\mathbf{v}/Dt = (\mathbf{v}(\mathbf{x}) \cdot \nabla)\,\mathbf{v}(\mathbf{x}), \tag{3}$$

where $(\mathbf{v}(\mathbf{x}) \cdot \nabla) = \sum_{i=1}^{d} v_i \frac{\partial}{\partial x_i}$. It follows that at fixed points, where $\mathbf{v}$ is zero, the acceleration also equals zero and its magnitude $\|\mathbf{a}\|$ takes its *minimum value* 0, cf. Fig. 2(a). Thus, the set of fixed points is a subset of the zeros of the acceleration field.

The next step is to look at time-dependent vector fields. It turns out that for unsteady fields it is not sufficient to consider points where the acceleration of a fluid particle is equal to zero. In general unsteady flow fields, structures like centers and saddles evolve over time, and thus the acceleration does not vanish. This fact also follows from Eq. (2), since at fixed points the squared magnitude of the particle acceleration equals $\|\partial_t \mathbf{v}\|^2$, which does not vanish in general. However, the acceleration at fixed points is not zero but is still small compared to its neighborhood for time-dependent fields. This allows us to relax the condition of a vanishing acceleration to a less strong requirement, the *minimality* of $\|\mathbf{a}\|^2$.

As an illustrating example, Fig. 3 shows a steady and a convecting version of the Stuart vortex. For a detailed discussion on Stuart vortices see Panton [1]. Note that convection adds a non-stationary component to the steady field due to the moving frame of reference. For both fields the same acceleration minima are detected, while the location of fixed points are different.

As a result, acceleration and its minima are used as key features for unsteady flow fields. In the following, space-time points $(\mathbf{x_0}, t_0)$ where $\|\mathbf{a}(\mathbf{x_0}, t_0)\|$ takes a local minimum in space are called *Lagrangian equilibrium points*.

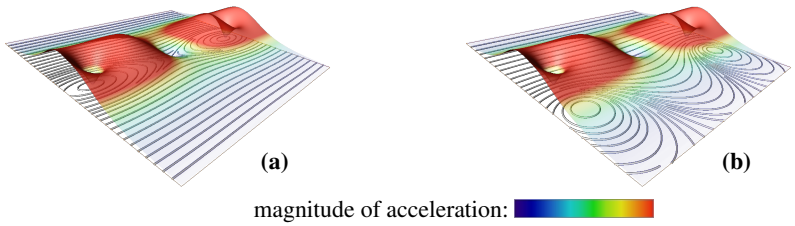


Fig. 3. Stuart vortices in two inertial frames of references: (a) vortices at rest, (b) convecting vortices such that the velocity at the bottom becomes zero. Heightfield and color represent the absolute value of the particle acceleration. The acceleration field is the same in both cases, while the streamline patterns, observed from different inertial systems, differ (color plate C. 16, page 259).

3.2 Feature Lifetime and Long-Living Flow Structures

In real-world data sets the high density of features often complicates a proper analysis. An appropriate filter mechanism differentiating between important and unimportant structures can ease this problem. Since fluid flow researchers are mainly interested in long-living structures, the lifetime of features is a meaningful filter criterion. Thus, special attention is paid to particles that carry the minimality property of $\|\mathbf{a}\|$ for at least a small period of time. The proposed feature identifier makes use of the 'feature lifetime' in two ways: (i) Considering and averaging the acceleration magnitude along pathlines over a lifetime interval reinforces the Lagrangian perspective of the approach. (ii) An explicit *feature lifetime* filter selecting particles that stay in a feature state a certain time period enables the extraction of long-living features.

4 Proposed Feature Extraction Technique

4.1 Method

The center of a Lagrangian point of view is the behavior of particles, represented by pathlines and the evolution of flow properties along these lines. Each pathline is identified by its initial position $\mathbf{x}_0$ at time t_0 and the corresponding trajectory $\mathbf{s}(t,\mathbf{x}_0,t_0) = \mathbf{s}$ depending on the time parameter t. The contribution of a pathline to a certain feature $\mathscr{F}$ is measured using a *feature importance* $I_{\mathscr{F}}$; it is defined as the average of a scalar *feature identifier* $f(\mathbf{x},t)$ over some feature time span $[t_{\min}(\mathbf{x}_0,t_0), t_{\max}(\mathbf{x}_0,t_0)]$

$$I_{\mathscr{F}}(\mathbf{x}_0,t_0) = \frac{1}{t_{\max}(\mathbf{x}_0,t_0) - t_{\min}(\mathbf{x}_0,t_0)} \int_{t_{\min}(\mathbf{x}_0,t_0)}^{t_{\max}(\mathbf{x}_0,t_0)} f(\mathbf{s}(t,\mathbf{x}_0,t_0),t)^2\, dt. \tag{4}$$

The choice of the parameters $t_{\max}(\mathbf{x}_0,t_0)$ and $t_{\min}(\mathbf{x}_0,t_0)$ is crucial, since they determine the time range of influence to the local value. They are determined by the time a pathline

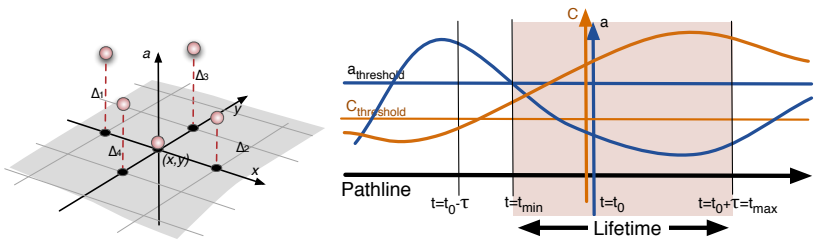


Fig. 4. (a) The minimality of the acceleration can be measured by the Laplacian of the scalar field $\|a\|$, which can be computed by central differences. (b) Definition of a feature's lifetime along a pathline. The parameter $t_{\min}$ is determined by the acceleration threshold and $t_{\max}$ by the maximum lifetime window.

exhibits a certain feature state. Thus, they depend on the feature considered and are derived for each pathline segment. The feature lifetime is defined as

$$T_{\mathscr{F}}(\mathbf{x}_0,t_0) = t_{\max}(\mathbf{x}_0,t_0) - t_{\min}(\mathbf{x}_0,t_0). \tag{5}$$

More specifically, for *Lagrangian equilibrium points* the feature identifier is the acceleration magnitude $a(\mathbf{x},t) = \|\mathbf{a}(\mathbf{x},t)\|$. The lifetime parameters $t_{\min}(\mathbf{x}_0,t_0)$ and $t_{\max}(\mathbf{x}_0,t_0)$ are based on three quantities: acceleration magnitude a, a minimality measure of the acceleration C_{a}, and a maximum lifetime window τ. To measure the minimality the differences of $a(\mathbf{x}_0,t_0)$ at neighboring points are averaged in the four main directions: $C_{\mathrm{a}} = 1/4\sum_{i=1}^{4}\Delta_i$, where $\Delta_i, i = 1,..,4$ are defined in Figure 4(a). $C_{\mathrm{a}} > C_{\mathrm{threshold}}$ indicates that a particle has low acceleration compared to its neighbors.

A maximum lifetime window $[t_0-\tau, t_0+\tau]$ restricts the values of $t_{\max}$ and $t_{\min}$. Since saddle and vortex regions exhibit different characteristic behavior, the parameter τ can be chosen for each of these structures separately. Possible criteria to distinguish saddle and vortex regions can be based on the instantaneous Jacobian matrix and its characteristics, Q, λ_2 or the Lagrangian approach of Haller [18]. In the following the Jacobian matrix is used.

Finally, the lifetime parameters are defined as (cf. Figure 4(b))

$$\begin{aligned} t_{\min}(\mathbf{x}_0,t_0) = \min(t' \in [t_0-\tau,t_0] \mid \forall t \in [t',t_0] : \\ a(\mathbf{s}(t,\mathbf{x}_0,t_0),t) \le a_{\mathrm{threshold}} \text{ and} \\ C_{\mathrm{a}}(\mathbf{s}(t,\mathbf{x}_0,t_0),t_0) \ge C_{\mathrm{threshold}}), \end{aligned} \tag{6}$$

and

$$\begin{aligned} t_{\max}(\mathbf{x}_0,t_0) = \max(t' \in [t_0,t_0+\tau] \mid \forall t \in [t_0,t'] : \\ a(\mathbf{s}(t,\mathbf{x}_0,t_0),t) \le a_{\mathrm{threshold}} \text{ and} \\ C_{\mathrm{a}}(\mathbf{s}(t,\mathbf{x}_0,t_0),t_0) \ge C_{\mathrm{threshold}}), \end{aligned} \tag{7}$$

If one of the criteria is not fulfilled at particle position $\mathbf{x}_0$ and time t_0, the feature lifetime is defined as zero and $t_{\max}(\mathbf{x}_0,t_0) = t_{\min}(\mathbf{x}_0,t_0) = t_0$; furthermore, the acceleration is not

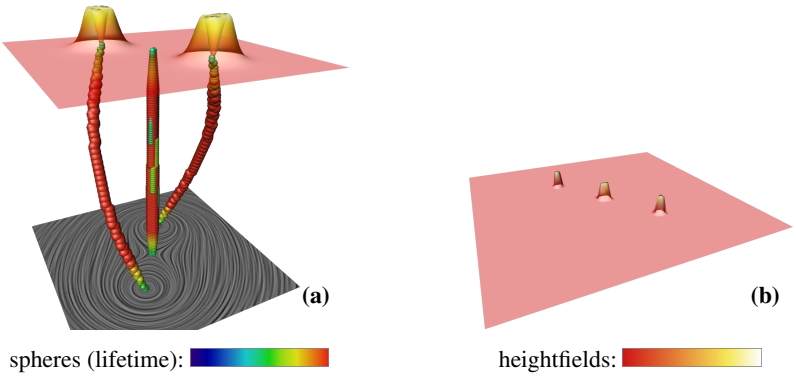


Fig. 5. Illustration of the *Lagrangian equilibrium point* concept for two co-rotating Oseen vortices. The color-coded heightfield represents: (a) integrated acceleration, (b) and lifetime for the last time step (color plate C. 17, page 260).

averaged over the lifetime and the resulting importance value is set to the square of the local acceleration magnitude.

Subsequently, feature candidates are extracted by searching minima in the resulting scalar importance field $I_{\mathscr{F}}$. Finally, a filtering with the extracted lifetime distinguishes important and unimportant features.

4.2 Implementation

The input to the algorithm is a 2D vector field defined on a sample grid. The algorithm consists of three main steps: integration of the acceleration values, extraction of the minimum points and filtering of these points with the lifetime.

Integration. The first step is the determination of the lifetime parameters, according to Eqs. (5), (6) and (7). A suitable threshold value for a is extracted by analyzing the acceleration characteristics of the first time-slice of the dataset. In this exploratory work it is simply set to ten percent of the maximum value. The threshold C_a has to be set a little above zero, to avoid setting a long lifetime for regions with low acceleration at all. For each discrete point $(\mathbf{x}_0, t_0)$ a backward search in time on the trajectory $\mathbf{s}$ determines $t_{\min}(\mathbf{x}_0, t_0)$. The lifetime criteria at each sample step on $\mathbf{s}$ is tested until either one of the thresholds is violated, or the maximum time window or the domain boundary is reached. Then the feature importance is computed according to Eq. (4). For numerical integration a *Runge-Kutta* integrator RK4(3) with adaptive step-size control is used. After the starting point $t_{\min}$ for the forward integration is found, the accumulation of the acceleration magnitude along $\mathbf{s}$ is started. The integration is terminated if one of the lifetime criteria is not fulfilled. Then the resulting values are normalized by the factor $1/T_{\mathscr{F}}(\mathbf{x}_0, t_0)$ and both lifetime $T_{\mathscr{F}}(\mathbf{x}_0, t_0)$ and importance measure $I_{\mathscr{F}}$ are stored as scalar fields.

Extraction. Feature candidates are extracted by searching local minima in the importance field I using a discrete neighbor analysis. Alternatively, other methods like the watershed transformation [19] could be used for locating local minima.

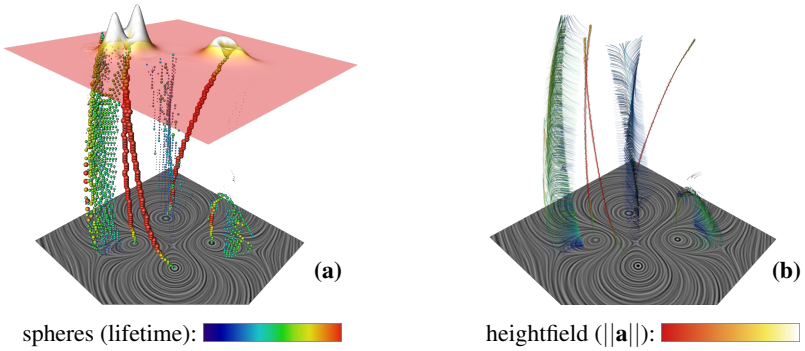


Fig. 6. (a) Extraction of features for a motion of six Oseen vortices. Since particles leave the saddle regions quickly, the saddles do not emerge as prominently as the vortex cores. The integration windows of 0.3 in each direction are too large for particles passing through the saddle in the center. In comparison, all other long-living structures such as the vortex cores are extracted effectively. The heightfield represents the integrated acceleration. (b) Visualization of the feature lifetime. The illuminated pathline segments show the interval of the lifetime used for the integration of the acceleration. Pathlines seeded in vortex-like feature points are long centerlines, while pathlines seeded in saddle-like feature points diverge rapidly (color plate C. 18, page 260).

Filtering. After these initial feature candidates are found, the lifetime filter is applied. Using a threshold for lifetime, it is now possible to emphasize long-living structures. The threshold can be chosen separately for saddles and centers to account for the different lifetime characteristics.

5 Visualization

All results in this paper are visualized in a volume spanned by two spatial coordinates and time. The extracted feature points are illustrated using spheres. The spheres are scaled and colored according to the associated lifetime, choosing a color table where high lifetime values are marked red, see Fig. 5 and 6(a). In some images illuminated pathline segments are seeded in the extracted feature points to get a more intuitive notion of the lifetime. The pathlines are terminated after exceeding their feature lifetime, as shown in Fig. 6(b). Color-coding is the same as for the spheres. The scalar fields used for the feature extraction can be added as heightfield for one time step.

To understand the local flow structure it is helpful to observe not only single pathlines but also the behavior of bundles. Such an exploratory analysis is facilitated by the possibility to select a point of interest in the LIC image. For this location the pathline is displayed together with a moving disk depicting the flow relative to this pathline, see Fig. 2(b).

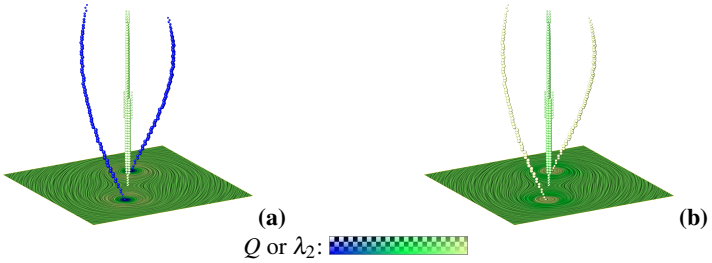


Fig. 7. Comparison of (a) λ_2 and (b) Q with the proposed Lagrangian equilibrium points. The extracted feature points include the vortex cores marked by high values of Q or low values of λ_2. In addition the saddle between the two vortices is detected (color plate C. 19, page 260).

6 Results

To evaluate its effectiveness, the proposed method has been applied to two different datasets. The first dataset represents a pair of co-rotating Oseen vortices, see Fig. 5. The Oseen vortex models a line vortex that decays due to viscosity. The velocity V_Θ in the circumferential direction θ is given by

$$V_\Theta(r) = \frac{\Gamma}{2\pi}\frac{1-e^{-(\frac{r}{r_c})^2}}{r},$$

where r is the current radius, r_c the core radius and Γ the circulation contained in the vortex. For further information we refer to Rom-Kedar et al. [20] or Noack et al. [21]. A more complicated flow field is generated by the interplay of six moving Oseen vortices, see Fig. 6. Both datasets are given for a temporal bounding box of $[-1.0, 1.0]$. The maximum lifetime window τ for integration is set to 0.3 in each direction. The extraction process takes a couple of minutes on standard hardware using non-optimized software.

For both datasets it can be seen in Fig. 5(a) and Fig. 6(a) that the integrated acceleration is low at vortex centers and saddle points. While the lifetime is high for all features of interest in the co-rotating case, the lifetime only marks centers clearly in the complicated field. The structures extracted are displayed as spheres, using the lifetime to define color and size. Due to the finite time window the lifetime is low at the beginning, grows and then drops off to the end. After applying the lifetime filter vortices are marked as important, but interesting saddles are also removed. This is a consequence of the fact that particles stay longer in the vicinity of centers than in the vicinity of saddles. The illuminated pathline segments in Fig. 6(b) indicate the interval used for the integration. Pathlines seeded in vortex-like features form long centerlines due to the strong rotation within the vortex. In contrast, pathline segments seeded in saddle-like structures diverge. This is consistent with the observation that interesting saddles are removed.

Since vortex cores are extracted, Fig. 7 shows a comparison with standard vortex indicators such as λ_2 and Q. The values of λ_2 and Q are color-coded in the LIC texture and in the spheres. The minima of λ_2 or the maxima of Q reveal nearly the same structures as our approach. With λ_2 or Q, however, separating structures such as saddles cannot be extracted.

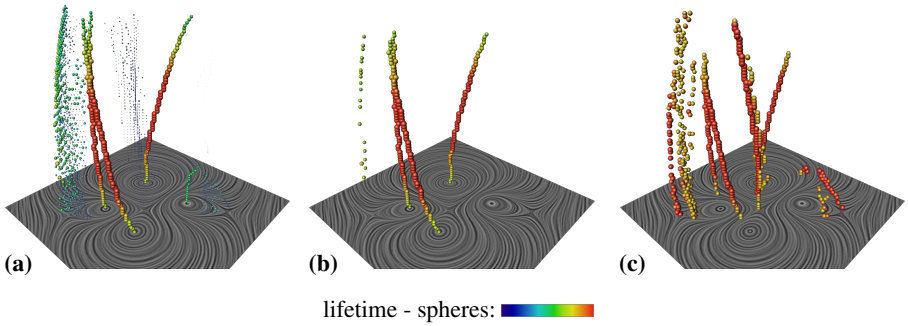


Fig. 8. Motion of six Oseen vortices: (a) All features before filtering; (b) Applying the lifetime filter filters the short-living out. (c) Employing a shorter life-time window also saddles are extracted (color plate C. 20, page 261).

The ability of our approach to filter out short-living features is illustrated in Fig. 8. The chosen time window determines the maximal lifetime. All features with higher lifetime cannot be differentiated. Choosing a time window of length 1.0 only three long-living vortices remain.

Distinguishing between saddle and vortex regions by using the Jacobian leads to the results depicted in Fig. 8(c). In the example the lifetime for vortex-like regions is 0.6 and for saddle-like regions 0.1. With this differentiation all salient features including saddles are visible.

7 Conclusions

The proposed method enables a Galilean invariant extraction of long-living structures, based on the concept of Lagrangian equilibrium points. The method features the following characteristics, which demonstrate that the concept is a first step to overcome the limitations of standard vector field topology. The Lagrangian viewpoint helps to analyze time-dependent structures. The filtering of the extracted points by lifetime enables to mark salient structures. It provides a generalization of critical points of standard vector field topology since fixed points are also Lagrangian equilibrium points for the steady case.

In the current state, the method is still based upon two major thresholds $a_{\text{threshold}}$ and τ. While the first parameter determines whether a particle carries a feature, the second parameter represents a characteristic feature lifetime and depends on the scale of feature lifetimes in the given dataset. Currently, the acceleration threshold is chosen heuristically. We leave it to future work to identify the relevant scale.

The application to datasets that are well understood proves the effectiveness of the proposed extraction scheme for salient structures in time-dependent flow fields. The next steps will be to improve the efficiency of the algorithm, to apply it to real world datasets, which also includes three-dimensional data, and to find ways to select appropriate parameter values automatically.

Acknowledgments

The project is part of the SFB 557 "Control of complex turbulent shear flows" and is partially supported by the DFG Emmy Noether program. The authors wish to thank George Haller, Gilead Tadmor, and Igor Mezić for fruitful discussions. All visualizations have been created using Amira - a system for advanced visual data analysis (http://amira.zib.de). The authors further want to thank the reviewers for their suggestions, which helped to improve the paper significantly.

Appendix

A transformation from one inertial frame F to another F', moving with a relative velocity $\mathbf{u}$, i.e., a Galilean transformation, is defined in space-time by $x \equiv (t, \mathbf{x}) \Rightarrow x' \equiv (t', \mathbf{x}') = (t, \mathbf{x} - \mathbf{u}t)$. The corresponding transformation for the four-velocity then is $v \equiv (1, \mathbf{v}) \Rightarrow v' \equiv (1, \mathbf{v}') = (1, \mathbf{v} - \mathbf{u})$. The differential operators $\partial_t \equiv \partial/\partial t$ and $\partial_k \equiv \partial/\partial x_k$ ($\partial_{k'} \equiv \partial/\partial x'_k$) are transformed according to $\partial_t \Rightarrow \partial_{t'} - (\mathbf{u} \cdot \nabla')$, where $\nabla \equiv (\partial_1, \partial_2, \partial_3)$ and $\nabla' \equiv (\partial_{1'}, \partial_{2'}, \partial_{3'})$ From this it is easy to see that the material derivative $D_t \equiv \partial_t + (\mathbf{v} \cdot \nabla)$ is invariant under Galilean transformations:

$$\partial_t + (\mathbf{v} \cdot \nabla) = \partial_{t'} + (\mathbf{v}' \cdot \nabla').$$

A specific consequence is that the total acceleration a flow particle experiences is Galilean invariant, $\mathbf{a} \equiv D_t \mathbf{v} = D_{t'} \mathbf{v}' \equiv \mathbf{a}'$, i.e. independent of the frame of reference.

References

1. Panton, R.L.: Incompressible Flow. Wiley & Sons (2005)
2. Tobak, M., Peake, D.: Topology of three-dimensional separated flows. Ann. Review of Fluid Mechanics **14** (1982) 61–85
3. Haller, G.: Lagrangian coherent structures from approximate velocity data. Physics of Fluids **14**(6) (2002) 1851–1861
4. Helman, J., Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. Computer **22**(8) (1989) 27–36
5. Helman, J., Hesselink, L.: Visualizing vector field topology in fluid flows. IEEE Comput. Graph. Appl. **11**(3) (1991) 36–46
6. Laramee, R., Hauser, H., Zhao, L., Post, F.: Topology-based flow visualization, the state of the art. In Hauser, H., Hagen, H., Theisel, H., eds.: Topology-based Methods in Visualization, Springer, Berlin (2007) 1–19
7. Hunt, J.: Vorticity and vortex dynamics in complex turbulent flows. CSME Trans. **11**(1) (1987) 21–35
8. Jeong, J., Hussain, F.: On the identification of a vortex. Journal of Fluid Mechanics **285** (1995) 69–94
9. Sadarjoen, I., Post, F.: Detection, quantification, and tracking of vortices using streamline geometry. Comput. Graph. **24**(3) (2000) 333–341
10. Peikert, R., Roth, M.: The parallel vectors operator - a vector field visualization primitive. In: IEEE Visualization '00. (2000) 263–270

11. Theisel, H., Weinkauf, T., Hege, H.C., Seidel, H.P.: Topological methods for 2D time-dependent vector fields based on stream lines and pathlines. IEEE Trans. Vis. Comput. Graph. **11**(4) (2005) 383–394
12. Weinkauf, T., Sahner, J., Theisel, H., Hege, H.C.: Cores of swirling particle motion in unsteady flows. IEEE Trans. Vis. Comput. Graph. **13**(6) (2007) 1759–1766
13. Fuchs, R., Peikert, R., Sadlo, F., Alsallakh, B., Gröller, E.: Delocalized unsteady vortex region detectors. In O. Deussen, D. Keim, D.S., ed.: VMV '08. (October 2008) 81–90
14. Shi, K., Theisel, H., Weinkauf, T., Hege, H.C., Seidel, H.P.: Finite-time transport structures of flow fields. In: IEEE Pacific Visualization '08. (2008) 63–70
15. Haller, G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. Physica D **149** (2001) 248–277
16. Sadlo, F., Peikert, R.: Efficient visualization of lagrangian coherent structures by filtered AMR ridge extraction. IEEE Trans. Vis. Comput. Graph. **13**(6) (2007) 1456–1463
17. Garth, C., Gerhardt, F., Tricoche, X., Hagen, H.: Efficient computation and visualization of coherent structures in fluid flow applications. IEEE Trans. Vis. Comput. Graph. **13**(6) (2007) 1464–1471
18. Haller, G.: Lagrangian structures and the rate of strain in a partition of two-dimensional turbulence. Physics of Fluids **13** (2001) 3365–3385
19. Soille, P.: Morphological image analysis. Springer Berlin (1999)
20. Rom-Kedar, V., Leonard, A., Wiggins, S.: An analytical study of transport, mixing and chaos in an unsteady vortical flow. Journal of Fluid Mechanics **214** (1990) 347–394
21. Noack, B., Mezić, I., Tadmor, G., Banaszuk, A.: Optimal mixing in recirculation zones. Physics of Fluids **16**(4) (2004) 867–888

Stream Volume Segmentation of Grid-Less Flow Simulation

Harald Obermaier[1], Jörg Kuhnert[2], Martin Hering-Bertram[2], and Hans Hagen[1]

[1] University of Kaiserslautern, Germany
[2] Fraunhofer ITWM Kaiserslautern

Abstract. We present a novel algorithm for the geometric extraction of stream volume segmentation for visualization of grid-less flow simulations. Our goal is the segmentation of different paths through a mixing tube where the flow is represented by scattered point sets approximated with moving least squares. The key challenges are the watertight construction of boundary representations from separatrices. These are obtained by integrating and intersecting stream surfaces starting at separation and attachment lines at boundaries of flow obstacles. A major challenge is the robust integration of stream lines at boundaries with no-slip condition such that closed volume segments are obtained. Our results show the segmentation of volumes taking consistent paths through a mixing tube with six partitioning blades. Slicing these volumes provides valuable insight into the quality of the mixing process.

1 Motivation

Topological features of flow fields such as separatrices are of specific interest in vector field visualization, since they partition the domain into connected segments with consistent limit behavior of stream lines, classifying (stationary) flow regions based on their sources and targets. In planar vector fields, critical points often determine these flow targets besides cycles and boundary segments [5]. In three-dimensional incompressible flow critical points are rather rare. Sinks and sources do not exist, due to conservation of mass, and only saddles of different types may occur. Thus, separatrices mostly emerge from separation and attachment lines on boundary surfaces from where they may be integrated with sophisticated stream surface extraction.

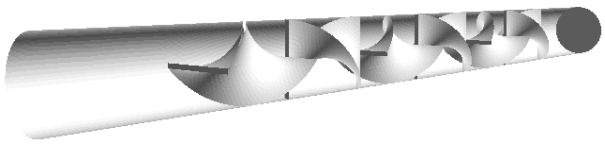


Fig. 1. Cylindrical mixing tube with six blades.

In the present work we contribute a novel algorithm for the extraction of watertight boundary representations for stream volumes. The algorithm is used to analyze a three-dimensional vector field describing a mixing process simulated by the *Finite Pointset Method* (FPM) [14]. The underlying application is concerned with flow of liquid glass at high temperature through a cylindrical mixing tube (see Fig. 1). Six twisted blades located sequentially inside the tube partition the flow into 64 different paths, each associated with a stream volume composed of all stream lines taking the same path. Both, the granularity and the geometric shape of these separating structures are indicators of the quality of the mixing process, i.e. maintaining homogeneous optical properties of the glass before it is cast into its final shape.
Our application data set describes a stationary velocity field of viscous incompressible flow, represented by a finite point set with associated field attributes. Due to the absence of a computational grid, the data needs to be approximated with a local method like *Moving Least Squares* (MLS) which is also used in the simulation code. A major challenge is the integration of stream lines along surfaces with no-slip boundary, since the approximated vector field may not get exactly zero on the boundary and may even reverse its direction. To obtain a valid segmentation, stream lines on boundaries are projected on the surface and released on intersection with a separation line. The framework composed of such separating boundary stream lines, separation, and attachment lines connects the material boundaries to the complex of inner separatrices that need to be constructed carefully by adaptive integration due to varying complexity of the vector field.
We present a novel algorithm for the adaptive construction of watertight stream volumes by mutual intersection of separatrices in three-dimensional flow, used for quality analysis of mixing processes. These are the main challenges arising during stream-volume construction:

- **No neighborhood relation between data points.** The vector field is approximated by a mesh-free approximation technique called "Moving Least Squares". For each evaluation, a local point set is defined using weighting and visibility queries.
- **No-slip boundary and flow obstacles.** To create watertight stream volumes, one needs to be capable of integrating stream lines along geometry. Additionally, flow obstacles have an impact on the construction of stream surfaces due to their splitting behavior.
- **Intersection of separatrices.** Intersected stream surfaces generated by separation or attachment lines on flow obstacles yield parts of separatrices, that are recombined to define boundaries of stream volumes.

In Sect. 2 we summarize fundamentals and refer to related work that has been done in the field of vector field approximation and stream surface construction. Sect. 3 describes our approaches to stream volume construction and visualization. We provide numerical examples of stream volume visualizations in Sect. 4, which as well contains an analysis of the test data set and an outlook on future work.

2 Fundamentals and Related Work

2.1 Moving Least Squares Approximation

Let S be a field of scattered points $x_i \in \Omega \subseteq \mathbb{R}^n$ with function values $f_i \in \Pi \subseteq \mathbb{R}^m$. A method suitable to approximate such grid-less data is the MLS approach [8,9]. MLS is a weighted, local generalization of the well-known "Least Squares" technique, which fits a polynomial of given degree to a set of points while minimizing the squared distance to corresponding field values.
For a large set of points, it is not feasible to find a globally defined polynomial f that still provides a fairly small minimum overall error, as obtained by a standard Least Squares approximation. To construct a locally supported function $f_{x'}$ for arbitrary fixed $x' \in \Omega$, the standard LS equation is altered by introducing a compactly supported weighing function. The approximating polynomial $f_{x'}$ must then satisfy (1).

$$\sum_i \omega(x', x_i)\, ||f_i - f_{x'}||^2 \rightarrow \min \quad \text{with } \omega(x', x_i) = e^{-\alpha \frac{||x'-x_i||^2}{r^2}} - e^{-\alpha} \tag{1}$$

where r is given by the simulation and defines the radius of influence or "smoothing length". Only points whose distance to x' does not exceed r are used for evaluation. MLS uses (1) to construct an approximating function f by moving x' over the domain of S. Therefore f is not defined by a single fixed $x' \in \Omega$, but needs to be evaluated at every $x' = x \in \Omega$ separately. This construction creates f as a composition of multiple $f_{x'}$. Let $m = 1$ and $f_{x'}$ be a polynomial of the general form $f_{x'}(x) = a_{x'}^T \cdot v(x)$ with $a_{x'}$ an unknown vector of coefficients and $v(x)$ a given polynomial base vector of degree d. For $n = 2$, $m = 1$, and $d = 1$, (2) needs to be solved for a to obtain an approximating polynomial.

$$\sum_i \omega(\mathbf{x}', \mathbf{x_i}) \begin{pmatrix} 1 & x & y \\ x & x^2 & xy \\ y & xy & y^2 \end{pmatrix} a = \sum_i \omega(\mathbf{x}', \mathbf{x_i}) \begin{pmatrix} 1 \\ x \\ y \end{pmatrix} f_i \tag{2}$$

2.2 Line, Surface, and Volume Integration

Stream lines provide a simple way of visualizing particle traces in stationary fields [12]. Their computational complexity depends on the method used for vector field evaluation as well as on the integration method and adaptivity scheme. When integrating stream lines, appropriate measures have to be taken to guarantee a given accuracy. We use an embedded Runge-Kutta integration scheme of 4th/5th order [10] to generate adaptive stream lines. Hereby a comparison of the two different results with respect to angular deviation is used to scale the step size for a fifth order Runge-Kutta integration.
Stream surfaces represent a generalization of the uni-variate stream lines. They are of special importance to the analysis of mixing processes as they provide the basis for constructing three-dimensional separatrices. The introduction of stream surfaces necessitates new concepts of adaptivity. Such a concept was presented by Hultquist et al. [6] and has been refined in the context of grid-based data sets by different authors such

as Scheuermann et al. [11] and Garth et al. [4]. The underlying stream surface generation algorithm used in this paper is based on an extended version of this ribbon-based approach. Stream surfaces generated by methods discussed in this paper represent separating structures as proposed by Wiebel et al. [15].
Traditional approaches of stream volume generation use a closed polygon as rake for stream surfaces and are hardly more than an adapted version of the stream surface algorithm introduced by Hultquist. However, there are more sophisticated methods based on scalar field generation [16] or tetrahedral volumes [2]. This work introduces a novel surface-based algorithm to create watertight stream volumes from parts of separatrices, making knowledge about the starting and ending position of the volume unnecessary.

3 Algorithm

3.1 Outline

These are the basic steps of our algorithm:

1. Compute separation and attachment lines on flow obstacles.
2. Generate three-dimensional separatrices by construction of stream surfaces starting at separation and attachment lines (forwards and backwards, resp.).
3. Intersect separatrices and split them into multiple surface segments.
4. Compose stream volumes of parts of separatrices.

In the following sections, these steps are explained in detail.

3.2 Methods

Computation of Separation Lines. Separation and attachment lines define locations in the two-dimensional projection of the vector field onto the boundary object where flow separates from an object, or attaches to it. In the following *separation line* will be used to denote both types.
For the given mixing simulation we assume, that all significant separation lines are located on the triangulation of obstacles. Such triangles separate incoming flow in one of two ways:

Points with Flow Parallel to Eigenvectors. A method proposed by Kenwright et al. [7] finds points of separation lines on edges of triangles. This is done by eigenvector analysis of the Jacobian of the two-dimensional projected vector field. Points on a separation line are classified by flow parallel to the direction of one of the Jacobian's real eigenvectors. If this eigenvector corresponds to the smallest real eigenvalue, the point is on an attachment line. If it corresponds to the greatest eigenvalue, a point on a separation line was found.

Separating Edges. Convex edges between triangles of obstacles may have separating properties. The two variables listed in the following determine, whether a point p on the edge between two adjacent triangles Δ_i and Δ_j separates the flow.

$$d_1 = ((\Delta_i.p_1 - p) \times \Delta_i.n) \cdot f(p)$$
$$d_2 = ((\Delta_j.p_1 - p) \times \Delta_j.n) \cdot f(p)$$

If d_1 and d_2 do have the same sign as illustrated by Fig. 2, point p has the desired separating properties. This examination allows classification of edges by the separating behavior of their adjacent corner vectors.

The separation lines of our test data are located near the sharp edges of the mix-

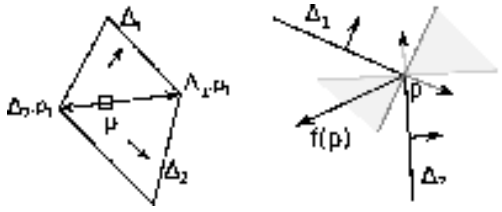


Fig. 2. Flow separation at convex edge. Regions, where d_1 and d_2 have matching signs are marked in gray.

ing blades. These lines will be used as rakes for stream surfaces representing three-dimensional separatrices. As our data does not include saddles, cycles or conventional sinks and sources, separatrices originate from separation lines only.

Integration of Stream Lines. Stream line integration to construct surfaces in grid-less flow simulations yields certain challenges:

Visibility of Data Points. To avoid inclusion of points during MLS approximation of the vector field that are not visible from the position of evaluation, a visibility check needs to be implemented. Grid-based approaches to vector field visualization avoid such calculations, as the boundary is integrated into computational meshes. Figure 3 shows the effects of visibility on line integration. To check for a visibility block between points x and x', we find intersections of the line x-x' with triangles of the boundary. We restrict triangle search to a local neighborhood by insertion of triangles into a regular grid, thus reducing performance impact on distant inner lines to virtually zero.

Integration along Boundary Geometry. Stream lines may start on geometry or cause intersections with it due to numerical inaccuracies of the MLS approximation. Both cases require the ability to integrate stream lines on geometry.

To guarantee watertight surface construction, stream lines that are integrated on the boundary object are forced to stay on geometry until they meet with a separation line. A basic approach to stream line integration on triangulated geometry assumes, that

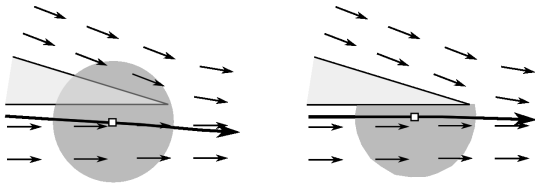


Fig. 3. Impact of visibility on stream lines.

every position on a stream line is located on exactly one triangle t. A new triangle is entered as soon as the stream line leaves its current triangle i.e. the stream line crosses one of the edges of t. The appropriate neighboring triangle is chosen as new plane of projection. While this simple method works in most common cases, it however ignores some of the more complex situations illustrated in Fig. 4. Stream line integration on

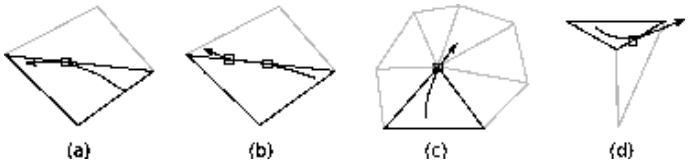


Fig. 4. Situations a) and b) show tangential behavior at triangle edges. In situation c), a stream line crosses one of the corners of a triangle, continuing on only one of multiple neighboring triangles. Situation d) depicts a stream line leaving its current triangle, being released on a convex edge. There are numerous similar situations, where the simple approach mentioned above falls short and oscillation might occur.

triangles of two-dimensional linear vector fields has been presented before by Battke et al. [1] in 1997. Our projective approach with focus on edge cases is presented in the following. Let p be the position of a stream line on geometry, with a list $T = \{t_i\}$ of triangles directly adjacent to this position. This method repeats the following steps:

1. **Evaluation of the vector field.** The field is evaluated at point p, being located on at least one triangle. Due to the approximating properties of MLS, the no-slip condition is weakened and the field is therefore not evaluated to zero at its boundary, but is aligned with neighboring velocity vectors.
2. **Projection.** The resulting vector v is projected onto all triangles $t_i \in T$.
3. **Choice of Next Point and Determining Triangles.** There are three possible cases. Either p is located on exactly one triangle, on an edge between two triangles, or on multiple triangles. p is advanced to $p + v_i$ for the appropriate triangle t_i or the intersection of this line with any of the edges of t_i and appended to the stream line. T is updated accordingly.

Degree reduction. If a stream line reaches a point where the number of neighboring data points is not sufficient to provide a uniquely solvable LSE for the evaluation of MLS,

the polynomial degree of the approximating polynomial is reduced step by step, as lower degrees require fewer non-coplanar data points. These situations occur especially close to boundaries of the data set. When integrating on or near geometry, extrapolation may flip the vector field's orientation, which can be avoided by decreasing the degree of integration to zero. With geometric integration being unsteady and restricted to a low order due to the low number of available points in the neighborhood, this represents a valid and fast alternative to complex point-in-volume checks.

Construction of Stream Surfaces. One special case that arises when constructing stream surfaces in fields with obstacles is the event of surface splitting. If neighboring stream lines diverge to different sides of an obstacle, triangulation of this stream line pair is canceled. This results in two different fronts of the stream surface. To take advantage of caching strategies, those two fronts are advanced separately and sequentially (see Fig. 5). If at least one of the two affected stream lines is not integrated on the boundary object, a new stream line is inserted on the boundary geometry to maintain a closed representation of the stream surface.
For volume creation, stream surfaces are generated at every separation line of the data set. Resulting separatrices are combined to stream volumes as explained in the following.

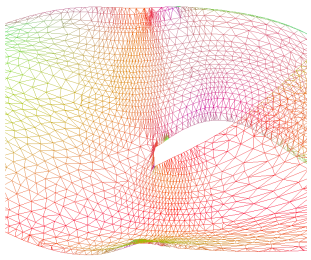


Fig. 5. Obstacle splitting a stream surface. Resulting fronts are advanced separately. The mesh is colored according to normal directions (Color plate C. 21, page 261).

Intersecting Separatrices. An overall look at the division of space is provided by the generation of separatrices. To observe one distinct volume at a time, these separatrices are split and reorganized to form closed boundaries of stream volumes. As illustrated in Fig. 6, in directed vector fields such as the mixing process considered in this work, intersections resulting in the splitting of stream surfaces can be of two different types:
Intersections caused by separatrices of opposing directions are **crossings** of separatrices, which cannot occur between separatrices of identical direction[3], as stream surfaces are not able to cross each other if their rakes do not.
The remaining intersections are **T-intersections** rather than x-crossings. They occur whenever an outer stream line of a separatrix, that is integrated along the boundary geometry meets with the rake of a separatrix of the same direction.

[3] backwards or forwards in the vector field

Surface Crossings. If two surfaces are crossing, so do their triangulations. Hence it is possible to operate on their triangulations when determining cuts of surfaces, rather than generating and inserting new stream lines into both surfaces to represent the cut. While the former method is less accurate than the latter, intersection of triangulations is computationally far less expensive than according stream line integration and insertion into the complex structure of stream surfaces. In this paper, former method was chosen to create cuts between surfaces of opposing directions. As soon as the triangle-based intersection is computed and affected surfaces are retriangulated, they are split into multiple parts along the trace of their cut. Surface crossings generally divide two surfaces into a total of four parts (see Fig. 7).

T-Intersections The second type of cut is produced between two surfaces of the same direction, where the left- or rightmost stream line s_{outer} of a surface S_1 leaves geometry at the rake of another surface S_2, whose rake represents a separation line.
This type of intersection cannot be handled by a straightforward cutting of the two triangulations. One reason for this is that the roots of S_2 do not yield a one-to-one representation of the original separation line due to rake discretization during the process of surface creation. As the starting points of s_{outer} and stream lines of S_2 usually differ, it is impossible to guarantee a continuous intersection.
An advantage over the crossing situation is the knowledge about the meeting point of s_{outer} with the rake of S_2. This point is identical to the position, where s_{outer} leaves geometry. Insertion of a stream line starting at this point that directly represents the cutting trace on S_2 becomes feasible in this case. This newly generated stream line keeps all data about the surface intersection that is needed for retriangulation and splitting of the affected surface.

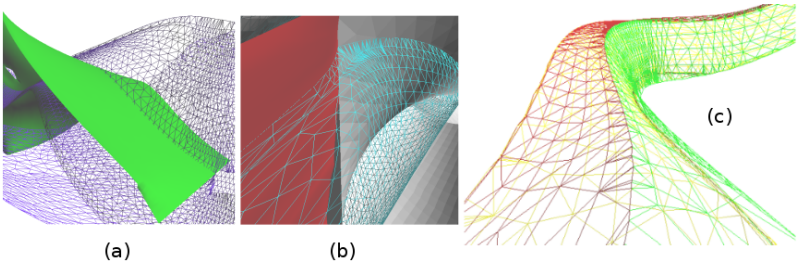


Fig. 6. Crossing behavior of stream surfaces (a), and t-intersection (b) between outer stream line and red surface. Stream volume composed of parts of three separatrices (c). Retriangulation during surface splitting guarantees a closed volume (color plate C. 22, page 261).

Composition of Stream Volumes. Previous work on stream volumes is not suitable to create the volumes desired in the context of this work, as no data about the starting or ending regions of volumes is available. Therefore a new approach of volume composition is introduced in the following.

Surface parts that originate from a common intersection yield a certain relation. As shown in Fig. 7, this relationship is governed by the normals of two adjacent triangles that are part of the intersection. These normals allow classification of the newly created surfaces (in this case four) by orientation of their normals, resulting in a neighborhood structure. A back-back neighborhood between two surfaces s_1 and s_2 with triangles Δ_1 and Δ_2 is for example indicated by:

$$d_1 = \Delta_1.n \cdot (\Delta_2.o - \Delta_1.o) < 0$$
$$d_2 = \Delta_2.n \cdot (\Delta_2.o - \Delta_1.o) > 0$$

Where $\Delta_1.o$ and $\Delta_2.o$ denote triangle centers. Assembling of a stream volume follows these steps:

1. Choose an arbitrary stream surface.
2. Choose whether the surface faces the inside or the outside of the volume to be generated. Therefore indicating, whether the normal points into the volume or not.
3. Save a list of all relevant intersections of this surface.
4. Add all surfaces to the volume that share any of the relevant stored intersections and face in the correct direction as defined by d_1 and d_2. Intersections of the new surface are processed and added to the list of intersections.
5. Repeat steps 3-4 until no more surfaces are added.

After all five steps are completed, every surface contributing to the volume is assigned at most two volume indices.

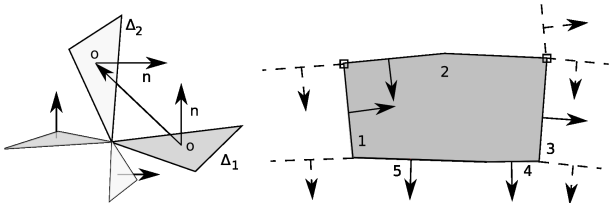


Fig. 7. Neighboring triangles of different surfaces (left) and composition with intersected surfaces (right).

Constructing Slicing Planes. Slicing planes are traditionally used in volume visualization of scalar data sets. Such textured planes cut through a data set and are colored by the scalar values associated with the data. A visualization of the volume is obtained by placing parallel, transparent planes throughout the data set. A similar method is used in this work to avoid occlusion when visualizing stream volumes. The boundary and surface intersections with slicing planes are found and projected onto the plane. While the boundary object of the test data set may produce multiple outlines on the plane,

intersections between the slicing plane and volumes consist of cuts with all surfaces of a certain stream volume, resulting in closed area representations.
During visualization slicing planes are rendered as RGBA textured quads with outlines and cuts mapped and plotted using Bresenham's line algorithm [3]. This produces a slicing plane displaying all intersections with geometry. To avoid ambiguity, the interior area of a volume cut is colored in a distinct color.

4 Results

4.1 Application

As expected due to the number of obstacles, $2^6 = 64$ volumes are created in our data set. These volumes consist of a total of 264 surfaces, implying that every separatrix was on average divided into 12 parts.
Figure 9 depicts visualizations of a single stream volume and the complete set of 64 volumes. Inspection of multiple volumes, as depicted in Fig. 8 provides more general information on the segmentation of flow. Figure 10 displays slicing planes through volumes of the data set. Figure 11 illustrates the use of transparent slicing planes for volume visualization. All slicing planes are aligned perpendicular to the main direction of flow.

4.2 Discussion

Stream volumes provide a general overview of the quality of a mixing process, as their form and start/end positions contain information about sets of particle traces. Slicing planes simplify analysis by extracting local information about stream volumes. This way multiple observations can be made using the visualization techniques presented in this paper: The simulation has a symmetrical rotating behavior, incoming groups of stream lines are rotated by at least 90°. The cuts in Fig. 10 reveal a shuffling or squeezing motion caused by the vector field, resulting in volume deformations. Additionally, a comparison between cuts through either end of the simulation suggests a mixing property, as neighboring stream volumes describe differing paths. These results suggest, that the mixing process is of good quality. As is seen by analysis of flow obstacles, the mixing character is directly influenced by the number of obstacles. The desired number of obstacles depends on the concrete application of the mixing process.

Future work in this direction may include parallelization of intersection operations, analysis of volume divergence and extension of the work to non-stationary fields.

Acknowledgments This work was supported by the German Research Foundation (IRTG 1131) and by the Center for Mathematical and Computational Modeling $(CM)^2$.

References

1. H. Battke, D. Stalling, H-C. Hege: *Fast line integral convolution for arbitrary surfaces in 3D.* Visualization and mathematics: experiments, simulations and environments, 1997, 181–ff

Fig. 8. Visualization of several stream volumes. A stretching, rotating and diverging motion between adjacent volumes can be observed (color plate C. 23, page 262).

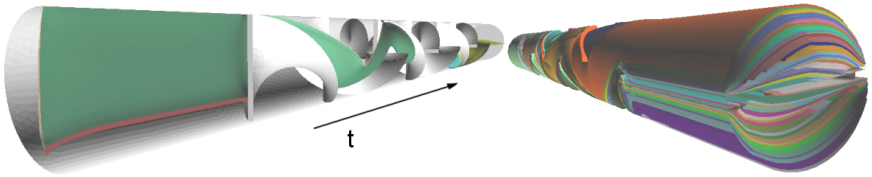


Fig. 9. Visualizations of a single twisted and stretched volume and all volumes of the data set. Colors identify adjacent stream volumes. Occlusion of volumes hides important information about the mixing process, which can be visualized by slicing (color plate C. 24, page 262).

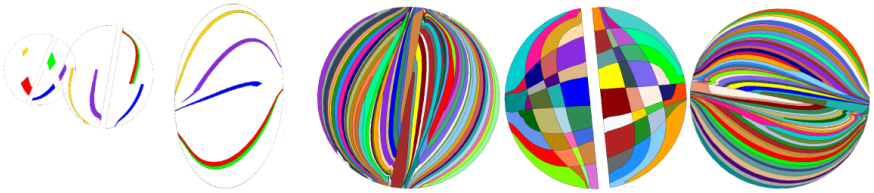


Fig. 10. Slicing planes through five (left) and all volumes of the data set (right) at positions $t = 0.3, 0.6, 1.0, 0, 0.5, 1.0$. It is clearly visible, how neighboring volumes take different paths through the data set. Comparison between individual slices allows observation of stretching, rotational, and mixing behavior. The distribution of volume colors indicate a good mixing process (color plate C. 25, page 262).

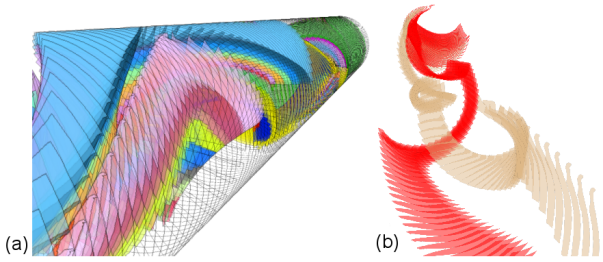


Fig. 11. Volume visualization of a selection of stream volumes by slicing planes (a) and volume visualization of two stream volumes (b). This type of visualization lessens the effect of volume occlusion by the use of transparency (color plate C. 26, page 262).

2. B. G. Becker, D. A. Lane, N. L. Max: *Unsteady Flow Volumes.* Proceedings of Visualization 1995, 329
3. J. E. Bresenham: *Algorithm for Computer Control of a Digital Plotter.* IBM Systems Journal, Vol. 4, No. 1, 25–30 (1965)
4. C. Garth, H. Krishnan, X. Tricoche, T. Bobach, K. I. Joy. *Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields.* IEEE TVCG, Vol. 14, No. 6, 1404–1411 (2008)
5. A. Globus, C. Levit, T. Lasinski: *A Tool for Visualizing the Topology of Three-Dimensional Vector Fields.* NASA Ames Research Center, Computer Sciences Corporation
6. J. P. M. Hultquist: *Constructing stream surfaces in steady 3D vector fields.* Proceedings of Visualization 1992, 173–175
7. D. N. Kenwright, C. Henze, C. Levit: *Feature Extraction of Separation and Attachment Lines.* IEEE TVCG 5, 2 1999, 135–144
8. D. Levin: *The Approximation Power of Moving Least-Squares.* Mathematics of Computation, Vol. 67, No. 224. October 1998, 1517–1531
9. A. Nealen: *An As-Short-As-Possible Introduction to the Least Squares, Weighted Least Squares and Moving Least Squares Methods for Scattered Data Approximation and Interpolation.* Technical Report, Discrete Geometric Modeling Group, TU Darmstadt
10. W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press. 1992.
11. G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, K. I. Joy, W. Kollmann: *A tetrahedra-based stream surface algorithm.* Proceedings of IEEE Visualization 2001, 151–158
12. S. U. Sikorski, C. K. Ma: *Efficient Streamline, Streamribbon, and Streamtube Constructions on Unstructured Grids.* IEEE TVCG, Volume 2, No. 2, 100–110 (1996)
13. M. Spiegelman: *Myths and Methods in Modeling.* LDEO, Columbia University. chapter 4 (2000)
14. S. Tiwari, J. Kuhnert: *A numerical scheme for solving incompressible and low mach number flows by Finite Pointset method.* Technical Report, Fraunhofer ITWM, Kaiserslautern
15. A. Wiebel, X. Tricoche, G. Scheuermann: *Extraction of Separation Manifolds using Topological Structures in Flow Cross Sections.* Topology-Based Methods in Visualization II, Springer (2009)
16. J. J. van Wijk: *Implicit Stream Surfaces.* Proceedings of the 4th Conference on Visualization 1993, 245–252

Substructure Topology Preserving Simplification of Tetrahedral Meshes

Fabien Vivodtzev[1], Georges-Pierre Bonneau[2], Stefanie Hahmann[2], and Hans Hagen[3]

[1] CEA/CESTA (French Atomic Energy Commission)
[2] LJK, University of Grenoble and INRIA
[3] University of Kaiserslautern

Abstract. Interdisciplinary efforts in modeling and simulating phenomena have led to complex multi-physics models involving different physical properties and materials in the same system. Within a 3d domain, substructures of lower dimensions appear at the interface between different materials. Correspondingly, an unstructured tetrahedral mesh used for such a simulation includes 2d and 1d substructures embedded in the vertices, edges and faces of the mesh.
The simplification of such tetrahedral meshes must preserve (1) the geometry and the topology of the 3d domain, (2) the simulated data and (3) the geometry and topology of the embedded substructures. Although intensive research has been conducted on the first two goals, the third objective has received little attention.
This paper focuses on the preservation of the topology of 1d and 2d substructures embedded in an unstructured tetrahedral mesh, during edge collapse simplification. We define these substructures as simplicial sub-complexes of the mesh, which is modeled as an extended simplicial complex. We derive a robust algorithm, based on combinatorial topology results, in order to determine if an edge can be collapsed without changing the topology of both the mesh and all embedded substructures. Based on this algorithm we have developed a system for simplifying scientific datasets defined on irregular tetrahedral meshes with substructures. The implementation of our system is discussed in detail. We demonstrate the power of our system with real world scientific datasets from electromagnetism simulations.

1 Introduction

In this paper we introduce a system that is able to robustly preserve surfaces and polylines defined as substructures in a tetrahedral mesh simplified by repeated edge collapses. This problem originates from an application in electromagnetism, as detailed in the next section. The surfaces (resp. polylines) we are dealing with consist in a subset of faces (resp. edges) of the tetrahedral mesh. Thus, collapsing an edge of the mesh may result in modification of the surfaces and polylines. Preserving these substructures during simplification is a new topic in the literature. There have appeared several papers that have tackled a more specific issue: the preservation of *boundary* surfaces in tetrahedral meshes. The proposed solutions to boundary surface preservation, however, are often too restrictive. Some systems reject any edge collapses in the neighborhood of boundaries, while others do not allow collapses between boundary and internal vertices.

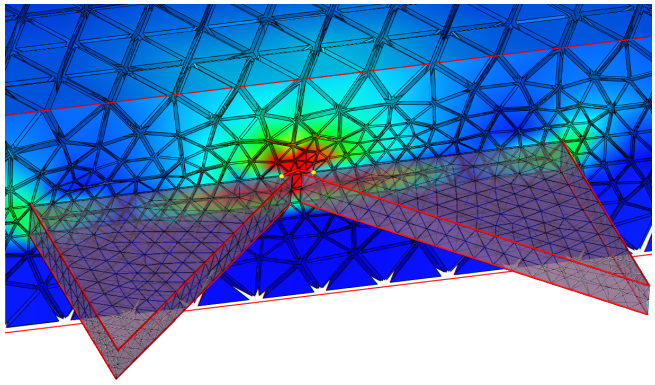


Fig. 1. Electromagnetism simulation for furtivity studies on a large irregular tetrahedral mesh where surface and linear substructures are the heart of the simulation as they influence the field propagation (color plate C. 27, page 263).

In contrast our system is less restrictive: edges may be collapsed anywhere in the mesh, even in the neighborhood of substructures, and even between a vertex in the substructures and a vertex outside the substructures. This leads to higher simplification rates. Furthermore the simplification is *uniformly spread out* across the mesh, regardless of the underlying substructures.

The main contribution of our paper is a robust algorithm to detect if an edge can be collapsed without modifying the topology of the mesh and of its substructures. Our system combines this topological validity test with simple geometric and numeric error measures in order to drive the simplification. However the focus of this paper is clearly on preserving topology – not on preserving the geometry of the mesh or the numerical values attached to it. As a matter of fact, the numerous geometric error measures proposed in other papers can be straightforwardly extended to take into account the geometry of substructures in a mesh, whereas the preservation of the *topology* of the substructures is the real challenge.

In a previous work [14] we introduced a topological test for preserving polylines in non-manifold triangular meshes. Our current work extends the algorithms of [14] to tetrahedral meshes with 2D and 1D substructures. This extension is not trivial, both from a theoretical point of view and for the implementation. The latter is done very carefully in our system. Efficient data structures and algorithms are precisely described. Following [14], the mesh is modeled as a simplicial complex extended by new simplices connecting the substructures to a dummy vertex. The validity of the edge collapse is checked in the extended complex using results from [4]. Thus there is *only one consistent* test to ensure the preservation of the topology of the mesh and of all embedded substructures. In particular we do not rely on heuristic solutions that would treat the mesh and the substructures independently.

The remainder of the paper is structured as follows. Section 1.1 and 1.2 give some additional motivation for this work. Section 2 reviews related work. Section 3 presents the theory and Section 4 describes the implementation in detail. Section 5 presents

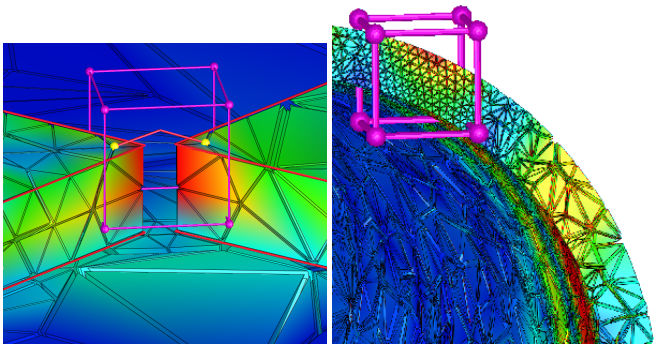


Fig. 2. $(left)$ Detail of a substructure shown at high resolution only inside a volume of interest. The topology of the mesh and the substructures is preserved everywhere. $(right)$ Multiresolution visualization of a thin layer of material with the associated electromagnetic field magnitude (color plate C. 28, page 263).

the integration of the topological test in our visualization system, with examples of application on real world datasets.

1.1 Strength of the Topological Criteria

Geometric error metrics in simplification are coupled with a threshold value that one can adapt to coarsen or refine the data. No matter how good the error computation is, the decision of removal greatly depends on this threshold which is application depend in order to handle unexpected cases often met in Scientific Visualization. In contrast, topological tools guaranty the integrity of the data after simplification without integrating global information or any metric between elements. Combinatorial Topology criteria are general to any application domain that uses meshes.

1.2 Electromagnetic Simulation

Electromagnetism is a wide field used in many applications such as electromagnetic compatibility, furtivity, or the modeling of new absorbing media. Stealth technology relies on the conception and simulation of new absorbing materials in order to decrease the signals reflected from the target to the radar receiver. This ability to reflect radar signals is characterized through Radar Cross Section (RCS) represented as a single scalar. However, to minimize this value, designers need to fully understand the electromagnetic field behavior on the object surface and in its interior. Numerical simulations of this phenomenon often use volumetric finite element methods coupled with a domain decomposition.

The main challenges when simplifying these data for multiresolution visualization come from the complex topology of the embedded structures and the large amount of data. Material boundaries (e.g. interfaces) and various substructures of different dimensions (possibly point sources and polylines) that exist in specific layers introduce

a complex topology. The multiple crossing interfaces can be represented as one non-manifold surface embedded in the volume mesh. The union of all polylines can also be seen as a unique linear structure which intersects the non-manifold surface. Fig. 1 and 3 illustrate such data.

Also, due to the high frequency of the incident electromagnetic field, the mesh needs to be subdivided to the scale of the wave length leading to millions of cells. Because of the thin domain decomposition required by the numerical simulation and the design of specific objects, current tetrahedral meshes have only a few tetrahedra in the thickness of a layer (see Fig. 7) even though the entire mesh may be composed of a hundred of millions of cells.

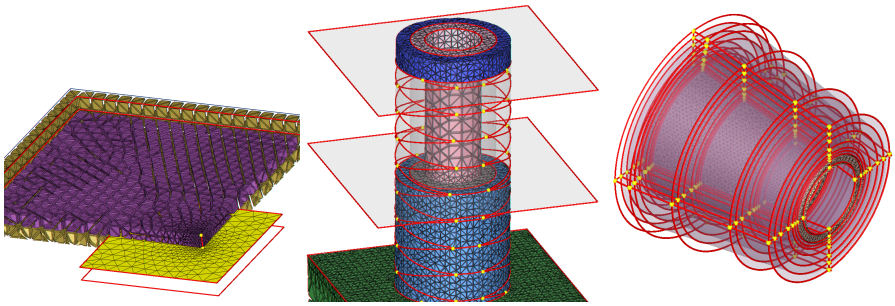


Fig. 3. $(left)$ Polyline emerging from an orthogonal base plan. The mesh is subdivided around this structure. $(center)$ Linear structure with multiple self intersections twisted around a cylindrical mass with orthogonal interfaces. $(right)$ Multiple crossing interfaces of thin material layers and polylines of the structure (color plate C. 30, page 264).

Although we introduced our method on electromagnetism simulations, the exact same topological test can be applied to any domain where the topology of embedded structures needs to be preserved through simplification of the tetrahedral mesh (geology, medical imaging ...).

2 Related Work

Multiresolution approaches provide techniques to efficiently and accurately explore large unstructured polygonal and volume data in Computer Graphics and Scientific Visualization. In order to create a multiresolution representation, many mesh simplification algorithms have been proposed over the last decade ranging from surface to volume based methods. As we focus on topology criteria, we will only survey these criteria used in previous algorithms and omit geometry/attribute error metrics and surface/volume mesh simplification algorithms [2, 10]. In fact any error metrics can be combined with our algorithm without any side effect in order to improve the attribute or geometric quality of the simplified mesh.

Among the wide range of simplification methods, only a few of them introduce specific topological tests and almost none of them tackle the problem of linear/surface

features in a volume mesh. Early work on topological criteria has been done on surface simplification in [13] where the vertices candidate to the removal are classified by analyzing their local neighborhood.

With the notions from computational topology, [8] has proven that the topology of a triangular mesh can be preserved, through simplification, if the 3 tests introduced in the paper are fulfilled. Later, in the PM method [7], these tests are completed with local heuristics in order to preserve the topology of discontinuity curves defined on a surface. The author proposes 6 rules based on an enumeration of the vertices around an edge representing a linear feature.

[9] successfully simplifies non-convex tetrahedral meshes by triangulating the non convex regions. The authors check for tetrahedron flipping avoiding topological and geometrical problems only at the boundary. However, the authors point out that some topological problems can still be undetected with their test during the simplification. All these previous topological criteria are not general enough to preserve the topology of both 1D and 2D structures in a 3D mesh.

[4] is the groundbreaking paper on topology preservation. It has introduced the *link conditions* to robustly check if an edge collapse preserves the topology of a 3-complex. Our paper makes extensive use of these link conditions (reviewed in Section 3).

Intuitively, the *link* of a simplex u is the subset of simplicies *being in contact* with all adjacent simplicies of u and the rest of the mesh. In a 3-complex without boundary, [4] has proven that an edge collapse uv preserves the topology of the complex if the intersection of the vertex links is equal to the edge link. This condition is known as the *link condition*. For general 3-complexes, this link condition has to be evaluated in a succession of complexes built by iteratively filtering the simplicies having a simpler topological neighborhood.

Extending the multiresolution representation of [3], [1] first integrates the link conditions to preserve the topology of both the interior cells and the boundary of the mesh. [12] also uses the link conditions but simplifies them by assuming that the tetrahedral mesh is manifold. The link condition is computed for a 3-complex without boundary for the interior tetrahedra and if a boundary tetrahedron is encountered a second link condition for the 2-complex at the boundary is added.

Because none of these papers fullfil our needs we introduced a new algorithm which successfuly simplifies general simplicical 3-complex in preserving both the topology of this complex and the topology of sub-complexes embedded into it. The strength of this new algorithm comes from: (i) its *locality*, (It only uses the direct neighborhood of the simplicies) (ii) its *unicity*, (It can handle any complicated topological neighborhood regardless of its dimension lower than 3.) and (iii) its *exactness* (It is based on theoretical results taken from combinatorial topology rather than local heuristics).

3 Topology preserving edge collapse in an extended 3-complex

3.1 Link conditions in a 3-complex

The link conditions introduced in [4] can be used in a tetrahedral mesh to test if an edge e between two vertices u and v can be collapsed without modifying the topology of

the mesh. The mesh is modeled as a simplicial complex K. The link conditions are expressed using basic operators from algebraic topology. We assume the reader is familiar with elementary notions in this field. Precise definitions of the link Lk T, the star St T and the closure $\overline{T}$ of a set T of simplices can be found in [10] and illustrated in [14].

If K is a manifold 3-complex, the link conditions reduces to:

$$\text{Lk } u \cap \text{Lk } v = \text{Lk } e \tag{1}$$

If the 3-complex is non-manifold, the condition (1) has to be tested in three successive smaller complexes of dimensions 3, 2 and 1. This is detailed in Section 3.3.

3.2 Extending the mesh with the substructures

In order to use this link condition in our setting, the embedded structures need to be *visible* from the surrounding mesh. To do so, we increase the neighborhood complexity of simplicies defining the sub-structures as follow. The tetrahedral mesh is modeled as a simplicial complex K. The 2D (resp. 1D) substructures are defined as a set L_f (resp. L_e) of faces (resp. edges) in K. We define an *extended complex* $\widetilde{K}$ by adding to K the cones from a dummy vertex ω to each simplex in the closure of the substructures, as illustrated in Fig. 4. The key idea is to consistently encode the topology of the mesh and

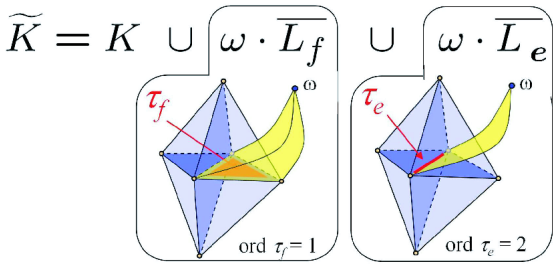


Fig. 4. Construction of the extended complex. The red simplicies correspond to the substructures L_f and L_e. The yellow simplicies represent the elements added to the original complex K (shown in blue) in order to extend K to $\widetilde{K}$.

its substructures in a single extended complex.

3.3 Link conditions in the extended complex

To preserve the topology of the mesh and its substructures, we apply the link conditions in the extended complex defined in Section 3.2. Even if the original mesh is manifold, the extended complex is non manifold. Thus, the link condition (1) must be evaluated respectively in the following complexes:

$$\begin{aligned} \widetilde{K}^{\omega} &= \widetilde{K} \cup (\omega \cdot \text{Bd}_1\ \widetilde{K}) \\ \widetilde{G}^{\omega} &= \text{Bd}_1\ \widetilde{K} \cup (\omega \cdot \text{Bd}_2\ \widetilde{K}) \\ \widetilde{H}^{\omega} &= \text{Bd}_2\ \widetilde{K} \cup (\omega \cdot \text{Bd}_3\ \widetilde{K}) \end{aligned} \tag{2}$$

where $\mathrm{Bd}_i\ \widetilde{K}$ is the set of all simplicies of order i or less in $\widetilde{K}$. The link of a simplex in $\widetilde{K}^\omega$, $\widetilde{G}^\omega$ and $\widetilde{H}^\omega$ is respectively noted as Lk_0^ω , Lk_1^ω and Lk_2^ω . The precise definition of the order of a simplex is given in [4]. Simplicies of dimension k in a 3-complex have an order not greater than $3-k$. Therefore $\widetilde{K}^\omega$ is a 3-complex, while $\widetilde{G}^\omega$ contains only edges and $\widetilde{H}^\omega$ only vertices.

Note that our algorithm can deal with non-manifold tetrahedral meshes. However for clarity reasons, the following discussion is restricted to the case where the tetrahedral mesh is a manifold with boundary.

In this case, $\mathrm{Bd}_1\ \widetilde{K}$ contains the boundary faces of K and the faces of L_f. It contains also the edges and vertices of these faces. $\mathrm{Bd}_2\ \widetilde{K}$ contains the edges of L_e and the edges adjacent to exactly one, or else three or more faces of L_f (i.e., the boundary and the non-manifold edges of L_f). It contains also the vertices of these edges. Finally $\mathrm{Bd}_3\ \widetilde{K}$ contains the vertices adjacent to one or else three or more edges of $\mathrm{Bd}_2\ \widetilde{K}$ (i.e. the boundary and non-manifold vertices of $\mathrm{Bd}_2\ \widetilde{K}$). This includes vertices at the intersections created when a polyline in L_e crosses a surface in L_f.

By experience, the simplicies of high order correspond to regions of interest that the user visualizes in detail. Important features in the mesh (detected by cells marked as simplices of high order) are usually correlated with high variations in the data. This point is clearly illustrated in the electromagnetic simulations where the field greatly varies around the linear structure or the points of contact in between this element and a mass plan.

4 Implementation of the Topology Preservation Test

4.1 Datastructure of the embedded simplices

The implementation of the link condition requires to iterate through all adjacent simplicies around a vertex and an edge. For memory requirement and efficiency we do not store explicitly all simplicies (faces, edges and vertices) of our substructures. Instead, as both the substructures and the link simplicies are embedded in cells of the original tetrahedral mesh, all of them are encoded as *relative element indices* in a tetrahedron. For each tetrahedron 4 bits (resp. 6 bits) are used to indicate which faces (resp. edges) support a 2D (resp 1D) substructure. Both the triangle mesh connectivity of the 2D substructures and the linear mesh connectivity of the 1D substructures are retrieved through the tetrahedron mesh connectivity without using additional memory. Moreover, during the simplification process, such a data structure eases and optimizes the updates of the substructures after an edge collapse. A modification on the tetrahedral mesh will be directly passed on to the substructures, without any additional treatment.

4.2 Algorithmic implementation

Algorithm 1 describes the evaluation of the link conditions in order to detect if the collapse of the edge $e = uv$ preserves both the topology of the mesh and the topology of any embedded structure. The extraction of the link of a simplex is the most repeated operation when evaluating the link conditions. It is also time consuming as the star of the reference simplex has to be exhaustively visited.

Therefore, the links are directly obtained without explicitly building the extended complex.

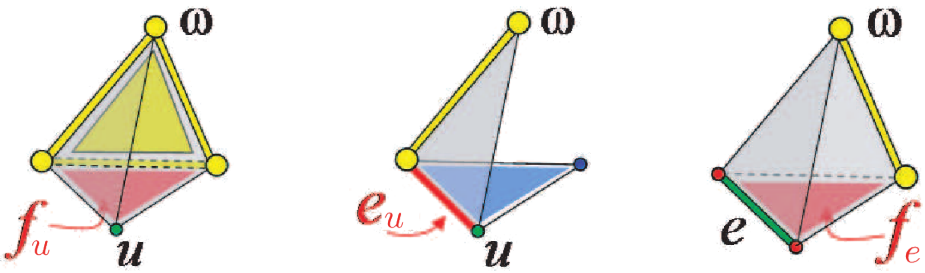


Fig. 5. Contribution to the links from the extended simplicies around a reference simplex (u or e shown in *green*). ω represents the dummy vertex. The substructures are shown in *red* and the simplicies to insert into the links in *yellow*.

In order to understand this optimized implementation, one has to associate the different members of equation (2) to the algorithm steps. For example, starting at line 5 and illustrated in Fig. 5 *left*, if f_u is a face of a substructure then the dummy tetrahedron T^ω increases the order of f_u to 1. Thus, f_u is included in $\mathrm{Bd}_1\ \widetilde{K}$ and contributes to Lk_0^ω but also to Lk_1^ω because of the member $\omega \cdot \mathrm{Bd}_1\ \widetilde{K}$ of $\widetilde{G}^\omega$ (line 8). The simplices represented in yellow on the figure 5 *middle* are the contribution of the triangle t^ω defined on line 13 to Lk_0^ω and Lk_1^ω. This extension increases the order of the edge e_u to 2 making it visible in $\mathrm{Bd}_2\ \widetilde{K}$. Thus, line 15 of the algorithm implements the contribution of the simplicies present in the member $\mathrm{Bd}_2\ \widetilde{K}$ of $\widetilde{H}^\omega$. A last example, shown on figure 5 *right*, illustrates the contribution of the dummy tetrahedron T^ω introduced on line 28. In this case only the opposite edge (yellow) to e (green) is inserted to the link of $\mathrm{Lk}_0^\omega\ e$ as no face can be present in the link of an edge.

One has to note that due to the strong theoretical background, the topological criteria is insensitive to the complexity of the neighborhood of the collapse (multiple surfaces and polylines).

5 Applications

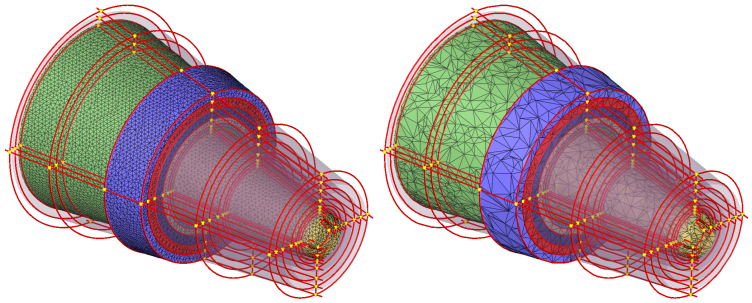
5.1 Volumetric Simplification

This section presents various applications of the topology preserving visualization system introduced in the present paper.

The example on figure 6 shows the simplification of a mesh used in electromagnetism simulation. The object is composed of approximatively twenty thin layers of materials encapsulated in each others. The geometry is meshed with about two millions of tetrahedra. In order to study the circulation of the electromagnetic field at the boundary of each material, all interfaces are explicitely described as embedded surfaces. Additional linear structures with self intersections are inserted in the model. Fig. 6(b)

Algorithm 1 IsEdgeCollapsePreservingTopology(K, u, v, e)

1: # Lines 2 to 21: Compute $\mathrm{Lk}_0^\omega\ u$, $\mathrm{Lk}_1^\omega\ u$ and $\mathrm{Lk}_2^\omega\ u$
2: **for all** T in $St(u)$ **do**
3: Add to $\mathrm{Lk}_0^\omega\ u$ the 2-face of T opposite to u, its 3 edges and its 3 vertices
4: **for all** adjacent face f_u to u in T **do**
5: **if** f_u is a face of a substructure **then**
6: A dummy tetrahedron is defined as $T^\omega = \omega \cdot f_u$
7: Add to $\mathrm{Lk}_0^\omega\ u$ the face of T^ω opposite to u, its 3 edges and its 3 vertices
8: Add to $\mathrm{Lk}_1^\omega\ u$ the edge of f_u opposite to u and its 2 vertices
9: **end if**
10: **end for**
11: **for all** adjacent edge e_u to u in T **do**
12: **if** e_u is an edge of a polyline **then**
13: A dummy triangle is defined as $t^\omega = \omega \cdot e_u$
14: Add to $\mathrm{Lk}_0^\omega\ u$ and $\mathrm{Lk}_1^\omega\ u$ the edge of t^ω opposite to u and its 2 vertices
15: Add to $\mathrm{Lk}_2^\omega\ u$ the vertex of e_u opposite to u
16: **end if**
17: **end for**
18: **if** u is an order 3 vertex **then**
19: Add ω to $\mathrm{Lk}_2^\omega\ u$
20: **end if**
21: **end for**
22: # To compute $\mathrm{Lk}_0^\omega\ v$, $\mathrm{Lk}_1^\omega\ v$ and $\mathrm{Lk}_2^\omega\ v$ apply lines 2 to 21 after changing u in v.
23: # Lines 25 to 37: Compute $\mathrm{Lk}_0^\omega\ e$ and $\mathrm{Lk}_1^\omega\ e$ ($\mathrm{Lk}_2^\omega\ e$ is alway empty)
24: **for all** T_e in $St(e)$ **do**
25: Add to $\mathrm{Lk}_0^\omega\ e$ the edge of T_e opposite to e and its 2 vertices
26: **for all** f of T_e adjacent to e **do**
27: **if** f is a face of a substructure **then**
28: A dummy tetrahedron is defined as $T^\omega = \omega \cdot f$
29: Add to $\mathrm{Lk}_1^\omega\ e$ the edge of T^ω opposite to e and its 2 vertices
30: Add to $\mathrm{Lk}_1^\omega\ e$ the vertex of f opposite to e
31: **end if**
32: **end for**
33: **if** e is an edge of a polyline **then**
34: Add to $\mathrm{Lk}_0^\omega\ e$ and $\mathrm{Lk}_1^\omega\ e$ the vertex ω
35: **end if**
36: **end for**
37: # Lines 39 to 41: Evaluate the link conditions
38: **if** $\mathrm{Lk}_0^\omega\ u \cap \mathrm{Lk}_0^\omega\ v \neq \mathrm{Lk}_0^\omega\ e$ **then return** `FALSE`
39: **if** $\mathrm{Lk}_1^\omega\ u \cap \mathrm{Lk}_1^\omega\ v \neq \mathrm{Lk}_1^\omega\ e$ **then return** `FALSE`
40: **if** $\mathrm{Lk}_2^\omega\ u \cap \mathrm{Lk}_2^\omega\ v \neq \emptyset$ **then return** `FALSE`
41: **return** `TRUE`

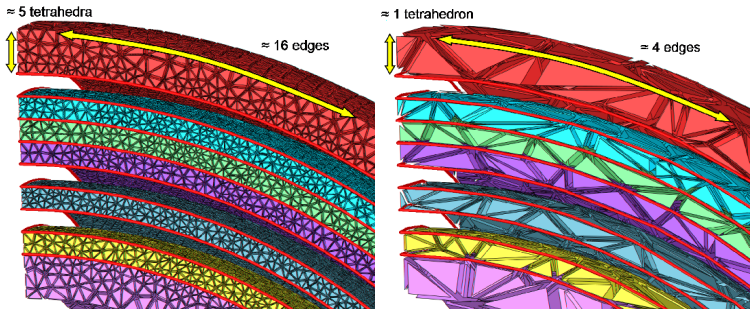


(a) Original mesh (b) After 90% of simplification

Fig. 6. Volumetric simplification of a model with multiple substructures. A few material layers are shown as opaque triangle mesh and all interfaces as semi-transparent surfaces. The linear features are in red and their self intersections marked by yellow shperes (color plate C. 30, page 264).

shows the simplified mesh after removing 90% of the vertices with edge collapses. This example illustrates the strength of the algorithm on data composed of a large amount of substructures of different dimension with multiple self intersections (e.g. increasing the topological complexity).

Achieving aggressive simplification rates becomes challenging when dealing with large volumetric meshes organised in thin material layers. Fig. 7 shows a detail of such



(a) Original mesh (b) After 98% of simplification

Fig. 7. Aggressive simplification of a volume mesh composed of thin material layers. The topological criteria preserve both the thickness of the layers and the topology of the substructures while allowing collapses of edges on the substructures (color plate C. 31, page 265).

data where the material layers have no more than 5 tetrahedra in their thickness. Linear features are also defined on specific boundaries. Fig. 7(b) shows the result after a massive simplification where 98% of the vertices have been removed by edge collapses. In this example the simplification stopped when all candidate edges are rejected due to

the topological criteria. It is important to note that the topological constraints on the 2D interfaces ensure the preservation of the volume of each layer (at least 1 tetrahedron in the thickness and usually no more). Another interesting aspect of this example is the fact that strong constraints on the substructure topology do not disturb or prevent an high simplification rate of the substructures. The arrow drawn along a linear feature living on an interface covers about 16 edges on the initial mesh but only 4 edges after simplification. Thus many edges of the interface and of the polyline have been collapsed.

5.2 Multiresolution Visualization

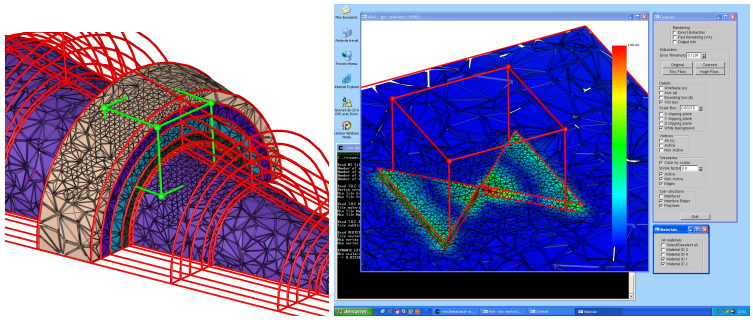


Fig. 8. $(left)$ Variable resolution visualization of a volume mesh with multiple linear features. The topology of the substructures is guaranteed to be preserved. $(right)$ Snapshot of the multiresolution visualization tool to explore simulation data with embedded structures on a desktop PC (color plate C. 32, page 265).

In order to provide a visualization system taking advantage of the substructure topology preserving criteria, the algorithm has been integrated into an existing multiresolution framework. The `MT` library [5], freely available and based on the formal approach of [6], provides the tools to build and exploit a multiresolution representation from a sequence of valid transformation on a mesh. `MT` stores a partial order amongst the modifications into a DAG which can then be querried with static or dynamic criteria to produce various resolutions. Global or local extractors allow the user to extract meshes at variable resolution centered around a volume of interest (VOI). Fig. 8 and 1 show a multiresolution mesh at high resolution only inside a VOI interactively moved by a user. The mesh at low resolution reduces the use of the graphic processor allowing the application to maintain an interactive frame rate even with an original mesh of several millions of cells. As the substructure preserving topology criteria have been used to build this representation, the topology of the substructures is guaranteed to be preserved on the multiresolution visualizations.

6 Conclusion

This paper has introduced new results for simplifying tetrahedral meshes with 2D and 1D substructures. As a specific application our system can simplify multi-material tetrahedral meshes while accurately preserving the topology of the material parts. To the best of our knowledge, this is the first paper that deals with the simplification of multi-material tetrahedral meshes. Our system can also handle more general 2D and 1D substructures. An application in the field of electromagnetism simulation, from which the system originates, has been shown. This application and the technology explained (multiresolution) are part of the strategy of the CEA in order to visualize large data computed on a remote super-computer. While focusing on topology preservation, our system can accommodate any geometric and numeric error measures as well.

References

1. P. Cignoni, D. Costanza, C. Montani, C. Rocchini, and R.Scopigno. Simplification of tetrahedral volume with accurate error evaluation. In *Proceedings IEEE Visualization '00*, pages 85–92, 2000.
2. P. Cignoni, L. D. Floriani, P. Lindstrom, V. Pascucci, J. Rossignac, and C. Silva. Multiresolution modeling, visualization and streaming of volume meshes. In *Eurographics 2004, Tutorials 2: Multi-resolution Modeling, Visualization and Streaming of Volume Meshes*. INRIA and the Eurographics Association, September 2004.
3. P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Multiresolution representation and visualization of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):352–369, oct–dec 1997.
4. T. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge contraction. *Publications de l'Institut de Mathématiques (BEOGRAD)*, 66(80):23–45, 1999.
5. L. D. Floriani, P. Magillo, and E. Puppo. *The MT (Multi-Tesselation) package*. DISI, University of Genova, Italy, http://www.disi.unige.it/person/MagilloP/MT, Jan. 2000.
6. L. D. Floriani, E. Puppo, and P. Magillo. A formal approach to multiresolution modeling. In W. Straer, R. Klein, and R. Rau, editors, Theory and Practice of Geometric Modeling. SpringerVerlag, 1997., 1997.
7. H. Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996.
8. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. Technical Report TR 93-01-01, Dept. of Computer Science and Engineering, University of Washington, Jan. 1993.
9. M. Kraus and T. Ertl. Simplification of nonconvex tetrahedral meshes, 2000.
10. D. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.
11. J. Munkres. *Elements of algebraic topology*. Perseus publishing, Cambridge, 1984.
12. V. Natarajan and H. Edelsbrunner. Simplification of three-dimensional density maps. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):587–597, 2004.
13. W. Schroeder, J. Zarge, and W. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.
14. F. Vivodtzev, G.-P. Bonneau, and P. Letexier. Topology preserving simplification of 2d nonmanifold meshes with embedded structures. *The Visual Computer*, 21(8):679–688, 2005.

Stripe Parameterization of Tubular Surfaces

Felix Kälberer, Matthias Nieser, and Konrad Polthier

Freie Universität Berlin

Abstract. We present a novel algorithm for automatic parameterization of tube-like surfaces of arbitrary genus such as the surfaces of knots, trees, blood vessels, neurons, or any tubular graph with a globally consistent stripe texture. Mathematically these surfaces can be described as thickened graphs, and the calculated parameterization stripe will follow either around the tube, along the underlying graph, a spiraling combination of both, or obey an arbitrary texture map whose charts have a 180 degree symmetry.
We use the principal curvature frame field of the underlying tube-like surface to guide the creation of a global, topologically consistent stripe parameterization of the surface. Our algorithm extends the QuadCover algorithm and is based, first, on the use of so-called projective vector fields instead of frame fields, and second, on different types of branch points. That does not only simplify the mathematical theory, but also reduces computation time by the decomposition of the underlying stiffness matrices.

1 Introduction

Tubular surfaces appear in many application areas such as networks of blood vessels and neurons in medicine, or tube and hose systems in industrial environments. Often a tubular structure must be recovered and segmented from noisy scan data. Our *stripe parameterizer* is an efficient and automatic method for the parameterization and remeshing of free-form tubular surfaces given as triangle meshes. Our special focus lies on free-form surfaces which are not made out of regular, cylindrical tube pieces - those can be handled better by other algorithms from CAD. An additional benefit of the stripe parameterization is the enhanced visualization of the underlying geometric structure.

1.1 Previous work

Surface parameterization is an active research area. We will not attempt a complete review of the literature but instead refer the reader to the surveys by Floater and Hormann [6] and [11].

A very early surface parameterization method is the Tutte's [24] barycentric graph embedding. Tutte embeddings are of combinatorial structure and do not capture the geometry of the surface. Early global parameterization methods were introduced by Haker, Gu and Yau [8,9,13] who studied conformal parameterizations. Conformal maps are angle preserving at the cost of possibly large length distortions, as angles and lengths can not be preserved at the same time.

To reduce length distortion, Kharevych et al. [16] used cone singularities for conformal parameterization, which relax the constraint of a flat parameter domain at few

isolated points. Such singularities have proven to be essential for high quality parameterizations and have been used in other parameterization schemes as well.

Tong et al. [23] use singularities at the vertices of a hand-picked quadrilateral meta layout on the surface. The patches the meta layout are then parameterized by solving for a global harmonic one form. Dong et al. [3] use a similar idea for parmeterization but create the quadrilateral meta layout automatically from the Morse-Smale complex of eigenfunctions of the mesh Laplacian.

Ray et al. [21] parameterize surfaces of arbitrary genus with periodic potential functions guided by two orthogonal input vector fields. This leads to a continuous parameterization except in the vicinity of singular points on the surface. These singular regions are detected and reparameterized afterwards. With the QuadCover algorithm [15] we built upon their idea to use an input field as guiding directions for parameter lines. Input fields can be principal curvature directions, for example, or user-designed fields using one of the recent tools for the design of rotational symmetry fields like [22], [18], [25], or [17]. The idea of QuadCover is to find a parameterization whose gradient matches the input directions as well as possible.

The literature on parameterization of tubular objects is by far not as extensive as for general surfaces. Huysmans et al. [12] construct a progressive mesh which they map to an open cylinder. A subsequent iterative scheme optimizes the vertex positions in the cylindrical domain. Unfortunately, that method can not handle bifurcations. Antiga and Steinman [1] handle blood vessels with bifurcations by splitting the vessel tubes at their branches, and parameterize each segment separately which leaves visible artifacts at the joints of the segments. Zhu et al. [26] use conformal parameterizations on tubular objects. Since conformal maps do not allow precise control over the direction of parameter lines, they cannot be aligned with the tube axis.

1.2 Contributions

We introduce the stripe parameterizer, an algorithm for the generation of globally consistent stripe parameterizations, see Fig. 1 and 9. Each parameterization is a collection of texture maps which may also be used to remesh and segment a surface. The stripe parameterizer is a generalization of QuadCover, which parameterizes general surface meshes. The stripe parameterizer allows to map stripe patterns onto a surface, *i. e.*, texture maps whose individual charts are symmetric with respect to rotations of 180 degrees. In contrast to QuadCover where all texture image charts have to be symmetric with respect to 90 degree rotations, the stripe parameterizer allows a more general set of texture images with only 180 degree symmetry.

We develop the mathematical theory for stripe parameterizations and discuss the differences to grid parameterization techniques including those in QuadCover. Stripe parameterizations allows only a subset of the branch point types of QuadCover. For example, cone points of index 1/4 at the corners of a cube can not be used in stripe parameterizations since 90 degree rotational symmetric textures charts would be required, see the cube in Fig. 2.

Only one type of branch points can occur on a 2-sheeted covering, so there is no need to handle different branch types. The 4-sheeted branched covering surface from QuadCover projects onto a unique 2-sheeted branched covering surface for the stripe

parameterizer. Furthermore, the stiffness matrix from QuadCover decomposes into two matrices of quarter size. Thus, the numerical effort of computing a stripe parameterization is seriously reduced.

We tested the stripe parameterizer on several test models and real world examples including clinical data and various tree-like surfaces.

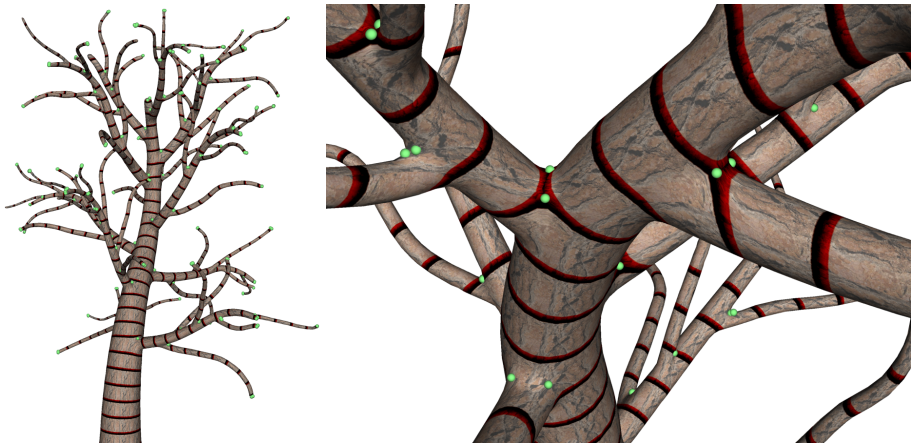


Fig. 1. Tree with stripe parameterization. Singularities are marked in green. The texture image consists of a vertical stripe visualizing the u-isolines of the parameterization (color plate C. 33, page 265).

2 Overview

A stripe parameterization is a special case of a (u,v)-parameterization, where the parameter lines can be globally separated into u-lines and v-lines, as in Fig. 1. This separation property is not present on general surfaces, if singularities of quarter index are present.

Stripe parameterizations can be used for mapping texture images which are symmetric by rotations of 180 but not necessarily 90 degrees, such as stripe textures. An example of a parameterization which is not suitable for mapping stripe textures is shown in Fig. 2.

Projective fields. The parameterization is guided by a so-called *projective field*, which is a vector field on M, where the vectors v and $-v$ are identified for all $v \in T_pM$, $p \in M$. Thus, the vectors may change their sign without producing a discontinuous projective field. Note, that projective fields are a special case of N-RoSy fields for $N = 2$ as introduced by Palacios and Zhang [18].

The algorithm takes two projective fields as input and generates two scalar functions (u and v), whose gradients match up with the input fields as well as possible. The coordinates u and v can be used as texture coordinates in order to map a pattern onto

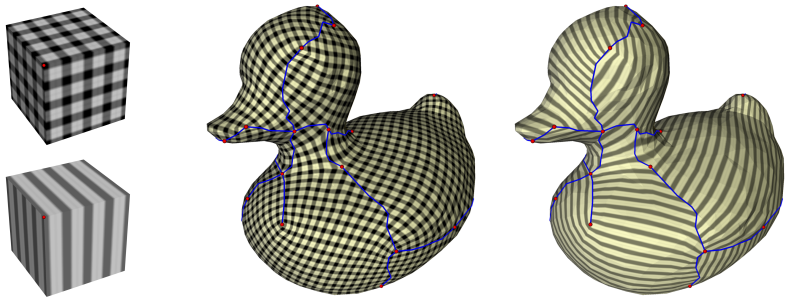


Fig. 2. QuadCover parameterization with quad texture and stripe texture. Stripe textures can not simply be used in parameterizations from QuadCover (color plate C. 34, page 266).

the surface. If you are only interested in a stripe pattern, you could use only one input field and skip the computation of v.

Construct an input field. It is up to the user to construct an input field. A canonical choice is the field of minimum curvature: In each point, the corresponding vector points in direction of the (absolute) smaller principle curvature and has unit length. Using this field (together with the 90 degrees rotated field) as the input fields yields nearly curvature line parameterizations.

The algorithm. Starting from a given projective field, the algorithm first constructs a locally integrable field. Second, the surface is cut open to a topological disk and this field is integrated yielding a parameterization. Third, the parameterization is adapted such that the grid lines are connected across the cuts. Details are given in Sect. 4.

Special issues arise when the projective field has singularities. They are resolved by using branched covering spaces. The projective field naturally simplifies to a single vector field on the covering, and then standard Hodge decomposition techniques are used to assure global integrability. Details are explained in Sect. 3.2.

3 Mathematical Setting

We use the theory of QuadCover's 4-fold symmetric fields and apply it to the projective vector field setting with 2-way symmetry properties. We introduce the notion of projective vector fields and discuss consequences for the branched 2-fold covering spaces. We will describe our concepts in the smooth cast first, followed by the discretization for triangle meshes.

3.1 Parameterizations and Matchings

Smooth case. Given a smooth 2-manifold M with charts $U_i \subset M$, $\sum_i U_i = M$. A parameterization is a collection of diffeomorphisms $f_i = (u_i, v_i)$ that map all charts into the parameter domains $f_i : U_i \to \Omega_i \subset \mathbb{R}^2$. One can now visualize the parameter lines on M as the preimage under f_i of the unit grid $\mathbb{N} \times \mathbb{R}$ (u_i lines) and $\mathbb{R} \times \mathbb{N}$ (v_i lines).

A **globally continuous stripe parameterization** consists of parameter functions f_i in the charts U_i, such that the u_i lines (resp. v_i lines) coincide in all regions where two charts U_i, U_j overlap. Thus, the parameter lines of a stripe parameterization can be globally separated into u-lines and v-lines.

Given two guiding fields on the surface, we will only focus on computing the u-component from the first input field, as the same rules apply for computing v from the second field.

The transition functions between adjacent charts of a stripe parameterization satisfy two conditions: First, the gradients of u_i and u_j have to agree up to sign, because we do not distinguish u_i lines and $-u_i$ lines on the parameterized surface. Thus, the gradients of the charts are related by

$$\nabla u_i(p) = (-1)^{r_{ij}} \nabla u_j(p), \;\; p \in U_i \cap U_j \tag{1}$$

with a constant number $r_{ij} \in \{0,1\}$ on the intersection $\Omega_i \cap \Omega_j$. We call the values r_{ij} *matchings* between charts U_i and U_j.

Second, the values of u_i and u_j may differ only by integer values, since the u lines in the unit grid are invariant under translations by integer values.

Thus, we require the values of u in overlapping charts U_i and U_j to fulfill:

$$u_j(p) = (-1)^{r_{ij}} u_i(p) + t_{ij} \;\; , \quad r_{ij} \in \{0,1\}, \;\; t_{ij} \in \mathbb{N}, \;\; p \in U_i \cap U_j \;\; . \tag{2}$$

Discretization. Each triangle of the mesh is considered as a chart. The transition function between two adjacent triangles is fully determined by the matching and the translation vector associated to their common edge, see (2). See Sect. 4.1 for details on how we compute the matching.

3.2 Projective Fields

A parameter function u can be characterized by its gradient field. In each chart, the gradient field ∇u is a continuous vector field. At the transition between two charts U_i, U_j, the sign of the vectors may flip depending on the matching. Thus, we cannot describe the derivatives of u as a globally defined vector field, but use *projective fields* which are invariant under sign flips.

Definition 1. *Given a manifold M with charts U_i and matchings r_{ij}. A **projective field** K on M is a collection of one vector field K_i in each chart U_i, such that for all overlapping charts $U_i \cap U_j \neq \emptyset$:*

$$K_j = (-1)^{r_{ij}} K_i. \tag{3}$$

Discretization. The projective fields are piecewise constant on the triangles. Store one vector per triangle and the matching number on each edge. This fully defines a discrete projective field. An odd matching at any edge means that the vector in one adjacent triangle corresponds to the negated vector of the other triangle.

3.3 Branched Covering Spaces

We use the notion of branched covering surfaces for an equivalent description of projective fields. A projective field on the input surface can be regarded as a vector field on a covering surface. This allows us to apply standard vector field calculus to projective fields.

Coverings. First, recall some definitions on Riemann surfaces, see [5,7,14]. We will give an abstract definition of a covering and explain below how we actually construct one.

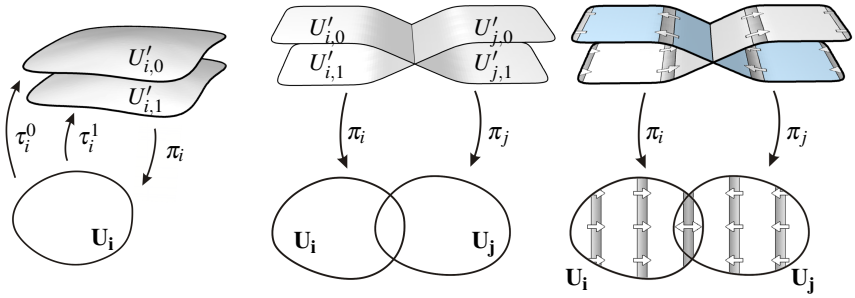


Fig. 3. From left to right: Trivial covering. / Patching two coverings together with matching $r_{ij} = 1$. / A projective field lifted to a vector field on the covering.

Definition 2. *Let M be a Riemann surface. A **2-sheeted covering** M' of M is a Riemann surface with a local homeomorphism $\pi : M' \to M$ with the property: For each point $p \in M$, there exists a neighborhood U_p whose preimage $\pi^{-1}(U_p)$ is the union of exactly two pairwise disjoint topological disks. Fig. 3, left shows a 2-sheeted covering.*

*We allow **branch points** p in our setting, where the preimage of a neighborhood of p is exactly one topological disk (instead of two), cp. Fig. 4, middle.*

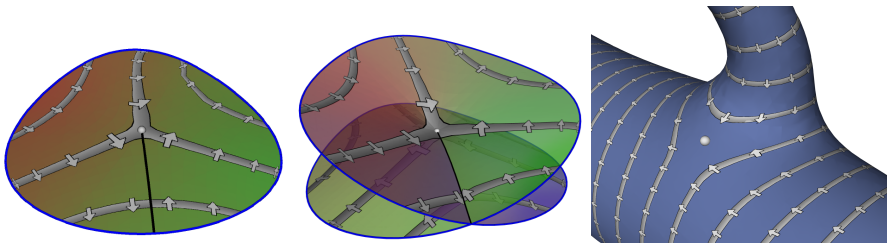


Fig. 4. Left: Stripe parameterization with branch point. The isolines of the u function and its gradient vectors are drawn. Middle: The same parameter function on the 2-sheeted covering (the covering surface is not embedded, it has self-intersections). Right: Branch point on a parameterized tube object (color plate C. 35, page 266).

Construction. We construct a covering of M as follows: From each chart U_i, make two copies (*layers*) and name them $U'_{i,0}$ and $U'_{i,1}$. Let $\pi_i : U'_{i,0} \cup U'_{i,1} \to U_i$ be the operator which projects the copies back to U_i and $\tau_{i,0} : U_i \to U'_{i,0}$, $\tau_{i,1} : U_i \to U'_{i,1}$ the inverse maps. The two layers $U'_{i,0} \cup U'_{i,1}$ together with π_i is called the *2-sheeted trivial covering* of the chart U_i (Fig. 3, left).

In the next step, we glue these layers at the overlaps of the adjacent charts together. For each pair of charts the layers can be glued in two different ways. The matchings r_{ij} define how the layers are identified.

Definition 3. *A **covering surface induced by matchings** r_{ij} is uniquely defined by the following construction:*

Let (U'_i, π_i) be 2-sheeted trivial coverings of the charts U_i. The covering surface is given as the union of all U'_i where the following points are identified: In each two overlapping charts U_i, U_j, identify all points $\tau_{i,0}(p)$ with $\tau_{j,r_{ij}}(p)$ and $\tau_{i,1}(p)$ with $\tau_{j,1-r_{ij}}(p)$, $p \in U_i \cap U_j$ (see Fig. 3, middle).

Since the trivial coverings of charts have no branch points and the charts cover the surface, we cannot construct any branch points this way. We allow branch points by removing single points from the surface. Depending on the matchings we obtain a branch point there as shown in Fig. 3.

Discretization. In the discrete setting, branch points are located at vertices. On a 2-sheeted covering they occur when the sum of all matchings of incident edges is odd. This means starting somewhere in the neighborhood of v and walking once around the vertex ends on a different layer in the covering than the start point.

3.4 Vector Fields on Covering Spaces

Projective fields can be described as vector fields on a covering surface. This result allows us to apply the classical vector field theory to projective fields.

A projective field K on M with matchings r_{ij} canonically lifts to a vector field K' on the covering induced by r_{ij}. In each chart U_i, define the vectors on its trivial covering as follows: For all $p \in U'_{i,0}$ set $K'(p) := K_i(\pi_i(p))$ and for $p \in U'_{i,1}$ set $K(p) := -K_i(\pi_i(p))$, see Fig. 3, third image.

The result is a globally well defined vector field K' on M', since the layers of the covering are connected in the same way as the vector fields permute when another chart is chosen.

Definition 4. *Let M be a manifold with matchings r_{ij} and M' the induced covering. A projective field lifted to a vector field K on M' is called a **covering field** of M.*

4 Stripe Parameterizer Algorithm

In this section we describe the main extensions and simplifications which have been made to QuadCover to yield the stripe parameterizer. An important difference is the use of projective vector fields instead of frame fields.

Compute the potential function. Given a surface M together with a projective field K, or, equivalently, a covering surface M' with a vector field K'. The parameter function is a scalar function $u' : M' \to \mathbb{R}$. It can be projected back to a parameter function $u : M \to \mathbb{R}$ by taking the values of u' in one of the two layers (it does not matter which layer is taken, because the parameter lines in both layers will be congruent).

The parameterization algorithm is divided into two stages: Assuring local continuity and global continuity. The two stages are explained in Sect. 4.3 and 4.4. Sect. 4.1 deals on the creation of an input field. For the integration of a projective field, we need to cut the surface open at a given cut graph. Sect. 4.2 explains the construction of such a cut graph.

4.1 Generate Input Field and Matching

Curvature field. In our experiments, we used the field of minimum principle curvatures as input to the parameterizer. Discrete principal curvature directions and values can be calculated as proposed in [2] or [10]. Note that we deal with curvatures given on triangles, not on vertices.

Finally, one gets a unit vector v in each triangle pointing in along that principle curvature direction which corresponds to the (absolute) smaller curvature value. In triangle t, set $K_0(t) := v$ and $K_1(t) = -v$.

We define matchings r_{ij} between every two adjacent triangles t_i and t_j by setting $r_{ij} = 0$ if $\langle K_0(t_i), K_0(t_j) \rangle \geq 0$ and $r_{ij} = 1$ otherwise. This ensures that the field does not turn around, but proceeds as straight as possible.

Note that the position of branch points immediately follows from the choice of matchings and the matchings are determined by the input field. A branch point arises at each vertex where the sum of matching of outgoing edges is odd.

4.2 Generating Cut Paths

A cut graph is a graph G embedded in the surface, such that $M \setminus G$ is a topological disk. We use a cut graph for the integration of projective fields in Sect. 4.3, and use cut paths for the global continuity in Sect. 4.4.

Cut paths on M. Loosely speaking, cut paths are a set of paths on the surface whose union is a cut graph. On closed surfaces, generating loops of the first fundamental group are suitable cut paths. In QuadCover, we use certain generators, namely the shortest system of loops as computed in Erickson and Whittlesey [4]. A system of loops is a set of $2g$ simple loops with a common base point, whose union is a cut graph.

We can treat branch points as tiny holes, as if they were removed from the surface (see Sect. 3.3). Erickson and Whittlesey handled closed surfaces only, but we might have a surface with boundary. Once we have more than one boundary component, each additional boundary component needs one path. Thus, in presence of $b > 1$ boundary components (or branch points) we need $2g + b - 1$ paths.

In our implementation for triangle meshes we identify all boundary vertices and branch points into one point B. On this surface (now without any boundary), we apply the method of [4] with B as the base point. When we undo the identification of boundary

points, the paths which looped through B now turn into paths that connect boundary components and branch points.

Cut paths on covering. For our algorithm, we need cut paths on the covering surface M'. We get the cut paths by computing them on M and lifting them to the covering. The resulting paths cut M' into two separate simply connected pieces. Thus, one of the paths could theoretically be discarded. It does not matter for our method that the covering decomposes into two pieces. It is more important, that the cut paths are symmetric with respect to a change of layers, *i. e.*, for each path there is another path which runs in the other layer and has the same projection onto M, see Sect. 4.4.

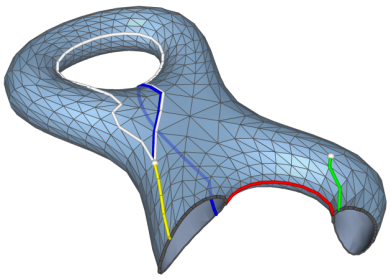


Fig. 5. Surface with boundary and two branch points. The colored lines visualize the five cut paths (color plate C. 36, page 266).

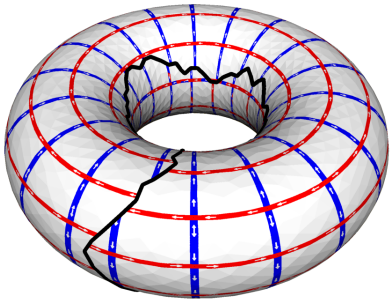


Fig. 6. Parameterization after the first stage. Grid lines are discontinuous across the cuts.

4.3 Local Continuity

The gradient of the parameterization should align with the given input field as well as possible, *i. e.*, u' should minimize the energy

$$E(u') = \int_{M'} \|\nabla u' - K'\|^2 dA. \tag{4}$$

Recall, that the vectors of K' are identical up to a different sign in the two layers. Since the energy has a unique minimum and due to the symmetric shape M', the solution u' is also a map with negated function values in different layers.

The optimization problem (4) can be solved using the Hodge-Helmholtz decomposition. It states, that any vector field K' has a unique decomposition

$$K' = P + C + H \tag{5}$$

with a gradient field P, a cogradient field C and a harmonic field H. P and H are curl free (locally integrable), whereas C contains the curl part. Furthermore, the three spaces of potential fields, copotential fields and harmonic vector fields are perpendicular in L^2 norm. Thus, discarding the second term leads to a curl free field $\tilde{X}' := P + H$ whose

integral is the minimizer of Energy (4). The middle term C of the Hodge-Helmholtz decomposition is a non-conforming function and is found by solving a linear system of equations with one variable per edge. For details on the Hodge decomposition and integration of discrete vector fields see [19].

So far, the parameterization algorithm outlines as follows:

1. Perform Hodge-Helmholtz decomposition of input field K'.
2. Discard the non-integrable curl part and obtain a locally integrable field.
3. Cut the surface open to be simply connected and lift the cut graph to the covering, such that the covering is cut into two connected pieces.
4. Obtain the parameterization u' by integration: Perform a linear run over all vertices (using a growing disk which does not cross the cut graph) and compute the parameter values at each vertex such that the gradient matches up with the vector field.

4.4 Global Continuity

The parameter lines of the solution u' from the previous paragraph are not necessarily continuous everywhere. They may have a mismatch at the cut graph G, see Fig. 6. Let γ_i be a set of cut paths. For each path γ_i and each point $p \in \gamma_i$, one can measure the **gap** d_i (discontinuous jump) as the difference of function values on the right and left side of the path.

The parameterization can now be "repaired" such that the parameter lines match up. This is exactly the case if all gaps are integer values. The repairing algorithm is based on the following observation: along each path γ_i, the gap is always a constant d_i, since the derivative of the function is locally integrable. Note, that there is an exception if two paths γ_i, γ_j merge and run on top of each other. In this case, the gap turns into $d_i + d_j$. For further details, see [15].

Thus, the grid lines are globally continuous if and only if all $d_i \in \mathbb{Z}$. In order to adapt the function to fulfill the global continuity condition, we add a scalar function ψ to u' such that $\tilde{u}' := u' + \psi$ satisfies $\tilde{d}'_j \in \mathbb{Z}$ (where $\tilde{d}'_j$ are the gaps of $\tilde{u}'$).

The remaining problem is to find this scalar function with given gaps. In order to minimally distort the initial parameterization, we let ψ be a harmonic function, as they are the smallest functions with given gaps in L^2 norm. ψ is found via minimizing the Dirichlet energy $E_D = \int_M \|\nabla\psi\|^2 dA$ under the constraint of given gaps.

The second stage of stripe parameterizer has the following outline:

1. Compute cut paths γ_i.
2. Measure gaps d_i.
3. Find harmonic map ψ with gaps $\text{round}(d_i) - d_i$.
4. Add ψ to u'.

In step 3, the gaps are rounded to the closest integer. Rounding the gaps such that the distortion is minimized is an NP hard combinatorial problem. As we do not solve this problem exactly, the rounding behavior slightly depends on the choice of cut paths.

5 Results

We have tested our method on different tube-like surfaces. Simple examples are the knots in Fig. 7 without branchings. The principal curvature directions are stable and can be computed very accurate in each point, so the algorithm produces a parameterization of high quality.

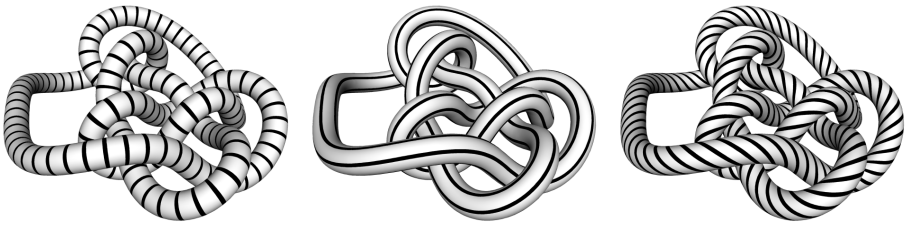


Fig. 7. Left: Only u coordinates are used to map stripes on the knot surface. Middle: Only v coordinates are used. Right: u and v coordinates are used. The texture image is a diagonal line which connects two opposite corners.

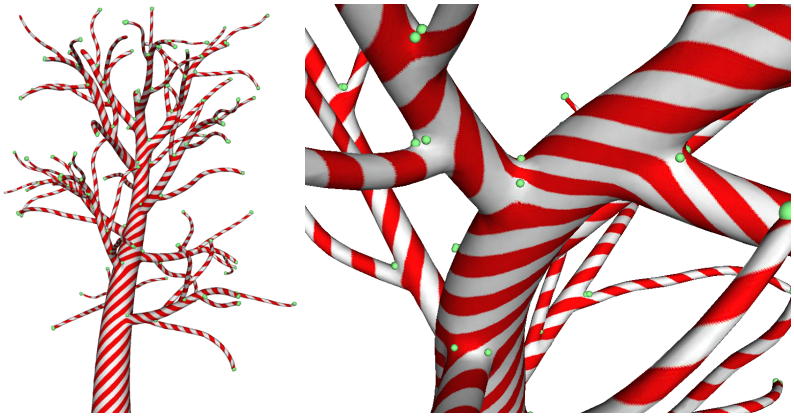


Fig. 8. The tree model of Fig. 1 with diagonal stripe pattern generates a candy cane. Singularities are marked in green (color plate C. 37, page 267).

The tree in Fig. 8 has a more complicated shape. It bifurcates and the thickness of the twigs vary. Note the accurate placement of branch points. There are two branch points at each bifurcation, allowing the circular stripes to split.

Fig. 10 shows a complex neuron model of genus 23, captured using confocal microscopy. The produced parameterization has very little distortion even on this complicated object.

The unshaded version in the top demonstrates how a stripe pattern helps to perceive the complicated shape of the neuron. But also in the fully shaded images, the stripes help to capture the object more easily.

The surface in Fig. 9 shows a human blood vessel which contains parts with a very large tube radius as well as very filigrane branches. Regardless of this difference in the scaling, the stripe density stays nearly constant everywhere.

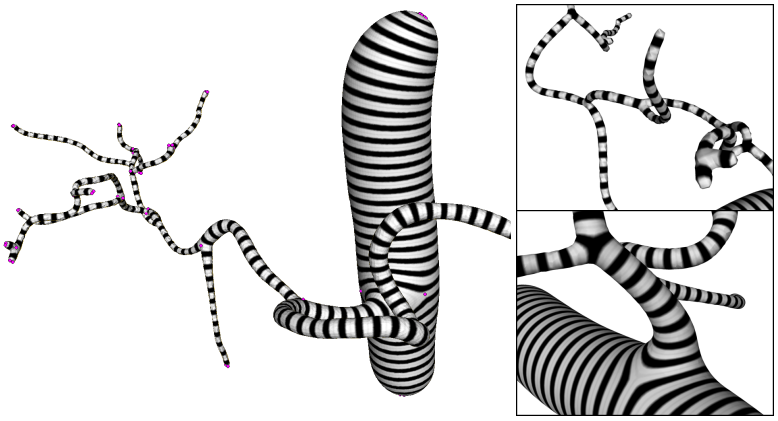


Fig. 9. Parameterized blood vessel, captured by MRT. Courtesy of Fraunhofer MEVIS.

The parameterization of these models was fully automatic. We only chose the density of the lines and the amount of curvature field smoothing. The models had approximately 20k to 40k triangles and the algorithm terminated in less than a minute.

6 Acknowledgements

The authors are grateful to Christian Hansen of Fraunhofer MEVIS (Bremen, Germany) for providing clinical 3D models of vascular structures and fruitful discussions concerning this work.

Many thanks to Sabine Krofczik and Jürgen Rybak, Department of Neurobiology at Freie Universität Berlin, as well as Steffen Prohaska and Anja Ku, Zuse Institute Berlin (ZIB) for supplying the neuron geometry.

This research was supported by the DFG Research Center MATHEON and by mental images.

References

1. L. Antiga and DA Steinman. Robust and objective decomposition and mapping of bifurcating vessels. *IEEE Trans. on Medical Imaging*, 23(6):704–713, 2004.
2. David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *Proc. of Symp. on Comp. Geom.*, pages 312–321. ACM Press, 2003.
3. S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J.C. Hart. Spectral surface quadrangulation. *ACM SIGGRAPH*, 2006.

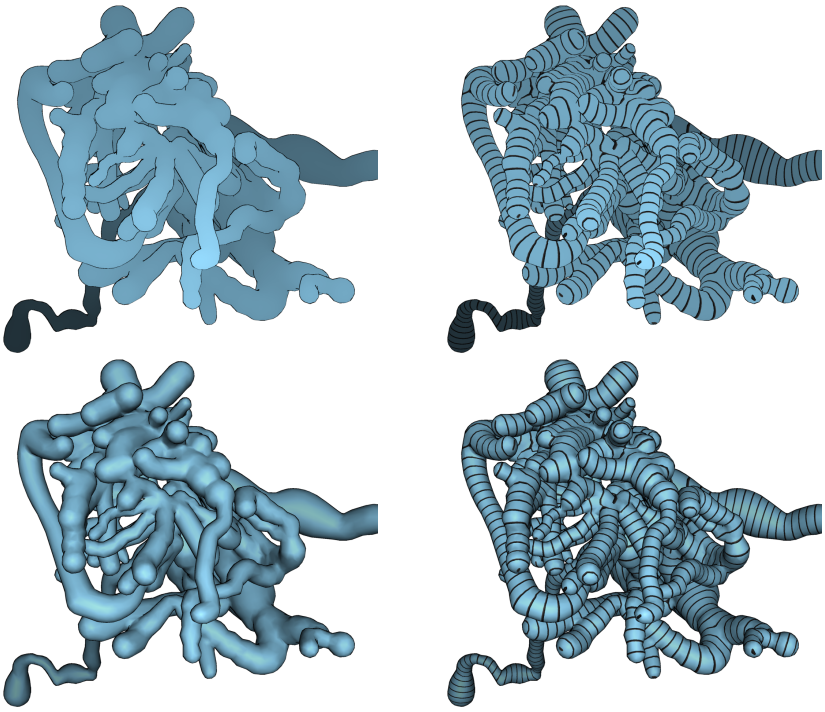


Fig. 10. Parameterized neuron by courtesy of Freie Universität Berlin, Department of Neurobiology. Top: depth shading. Bottom: full shading.

4. J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. In *Proc. 16th ACM-SIAM Symp. on Discrete Algorithms*, pages 1038–1046, 2005.
5. Hershel M. Farkas and Irwin Kra. *Riemann Surfaces*. Springer Verlag, 1980.
6. Michael S. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 157–186. Springer Verlag, 2005.
7. William Fulton. *Algebraic Topology, A first course*. Springer Verlag, 1995.
8. Xianfeng Gu and Shing-Tung Yau. Global conformal surface parameterization. In *Symp. on Geom. Proc.*, pages 127–137, 2003.
9. Steven Haker, Sigurd Angenent, Allen Tannenbaum, Ron Kikinis, Guillermo Sapiro, and Michael Halle. Conformal surface parameterization for texture mapping. *IEEE Trans. on Visualization and Computer Graphics*, 6(2):181–189, 2000.
10. Klaus Hildebrandt and Konrad Polthier. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum*, 23(3):391–400, 2004.
11. K. Hormann, K. Polthier, and A. Sheffer. Mesh parameterization: Theory and practice. In *SIGGRAPH Asia 2008, Course Notes*, number 11, 2008.
12. Toon Huysmans, Jan Sijbers, and Brigitte Verdonk. Parametrization of tubular surfaces on the cylinder. In *WSCG (Journal Papers)*, pages 97–104, 2005.
13. Miao Jin, Yalin Wang, Shing-Tung Yau, and Xianfeng Gu. Optimal global conformal surface parameterization. In *In IEEE Visualization*, pages 267–274, 2004.
14. Jürgen Jost. *Compact Riemann Surfaces*. Springer, 2002.

15. Felix Kälberer, Matthias Nieser, and Konrad Polthier. Quadcover - surface parameterization using branched coverings. *Comput. Graph. Forum*, 26(3):375–384, 2007.
16. Liliya Kharevych, Boris Springborn, and Peter Schröder. Discrete conformal mappings via circle patterns. *ACM Trans. on Graphics*, 25(2), 2006.
17. Yu-Kun Lai, Miao Jin, Xuexiang Xie, Ying He, Jonathan Palacios, Eugene Zhang, Shi-Min Hu, and Xianfeng David Gu. Metric-driven rosy fields design. Technical report, Tsinghua Univ., Beijing, 2008.
18. Jonathan Palacios and Eugene Zhang. Rotational symmetry field design on surfaces. *ACM Trans. on Graphics*, 26(3):55:1–55, 2007.
19. Konrad Polthier and Eike Preuss. Identifying vector field singularities using a discrete Hodge decomposition. In *Visualization and Mathematics III*, pages 113–134. Springer, 2003.
20. Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *SIGGRAPH*, page 581, 2001.
21. Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460–1485, 2006.
22. Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2):1–13, 2008.
23. Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. In *Eurographics Symp. on Geom. Proc.*, 2006.
24. William T. Tutte. How to draw a graph. *Proc. London Math. Soc.*, s3-13(1):743–767, 1963.
25. E. Zhang, J. Hays, and G. Turk. Interactive Tensor Field Design and Visualization on Surfaces. *IEEE Trans. on Visualization and Computer Graphics*, pages 94–107, 2007.
26. L. Zhu, S. Haker, and A. Tannenbaum. Flattening maps for the visualization of multibranched vessels, 2005.
27. Malte Zöckler, Detlev Stalling, and Hans-Christian Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5):241–253, 2000.

Eigenvector-based Interpolation and Segmentation of 2D Tensor Fields

Jaya Sreevalsan-Nair[1], Cornelia Auer[2], Bernd Hamann[3], and Ingrid Hotz[2]

[1] Texas Advanced Computing Center, University of Texas at Austin
[2] Visualization and Data Analysis, Zuse Institute Berlin, Germany
[3] Institute for Data Analysis and Visualization, Dept. of CS, UC Davis

Abstract. We propose a topology-based segmentation of 2D symmetric tensor fields, which results in cells bounded by tensorlines. We are particularly interested in the influence of the interpolation scheme on the topology, considering eigenvector-based and component-wise linear interpolation. When using eigenvector-based interpolation the most significant modification to the standard topology extraction algorithm is the insertion of additional vertices at degenerate points. A subsequent Delaunay re-triangulation leads to connections between close degenerate points. These new connections create degenerate edges and triangles. When comparing the resulting topology per triangle with the one obtained by component-wise linear interpolation the results are qualitatively similar, but our approach leads to a less "cluttered" segmentation.

1 Introduction

Generally tensor fields are not easy to understand owing to their complexity. Segmentation into regions of similar directional behavior offers a way to extract the essential structure of the field. A segmentation based on tensor field topology guarantees that all significant structural irregularities are captured. We build on the basic work by Delmarcelle et al. [5], which introduces tensor field topology to visualization, and the further advancement by Tricoche et al. [17], with the goal of a more complete segmentation. In contrast to previous work we consider the topological structure of both eigenvector fields as a whole. As both eigenvector fields are orthogonal the topological graphs deliver a descriptive cell structure bounded by tensorlines. The field in the neighborhood of degenerate points of the combined topology is characterized by "half-sectors" instead of "sectors" and allows a more general structure. The second focus of our work is the influence of the chosen interpolation. Dealing mostly with discrete data interpolation is an essential step in the visualization process. In order to keep our method simple, we consider only piecewise linear methods defined in a triangulated domain. Besides using the standard component-wise interpolation, we introduce an eigenvector-based approach. This method minimizes the number of eigen-analysis by restricting it to mesh vertices and supports an exact integration of tensorlines. By decoupling "shape" and "direction" we achieve a shape-preserving interpolation. Introducing new vertices at degenerate points and re-triangulating the domain leads to a simplified global topological structure without any modifications to the initial tensor field.

Our algorithm is designed to be simple and to follow clear rules. The algorithm consists of the following major steps: Eigenvector analysis at the vertices; edge labeling to ensure consistent direction interpolation; localization of degenerate points in each triangle and insertion of new vertices; determination of radial directions for both eigenvector fields; classification of the half-sectors defined by radial lines; integration of separatrices, and finally generation of a segmentation as intersection of the dual tensor field topologies.

2 Related Work

Most existing tensor field visualization methods are either specific to diffusion tensor fields or mechanical engineering applications, the latter being the focus of our work. The needs of domain experts in the area of diffusion tensor fields are well-defined as opposed to the lack of specific questions with respect to the tensors used in mechanical engineering. This difference in the driving force strongly reflects on their respective tensor field visualization techniques. Direct tensor visualization approaches focus on displaying tensor values in selected points. In this context research issues usually deal with the definition and placement of glyphs. Commonly used glyphs are, e.g., ellipsoids, Haber glyphs [7], or superquadrics [9], with improved perceptional properties. Different placement strategies are used to maximize the information displayed per image [6, 11]. While glyphs are appropriate for displaying single tensors, they are limited to low resolution and fail to give insight into the structure of the entire field. A more continuous view of 2D fields can be obtained by using tensor splats [3] or textures based on line integral convolution [8,19]. The idea of using topological methods to analyze the structures of tensor fields goes back to Delmarcelle [5] and Lavin et al. [12]. Tricoche et al. [17] improvised these ideas for applications of complex 2D tensor fields, by developing algorithms to simplify the tensor topology and to track it over time. Alliez et al. [2] proposed an application to curvature tensors for polygonal remeshing of surfaces. First analysis of tensor field topology 3D tensor fields shows that in 3D degenerate features form lines [20].

In a similar vein, we investigate the influence of different interpolation methods on the topology extraction process and the resulting topological structure. Traditionally the component-wise linear interpolation has been used on tensor fields. However, this interpolation generates artifacts, e.g., the swelling effect. In an effort to alleviate these artifacts, methods separating direction and shape interpolation have gained more attention lately. In context of diffusion MRI data some direction-based interpolation methods based have been proposed for tracing anatomical fibers, [4, 10, 13]. Several other advanced interpolation methods based on components have been developed to achieve noise reduction or feature preservation [1, 14, 18]. The emphasis of our interpolation method is to provide a simple and consistent method, based on eigenvectors and eigenvalues, with a focus on the behavior in the neighborhood of degenerate points.

3 Basics and Notations

3.1 Tensors and Tensor Fields

In the rest of the paper, we will refer to symmetric 2D tensors of second order, defined on a triangulated 2D domain, as tensor field. Using a fixed coordinate basis, each tensor $\mathbf{T}$ can be expressed as a symmetric 2×2 matrix, given by three independent scalars. We use the following notation

$$\mathbf{T} = \begin{pmatrix} d+\Delta & F \\ F & d-\Delta \end{pmatrix}. \tag{1}$$

$\mathbf{T}$ is fully represented by its *eigenvalues* λ and μ and corresponding *eigenvectors* $\overleftrightarrow{v}$ and $\overleftrightarrow{w}$. Since the multiplication of an eigenvector by any non-zero scalar yields an additional eigenvector, eigenvectors should be considered without norm and orientation. For symmetric tensors, the eigenvalues are real and the eigenvectors are mutually orthogonal. Integrating the eigenvector fields results in two orthogonal families of continuous curves. These curves are called *major* (red) and *minor* (blue) tensorlines according to the eigenvector field integrated.

Usually the tensor data-sets represent a discretized tensor field, whose geometry is represented by a triangulated mesh. Inside a triangle with vertices P_1, P_2, and P_3 we use barycentric coordinates $\beta = (\beta_1, \beta_2, \beta_3)$. The edge opposite vertex P_i is named e_i, for $i = 1,2,3$. We use $\overleftrightarrow{v}$ and $\overleftrightarrow{w}$ when referring to eigenvectors to allude to the fact that the eigenvectors are bidirectional. We use $\mathbf{v}$ and $\mathbf{w}$ when referring to vectors representing normalized eigenvectors with an arbitrarily but fixed direction, e.g., when using the results of the numerical computation. The direction of $\mathbf{w}$ is defined in such a way that $\mathbf{v}$ and $\mathbf{w}$ form a right-handed system. We assign the names λ and μ, such that always $\lambda \geq \mu$.

3.2 Tensor Field Topology

Similar to vector fields, the structure of eigenvector fields is represented by its topology. It defines a skeleton consisting of distinguished points, so called degenerate points, and connecting edges, the separatrices. This skeleton segments the domain into regions with characteristic tensorline behavior. In contrast to previous work we do not look at the topology of each eigenvector field separately but consider both topologies as a whole. In the following we resume the basics of tensor field topology, concentrating on the aspects that we need later on. For a more detailed discussion we refer the reader to [5, 16].

Degenerate points - Degenerate points are points where the two eigenvalues are identical $\lambda = \mu$, and the eigenvectors are no longer defined uniquely. The tensor is proportional to the identity matrix and all vectors are eigenvectors. Degenerate points in tensor fields correspond to critical points in vector fields. Due to orientation indeterminacy of tensorlines, these points exhibit structures different from those seen in vector fields. A condition for degeneracy of a point is $\Delta = 0 \wedge F = 0$. Independent of the eigenvalues, an

isolated degenerate point can be categorized by the number of windings an eigenvector performs when encompassing it on a closed curve. The undirected eigenvector field allows winding-numbers to be multiples of one half. Only simple degenerate points (winding-number of $\pm\frac{1}{2}$) can exist in a tensor field, defined by linear interpolation of its components.

Separatrices - The behavior of the tensorlines in the vicinity of degenerate points follows certain characteristic patterns, with respect to a characteristic composition of basic sectors, explained further in Section 5.4. These sectors are separated by distinguished tensorlines, which enter the degenerate point radially. Radial tensorlines bounding hyperbolic sectors are called *separatrices* and constitute the edges of the topological graph.

4 Eigenvector-based Interpolation

The definition of tensor field topology is based on continuous data and hence we rely on an interpolation of the available discrete data. The standard interpolation is linear in tensor components. Instead we propose the use of an interpolation based on eigenvectors and eigenvalues, see Figure 1. This method minimizes the number of eigenvector computations. The interpolation is defined such that the resulting topology per triangle is qualitatively the same as for component-wise interpolation, see Figure 3. The main steps of this interpolation are: edge labeling, location and insertion of degenerate points, subdivision of triangles, and vector interpolation.

Assignment of directions to eigenvectors - edge labeling
Using vectors for the interpolation we first have to assign orientations to the eigenvectors to specify the interpolation uniquely. Doing so we have to consider the fact that not all structures occurring in tensor fields can be simulated by global vector fields, e.g., winding numbers of half integers. Thus a consistent global orientation of the eigenvector field is not possible. Therefore we keep the arbitrarily directed eigenvectors at vertices as generated by numerical computations and only encode relative directions between neighboring vertices using edge labels. For the label definition we follow the eigenvector behavior given by the component-wise interpolation. When moving from point P_i to P_j the absolute value of the rotation angle of the eigenvectors is limited to $\pi/2$. The direction of the rotation is given by the value $F_j\Delta_i - F_i\Delta_j$. If the value is equal to zero, then either both eigenvectors are the same or they encompass an angle of $\pi/2$ and the rotation direction is undetermined. Then there exists a degenerate point on the

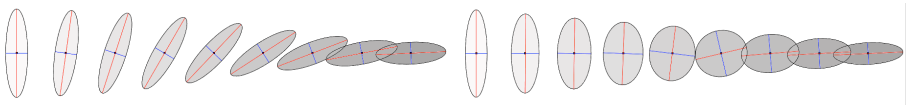


Fig. 1. Comparison of interpolation methods: left: eigenvector-based (shape preserving), right: component-based.

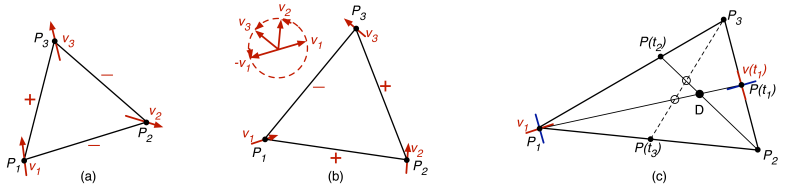


Fig. 2. Triangle (a) without and (b) with degenerate point, edge labels indicate whether two adjacent eigendirections match. (c) The location of a D is well-defined if the three lines connecting the vertices and their opposite points intersect in one point.

edge. The edge label of an edge e_k with endpoints P_i and P_j is defined as:

$$l(e_k) = \begin{cases} 1 \text{ if the directions of } \mathbf{v}_i \text{ and } \mathbf{v}_j \text{ match the direction propagation,} \\ \quad \text{meaning } \mathbf{v}_i \cdot \mathbf{v}_j > 0, \\ -1 \text{ if the directions of } \mathbf{v}_i \text{ and } \mathbf{v}_j \text{ do not match the direction propaga-} \\ \quad \text{tion, meaning } \mathbf{v}_i \cdot \mathbf{v}_j < 0, \\ 0 \text{ if there exists a degenerate point on the edge, } \mathbf{v}_i \cdot \mathbf{v}_j = 0. \end{cases}$$

The existence of a degenerate point inside a triangle P_i, P_j, P_k with edges e_i, e_j, e_k, is directly related to the product of its edge labels, see Figure 2. It is:

$$\prod_{i=1}^{3} l(e_i) = \begin{cases} 1 \text{ no degenerate point in triangle,} \\ \text{-1 one isolated degenerate point in triangle,} \\ 0 \text{ there is a degenerate point on at least one of the edges.} \end{cases} \tag{2}$$

If there exist two edges with degenerated points, we have a degenerate line. If there are three degenerate edges, the entire triangle is degenerate.

Interpolation in triangles without degenerate point

The tensor in point $P(\beta)$, $\beta = (\beta_1, \beta_2, \beta_3)$, which are the barycentric coordinates of point P inside a triangle P_1, P_2, P_3 is defined by its eigenvectors $\mathbf{v}$ and $\mathbf{w}$, which are not normalized, and eigenvalues λ and μ given by

$$\begin{aligned} \mathbf{v}(\beta) &= \beta_1 \mathbf{v}_1 + \beta_2 l(e_3)\mathbf{v}_2 + \beta_3 l(e_2)\mathbf{v}_3, \\ \mathbf{w}(\beta) &= \beta_1 \mathbf{w}_1 + \beta_2 l(e_3)\mathbf{w}_2 + \beta_3 l(e_2)\mathbf{w}_3, \\ \lambda(\beta) &= \sum_{i=1}^{3} \beta_i \lambda_i, \text{ and } \mu(\beta) = \sum_{i=1}^{3} \beta_i \mu_i. \end{aligned} \tag{3}$$

Interpolation in triangles with degenerate point

In such triangles it is not possible to define a continuous vector field representing the tensor field structure. However, we can resolve this problem by inserting an additional vertex D at the degenerate point and subdividing the triangle to triangles without interior degenerate point. To determine the eigenvalue at D we linearly interpolate the mean eigenvalue $d = (\lambda_i + \mu_i)/2$ in the original triangle and set the deviator $\Delta = (\lambda_i - \mu_i)/2$ to zero. Thus we can reconstruct the triangular domain by using piecewise linear interpolation in the subdivided domains. The tensor at point D is defined as a multiple of

the unit tensor. The eigenvectors at D are set to zero, in correspondence to vector field singularities. Each new triangle with vertices P_i, P_j, D is interpolated separately. With $P(\beta) := \beta_i P_i + \beta_j P_j + \beta_k D$, (cyclic indices) eigenvalues and eigenvectors are interpolated using Equation 3. The resulting eigenvectors are independent from the coordinate β_k.

5 Topology Extraction

The basic steps for extracting topology are: location and classification of degenerate points, determination of separatrix directions, and their integration. We restrict this section to eigenvector-based interpolation; we refer readers to [5, 16] for further details on linear component-wise interpolation. The main difference between the two interpolation schemes is caused by the triangle subdivision. All degenerate points lie on vertices with piecewise linear behavior in the vicinity and can exhibit structures different from trisector and wedge points.

5.1 Location of Degenerate Points

Since degenerate points at vertices can be detected easily, this section is restricted to triangles without degenerate vertices. Initially, we also assume that there is no degenerate point along edges, and thus from Equation 2, the edge label product has to be -1. We define the location of the degenerate point exclusively dependent on the eigenvector field. Starting with a linearly interpolated eigenvector field on the edges e_i with endpoints P_k and P_j, $i, j, k \in \{1,2,3\}$ cyclic, we compute

$$\mathbf{v}(t) = (1-t) \cdot \mathbf{v}_j + t \cdot l(e_i) \cdot \mathbf{v}_k, \qquad t \in]0,1[. \tag{4}$$

Even though the resulting vector field $\mathbf{v}$ on the boundary may not be continuous at all vertices, the corresponding un-oriented direction field $\overleftrightarrow{v}$ is. It defines a continuous rotation angle varying from zero to $\pm\pi$. The intermediate value theorem implies that for each vertex a parameter $t_i \in]0,1[$, i=1,2,3, exists such that $\mathbf{v}_i \cdot \mathbf{v}(t_i)$=0. Thus, for every vertex there exists a point on the opposite edge, called *opposite point of the vertex*, with

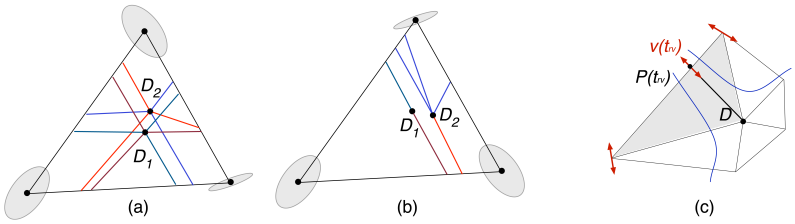


Fig. 3. Interpolation comparison in one triangle: Degenerate points D_1 from component-wise, D_2 from eigenvector-based interpolation, in the case of (a) trisector, and (b) wedge point. (c) Radial tensorline entering degenerate point D.

rotation angle $\pm\pi/2$ with degenerate point on its connection. The parameters t_i are given by

$$\mathbf{v}_i \cdot ((1-t_i)\mathbf{v}_j + l(e_i)t_i\mathbf{v}_k) = 0, \tag{5}$$

where $i, j, k \in \{1,2,3\}$ are cyclic indices. This leads to the following definition: *The location of the degenerate point is defined as the intersection of the connections of triangle vertices to their opposite points*, see Figure 2(c)
It can easily be seen that the point D is well-defined. From the definition of t_i in Equation 5, it follows $t_1t_2t_3 = (1-t_1)(1-t_2)(1-t_3)$, which is the condition that three lines connecting the vertices to points on the opposite edge, defined by parameters t_i, intersect in one point. For degenerate points on edges the three connecting lines degenerate to a line. In this case we use the eigenvalues at the vertices to determine the degenerate point.

5.2 Non-isolated Degenerate Points

Two degenerate vertices connected by an edge gives a degenerate line. The resulting eigenvector field inside adjacent triangles is constant and does not contribute to the final structure. Similarly a degenerate triangle, where all vertices are degenerate points, is enclosed by three triangles with constant eigenvector field. Thus from a structural point of view it is enough to consider the vertices of the degenerate entity and ignore the connecting edges. It is not uncommon to see degenerate polylines when applying a subsequent Delaunay re-triangulation.

5.3 Determination of Radial Directions

The neighborhood of the degenerate point is characterized by segments separated by radial tensorlines. For linear eigenvector interpolation, radial tensorlines are straight lines and are determined by their intersection $P(t_r)$ with the edges of adjacent triangles, see Figure 3. For each edge of the triangle,

$$\mathbf{v}(t_{rv}) \times (P(t_{rv}) - D) = 0, \qquad \mathbf{w}(t_{rw}) \times (P(t_{rw}) - D) = 0, \;\; t_{rv}, t_{rw} \in [0,1]. \tag{6}$$

t_{rv} and t_{rw} specify the radial directions for the eigenvector fields v and w respectively. In contrast to component-wise interpolation, where radial directions are given by one cubic equation, we obtain one quadratic equation per edge and per eigenvector field. If not trivially fulfilled this leads to a maximum of two solutions per edge and eigenvector.

5.4 Sector Classification

For the skeleton computation only radial lines, which are boundaries of hyperbolic sectors, are relevant. In the case of component-wise interpolation a point classification into trisector or wedge points serves as basis for the classification. To cover all possible cases of degenerate points, for piecewise linear behavior, we built on an immediate sector analysis similar to [17]. In contrast we classify "half-sectors", as we consider the topology of both eigenvector fields together. These are radial segments enclosed by two neighboring radial directions, independent of the eigenvector field, either red or blue, see Figure 4.

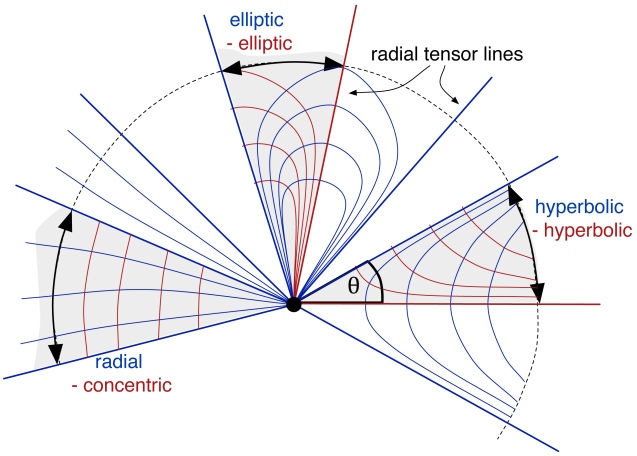


Fig. 4. The neighborhood of a degenerate points is characterized by a number of half-sectors with specific behavior.

Hyperbolic sector - bounded by one red and one blue radial line: tensorlines approach, sweep past the degenerate point and leave the sector through one bounding radial line.

Parabolic sector - bounded by two radial lines of same color: all tensorlines, of this color, start from the degenerate point and then diverge. The tensorlines of the other color enter and leave the sector through bounding lines.

Elliptic sector - bounded by one red and one blue radial line: the tensorlines start from the degenerate point, and leave the sector through one of the bounding lines.

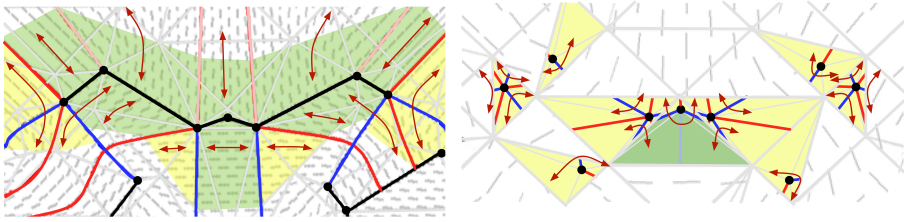


Fig. 5. A close-up of sector classification for the one-point load data-set using linear interpolation of eigenvectors, with (left) and without (right) subsequent re-triangulation. Shaded regions show the sectors: green and yellow for non-hyperbolic and hyperbolic, respectively; red and blue lines show radial lines, which are not integrated; black points and lines are the degenerate points and lines (color plate C. 38, page 267).

To classify the sectors the rotation angle of the eigenvectors $\Delta\alpha$ is compared to the opening angle of a half-sector $\Delta\Theta$ as shown in Figure 4

$$\begin{aligned} \Delta\alpha &= \Delta\Theta && \text{radial, concentric,} \\ \Delta\alpha &= \Delta\Theta - \pi/2 && \text{hyperbolic,} \\ \Delta\alpha &= \Delta\Theta + \pi/2 && \text{elliptic.} \end{aligned}$$

The same sector classification can also be used for degenerate lines and triangles. In this case all radial lines entering one of the participating vertices have to be considered. An example from a real data-set is shown in Figure 5.

5.5 Separatrix Computation

To complete the topological skeleton we integrate all radial tensorlines bounding the hyperbolic sectors using Runge-Kutta 4th-order integration scheme with adaptive step size. Alternatively, an exact tensorline integration for the linear eigenvector field can be used [15]. Direction consistency is not an issue in our approach, as in the case for component-wise interpolation, since eigenvector interpolation gives directed eigenvectors. We implemented the following termination conditions, to obtain a clean integration of tensorlines.

1. A separatrix leaves the domain, a trivial condition.
2. A separatrix gets close to a degenerate point, line or triangle. It is terminated at its intersection with the degenerated entity, see Figure 6a.
3. A separatrix describes a circle or spiral and passes itself closely in parallel integration direction , see Figure 6b and 6c. Circulating separatrices overload the topological graph without adding structural information for the final segmentation. We delete circulating tensorlines in a clean up process, which starts at the end of the separatrix and continues as long as the separatrix has a neighboring separatrix of the same color. The cleanup process ends in a point of intersection with a separatrix of the other color.

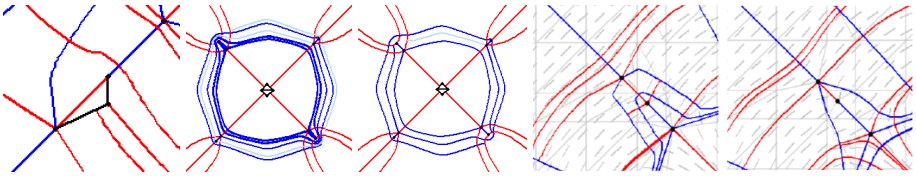


Fig. 6. Close-up from one-point load data-set: (a) tensorline runs into a degenerate line (black line); circulating tensorline (a) before and (b) after clean up; (d,e) comparison of separatrix integration for component-wise and eigenvector-based interpolation (color plate C. 39, page 267).

6 Segmentation

After computing the topological skeleton for both eigenvector fields, we find the intersections of the red and blue tensorlines. The properties of the resulting segmentation as can be seen in Figure 4 are (a) Cells without degenerate point are quadrangular with two red and two blue tensorlines as boundary, in an alternating order. All red tensorlines passing through this segment enter at one of the blue boundaries and leaves the cell at the opposite boundary and vice versa. All angles are orthogonal. (b) Cells with one degenerate vertex lying in a hyperbolic sector are quadrangular. The angle at the degenerate point is in general not orthogonal. (c) Cells having a degenerate point in one vertex, lying in a parabolic segment, degenerate to a triangular shape. (d) In elliptic sectors, cells with either two or three vertices are possible. (e) Cells containing degenerate lines as edges can exhibit more complicated structures.

7 Results

We tested our method on 2D slices of three different data-sets simulating stress behavior in a solid block: one and two forces applied to the top of a solid block and multiple forces applied to a notched block. Since the results for the eigenvector- and component-wise interpolation schemas are qualitatively similar we focus in the results in close-up views showing the major and typical differences. In Figure 6(d,e) a section of the one-point load containing three degenerate points is shown. While the basic structure is the same, the changes of the eigenvector directions is smoother for the eigenvector-based interpolation, resulting in less curved tensorlines in the vicinity of degenerate points. As a consequence, a proper step size adaption is especially important for component-wise interpolation to obtain tensorlines of the same quality. The calculation of the topological skeleton using eigenvector-based interpolation is in general faster than the one based on the component-based interpolation. This speed gain is a result of restricting the eigenanalysis to the vertices. For the component-wise interpolation it has to be performed for each integration step.

The effect of the re-triangulation on the complexity of the resulting topological structure is shown in Figure 5. These images are close-up views of the one-point load data-set, both using eigenvector-based interpolation. The left image was computed using a Delaunay re-triangulation after vertex insertion combining seven degenerate points in one degenerate line, which nicely represents the dominant radial structure of the red tensorlines. The number of separatrices that have to be integrated is reduced from 35 to 14. Details of the local topological structure are often not features of the data-set, but instead are by-products of the chosen interpolation schema. This is an incentive to keep the resulting topological structure simple while still being consistent with the data.

We have applied our method to data-sets representing the simulation of different force combinations acting on a solid block. Figure 7 shows complete segmentations of a slice of each data-set. In the top right image the cells are randomly colored. The other images displays the blue and red tensorlines bounding the segments. Black dots, lines and triangles show the degenerate entities. The one- and two-point load data-sets are simulated with very low resolution resulting in artifacts that are reflected in

the complicated topological structure. An adaptive finite element method was used in the third dataset which results in a much clearer structure, even though the physical configuration is more complex.

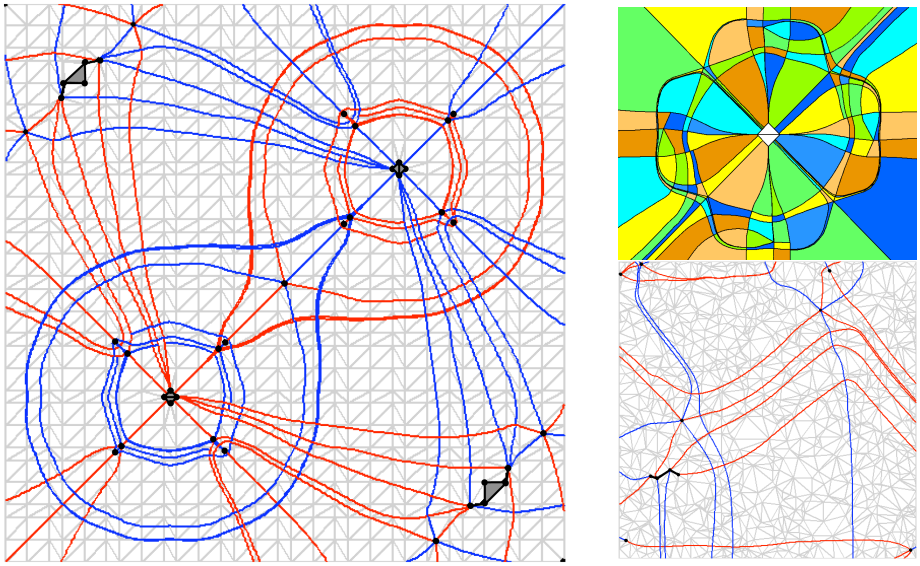


Fig. 7. Full segmentation: left: two-point load, right top: one-point load with randomly colored segments, right bottom: slice of strain simulation of forces on notched block (color plate C. 40, page 268).

8 Conclusions

We have presented a method that delivers a segmentation that capture the eigenvector behavior in a 2D tensor field. With the classification of the sectors we are able to extract separatrices that build the topological structure. Simultaneously the insertion of the degenerated points in the Delaunay triangulation decreases the number of separatrices and therefore the number of segments. The resulting degenerated lines and triangles capture the invariants in the field in a simplified way, without changing the given tensor values. Though the results for the topology extraction of the component-wise interpolation and our eigenvector-based method are qualitatively similar, the latter is faster. Albeit slight differences in the segmentation obtained using both the methods, the quality of the segments is the same. Future work includes a further clean up to simplify the segmentation by combining similar elements and a further refinement of large cells, using the boundary topology.

Acknowledgments This work was supported in part by the German Research Foundation (DFG) through a Junior Research Group Leader award (Emmy Noether Program), and in part by the the National Science Foundation under contract CCF-0702817. We thank our colleagues at the Zuse Institute Berlin and the Institute for Data Analysis and Visualization (IDAV), UC Davis.

References

1. A. Aldroubi and P. Basser. Reconstruction of vector and tensor fields from sampled discrete data. *Contemp. Math.*, 247:1–15, 1999.
2. P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun. Anisotropic polygonal remeshing. *Siggraph'03*, 22(3):485–493, Jul 2003.
3. A. Bhalerao and C.-F. Westin. Tensor splats: Visualising tensor fields by texture mapped volume rendering. In *MICCAI'03*, pages 294–901, 2003.
4. O. Coulon, D. C. Alexander, and S. Arridge. Tensor field regularisation for dt-mr images. In *MIUA01*, pages 21–24, 2001.
5. T. Delmarcelle. *The Visualization of Second-order Tensor Fields*. PhD thesis, Stanford University, 1994.
6. L. Feng, I. Hotz, B. Hamann, and K. Joy. Anisotropic noise samples. *IEEE TVCG*, 14(2):342–354, 2008.
7. R. B. Haber. Visualization techiques for engineering mechanics. *Computing Systems in Engineering*, 1(1):37–50, 1990.
8. I. Hotz, L. Feng, H. Hagen, B. Hamann, B. Jeremic, and K. I. Joy. Physically based methods for tensor field visualization. In *IEEE Visualization 2004*, pages 123–130, 2004.
9. G. Kindlmann. Superquadric tensor glyphs. In *Eurographics Symposium on Visualization*, pages 147–154, May 2004.
10. G. Kindlmann, R. S. J. Estepar, M. Niethammer, S. Haker, and C.-F. Westin. Geodesic-loxodromes for diffusion tensor interpolation and difference measurement. In *MICCAI'07*, pages 1–9, 2007.
11. G. Kindlmann and C.-F. Westin. Diffusion tensor visualization with glyph packing. *IEEE TVCG*, 12(5):1329–1336, 2006.
12. Y. Lavin, R. Batra, L. Hesselink, and Y. Levy. The topology of symmetric tensor fields. *AIAA 13th Computational Fluid Dynamics Conference,*, page 2084, 1997.
13. M. Martin-Fernandez, C.-F. Westin, and C. Alberola-Lopez. 3d bayesian regularization of diffusion tensor MRI using multivariate gaussian markov random fields. In *MICCAI'04*, pages 351–359, 2004.
14. M. Moakher and P. G. Batchelor. Symmetric positive-definite matrices. In *Visualization and Image Processing of Tensor Fields*, pages 285–297. Springer, 2006.
15. G. M. Nielson and I.-H. Jung. Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE TVCG*, 5(4):360–372, 1999.
16. X. Tricoche. *Vector and Tensor Field Topology Simplification, Tracking and Visualization*. PhD thesis, TU Kaiserslautern, 2002.
17. X. Tricoche, G. Scheuermann, H. Hagen, and S. Clauss. Vector and tensor field topology simplification on irregular grids. In *VisSym '01*, pages 107–116, 2001.
18. J. Weickert and M. Welk. Tensor field interpolation with pdes. In *Visualization and Processing of Tensor Fields*, pages 315–324. Springer, 2005.
19. X. Zheng and A. Pang. Hyperlic. In *IEEE Visualization'03*, pages 249–256, 2003.
20. X. Zheng and A. Pang. Topological lines in 3d tensor fields. In *IEEE Visualization'04*, 2004.

Time-Dependent Visualization of Lagrangian Coherent Structures by Grid Advection

Filip Sadlo[1], Alessandro Rigazzi[2], and Ronald Peikert[2]

[1] VISUS, Universität Stuttgart, Germany
sadlo@visus.uni-stuttgart.de
[2] Computer Graphics Laboratory, Computer Science Department,
ETH Zurich, Switzerland {arigazzi, peikert}@inf.ethz.ch

Abstract. Lagrangian coherent structures play an important role in the analysis of unsteady vector fields because they represent the time-dependent analog to vector field topology. Nowadays, they are often obtained as ridges in the finite-time Lyapunov exponent of the vector field. However, one drawback of this quantity is its very high computational cost because a trajectory needs to be computed for every sample in the space-time domain. A focus of this paper are Lagrangian coherent structures that are related to predefined regions such as boundaries, i.e. related to flow attachment and flow separation phenomena. It presents an efficient method for computing the finite-time Lyapunov exponent and its height ridges only in these regions, and in particular, grid advection for the efficient computation of time series of the finite-time Lyapunov exponent, exploiting temporal coherence.

1 Introduction

One of the major challenges in scientific visualization is the visualization of time-dependent velocity fields represented by hundreds of time steps, each given as a large numerical dataset. Velocity fields are among the most important results of computational fluid dynamics (CFD) simulations, and therefore visualization of such data has been extensively studied. It is quite commonly agreed that, due to the complexity of the data, a single visualization technique is in general not able to reveal all the relevant structures in the flow. Structures can not only appear at many different spatial and temporal scales, but their recognition may also depend on the correct frame of observation.

The higher resolution of today's simulation results leads to more and more intricate flow details which are to be captured by appropriate visualization techniques. Methods for such structural visualization can be divided into feature-based and topological approaches. The latter have, until recently, largely been seen as synonymous to *vector field topology* [1, 8]. Because of its rigorous foundation on the theory of dynamical systems, vector field topology is very popular in the visualization community. One of its most powerful concepts is the *separatrix* which separates two regions of qualitatively different flow behavior. Vector field topology can also be applied to the wall shear stress field on no-slip boundaries where the velocity vanishes. By combining it with the topology of the velocity field in the interior, Surana et al. were able to give exact definitions of

separation and attachment surfaces, and they showed that for Navier-Stokes flows, the separation slope and angle formulas depend on on-wall quantities only [19, 20].

In a strict sense, vector field topology is only applicable to steady or instantaneous velocity fields. But the lack of alternatives, the simple concepts, and the availability in visualization systems led to the frequent use of vector field topology also for the visualization of unsteady fields. Even if most researchers were probably aware that such a visualization based only on snapshots cannot be correct, this was mostly seen as a theoretical blemish only. Shadden et al. [17] demonstrated with their simple two-dimensional "double gyre" example that the separatrix can be clearly dislocated from the actual flow separation. As a reaction, there is currently a growing interest in the search for time-dependent variants or extensions of vector field topology. Theisel et al. [21] and Shi et al. [18] presented such concepts for aperiodic and periodic velocity fields, respectively. As a more radical approach, visualization researchers started to look into the theory of *Lagrangian coherent structures* (LCS) as a replacement for vector field topology. In the original sense, an LCS was defined as a region of coherent flow behavior. In Hussain's definition [9] flow behavior is expressed by vorticity alone, while Robinson [14] defined coherent motion as "a region over which at least one fundamental flow variable exhibits significant correlation with itself or with another variable over a range of space and/or time that is significantly larger than the smallest local scales". In a more modern sense, LCS are understood as the boundaries of such regions. As Haller showed [7], they can be computed as *ridges* of the (maximal) *finite-time Lyapunov exponent* (FTLE). These ridges are lower-dimensional structures, which can be classified into attracting and repelling LCS, correspond to the unstable and stable manifolds (separatrices) in vector field topology.

Since the Lyapunov exponent is constant along a trajectory, this holds approximately for its finite-time version if the integration time is chosen to be sufficiently long. Therefore, LCS computed numerically from this quantity are close to material surfaces, i.e. they are essentially advected with the flow. Ideal LCS are material surfaces [7]. For that reason, these structures are of interest for the study of transport and mixing processes in fluid dynamics. In visualization, LCS have been used only recently. Garth et al. [5, 6] visualized the underlying FTLE (scalar) field with slicing and direct volume rendering techniques, using appropriate transfer functions to make LCS recognizable as the ridges of the field. Sadlo et al. [16] compared visualizations based on vector field topology and on LCS, and introduced visualization of the latter by explicit extraction of height ridges of the FTLE field [15]. Bürger et al. [3] computed LCS for the purpose of controlling the seeding in particle based visualizations.

In this paper, we present an efficient method for computing the finite-time Lyapunov exponent and its height ridges as time series. The method maintains a sampling grid that grows and shrinks with the ridges that it contains and that is advected with the flow between the steps of the time series. The grid is initialized by the user in a region of interest which can be located anywhere in the domain. By initializing the grid near a solid boundary, flow separation and attachment surfaces are obtained. An advantage of this visualization method is that it does not rely mainly on the data next to the boundary, and in particular does not need the computation of derivatives in cells adjacent to the boundary.

2 Background

In this section we give a short introduction to the two concepts which are central for this paper, the finite-time Lyapunov exponent and the height ridge, and we briefly discuss practical aspects of their computation from discrete data.

2.1 Finite-Time Lyapunov Exponents

Given a time dependent velocity field $\mathbf{v}(\mathbf{x},t)$ on a domain $D \subseteq \mathbb{R}^n$, a trajectory (or path line) $\mathbf{x}(t;t_0,\mathbf{x}_0)$ starting at point $\mathbf{x}_0$ at time t_0 is a solution of the initial value problem

$$\dot{\mathbf{x}}(t;t_0,\mathbf{x}_0) = \mathbf{v}(\mathbf{x}(t;t_0,\mathbf{x}_0),t), \quad \mathbf{x}(t_0;t_0,\mathbf{x}_0) = \mathbf{x}_0. \tag{1}$$

For fixed times t_0 and t, the trajectories give rise to the *flow map*

$$\phi_{t_0}^t : D \to D, \quad \mathbf{x}_0 \mapsto \mathbf{x}(t;t_0,\mathbf{x}_0). \tag{2}$$

The gradient $\nabla\phi_{t_0}^t$ of the flow map describes the deviation of infinitesimally close trajectories started at the same time t_0, and the tensor

$$\Delta_{t_0}^t = \left(\nabla\phi_{t_0}^t(\mathbf{x}_0)\right)^\top \nabla\phi_{t_0}^t(\mathbf{x}_0) \tag{3}$$

expresses the deformation of the neighborhood of $\mathbf{x}_0$ under the flow map. This symmetric tensor has real eigenvalues λ_i based on which the i-th *Lyapunov exponent* is defined as follows:

$$\sigma_i = \lim_{T\to\infty} \frac{1}{T} \ln \sqrt{\lambda_i(\Delta_{t_0}^{t_0+T})}. \tag{4}$$

The spectrum of Lyapunov exponents is a property of an entire trajectory, i.e. it does not depend on the choice of t_0 on that trajectory. By replacing the limit with a fixed integration time T, the finite-time Lyapunov exponent (FTLE) is obtained. Usually, only the maximum FTLE is of interest, which is given by:

$$\sigma_{t_0}^{t_0+T} = \frac{1}{T} \ln \sqrt{\lambda_{\max}(\Delta_{t_0}^{t_0+T})}. \tag{5}$$

Unlike the Lyapunov exponent, the FTLE depends on both the starting time t_0 and the integration time T.

For the numerical computation of either Lyapunov exponents or FTLE, one has to estimate the flow map gradient by using trajectories started very close to the reference trajectory. However, trajectories may separate at an exponential rate from the reference trajectory. In fact, detecting this behavior is the main motivation behind these concepts. Therefore, trajectories must undergo frequent *renormalization* [2], which is equivalent to breaking up the integration in pieces and computing the flow map gradient as the product of the piece-wise obtained gradients.

The FTLE, and even more the Lyapunov exponents, can exhibit finely detailed structures with a spatial variation that can far exceed that of $\mathbf{v}(\mathbf{x},t)$. Therefore, it is often not the goal to do an accurate computation of an FTLE at a given point in the domain, but rather to compute a spatial average at a resolution defined by a discretization grid. This leads to a discrete version of the FTLE [7] where the flow map is sampled on the nodes of a grid and gradients are then estimated by finite differences (rather than using trajectories in close vicinity and applying renormalization).

2.2 Height Ridges

The notion of a local maximum of a scalar field $s : \mathbb{R}^n \rightarrow \mathbb{R}$ is unambiguously defined by a vanishing gradient and negative second derivatives in all possible directions. In contrast to this, there are several ways of relaxing this definition in order to obtain k-dimensional maxima or minima. The *height ridge* [4] is the most straightforward and the most widely used such definition. For a point on a k-dimensional ridge it requires vanishing first derivatives and negative second derivatives only in a $n-k$-dimensional subspace. Formally, if $\mathbf{H}$ denotes the Hessian of s, $\mathbf{H} = \left(\partial^2 s / \partial x_i \partial x_j\right)_{ij}$, and $y_1, \cdots y_n$ are the unit eigenvectors of $\mathbf{H}$ ordered by the associated eigenvalues $\lambda_1 \leqslant \cdots \leqslant \lambda_n$, then the two conditions for a point on a k-dimensional height ridge are $\partial s / \partial y_1 = \cdots \partial s / \partial y_{n-k} = 0$ and $\lambda_{n-k} < 0$. *Valleys* of s are obtained by applying the height ridge definition to the negative field $-s$.

While the height ridge definition is elegant, practice has shown that the generated features need to be filtered. The purpose of filtering is to remove false positives as well as "weak" features. For the case of 1-dimensional ridges, several such filters are known which can be used alone or in combination [13].

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

$$s \geq s_{min}. \tag{6}$$

In the case of FTLE ridges, the effect of this filter is to suppress ridges with low separation property. The reader is referred to [16] for further details on the influence of this filtering criterion.

A related filtering criterion would be to prescribe a maximum for the second derivative λ_n across the ridge, which results in suppressing regions with too "flat" ridge property:

$$\lambda_n \leq \lambda_{max}. \tag{7}$$

In the case of FTLE ridges this filter is relevant, since the "sharpness" of an FTLE ridge was shown [17] to be a measure for the flux across an LCS, i.e. the quality of an LCS as a flow barrier.

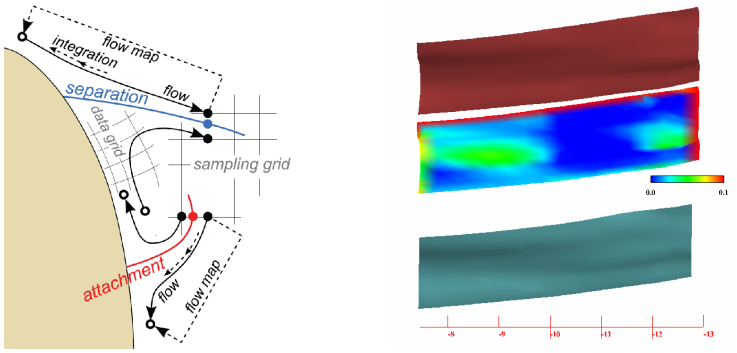
For a reliable and temporally coherent visualization, it is important that criteria such as (6) or (7) are not implemented as binary "pass/fail" filters. By allowing for a certain ratio of exceptions per neighborhood, holes and popping artifacts can be avoided to a large extent. But there is also the problem of noisy ridge extraction results containing many small ridges. These are not necessarily removed by (6) or (7) because the ridge might be sharp and at a high field value. Furthermore, the application of filters can generate additional small ridges. Therefore it is important to filter the ridges also by their size, which requires a connected component labeling of the set of ridges.

3 Motivation

This paper, i.e. the adaptation of the uniform sampling grid to the regions containing ridges, is motivated by the work by Sadlo et al. [15]. The goal is to optimize the computation of time series of FTLE ridges to make the method more applicable in every-day

applications in research and engineering. The increase in efficiency is achieved by restricting the computation to regions that contain the LCS of interest, and, in particular, by exploiting the temporal coherence of unsteady vector fields for the computation of time series of FTLE by advection of the sampling grid.

One of the application goals of this paper is to offer a method for the analysis of unsteady flow separation and attachment. Separation phenomena are the cause of many undesired effects in engineering, such as development of recirculation zones, reduced throughput, reduced lift, vortex generation, lowered mixing, and reduced flow control in general. Flow separation exhibits diverging trajectories in backward time and flow attachment exhibits diverging trajectories in forward time (see Figure 1a). This fact is the reason why separation and attachment lines (or points) are usually accompanied by corresponding LCS and why these processes are amenable to an analysis by FTLE ridges. Shadden et al. [17] have already demonstrated the utility of those ridges for the analysis of unsteady flow separation in their example of flow separation over an airfoil. We also believe that an analysis based on LCS provides a deeper and more precise insight into these unsteady phenomena compared to standard techniques such as stream surface integration or particle tracing.



(a) (b)

Fig. 1. (a) Flow separation and flow attachment. The unstable manifold (blue) attracts the fluid along the boundary and guides it into the interior of the domain whereas the stable manifold (red) guides the fluid in opposite direction. (b) Intake dataset. Comparison of ridge from advected grid (red) and uniform grid (blue) together with ridge from advected grid color-coded by distance to ridge from uniform grid (colored) (color plate C. 41, page 268).

4 Method

The proposed method can be subdivided into two parts: one that constrains the sampling grid to filtered ridges of interest at a given time step (or the only time step in case of steady vector fields), described in Sections 4.1 and 4.2, and one that exploits temporal

coherence to speed up the computation of time series of quantities computed from trajectories, such as the FTLE, in the case of unsteady vector fields (Section 4.3). The time series are obtained by variation of the t_0 parameter of the FTLE (compare Section 2.1).

Algorithm 2 describes the methods presented in Sections 4.1 to 4.3 for the case of FTLE ridge extraction. However, it also holds for ridges of other quantities based on trajectories and could be easily modified for the scalar quantity itself instead of its ridges. If the quantity is not computed using local operators such as gradients, larger distortions may be acceptable and hence longer advection times could be used, leading to a further increase in speedup.

4.1 Grid Initialization

In the filtered AMR ridge extraction method [15] the complete domain (or region of interest) is sampled at low resolution and the sampling is adaptively refined in regions containing filtered ridges. Although this results in a speedup compared to a uniform sampling at the finest subdivision level, the method suffers from several drawbacks when applied to quantities that can not be evaluated in a point-wise manner but that are computed using local operators, such as gradients in the case of FTLE ridges. The main problem here is that the value is inherently sampling dependent because the gradient can be underestimated if the sampling is too coarse. Together with a restrictive threshold for the ridge filtering this sometimes results in missed ridges, because they are not detected in the coarse sampling and hence the corresponding regions are not refined which would capture the ridges later on. The remedy is either to use a finer initial sampling, a lower threshold for filtering, or to increase the look-ahead count (Section 3.1 of [15]), all leading to an increased number of samples and hence lowered speedup. See also Section 3.2 of [15] for further information on the implications for quantities based on local operators.

In the present approach one typically avoids sampling the whole domain (or region of interest). Instead, we require initial sampling regions that already capture part of the ridges (cf. Figures 4a and 5a) and adapt the sampling regions to the present ridges (Figures 4b and 5b). This allows to use initial samplings of sufficient resolution and avoids the need for lowered filtering thresholds. In the case of FTLE analysis of flow separation and attachment, possible ways for choosing the initial regions include:

- Definition from special regions of the simulation mesh, e.g. the complete boundary of the domain, or a subset thereof such as the blades of a turbine. These regions are often explicitly available from the simulation file formats.
- Automatic definition by quantities such as “surface divergence” or its local maxima as presented by Tricoche et al. [22].
- Automatic definition by features. A possibility is to extract separation and attachment lines according to Kenwright [11] or Tricoche et al. [22] and to place initial sampling regions around (part of) those.
- Manual identification and definition by preceding interactive exploration using standard techniques such as path line integration or (AMR) ridge extraction [15] of the FTLE in regions of interest. It might seem cumbersome to extract FTLE ridges in a first step with a standard technique, but this can be afforded if the goal is to compute time series of FTLE (Section 4.3).

We require the initial sampling regions and resampled regions (Section 4.3) to be parts of a virtual uniform grid that covers the complete domain. This makes sure that separated grids are consistently sampled and hence can merge (even after advection) when cells are added by the procedure described in Section 4.2.

4.2 Grid Adaptation

This section describes how the initial sampling grid from Section 4.1 is adapted to the filtered ridges (cf. Figures 4b, 4d, 5b, and 5d). To prevent long extraction times in cases where the ridges extend into regions that are of no interest to the user, a region of interest can be defined which restricts the adaptation, possibly leading to truncated ridges.

Grid Growing The first adaptation step is to add new cells to the boundary of the sampling grid where necessary. We define a *ridge cell* to be a cell that has an edge intersected by a filtered ridge according to (6) or (7). Because we aim at results that are identical to those from a uniform sampling, the support range of the Hessian, which is needed for the height ridge extraction, has to be taken into account. If the underlying scalar quantity is computed using a local operator (as in the case of FTLE), its support radius has to be added to that of the Hessian as well. Having the total support range, one needs to add all cells to the grid that are within that topological neighborhood of any ridge cell.

In cases of steady vector fields, where the grid advection from Section 4.3 does not apply, the sampling grid is uniform and adding cells is a trivial procedure. However, if the grid is advected, adding cells is a challenging problem due to the distortion of the grid. Nevertheless, the initial grid is uniform and the grid gets uniformly resampled from time to time. If we need to add a cell to the distorted grid, we simply go back to the last time step where the grid was uniform, add the nodes of the corresponding cell there and advect the added nodes to the actual time step. This makes the cell fit to the desired position. Additionally, the computed trajectories for the advection of the nodes can be reused for computing the quantity (FTLE), resulting in little overhead.

However, if a node of the cell in the uniform grid is located outside of the domain, there is no vector field that could be used to advect it to the desired timestep and position. In this case the cell can be constructed by extrapolation of the grid or any standard meshing technique. The grid growing procedure is iterated until convergence, meaning that each added cell and its neighbors are tested for being a ridge cell and if this is the case, it is attempted to add the cells inside the neighborhood range. This way, the sampling grid grows to the necessary extent.

Grid Shrinking The next step is to remove unnecessary cells from the grid. These are cells that are neither ridge cells nor in the relevant neighborhood of any ridge cell, or cells where one or more nodes are outside of the domain.

4.3 Grid Advection

Lagrangian coherent structures are material lines or material surfaces [7], in other words, they get advected with the flow, such as streak lines (surfaces) and time lines

(surfaces). This would allow to exploit temporal coherence for the generation of time series of FTLE ridges by advection of the extracted ridges. One could compute the ridges only after every n time steps and advect them with the flow in between. However, this would not account for changes of the FTLE during advection. New ridges can originate and existing ones can grow, shrink, or disappear, especially if the ridges are filtered by the FTLE value as in our case. Therefore we propose a different approach: the advection of the sampling grid itself during the advection intervals. This results in a generic method for quantities based on trajectories, not only FTLE.

During advection, a short trajectory has to be computed for each node of the sampling grid to advance it to the next position. The striking advantage is that these short trajectories can be appended to the existing trajectories needed for the computation of the FTLE, making it possible to reuse large parts of the trajectories and hence improving efficiency, see Figures 4c and 5c.

As already mentioned, advection of the sampling grid tends to distort its cells and this in turn tends to affect the computation of derivatives, which are needed for FTLE computation and ridge extraction. Additionally, the FTLE tends to be sampling dependent. All in all this generally leads to artifacts in the extracted FTLE ridges such as deformation, false negatives, and even false positives.

To restrict the artifacts to an acceptable level, the FTLE is periodically resampled on a subset of the virtual uniform grid spanning the whole domain: only those cells of the grid are generated (and the corresponding trajectories are computed) which overlap with the advected grid or which are contained in the region of the initial sampling. An additional strategy is to place the sampling grid outside regions producing high distortion such as wakes and vortices. Although this looks like a compromise, it is often a natural choice to analyze LCS away from disturbing phenomena since they would also distort them, even when uniformly sampled, and hence complicate their interpretation.

Because the flow map is computed by integrating trajectories in numerical vector fields and because of the intricacy of gradient computation on unstructured grids, aside from the difficulty to provide an error measure between the ridges extracted from the distorted grid and those extracted from a corresponding uniform grid, it is generally not possible to provide error bounds regarding the distortion of the grid. Garth et al. [5] measure the error for their subdivision scheme in the flow map. Similarly, we propose to measure the error based on the FTLE, not its ridges, and to use it for triggering the resampling procedure.

The grid is uniformly resampled (recomputing the trajectories and the FTLE) after every r advection steps with an initial value of $r = 2$. After the FTLE has been computed on the resampled grid, the FTLE of the advected grid is interpolated at the node positions of the new grid and the root mean square (RMS) of the difference over all nodes is computed. The RMS is then compared to a user-defined tolerance and a new r is estimated from the RMS and from the tolerance by linearization of the RMS over the advection steps (line 40 of the algorithm). The algorithm then proceeds to the next advection phase. However, the linearization of the error can fail in the sense that r is chosen too large such that after the next r advection steps the RMS exceeds the prescribed tolerance. One solution to this problem is to enforce the tolerance by reducing r (and hence taking back advection steps) until the RMS tolerance is fulfilled. How-

ever, experiments have shown that it is usually sufficient not to enforce the tolerance and instead to prescribe a reduced tolerance, e.g. by 15 percent, to satisfy the intended precision.

To support the user in an appropriate choice of the RMS tolerance and the sampling region, we allow visualization of both sets of ridges, those before and after resampling, or color-coding the former ones by their distance to the latter ones as in Figure 1b, serving as uncertainty information. Another approach is to judge the popping artifacts visually when moving from a time step based on an advected grid to a subsequent time step where the grid was uniformly resampled.

Note that for the analysis of separation, time series of FTLE ridges have to get generated by advecting the grid in positive time direction (Figure 4), whereas for attachment the grid has to get advected in negative-time direction (Figure 5). This is necessary since the ridges are captured at the regions of interest (at the boundaries) and FTLE ridges for attachment approach the boundary in positive time. Hence it is necessary to start with the last time step and to compute them in negative time direction in order to capture all of them at the boundary, even those that separate from the boundary. So finally, all ridges (LCS) that come in contact with the boundary (or region of interest) at any time will get captured.

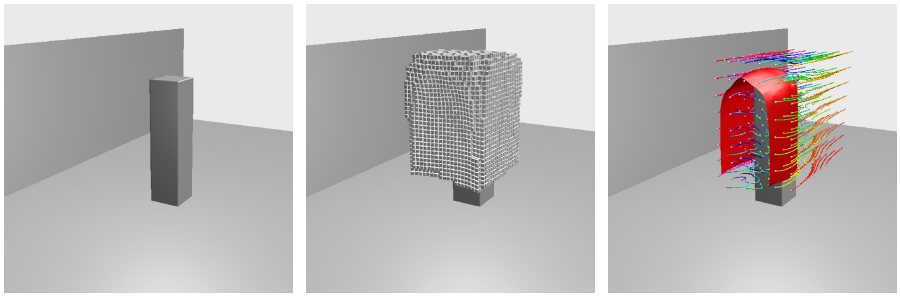
5 Results

In this section the presented method is applied to two unsteady CFD simulations. The first one examines the flow around a cuboid, exhibiting flow similar to a von Kármán vortex street, but in this case the vortices become tilted soon after they detach from the cube. This leads to flow separation behavior that differs from the standard von Kármán vortex street. Secondly, the method is applied to a simulation of an intake of a river power plant. The scope there is a construct that prevents salmon from getting into the runner of the turbine.

5.1 Flow around a Cuboid

This example produces a kind of a von Kármán vortex street. The unsteady flow comes from the right back and follows to the left front (Figure 2a). The main difference to a common von Kármán vortex street is that there is also flow over the “top” face of the cuboid. The flow separation at the cuboid is the subject of analysis in this case. The resulting FTLE ridge (Figure 2c) shows that flow separation is in progress on both sides and on the top of the cuboid. It can be seen that the FTLE ridge separates the vortex street region from the outer flow. However, further downstream the FTLE ridge does not exhibit this property anymore: it crosses the vortices. Time series of FTLE ridges reveal that the separation zones are oscillating consistently with the von Kármán vortex street.

Table 1 shows some performance details for this example. The achieved speedup in this case is only about 2.3. This is due to the relatively short trajectories. The prescribed RMS tolerance was 15.0 and at step 33 this was exceeded by 0.88 percent. There have been 12 advection phases, each performed 5 advection steps in average. Because of the



(a) (b) (c)

Fig. 2. Flow around a cuboid. (a) Geometry. (b) Sampling grid adapted to ridge region and advected. (c) Resulting FTLE ridge with some upstream trajectories (colored) from uniform grid, and their seeds (white spheres) (color plate C. 42, page 269).

	uniform	grid advection
grid [nodes]	16399	14220 (step 33)
flow map [s]	13704.87	2944.88
total [s]	13707.92	5800.31
speedup	1	2.36
Figure		2b

Table 1. Performance analysis for the cuboid dataset. 60 steps of grid advection compared to 61 direct evaluations on uniform grid. See also Figure 2b.

	uniform	direct on adapted grid	grid advection
grid [nodes]	8800	5007	3913 (step 39)
flow map [s]	15369.17	9062.73	355.62
total [s]	15374.22	9489.96	1026.81
speedup	1	1.62	14.97
Figure	3a		3b

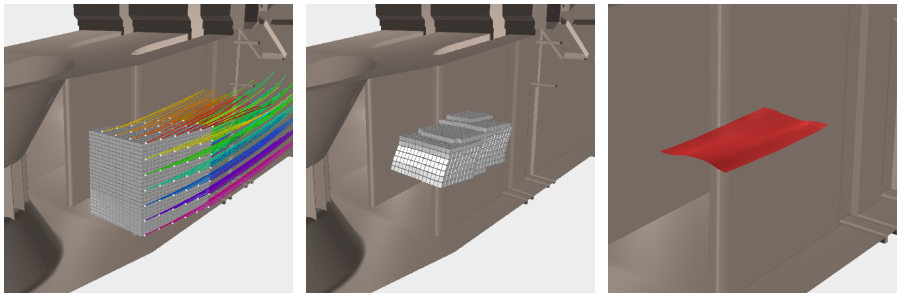
Table 2. Performance analysis for the turbine intake dataset. 100 steps of grid advection compared to 101 direct evaluations (on uniform grid and adapted grid). See also Figures 3a and 3b.

shape of the FTLE ridge and because the initial sampling grid is already well adapted to the FTLE ridge, the expected speedup from the grid adaptation is small and was therefore not measured.

5.2 Intake of a Power Plant

The underlying data of this section is an existing run-of-river plant in the US. All devices shown are installed in the intake of the plant in order to protect juvenile salmon from passing through the runner. The water flow of the unsteady CFD simulation comes from the right back and follows to the left front where it enters the turbine (Figure 3).

The horizontal rods at the right hand side of the image lead the salmon into the vertical channels at the top in the installation. However, these rods produce a noticeable wake in the upper part of the main channel (see path lines in Figure 3a). Additionally, the backflow from the salmon channel (the opening at the top downstream from the rods) also is involved in a recirculation zone at the top wall, located above the sampling grid of Figure 3a. On the one hand, a FTLE ridge was extracted using a regular grid at the confluence of the three main channels (Figure 3a), on the other it was extracted using the presented grid advection method (Figures 3b and 3c). The obtained FTLE



(a) (b) (c)

Fig. 3. Intake of a water turbine. (a) Uniform grid with some of the upstream trajectories (colored) used for FTLE computation, and their seeds (white spheres). (b) Sampling grid adapted to ridge region and advected. (c) Resulting FTLE ridge (color plate C. 43, page 269).

ridge separates well the fast flow at the bottom of the channel from the slower flow in the upper half of the channel.

Table 2 shows some performance measurements of the presented case. The speedup from the grid adaptation is quite low (1.62) because of the relatively low resolution of the sampling grid and because the sampling region was already quite well adapted to the ridge. The speedup from including grid advection is significantly higher (about 15) and would further increase with increasing the integration time for the trajectories. The RMS tolerance was set to 0.012 and at step 39 this was exceeded by 14.2 percent, which was the maximum during the 13 advection phases. Figure 1b shows the corresponding distance error of the ridge. In average, 7.7 advection steps were performed per advection phase.

6 Conclusion

We presented a generic method for accelerating the computation (of time series) of quantities based on trajectories, such as FTLE. On the one hand the efficiency is improved by restricting the sampling grid to the phenomena of interest, on the other hand and more important, the computation is accelerated by reusing part of the trajectories, which is made possible by advection of the sampling grid. In the case of gradient-based visualizations, such as FTLE ridges, the quality tends to suffer if the distortion caused by the advection of the grid is high. Therefore, the obtained quality is inferior to evaluations on regular grids or that by Sadlo et al. in terms of quality, but superior in terms of speed if time series of FTLE ridges are computed. A comparison to the approximative method by Garth et al. deduces from a comparison of that method to FTLE samplings on a regular grid. All in all we propopse to use the method at least as a fast preview technique and to use low RMS error thresholds (leading to low acceleration) or even exact methods, such as direct computation on uniform grids or that by Sadlo et al. [15], if exact time series are required. Future work could include local strategies for reducing the distortion of the grid and thus lowering the frequency at which resampling is needed. We would like to thank Sulzer Innotec for the cuboid dataset and VATECH Hy-

dro for the intake dataset. This work was funded by Swiss Commission for Technology and Innovation grant 7338.2 ESPP-ES.

References

1. D. Asimov. Notes on the Topology of Vector Fields and Flows. Technical Report RNR-93-003, NASA Ames Research Center, 1993.
2. G. Bennetin, L. Galgani, A. Giorgilli, and J. Strelcyn. All Lyapunov exponents are effectively computable. *Physical Review A*, 14:2238, 1976.
3. K. Bürger, P. Kondratieva, J. Krüger, and R. Westermann. Importance-Driven Particle Techniques for Flow Visualization. In *Proceedings of IEEE VGTC Pacific Visualization Symposium 2008*, pages 71–78, March 2008.
4. D. Eberly. *Ridges in Image and Data Analysis. Computational Imaging and Vision*. Kluwer Academic Publishers, 1996.
5. C. Garth, F. Gerhardt, X. Tricoche, and H. Hagen. Efficient Computation and Visualization of Coherent Structures in Fluid Flow Applications. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1464–1471, 2007.
6. C. Garth, G.-S. Li, X. Tricoche, C. Hansen, and H. Hagen. Visualization of Coherent Structures in Transient 2D Flows. In *Topology-based Methods in Visualization II*. Springer, pages 1–13, 2008.
7. G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D*, 149:248–277, 2001.
8. J. Helman and L. Hesselink. Representation and Display of Vector Field Topology in Fluid Flow Data Sets. *IEEE Computer*, 22(8):27–36, 1989.
9. F. Hussain. Coherent structures and turbulence. *Journal of Fluid Mechanics*, 173:303–356, 1986.
10. J. Jeong and F. Hussain. On the identification of a vortex. *Journal of Fluid Mechanics*, 285(69):69–94, 1995.
11. D. N. Kenwright. Automatic detection of open and closed separation and attachment lines. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 151–158, Los Alamitos, CA, USA, 1998.
12. K. J. Lockey, M. Keller, M. Sick, M. H. Staehle, and A. Gehrer. Flow induced vibrations at stay vanes: Experience at site and CFD simulation of von Kármán vortex shedding. In *Proceedings of Hydro2006*, pages 25-28, 2006.
13. R. Peikert and F. Sadlo. Height Ridge Computation and Filtering for Visualization. In I. Fujishiro, H. Li, and K.-L. Ma, editors, *Proceedings of IEEE VGTC Pacific Visualization Symposium 2008*, pages 119–126, March 2008.
14. S. K. Robinson. Coherent motions in the turbulent boundary layer. *Annu. Rev. Fluid Mech.*, 23:601–639, 1991.
15. F. Sadlo and R. Peikert. Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1456–1463, 2007.
16. F. Sadlo and R. Peikert. Visualizing Lagrangian Coherent Structures and Comparison to Vector Field Topology. In *Topology-based Methods in Visualization II*. Springer, pages 15–30, 2008.
17. S. C. Shadden, F. Lekien, and J. E. Marsden. Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D Nonlinear Phenomena*, 212:271–304, Dec. 2005.

18. K. Shi, H. Theisel, T. Weinkauf, H. Hauser, H.-C. Hege, and H.-P. Seidel. Path Line Oriented Topology for Periodic 2D Time-Dependent Vector Fields. In *Proc. Symposium on Visualization (EuroVis '06)*, pages 139–146, 2006.
19. A. Surana, O. Grunberg, and G. Haller. Exact theory of three-dimensional flow separation. Part I: Steady separation. *J. Fluid Mech.*, 564:57-103, 2006.
20. A. Surana, G. Jacobs, and G. Haller. Extraction of Separation and Reattachment Surfaces from 3D Steady Shear Flows. *AIAA Journal*, 45(6):1290–1302, 2007.
21. H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Stream Line and Path Line Oriented Topology for 2D Time-Dependent Vector Fields. In *IEEE Visualization*, pages 321–328, 2004.
22. X. Tricoche, C. Garth, and G. Scheuermann. Fast and Robust Extraction of Separation Line Features. In *Scientific Visualization: The Visual Extraction of Knowledge from Data*, Berlin, 2005. Springer.

Algorithm 2 Grid Advection for FTLE Ridges

```
1: steps: number of steps for the FTLE ridge time series
2: range: topological neighborhood range around ridges
3: tolerance: tolerance for RMS of FTLE
4:
5: place initial sampling grid
6: compute all trajectories and compute FTLE
7: ℛ ← detect ridge cells from FTLE
8: 𝒩 ← cells in range around ℛ  // 𝒩 may contain existing cells and cells to add
9:
10: // compute frames of FTLE time series
11: r ← 2  // number of advection steps
12: lastResampleStep ← 0
13: for step=1 to steps do
14:     // grid growing
15:     while first iteration at step or grid changed do
16:         // add cells in neighborhood range around ℛ
17:         for all cells c ∈ 𝒩 and not yet in sampling grid do
18:             add c directly if grid regular, or by advection or meshing
19:         end for
20:         compute (or reuse) trajectories and compute FTLE
21:         ℛ ← detect ridge cells from FTLE
22:         𝒩 ← cells in range around ℛ
23:     end while
24:
25:     // grid shrinking
26:     for all cells c of sampling grid do
27:         if c outside domain or c ∉ (ℛ ∪ 𝒩) then
28:             remove c
29:         end if
30:     end for
31:
32:     // grid advection
33:     if step < steps then
34:         advect grid nodes to next time step
35:         compute (or reuse) trajectories and compute FTLE
36:         // resampling
37:         if step − lastResampleStep > r then
38:             resample uniformly, recompute all trajectories and compute new FTLE
39:             RMS ← measure RMS between old FTLE and FTLE on resampled grid
40:             r ← max(1, ⌊r∗ tolerance / RMS⌋)
41:             lastResampleStep ← step
42:         end if
43:         ℛ ← detect ridge cells from FTLE
44:         𝒩 ← cells in range around ℛ
45:     end if
46: end for
```

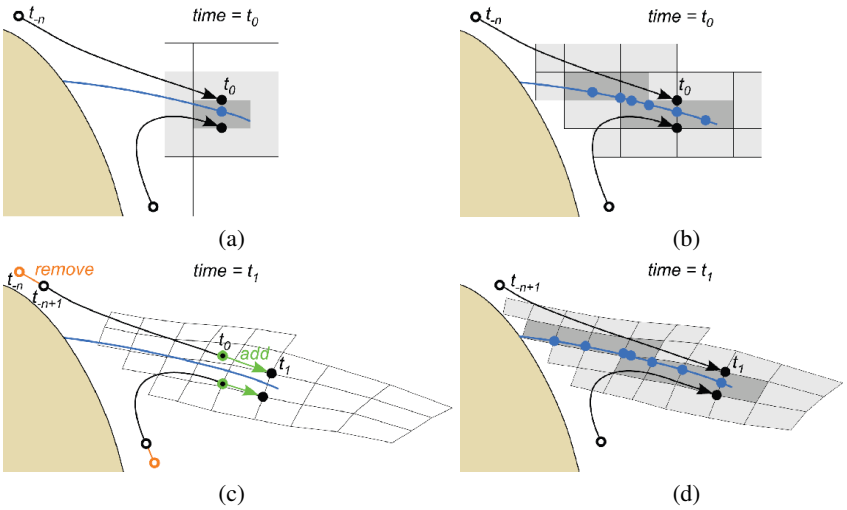


Fig. 4. Grid advection for flow separation. (a) Initial sampling grid. Ridge cells (dark gray) and their neighboring cells (light gray). Cell edge intersected by negative-time FTLE ridge (blue point). Neighborhood range is 1 for illustration purposes. (b) After grid adaptation. (c) After one step of grid advection. (d) After grid adaptation of advected grid (color plate C. 44, page 270).

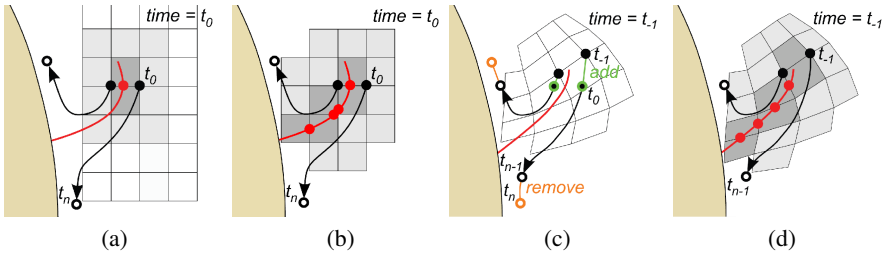


Fig. 5. Grid advection for flow attachment. (a) Initial sampling grid. Ridge cells (dark gray) and their neighboring cells (light gray). Cell edge intersected by positive-time FTLE ridge (red point). Neighborhood range is 1 for illustration purposes. (b) After grid adaptation. (c) After one step of grid advection. (d) After grid adaptation of advected grid (color plate C. 45, page 270).

Practical Considerations in Morse-Smale Complex Computation

Attila Gyulassy[1,2], Peer-Timo Bremer[2], Bernd Hamann[1], and Valerio Pascucci[3]

[1] Institute for Data Analysis and Visualization (IDAV),
Department of Computer Science, University of California, Davis, CA 95616
[2] Center for Applied Scientific Computing,
Lawrence Livermore National Laboratory
P.O. Box 808, L-561, Livermore, CA 94551
[3] Scientific Computing and Imaging Institute, University of Utah
72 S Central Campus Drive, 3750 WEB, Salt Lake City, UT 84112

Abstract. The Morse-Smale complex is an effective topology-based representation for identifying, ordering, and selectively removing features in scalar-valued data. Several algorithms are known for its effective computation, however, common problems pose practical challenges for any feature-finding approach using the Morse-Smale complex. We identify these problems and present practical solutions: (1) we identify the cause of spurious critical points due to simulation of simplicity, and present a general technique for solving it; (2) we improve simplification performance by reordering critical point cancellation operations and introducing an efficient data structure for storing the arcs of the complex; (3) we present a practical approach for handling boundary conditions.

1 Introduction

Scientific data is becoming increasingly complex, and sophisticated techniques are required for its effective analysis and visualization. The Morse-Smale (MS) complex is an efficient data structure that represents the complete gradient flow behavior of a scalar function, and can be used to identify, order, and selectively remove features. Although several algorithms are known for its computation, achieving an efficient implementation is still a challenge. Key optimizations remain un-addressed in the literature without which computation of the complex might have a memory footprint that grows past practical limits, simplification times that take days instead of seconds, and undesirable artifacts on the boundaries. We examine the causes of such problems and present the necessary techniques for overcoming them.

2 Related Work

The MS complex is a topological data structure that provides an abstract representation of the gradient flow behavior of a scalar field [12, 13]. Edelsbrunner et al. [3] defined the MS complex for piecewise-linear 2-manifolds by considering the PL function as the limit of a series of smooth functions, and used this interpretation to transfer ideas from

the smooth case. They also provided an efficient algorithm to compute the MS complex, restricted to edges of the input triangulation, and construct a hierarchical representation by repeated cancellation of pairs of critical points. Bremer et al. [1] improved the algorithm and described a multi-resolution representation of a scalar field. Both algorithms trace paths of steepest ascent and descent beginning at saddle points. These paths constitute boundaries of two-dimensional cells of the MS complex.

Cells in the MS complex of a trivariate scalar field can be of dimension zero, one, two, or three. Tracing boundaries of the three-dimensional cells while maintaining a combinatorial valid complex is a difficult task and a practical implementation of such an algorithm remains a challenge [2]. Nevertheless, the MS complex has been computed for trivariate scalar field data and successfully used to identify features through repeated application of atomic cancellation operations [7]. Computation of the complex in this manner makes necessary a preprocessing step that subdivides every voxel by inserting "dummy" critical points, and therefore has a large computational overhead. This approach was improved by using a sweeping plane [8], but data size and computational overhead still turned out to be a limiting factor. An algorithm based on region-growing [9] was introduced for simplicial meshes of three dimensions, with a tenfold improvement in efficiency, however, the need to store the ascending and descending manifold membership information at each cell of the input, and the requirement to represent the entire output explicitly limits the scalability of this approach.

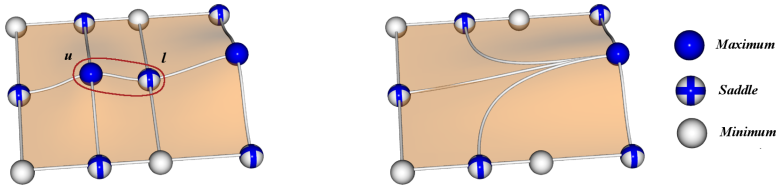
Discrete Morse theory, as presented by Forman [5], has also been used to compute the MS complex. Lewiner et al. [11] showed how a discrete gradient field can be constructed and used to identify the MS complex, however, this construction requires modification of the input mesh and an explicit representation of gradient paths, restricting the applicability of the method. King et al. [10] presented a method for constructing a discrete gradient field that agrees with the large-scale flow behavior of the data defined at vertices of the input mesh. Gyulassy et al. [6] presented an algorithm to compute the MS complex for data of any dimension in a memory-efficient manner by subdividing the data into parcels, computing the discrete gradient and complex on each parcel, and gluing the complex of each parcel back together using the discrete gradient flow across the boundaries. This method was the first scalable algorithm published for constructing MS complexes for large data. We use the slicing version of this algorithm to generate memory and run-time statistics is this paper.

3 Background

We present some basic definitions and an explanation of cancellations in the Morse-Smale complex.

3.1 Morse Functions and the Morse-Smale Complex

A real-valued smooth map $f : \mathbb{M} \to \mathbb{R}$ defined over a compact d-manifold $\mathbb{M}$ is a *Morse function* if all its critical points are non-degenerate (*i.e.*, the Hessian matrix is non-singular for all critical points) and no two critical points have the same function value. An integral line of f is a maximal path in $\mathbb{M}$ whose tangent vectors agree with the



(a) (b)

Fig. 1. The circled arc connects a saddle l to a maximum u (a). cancellation of (l,u) removes all arcs attached to l or u, and creates new arcs from the lower neighbors of u to the upper neighbors of l (b) (color plate C. 46, page 270).

gradient of f at every point of the path. Each integral line has a natural origin and destination at critical points of f where the gradient becomes zero. *Ascending* and *descending* manifolds are obtained as clusters of integral lines having common origin and destination respectively. The *Morse-Smale (MS) complex*, denoted Γ, is a partition of $\mathbb{M}$ into regions clustering integral lines that share common origin and destination. In Morse-Smale functions, the integral lines only connect critical points of different indices.

Each critical point of index n is the origin of a set of integral lines that forms an ascending $d-n$-manifold. Symmetrically, it is the destination of a set of integral lines that forms a descending n-manifold. All ascending and descending manifolds of a Morse-Smale function intersect transversely. Therefore, given two critical points a and b, where the index of a is one less than the index of b, the intersection of the ascending manifold of a and the descending manifold of b is either empty or a 1-manifold. The critical points and these 1-manifolds are called *nodes* and *arcs*. The one-skeleton formed by the nodes and arcs forms the *combinatorial structure* of the MS complex. The combinatorial structure contains much of the semantic information of f, and is useful for simplification and feature identification. The *neighborhood* of a node a of an MS complex Γ is the set of nodes N_a that are connected to a by an arc in Γ.

3.2 Cancellations

A function f is simplified by repeated cancellation of pairs of critical points. The local change in the MS complex leads to a smoothing of the gradient vector field and hence of the function f. A cancellation operation is *valid* (*i.e.*, it can be realized by a local perturbation of the gradient vector field) for a pair of critical points if and only if there exists exactly one arc connecting them in the complex. Therefore, the indices of the two critical points must differ by one. Also, any critical point pair that is connected by multiple arcs represents a configuration known as a *strangulation* or a *pouch*, for which there is no direct perturbation of the gradient that removes the critical point pair. The atomic cancellation operations that are the basis for simplifying an MS complex were characterized in Gyulassy et al. [6] for complexes of general dimensions:

Cancellation: Let Γ be an MS complex for a scalar function defined on a closed d-manifold $\mathbb{M}$. Let l and u be the lower and upper nodes of an arc a in Γ, with indices i

and $i+1$, respectively. Let A_l be the set of arcs that have l as one end point, A_u the set of arcs that have u as one end point, N_l the set of nodes in the neighborhood of l, and N_u the set of nodes in the neighborhood of u.

The *combinatorial cancellation* of (l,u) changes the combinatorial structure of the MS complex and is based on these two steps:

1. Creation of a new arc connecting every critical point of indices $i+1$ in N_l to every critical point of index i in N_u, and adding them to Γ.
2. Removal of arcs in A_l and A_u, and removal of l and u from the complex.

Figure 1 shows this operation for a two-dimensional MS complex. The cancellation operation creates and destroys several arcs in the MS complex, and therefore an efficient data structure representing the nodes and arcs is necessary to handle the operations involved. Let $a = (l,u)$ be an arc that is canceled, n be the number of critical points of index $i+1$ in N_l, and m be the number of critical points of index i in N_u. In the fist step of canceling a, $n \times m$ new arcs are added to the MS complex. Each arc that is created must be inserted into the set of arcs of the nodes at its endpoints, resulting in $O(n \times m)$ INSERT operations. In the second step, the arcs connecting with the canceled nodes are removed from the complex. Each arc that is removed from the complex must be deleted from the set of arcs of the nodes at its endpoints, resulting in $O(|A_l| + |A_u|)$ DELETE operations. Therefore, a data structure for storing the arcs at each node must support efficient INSERT and DELETE operations.

In previous implementations [6, 7, 9], all arcs connected to a node were stored in a linked list. INSERT operations were performed in constant time, however, the DELETE operations were linear time in the number of arcs $|A_n|$ connected to a node n. Therefore, the total running time for a cancellation was $O((n \times m) + ((|A_l| + |A_u|) \times A_m))$ where m is the node with the largest number $|A_m|$ of connected arcs in $N_u \cup N_l$.

The first term in this running time can lead to a quadratic increase in the number of arcs in the complex: not only does it create $n \times m$ new arcs, but every index-i node in N_u has m arcs added to its set of connected nodes, and every index-$(i+1)$ node in N_l has n arcs added. Therefore, future cancellations between pairs of these nodes become more costly. Therefore, performing k cancellations in a naive ordering can therefore lead to an actual $kn \times km$ cost. Such situations often arise when large flat regions have many adjacent 0-persistence arcs, as is the case in most integer-valued data. The second term in the running time can also be prohibitive, as nodes with a large number of connected arcs (high valence) slow the cancellation of every node that is connected to it. In fact, a linear-time DELETE operation in the number of arcs connected to a node leads to quadratic-time cancellations in the number of arcs connected to each endpoint.

3.3 Data Sets

The timing and memory performance statistics presented throughout this paper were generated on an off-the-shelf 2.21GHz AMD Athlon processor with 2.0Gb memory. Data sets were chosen to stress particular aspects of the MS complex construction algorithm. The data sets used and the particular challenge posed by each are summarized in table 1.

Name	Size	Values	Description	Challenge
Artificial	$64 \times 64 \times 64$	Byte	An artificially generated data set where every cell is critical and all arcs have zero persistence	No implicit ordering of arcs for cancellation
Hydrogen	$128 \times 128 \times 128$	Byte	The spatial probability distribution of an electron around a hydrogen atom in a strong magnetic field	Large flat regions with simple persistent features
Aneurism	$256 \times 256 \times 256$	Byte	A rotational X-ray scan of an aneurism	Flat regions interspersed with noise
Porous	$230 \times 230 \times 325$	Float	Distance to an interface surface in a simulated porous solid	Noisy floating-point data with flat ridge lines

Table 1. The data sets used for the generation of the timing data presented in this paper were selected to examine particular aspects of performance.

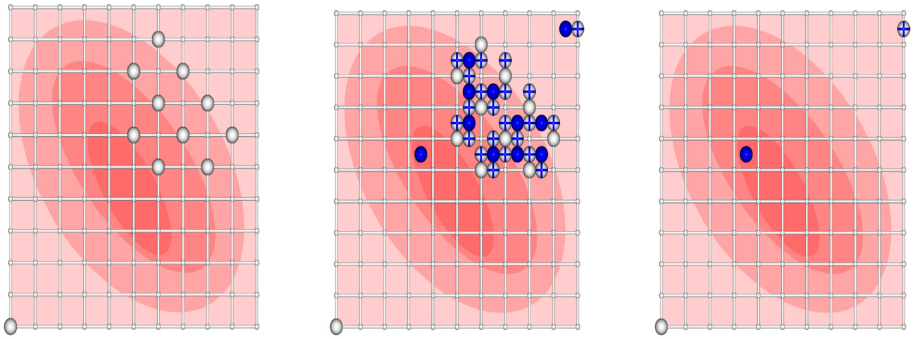
4 Improved Simulation of Simplicity

An interpolated function derived from real-life data is not necessarily Morse-Smale. One particular property necessary for algorithms computing the MS complex is differentiability, *i.e.*, vertices have pairwise distinct value, and hence no "flat" regions exist. Edelsbrunner and Mücke [4] introduced simulation of simplicity to resolve degenerate conditions where the input function is not a Morse function. Differentiability is simulated by assigning a strictly increasing ordering to vertices. Typically this is achieved with a pre-sort of the data, and vertex a has lower value than a vertex b if and only if it has lower index in the sorted array. In this manner, the order given by distinct data values is preserved, and an ordering for same-valued vertices is created. This sorting can be done implicitly, as shown in figure 2.

While this simple simulation of simplicity breaks ties and provides a complete ordering of the vertices of a data set, it introduces critical points that only exist due to index comparison. Figure 2 illustrates a scalar function with flat regions, where such an ordering produces spurious critical points at the boundaries of the flat regions. Although such critical points can be detected as noise and removed through simplification, they introduce a significant processing and data size overhead when computing the MS complex.

We improve the standard simulation of simplicity by introducing a new sorting order for the vertices that uses a greedy region-growing approach to eliminate the extra critical points due to flat regions. Our technique uses two priority queues to create a sorting order: the first, *simpleOrder*, returns the lowest unprocessed element ordered by function value and index in the data structure; the second, $bfsOrder$, has the same ordering, but only contains unprocessed vertices that are neighboring previously processed vertices. When a query asks for the lowest unprocessed vertex, the top elements of *simpleOrder* and $bfsOrder$ are compared, and the one with lower function value is returned. If they have the same function value, the top element of $bfsOrder$ is returned. When a vertex is processed, all of its unprocessed neighbors are added to $bfsOrder$.

In this manner, the ordering corresponds to the insertion time of a vertex in a flood-fill of the ascending manifolds of a function. It is a greedy technique that crosses flat



(a) (b) (c)

Fig. 2. Let the *index* of a vertex $v = (x, y)$ be $x + (y * xdim)$. Using standard simulation of simplicity, minima are identified at the boundaries of flat regions (a). When constructing an MS complex, these minima further generate the necessary separating saddles and maxima (b), far more than the actual three persistent critical points (c) (color plate C. 47, page 271).

regions in a breadth-first search, and therefore prevents extra minima from appearing. Note that it is not necessary to build the sorting order explicitly when using an algorithm based on region growing, since often these algorithms only use the ordering to extract the lowest unprocessed element. This same ordering is particularly effective for ordering cells in the computation of a discrete gradient field. For example, in the algorithm presented by Gyulassy et al. [6], cells of each dimension d are iterated, and local minima correspond to critical cells of index d. The performance of this algorithm, number of critical points found, and memory footprint are summarized in Table 5. Although this method was primarily designed to help overcome the flat regions in integer-valued data, a large improvement was also seen in the porous data set, a floating-point data set. This behavior indicates that discretizing a function makes it very sensitive to the ordering of cells, even when the original values are reasonably distinct floating-point numbers.

5 Efficient Cancellations

We use the description of the atomic cancellation operations provided in Gyulassy et al. [6] as a basis for examining the necessary major improvements to enable simplification of the MS complex in practice. We increase the performance of cancellations by introducing a data structure for constant-time DELETE operations, and also present several strategies for preventing a dramatic increase in the number of arcs due to the potential $kn \times km$ new arcs created in a saddle-saddle cancellation.

5.1 Efficient Data Structure

We use a double-linked list to store the set of arcs A_n connected to a node n. The INSERT operation can be done in constant time, since new arcs are always inserted at the beginning of the list. A DELETE operation in a linked list requires finding the element

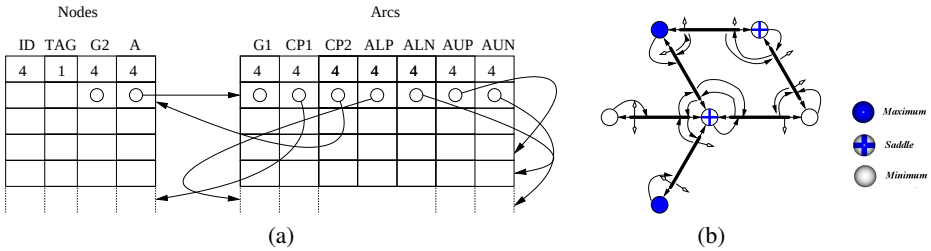


Fig. 3. The nodes and arcs of the complex are stored as fixed-size elements in an array (a) with the arcs containing the link information of the arc lists of their lower and upper nodes. ALN and ALP are the indices of the next and previous arcs in the double-linked arc list of the lower node, and AUN and AUP are the indices of the next and previous arcs in the list of the upper node. Each arc additionally has a pointer to its lower and upper nodes, and a pointer to the geometry of the arc. (b) shows a sample two-dimensional complex with the pointers at each node and arc.

Data set	Artificial	Hydrogen	Aneurism	Porous
(a)	620	951	638	3224
(b)	5130	5426	5006	5510

Table 2. A chart showing the sustained cancellation rate to cancel arcs 20% of maximum persistence for various data sets. (a) lists the number of cancellations per second using a standard linked list, and (b) shows the cancellations/second using a double-linked list. Note that the porous data set has a high rate of cancellations even when using a linked list, because it has very few high-valence nodes.

to remove, the previous element, and the next element, and reconnecting to remove the element. Finding an element is linear in the size of the list, while reconnecting is constant. A key observation is that in a cancellation operation, finding an arc in A_n is done in constant time, since the arc element is previously identified during an iteration of the arcs of its other endpoint. The previous and next elements must still be found, and in a double-linked list this is done in constant time. Therefore, the entire DELETE of an arc takes constant time in a cancellation operation. This behavior represents an improvement over the linear time for the linked list used in previous implementations [6].

Figure 3 illustrates the data structure used to store the nodes and arcs of the MS complex. Although this data structure is 1.4 times larger than a single-linked list, the majority of the memory footprint of a complex is taken up by the geometry components, such as storing the set of line segments that define an arc. In fact, for a typical complex, the additional space required by the arcs to store the double-linked list is only 0.1%. Table 2 compares the performance of using a standard linked list versus a double-linked list.

5.2 Cancellation Strategy

simplification of a function is achieved by repeated cancellation of critical point pairs. The ordering of critical point pairs is defined by *persistence*, which quantifies the importance of the topological feature associated with a pair. The *persistence* of a critical point pair is the absolute difference in value of f between the two points. Typically,

in this ordering, the arc with the lowest persistence is canceled first. However, such an ordering can lead to a memory footprint that grows past practical limits, due to the creation of $kn \times km$ new arcs. Instead, we design a strategy to determine when a cancellation can be delayed, based on conditions imposed on the neighborhood of the critical points to be canceled. We still maintain the property that all cancellations must be valid, *i.e.*, the pair of nodes is connected by exactly one arc, and the arc to be canceled has the lowest persistence of any in the neighborhood around the two nodes. Note that we do not follow a strict global persistence ordering, since there are small features that will not be removed until a later point in the simplification.

Limit $n \times m$ A straightforward technique to prevent the dramatic increase in memory is to delay a cancellation until the number of new arcs it would create, $n \times m$, is below some threshold t. Arcs are initially ordered by persistence. However, if canceling the arc with lowest persistence were to create more than t new arcs, the arc can be removed from the ordering and put in a delayed list, and the next arc in the ordering is selected for cancellation. cancellation of arcs in the neighborhood of delayed arcs can reduce the number of arcs incident at the nodes at either endpoint. Delayed arcs are canceled as soon as they create fewer than t new arcs. Table 3 shows the memory footprint and running times for several data sets as a function of t. While lower t values generally improve performance, they can lead to artifacts in the form of low persistence arcs that are perpetually delayed. When t is too high, the memory cost and run time increases, sometimes preventing the termination of the algorithm. However, t should be chosen as high as possible to keep the number of artifacts small.

Global Valence Control The *valence* of a node is the number of arcs that have it as an endpoint. Nodes with a high valence are expensive to cancel, and tend to generate many new arcs, leading to more high-valence nodes. We present a strategy to prevent the creation of high-valence nodes by delaying the cancellation of arcs that lead to the creation of a high-valence node. When an arc is canceled, the valence of nodes in a neighborhood around the canceled pair changes. Let n be the number of index-$(i+1)$ nodes in the neighborhood N_l of the lower node, m be the number of index-i nodes in the neighborhood N_u of the upper node. Let $a_i \in N_l$ be a node in the neighborhood of the lower node, and $b_i \in N_u$ be a node in the neighborhood of the upper node, and define $V(a)$ to return the valence of node a. We define the weight of an arc as $W(a) = max(max(V(a_i)+m|a_i \in N_l), max(V(b_i)+n|b_i \in N_u))$. Intuitively this value is the largest valence that canceling a would create. A cancellation is delayed if the weight of the arc is greater than a threshold t. Table 4 shows the memory footprint and running times for several datasets as a function of t. As with the new arc limiting method, lower t values correspond to higher performance and an increased number of artifacts. Computing $V(a_i)$ requires storage of the valence at each node, and an update of it whenever the neighborhood changes.

Data Set	t=20	t=40	t=80	t=160	t=360	t=720
Artificial-Time	–	6m 34s	12m 40s	11m 51s	25m 31s	–
Artificial-#Arc	–	13,054,772	14,392,886	14,398,002	25,652,629	–
Artificial-Atf.	–	887,281	145,332	24,045	10,244	–
Hydrogen-Time	0.86s	1.20s	2.03s	2.45s	2.63	3.30s
Hydrogen-#Arc	34,997	35,670	42,005	42,325	45,860	52,483
Hydrogen-Atf.	186	4	0	0	0	0
Aneurism-Time	–	1m 35s	4m 59s	8m 10s	16m 45s	47m 32s
Aneurism-#Arc	–	1,971,735	2,450,350	4,508,327	7,616,923	12,320,101
Aneurism-Atf.	–	23,320	1,625	122	34	6
Porous-Time	6m 25s	7m 04s	9m 32s	15m 45s	17m 16s	34m 47
Porous-#Arc	6,202,863	6,402,892	8,300,210	9,224,373	12,535,309	35,541,770
Porous-Atf.	352	202	54	17	6	5

Table 3. For each data set, we provide as a function of t: the time required to cancel up to 20% persistence, the maximum number of arcs in the complex, and the number of low-persistence artifacts that could not be cancelled, delaying arcs whose cancellation would generate more than $n \times m$ new arcs. In general, a higher threshold results in longer run times, a larger memory footprint, but fewer artifacts. Blanks indicate non-termination. The MS complexes were generated using the improved simulation of simplicity from Sect. 5.

Data Set	t=10	t=20	t=40	t=60	t=80
Artificial-Time	5m 20s	7m 35s	12m 40s	11m 51s	30m 52s
Artificial-#Arc	12,409,860	13,003,472	15,492,382	16,255,012	25,652,629
Artificial-Atf.	724,381	45,351	15,045	1,244	
Hydrogen-Time	1.06s	1.25s	2.18s	3.06s	3.54s
Hydrogen-#Arc	35,206	37,288	43,110	47,757	52,123
Hydrogen-Atf.	72	0	0	0	0
Aneurism-Time	1m 32s	2m 15s	10m 21s	17m 45s	25m 26s
Aneurism-#Arc	1,870,930	1,930,945	4,603,157	8,142,825	11,242,050
Aneurism-Atf.	26,076	18,450	102	3	2
Porous-Time	6m 05s	8m 36s	12m 29s	15m 54s	19m 24s
Porous-#Arc	5,730,043	7,404,100	10,004,540	13,224,373	15,853,612
Porous-Atf.	266	65	23	6	4

Table 4. For each data set, we provide as a function of t: the time required to cancel up to 20% persistence, the maximum number of arcs in the complex, and the number of low-persistence artifacts that could not be cancelled, delaying arcs whose cancellation would produce a node of valence greater than t.

Data set	Hydrogen	Aneurism	Porous
Standard Simp.	846,784	2,661,251	13,172,740
BFS Simplicity	3571	180,267	1,074,734

Table 5. The total number of critical points identified using a slice-by-slice computation of the discrete gradient. The top row lists the numbers of critical points found when the standard simulation of simplicity is used to order cells. The bottom row shows the numbers of critical points identified with the on-the-fly ordering.

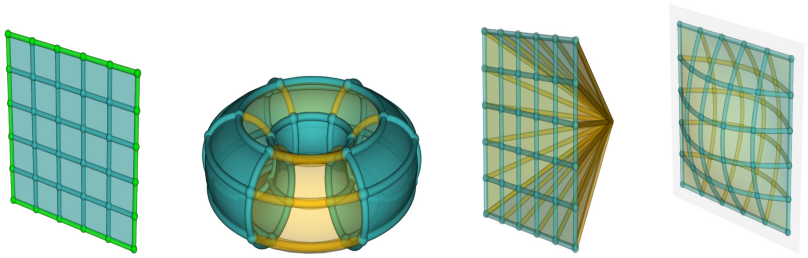
6 Boundary Conditions

The MS complex is well-defined for scalar functions on manifolds without boundary. Boundaries of a manifold present problems in computation of the MS complex, since they require special rules for the identification and simplification of critical points and cells. In practice, it is much easier to simulate the domain being a manifold without boundary, avoiding the special boundary cases. When the data is periodic, the underlying manifold has no boundary, and the MS complex is well-defined. When the data is not periodic, some other interpretation of the manifold must be used. Two standard techniques for removing the boundary of a manifold are one-point compactification and mirroring. Figure 4 illustrates how each boundary condition implies a different underlying space.

In one-point compactification, a vertex with infinite persistence is added for every boundary component, with every boundary cell attached to its corresponding vertex. While this can be simulated in practice, it can lead to artifacts, such as minima on the boundary disappearing and 1-saddles on the interior have arcs that connect outside the data. Additionally, the simulated vertices will necessarily have high valence, leading to slower execution times, and often require special data structures.

In mirrored boundary conditions, a copy of the underlying manifold is attached to itself along its boundary cells. This technique has the advantage of not introducing high-valence nodes, and small features near the boundary are preserved. However, it is often impractical to keep a redundant copy of the "mirrored" complex due to memory limitations. One solution is to modify the data structure of cells crossing the boundary, since differences between the MS complex on the original manifold and the complex on the mirrored manifold will be restricted to a neighborhood around the boundary. The biggest problem with this technique is that cancellation operations (in particular saddle-saddle cancellations) are not necessarily symmetric, therefore, canceling a node on the boundary with a node in the interior can lead to inconsistencies between the complex of the original manifold and the mirrored complex, and the technique loses its intuitive appeal.

A simple solution is possible in practice. We avoid picking a particular technique to remove the boundary by enforcing the condition that gradient flow not cross the boundary. This is done by computing the discrete gradient first on the boundary, then the interior of the domain. Additionally, when canceling critical points, boundary nodes are only canceled with other boundary nodes, and interior nodes with interior nodes. Therefore, the flow touching the boundary from the interior will always end at a critical point on the boundary, and flow touching the boundary from "outside" will end at a boundary critical point, effectively isolating the interior of the data. One major advantage of this technique is that it requires no special conditions for finding critical points and cells of the MS complex. The disadvantage is that small features on the boundary are often preserved, since they cannot be canceled with interior critical points. In practice, this is an effective technique that works for most cases. Any boundary artifacts can usually be removed from consideration by careful filtering of the MS complex.



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

Fig. 4. A 2-manifold with interior (blue) and boundary (green) cells (a). Periodic boundary conditions (b) attach the boundary on opposing sides. One-point compactification (c) attaches the boundary to a point with infinite persistence. Mirrored boundary (d) duplicates the interior and attaches it along the boundary (color palte C. 48, page 271).

7 Conclusions

We present techniques that make computing and simplifying MS complexes possible in practice. Using the improved simulation of simplicity results in a more efficient implementation, especially in data sets with large flat regions, since it reduces the initial number of critical points found. In the case of the hydrogen dataset, the improved simulation of simplicity generated 200 times fewer critical cells than the standard simulation of simplicity. While particularly designed for computing MS complexes using a discrete approach, this simulation of simplicity can be applied to any topology-based technique to reduce the initial number of low-persistence critical points found. The techniques presented for reordering cancellations make it possible to simplify an MS complex without entering a situation where unregulated cancellations lead to memory usage that exceeds practical capabilities. The double-linked list we use to store the connectivity of the 1-skeleton of the MS complex further provides a decrease in run-time complexity of cancellation operations. Finally, we present a simple solution to enable a simple implementation when boundaries are present.

The techniques we have presented make possible computation and simplification of the MS complex. For example, without the improved simulation of simplicity, the MS complex of the hydrogen data set has almost one million low-persistence critical points. simplification of this large complex does not terminate without using a re-ordering strategy for cancellations. The hydrogen data set is a small data set with fewer than 100 persistent critical points, and the MS complex can be computed and simplified in under a minute when using the techniques discussed in this paper. These techniques are simple solutions to practical problems in designing efficient implementations of algorithms to compute and simplify the MS complex.

8 Acknowledgments

Attila Gyulassy was supported by the Lawrence Scholar Program (LSP). In addition, this research was supported in part by the National Science Foundation, under grant

CCF-0702817. We would like to thank the members of the Center for Applied Scientific Computing (CASC), at LLNL, and the members of the Visualization and Computer Graphics Research Group of the Institute for Data Analysis and Visualization (IDAV), at UC Davis. This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

1. P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):385–396, 2004.
2. H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proc. 19th Ann. Sympos. Comput. Geom.*, pages 361–370, 2003.
3. H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry*, 30(1):87–107, 2003.
4. H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990.
5. R. Forman. A user's guide to discrete morse theory, 2001.
6. A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci. A practical approach to morse-smale complex computation: Scalability and generality. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1619–1626, 2008.
7. A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. Topology-based simplification for feature extraction from 3d scalar fields. In *Proc. IEEE Conf. Visualization*, pages 535–542, 2005.
8. A. Gyulassy, V. Natarajan, V. Pascucci, P. T. Bremer, and B. Hamann. A topological approach to simplification of three-dimensional scalar fields. *IEEE Transactions on Visualization and Computer Graphics (special issue IEEE Visualization 2005)*, pages 474–484, 2006.
9. A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of morse-smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1440–1447, 2007.
10. H. King, K. Knudson, and N. Mramor. Generating discrete morse functions from point data. *Experimental Mathematics*, 14(4):435–444, 2005.
11. T. Lewiner, H. Lopes, and G. Tavares. Applications of forman's discrete morse theory to topology visualization and mesh compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):499–508, 2004.
12. S. Smale. Generalized Poincaré's conjecture in dimensions greater than four. *Ann. of Math.*, 74:391–406, 1961.
13. S. Smale. On gradient dynamical systems. *Ann. of Math.*, 74:199–206, 1961.

Topological Feature Extraction for Comparison of Terascale Combustion Simulation Data

Ajith Mascarenhas[1], Ray W. Grout[1], Peer-Timo Bremer[2], Evatt R. Hawkes[4], Valerio Pascucci[3], Jacqueline H. Chen[1]

[1] Sandia National Laboratories, Livermore, CA, USA
[2] Lawrence Livermore National Laboratories, Livermore, CA , USA
[3] University of Utah, Salt Lake City, UT, USA
[4] University of New South Wales, Sidney, Australia

Abstract. We describe a combinatorial streaming algorithm to extract features which identify regions of local intense rates of mixing in two terascale turbulent combustion simulations. Our algorithm allows simulation data comprised of scalar fields represented on 728x896x512 or 2025x1600x400 grids to be processed on a single relatively lightweight machine. The turbulence-induced mixing governs the rate of reaction and hence is of principal interest in these combustion simulations. We use our feature extraction algorithm to compare two very different simulations and find that in both the thickness of the extracted features grows with decreasing turbulence intensity. Simultaneous consideration of results of applying the algorithm to the HO_2 mass fraction field indicates that autoignition kernels near the base of a lifted flame tend not to overlap with the high mixing rate regions.

1 Introduction

The objective of this work is twofold: first, to develop a method to characterize the mixing length scales in a turbulent flow simulation on an instantaneous and local basis. Second, to explore the interaction between mixing and autoignition. To accomplish this we extend a segmentation method previously used to identify instabilities in a mixing layer to identify regions where the local rate of mixing is large and to collect statistics on the size of the resulting features. We also apply the segmentation algorithm to a scalar representative of autoignition and use the resulting overlap to investigate the relationship between the mixing and autoignition process.

Generalizing scientific insight obtained from detailed turbulent combustion simulations requires appropriate quantification of the parameters of the simulations. Typically, dimensionless quantities based on the expected value — in a statistical sense — of relevant length scales are employed to characterize the simulation. These quantities provide a characterization of local effects based on global expected values. For combustion processes, which are highly local phenomena, it is advantageous to have a local identification and measure of the length-scale.

In our approach, we extract features defined as isosurfaces of the scalar dissipation rate field χ surrounding a local maxima, where the isovalue is a percentage of the magnitude of the local maxima. The scalar dissipation rate indicates the rate of mixing and will be defined more precisely in Section 3. The thickness of these regions is a direct measure of the local mixing length-scale: we will conduct the analysis for two combustion simulation datasets.

The remainder of this paper is organized as follows: first, we will describe the feature detection and extraction pipeline in Section 2.2 after a short review of prior work in this area. Next, we will briefly review the combustion physics which motivate this effort in Section 3. Finally, in Section 5, we will apply the method to two combustion simulation datasets and generate thickness distributions to validate the expected behaviour before exploring the mixing / autoignition relationship in the second dataset. Specifically, our results are:

1. A topology based segmentation of the high χ regions.
2. A new hierarchical merge tree structure encoding segmentation for a wide range of thresholds. In a single pass over the data we extract an adjustable pre-segmentation which can be quickly adapted to any particular threshold in a post-processing step.
3. Statistics on the morphology of the high χ regions
4. A local measure of the mixing length-scale necessary to characterize the combustion regime
5. A demonstration that the autoignition process is most likely to proceed in the absence of locally high mixing rates

2 Segmentation Algorithm

Our analysis pipeline has three stages. In the first stage, we connect the regularly gridded input data into a mesh and output a stream of edges sorted on function value at their lower vertex. In the second stage, we use the streaming edges as input and compute a segmentation and merge-tree of the data. Both these stages are described below. In the third stage, we compute the morphological statistics for analyzing the data, as shown in Section 5.

2.1 Prior work.

We provide a brief review of related work in feature detection, and shape analysis with an emphasis on topology based methods.

Recently, topology based segmentation and feature detection has become increasingly popular. One of the most widely used topological structure is the *contour tree* [4, 17] which encodes the topology of all isosurfaces in a particular data set. The contour tree is used, for example, to enhance isosurface rendering [7] and the automated design of transfer functions [6, 19]. Carr et al. [2] use the contour tree simplified by various geometric measures to identify isosurfaces of interest. The methods presented in this paper are similar in that we use the *merge tree* a sub-structure of the contour tree to represent regions of locally enhanced mixing rates. However, as discussed in Section 2.2 we use a novel metric called *relevance* to simplify the merge tree and thus define features.

A closely related family of methods use the Morse-Smale complex or sub-sets thereof to represent features. The Morse-Smale complex was first introduced for surfaces by Edelsbrunner et al. [4] and extensions to the three-dimensional case can be found in [5, 10, 11].

Laney et al. [10] use the stable manifolds of the two dimensional Morse-Smale complex to analyze the mixing layer in a Rayleigh-Taylor instability. Gyulassy et al. [9] use the ascending one-manifolds of the three-dimensional complex to define and analyze core lines in atomistic simulations of porous media.

2.2 Current Method

Feature Segmentation Our feature definition is based on the *merge tree* of a scalar function. Given a smooth function $f : \mathbb{R}^n \to \mathbb{R}$ the *level set* of f at isovalue s is defined as all points in $\mathbb{R}^n$ with function value s. A connected component of a level set is called a *contour*. The merge tree of f records the merging behavior of the contours of f as the isovalue is swept top-to-bottom through the function range; see Figure 1. Each time the isovalue passes a maximum a new contour is born and a new leaf appears in the merge tree. Each time two contours meet they merge into a single component represented in the merge tree as a joining of two branches.

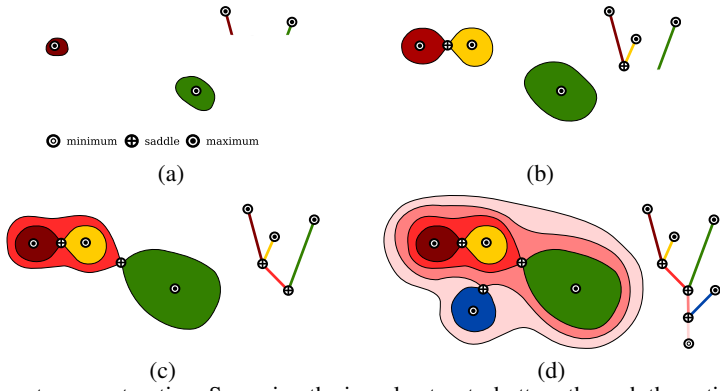


Fig. 1. Merge tree construction: Sweeping the isovalue top-to-bottom through the entire function range one keeps track of those portions of space that lie above the isovalue. (a)-(d) Show several snapshots with the colored regions indicating regions above the current isovalue. The leaves of the tree are the local maxima at which contours are born, the interior nodes saddles at which contours merge, and the root is the global minimum.

Each branch of the merge tree can be used to represent the subset of $\mathbb{R}^n$ defined by the union of all its contours, indicated by the colored regions in Figure 1. In this paper we are interested in regions around high χ values and we define these regions starting from the leaf branches of the merge tree.

However, not every leaf branch should be considered. For example, regions with low absolute χ value should not be included. Furthermore, two neighboring regions with similar χ values should count as a single region. We represent these adaptations by cutting and simplifying the merge tree. Given a small absolute χ value under which no region should be considered we cut the tree at this value by disregarding all merges below the threshold and deleting all branches completely below the threshold; see Figure 2(a). The cutting operation transforms the tree into a forest in which only the roots have a χ value below or equal to the threshold.

After disregarding low χ regions the resulting trees are simplified by removing branches. The traditional approach in topological analysis is to pair nodes for cancellation and compute their *persistence*, the difference in their function values, as importance measure and remove pairs of nodes in order of increasing persistence. The problem with this approach is that persistence ranks pairs of nodes relative to the global function range. Only branches with a large absolute span in function value will survive. However, for our analysis the relative change in χ with respect to the local maximum

is more important than its absolute value. Therefore, we define a new measure called *relevance* to rank branches. The relevance of a branch b with maximum u and saddle v is defined as $rel(b) = (f(u) - f(v))/(f(u) - f_{min})$ where f_{min} is the global minimum of f. By scaling the local function value difference by the local maximum the relevance measures the differences in function value with respect to the local neighborhood rather than the global function range. Similar to the persistence based algorithm we then remove branches in order of increasing relevance, see Figure 2(b). Whenever we remove a branch we merge its corresponding region with its neighboring branch. Thus we combine neighboring regions if their merge happens at a function value close to one of their maxima.

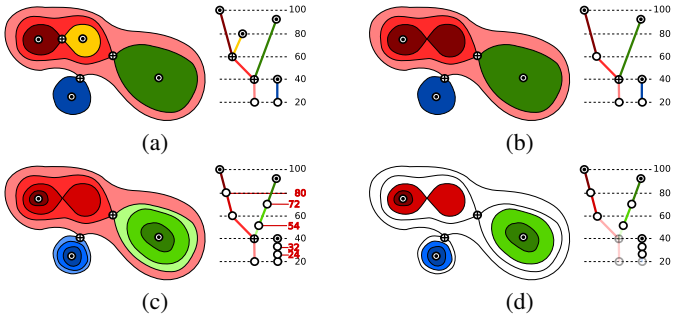


Fig. 2. Merge tree adaptation: (a) The tree of Figure 1(d) cut at an absolute threshold of 20. The absolute function values are on the right of each figure and we assume a global minimum at 0. The lowest saddle is ignored and split into two local nodes creating a forest of two independent merge trees; (b) The tree of (a) simplified using a relevance of 0.25. The yellow branch and its corresponding region are merged with its neighbor; (c) The simplified tree of (b) with split points at multiples of relevance 0.2 (function values in red); (d) The tree of (c) adapted to a relevance of 0.4. All non-selected segments are transparent.

Once the tree is cut and the resulting forest simplified each leaf branch represents a region. However, we are not only interested in counting the regions but also in their geometric properties. Unfortunately, the regions corresponding to a branch do not necessarily represent its geometry well. Merge tree branches can span a very large range in function value and their length can vary widely depending on local configurations. Therefore, one branch might have a relevance of 0.1 thus representing contours from its local maximum down to 90% of its value while another has a relevance of 0.5 representing a much larger volume in space. To increase the flexibility of the segmentation we split all branches at fixed relevance values creating *segments* of bounded length, see Figure 2(c).

Finally, we create a particular segmentation by choosing a relevance threshold and only consider segments at this relevance value or below, see Figure 2(d). Notice, that this final parameter choice represents a simple selection of a root branch and its sub-tree all of whose branches have a lower relevance value. Denote this root as the *leader* of the segment. Selecting segments can be performed in a post-processing step. Therefore, the split merge forest represents a highly flexible pre-segmentation that can be stored compactly and is easily adaptable.

Streaming Construction. We take as input a regular grid tetrahedralized as follows: Each unit cube is decomposed into tetrahedra along the main diagonal and the scalar function sampled at the vertices of the grid is extended throughout the domain by piecewise-linear interpolation. We output the merge tree of the function and a labeling of each vertex in the grid to a branch in the merge tree. For a vertex p denote its branch id as $S(p)$.

To construct the initial merge tree we use a variant of the standard algorithm described in [4]. The aim is to maintain a set of contours as the isovalue is swept top-to-bottom through the function range recording the birth, merge, and the final death event. However, instead of finding all local maxima and using priority queues to represent the contours we pre-sort the data. We sort all edges in our mesh based on the function value of their lower vertex and stream them to the merge tree module in order of decreasing function value. The advantage of this strategy is that the merge tree computation becomes very simple and that we can easily attach finalization information [13] to the vertices.

Each input edge is handled in one of three cases: A birth of a new contour, a merging of contours, or a expansion of a contour. For a birth, we create a new branch and assign its unique id to the vertices of the edge. For a merge, we also create a new branch and assign it as parent of the two branches representing the two merging contours. For an expansion, the edge connects a new vertex to an existing contour/branch. We find the lowest current ancestor of this branch and assign its id to the new vertex. We maintain ancestor information in a standard union-find structure using shortcuts to improve performance. By creating new contours/branches at merge events we label vertices by branch rather than by highest local maximum as is done in the standard algorithm. As discussed above this correspondence is a crucial part of our segmentation strategy. Once all edges have been processed all branches without a lower node are "capped" with a minimum. At any time we only hold the un-finalized vertices in memory which results in a very low memory foot print. As a vertex gets finalized we store its index on disk to be used later.

The cutting of the tree is performed implicitly by excluding all edges completely below the given threshold. This leaves all branches that would normally span the threshold value to be capped with a minimum resulting in the desired cut. We then perform the simplification and split of the merge tree recording the corresponding mapping. Finally, we re-process the segmentation file: For each vertex we read its branch id during the initial construction and determine whether it was affected by either a simplification and/or a split and change the id if necessary. Finally, we store the simplified merge tree on disk. Once the computation is completed we have labeled each vertex p above the cut-off with its corresponding branch id $S(p)$ in the simplified and split merge tree. Furthermore, we can use the stored merge tree to further adapt the segmentation to explore different parameter choices.

3 Combustion Dataset

In turbulent non-premixed combustion, where the fuel and oxidizer reactant streams are segregated, the reactant streams must be molecularly mixed before reaction can occur. Therefore, the turbulent mixing rate is a key quantity in determining the overall burning

rate and efficiency. In general, as the mixing rate increases, reaction rates increase and the overall efficiency increases. A non-premixed flame has a well defined internal structure (see Figure 3). Beyond a critical rate of mixing reactions cannot keep up with the mixing and the flame quenches locally. This undesirable situation can lead to increased emissions. If quenching is pervasive, then global blow-out can occur, which would be catastrophic, for example, in an aero gas-turbine engine. In an autoignition situation, there is a similarly well structured relationship between the species concentrations but the relationship is in time instead of space as radical concentrations build up to sufficient levels to establish a flame.

Turbulent mixing is characterized locally by the scalar dissipation rate, χ which is equal to twice the product of the molecular diffusivity and the square of the mixture fraction gradients:

$$\chi \equiv 2D \frac{\partial \xi}{\partial x_i} \frac{\partial \xi}{\partial x_i}, \tag{1}$$

and is the rate at which scalar fluctuations decay due to diffusive processes. The mixture fraction ξ is the mass fraction of atoms from the fuel stream in the reactant mixture and varies between 0 in the oxidizer stream and unity in the fuel stream. The optimal condition for combustion typically occurs at stoichiometric conditions where the fuel and oxidizer are consumed in their entirety. The scalar dissipation rate is required in virtually every turbulent combustion model [15, 17, 18] and its dependencies upon the turbulent strain rate and combustion heat release rate are highly sought after. In the

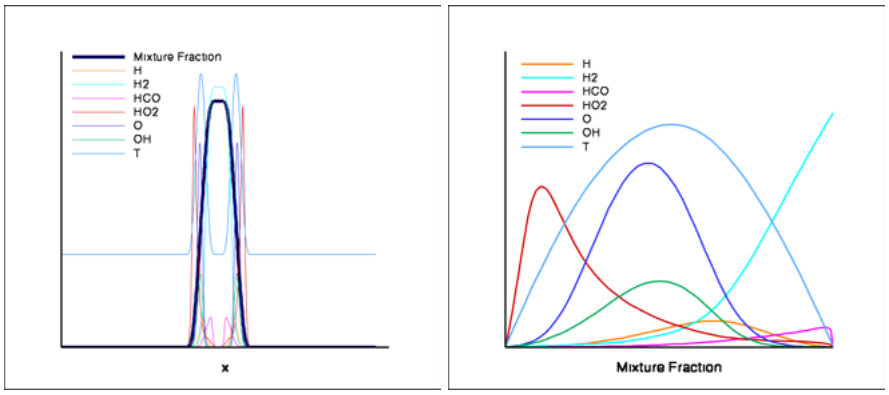


Fig. 3. Laminar flame profile illustrating highly structured ordering of key quantities in physical space at left, and mixture fraction coordinates at right.

absence of flow, the solution to the mixing problem can be obtained analytically. The presence of turbulent flow results in a highly complex and chaotic influence on the mixing process, which ultimately influences the combustion process. Hence, when we discuss the influence of turbulence on combustion, we are largely referring to the influence of turbulence on mixing. The presence of combustion and associated heat release results in a complicated feedback mechanism where the combustion alters the turbulent flow, further complicating the situation.

It is known from experiments (e.g., [14]) and previous analysis of DNS results [8, 20] that regions of locally intense mixing resulting from turbulent strain in a non-reactive jet or shear layer lead to ramp-cliff structures or intense mixing rates inclined at $\sim$45 degrees to the jet axis, i.e. along the directions of principal strain rates. These structures are characterized by relatively large dimensions in the tangent plane of the principal strain rates, and a much smaller dimension in the direction normal to the principal strain rate. It is less well understood what the orientation and morphology of χ is in a reactive flow undergoing dilatation effects from heat release rate. Previous statistical models of χ have relied on analogous integral quantities such as the turbulent kinetic energy and mechanical dissipation rate, or assumed scalar variance and dissipation rate. However, 3D measurements have shown that the thickness of χ scales not with the integral scale of turbulence, but rather, with the small-scale turbulence, i.e. the Kolmogorov or Batchelor scales. The advent of terascale 3D direct numerical simulations (DNS) of moderate Reynolds number turbulent reactive jet flames has enabled the direct computation of χ and its evolution [12]. For the purpose of characterizing the regime of the simulation, we no longer need to estimate local mixing dimensions from the large scale expected values of the turbulence field, as we have the scalar dissipation rate field itself available. The method presented here is devised to use the resolved χ field to identify and measure the structures described above.

The first simulation data we treat is a temporally-evolving turbulent CO/H_2 jet flame undergoing extinction and reignition at different Reynolds numbers [12]. The simulations were performed with up to 0.5 billion grid points. The configuration is shown on the left of Figure 4. Periodic boundary conditions in the mean flow (x) direction results it a situation where the mixing rates increase until approximately midway through the simulation, after which point they begin to decay.

The second simulation is a spatially-evolving lifted turbulent Ethylene/air jet flame. This simulation was performed on a much larger grid, up to 1 billion grid points, and is depicted on the right of Figure 4. In this arrangement, the configuration is statistically stationary in time. Spatially, the rates of mixing decay downstream from the inlet. The

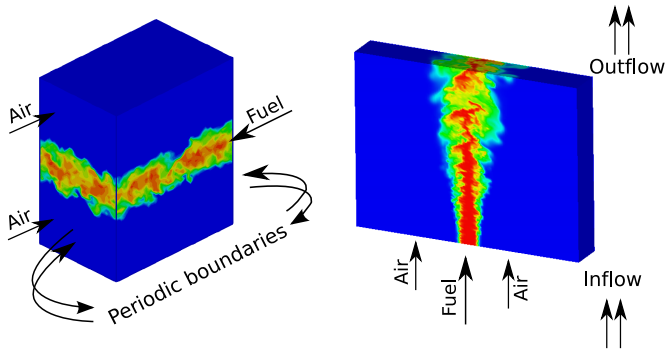


Fig. 4. Schematic of combustion scenarios. Left, the temporally evolving jet. Right, the lifted jet.

task we set for ourselves is to answer the following question: can we measure, on a local basis, the mixing lengthscale in such a way that we can obtain a distribution of the mixing length within a particular volume.

4 Application and Results

4.1 Application to Dataset A: temporal jet flame

In the application to the first dataset, we apply the method to several time steps, with relevance $R = 0.85$, to validate that the resulting features behave as we would expect from the physics of the situation. Two typical features are shown in Figure 5.

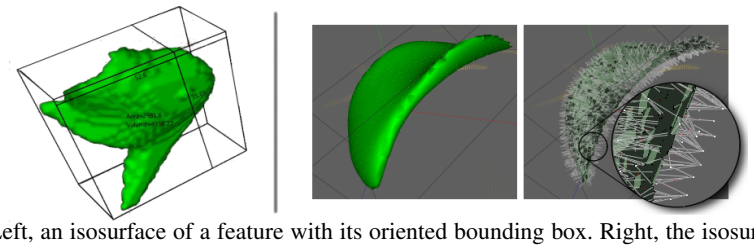


Fig. 5. Left, an isosurface of a feature with its oriented bounding box. Right, the isosurface of a feature and its medial axis. The inset zooms a portion showing the medial axis vertices and lines connecting them to the closest point on the isosurface (color plate C. 49, page 272).

The high scalar dissipation rate features are predominantly pancake shaped but are embedded in $\mathbb{R}^3$ with undulations as shown in Figure 5. Using the shortest side of an oriented bounding box (such as obtained from principal component analysis) for thickness is an over-estimate. Instead, we first compute the medial-axis of the isosurface that represents the feature. For each vertex on the medial axis we compute the distance c to the closest point on the isosurface; the thickness at that point of the medial axis is defined as $2c$. This method gives us a distribution of thickness values for each feature.

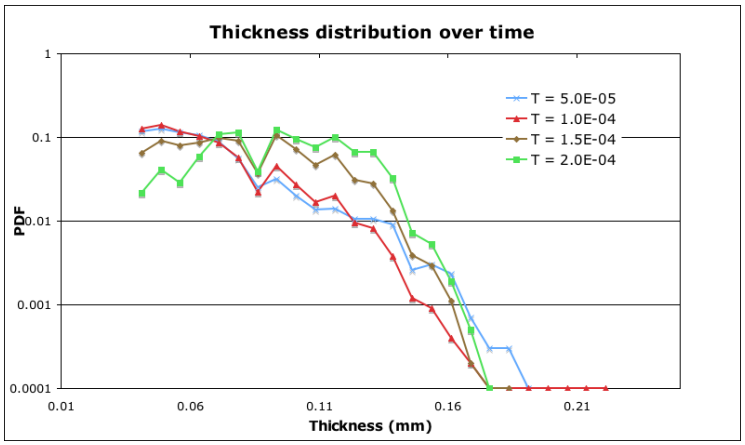


Fig. 6. Distribution of segment thickness for Dataset A (temporal jet).

Figure 6 shows the thickness distribution of segments within four time-steps. This is our primary interest, as this thickness indicates the physical lengthscale of the mixing region. The distribution functions show that thickness tends to decrease from $t = 0.05$ms to $t = 0.1$ms and then increases through $t = 0.2$ms. This is the expected behavior; as mentioned earlier, the peak mixing rate is midpoint in the simulation, near $t = 0.1$ms, after which the mixing rate decays.

4.2 Performance.

The data is available for dataset A is a sequence of 241 time-steps; each time-step is a regularly gridded sample of $768 \times 896 \times 512$ float values. We run all our computations on a 64bit Linux cluster with 3.4GHz Intel processors and 6GB memory per node. All our computations can be submitted to a batch system. We present memory usage and running times for each stage of the computation pipeline in Table 1. All numbers are averages for the processing of a single time-step. Overall the batch system schedules several jobs together and we are able to process all time-steps in the order of a few hours. Although we use a conservative cut-off for ignoring low values of χ while computing the segmentation we are able to compute it in a few minutes. We could choose to be more conservative by including more edges at the cost slightly more running time. The most time consuming stage is the statistics computation. This is expected as this is also the most compute intensive stage; we extract an isosurface for each segment and compute its medial axis using tcocone [3]. However once this stage is complete we store all required statistics in a file per time-step so that we can extract various plots from this data quickly.

	time (hh:mm:ss)	Memory
Streaming edges	00:29:28	4.4GB
Segmentation	00:03:18	361.0MB
Statistics	01:03:36	370.0MB

Table 1. Performance statistics for one time-step at each stage of the pipeline.

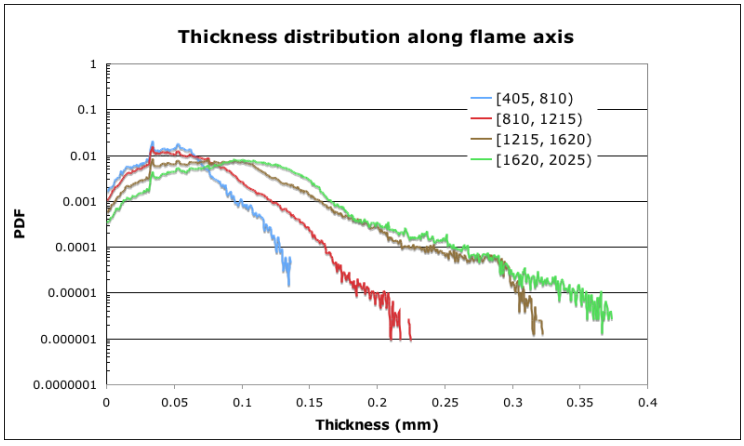


Fig. 7. Distributions of segment thickness for Dataset B (lifted jet).

4.3 Application to Dataset B: lifted flame

The second dataset is statistically stationary in time, so we perform our analysis for only one time-step. However, it is spatially evolving, so we collect statistics for several positions downstream of the inlet. The scalar dissipation rate decays (with regard to the

peak values and mean) with the distance from the inlet. In Figure 7, thickness statistics analogous to those shown above for the temporally evolving jet are shown.

In comparison to the jet dataset, the χ segments in the lifted dataset are much smaller and thinner. This is consistent with the physical differences between the two simulations; the lifted data is much more highly strained near the jet inlet, so one would expect that the mixing lengths would be smaller (corresponding to higher rates).

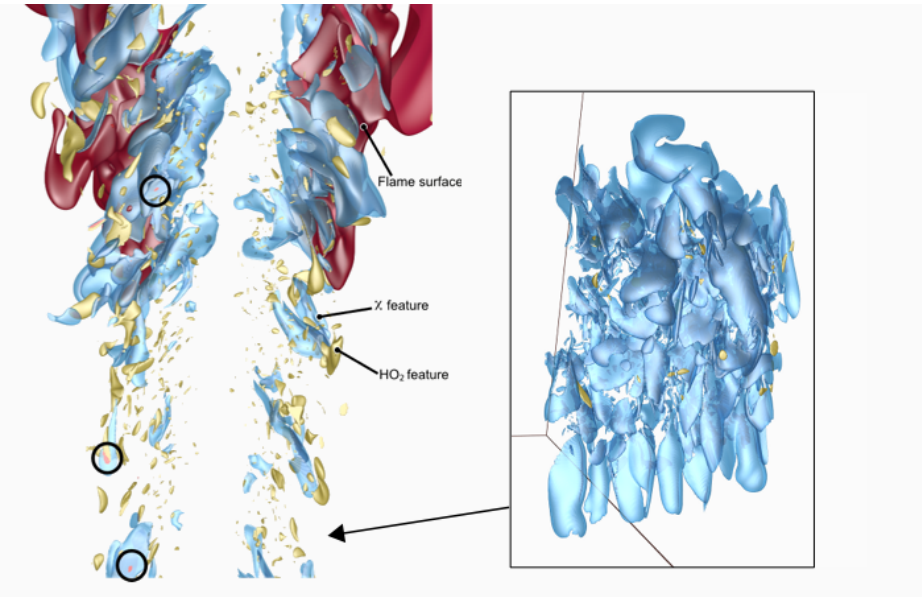


Fig. 8. Relationship between χ and HO_2 fields. Left, all HO_2 kernels below the base of the flame with χ segments that overlap them. Right, a close-up side view showing all χ segments and HO_2 kernels (color plate C. 50, page 272).

In the region near the inlet, autoignition kernels, which can be identified by a local maxima in the HO_2 concentration, develop and eventually transition into a fully burning flame. Figure 8 (left) shows features obtained by segmenting the HO_2 field along with the overlapping χ features. As the χ features identify regions where the rates of mixing could dissipate the radical build up and destroy the autoignition kernel, it is expected that the kernels will live outside the high χ regions. This is indeed the case; Figure 8 (left) shows only those χ features that overlap with the autoignition features. Figure 8 (right) shows a close-up view with all of the χ features, which clearly occupy a large percentage of the space, yet overlap minimally with the autoignition features.

As the kernels grow and transition to a fully burning flame, they are less adversely affected by high mixing rates. Table 2 gives an account of the variation of the overlap with distance from the inlet: the fraction of points within an autoignition feature overlapping with a dissipation rate feature is small, and smallest near the inlet.

5 Conclusions

We have presented a method that is capable of determining a highly localized measure of the mixing length. Application to two terascale combustion data sets indicates that

Axial position	**# overlap points**	**% of HO_2 overlap points**
[0, 250]	12	0.39
[250, 500]	204	1.00
[500, 750]	2,182	2.40
[750, 1000]	4,699	5.01

Table 2. Count of overlap between HO_2 and χ features

the method can be applied to very large datasets on relatively modest hardware. This is a particularly useful feature of the method, as the hardware to perform the original simulation is out of reach of a large fraction of the combustion community. Future work will focus on comparison of the mean mixing length determined by this method to the traditional scaling laws when averaged over a large spatial / temporal region. Although we extracted statistics for absolute measurements in this work to avoid ambiguity, we could have as easily normalized the thickness by the chemical lengthscale characterizing the reaction. In his form, the method can provide a direct measure of the ratio between the mixing length resulting from the turbulent flow and the reaction lengthscale which can be used to characterize the combustion simulation. The same local feature which is useful to measure the mixing length is also a volumetric entity; comparing the results of segmenting multiple fields provides information about their spatial proximity. In this case, segmentation of χ features considered together with segmentation of HO_2 features indicates that autoignition kernels are preferentially found in regions of low mixing.

References

1. H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Comput. Geom. Theory Appl.*, 24(3):75–94, 2003.
2. H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *IEEE Vis. '04*, 497–504. IEEE Comp. Society, 2004.
3. T. K. Dey and S. Goswami. Tight Cocone: A water-tight surface reconstructor. *J. of Comp. Infor. Sci. Eng.*, 3:302–307, 2003.
4. H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete Comput. Geom.*, 30:87–107, 2003.
5. H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proc. 19th Sympos. Comput. Geom.*, 361–370, 2003.
6. I. Fujishiro, T. Azuma, and Y. Takeshima. Automating transfer function design for comprehensible volume rendering based on 3D field topology analysis. In *Proc. IEEE Vis. '99*, 467–470, Oct. 25–29, 1999.
7. I. Fujishiro, Y. Takeshima, T. Azuma, and S. Takahashi. Volume data mining using 3d field topology analysis. *IEEE Comp. Graphics and Applications*, 20:46–51, 200.
8. R. W. Grout, E. R. Hawkes, J. H. Chen, A. Mascarenhas, P.-T. Bremer and V. Pascucci. 32^{nd} *Int. Symp. on Combustion*, WiPP05-23, 2008.
9. A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E.Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Trans. on Comp. Graphics and Vis. (TVCG)*, 13(6):1432–1439, 2007.
10. A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann. Topology-based simplification for feature extraction from 3D scalar fields. *IEEE Trans. on Comp. Graphics and Vis. (TVCG)*, 12(4):474–484, 2006.

11. A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of Morse-Smale complexes for three-dimensional scalar functions. *IEEE Trans. on Comp. Graphics and Vis. (TVCG)*, 13(6):1440–1447, 2007.
12. E. R. Hawkes, R. Sankaran, J. C. Sutherland, and J. H. Chen. *Proc. Comb. Inst.*, 31:1633–1640, 2007.
13. M. Isenburg and P. Lindstrom. Streaming meshes. In *Proceedings of the IEEE Vis. 2005 (VIS'05)*, 231–238. IEEE Comp. Society, 2005.
14. S. A. Kaiser and J. H. Frank. *Proc. Combust. Inst.*, 31:1515–1523, 2007.
15. A. Klimenko and R. W. Bilger. *Prog. Energy Combust. Sci.*, 25(6):595–687, 1999.
16. D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Trans. Vis. and Comp. Graphics (TVCG) / Proc.of IEEE Vis.*, 12(5):1052–1060, 2006.
17. N. Peters. *Prog. Energy Combust. Sci.*, 10:319–339, 1984.
18. S. B. Pope. *Prog. Energy Combust. Sci.*, 11(2):119–192, 1985.
19. G. Weber, G. Scheuermann, H. Hagen, and B. Hamann. Exploring scalar fields using critical isovalues. In M. Gross, K. I. Joy, and R. J. Moorhead, editors, *Proc. IEEE Vis. '02*, 171–178, IEEE Comp. Society Press, 2002.
20. P. Vaishnavi, A. Kronenburg, and C. Pantano. *J. Fluid Mech.*, 596:103–132, 2008.
21. M. J. van Kreveld, R. van Oostrum, C. L. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Symp. on Computational Geometry*, 212–220, 1997.

Topological Extraction and Tracking of Defects in Crystal Structures

Sebastian Grottel[1], Carlos A. Dietrich[2], João L. D. Comba[2], and Thomas Ertl[1]

[1] VISUS - Universität Stuttgart, Germany
[2] Instituto de Informática - UFRGS, Brazil

Abstract. Interfaces between materials with different mechanical properties play an important role in technical applications. Nowadays molecular dynamics simulations are used to observe the behavior of such compound materials at the atomic level. Due to different atom crystal sizes, dislocations in the atom crystal structure occur once external forces are applied, and it has been observed that studying the change of these dislocations can provide further understanding of macroscopic attributes like elasticity and plasticity. Standard visualization techniques such as the rendering of individual atoms work for 2D data or sectional views; however, visualizing dislocations in 3D using such methods usually fail due to occlusion and clutter. In this work we propose to extract and visualize the structure of dislocations, which summarizes the commonly employed filtered atomistic renderings into a concise representation. The benefits of our approach are clearer images while retaining relevant data and easier visual tracking of topological changes over time.

1 Introduction

Compound materials are used in several applications, and it is of uttermost importance to understand their properties and how they react under external forces. An important aspect to consider is the appearance of defects in their atomic structure [4], which have a direct relation to material properties such as resistance, strength, etc. For this purpose, Molecular Dynamics (MD) simulations are used, which reproduce the behavior of atoms and molecules for a period of time. MD generates a great number of components which require complex analysis, and often only statistical measurements are evaluated.

However, the analysis of defects can be restricted to a subset of this data. Defects are local changes in symmetry or regularity of the crystal lattice. Because the atomic structure of compound materials in normal conditions corresponds to crystal lattice structures, it suffices to consider those atoms whose neighborhoods deviate from the regularity established by this lattice. Common lattices include the body-centered cubic (BCC), face-centered cubic (FCC) or hexagonal-closed packed (HCP). The atoms of irregular lattice structures form defects called *dislocations* (a 1-D defect) and *stacking faults* (a 2-D defect). There are special properties from the application domain on which we can rely. Dislocations always form cycles or networks, and never have open, unconnected endpoints. Stacking faults are always bordered by dislocations, which makes it easy to define where they end or start. For FCC lattices, atoms which participate in

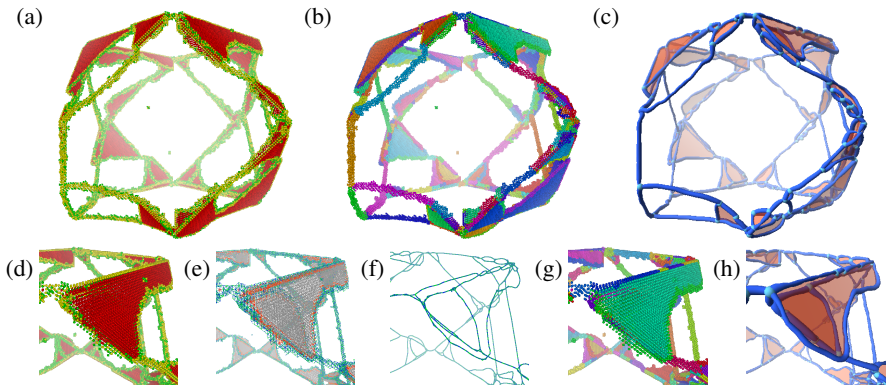


Fig. 1. Upper row: (a) original atom data, (b) after segmentation, (c) extracted defects. Lower row: Detail views - (d) original atom data, (e) relative neighborhood graph, (f) graph after simplification, (g) segmented atom data, (h) extracted defects (color plate C. 51, page 273).

these defects can be identified by testing the geometric properties of a 12-atom neighborhood, which has already been done successfully by domain scientists. For the other crystal types (BCC or HCP) such a classification is not yet known. Therefore, in the remainder of this paper we will only handle FCC crystals.

The topological structure induced by such defects and their interplay helps evaluate the material properties. The identification of dislocations and stacking fault structures from atomistic simulations is similar to techniques for skeleton extraction from discrete data, since only particle positions are given. In this paper we present an approach that extracts the topological structure of dislocations and stacking faults from atom positions, and tracks them throughout the simulation. Our proposal is composed of a segmentation algorithm based on simplification of a neighborhood graph that allows defects to be identified and tracked over time. In particular, we identify the formation of junctions among dislocations and stacking faults. In Figure 1 we show different steps of our algorithm using a MD simulation of a block of compound material underlying a stretching force. The two basic materials are Ni and Ni_3Al. The data set originally contains about 1,200,000 atoms. Since the data is filtered, which is described in section 3, the loaded files only contain between 13,000 and 87,000 atoms, depending on the scene complexity.

2 Background Material and Related Work

Several works in MD discuss the evaluation of material properties under external forces (see Bulatov et al. [4] for a good introductory book on the subject). Since the atomic structure of many solid materials in normal conditions corresponds to crystal lattice structure, defects are observed when changes to the lattice structure occur. Changes in the regularity, symmetry, or ordering of this 12-atom neighborhood create topological defects on the atomic structure [18]. Atoms with irregular neighborhood align in 1-D dislocations surrounding 2-D stacking faults. Geometric measures give one way

to evaluate these defects, such as the Burgers vector [4] that represents the magnitude and direction of the lattice distortion of a given dislocation. Several papers discuss how such geometric measures can be used to track the motion of dislocations. For instance, Schall et al. [16, 17] discuss ways to track dislocation in colloidal crystals, and techniques to visualize the distribution of Burgers vectors (called Nye tensors). Hartley et al. [11] also discusses a similar approach that measures Nye tensor distributions. Topological analysis of the interplay of dislocations is more elaborate if done at the atomistic level, and therefore meso-scale representations are often used. Dislocation dynamics is an example of such a meso-scale simulation. The work by Lipowsky et al. [13] discusses a way to track and visualize dislocation directly in grain-boundary scars. Bulatov et al. [3, 5] suggests tracking topological features of dislocations, such as junctions or multi-junctions (three or more dislocations in one place). They observe in [6]: "In large-scale dislocation dynamics simulations, multi-junctions present very strong, nearly indestructible, obstacles to dislocation motion and (...) thereby playing an essential role in the evolution of dislocation microstructure and strength of deforming crystals simulations".

Our proposal is based on extracting a skeleton-like structure from the atomistic data, which identifies features like dislocations and junctions between them. There is a vast literature on skeletonization algorithms, and a good introductory survey can be found in [8]. In this survey there are four classes of skeletonization algorithms: thinning and boundary propagation, distance-field based, geometric methods, and general-field functions. In our work no implicit boundary is formed, but a neighborhood graph [1,19] that defines atom proximity and allows thinning contraction operations. This is similar to thinning algorithms in distance-field methods, with respect to the approach, but different since no distance field is used. Several possibilities for controlling contraction operations can be defined, and smoothing procedures are often employed [2]. We use a simple mass-spring approach (see Section 3.1). Geometric methods are based on proximity structures, such as the Voronoi Diagram, or structures derived from Morse Theory, such as Reeb Graphs or Contour Trees [7,9,10]. General field methods such as potential field functions are used in a similar application described in [14]. In their work, crystal dislocation data is given as a potential field function, and a Morse-Smale complex is used to evaluate the structure of dislocations. Since we do not have such a field, we can not employ these methods.

3 Structure Defects Extraction

Visualizations for dislocation dynamics simulations can use the meso-scale data to construct concise representations of the dislocations, which allow easy tracking over time. In atomistic simulations there is no such data description. To generate an analogous visualization, extraction of dislocations from the atomistic data is necessary. We propose an algorithm with the following steps:

1. Reconstruction of the neighborhood graph
2. Extraction of dislocations
   2.1. Contraction of the graph using a simple mass-spring system
   2.2. Edge collapse

2.3. Dislocation junction identification
2.4. Atom data segmentation
3. Extraction of stacking faults
4. Tracking over time.

The input data consists of a list of atoms A, each storing its position, a unique ID and a lattice classification c_a, for the atom $a \in A$, based on the nearest neighbors of the atom according to [12]. We will use this classification to identify atoms creating defects. We assume that the undistorted crystal of our material forms an FCC lattice. Atoms with such a neighborhood are classified $c_a = 0$. The atoms which form a HCP lattice have $c_a = 1$. Atoms which have 12 closest neighbors but neither form a FCC lattice nor a HCP lattice have $c_a = 2$. All remaining atoms have $c_a = 3$. Since we are interested in only visualizing the crystal defects, we can completely omit the more than 90 percent of the atoms in class $c_a = 0$. Changes from FCC to HCP ($c_a = 1$) indicate planar displacements and hence stacking faults while the other two classes form dislocations (the classification can be seen in all images showing the original data, e. g. figure 1 a & d: red atoms are $c_a = 1$, green $c_a = 2$, and yellow $c_a = 3$). Using this lattice classification for identifying crystal defects is much easier to compute than Burgers vectors for all atoms.

To extract the topological structure, we first need to define neighborhood information between atoms. We create a neighborhood graph using a cutoff radius automatically chosen, based on the domain knowledge that atom distances are uniform within narrow bounds as shown in figure 2. This is a graph similar to the one used by the MD simulation. We have to recreate this graph since the simulation graph was not stored with the MD data due to data file size considerations.We propose to simplify this graph using a very simple mass-spring system. We define N to be the list of all nodes in the graph, where each node $n \in N$ stores a set of atom indices $a_i \in n$ for all atoms which are represented by this node. Initially we create one node for each atom $n_i = \{a_i\}$ and place the node at the position of that atom $p(n_i) = p(a_i)$. We create edges between nodes of atoms which are within the neighborhood radius. These edges are stored in three lists E_d, E_s, and E_b, depending on the atom classifications c_a, since these edges will be used by different parts of our algorithm. E_d holds edges connecting two nodes which are initialized with two atoms a_1, a_2 with $c_{a_1} > 1 \wedge c_{a_2} > 1$ (shown in green). These edges form *dislocations* and will be used by the graph contraction in the dislocation extraction described in section 3.1. E_s holds edges connecting a_3, a_4 with $c_{a_3} = 1 \wedge c_{a_4} = 1$ (shown in gray). These edges form *stacking faults* and will be used in our region growing approach described in section 3.4. E_b (*border*) holds all remaining edges of the neighborhood graph (shown in red). These edges separate dislocations and stacking faults. They will ensure that the graph contraction does not destroy small stacking faults and they will be used to terminate the region growing.

3.1 Graph Contraction

The first step of the graph simplification is a contraction implemented using a simple mass-spring system. Each node n_i manages a speed vector v_i, which is initialized to zero. However, only nodes connected to at least one edge $e \in E_d$ can be moved, while

other nodes are fixed. All edges of E_d and E_b are treated as springs. Edges $e \in E_d$ try to collapse to length of zero and edges $f \in E_b$ try to keep their initial length. These edges f will keep the subgraph of the edges e out of the stacking fault regions and will prevent our method from collapsing small stacking fault regions. This is why we are using a mass-spring approach instead of simple clustering (e. g. shortest-edge-first). This way we can collapse edges of various lengths and still be sure not to loose small features.

In addition to simply moving the nodes in the graph we collapse small edges. The corresponding threshold is based on the neighborhood radius which was used at initialization to build the graph. In our examples we obtained good results using a collapsing threshold which is half of the neighborhood radius. However, this value might be adjusted by the user. When collapsing the edge $e_i \in E_d$ we combine all attributes of the two original nodes $n_{e_i,1}$ and $n_{e_i,2}$ weighted by the relative number of atoms assigned to each node: $n_{e_i,1} = n_{e_i,1} \bigcup n_{e_i,2}$, $p(n_{e_i,1}) = (|n_{e_i,1}| + |n_{e_i,2}|)^{-1}(|n_{e_i,1}|p(n_{e_i,1}) + |n_{e_i,2}|p(n_{e_i,2}))$ (and all other attributes, e. g. speed vector, accordingly). Note that the position of the node $p(n_{e_i,1})$ is no longer directly related to the positions of the atoms it contains. We iterate this procedure until the graph reaches a stable status, which is measured by the maximum overall speed of all nodes (usually after 50 to 80 iterations). A result can be seen in figure 3.

3.2 Graph Simplification

Although the resulting graph visually seems to be the structure we want to extract, it is still too complex for the upcoming segmentation of the atom data. We need to separate nodes in the graph which represent dislocations from nodes which represent junctions of dislocations. We propose two different approaches to classify nodes in the graph as junctions or non-junctions: based on the degree of the node or based on the positions of the direct neighboring nodes. The first approach is trivial (using only the list of edges). All nodes connected to more than two edges can be considered junctions. However, the graph data created by the mass-spring system still contains structures that are too complex for this method to work as intended, such as *unclean* and *thicker* junction regions (see upper-right region of lower image in figure 3). Therefore we use a second method where we check the directions of the edges connected to the node. When classifying the node n_i, we define E_i to be the set of connected edges $E_i = \{e_j \in E_d | n_i = n_{e_j,1} \vee n_i = n_{e_j,2}\}$ and N_i the set of nodes directly connected by these edges $N_i = \{n_k \in N | \exists e_j \in E_i : n_k = n_{e_j,1} \vee n_k = n_{e_j,2}\} \setminus \{n_i\}$. Given V_i a set of normalized vectors representing the edges of E_i formed between the positions $p(n_k)$ and a supporting position $p'(n_i) = (|N_i|)^{-1} \sum_{n_k \in N_i} p(n_k)$, where n_k is the second node connected to the edge apart from n_i. The supporting position is the mean of all neighbor positions. It is used to compensate for small curvature in the dislocations. If the dot product between all pairs of vectors in V_i is 1 or -1 (within a small threshold) the vectors form a linear structure and the node n_i is classified as a non-junction.

Using this classification we continue the simplification of the graph by again employing a mass-spring system, similar to the one presented in section 3.1, but without considering edges from E_b. In addition, we increase the collapsing threshold, but only collapse edges between nodes which are both junctions or both non-junctions. The classification is recalculated after each iteration. We terminate this phase of our algorithm

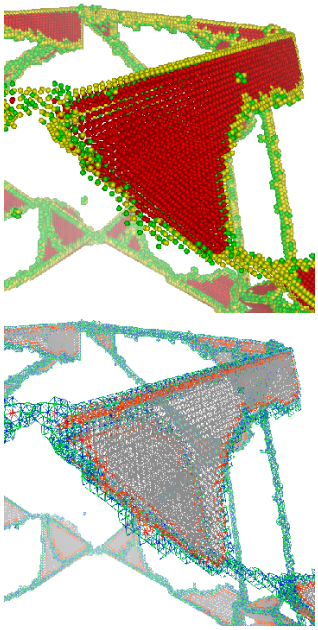


Fig. 2. Construction of the neighborhood graph (bottom) from the original atom data (top). Edges are color-coded: green/blue edges are from E_d, red/orange from E_b, and gray from E_s (color plate C. 53, page 274).

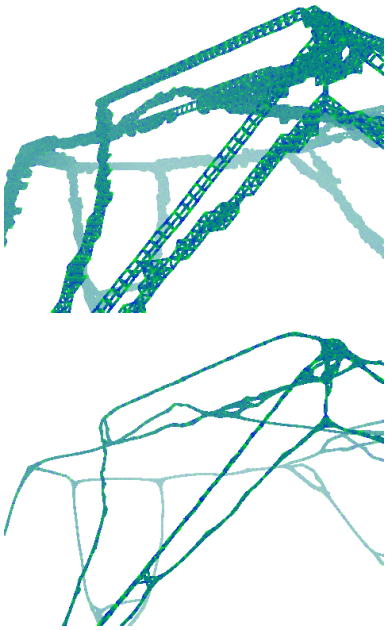


Fig. 3. Top image: initial neighborhood graph (only edges from E_d are shown); bottom: graph after contraction of the mass-spring system (50 iterations). Note the *unclean*, *thicker* regions near junctions (color plage C. 54, page 274).

when the graph stabilizes (no more edge collapsing). The left image of figure 6 shows the result from the graph in the bottom image of figure 3.

Figure 4 shows a simplified situation in which the graph can reach a stable state, thus preserving cycles inside one linear structure, when all edges of this cycle are too large for the collapsing threshold. There is also a very rare case where cycles of four edges are formed within a single dislocation. This happens due to the varying size of the original atom data, and thus the varying size of the neighborhood graph near this location. The four edges form a diamond structure connecting two junctions and two non-junctions, where the two non-junctions only hold very few atoms in their sets (figure 5). We use heuristics to identify these situations and collapse these cycles into single nodes.

3.3 Atom Data Segmentation

The simplified graph represents the topology of the original data set and can be used to segment it into individual dislocations. To use this graph directly for segmentation we further simplify its structure, by collapsing all edges connecting nodes of the same classification (junction or non-junction). We remove all atoms $a_j \in n_j$ from the sets

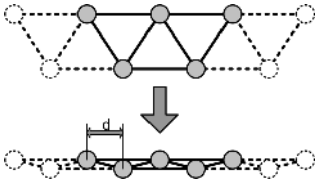


Fig. 4. The mass-spring system contracts the relative neighborhood graph to the lower image. When the distance d between atoms is large, edges do not collapse and the three-edge cycles become stable.

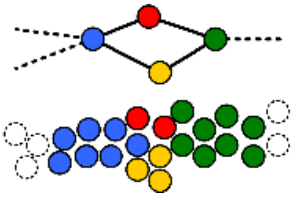


Fig. 5. Four nodes and edges in the graph forming a diamond which represents a single dislocation. Yellow and red atoms are incorrectly separated.

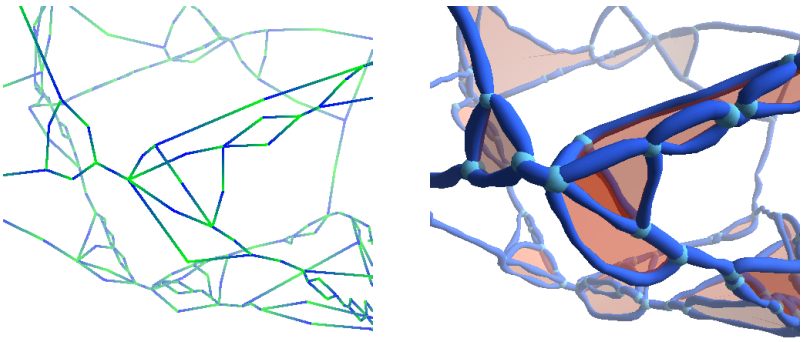


Fig. 6. Left: the graph after all simplification steps; Each edge connects a junction to a non-junction node. All atoms are associated with non-junction nodes. Right: the final result; Junctions are represented by cyan spheres, dislocations by blue tubes, and stacking faults by orange planes (color plate C. 52, page 273).

of junction nodes $n_j \in N$, by reassigning these atoms to the set of the connected non-junction node which contains the atom which is closest to the atom to be reassigned $n_i \in N$ with $\min_{a_i \in n_i} |a_i - a_j|$. After this operation all junction nodes have empty atom sets. However their position is calculated using the atoms which they contained before reassignment.

After this operation is finished, each edge in the graph connects a junction node to a non-junction node, and all atoms are assigned to non-junction nodes. The segmentation of the atom data is now done by assigning all atoms in the set of each non-junction node the same ID value. For the final representation of dislocations we fit a tube into each atoms segment. Since the junction nodes define the end points of each dislocation, a simple linear distance threshold based fitting of spline tubes can be used. Our final results (figure 6) also show junctions as spheres. These are placed at the positions of the junctions nodes and use a radius depending on the radii of the connected segments.

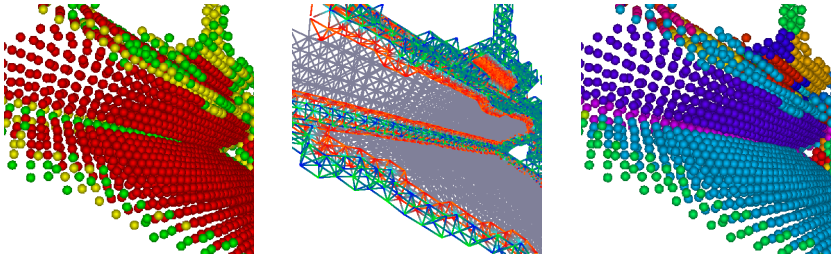


Fig. 7. Shows a problematic situation for the stacking fault segmentation using region growing, from left to right: The original data set; In the relative neighborhood graph atoms from different stacking fault segments are connected; the final segmentation (color plate C. 55, page 274).

3.4 Stacking Fault Extraction

After we extracted the dislocation structures and created a concise representation using tubes, we can now focus on creating a similar visualization on the stacking faults. Since stacking faults are always aligned to a crystal plane we can use a flat polygon as representation. Furthermore stacking faults are always bordered by dislocations. Since we already extracted the dislocations as tubes, we can just connect them with a polygon.

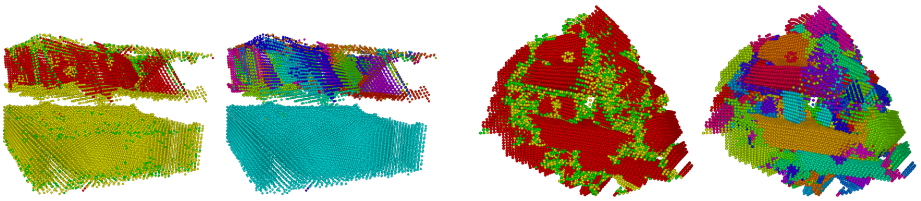
The main idea of segmenting the atom data for the stacking faults is to perform a region growing on the neighborhood graph using edges $g_i \in E_s$. However, the data we used have crystal planes meeting under sharp angles (60 degree), which results in neighboring stacking faults meeting at one dislocation and being connected in the graph (figure 7). Simply ignoring edges from E_b is not possible, since some stacking faults are only one atom layer thick and therefore do not have edges from E_s. To fix this, we define a border region of atoms N_B to be a set of all atoms connected by at least one edge $g_i \in E_s$ and at least on edge $e_j \in E_b$. We define N_G to be the set of atoms connected only to edges $g_i \in E_s$, and grow a region outwards from a seeding point chosen from N_G. We assign a new segment ID for this region and repeat this method until all atoms from N_G are segmented. However, atoms from N_B are not guaranteed to be assigned to a segment. To assign these atoms to a segment we look at all neighbor atoms of N_B which are $a_i \in N_B \bigcup N_G$ and have already a segment ID. We save the segment ID which is assigned to most neighbors for the atom we want to segment, and we assign these IDs all at the same time at the end of an iteration step. By doing so, all segments grow with the same speed (one atom per iteration). Since the border area is equally thick (one atom) we get clean results in the problematic scenarios described above.

In our second step we collapse all edges $g_i \in E_s$ which connect nodes with the same segment ID by merging atom sets, similar to the approach of the dislocation extraction. Edges $f_i \in E_b$ are used to determine correspondences between the stacking fault segments and the dislocation segments. We sort the list of dislocation segments based on the junction nodes they use to form a cycle surrounding the stacking fault. We then simply fit a polygon into the atom set and stitch it to the surrounding dislocation cycle (e. g. sub8/figures 6 and 9). Rendering these polygons using transparency also reduces the occlusion problem compared to the original atomistic representation.

3.5 Segment Tracking

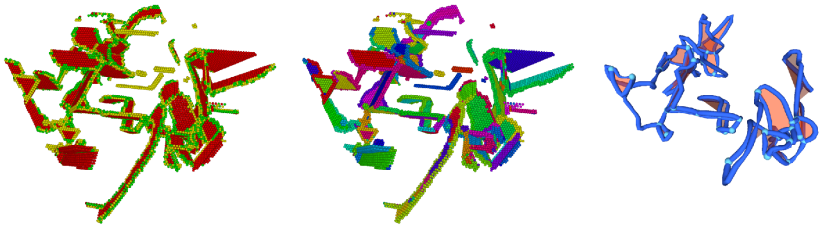
The last remaining feature of our visualization is the tracking of the extracted features (dislocations and stacking faults) over time. There have been many publications on feature tracking in general (e. g. by Samtaney et al. [15]). However, we can use a simple approach by utilizing the fact that all of our features still know the atoms they were created from. Using the unique atom IDs we can easily trace the atoms over time. We perform the segmentation of the atom data independently for each time step. The actual tracking is done by relabeling the extracted segments with IDs from the last time step. We compute the overlap of the atom sets of the segments in one time step with the sets from the previous time step and pair those segments with the largest overlap. Although we handle dislocations and stacking faults independently in this tracking process, we apply the same method to both feature types. Due to our graph simplification, our results are a bit unsteady at the junction regions, but the tracking shows that even in these regions the main segments are quite stable.

4 Results and Future Work



(a)Data set with two layers (upper one Ni_3Al; lower one Ni) pulled apart.

(b)Data set with only Ni_3Al pulled in one direction.



(c)Data set with Ni atoms pulled in one direction.

Fig. 8. Visualization examples. Each sequence shows the original data set with the lattice classification (left) and the results of the atom data segmentation (right; center in figure 8(c)). Right image in figure 8(c) shows the results of the structure extraction (color plate C. 56, page 275).

We presented an approach for generating simplified representations of MD data sets containing crystal structure defects, especially dislocations and stacking faults. We thereby created a connection between visualizations of atomistic data and data from dislocation dynamics. The proposed feature extraction and tracking is performed in a

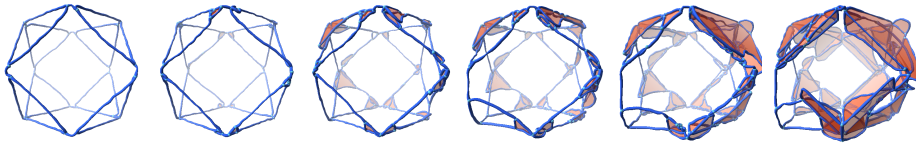


Fig. 9. Extracted crystal defects of time steps 10, 250, 350, 400, 500 and 595.

preprocessing step. On a common desktop computer (Intel Core2Duo 6600@2.40 GHz, 2 GB RAM, NVidia GeForce 7900 GT) these calculations take an average time of about 20 seconds per time step (depending on the complexity of the structure of the crystal lattice defect to be extracted) for the Ni-Ni_3Al data set mainly used in this paper. These preprocessing results can then be visualized and explored interactively on the same machine. The rendering of the atoms is done using point-sprites, so we have no problems interactively rendering (more than 10 FPS) up to millions of atoms. However, we only have this many atoms in artificial test data sets. The real world data sets we worked with contained between 13,000 and 87,000 atoms, and can be rendered with far more than 60 FPS. The structure extracted is stored as a triangle mesh. Our data sets resulted in meshes containing between 28,000 and 93,000 triangles. Interactive exploration of the time-dependent data set can be done employing an out-of-core streaming approach.

In addition to the Ni-Ni_3Al data set, we tested our approach with several other data sets, all from MD simulations and all containing crystal structure defects (Figure 8). Figure 8(a) shows a simulation of a probe consisting of one layer of Ni_3Al and one layer of Ni. The two layers are pulled apart. However, the feature extraction cannot be performed because the lower layer has a BCC lattice (see discussion below). The other two of these data sets are simulations of clean materials (Ni_3Al in figure 8(b) and Ni in figure 8(c), each pulled in one direction). In the Ni_3Al data set the stacking faults group together in big blocks, thus making our approach of using single flat polygons inapplicable. Nevertheless, just segmenting the atoms allows a clearer view of the crystal planes in complex regions of the material. The Ni data set shown in figure 8(c) also works with our feature extraction (right image). However, our current implementation misses some of the features due to some problems with tracking the atom segments over the periodic boundary conditions of the simulation area (truncated octahedron). Figure 9 shows the crystal structure evolution of the Ni-Ni_3Al data set in six relevant time steps. The data sets does not change significantly until about time step 300. Then dislocations start to split up and change their form and position (e. g. in the lower left area, in front) and large stacking faults emerge (e. g. in the upper right area).

Our approach still has room for improvements. In particular, the heuristics applied to simplify the neighborhood graph are not completely robust, although they work very well on all data sets we used and they are based on knowledge from the application domain. We want to enhance their quality by taking the coherence between different time steps into account. The result of the atom data segmentation could be used in a following time step as starting point. Our initial tests are encouraging.

Another problem already mentioned earlier is that our approach currently only works for FCC crystals, because we rely on the atom neighborhood classification in

the input data for identifying atoms forming dislocations and junctions (see [12]). We want to extend our approach based on a similar classification to BCC crystals. However, it is not clear at the moment how such classification can be made.

Domain scientist partners in this project gave us very positive feedback on our visualization. The clear representation of the crystal planes is really beneficial, as well as the tracking of the dislocation. The visualization aids them in perceiving the structure of the data and its evolution over time. The semi transparent rendering of the stacking faults not only makes the occlusion less problematic, it also allows the user to identify the crystal planes more easily on which they occur. Since all features and the tracking method rely only on the original atom data, they believe that this visualization is valid. There is currently no visualization tool available which is capable of representing atomistic data sets from MD simulations in the described manner. For future work we want to optimize our approach, and integrate our visualization in a framework allowing better user interaction, making our tool more usable for the domain experts.

Acknowledgements

This work is partially funded by Deutsche Forschungsgemeinschaft (DFG) as part of SFB 716. The work of J. Comba and C. Dietrich is supported by CNPq grant 485853/2007-0.

References

1. Pankaj K. Agarwal and Jiří Matoušek. Relative neighborhood graphs in three dimensions. In *SODA '92: Proceedings of the 3rd annual ACM-SIAM symposium on Discrete algorithms*, pages 58–65, Philadelphia, USA, 1992. SIAM.
2. Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Trans. Graph.*, 27(3):1–10, 2008.
3. Vasily Bulatov, Farid F. Abraham, Ladislas Kubin, Benoit Devincre, and Sidney Yip. Connecting atomistic and mesoscale simulations of crystal plasticity. *Nature*, 391:669–672, 1998.
4. Vasily Bulatov and Wei Cai. *Computer Simulation of Dislocations*. Oxford University Press: Oxford, New York, 2006.
5. Vasily Bulatov, Wei Cai, Jeff Fier, Masato Hiratani, Gregg Hommes, Tim Pierce, Meijie Tang, Moono Rhee, Kim Yates, and Tom Arsenlis. Scalable Line Dynamics in ParaDiS. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 19, Washington, DC, USA, 2004. IEEE Computer Society.
6. Vasily V. Bulatov, Luke L. Hsiung, Meijie Tang, Athanasios Arsenlis, Maria C. Bartelt, Wei Cai, Jeff N. Florando, Masato Hiratani, Moon Rhee1, Gregg Hommes, Tim G. Pierce, and Tomas Diaz de la Rubia. Dislocation multi-junctions and strain hardening. *Nature*, 440:1174–1178, 2006.
7. Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. *Comput. Geom. Theory Appl.*, 24(2):75–94, 2003.
8. Nicu D. Cornea, Patrick Min, and Deborah Silver. Curve-Skeleton Properties, Applications, and Algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007.

9. Herbert Edelsbrunner, John Harer, Ajith Mascarenhas, Valerio Pascucci, and Jack Snoeyink. Time-varying reeb graphs for continuous space–time data. *Comput. Geom. Theory Appl.*, 41(3):149–166, 2008.
10. Attila Gyulassy, Vijay Natarajan, Valerio Pascucci, Peer-Timo Bremer, and Bernd Hamann. A Topological Approach to Simplification of Three-Dimensional Scalar Functions. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):474–484, 2006.
11. C.S. Hartley and Y. Mishin. Characterization and visualization of the lattice misfit associated with dislocation cores. *Acta Materialia*, 53:1313–1321, 2005.
12. J. Dana Honeycutt and Hans C. Andersen. Molecular dynamics study of melting and freezing of small Lennard-Jones clusters. *Journal of Physical Chemistry*, 91(19):4950–4963, 1987.
13. Peter Lipowsky, Mark J. Bowick, Jan H. Meinke, David R. Nelson, and Andreas R. Bausch. Direct visualization of dislocation dynamics in grain-boundary scars. *Nature Materials*, 4:407–411, 2005.
14. Vijay Natarajan, Prof Herbert Edelsbrunner, Prof Lars Arge, Prof John Harer, and Prof Xiaobai Sun. *Topological Analysis of Scalar Functions for scientific Data Visualization*. PhD thesis, Duke University, 2004.
15. Ravi Samtaney, Deborah Silver, Norman Zabusky, and Jim Cao. Visualizing features and tracking their evolution. *Computer*, 27(7):20–27, 1994.
16. Peter Schall, Itai Cohen, David A. Weitz, and Frans Spaepen. Visualization of Dislocation Dynamics in Colloidal Crystals. *Science*, 24:1944–1948, 2004.
17. Peter Schall, Itai Cohen, David A. Weitz, and Frans Spaepen. Visualizing dislocation nucleation by indenting colloidal crystals. *Nature*, 440:319–323, 2006.
18. James P. Sethna. *Statistical Mechanics: Entropy, Order Parameters and Complexity*. Oxford University Press: Oxford, New York, 2006.
19. Kenneth J. Supowit. The Relative Neighborhood Graph, with an Application to Minimum Spanning Trees. *J. ACM*, 30(3):428–448, 1983.

Extracting and Visualizing Structural Features in Environmental Point Cloud LiDaR Data Sets

Patric Keller[1], Oliver Kreylos[2], Marek Vanco[2], Martin Hering-Bertram[3], Eric S. Cowgill[4], Louise H. Kellogg[4], Bernd Hamann[2], and Hans Hagen[1]

[1] University of Kaiserslautern, Department of CS., 67663 Kaiserslautern, Germany
[2] Institute for Data Analysis and Visualization (IDAV), Department of Computer Science, UC Davis, CA 95616
[3] Fraunhofer-Institute ITWM, 67663 Kaiserslautern, Germany
[4] University of California, Department of Geology, UC Davis, CA 95616

Abstract. We present a user-assisted approach to extracting and visualizing structural features from point clouds obtained by terrestrial and airborne laser scanning devices. We apply a multi-scale approach to express the membership of local point environments to corresponding geometric shape classes in terms of probability. This information is filtered and combined to establish feature graphs which can be visualized in combination with the color-encoded feature and structural probability estimates of the measured raw point data. Our method can be used, for example, for exploring geological point data scanned from multiple viewpoints.

1 Introduction

Exploration of environmental point data sets is a challenging problem of recent interest. The LiDaR (Light Detection and Ranging) technology makes it possible to capture fast and accurate point data over large regions. As a consequence,
LiDaR data sets are typically very large containing diverse and highly complicated objects. Occlusion, semi-transparent and reflecting objects, structures with small fractured and under-sampled surface components like trees, as well as scanning error due to motion of cars, humans, animals, etc. make the data difficult to interpret. While the raw data contains scientifically relevant structural information, this information, unless proper filtering and pre-processing methods are applied to the raw data, remains ”hidden” in most direct visual representation due to data over-load and noise.

A possible way to explore point data is based on surface reconstruction [4, 9, 10]. Building consistent triangular meshes from such data is a difficult and often ambiguous task. Point-based approaches [1, 24] operate directly on the sample points without requiring the computation of mesh-based connectivity information. These approaches often use surface elements (*surfels*), assuming that point sets define smooth surfaces with well-defined normal vectors. For LiDaR data sets, this assumption may not always hold and the surfel-based approach may also hide topological ambiguities and, as a consequence of smooth interpolation schemes, eliminate structural artifacts, due to discontinuities in the original scanned geometry, that might be important. To avoid these drawbacks, we use a probability-based classification of the point cloud into subsets having different structural characteristics. Depending on this classification and a user's

choices of parameter values we perform the extraction of feature graphs describing the structural composition of the point cloud. We obtain an explicit structural description of the whole data which can be used for further processing (e.g., identification and classification of objects in environmental data) or for exploration and visualization.

For geologists, our approach is of special interest since it makes possible the detection and visualization of features like creases in structures caused by earthquakes or rapid and automatic identification of key features in environmental data such as ridge lines, stream beds, and edges of terraces. When combining data from multiple scans, registration errors make feature detection and visualization even more challenging. Our goal is to provide a user with effective tools for the identification and interactive exploration of structural features in LiDaR data sets.

In section 2, we review related work on point set rendering and surface reconstruction. Section 3 discusses the specific problems of analyzing LiDaR data and proposes a classification of features used in our algorithm. The construction of feature graphs is discussed in section 4.

2 Related Work

The potential of point-based methods has been demonstrated in different applications [2]. However, algorithms to process point sets are still in their infancy when compared with algorithms for common mesh-based approaches. In computer-aided design (CAD), point sets may be used as underlying representation for surface editing [24] (like the Pointshop 3D framework proposed by Zwicker et al. [28]). Several authors based their segmentation and surface recognition algorithms directly on point sets, [6], [7]. Concerning surface reconstruction from laser-scanned point sets, much research has been done following the approach by Eck and Hoppe [13].

In the context of complex geology-driven applications, besides the problem of correctly recovering surface topology, sharp feature lines like ridges/ravines or crest lines [17], [27], [25] need to be identified. Concerning point sets, there have been several efforts for robust feature detection [15] [5] [8]. Neal [23] provided an instructive survey of such techniques. However, these approaches do not differentiate between the structural types a feature might belong to, which makes them inappropriate for data sets of arbitrary topology especially point clouds resulting from environmental scans. Some of these methods perform feature extraction based on the estimation of curvature values obtained from surfaces that are approximated based on a local point neighborhood via methods like moving least squares (MLS) [20]. The principal curvatures and curvature directions carry important information about local surface behavior, but their estimation is numerically sensitive to noise present in the data, and many estimation methods have been proposed [11], [16], [18]. Hamann [12] [16] approximated principal curvature values based on considering a local, triangulated point neighborhood and determining a least-squares quadratic bivariate polynomial. Feature lines also produce important landmarks for constructing meshes [14], [21] or for recognizing shapes like buildings [3]. Regarding the post-processing of laser range data, a variety of techniques improving the quality can be applied [22], [26].

3 Feature Detection in LiDaR Data

Our work represents a first step toward the over-arching scientific goal of identifying and extracting regions in LiDaR data sets matching individual structural characteristics. One long-term goal was to be able to automatically differentiate from natural and man-made objects (e.g., trees and buildings). Detecting and joining objects corresponding to similar structural classes is another task we wish to accomplish. To reach these goals it is necessary to provide methods for detecting and extracting special structural features like corners, border-, crease- or ridge-lines in the form of a feature graph which can be used to perform further processing. In general, feature detection in environmental indexLiDAR LiDaR data is a complicated task due to the nature of this kind of data. LiDaR data are generally collected either on the ground, using tripod-mounted scanners, from the air, using airplane mounted scanners or from satellites. These acquisition methods produce data sets typically exhibiting substantial noise levels since the measurement is taken from great distance. Under-sampling, occlusion and movements in the scanned environment are just a few more problems to mention.

Hence, we must assume that the resulting point cloud data in general does not provide a reliable base to directly perform feature extraction. We introduce a method that classifies points by considering whether they are part of a surface, a curve or a junction of either one. This approach allows us to adapt the feature graph extraction to the proper point class. By using stochastic and multi-scale-based means when processing the raw point cloud (especially for processing huge data sets) we introduce a rather stable approach capable of handling robustly relatively high noise levels in a data set. Our method combines the following steps:

1. Pre-processing of the 3D point cloud (sec. 3.1). This step includes an hierarchical decomposition of the point cloud into an octree-based voxel grid as well as the elimination of outlier points (sec 3.1).
2. Likelihood-based point classification depending on the characteristics of the local point neighborhood as well as computation of feature-specific values based on multiple scales (sec. 3.2).
3. Post-processing by smoothing the obtained feature values (sec. 3.4).
4. Feature graph construction (sec. 4).

3.1 Pre-processing

Hierarchical Point Cloud Decomposition: Since we are primarily dealing with environmental LiDaR data sets that are typically the result of several high-resolution scans (several million or even billion points) efficient data handling is important. Organizing the points into an octree-based voxel structure allows us to speed up frequently used operations like k-nearest neighbor search. To be able to visualize large-scale data, we follow an additional level-of-detail (LOD) approach. We modify point sets of each octree level to match a level-specific resolution and store them on disc. This approach allows us to dynamically reload point sets of a certain resolution if necessary.

Outlier Removal: Environmental LiDaR scans do not distinguish between the significant or irrelevant components; everything is scanned e.g., trees and other vegetation. Weyrich et al. [26] proposed a method assuming that potential outliers draw their local neighborhood from a larger vicinity than points within a well-sampled environment. Instead of using a graph-based approach we compute the mean distance of a point to its k-nearest neighbors and filter out all points whose mean distance exceeds a given threshold. To avoid loosing important structural information this threshold is chosen interactively by the user.

3.2 Stochastic Point Classification

The goal of point classification is to decompose the point cloud P into subsets of points having different structural properties. We determine whether a point $p \in P$ is part of a curve- or a surface-like structure or is treated as a so-called critical point. We call those points critical points that are not part of a surface or a curve. This point type represents junctions of features lines (e.g., at corners) or borders/intersections between surfaces and/or curves. In general they are meant to represent discontinuities not associated with these structures. Hence, critical points are not suitable for directly performing methods of feature-line extraction but are used in the context of graph extraction (sec. 4) to represent underlying discontinuities.

In order to identify the type of a point p we determine the "shape" of the local point neighborhood $\mathscr{N}_r(p)$ of p using principal component analysis (PCA). Let $\mathscr{N}_r(p)$ denote the point neighborhood within a sphere of radius r centered at p. The radius is specified by the user and depends on the features being focused on. Determining the shape of $\mathscr{N}_r(p)$ is accomplished by comparing the distribution of $\mathscr{N}_r(p)$ in each dimension. The distribution is measured by using the eigenvalues of the covariance matrix $\mathbf{C}$ of $\mathscr{N}_r(p)$. Let $\lambda_0 \leq \lambda_1 \leq \lambda_2$ be the eigenvalues of $\mathbf{C}$. It is a well-known fact that $\mathscr{N}_r(p)$ has a spherical shape if $\lambda_0 \approx \lambda_1 \approx \lambda_2$, a disc-like shape if $\lambda_0 \ll \lambda_1 \wedge \lambda_1 \approx \lambda_2$ or a cylindrical shape if $\lambda_0 \approx \lambda_1 \wedge \lambda_1 \ll \lambda_2$. Using this information allows us to establish three point classes by defining the following sets:

$$\mathscr{P}_{cp} = \{p \in P | \lambda_0/\lambda_2 \geq \varepsilon\} \tag{1}$$

$$\mathscr{P}_{c} = \{p \in P | \lambda_1/\lambda_2 < \varepsilon\} \tag{2}$$

$$\mathscr{P}_{d} = \{p \in P | \lambda_0/\lambda_2 < \varepsilon\} \tag{3}$$

Here, $\mathscr{P}_{cp}, \mathscr{P}_c, \mathscr{P}_d$ define the points with spherical, cylindrical and disc-like neighborhood (with $\mathscr{P}_{cp} \cap \mathscr{P}_c = \emptyset$, $\mathscr{P}_{cp} \cap \mathscr{P}_d = \emptyset$ and $\mathscr{P}_c \subseteq \mathscr{P}_d$). The variable ε is a feature-specific value and determines when a point becomes a critical point. Usually ε is chosen to be $\frac{1}{2}$ but it can be adapted to meet a user's needs. For example, reducing ε increases the sensitivity of a point for being classified as a critical point, causing some points of $\mathscr{P}_c$ and $\mathscr{P}_d$ to migrate to $\mathscr{P}_{cp}$. This effect can be geometrically interpreted as extending the boundaries around a discontinuity (e.g., an intersection). This form of point differentiation implies $\mathscr{P}_c \subseteq \mathscr{P}_d$, since in some cases surfaces collapse to line-like structures. However, this characterization heavily depends on the size of the local neighborhood $\mathscr{N}_r(p)$ and would not lead to reliable shape evaluation results in presence of noise. Thus, we allow the neighborhood $\mathscr{N}_r(p)$ to vary in size by a given percentage of its original magnitude. We compute probability estimates by measuring the number of times $\mathscr{N}_r(p)$ can be assigned to one of the classes defined by (1), (2), (3) by evaluating

$$\mathscr{L}_{cp,c,d}(p) = \frac{1}{n+1} \sum_{j=0}^{n} \varphi_{cp,c,d}^{r_{min}+j\delta}(p), \tag{4}$$

$$\varphi_{cp}^{s}(p) = \begin{cases} 1 & \text{if } \lambda_0^s \geq \varepsilon\lambda_2^s \\ 0 & \text{else} \end{cases}, \varphi_{c}^{s}(p) = \begin{cases} 1 & \text{if } \lambda_1^s < \varepsilon\lambda_2^s \\ 0 & \text{else} \end{cases},$$

$$\varphi_{d}^{s}(p) = \begin{cases} 1 & \text{if } \lambda_0^s < \varepsilon\lambda_2^s \\ 0 & \text{else} \end{cases}$$

Here r_{min} (r_{max}) denotes the minimum (maximum) radius and n represents the number of considered neighborhoods. Hence, $\delta = (r_{max} - r_{min})/n$ and λ_i^s being the eigenvalues of $\mathbf{C}^s$ of $\mathscr{N}_s(p)$. $\mathscr{L}_{cp}, \mathscr{L}_c$ and $\mathscr{L}_d$ defining the likelihood of a point p corresponding to the shape classes defined by $\mathscr{P}_{cp}, \mathscr{P}_c$ and $\mathscr{P}_d$. We now divide the original point cloud based on the likelihood values into subsets of different character. For graph extraction we use additional feature values capturing the strength of an underlying feature. The feature values I_{cp}, I_c and I_d for the points of the classes $\mathscr{P}_{cp}, \mathscr{P}_c$ and $\mathscr{P}_d$ are defined by

$$I_{cp,c,d}(p) = \frac{1}{n+1} \sum_{j=0}^{n} f_{cp,c,d}^{r_{min}+j\delta}, \tag{5}$$

$$f_{cp}^{s} = \lambda_0^s \lambda_1^s / (\lambda_2^s)^2 \tag{6}$$

$$f_{c}^{s} = (\lambda_2^s - \lambda_1^s)\lambda_0^s / (\lambda_2^s \lambda_1^s) \tag{7}$$

$$f_{d}^{s} = \kappa^s \tag{8}$$

where κ^s denotes the maximum absolute curvature of polynomial patches fitted to $\mathscr{N}_s(p)$. Since we are also interested in feature extraction from surface-related structures, using curvature is an obvious approach. The feature values I_{cp}, I_c and I_d express the strength of a feature associated with a point p by a value averaged over multiple scales. The values f_{cp}^s, f_c^s increase when the shape of $\mathscr{N}_s(p)$ approaches the ideal form of a sphere, a line and decrease otherwise. f_d^s increases as the curvature increases.

3.3 Estimating Curvature

There are several approaches concerning the approximation of curvature estimates from point clouds [11], [16], [18]. Our curvature estimation process is based on computing curvature values from polynomials fitted MLS to the point neighborhoods $\mathscr{N}_r(p)$. We use the root mean squared (RMS) curvature

$$\kappa = \sqrt{\frac{1}{2}(\kappa_1^2 + \kappa_2^2)}, \tag{9}$$

where κ_1 and κ_1 are the two principal curvature values, to approximate the "feature strength". There are several other curvature measures like mean, Gaussian or maximal curvature we have considered and tested. For our purposes the RMS curvature was most appropriate and produced best results.

3.4 Post-processing

We are interested in strengthening feature values corresponding to selected features and smoothing out remaining feature values. A Gaussian like filter applied on the local point neighborhood $\mathscr{N}_r(p)$ ensures that noise caused by small artifacts is eliminated.

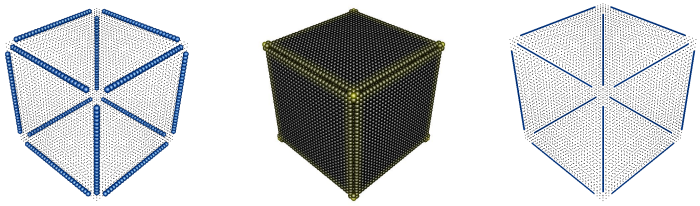
4 Generating the Feature Graph

The structural features are detected and extracted following the idea of [8], [29] and [25] of identifying initial seed nodes and performing an edge-growing approach to connect related nodes. The seed nodes for a graph are represented by all points having a minimal type probability. It remains difficult to define connections between seed nodes such that the resulting graph reflects the underlying feature-lines. Most methods start by connecting seed nodes by proximity. However, this can cause a variety of branches inducing perturbations in further propagation. Therefore, controlling the propagation direction plays an important role.

Several extraction methods determine the derivatives of extracted feature values, like curvature, to identify and follow the direction of its minimal descend. Since these values vary considerably due to noise this differential approach would lead to poor results. Instead, we propose defining the propagation direction of a node by applying PCA to a set of surrounding nodes. This approach is more stable assuming that the local environment of the actual node is of sufficient size. In contrast, critical points are not subject to this graph extraction procedure since they represent discontinuities. These points are used to establish critical nodes to recover junctions of feature lines. The final step consists of graph simplification aiming at a simpler visual representation. We summarize the basic principles of our method.

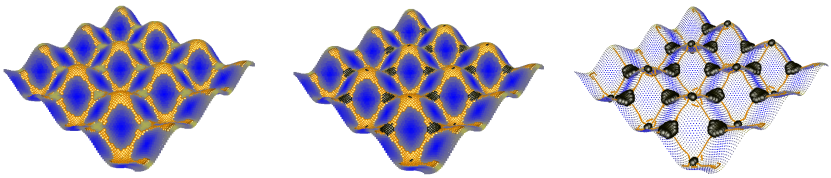
4.1 Selecting Seed Nodes

The seed nodes defining the basis of the individual graphs are identified by using the probability values computed according to the equations in sec. 3.2. A graph $\mathscr{G} = (V, E)$



(a) (b) (c)

Fig. 1. Principle of graph extraction for a simple test cube data set (3000 points): (a) Initial seed nodes (blue), (b) corresponding critical nodes (black), (c) extracted edges after propagation (color plate C. 57, page 275).



(a) (b) (c)

Fig. 2. Principle of graph extraction for surface-related features using a curved sheet data set (10000 points): (a) Initial seed nodes (orange) of the underlying surface (blue), (b) seed nodes with corresponding critical nodes (black), (c) critical nodes together with the extracted edges after propagation (color plate C. 58, page 275).

formally consists of a set of nodes V and set of edges $E \subseteq V \times V$ connecting the nodes of V. The initial graph for representing curve-related features is defined by $\mathscr{G}_c^0 = (V_c, \emptyset)$ with

$$V_c = \{p \in P | \mathscr{L}_c(p) = max(\mathscr{L}_{cp}(p), \mathscr{L}_c(p), \mathscr{L}_d(p))\}. \tag{10}$$

Figure 1(a) shows an example for identified seed nodes for a simple test data set. The graph for representing surface-related features is set up differently. The user specifies curvature-related thresholds σ_{min} and σ_{max} restricting the strength of the features to be extracted, balancing critical and feature-specific nodes among the points of $\mathscr{P}_d$. For example, σ_{min} separate features like creases and non-creases appearing on a surface. The parameter σ_{max} controls the transition between creases and corners. The initial graph representing surface-related features is $\mathscr{G}_d^0 = (V_d, \emptyset)$ with

$$V_d = \{p \in P | \mathscr{L}_d(p) = max(\mathscr{L}_{cp}(p), \mathscr{L}_c(p), \mathscr{L}_d(p)) \wedge \sigma_{min} < I_d(p) \leq \sigma_{max}\}. \tag{11}$$

Fig. 2(a) shows an example concerning surface-related features. The critical nodes corresponding to $\mathscr{G}_c^0$ and $\mathscr{G}_d^0$ are defined as

$$CV_c = \{p \in P | \mathscr{L}_{cp}(p) = max(\mathscr{L}_{cp}(p), \mathscr{L}_c(p), \mathscr{L}_d(p))\} \tag{12}$$

$$CV_d = CV_c \cup \{p \in V_d | \sigma_{max} < I_d(p)\} \tag{13}$$

with $\mathscr{L}_{cp}(p), \mathscr{L}_c(p), \mathscr{L}_d(p)$ determined according (4) Figs. 1(b) and 2(b) show an example of extracted critical nodes for the cube data set and the sheet test set. The nodes of the graph $\mathscr{G}_c^0$ and $\mathscr{G}_d^0$ are not yet connected. A subsequent edge propagation step, discussed in the next section, produces a first approximation of the underlying features.

4.2 Edge Propagation

Edge propagation is performed sequentially starting from the nodes having the highest feature values. Propagation is the process of connecting two nodes of the graph $\mathscr{G} = (V,E)$ by an edge. Principally, each seed node $v \in V$ is allowed to propagate into two directions d and $-d$. We determine d by applying PCA to the local node neighborhood

$$\mathscr{M}_r(v) = \{w \in V \cup CV \,|\, \|v-w\| \leq r\}$$

with r being the user-specified neighborhood size. The vector d is the eigenvector with the largest eigenvalue of the covariance matrix of $\mathscr{M}_r(v)$. Provided that no other node u is connected to v with $d \cdot (\frac{u-v}{|u-v|}) > 0$, v is connected to $w \in \mathscr{M}_r(v)$ whereas

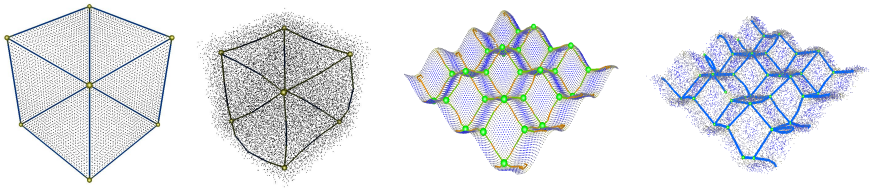
$$w = \text{argmin}_{w \in \mathscr{M}_{r,d}(v)} \left(\|proj_d(w-v) \cdot d + v - w\| / \|w-v\| \right).$$

The nodes $w \in \mathscr{M}_{r,d}(v) \subseteq \mathscr{M}_r(v)$ satisfy the inequality $\left(\frac{w-v}{\|w-v\|}\right) \cdot d > 0$. Here, v and w denote the coordinates of v and w in 3D space. To prevent unwanted edges all disconnected nodes $w \in \mathscr{M}_r(v)$ lying closer to v than the most distant connected neighbor are forbidden to propagate. Moreover, connections establishing acute angles with d are forbidden. Examples of resulting graphs are shown on Figs. 1(c) and 2(c). Rather than using a minimal spanning graph as proposed by Pauly et al. [29], this approach provides greater consistency between edges and feature directions.

4.3 Connecting Critical Nodes

We have to connect the critical nodes to the already established edges of $\mathscr{G}$. Assume CV to be the set of critical nodes defined by (12) and (13). To recover junctions the nodes of CV have to be connected or merged with nodes of $\mathscr{G}$. We start clustering the nodes of CV by grouping neighbored critical nodes. The resulting clusters are defined by $\mathscr{D} = \{\ldots, D_i, \cdots\}$ with cluster $D_i = \{w \in CV |\ v \in D_i \wedge \|v-w\| < r\}$ and $D_i \cap D_j = \emptyset$ for $i \neq j$. The initial cluster set is given by $\mathscr{D}^0 = \{\cdots, \{v\}, \{w\}, \cdots\}$ with $v, w \in V$ and $v \neq w$. We proceed by determining the nodes $d \in D_i$ of greatest feature values. These nodes represent the centroids of $D_i \in \mathscr{D}$. Depending on the user-specified radius r and the shape of D_i a cluster can have several centroidal nodes $d \in D_i$.

Next, we connect all nodes $v \in V$ to surrounding critical nodes $w \in CV$ with $\|v-w\| < r$. This step is performed in a way forcing the graph not to develop unwanted branches. In a subsequent step all nodes $w \in D_i$ connected to a node $v \in V$ are merged with the centroidal node $d \in D_i$. Should there exist more than one centroidal node in a cluster D_i we force w to connect to the centroidal node closest to w. Finally, we delete all unconnected critical nodes producing the final graph representation. Fig. 2(c) shows an example with the critical node cluster shown in black. The remaining connected critical/centroidal nodes are shown in Fig. 3(c).



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

Fig. 3. Results of graph extraction for the cube data set (a) without and (b) with 5% noise. Images (c) and (d) show the resulting graphs for the curved sheet data set without and with 5% noise (color plate C. 59, page 276).

4.4 Graph Simplification

We note that $\mathscr{G}_c$ and $\mathscr{G}_d$ possess certain unwanted structures like short branches, loops etc. To get rid of these artifacts we have developed the following framework.

Graph Simplification: In our context, simplification refers to the process of collapsing nodes that are close to each other in order to reduce complexity of an extracted graph. We collapse nodes v and w of a graph $\mathscr{G}$ if the following criteria are fulfilled:

a) $\|v-w\| \leq r$
b) $(v,w) \in E$ and $\nu(v) > 1$ and $\nu(w) > 1$

Here, $\nu(v)$ defines the number of edges incident to v. Since $\mathscr{G}_c$ and $\mathscr{G}_d$ are intended to reflect structural features we force the position of the resulting graph node to agree with the position of the node possessing the highest feature value. In case of participation of critical nodes the rules are adapted to preserve critical nodes. The entire collapse procedure is performed sequentially in accordance with ascending distance between the node pairs.

Additional Pruning: The remaining graph still exhibits features irrelevant or unwanted for visual exploration. To reduce visual clutter and direct the user to the regions of interest we prune small branches by cutting off all edges $e = (v,w)$ of $v,w \in \mathscr{G}$ with $\|v-w\| < r$ and valence $\nu(v) = 1 \wedge \nu(w) \neq 2$ which are not of immediate interest.

Smoothing: Finally, we enhance the visual appearance of the final graph set $\mathscr{G}$ by applying common graph smoothing.

5 Results

Our method provides a highly effective means for exploring the underlying structural characteristics in environmental LiDaR point data sets. The graph-based approach represents user-specified features of two different types in an efficient and reliable way, driven by a few pre-defined input values. Most important are the radius r of the feature size and the values σ_{min} and σ_{max} confining the type of the considered surface-related

features. The radius r has to be chosen in a way that noise does not interfere with the classification process but also that the extraction of structural information is maintained. Generally, σ_{min} and σ_{max} are less crucial to determine. Once the radius is found the remaining feature extraction can be performed rather fast in an interactive way. Our method distinguishes features of three different types and allows classification of points in terms of probability. Moreover, we are able to identify junctions of feature lines not only by means of graph processing but using implicit information inherited from the initial point cloud. Structural discontinuities are represented in the final graph by critical nodes.

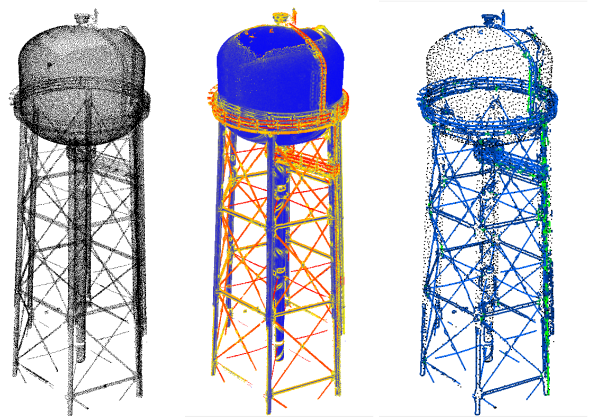
Our multi-scale approach also reduces the influence of noise. It produces reliable results up to a noise level of 5-8%. Fig. 3 shows the resulting graph related to curve-like structures without and with 5% noise relative to the diagonal of the smallest bounding box containing the given point cloud. The examples show that all resulting graphs of the test data set reflect the correct underlying topology. Since determining likelihood and feature values is computationally expensive we de-couple this step from the graph extraction process and perform it in a pre-processing step. This allows the user to extract feature graphs in an nearly interactive manner.

We have applied our method to several other environmental data sets. The data was collected using a LiDaR scanner mounted on a tripod, a method that enables fully three-dimensional scans of engineered structures. For example, Fig. 4(a) through Fig. 4(c) show the results of processing a more complex structure. In Fig. 4(a) we see the original point cloud of a water tower consisting of approximately 1.7 million points. Fig. 4(b) shows the computed feature values regarding curve-related features for each point. The extracted feature graph is presented in Fig. 4(c). More detailed parts of the graph are presented in Fig. 5. Fig. 5(a) shows a relative complex under-sampled object which is part of the tower. In this example the underlying feature lines and junctions were almost completely recovered. The second object shown in Fig. 5(b) is a crosspiece of the main pole of the lower part of the tower having mounted stiffeners. The extracted feature graph exhibits critical nodes colored in green representing discontinuities between surface- and curve-like structures.

The environmental point cloud shown in Fig. 6 is a part of the San Andreas Fault in California, USA. Since the scan was performed airborne the resulting point cloud contains a relatively higher level of noise. The edges of the extracted surface-related graph depicted in red represent the underlying ridges of the landscape. The critical nodes depicted as green points represent either underlying discontinuities or features having curvature values which exceed the user-specified limit σ_{max}. In this example the parameter values were chosen to capture ridges associated with relatively high feature/curvature values. Those parts of the landscape with high curvature values (tapered ridges or acute hills) are represented by critical nodes. For this example we chose $\sigma_{min} = 0.2, \sigma_{max} = 1.5$ and $r \approx 0.0046\%$ of the length of the data set diagonal of the smallest bounding box containing P.

In order to allow the user to perform the feature extraction interactively we have divided the whole approach into two steps. We provide a computational expensive preprocessing step concerning the point classification and the actual graph extraction step. The time complexity for the point classification is estimated to be $O(k^2 \cdot n \cdot log(n))$

(with n number of input points, k the average neighborhood size). The complexity of the graph extraction is assumed to be $O(m \cdot log(m) \cdot n)$ (with m being the number of identified seed nodes). This has been validated by our observations. For example, the point classification performed at the finest resolution for the water tower point cloud (see Fig. 4(a)) having about 1.7 million points has last 3h29m. The subsequent graph extraction was performed in approximately 116 secs. Computing the point classification for the geographic data set in Fig. 6 was completed after 72 minutes. The final graph was constructed after 29 secs. These results confirm that our approach holds great promise for feature detection and extraction for LiDaR data processing, analysis and understanding.

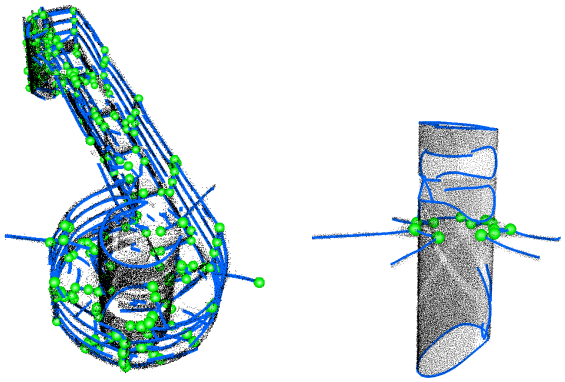


(a) (b) (c)

Fig. 4. (a) original point cloud of a water tower consisting of 1.7 million points; (b) color coded feature values of each point regarding curve-related features(blue represents low, red high feature values); (c) corresponding feature graph (blue lines) for $\varepsilon = 0.5$ and $r \approx 0.038\%$ of the bounding box diagonal (color plate C. 60, page 276).

6 Conclusions

We have introduced a novel approach for extracting and visualizing features from LiDaR data sets. Our examples demonstrate the benefits of this method for extracting feature graphs from point clouds representing surface components like ridges and creases as well as curve-like components. As our method was designed as an interactive, user-controlled method offering also a high degree of automation, one specific strength is the ability for a user to influence the analysis process and obtain instant visual feedback. This work is fundamental for future research, for example to detect and measure deformation of environmental structures such as buildings and bridges exposed to terrestrial influences. Future work will be directed at feature-based segmentation of terrestrial data classifying objects of different type, such as trees, buildings, streets, etc. One next goal is to be able to extract topological information from landscapes by extracting and com-



(a) (b)

Fig. 5. Closeup of selected parts of the extracted graph of Fig. 5(c) (color plate C. 61, page 277).

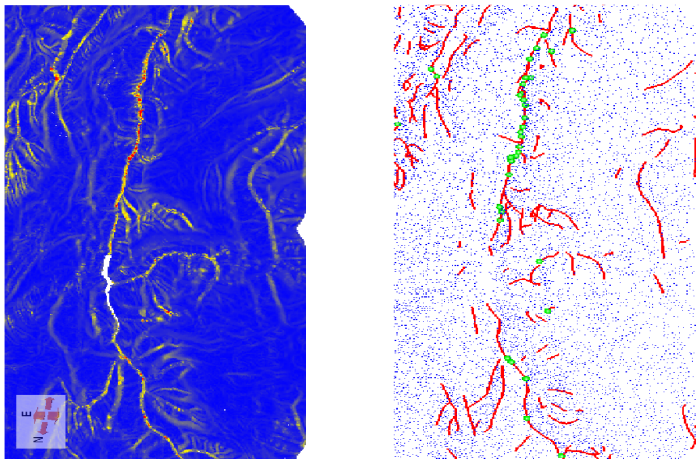
paring ridge lines from multiple scans taken at different times to identify deformations by a comparative visualization method.

Acknowledgements

This work was supported by the DFG's International Research Training Group (IRTG) 1131 at the University of Kaiserslautern and the Center for Mathematical and Computational Modeling $(CM)^2$ It was also supported in part by the W.M. Keck Foundation through the UC Davis Center for Active Visualization in the Earth Sciences (KeckCAVES), Department of Geology. We thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV) at UC Davis. We thank Dr. Gerald Bawden of the US Geological Survey for sharing tripod LiDaR data.

References

1. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.
2. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Conference on Visualization*, pages 21–28. IEEE Computer Society, 2001.
3. A. Alharthy and J. Bethel. Detailed building reconstruction from airborne laser data using a moving surface method. In *20th Congress of International Society for Photogrammetry and Remote Sensing*, pages 213–218, 2004.
4. N. Amenta, S. Choi, and R. K. Kolluri. The power crust. In *6th ACM symposium on Solid Modeling and Applications*, pages 249–266. ACM Press, 2001.
5. A. Belyaev and E. Anoshkina. Detection of surface creases in range data. In *Mathematics of Surfaces*, pages 50–61, 2005.
6. P. Benkö and T. Várady. Direct segmentation of smooth, multiple point regions. In *Geometric Modeling and Processing - Theory and Applications*, page 169. IEEE Computer Society, 2002.
7. P. Benkö and T. Várady. Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design*, 36(6):511–523, 2004.



(a) (b)

Fig. 6. Scan of San Andreas fault in California, USA, on a length of 6 kilometers. (a) presents the corresponding feature/curvature values (blue means low, red equals high curvature value). (b) shows the corresponding surface-related feature graph with feature lines depicted in red, and critical nodes as green points (color plate C. 62, page 277).

8. J. I. Daniels, L. K. Ha, T. Ochotta, and C. T. Silva. Robust smooth feature extraction from point clouds. In *IEEE International Conference on Shape Modeling and Applications, 2007. SMI '07.*, pages 123–136, June 2007.
9. T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. In *20th Annual Symposium on Computational Geometry*, pages 330–339. ACM Press, 2004.
10. T. K. Dey and J. Sun. An adaptive mls surface for reconstruction with guarantees. In *3rd Eurographics Symposium on Geometry Processing*, pages 43–52, 2005.
11. I. Douros and B. F. Buxton. Three-dimensional surface curvature estimation using quadric surface patches. In *Scanning*, 2002.
12. B. Hamann. Modeling contours of trivariate data, Mathematical Modeling and Numerical Analysis. In *Modelisation Mathematique et Analysis Numerique*, vol. 26(1), Gauthier-Villars, France, pages 51–75, 1992.
13. M. Eck and H. Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *ACM Siggraph*, pages 325–334, 1996.
14. S. Fleishman, D. Cohen-Or, and C. Silva. Robust moving least-squares fitting with sharp features. In *ACM Siggraph*, pages 544–552, 2005.
15. S. Gumhold, X. Wang, and R. MacLeod. Feature extraction from point clouds. In *10th International Meshing Roundtable*, pages 293–305, 2001.
16. B. Hamann. Curvature approximation for triangulated surfaces. *Geometric Modelling, Computing Suppl. 8*, pages 139–153, 1993.
17. K. Hildebrandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, pages 85–90. Eurographics Association, 2005.
18. P. Krsek, G. Lukács, and R. Martin. Algorithms for computing curvatures from range data. In *The Mathematics of Surface VIII*, pages 1–16. Information Geometers. Winchester, UK, 1998.
19. C. Lange and K. Polthier. Anisotropic smoothing of point sets. *Computer Aided Geometric Design*, 22:680–692, 2005.

20. D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.
21. B. Mederos, L. Velho, and L. H. de Figueiredo. Moving least squares multiresolution surface approximation. In *Sibgraphi*, 2003.
22. L. V. B. Mederos and L. H. de Figueiredo. Robust smoothing of noisy point clouds. In *SIAM Conference on Geometric Design and Computing*, 2003.
23. A. Nealen. An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. In *Technical Report, TU Darmstadt*, 2004.
24. M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *ACM Siggraph*, pages 641–650, 2003.
25. G. Stylianou and G. Farin. Crest lines for surface segmentation and flattening. *IEEE Transaction On Visualization and Computer Graphics*, 10(5):536–544, Sept. 2004.
26. T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. Gross. Post-processing of scanned 3d surface data. In *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, pages 85–94, 2004.
27. S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Fast and robust detection of crest lines on meshes. In *ACM Symposium on Solid and physical modeling*, pages 227–232. ACM Press, 2005.
28. M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. In *ACM Siggraph*, pages 322–329, 2002.
29. M. Pauly, R. Keiser, and M. Gross. Multi-scale Feature Extraction on Point-sampled Surfaces. In *Proceedings of Eurographics 2003*, pages 281–289, 2003.

Topological Flow Structures in a Mathematical Model for Rotation-Mediated Cell Aggregation

Alexander Wiebel[1], Raymond Chan[2], Christina Wolf[3], Andrea Robitzki[3], Angela Stevens[4], and Gerik Scheuermann[5]

[1] Max-Planck-Institut für Kognitions- und Neurowissenschaften, Stephanstraße 1A, 04103 Leipzig, Germany, wiebel@cbs.mpg.de

[2] Max-Planck-Institut für Mathematik in den Naturwissenschaften, Inselstraße 22, 04103 Leipzig, Germany, rchan@mis.mpg.de,

[3] Universität Leipzig, Center for Biotechnology and Biomedicine (BBZ), Abteilung Molecular biological-biochemical Processing Technology, Deutscher Platz 5, 04103 Leipzig, Germany, andrea.robitzki@bbz.uni-leipzig.de, christina.wolf@bbz.uni-leipzig.de,

[4] Universität Heidelberg, Angewandte Mathematik und Bioquant, Im Neuenheimer Feld 267, 69120 Heidelberg, Germany angela.stevens@uni-hd.de

[5] Universität Leipzig, Abteilung Bild- und Signalverarbeitung, PF 100920, 04009 Leipzig, Germany, scheuer@informatik.uni-leipzig.de

Abstract. In this paper we apply vector field topology methods to a mathematical model for the fluid dynamics of reaggregation processes in tissue engineering. The experimental background are dispersed embryonic retinal cells, which reaggregate in a rotation culture on a gyratory shaker, according to defined rotation and culture conditions. Under optimal conditions, finally complex 3D spheres result. In order to optimize high throughput drug testing systems of biological cell and tissue models, a major aim is to understand the role which the fluid dynamics plays in this process. To allow for a mathematical analysis, an experimental model system was set up, using micro-beads instead of spheres within the culture dish. The qualitative behavior of this artificial model was monitored in time by using a camera. For this experimental setup a mathematical model for the bead-fluid dynamics was derived, analyzed and simulated. The simulations showed that the beads form distinctive clusters in a layer close to the bottom of the petri dish. To analyze these patterns further, we perform a topological analysis of the velocity field within this layer of the fluid. We find that traditional two-dimensional visualization techniques like path lines, streak lines and current time-dependent topology approaches are not able to answer the posed questions, for example they do not allow to find the location of clusters. We discuss the problems of these techniques and develop a new approach that measures the density of advected particles in the flow to find the moving point of particle aggregation. Using the density field the path of the moving aggregation point is extracted.

1 Introduction

The topology of two-dimensional time-dependent vector fields has been an active field of research in recent years [7,17,18]. In this paper we investigate the features of a mathematical model for the fluid flow of a mixture of beads and growth medium in a layer close to the bottom of a petri dish, which rotates on a gyratory shaker. A moving zero in the instantaneous vector field is the most striking feature of the flow. This suggests the application of topological methods for the desired analysis. To provide the basis for the discussion of the vector field we will first describe the experiment, the mathematical model and its simulation, which leads to the vector field.

2 The Experiment, the Mathematical Model and its Simulation

The biological experiment, which builds the background for the mathematical model, its simulation, visualization and visualization techniques, investigates rotational tissue cultures, which are relevant for high throughput drug testing systems in regenerative medicine. A petri dish, which contains growth medium and dispersed embryonic cells, is located on a gyratory shaker. The specificities of the rotation affect the fluid flow in the petri dish and thus the motion of the cells. Without any movement of the petri dish, the cells generally form a mono-layer at the bottom and grow in a disorganized manner. However, under a specific rotation of the petri dish, the cells finally form several 3D spheroids. Details about these methods and further results can be found in [8–10].

To understand the role the fluid dynamics play in this reaggregation and structure forming process, an experimental model system was set up. Microscopic beads were put into the culture dish and rotated under the same conditions as the cell systems. This system is assumed to serve well as a model system for the cell-based fluid dynamics under consideration. In the experiment clustering of the beads was observed for a rotation speed of 70RPM but not for a rotation speed of 60RPM. Further interesting patterns and phase transitions occurred. To confirm the hypothesis that *mechanical* aggregation plays a key role in the initial clustering of the beads, a mathematical model for the fluid dynamics was derived and numerically analyzed. The basis of the mathematical model are the incompressible Navier-Stokes Equations, which are solved in a domain representing the petri dish. Fictitious body forces, acting on the fluid, are added. These result from the rotation of the petri dish. A dimensional analysis was performed and by regular perturbation techniques the model was reduced to a shallow water type of problem. The main assumption is, that the Reynolds number in horizontal direction is much larger than the Reynolds number in vertical direction. For the numerical discretization a finite volume method was employed, which takes into account that the flow is mainly laminar. The qualitative behavior of the mathematical model compares well to the aggregation behavior of the beads observed in the experiment. Clusters of particles are found to rotate around the center of the petri dish. Further details on the mathematical model, its simulation and first visualizations are given in [3]. The simulations of this model provide the data, which are analyzed in the following.

To justify our use of two-dimensional visualization techniques it should be noted that the beads (in the rest of the paper often called *the particles*) are expected to stay

in a layer close to the bottom of the petri dish. The particles are only expected to leave this layer at singular points in the flow.

3 Related Work in Vector Field Topology Visualization

As mentioned, topological methods seem to be a good choice for the visualization of the presented application. Of particular relevance for the present work are techniques that permit to track the continuous evolution of the topology as it evolves over time. Improving on a scheme introduced by Helman and Hesselink [7], which graphically reconnects the topological skeletons extracted in successive time steps, Tricoche *et al.* [18] proposed a scheme that computes the continuous path followed by two-dimensional *singularities* (where the flow velocity vanishes) across the space-time domain. Their approach explicitly characterizes bifurcations, which correspond to critical changes affecting the structure of the topological skeleton. An alternative method that extracts the topological evolution by means of numerical integration over the space-time continuum was introduced by Theisel *et al.* [16]. Extensions to three-dimensional transient flows have been presented for both methods [5, 15]. Just recently Wiebel *et al.* [19] introduced a technique allowing to track singularities on curved surfaces by using parameterizations in combination with the existing two-dimensional techniques.

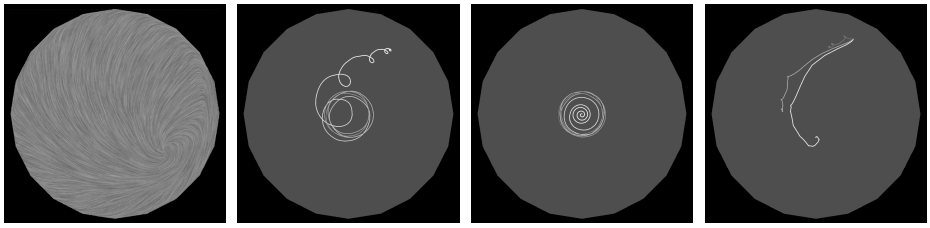
A first approach to time-dependent topology not using streamlines, called path line oriented topology, was undertaken by Theisel *et al.* [17]. They distinguish sectors of attracting, repelling and saddle-like behavior of the path lines. This is different from the usual concept of topology, which is to observe how trajectories behave under an integration *until infinity* while their method only considers local properties of the path lines. They call this approach topological because it also aims at segmenting the domain into areas of different flow behavior. To be able to apply an asymptotic analysis for path lines Shi *et al.* [14] restrict themselves to periodic fields and present a path line oriented topology for periodic 2D time-dependent vector fields. Unfortunately, this approach is not applicable in our case as the flow considered in this paper is not periodic.

4 Application of Common Visualization Techniques

In this section we describe the application of a number of standard visualization techniques to the petri dish flow field. We describe the different viewpoints taken by these techniques and show their deficiency to illustrate certain features of the flow. This will promote the need for the new technique developed in Section 5.

4.1 Streamlines

Streamlines and their dense counterpart the line integral convolution [2] (LIC) are the most frequently used visualization techniques for flow fields. As they only illustrate the momentary direction of a flow, animations are often used for time-dependent fields. For the current application such an animation only shows the overall rotation behavior and the existence of an attracting singularity in the flow (see Figure 1(a)). The information



(a) LIC of instantaneous vector field for $t = 37$. Note the sink in the right part.

(b) Path line from outer area.

(c) Path line from center.

(d) Streak lines started at $t = 2$ shown for $t \in \{25, 37, 79\}$.

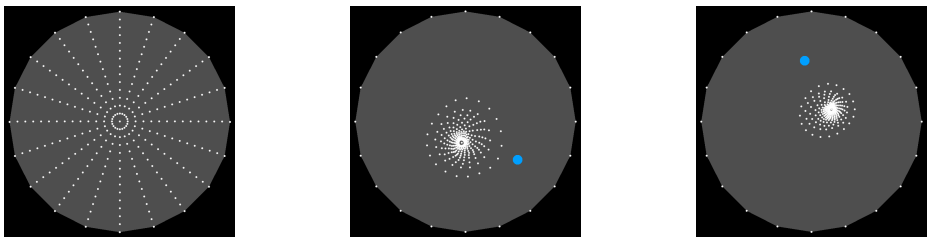
Fig. 1. Application of different standard visualization techniques for vector fields.

about the long-term behavior of particles given by this visualization is very small. A more detailed discussion of the attracting singularity will be given in the section about streamline oriented topology (Section 4.4).

4.2 Path Lines

As path lines reflect the path of a particles they should be better suited for finding the path of the moving agglomeration. Indeed, the path lines shown in Figures 1(b) and 1(c) (coming from the border respectively the center of the domain) show that the particles approach a kind of common cyclic structure. A naive interpretation of this cyclic structure could be that the particles are distributed around the circle and form a kind of ring. It does not become clear that the particles tend to agglomerate around one position and that this agglomeration moves on the displayed circle (see Figures 2(b) and 2(c)).

4.3 Streak Lines



(a) (b) (c)

Fig. 2. Particles traced in the rotating flow show that the point attracting the particles is not the singularity moving through the flow. (a) Particles (white) at $t = 1$. (b) Particles after advection until $t = 37$ with attracting singularity (blue) at $t = 37$. Compare the position of the singularity to LIC in Figure 1(a). (c) Particles after advection until $t = 40$ with attracting singularity at $t = 40$.

Well known from physical flow experiments, streak lines seem to be a good standard choice to investigate the two-dimensional flow field. Most images produced by streak lines seem to yield a good representation of the underlying flow. Good examples can be found in a paper by Sheard *et al.* [13] and many other papers in the same journal as well as in a book by Batchelor [1]. However, it is also known that care has to be taken with the interpretation of streak lines [6]. Representative streak lines for our example are shown in Figure 1(d). The image is only able to illustrate the fact that the particles come closer to the center of the dataset with longer advection time. An animation over a certain time interval shows the general rotation but does not reveal any salient features either.

4.4 Streamline Oriented Topology

The best candidates for showing the aggregation of the particles seem to be topological methods as they can track the path of attracting singularities (also known as *sinks*) through time. Naively, one would expect that the particles at a certain point of time tend to agglomerate at the momentary position of the sinks. In our example we have exactly one such sink and thus would expect the particles to converge to this point.

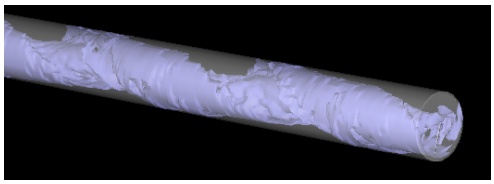
To inspect these presumptions we traced the particles shown in Figure 2(a) a certain amount of time t and computed the singularity of the instantaneous vector field at time t. Figures 2(b) and 2(c) show the resulting images. The images clearly contradict the presumptions. The particle agglomeration is not located around the singularity, it rather lags approximately one third of a rotation period behind the zero of the vector field. A possible explanation for the difference between the point of aggregation and the singularity might be that the singularity lies in a large area with nearly zero vectors. Thus the concentration process can not (or only very slowly) appear there. It is probably much stronger in areas with vectors of large magnitude. Additionally, even a significantly moving and strongly attracting singularity would not be able to drag the agglomeration along its path because vectors on opposite sides of the singularity point in opposite directions. The particles thus are moved in one direction before the singularity passes (close to) their position and in the opposite direction thereafter.

As our example shows, there are probably only few cases for which a singularity in time-dependent flow has an interesting meaning. One of these few cases is a singularity on a surface that indicates separation of the flow from the surface (see e.g. [19]).

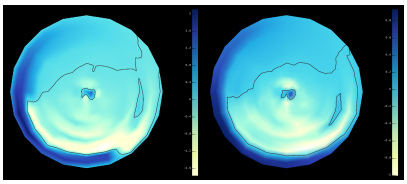
4.5 Path Line Oriented Topology

As discussed in Section 4.2, path lines are more effective for illustrating the time-dependent nature of the flow than streamlines. Thus it is worth looking at the visualization yielded by the so-called path line oriented topology as introduced by Theisel *et al.* [17]. Although, this approach does not really compute path lines and analyze their behavior it still classifies the local time-dependent behavior as saddle-like, attracting (converging) or repelling (diverging).

The application of path line oriented topology to our data reveals no areas of saddle-like behavior. The domain is only divided into attracting and repelling parts. Figure 3(a) shows the attracting parts enclosed by an isosurface. It is apparent that not only specific



(a) Path line oriented topology in complete time-dependent field. Third dimension used to stack time steps ($60 < t < 80$ visible). Isosurface encloses attracting behavior.



(b) Path line oriented topology (left) compared to divergence (right) for $t = 70$. Colormap and zero isolines divide the field into areas with converging (negative values) and diverging behavior (positive values)

Fig. 3. Path line oriented topology (color plate C. 63, page 278).

points are marked as attracting but large areas. Thus, the approach does not identify the features we are searching for, i.e. points of particle attraction. It only illustrates that the attracting and repelling parts revolve around the center of the dish over time.

Figure 3(b) shows a comparison of one of the characteristic fields of the path line oriented topology (here sum of eigenvalues) and the divergence of the instantaneous vector field. Both images illustrate the behavior at $t = 70$. The comparison makes a strong statement about the meaning of the path line oriented topology for our data. It seems that the areas marked as attracting or repelling closely correspond to areas of negative or positive divergence. This is interesting because the computation of path line oriented topology is much more time consuming than the computation of divergence.

4.6 FTLE

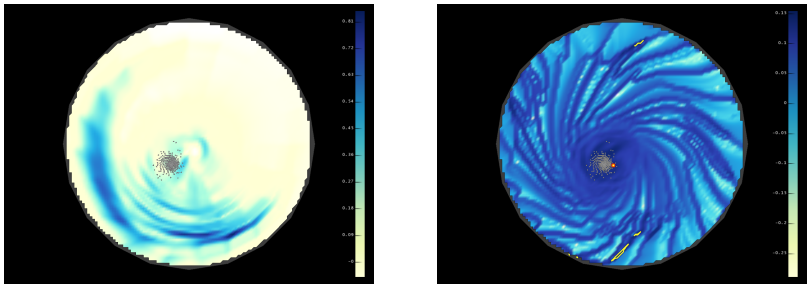


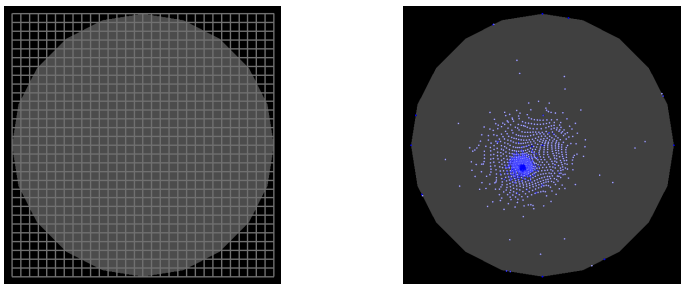
Fig. 4. FTLE$^-$ with particle cluster for $t = 70$. Integration for 1 (left) and 20 time units (right). Very high magnitude of FTLE$^-$ marked yellow. Note the small (enlarged) yellow point fairly close to the particles (color plage C. 64, page 278).

As we are searching for converging particles, one might suggest to use FTLE (Finite-Time Lyapunov Exponent) fields for visualization (see e.g. [4, 11]). Our experiments,

however, showed that while FTLE is able to characterize the time-dependent convergence better than path line oriented topology does, it does not identify the point of attraction either. Figure 4 shows a color mapping of FTLE^{-} (i.e. FTLE using backward integration) for $t = 70$ together with a particle cluster at the same instant. The left image results from a very short integration and does not help us, as the FTLE is dominated by interpolation artifacts resulting from the coarse grid. This influence vanishes for longer integration times (right image). But even in this image the center of the particles can not be determined correctly as strong FTLE^{-} can be found in a large area around the particles. Taking the maximum does not help as there are several very large values (marked yellow), none of which represents the center correctly. The relatively "noisy" FTLE field (right image) results from the fact that many particles leave the domain during long-time backward integration.

5 Detection of Point of Attraction

In the previous section we have shown that popular existing visualization techniques for unsteady two-dimensional flow fields, while being able to give hints as to where particles converge, are not capable of finding the point of aggregation of the particles. In the rest of the current section we will describe a new approach specially designed to find and track the point of particle aggregation.



(a) (b)

Fig. 5. Particle advection for density computation. (a) Regular grid for starting particles. (b) Particles started at earlier time steps lie in convex hull of particles from later times (except points started and remaining exactly on the boundary). Points color coded by starting time.

5.1 Idea

The basis for the presented technique is the following observation: Particles started at earlier times, in general, can be found in the convex hull of particles started at later time steps (see Figure 5(b)). This means that the point of the largest particle density is the point that is attracting particles. So the main procedure of the proposed technique is the following: We trace particles from a grid at a number of time steps before the

observation time. We then compute the density of the particles and extract maxima of this density. The maxima show us the positions of the points of particle attraction. As these points move over time we repeat the procedure for several observation times. Connecting the points of the different observation times yields the path of a point of attraction.

5.2 Seeding of Particles

The first step of the technique is seeding the particles for the advection. Care has to be taken because the initial distribution influences the density of the particles after advection. As can be seen from the initial particle positions in Figure 2(a), the grid of the simulation is a structured radial grid. Particles seeded on this grid have a non-uniform density, i.e. a higher density near the center. Figure 2(b) shows that the non-uniformity of the initial distribution is still visible after particle advection. This can cause the existence of density maxima (resulting from initially high density) at points where advection has not increased density. This, again, can make finding the correct point of attraction impossible.

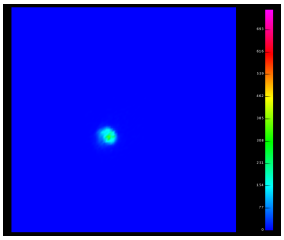
Consequently, we need a uniform density at the beginning. A simple distribution fulfilling this constraint can be obtained by seeding the particles at the points of regular grid as shown in Figure 5(a). For starting particles at different time steps we choose the seeding grids to be equidistant in time.

5.3 Density Computation

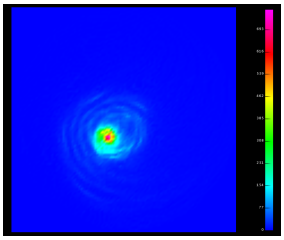
We store the density on a second regular grid. It may have a different resolution than the one used for seeding the particles. For every point of the density grid we simply count the particles that are closer to this point than to any other. This is equivalent to counting the particles in all cells of the dual of the density grid. Since the grid is regular, each cell of the dual grid is defined by two intervals on the two coordinate axes. Knowing the resolution and the bounding box of the grid, it is easy to compute the intervals and thus the grid position each particle position belongs to. We increment the particle count of the density grid position we have found a particle to be closest to and repeat this for all particles and starting times to obtain the desired density field. Examples of density fields for our data set are shown in Figure 6.

5.4 Extraction of Density Maxima

The density field computed in the previous step is given on the points of the density grid. Usually one introduces a certain interpolation scheme to have continuous data over the whole domain. If the underlying grid consists of simplices, i.e. triangles in the two-dimensional case, one often uses a linear interpolation. Inside quadrangular cells bilinear interpolation is commonly used. For both interpolation schemes maxima and minima can only exist at vertices. Thus, their detection is straight forward. Using the connectivity induced by the grid we simply compare the value at each vertex with all directly adjacent vertices. If all adjacent vertices have smaller values than the considered

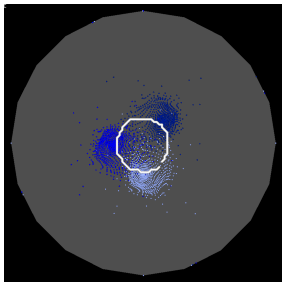


(a) Density of particles traced from one grid at $t = 50$.

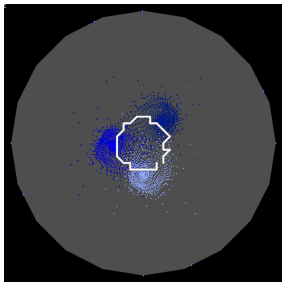


(b) Density of particles traced from ten grids at $t = 50, 52, \ldots, 68$.

Fig. 6. Color coded density distribution for time $t = 70$ and resolution of 100×100. The length and with of the square are two times the radius of the petri dish (color plage C. 65, page C. 65).



(a) Attraction path consisting of 50 observations with density field resolution 100×100 and seeding resolution 30×30



(b) Attraction path consisting of 30 observations with density field resolution 40×40 and seeding resolution 10×10.

Fig. 7. Attraction path for $72 < t < 80$ extracted by tracing from $t - 20$. Smoothness of path depends on resolution of density field and number of observations. Different colored particles for different observation times t (traced from $t - 15$) provide context.

vertex a maximum has been found. If only the global maximum is of interest we simply run through all positions and store the position of the largest value. In order to eliminate insignificant maxima and merge very close maxima, a low pass filter can be applied. This was not necessary for the presented data set. The only significant maximum in the petri dish flow at $t = 70$ is clearly visible in Figure 6.

5.5 Tracking of Density Maxima

So far only the positions of the maxima in the different time steps have been determined. These are the positions of the point of attraction at different times. To extract the complete path of the moving point we have to connect the positions of corresponding maxima between the time steps. In our case, where we have only one important maximum, we simply connect the positions by straight lines. Images of the tracked maximum are shown in Figure 7. As the maxima can only appear at vertices of the grid underlying the density field, the smoothness of extracted paths strongly depends on the

resolution of this grid. A second influencing factor is the distance of the density fields in time. This is why the paths in the figure are relatively jaggy. The paths in Figure 7 cover only relatively short time spans to avoid visual clutter by crossing and overlapping.

Multiple Maxima As mentioned, we connect the single maximum in our data from one time step to the next. When aiming at tracking several maxima in one density field over time, more elaborate techniques are needed. Several methods for tracking features in scalar fields can be found in the literature, see e.g. [12].

5.6 Performance, Discussion and Acceleration

The computation times for the paths strongly depend on the resolution of the seeding grid, the number of observations and the number of grids used to seed the particles for each observation. In Figure 7 they range from 4 minutes for the path in Figure 7(b) to one hour for the path in Figure 7(a).

Increasing the resolution of the density grid increases the computation time only marginally. This is why we chose the density resolution to be different from the seeding resolution. It allows us to increase the precision of the detected maximum position at nearly no cost. However, it is important to note that the density resolution can not be chosen completely independent from the seeding resolution. If the resolution difference becomes to large, the number of particles becomes insufficient to produce a reasonable density field. The result are many positions with one or two particles closest to them and no position being a significant maximum. This renders the extracted paths meaningless. We found density resolutions below 5 times the seeding resolution to yield good results.

The large number of particle advections is the dominating factor of the computation time. There is a large potential for acceleration. The most obvious acceleration is parallelizing the particle advection. This yields a computation time nearly inverse proportional to the number of tasks processable in parallel.

A second idea, more specific to the presented approach, is to reuse previously traced particles. This idea is applicable if we do not only compute the position for a single time but perform a tracking of the position through a number of time steps. Let $t_0^a < t_1^a < \ldots < t_n^a$ be equidistant times for which particles have been traced until $t^a = t_n^a$. Define $t_0^b < t_1^b < \ldots < t_n^b$ analogously. Let $t^a < t^b$. If we now store all positions of all particles traced for t^a in groups depending on their start time t_i^a we can reuse the particles of all but one group (t_0^a) for computing the positions of the particles for t_i^b. Let the distance between two times be Δt. Then the positions of the particles starting at $t_0^b, \ldots, t_{n-1}^b$ and being observed at t^b can be computed by tracing the particles of the groups $t_1^a, \ldots, t_n^a$ for Δt. Obviously, this saves a large amount of computation time, as nearly all particles[6] observed at t^b need only to be traced for the relatively short time span Δt. Without this acceleration the mean time span traced is $\frac{n\Delta t}{2}$. Thus, reusing the previous particle positions we achieve an acceleration by a factor of approximately $\frac{n}{2}$.

[6] The particles of group t_n^b do not need to be traced. They are already at time t^b.

6 Conclusion

We have discussed the application of several visualization techniques that are commonly used to illustrate two-dimensional time-dependent flow at a biofluid dynamic model. It turned out that all these existing techniques are not able to show the desired information or features of the vector field, i.e. the location of the point where particles tend to aggregate. Hence, a new technique computing the density of particles started at regularly distributed points at regularly spaced time steps is introduced. The density field obtained with this technique is then used to compute the desired location for any time step of interest. Thus, a curve representing the moving particle aggregation center can be defined. It is important to note that this curve is different from the path of the zero in the instantaneous velocity field.

We plan to improve our acceleration techniques to be able to extend our method to unsteady 3D vector fields where the computation time becomes even more crucial. Furthermore, we recommend the usage of our method only if the existence of an interesting point of aggregation is known, as otherwise a large number of false positives might corrupt the visualization. A method for detecting the existence of points of aggregation is needed.

Acknowledgments

We like to thank C. Garth for supplying his DOPRI5 code and W. Hackbusch, S. Luckhaus, and J.J.L. Velázquez for fruitful discussions on the mathematical model and the numerical code. During the course of this work A. Wiebel was supported by DFG grant SCHE 663/3-8 and hired by the BSV group at the University of Leipzig. R. Chan and C. Wolf were supported by the DFG Graduate College InterNeuro (GRK 1097) as well as A. Robitzki and A. Stevens as members and supervisors of this College. Further, A. Robitzki was supported by the DFG via SFB 610 - Z5. A. Stevens' work for this project was done, while she was hired by the Max-Planck-Institute for Mathematics in the Sciences, Leipzig.

References

1. G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 1967. Cambridge Mathematical Library Edition.
2. B. Cabral and L. C. Leedom. Imaging Vector Fields Using Line Integral Convolution. In *SIGGRAPH '93*, pages 263–270. ACM Press, 1993.
3. Raymond Chan. *A Biofluid Dynamic Model for Centrifugal Accelerated Cell Culture Systems*. PhD thesis, Fakultät für Mathematik und Informatik, Universität Leizig, DFG-Graduate College InterNeuro (GRK 1097), 2008. Preprint.
4. C. Garth, F. Gerhard, 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.
5. C. Garth, X. Tricoche, and G. Scheuermann. Tracking of Vector Field Singularities in Unstructured 3D Time-Dependent Datasets. In *IEEE Visualization 2004*, pages 329 – 336. IEEE Computer Society, October 2004.

6. F. R. Hama. Streaklines in a perturbed shear flow. *PoF*, 5(6):644–650, June 1962.
7. J. L. Helman and L. Hesselink. Surface Representations of Two- and Three-Dimensional Fluid Flow Topology. In *IEEE Visualization '90*, pages 6–13. IEEE Computer Society Press, 1990.
8. P.G. Layer, A. Robitzki, A. Rothermel, and E. Willbold. Of layers and spheres: the reaggregate approach in tissue engineering. *Trends Neurosci.*, 25:131–134, 2002.
9. A. Mack and A. Robitzki. The key role of butyrylcholinesterase during neurogenesis and neural disorders: An antisense-5'-butyrylcholinesterase DNA study. *Prog. Neurobiol.*, 60:607–628, 2000.
10. A. Rothermel, T. Biedermann, W. Weigel, R. Kurz, M. Rüffer, P.G. Layer, and A. Robitzki. Artificial design of 3d retina-like tissue from dissociated cells of the mammalian retina by rotation-mediated cell aggregation. *Tissue Eng.*, 11(11-12):1749–1756, 2005.
11. F. Sadlo and R. Peikert. Efficient visualization of lagrangian coherent structures by filtered AMR ridge extraction. *IEEE TVCG*, 13(6):1456–1463, 2007.
12. Ravi Samtaney, Deborah Silver, Norman Zabusky, and Jim Cao. Visualizing Features and Tracking Their Evolution. *IEEE Computer*, 27(7):20 – 27, 1994.
13. G. J. Sheard, T. Leweke, M. C. Thompson, and K. Hourigan. Flow around an impulsively arrested circular cylinder. *Physics of Fluids*, 19(8), 2007.
14. Kuangyu Shi, Holger Theisel, Tino Weinkauf, Helwig Hauser, Hans-Christian Hege, and Hans-Peter Seidel. Path line oriented topology for periodic 2D time-dependent vector fields. In *Proc. EuroVis 2006*, pages 139–146, Lisbon, Portugal, May 2006.
15. H. Theisel, J. Sahner, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Extraction of Parallel Vector Surfaces in 3D Time-Dependent Fields and Application to Vortex Core Line Tracking. In *IEEE Visualization 2005*, pages 631–638, October 2005.
16. H. Theisel and H.-P. Seidel. Feature Flow Fields. In *VISSYM '03: Proc. of the Sym. on Data Visualisation 2003*, pages 141–148. Eurographics Association, 2003.
17. H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Topological methods for 2d time-dependent vector fields based on stream lines and path lines. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):383–394, 2005.
18. Xavier Tricoche, Thomas Wischgoll, Gerik Scheuermann, and Hans Hagen. Topology Tracking for the Visualization of Time-Dependent Two-Dimensional Flows. *Computers & Graphics*, 26(2):249 – 257, 2002.
19. A. Wiebel, X. Tricoche, D. Schneider, H. Jänicke, and G. Scheuermann. Generalized streak lines: Analysis and visualization of boundary induced vortices. *IEEE TVCG*, 13(6):1735–1742, 2007.

A Categorical Approach to Contour, Split and Join Trees with Application to Airway Segmentation

Andrzej Szymczak

Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401-1887, aszymcza@mines.edu

Abstract. Contour, split and join trees can be defined as functors acting on the category of scalar fields, whose morphisms are value-preserving functions. The categorical definition provides a natural way to efficiently compute a variety of topological properties of all contours, sublevel or superlevel components in a scalar field. The result is a labeling of the contour, split or join tree and can be used to find all contours, sublevel or superlevel sets with desired properties.

We describe an algorithm for airway segmentation from Computed Tomography (CT) scans based on this paradigm. It computes all sublevel components in thick slices of the input image that have simple topology and branching structure. The output is a connected component of the union of all such sublevel components. This procedure can be viewed as a local thresholding approach, where the local thresholds are determined based on topological analysis of sublevel sets.

1 Introduction

Category theory has been extensively used in mathematics to relate structures of different types. In particular, it is widely used in algebraic topology as a convenient language allowing one to translate topological problems or properties into, often much simpler, algebraic ones.

In this work, we apply the categorical approach to commonly used combinatorial descriptions of the topology of scalar fields: contour, join and split trees. We show that each of the trees can be seen as a functor from the category of scalar fields (whose morphisms are value-preserving mappings) into itself. The functors introduced in this paper do not change the form of the input, since the domain and the range categories are the same. However, in practice, they highly reduce complexity: they transform any scalar field into one whose domain has a finite tree structure, with the number of vertices related to the topological complexity of the input scalar field, i.e. its number of critical points.

The categorical approach can be used to define mappings between the trees (for example, inclusion-induced mappings introduced in our earlier work [2]). We also introduce a label transfer operation, based on the approach of [11, 12] and generalizing the subdomain-aware labels of [2]. The labels can be designed to capture a variety of geometric or topological properties of contours, sublevel or superlevel sets while being computable in an efficient manner.

2 Prior work

Contour trees are an important tool allowing one to concisely describe the structure of isosurfaces of a scalar field as well as the way they evolve and interact as the isovalue is varied. They have been used as a tool to enhance scalar field visualization [1], speed up certain types of queries in geographical information systems [8] and facilitate isosurface extraction from volume datasets by helping to compute small seed sets [16,17]. These applications motivated efforts to develop increasingly simpler, faster and more general algorithms for computing contour trees. An $O(n \log n)$ algorithm for computing the contour tree in two dimensions was given in [8]. A simpler version of the 2D algorithm and an $O(n^2)$ algorithm for higher dimensions is given in [17]. An $O(n \log n)$ algorithm that works in three dimensions was proposed in [15]. A simplified and generalized (to any dimension) algorithm is introduced in [4]. This algorithm computes the contour tree by combining the *split* and *join trees*. The split and join trees are of independent interest since they capture the properties of sublevel and superlevel components of the scalar field, naturally related to thresholding algorithms (we exploit this in the airway segmentation application discussed in Section 7). In [5], the authors describe an $O(n + t \log t)$ algorithm for computing a contour tree in any dimension for any type of grid, where t is the number of critical points. Algorithms for computing the *Reeb graph*, a generalization of the contour tree to domains that are not simply connected, are described in [6,7,13]. The work [11,12] introduces a method for labeling the edges of the contour tree with Betti numbers of the corresponding contours in $O(n \log n)$ time as well as a parallelizable divide and conquer algorithm that runs in $O(n + t \log n)$ time for regular grids, where n is the size of the domain of the input scalar field. Similar technique is used in [2] to obtain a labeling of the tree with the number of isosurface boundary components or, more generally, the number of connected components of the intersection of a contour with a simply connected subdomain.

The goal of this work is to extend the results of [2] to enable one to build labellings of contour, join or split tree edges with labels that describe more sophisticated properties of contours or sublevel or superlevel components. As an application, we discuss a procedure for computing local thresholds for airway segmentation from Computed Tomography (CT) scans based on topological analysis of sublevel components. An early variant of this algorithm, with less robust results, is described in [14].

3 Scalar fields and related trees: categorical definitions

Let $f : X \to \mathbf{R}$ be a continuous function (scalar field), where X is a compact and simply connected topological space. Several definitions given in this section are standard and are stated only for the sake of completeness.

Definition 1 *By a* sublevel set *we mean the set* $f^{-1}((-\infty, c])$ *for a real number c. Similarly, a* superlevel set *is the set* $f^{-1}([c, \infty))$ *where c is any real number. A* sublevel component *is a connected component of a sublevel set. Analogously, a superlevel component is a connected component of a superlevel set.*

Sublevel and superlevel sets and components naturally arise when applying a threshold to an image, which is perhaps the simplest possible approach to image segmentation.

Definition 2 *By a* level set *of f we mean the set $f^{-1}(\{c\})$, where c is a real number. A* contour *is a connected component of a level set.*

Level sets are typically used to represent boundaries of a region with a certain property defined by the scalar field. For example, this could be the boundary of a certain type of tissue in a medical image. The goal of the next two definitions is to build a framework for representing the structure of contours.

Definition 3 *The category of scalar fields denoted by $\mathscr{S}$ is defined as follows.*

(i) *Its objects are continuous scalar fields, i.e. continuous functions $f : X \to \mathbf{R}$ where X is a compact topological space.*
(ii) *Its morphisms are value-preserving continuous mappings. More precisely, for two objects $f : X \to \mathbf{R}$ and $g : Y \to \mathbf{R}$, α is a morphism of f into g if and only if α is a continuous mapping of X into Y such that $g \circ \alpha = f$.*

Definition 4 *The* contour tree functor *$\mathscr{CT} : \mathscr{S} \to \mathscr{S}$ is defined as follows.*

(i) *For an object $f : X \to \mathbf{R}$ in $\mathscr{S}$, $\mathscr{CT}(f)$ is the scalar field $\hat{f}$ induced by f defined on the quotient space $X/\equiv_f$, where $\equiv_f$ is the equivalence relation on X defined by*

$$x_1 \equiv_f x_2 \Leftrightarrow f(x_1) \text{ and } f(x_2) \text{ belong to the same contour.}$$

Thus, $\hat{f}([x]_{\equiv_f}) = f(x)$ for any element $x \in X$.
(ii) *For a morphism $\alpha : f \to g$ in $\mathscr{S}$, where $f : X \to \mathbf{R}$ and $g : Y \to \mathbf{R}$, $\mathscr{CT}(\alpha)$ is the map of $\mathscr{CT}(f)$ into $\mathscr{CT}(g)$ induced by α defined by*

$$\mathscr{CT}(\alpha)[x]_{\equiv_f} := [\alpha(x)]_{\equiv_g}.$$

The points of the quotient space $X/\equiv_f$ (the domain of $\mathscr{CT}(f)$) are in one-to-one correspondence with the contours of f. To a point of $X/\equiv_f$ corresponding to a contour C, the induced function $\hat{f}$ assigns the value of f at any point of C.

The mapping $\mathscr{CT}(\alpha)$ in the above definition is well defined and value-preserving because α is value-preserving and therefore maps contours into contours corresponding to the same scalar value. Its continuity follows from the continuity of α. Points of the domain of the scalar field $\mathscr{CT}(f)$ are in one-to-one correspondence with contours of the scalar field f.

Now, we turn to definition of the split tree functor. Points of the split tree are in one to one correspondence with the sublevel components. We first introduce a few useful operations on sublevel components that will allow us to specify the topology of the split tree.

Definition 5 *Let $f : X \to \mathbf{R}$ be a continuous function with X a compact topological space.*

(i) *For a sublevel component C let $\upsilon(C)$ be the maximum value of f in C,*
(ii) *For a compact set K let $\lceil K \rceil_f$ be the smallest sublevel component of f containing K,*

(iii) *For a real number s, let $L_f(s)$ be the family of all sublevel components C with $\upsilon(C) > s$,*

(iv) *For a sublevel component C, let $U_f(C)$ be the family of all sublevel components of f contained in but not equal to C.*

With the notation introduced above, we can define the split tree functor.

Definition 6 *The* split tree functor $\mathscr{ST} : \mathscr{S} \to \mathscr{S}$ *is defined as follows.*

(i) *For an object $f : X \to \mathbf{R}$, define the domain* Split(f) *of $\mathscr{ST}(f)$ as the family of all sublevel components of f. The subbasis of the topology on* Split(f) *consists of the sets $L_f(s)$ and $U_f(C)$ for all real numbers s and all sublevel components C. The scalar function $\mathscr{ST}(f) :$ Split(f) $\to \mathbf{R}$ is induced by f, i.e. defined by*

$$\mathscr{ST}(f)(C) := \upsilon(C).$$

(ii) *For a morphism $\alpha : f \to g$ in $\mathscr{S}$, where $f : X \to \mathbf{R}$ and $g : Y \to \mathbf{R}$, $\mathscr{ST}(\alpha)$ is the map of $\mathscr{ST}(f)$ into $\mathscr{ST}(g)$ induced by α, i.e. given by*

$$\mathscr{ST}(\alpha)(C) := \lceil \alpha(C) \rceil_g.$$

The join tree functor $\mathscr{JT} : \mathscr{S} \to \mathscr{S}$ can be defined in analogous manner. The only difference is that it is based on superlevel rather than sublevel sets.

In what follows by the contour, split or join tree of a scalar field f we shall mean respectively the scalar functions $\mathscr{CT}(f)$, $\mathscr{ST}(f)$ or $\mathscr{JT}(f)$ or their domains.

4 Labels

We use integer labels assigned to points of the contour, split or join trees to represent topological properties of the corresponding contours, sublevel or superlevel sets. The labels are constant along the edges of the trees and therefore finitely representable. They can be used to find contours or sublevel or superlevel components with desired user-specified topological properties.

It is convenient to think about labelings as functions that assign an integer to an element of a set (in our case, the set consists of points of one of the trees). The label transfer operation that can be used to build complex labelings is described by the following definition (see Figure 2 for an example)

Definition 7 *Let A and B be two sets and $\alpha : A \to \mathbf{Z}$ be a labeling of the set A. A function $f : A \to B$ is called* summable *if and only if the pre-image of every point $b \in B$ is finite. The labeling β of the set B defined by*

$$\beta(b) := \sum_{a \in f^{-1}(\{b\})} \alpha(a)$$

is called the transfer of α with f *and denoted $f_{\#}(\alpha)$.*

Other operations on labels can be performed. Labelings of the same set can be pointwise added or subtracted from each other. Also, one can apply an integer function to a labeling (by applying it to the label of each element).

There are at least two elementary contour, split or join tree labelings that could be used to build more sophisticated ones. The simpler one assigns the label of 1 to every point of the tree. We shall denote this labeling by $\mathbf{1}_{\mathscr{C}\mathscr{T}(f)}$, $\mathbf{1}_{\mathscr{S}\mathscr{T}(f)}$ and $\mathbf{1}_{\mathscr{J}\mathscr{T}(f)}$ (respectively), where $f : X \to \mathbf{R}$ is the underlying scalar field. Clearly, this labeling is not of interest by itself, but it can be used to build more interesting ones by means of label transfer or other operations.

The other labeling, introduced in [11], assigns the Euler characteristic of the corresponding contour, sublevel or superlevel set (respectively) to every point of the tree. This labeling will be denoted by $\chi_{\mathscr{C}\mathscr{T}(f)}$, $\chi_{\mathscr{S}\mathscr{T}(f)}$ or $\chi_{\mathscr{J}\mathscr{T}(f)}$ (respectively).

The above described labelings (together with the contour, join or split trees) can be computed in $O(n \log n)$ time, where n is the size of the input scalar field (i.e. number of simplices if f is piecewise linear or number of samples for a regularly sampled volume dataset) using the algorithms of [4, 5, 11, 12, 15].

5 Finite representability

Contour, split or join trees may display pathologies for general continuous scalar fields but, in most practical cases, they are finite trees. In particular, this is true for piecewise linear scalar fields or regularly sampled scalar fields in any dimension (based on either multilinear or nearest neighbor interpolation), provided the domain is simply connected. Using nearest neighbor interpolation directly does not lead to continuous functions, but still leads to well-defined trees if the algorithm of [3] is used to extract contours, sublevel and superlevel sets. In this case, any sublevel set or component has the same homotopy type as union of voxels corresponding to the samples inside it.

For the purpose of this work, contour, join and split trees are represented as finite trees. We store a real value (the value of the induced function on the tree which we refer to as *height*) at every vertex of the tree. A labeling of any type of tree is represented as an assignment $\bar{\alpha}$ of an integer to any tree element, i.e. vertex or edge. The process of building the element of $\mathscr{S}$ from the combinatorial tree representation is straighforward and basically boils down to drawing the graph – a special case of geometric realization of a simplicial complex [10]. The scalar function is obtained by interpolating the heights from the vertices to edges in any monotone manner. The label of a vertex v is $\bar{\alpha}(v)$ and labels of all points in the interior of an edge e are $\bar{\alpha}(e)$.

The finite representation of a tree described above is not unique. One can subdivide edges of the tree to represent an isomorphic object in $\mathscr{S}$. This process is called *augmentation* in [11]. Augmentation is useful when dealing with label transfer: Whenever we want to transfer labels from a tree T_1 to a tree T_2 with a mapping g, we add images of all vertices of T_1 under g to the set of vertices of T_2. This enables us to represent the mapping g as a mapping of vertices of T_1 into vertices of T_2. Note that not all height-preserving maps of vertices of T_1 into vertices of T_2 define a valid morphism in $\mathscr{S}$ between the two trees: this is true only if, for every edge of T_1, there exists a monotone path in T_2 connecting the images of the endpoints of that edge (Figure 1). In practice,

Fig. 1. Left: a mapping of vertices of T_1 into vertices of T_2 that *does not* define a height-preserving continuous map between trees (i.e. a morphism in S). There is no monotone path between the images of vertices a and b in T_2. Right: a mapping that does define a morphism in S. For every edge of T_1 there is a monotone path connecting the images of the endpoints in T_2.

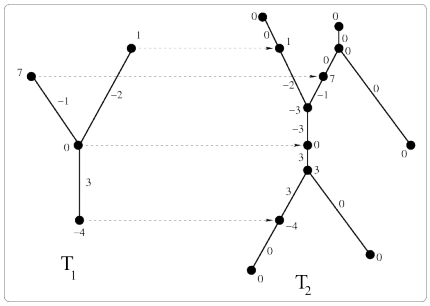


Fig. 2. An example of a label transfer operation. The map between the trees is height-preserving and is defined by the mapping of the vertices (dashed arrows). The labels are transferred from the left tree to the right tree.

this is not an issue since the contour, split or join tree functors always produce valid height-preserving maps between the trees, so their combinatorial representations have the above property. An example of a label transfer operation for a height-preserving map of two trees is shown in Figure 2.

6 Efficient label transfer

Labels can be transferred between trees in linear time using a generalization of the procedures described in [2, 11]. Based on the discussion of Section 5, we define the input to our procedure as:

(i) Two finite trees G_1 and G_2 with real values (heights) given at the vertices, representing two objects T_1 and T_2 in $\mathscr{S}$.
(ii) A representation of a labeling α of T_1: a function $\bar{\alpha}$ assigning an integer to any edge or vertex of G_1.
(iii) A mapping ι that assigns a vertex of G_2 to every vertex of G_1, representing a morphism $f : T_1 \to T_2$. ι is required to preserve the vertex heights.

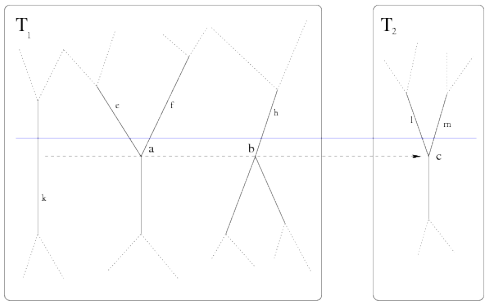


Fig. 3. Two vertices a and b of the tree T_1 are mapped into the vertex c of T_2. The labels $\bar{\beta}(l)$ and $\bar{\beta}(m)$ are not uniquely determined from the given information, but $\partial_+\bar{\beta}(c)$ is. To see that, take two points on edges l and m just barely above c (intersections of the solid horizontal line and the edges l and m in the figure). Pre-image of these two points under f contains exactly one points on each edge in $\mathcal{N}_+(a)\cup\mathcal{N}_+(b)$ (intersections of the blue line with the edges e, f and h in T_1). There may be other points in the pre-image (in this example, a point on edge k). The pre-image of c consists of a, b and a point on k. Now, a simple calculation based on the above observations and Definition 7 shows that $\partial_+\bar{\beta}(c) = (\bar{\beta}(l)+\bar{\beta}(m)) - \bar{\beta}(c) = (\bar{\alpha}(e)+\bar{\alpha}(f)+\bar{\alpha}(h)+\bar{\alpha}(k)) - (\bar{\alpha}(a)+\bar{\alpha}(b)+\bar{\alpha}(k)) = \partial_+\bar{\alpha}(a)+\partial_+\bar{\alpha}(b) = \delta_+(c)$.

The output is the function $\bar{\beta}$ assigning an integer to every vertex or edge of G_2 that represents the labeling $\beta = f_\#(\alpha)$.

In what follows, for a vertex v of a tree by $\mathcal{N}_+(v)$ (respectively, $\mathcal{N}_-(v)$) we shall denote the set of all edges connecting v with a vertex of a higher (respectively, lower) height. If $\bar{\gamma}$ is a function assigning an integer to every vertex and edge of the tree, we define $\partial_+\bar{\gamma}(v) := \sum_{e\in\mathcal{N}_+(v)}\bar{\gamma}(e) - \bar{\gamma}(v)$ and $\partial_-\bar{\gamma}(v) := \sum_{e\in\mathcal{N}_-(v)}\bar{\gamma}(e) - \bar{\gamma}(v)$ for each vertex v.

The algorithm will use two variables per vertex v of the tree G_2, $\delta_+(v)$ and $\delta_-(v)$. Their initial values are given by the following formula.

$$\delta_\pm(v) = \sum_{w:\iota(w)=v} \partial_\pm\bar{\alpha}(w).$$

To initialize them in linear time, start by setting $\delta_+(v)$ and $\delta_-(v)$ to zero. Then, for every vertex w of G_1, add $\partial_\pm\bar{\alpha}(w)$ to $\delta_\pm(\iota(w))$. It is not hard to see that $\bar{\beta}$, representing $f_\#(\alpha)$, satisfies

$$\partial_\pm\bar{\beta}(v) = \delta_\pm(v) \tag{1}$$

for every vertex v of G_2 (Figure 3). The rest of this section is focused on computing $\bar{\beta}$ that satisfies (1).

Pick a leaf vertex (i.e. vertex of degree 1) v of G_2. Let e be its incident edge and v' be the vertex of e other than v. There is a simple way to compute $\bar{\beta}(v)$ and $\bar{\beta}(e)$ from values of $\delta_\pm(v)$. To see this, consider two cases.

Case A: v is above v' (i.e. its height is larger). Since there are no edges having v as the lower endpoint and the only edge having v as the upper endpoint is e, $\partial_+\bar{\beta}(v) = -\bar{\beta}(v)$ and $\partial_-\bar{\beta}(v) = \bar{\beta}(e) - \bar{\beta}(v)$. Since we are looking for $\bar{\beta}$ satisfying Equation

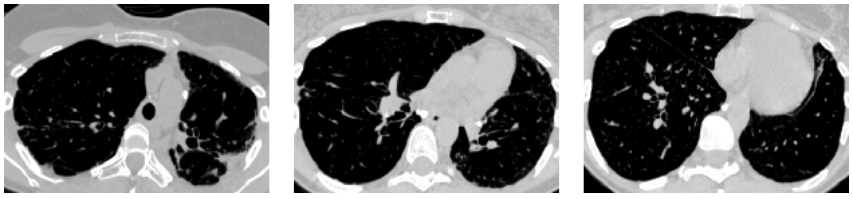


Fig. 4. Slices through one of the input CT scans. Airway sections appear as dark, usually regularly shaped blobs. Inside the lung area, they are surrounded by higher intensity border. The images suffer from the typical medical image artifacts, including variations of intensity inside the airways.

(1), $-\bar{\beta}(v) = \delta_+(v)$ and $\bar{\beta}(e) - \bar{\beta}(v) = \delta_-(v)$. Hence $\bar{\beta}(v) = -\delta_+(v)$ and $\bar{\beta}(e) = \delta_-(v) - \delta_+(v)$.

Case B: v is below v'. By an analogous argument, $\bar{\beta}(v) = -\delta_-(v)$ and $\bar{\beta}(e) = \delta_+(v) - \delta_-(v)$.

In order to compute $\bar{\beta}$ satisfying Equation (1), we select any leaf vertex v from G_2. Let v' and e be as described in the previous paragraph. We compute $\bar{\beta}(v)$ and $\bar{\beta}(e)$ as described above and remove v and e from G_2. Removal of the edge e affects $\partial_+\bar{\beta}(v')$ (in case A – since e disappears from $\mathcal{N}_+(v')$) or $\partial_-\bar{\beta}(v')$ (in case B). We make up for the removal as follows.

In case A: We subtract $\bar{\beta}(e)$ (i.e. $\delta_-(v) - \delta_+(v)$) from $\delta_+(v')$
In case B: We subtract $\bar{\beta}(e)$ (i.e. $\delta_+(v) - \delta_-(v)$) from $\delta_-(v')$

This process is repeated until no leaf vertices exist. Since G_2 is a tree, this reduces G_2 to a single isolated vertex, call it v_0. We finish by setting $\bar{\beta}(v_0) := -\delta_+(v_0)$.

To determine the next leaf vertex to be processed efficiently, we maintain a list containing all leaf edges of the current tree G_2. At startup, all leaf vertices of the input graph G_2 are inserted into the list. Whenever a leaf vertex v is removed from G_2 together with its incident edge e, we test if the other vertex v' of e becomes a leaf vertex. If it does, we insert it into the list.

It is not hard to see that the label transfer algorithm runs in linear time in the size of input (total sizes of trees G_1 and G_2).

7 Application to airway segmentation

The categorical approach is a convenient tool allowing one to detect features in scalar fields. In this section, we describe a procedure for computing local thresholds for airway segmentation from CT (Computed Tomography) scans. It is based on the expectation that good local thresholds should lead to locally topologically simple and structurally stable sets and hence can be seen as a local variant of the contour spectrum method of [1]. Slices through one of our test datasets are shown in Figure 4.

A CT scan is a 3D image of resolution $N_x \times N_y \times N_z$. We will ignore any anisotropy of the image and assume that the spacing between samples is 1 along every dimension.

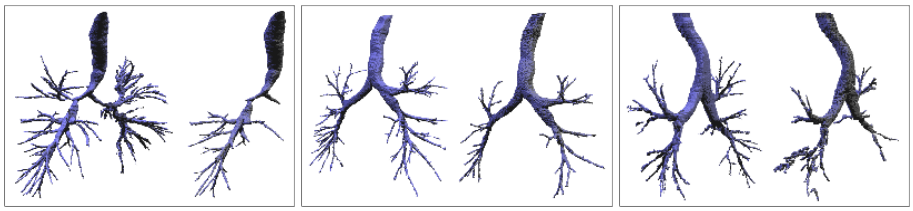


Fig. 5. Comparison between our local thresholding approach and global thresholding. The airway obtained with a global threshold is shown in the right image in each box.

Using nearest-neighbor interpolation, one can obtain a scalar field $f : D \to \mathbf{R}$, where $D = [0, N_x - 1] \times [0, N_y - 1] \times [0, N_z - 1]$ and the voxels of the CT scans correspond to the integer lattice points in D.

In what follows, we will consider slices and thick slices of the input CT scan. For integers i, j such that $0 \leq i \leq j < N_z$, let $D_{i,j} = \{(x,y,z) \in D | i \leq z \leq j\}$. We shall write D_i instead of $D_{i,i}$. Slices and thick slices are defined as the restrictions of f to the above defined sets: $f_i = f_{|D_i}$ and $f_{i,j} = f_{|D_{i,j}}$. By $\eta_{k \to i,j}$ $(i \leq k \leq j)$ we define the inclusion-induced morphism $f_k \to f_{i,j}$.

Our procedure uses a number of labelings of the edges of the tree $\mathscr{ST}(f_{i,i+T})$ for $i \in \{0, 1, \ldots, N_z - T - 1\}$ where T is a user-selected constant. In order to measure the total one-dimensional Betti number of the slices we use the labeling

$$L := \sum_{j=i}^{i+T} [\eta_{j \to i,i+T}]_{\#}(\mathbf{1}_{\mathscr{ST}(f_j)} - \chi_{\mathscr{ST}(f_j)}).$$

Note that f_j in the above formula is a 2D section of f. Its sublevel components are two-dimensional polygons. The Euler characteristics of a 2D polygon A is equal to $1 - C$, where C is the number of holes in A. Hence the labels that are summed in the above formula for L are all nonnegative.

To count the number of components in the intersection with each slice perpendicular the the z-axis we use labels

$$I_j := [\eta_{i+j \to i,i+T}]_{\#}(\mathbf{1}_{\mathscr{ST}(f_{i+j})})$$

for $j \in \{0, 1, \ldots, T\}$. We are also going to measure the Euler characteristics of the entire component using the labeling

$$T := \chi_{\mathscr{ST}(f_{i,i+T})}.$$

The airway reconstruction is obtained as follows. We set a conservative but small upper bound M on the intensity of a voxel inside the airway tree. We compute the labelings described above. Then, we mark all edges of $\mathscr{ST}(f_{i,i+T})$ with labels satisfying the following conditions: **(i)** $T = 1$, **(ii)** $L = 0$ and **(iii)** $0 < I_j \leq 5$ for $j \in \{0, 1, \ldots, T\}$. The conditions (i)-(iii) essentially enforce simplicity of the topology of the sections of the sublevel component, its global topology as well as cleanness of the branching structure. In particular, conditions (i) and (ii) force any marked component to have one and two-dimensional Betti numbers equal to zero. Condition (iii) constraints the sublevel set to

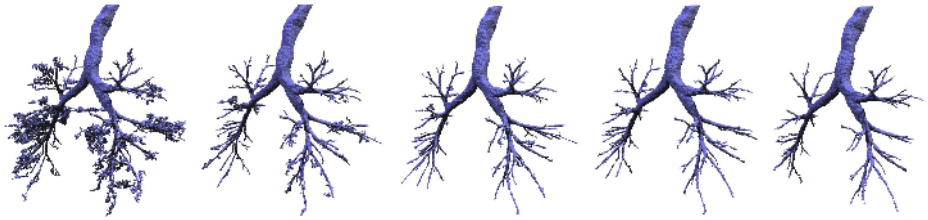


Fig. 6. Reconstructions for a high quality dataset for τ equal to (left to right) 0.0025, 0.005, 0.01, 0.02 and 0.04. Note that the results for $\tau = 0.01$ contains considerably more detail than the default value. However, decreasing τ below 0.01 causes leaks.

intersect any section, but at only a small number (at most 5) of connected components, all of which need to be simply connected by condition (ii). We use the upper bound of 5 since more than 4 airway bifurcations typically do not exist in a thick slice of width 10 (which is the width of a thick slice used here).

The number of marked edges is typically quite high. A standard approach to eliminate features that do not correspond to the structure of interest is to reject the ones that are not stable enough. Here, we essentially use the edge length as the stability measure. It works well for airway trees since the sublevel components that define the airway area are typically significantly darker than their surrounding wall, so one can expect few critical points near the wall. Thus, we go over all marked edges. If its vertices are at heights $h_1 < h_2$ we define its strength as $\min(h_2, M) - \min(h_1, M)$. For each edge stronger than a threshold τ, we compute the sublevel component (union of voxels) corresponding to the point on the edge with value just below $\min(h_2, M)$. We then union of all these components over $i \in \{0, 1, \ldots, N_z - 1 - T\}$.

The above procedure operates on slices and thick slices parallel to the *xy*-plane. We repeat the same process for slices parallel to the other two coordinate planes and union the results, obtaining a voxel set $\mathscr{V}$. Now, we compute the connected component of $\mathscr{V}$ containing a user-specified voxel inside the trachea. This is the output of our algorithm shown in Figure 5.

We used $T = 10$ to obtain results shown in this section. For each test scan, the voxel intensities were normalized before running the algorithm: for each axial slice, we scaled the original intensities to make the mean intensity of the slice equal to 1. For the normalized images, we set the conservative bound on intensity of a voxel inside an airway M to 0.5 and the bound on edge strength τ to 0.04.

We compared the output of the local thresholding procedure described in this section to segmentations obtained using the best possible global threshold (determined using the algorithm of [9]). Results for four of our test datasets are shown in Figure 5. Local thresholding generally yields more detail and has few artifacts such as leaks to the lung area. We also noticed that for higher quality input datasets decreasing τ below the default value of 0.04 may yield more detailed results (Figure 6). In practice, we expect τ to play the role of a regularization parameter, i.e. help one avoid local thresholds leading to noisy and structurally unstable local results. Ideally, this parameter should be

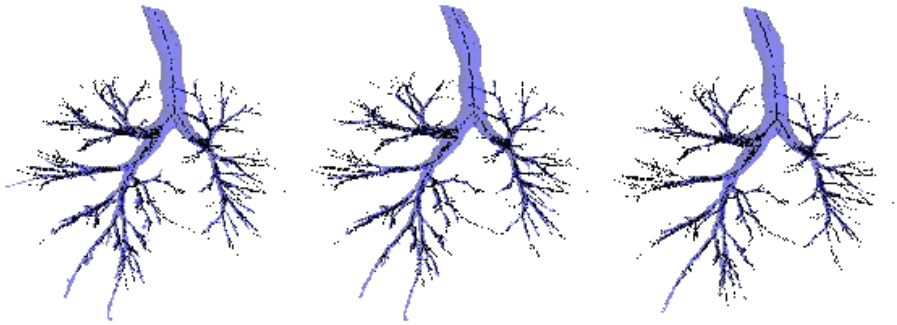


Fig. 7. Manually specified points (black) shown with the airway tree computed using our algorithm (blue transparent surface) for $\tau = 0.01$ (left), $\tau = 0.02$ (center) and $\tau = 0.04$. The algorithm is able to correctly identify the major airways, but, on average, is about 1 airway generation behind the human observer. There are a few small airway branches that were identified by the algorithm and were not originally marked by the observer. They were confirmed to be correct upon close examination of the image.

adjusted based on the quality of the input dataset. Figure 7 compares our output to a set of manually extracted points near the centerline of the airway tree.

8 Conclusion

We described a general framework for designing and computing topological descriptors for contours, sublevel and superlevel sets. The descriptors are represented as labelings of the contour, split or join trees and obtained by combining and transferring labels between trees. To transfer the labels, we take advantage of maps between the trees induced by value-preserving maps of scalar fields. We described a method for computing local thresholds for airway segmentation from CT scans using this framework.

It would be interesting to explore extensions of the framework to maps that are not value preserving or to general Reeb graphs. We would also like to work on decreasing the sizes of the trees arising in applications such as the airway tree segmentation described in Section 7 using ideas based on topology simplification algorithms. This may speed up the algorithm so that interactive or automatic search for the optimal parameter values (in particular, τ) becomes practical.

References

1. C. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *Proc. IEEE Visualization'97*, pages 167–175, 1997.
2. Nathanael Berglund and Andrzej Szymczak. Making contour trees subdomain-aware. In *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG'04)*, pages 188–191, 2004.
3. P. Bhaniramka, R. Wenger, and R. Crawfis. Iso-contouring in higher dimensions. In *Proceedings IEEE Visualization 2000*, pages 267–273, 2000.

4. H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24:75–94, 2003.
5. Yi-Jen Chiang and Xiang Lu. Simple and optimal output-sensitive computation of contour trees. Technical Report TR-CIS-2003-02, Polytechnic University, June 2003.
6. K. Cole-Mclaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. In *Proc. ACM Sympos. Comput. Geom.*, pages 344–350. ACM Press, 2003.
7. Kree Cole-McLaughlin, Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Loops in reeb graphs of 2-manifolds. *Discrete & Computational Geometry*, 32(2):231–244, 2004.
8. M. de Berg and M. van Kreveld. Trekking in the alps without freezing or getting tired. *Algorithmica*, 18:306–323, 1997.
9. K. Mori, J. Hasegawa, J. Toriwaki, H. Anno, and K. Katada. Recognition of bronchus in three-dimensional x-ray ct images with applications to virtualized bronchoscopy system. In *Proceedings of ICPR'96*, pages 528–532, 1996.
10. James R Munkres. *Elements of algebraic topology*. Addison-Wesley Publishing Company, Menlo Park, Calif., 1984.
11. V. Pascucci and K. Cole-McLaughlin. Efficient computation of the topology of level sets. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 187–194, Washington, DC, USA, 2002. IEEE Computer Society.
12. V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38(2):249–268, October 2003.
13. Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas. Robust online computation of reeb graphs: simplicity and speed. *ACM Trans. Graph.*, 26(3):58, 2007.
14. Andrzej Szymczak and James Vanderhyde. Airway segmentation by topology-driven local thresholding. In *Proc. SPIE Medical Imaging 2008 (Vol. 6914)*, page 69143D, 2008.
15. S. P. Tarasov and M. N. Vyalyi. Construction of contour trees in 3d in $o(n \log n)$ steps. In *Proc. 14th Ann. Sympos. Comput. Geom. 1998*, pages 68–75, 1998.
16. M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface generation. In Sanjay Rana, editor, *Topological Data Structures for Surfaces*, pages 71–86. Wiley Europe, March 2004.
17. M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. R. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proc. 13th ACM Annual Symposium on Computational Geometry (SoCG)'97*, pages 212–220, 1997.

Feature Tracking Using Reeb Graphs

Gunther H. Weber[1], Peer-Timo Bremer[2], Marcus S. Day[1], John B. Bell[1], and Valerio Pascucci[3]

[1] Lawrence Berkeley National Laboratory, {GHWeber|MSDay|JBBell}@lbl.gov
[2] Lawrence Livermore National Laboratory, ptbremer@llnl.gov
[3] University of Utah, pascucci@sci.utah.edu

Abstract. Tracking features and exploring their temporal dynamics can aid scientists in identifying interesting time intervals in a simulation and serve as basis for performing quantitative analyses of temporal phenomena. In this paper, we develop a novel approach for tracking subsets of isosurfaces, such as burning regions in simulated flames, which are defined as areas of high fuel consumption on a temperature isosurface. Tracking such regions as they merge and split over time can provide important insights into the impact of turbulence on the combustion process. However, the convoluted nature of the temperature isosurface and its rapid movement make this analysis particularly challenging.
Our approach tracks burning regions by extracting a temperature isovolume from the four-dimensional space-time temperature field. It then obtains isosurfaces for the original simulation time steps and labels individual connected "burning" regions based on the local fuel consumption value. Based on this information, a boundary surface between burning and non-burning regions is constructed. The Reeb graph of this boundary surface is the tracking graph for burning regions.

Key words: Topological data analysis, Feature tracking, Combustion simulation, Reeb graph, Tracking graph, Tracking accuracy

1 Introduction

Understanding combustion processes is a fundamental problem impacting areas such as engine and stationary power plant design, both in terms of production efficiency and pollutant emission. Fuel-lean flame configurations are of particular interest since such flames generically produce far lower pollutants than comparable fuel-rich or stoichiometrically mixed flames. However, such flames are difficult to stabilize in the sort of quasi-steady robust configurations necessary for practical applications, particularly when using advanced fuel mixtures, such as hydrogen-air and hydrogen-seeded methane-air. These advanced fuel mixtures, selected to reduce the use of carbon-based fuels and subsequent emissions, often burn in cellular patterns of intense chemical reaction, separated by regions of local extinction. A broad range of classical flame propagation models used in analysis and the engineering design of practical combustion systems are based on the notion that a flame is a thin continuous interface separating cold reactants from hot products. Such models are not suitable for modelling cellular flames. It therefore is of great practical interest to understand this mode of combustion, with the ultimate goal of incorporating the cellular burning behavior into revised engineering design models.

Fig. 1. Boundaries of the burning regions (Fig. C. 66, center illustration) identified by contouring.

In the present study, detailed numerical simulation is used to evolve a turbulent reacting hydrogen-air mixture in an idealized configuration. Characteristics of the flow and turbulence used in the simulation represent conditions similar to those found in a practical turbulent combustor. The fields computed include the velocity and a set of scalar quantities representing the temperature and mass densities of a large number of molecular species. In the simulation these quantities are evolved in detail, naturally forming into locally disconnected (cellular) burning structures. It has long been known that the cellular hydrogen flame patterns correlate with a narrow band of intermediate temperature. Because of this, we can build a simplified analysis of the detailed simulation data by looking at the variable rate of combustion along a representative isotherm (see Fig. C. 66, left illustration, for a representative example of such an analysis). Thresholding the rate of fuel consumption on the isotherm yields a geometric representation of the burning cells (Fig. C. 66, center illustration), and the time-evolution of these structures provides an important basic characterization of the flame.

Our goal is to compute a tracking graph whose nodes represent creation/destruction and split/merge events of burning cells and whose arcs represent evolution of cells over time. Tracking burning cells over time presents two main challenges. First, the aim is not to track surfaces per se, but features embedded in time-dependent surfaces. Second, to utilize the tracking graph fully, a correlation between a specific burning cell in one time step and a node/arc on the tracking graph must be maintained.

We derive this information by considering the boundaries of burning regions (Fig. 1), obtained via a second contouring operation based on the fuel consumption rate on the triangle mesh comprising the isotherm, and determining when these boundaries split and merge over time. Our approach extracts the boundary of the space-time volume that each of these contours sweeps out over the course of the simulation and reduces the four-dimensional tracking problem to the computation of the *Reeb graph* of the resulting surface (using time as a Morse function), which encodes merge and split events. We also provide an in-depth analysis of tracking accuracy by comparing simulations performed under varying conditions and with different temporal and spatial resolutions. While the examples provided in this paper focus on the analysis of combustion simulations, the underlying framework for topology-based tracking obviously applies to a wide range of application areas.

2 Related Work

2.1 Time-dependent Isosurface Extraction

Given a manifold $\mathbb{M}$ and a scalar function $f : \mathbb{M} \to \mathbb{R}$, an *isosurface* or *level set* of f at isovalue s is defined as all points on $\mathbb{M}$ with function value s. If $\mathbb{M}$ is described by a three-dimensional rectilinear grid, isosurfaces can be efficiently constructed using the Marching Cubes algorithm [11,13]. More recently, Bhaniramka et al. [2] have extended this approach using convex hull computations to define triangulations within a grid cell. Their method eliminates the need for extensive case tables, ensures consistency, and easily generalizes to higher dimensions.

One can exploit this generality to visualize time-varying isosurfaces by treating time as an additional dimension creating the space-time manifold $\mathbb{M} \times \mathbb{R}$. Extracting a level set of $\mathbb{M} \times \mathbb{R}$ results in a three-dimensional hypersurface (embedded in four-dimensional space-time) comprised of tetrahedra. Intersecting this volumetric space-time mesh with planes of constant time values results in traditional two-dimensional isosurfaces at a given time. Therefore, the hypersurface is a comprehensive description of the temporal evolution of the corresponding two-dimensional isosurfaces.

2.2 Topological Analysis and Reeb Graphs

By construction, a level set of f provides only limited information on f as a whole, but very detailed information about a specific function value. To understand the global behavior of f, it is useful to analyze its *contours*, which are the connected components of all level sets of f. Contracting the contours of a *Morse function* f into points creates the *Reeb graph*, which encodes the topology of all level sets of f. *Nodes* of the Reeb graph are formed by the contours passing through critical points of f, i.e., points where contour topology changes. Its *arcs* are formed by the remaining contours, i.e., by the family of contours that do not change topology. For more details on Reeb graphs and algorithms to construct them efficiently, we refer the reader to the description by Pascucci et al. [14].

In our application, the boundaries of the burning regions sweep out a two-dimensional space-time surface S (see Fig. C. 66, right illustration). Using the time-coordinate of all vertices of S as the function f, the level set of f at value t describes the cell boundaries at time t. Therefore, the Reeb graph of f encodes the temporal evolution of the boundaries and thus can be used as the tracking graph describing how cells merge and split over time.

2.3 Feature Tracking

Defining and tracking features of interest has long been of interest to the visualization community. Most relevant to our work is feature tracking in the context of scalar field visualization. Here, one usually is interested in tracking features defined by thresholding or isosurface extraction [12].

Tracking algorithms can roughly be divided into two categories: tracking by geometry and tracking by topology. Methods in the former category use various forms of overlap and/or distance between geometric attributes, e.g., the center of gravity [15] or volume overlap [16, 17] for tracking. Laney et al. [10] use a similar approach to track

bubble structures in turbulent mixing. Ji et al. [7, 8] track the evolution of isosurfaces in a time-dependent volume by extracting the 3D space-time isosurface in a 4D space.

Methods in the latter category [5, 19] compute tracking information topologically using, for example, Jacobi sets [4], which describe the paths all critical points take over time. Sohn and Bajaj [18] introduce a hybrid approach using volume matching similar to Silver and Wang [16, 17] instead of topological information [4, 19] to define correspondences between contour trees.

Geometric tracking, in general, is ill-suited for the flame surfaces of interest here. As illustrated by Fig. C. 66, flame surfaces may be convoluted and contain many densely packed burning cells. As a result, geometric distance is not a good predictor for tracking burning cells, as flame sheets may fold on themselves, creating cells in close proximity yet far apart relative to the flame surface. Current algorithms for topological tracking using Jacobi sets are not capable of dealing with large embedded surfaces.

3 Feature Tracking Algorithm

An important aspect of the combustion process is how burning cells evolve over time. In particular, scientists are interested in determining how and when cells are created/destroyed and merge/split. To obtain this information, we track the boundaries of burning cells over time. For each time step, these boundaries are a set of curves, as shown in Fig. 1.

If we follow these boundary lines over time, they sweep out surfaces; see Fig. C. 66, right illustration. Using a 3D time surface in four dimensions, it is possible to extract this swept surface directly. In a first step, one views the data set as a 4D data set, with time as the fourth dimension, by concatenating all available time steps. The resulting data set serves as input for Marching Cubes to extract a 3D time surface comprised of tetrahedra. Furthermore, one also calculates the fuel consumption rate at the vertex positions and associates the rates with the individual vertices. A second step computes an isosurface of fuel consumption rate on this tetrahedral mesh, resulting in the swept boundary surface as shown in Fig. C. 66, right illustration.

Based on this swept surface, it then is possible to track how burning regions change. If we take this swept boundary surface as a manifold, then the elapsed simulation time is a Morse function on it, and level sets correspond to boundaries at a given time. Critical points, where the number of contours of the level set changes, correspond to the changes of the number of burning cells. For example, in Fig. C. 66, right illustration, in the area marked with the letter "A" a new burning region is created, and in the area around the letter "B," a burning region splits into two separate burning cells. Consequently, the Reeb graph of the swept boundary surface with time as a Morse function also is a tracking graph for individual burning cells (after simplification).

Our actual implementation is based on this fundamental concept with some additional refinements that we describe in the following. Due to data set size, we pipeline individual processing steps and stream data sets through this pipeline. While our pipeline performs all these steps on 3D data sets, we use the 2D case shown in Fig. 2 to illustrate the underlying concepts.

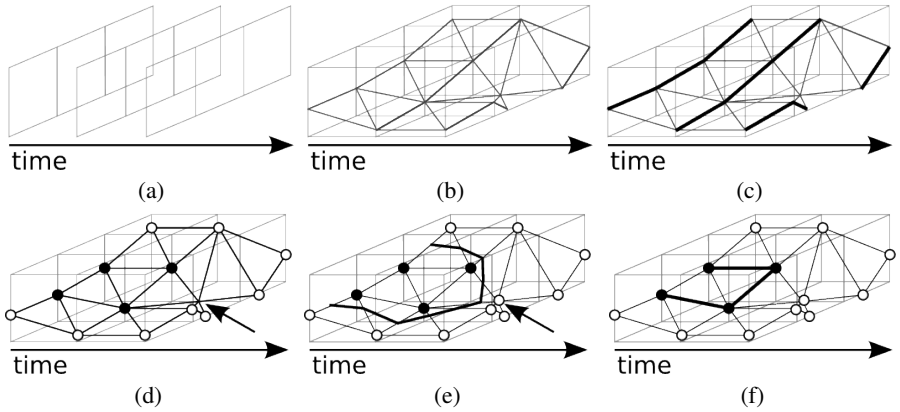


Fig. 2. Our method traces the evolution of burning regions and extinction pockets by tracking their boundaries. This figure illustrates the underlying concepts for the 2D case; the 3D case is analogous. (a) Input data comes as a set of discrete time slices. (b) We treat time as an additional, third dimension and extract an isosurface. The resulting time surface makes it possible to correlate isotherms from different time steps with each other. (c) We then extract isotherms (contour lines in 2D) for all original time steps by filtering all lines that have only vertices in the time step of interest (bold lines in the figure). (d) We classify isotherm vertices at original time steps as either burning (solid black discs in the figure) or non-burning (empty circles in the figure) based on the local fuel consumption rates and simplify the segmentation using a Morse complex-based method. We note that vertices between time steps (arrowed vertex in figure) are not classified, yet. (e) We classify vertices between time steps (arrowed vertex in figure) by thresholding, and extract boundaries separating burning regions and extinction pockets (bold lines). (f) To simplify this step, we snap intersection points on edges that connect burning and non-burning vertices to the burning vertex (bold lines). We do so because it simplifies data processing and does not change the topology of the boundary surface. The Reeb graph (not shown) of the resulting surface is the desired tracking graph.

3.1 Extracting the Time Surface and Isosurfaces at the Original Time Steps

To extract the boundary of burning cells over time, we need a means to correlate isosurfaces in different time steps to each other. For this purpose, we add time as an additional dimension to the original 3D grid, resulting in a (virtual) 4D hyper-grid containing all time steps of the simulation. We then extract a 3D isovolume that encodes the time evolution of the flame surface, see Fig. 2(b). Intersecting this 3D (tetrahedral) isovolume with a plane of constant time produces the flame surface for that particular time step. We use the algorithm of Bhaniramka et al. to compute the isovolume and refer the reader to [2] for a more detailed description.

Here we are interested only in isosurfaces at times that correspond to original time steps in the simulation. In this special case, intersecting with a plane of constant time can be reduced to a filter operation. Starting from the space-time tetrahedra, we obtain the isosurface as the set of all tetrahedra-faces (triangles) whose vertices all lie in the time step of interest. We perform this filter operation for all time steps of the original data set and save the corresponding flame surfaces. In addition to vertex positions, we also interpolate the fuel consumption scalar field and associate these values with the appropriate vertices.

3.2 Classification of Burning and Non-Burning Regions

Once all isosurfaces are extracted, we segment the surface for each time step into burning and non-burning regions. For this purpose, we do not apply simple thresholding to identify burning cells. Instead, we calculate the Morse complex of the isotherm for each individual time step, with fuel consumption as the associated Morse function [3]. This approach supports labeling each region with a unique identifier and performing persistence-based [6] simplification on the number of burning regions, merging small burning regions with nearby larger burning regions. While the hierarchy produced by this approach allows free selection of the fuel consumption threshold and the simplification persistence, we compute the segmentation for tracking using a persistence of 0.1 and a fuel consumption threshold of 2.6. These values are based on parameter studies performed in [3]. The result is a segmentation of the flame surfaces into individual burning cells that assigns a unique (within the time step) identifier to each cell. We propagate this identifier to all interior vertices of each cell.

Remember that the vertices of the flame surfaces are vertices of the space-time isovolume as well. Therefore, once we have computed the individual segmentations, only vertices in the regular time steps of the isovolume are classified as burning/non-burning, and the burning ones have a additional identifier attached. (We note that even though Fig. C. 66, right illustration, shows a projection of the swept surface in 3D space, the triangle mesh actually is embedded in four-dimensional space, where each vertex also has a time coordinate. It is possible that the time coordinate of a vertex lies between two time steps of the simulation.) The remaining non-classified vertices are those falling between time steps. We classify these as burning/non-burning based on their fuel consumption value, but assign no identifier. This value of fuel consumption must be interpolated from the data available at adjacent time steps.

3.3 Extracting the Swept Boundary Surface

Based on the classification results, we could use a standard marching tetrahedra algorithm to extract the boundary between burning cells and extinction regions, shown as the bold line in Fig. 2(e). Each vertex of this space-time surface lies on an edge of the isovolume that connects a burning to a non-burning vertex (of the isovolume). As will be discussed later, we only are interested in the connectivity of the space-time surface and not its geometry. Therefore, we can simplify the extraction by snapping the intersection points on the tetrahedra edges to the burning vertices of the isovolume as shown in Fig. 2(f). This snapping reduces the extraction of the space-time surface to another filter operation. Starting from the isovolume, we discard all tetrahedra whose vertices are either all burning or all non-burning. We also discard tetrahedra with only a single burning vertex since the snapping operation will reduce the iso-triangle to a single point that does not contribute to the surface connectivity. Tetrahedra that have two burning and two non-burning vertices require special attention. Here, snapping intersection points to the burning vertices results in two degenerate triangles, coinciding with the edge connecting the burning vertices. To maintain proper connectivity, we must add at least one of these triangles to the surface. Finally, tetrahedra with three burning vertices produce exactly one triangle identical to one of its faces that we add to the surface.

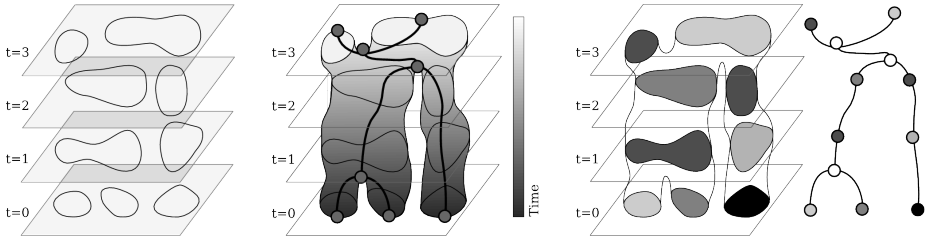


Fig. 3. Using the Reeb graph to compute a tracking graph. (a) Several time steps of a 2D data set with the boundaries of several cells indicated. (b) The space-time boundary created by the boundaries of the cells in (a) as they are interpolated over time. The surface grayscale illustrates time as the Morse function, and the Reeb graph for the surface is shown in black and grey. (c) Each cell within a time step is assigned a unique identifier, which we use to augment the Reeb graph. Note that the final tracking graph still contains nodes between time steps without any identifier (shown in white).

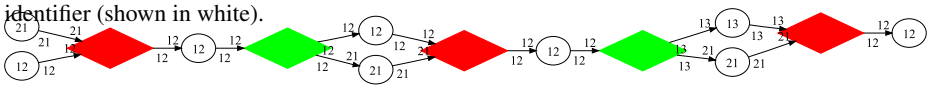


Fig. 4. An extended merge event: Two cells (initially labeled 12 and 21) merge and split twice before ultimately merging. These are inherent instabilities in segmenting by thresholding caused by saddles very close to the cut-off.

3.4 Computing the Tracking Graph

Ignoring the snapping operation, the space-time surface is the surface defined by the boundaries of the burning cells as they evolve over time, see Fig. 3(b). Computing the Reeb graph of the time function on this surfaces is the tracking graph of the cells, as shown in Fig. 3 (c). Since the Reeb graph ignores the embedding of the underlying manifold (the space-time surface) snapping the vertices to those of the isovolume leaves the Reeb graph unchanged. As discussed in Sect. 2.2, each arc of the Reeb graph corresponds to contours that do not change topology as the function value changes. The Morse function in this case is time and therefore, arcs describe boundaries of burning cells that, over time, neither change genus, nor interact with other cells. Furthermore, at each time value for which a flame surface has been extracted, all vertices forming a contour have the same identifier by construction. We augment the Reeb graph with this information if necessary by adding nodes of valence two. This approach allows us to correlate specific cells of the flame surface with points in the tracking graph. This correlation is crucial when analyzing the graph. Finally, the identifiers also enable us to compute the genus of the burning cells. Each hole in the interior of a cell creates its own boundary component and thus appears in the Reeb graph as a separate component. However, using the cell identifiers, we can collect easily all components belonging to a particular cell and thus compute its genus.

Once we have computed the Reeb graph, we simplify [1] all loops that span less than a full time step since they must represent artifacts of the construction (using linear interpolation between time steps no true feature should exist between time steps). For display purposes we also merge all nodes representing the same cell while keep-

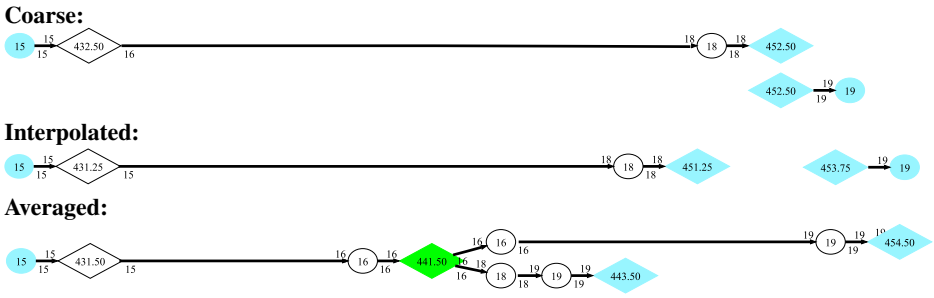


Fig. 5. "Lost" tracking of a burning region.

ing track of its genus. Finally, we simplify all loops spanning exactly one time step to streamline the graph. This simplification helps to resolve segmentation instabilities caused by saddles close to the threshold value. Using this information, we then construct a simplified graph representing the life of each component. Nodes of the graph indicate significant events: birth, death, splitting and merging. Diamond-shape nodes indicate events that occur between time steps. As shown in Fig. 4, one sometimes finds "extended" split/merge events in which two regions merge and split several time before finally merging/splitting. We also remove components of the graph that have an overall life-span of less than two time steps. We use "dot" [9] to layout the resulting graph.

4 Results

We have used our method to analyze simulations of a lean hydrogen flame burning under varying turbulence conditions labeled as "none," "weak" and "strong." With increasing levels of turbulence, we observe a qualitative change in the formation and propagation of burning cells. Our new tool can be used to quantify these changes. Fig. C. 67 shows a portion of the resulting tracking graph for the "none" turbulence case (the other two would produce a qualitatively similar result). The resulting graphs are very large, and evaluating differences between the different turbulence cases is the subject of ongoing research. However, the example clearly shows that the graph represents the split and merge events for the burning regions.

In the remainder of this section, we focus on evaluating tracking accuracy. Due to the size of the combustion simulations we are considering, we usually do not have every time step for the entire simulation available for analysis. Instead, the simulation code commonly saves only every fifth time step, and it is possible that features move several cells between subsequent saved time steps. Our tracking accuracy study was motivated by the fact that on first examination of the tracking graphs produced by our method, we noticed artifacts that indicated an occasional loss of some of the tracked components.

Fig. 5 shows an example of an artifact that arises due to limited temporal resolution. The first graph in Fig. 5, labeled "coarse," shows the tracking results obtained by using every fifth time step. Instead of showing a single death event for a burning region, the graph shows the death of a burning region and the simultaneous birth of another region that dies in the subsequent time step. The diamond shape of the death and birth events indicates that both occur between two real time steps.

We first assumed that such event sequences occur when our method failed to track a fast-moving component. In an attempt to remedy such errors, we inserted an interpolated time step between every two original simulation time steps. Adding interpolated time steps improved tracking performance in some instances, but not in the example shown here. In the second graph in Fig. 5, labeled "interpolated," the corresponding component dies off completely in the interpolated time step (452.5) and is "reborn" immediately before the next real simulation time step (455) and then immediately dies again. Closer examination of this "false death event" reveals that the burning region is moving several cells in a single time step so that there is no overlap of the cell in two subsequent time steps. Thus, whenever interpolated values are used, either by our classification between time steps in the coarse case, or when classifying the in-between time step in the interpolated case, interpolated values fall below the burning threshold and the region is classified as "extinguished."

For a small subrange of time (time step 431 to 511), we also had access to all time steps of a higher-resolution version of the simulation. We downsampled this higher-resolution data by averaging in space to the same resolution as the original coarse case. With every time step available we used this version as a "gold standard," with the caveat that there may be differences due to performing the simulation at a higher resolution. The bottom graph in Fig. 5 shows the result of using this finer data, labeled "averaged". With all time steps available, the death event is captured properly. The graph also differs for earlier time steps in that an additional burning region splits off and dies. However, the split (time step 441) and subsequent death (time step 444) events occur entirely between time steps available in the coarse case; this event simply is missed by tracking with data at every fifth time step.

We compared tracking results for the three cases: (i) coarse data set with every fifth time step, (ii) coarse data set with interpolated time steps, and (iii) averaged time steps of the finer resolution simulation. During this period approximately 29 burning regions existed in the domain (at the beginning and the end of the time period there were 29 burning regions; in between, the number of burning regions varied). The tracking graphs for 16 of these 29 regions differed between the various analysis approaches.

Approximately 10% of the cases investigated showed inconsistencies between the coarse and fine analyses that could not be attributed directly to the interpolation errors discussed above. In these cases, the source of the discrepancies was traced to the data itself: the coarse and fine values of the threshold quantity did not provide a consistent segmentation, even though the actual solution used for the study was sufficiently grid-independent. The explanation for the discrepancy is related to the numerical integration procedures used to generate the threshold criteria itself. For our study, the combustion solutions were generated using a locally adaptive solution technique that has the property that proportionately small timesteps are used near the highly reactive flame surface, which requires the smallest grid spacing in the domain (other regions in the solution were computed with cells that were 4-8 times larger). The thresholding quantity used for the study is derived from the state of the evolved system: the effective rate of fuel mass destruction over the local time step interval. However, this information is written to disk as snapshots at the larger time intervals related to the coarsest grid cells, so the finer diagnostic is undersampled by a factor of 4-8. Over this larger interval,

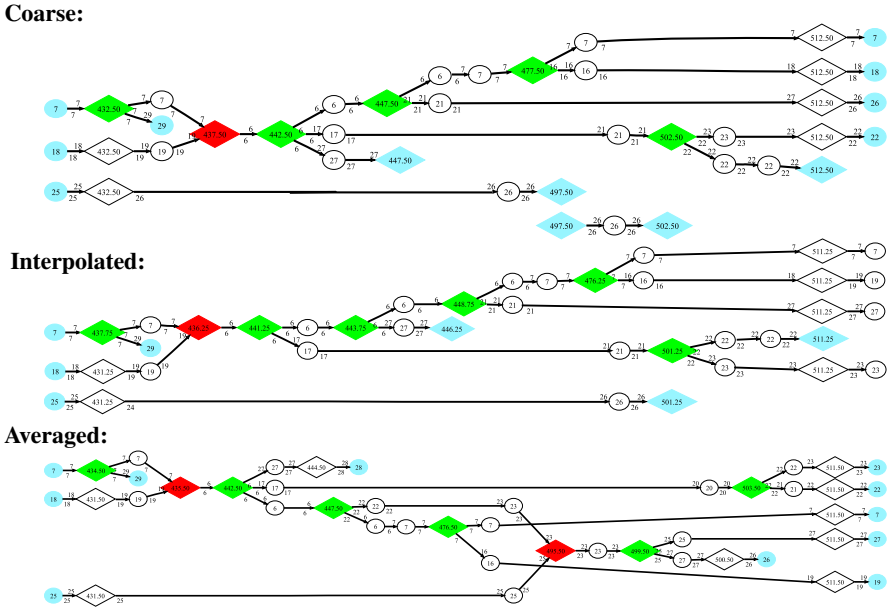


Fig. 6. A more complicated difference in tracking. In the averaged case the traces originating from regions 7 and 18 briefly merge with the trace of region 25. This merge is followed by an immediate split before the next available coarse time step, resulting in coarse and interpolated case missing it. The trace of region 25 is an example where interpolation eliminates a "false death event."

flame features can translate several fine cells and generate substantial sampling errors. Unfortunately, solutions to this problem in our analyses would require a modification to the on-the-fly diagnostics routines in the flow solver, and therefore, is not feasible for the present study. As an aside, we can reasonably expect that thresholds based on evolved quantities, rather than derived ones, would be substantially more robust to this mode of failure.

Three of the 16 tracking differences were due to these differences in the data, with the difference between coarse and fine simulation being mainly the exact time when a merge took place. We verified that these paths and the tracking in the coarse case are correct for the supplied data. Consequently, tracking in the coarse case was correct in 16 cases. Furthermore, the discrepancy in two additional paths is due to two burning regions briefly merging between time steps of the coarse simulation, as shown in Fig. 6. Thus, these events do not appear in the coarse data, and we count only the loss of tracking for region 25 as error. The tracking differences for one region consists of two burning regions splitting off and dying between time steps available in the coarse simulation. (Interestingly, the interpolated case detects one of these split/death events.) Discounting these differences, the coarse data allows correct tracking of 19 out of 29 regions. Most of the other differences involve erroneous merge events in the coarse case between available timesteps, with the general evolution of burning regions being captured satisfactorily.

5 Conclusions and Future Work

We have presented a method to track burning regions in combustion simulations that uses very general, topology-based techniques and that can be adapted for other application areas. Our biggest problem is the lack of temporal resolutions since not all time steps are available for analysis. In some instances, creating interpolated time steps helps to improve tracking accuracy. However, in other cases interpolation aggravates problems due to fast moving burning regions. While improved interpolation techniques, e.g., Lagrangian-based techniques, that take additional velocity information available from the simulation into account, may improve our diagnostic, it is likely that we cannot avoid interpolation related problems completely. One possible solution is to integrate tracking and topological analysis into the simulation code, giving it access to all time steps. Going forward, we will pursue a tighter coupling between simulation and analysis. We further plan to perform tracking of burning regions in a full three-dimensional setting to avoid the issues associated with a sampling surface. For example, extracting an isotherm first adds an additional parameter (value of temperature); it is desirable to eliminate completely the influence of this parameter on analysis results.

References

1. Pankaj K. Agarwal, Herbert Edelsbrunner, John Harer, and Yusu Wang. Extreme elevation on a 2-manifold. *Disc. Comput. Geom.*, 36(4):553–572, 2006.
2. Praveen Bhaniramka, Rephael Wenger, and Roger Crawfis. Isosurface construction in any dimension using convex hulls. *IEEE Trans. Vis. Comp. Graph.*, 10(2):130–141, 2004.
3. Peer-Timo Bremer, Gunther H. Weber, Valerio Pascucci, Marc S. Day, and John B. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 2009. In Press.
4. H. Edelsbrunner and J. Harer. Jacobi sets of multiple Morse functions. In *Found. of Comput. Math., Minneapolis 2002*, pages 37–57. Cambridge Univ. Press, England, 2002.
5. H. Edelsbrunner, J. Harer, A. Mascarenhas, V. Pascuccii, and J. Snoeyink. Time-varying Reeb graphs for continuous space-time data. *Computational Geometry*, 41(3):149–166, 2008.
6. H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Disc. Comput. Geom.*, 28(4):511–533, 2002.
7. G. Ji and H.-W. Shen. Efficient isosurface tracking using precomputed correspondence table. In *Proc. IEEE/Eurographics Symposium Visualization '04*, pages 283–292, 2004.
8. G. Ji, H.-W. Shen, and R. Wegner. Volume tracking using higher dimensional isocontouring. In *Proc. IEEE Visualization '03*, pages 209–216, 2003.
9. E. Koutsofios and S.C. North. Drawing graphs with dot. Technical Report 910904-59113-08TM, AT&T Bell Laboratories, Murray Hill, NJ, 1991.
10. D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Trans. Vis. Comp. Graph.*, 12(5):1052–1060, 2006.
11. W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comp. Graph.*, 21(4):163–169, 1987.
12. Ajith Mascarenhas and Jack Snoeyink. *Isocontour based Visualization of Time-varying Scalar Fields*, pages 41–68. Springer Verlag, 2009. ISBN 978-3-540-25076-0.

13. Gregory M. Nielson. On marching cubes. *IEEE Trans. Vis. Comp. Graph.*, 9(3):341–351, 2003.
14. Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas. Robust on-line computation of Reeb graphs: Simplicity and speed. *ACM Trans. Graph.*, 26(3):58.1–58.9.
15. R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing features and tracking their evolution. *IEEE Computer*, 27(7):20–27, 1994.
16. D. Silver and X. Wang. Tracking and visualizing turbulent 3d features. *IEEE Trans. Vis. Comp. Graph.*, 3(2):129–141, 1997.
17. D. Silver and X. Wang. Tracking scalar features in unstructured datasets. In *Proc. IEEE Visualization '98*, pages 79–86, 1998.
18. Bong-Soo Sohn and Chandrajit Bajaj. Time-varying contour topology. *IEEE Trans. Vis. Comp. Graph.*, 12(1):14–25, 2006.
19. A. Szymczak. Subdomain-aware contour trees and contour tree evolution in time-dependent scalar fields. In *Proc. Shape Modeling International '05*, pages 136–144, 2005.

Complementary Space for Enhanced Uncertainty and Dynamics Visualization

Chandrajit Bajaj, Andrew Gillette, Samrat Goswami, Bong June Kwon, and Jose Rivera

Center for Computational Visualization, University of Texas at Austin, Austin, TX 78712, USA,
http://cvcweb.ices.utexas.edu/ccv/

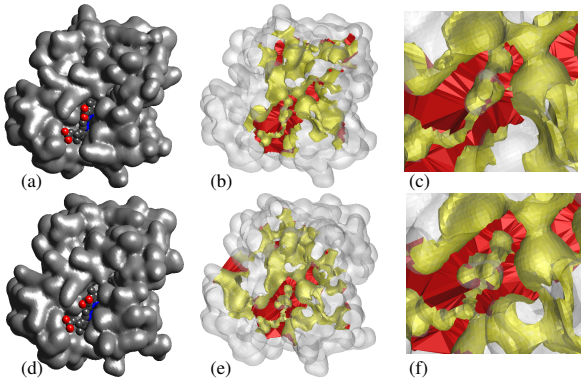


Fig. 1. Visualizations of the hemoglobin molecule undergoing dynamic deformation as oxygen binds to it. **(a)** A primal space visualization of the first time step with the heme group identified. **(b)** Visualization of the complementary space of the first time step shows the geometry of the interior. The surface has been made transparent, revealing a large tunnel through the surface (yellow) with many mouths (red). **(c)** Zooming in on the heme group reveals the structure of space around it while oxygen is bound. **(d-f)** The corresponding images of (a-c) for the final time step. Complementary space has changed dramatically both in the interior volume and near the heme group, though this is not evident from the primal space visualizations. Comparing (c) and (f), we observe that the connectivity of the mouth of the tunnel near the heme group has changed, illustrating the time-dependency of the topological features of complementary space. We quantify and discuss this example further in Section 3.5 (color plate C. 68, page 280).

1 Introduction

Many computational modeling pipelines for geometry processing and visualization focus on topologically and geometrically accurate shape reconstruction of "primal" space, meaning the surface of interest and the volume it contains. Certain features of a surface such as pockets, tunnels, and voids (small, closed components) often represent important properties of the model and yet are difficult to detect or visualize in a model of primal space alone. It is natural, then, to consider what information can be gained from

a model and visualization of complementary space, i.e. the space exterior to but still "near" the surface in question. In this paper, we show how complementary space can be used as a tool for both uncertainty and dynamics visualizations and analysis.

Uncertainty visualizations aim to elucidate the accuracy of a model by visibly identifying and subsequently quantifying potential errors in the model. For surface and volume models, drawing attention to topological errors is of particular importance as they represent a more fundamental inaccuracy in shape than geometrical errors. Topological errors include the presence of an unwanted tunnel in a surface, the absence of a desired tunnel, and the existence of small extraneous components. We also consider errors related to the existence or absence of "pockets" in a surface as topological errors; pockets on a primal space surface correspond to components in complementary space and the number of components of a space is a basic topological property. As we will discuss, it is important to determine if topologically distinct models occur within the inherent uncertainty bounds of a model and complementary space provides a natural means for visualizing uncertainty in topological structure.

Dynamics visualizations bring shape models to life by showing conformational changes the model may undergo in the application context. In this case, there is uncertainty not only in the particularities of the shape at any time step of the dynamics, but also in the plausibility of the simulated movement as a whole. The creation of a tunnel, the collapse of a pocket into a void, or the changing geometry of the interior of a tunnel may be highly relevant to understanding a dynamic situation and assessing its likelihood of simulating reality. By visualizing and as necessary quantifying complementary space at each stage, we better understand how our models need to be refined. In Figure 1 and the included videos, we show an example of a dynamical situation where complementary space visualization and quantification aids in understanding.

The specific problems we address in this paper touch on a variety of situations where modeling is sensitive to topological errors in primal or complementary space. In each case, we discuss how complementary space modeling provides a natural method for processing, visualizing, and managing such errors. Our methods are inspired by challenges encountered in our work with biological models and hence most of our illustrated examples use actual biological data from public sources and our academic collaborators. Nevertheless, the computational techniques we describe here are useful in a variety of settings including creation of exploded assembly images and videos, quantification of complementary space features in CAD models, and visualization of potential errors in any image-based modeling scheme.

We conclude this section with a brief review of related work in complementary space modeling. In Section 2 we explain the relevant theory behind our two approaches to complementary space modeling and fix notation. In Section 3 we describe particular approaches to complementary space solutions we have implemented in our lab and show some results. In Section 4 we conclude and discuss future work.

Related Work We discuss three approaches to complementary space modeling: alpha shapes, surface propagation, and Morse theory for distance functions.

The notion of alpha shapes was developed by Edelsbrunner et al. [10] in the context of molecular modeling with the aim of pocket detection. The method uses the Delaunay diagram on the atomic centers of a molecule and decomposes space into Delaunay tetra-

hedra near the molecule and into unbounded regions away from it. By assigning a flow across Delaunay faces based on the radius of nearby atoms, the authors distinguish between those finite tetrahedra belonging to the alpha shape (i.e. the molecule) and those belonging to a pocket. This method has been implemented and applied to a number of proteins with some success [16]. In instances of broad pocket mouths, however, it is difficult for this method to detect the pocket and its coarse treatment of the molecular model leaves geometrical refinement to be desired. It is also unclear how this approach could be applied to non-molecular models.

The method of surface propagation represents a surface implicitly as the zero level set of a distance function. The surface is subjected to an evolution equation first introduced by Osher and Sethian [18]. To turn this from an analytical theory to a computable implementation, Sethian developed a fast level set marching method [19] for propagating the given surface in the outward normal direction. Zhang and Bajaj [22] have used this method to propagate a molecular surface outward until the surface has the topology of a sphere and then back inward for an equal length of time. Pockets are then defined as those points exterior to the initial surface and interior to the final surface. This method has been implemented in our publicly available software package TexMol [7].

A more general approach to complementary space modeling uses the Morse complex for distance functions. The Morse complex canonically decomposes a space relative to its features, as identified by the critical points of a chosen function. Cazals, Chazal and Lewiner have computed the Morse complex for the Connolly function on 2-manifolds [4] which has shown some promise as a primal space method for shape analysis. Natarajan and Pascucci have used the Morse complex for aid in visualization of cryo-EM data [17]. The Morse complex for a distance function can be approximated by inducing a flow on the Voronoi diagram of a point sample of the surface, as was characterized by Edelsbrunner [9] and Giesen and John [12]. The distance function has been used in a variety of applications, including image feature identification [5, 21], stable medial axis construction [6], object segmentation and matching [8], annotation of flat and tubular features [13], and detection of secondary structural motifs in proteins [2]. In previous work, we have shown how the distance function can also be used to detect tunnels and pockets [1] for the purpose of surface curation. In this paper, we show how these techniques can be used for the much broader purpose of uncertainty and dynamics visualizations.

2 Background and Notation

We have implemented two complementary space modeling techniques: an out-and-back surface propagation method and a Morse complex based method. We describe each in detail below and compare them at the end of this section.

For out-and-back surface propagation, we begin with a meshed geometry of the initial surface Σ and set M to be a large, contractible, compact subset of $\mathbb{R}^3$ containing Σ (e.g. a filled bounding box). We define the distance function h_Σ by

$$h_\Sigma : M \to \mathbb{R}, \quad x \mapsto \pm \inf_{p \in \Sigma} ||x - p|| \tag{1}$$

where the sign of the output is determined by the location of the input x relative to Σ (inside or outside). We represent Σ implicitly as the zero level set $\{x \in \mathbb{R}^3 : h_\Sigma(x) = 0\}$. The evolution equation for the surface is given by

$$\begin{cases} \phi_t + F|\nabla\phi| = 0, \\ \phi(x,0) = h_\Sigma(x). \end{cases}$$

Here, F is a speed function in the normal direction, usually chosen to depend on the curvature. To define pockets, we use this method to propagate a molecular surface outward with constant speed ($F \equiv 1$) until some time t when the surface has the topology of a sphere. This surface is then propagated with speed $F \equiv -1$ also for time t, creating a final surface Σ'. Pockets are then defined as those points exterior to Σ and interior to Σ'. This method has been implemented in our publicly available software package TexMol [7].

The second approach we use for complementary space modeling employs the tools of Morse theory on the distance function h_Σ. We define

$$h_P : M \to \mathbb{R}, \quad x \mapsto \pm \min_{p \in P} ||x - p||,$$

where Σ is the Delaunay diagram of a point sample P. The function h_P is easy to compute as opposed to h_Σ which has no closed form in the general case. The critical points of h_P correspond to the intersection of Voronoi objects of Vor P with their dual Delaunay objects in Del P.

This explicit description of the critical point structure of h_P allows for the creation of a quickly computed discrete approximation of the Morse complex. The complex is used to construct a geometry of complementary space by an algorithm we call COMPSPACE. We described the details of this approach in detail in a previous issue of this series [1] and do not repeat them here. Once we have the geometry output by this method, we classify components with one mouth as pockets and components with two or more mouths as tunnels. We can then tag the mouth, pocket interior, and tunnel interior simplicies different colors for visualization or compute their areas and volumes for quantification.

Comparison of the Methods We have implemented both methods described above and find that they are each useful in different circumstances. The surface propagation method is ideal for detecting wide, shallow depressions on surfaces as the method detects regions of inward curvature. In molecular modeling, such depressions occur at the active site for the binding of of a protein; in neuronal cell modeling, such depressions occur at synapses, the intracellular region between two adjacent cells across which a voltage signal is passed. The Morse complex method excels in detecting deeper pockets and distinguishing tunnels from pockets as the Morse complex is laden with connectivity information. We next look at a number of such examples using this approach.

3 Applications

Complementary space visualization can be used for model checking, error analysis, detection of topologically uncertain regions, topological preservation in model reduction, and dynamic deformation visualization, as we outline in the following subsections.

3.1 Ion channel models

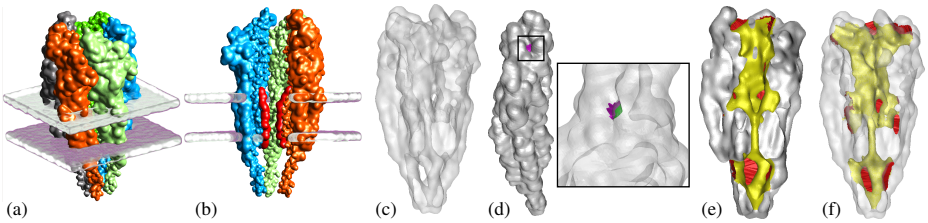


Fig. 2. Visualizations of the acetylcholine receptor molecule. **(a)** The molecule is shown as it would sit embedded in a bilipid cell membrane (grey) with the five identical subunit colored for identification. **(b)** A cut-away view of the same model showing where ions may pass through the center. **(c)** A transparent view of the molecular surface. **(d)** Each subunit contains a pocket where acetylcholine binds. A complementary space view of the pocket interior (green) and its mouth (purple) are shown in a zoomed in view after the surface has been made transparent. **(e)** A cut-away view of the surface with the interior of the tunnel (yellow) and its mouths (red) identified. **(f)** The same view as (c) with the tunnel geometry opaque, showing how it lies inside the surface. Visualizing the complementary space tunnel reveals that the dimensions of the pore opening on the extracellular side are much larger than on the intracellular side. This is less evident from the primal space visualizations (color plate C. 69, page 280).

Ion channels are a cell's mechanism for regulating the flow of ions into and out of the cell. They usually have two main structural confirmations: the 'open' configuration, in which the tunnel through its center is wide enough to allow passage of the ions, and a 'closed' configuration in which it is not. We look at the acetylcholine receptor (PDB ID 2BG9) as a particular example of an ion channel. This molecule is embedded in a cell membrane, as shown in Figure 2a, and is a control mechanism for the flow of sodium and potassium ions into the cell. It is made up of five homologous (in the biological sense) subunits. A conformational change from closed to open occurs when acetylcholine, a small neurotransmitter ligand, docks into the five small pockets on the exterior of the molecule near the tunnel opening in the extracellular region. When acetylcholine fills one of these pockets, it causes the attached chain subunit of 2BG9 to twist slightly. The combined effect from rotations in all five chains is a widening of the mouth of the tunnel, akin to the opening of a shutter on a camera.

From this description of the action of the acetylcholine receptor, the importance of accurate complementary space topology becomes evident. First, an accurate model of the channel must feature a tunnel passing completely through the length of the surface. Put differently, the complementary space should include a connected component running the length of the molecule with mouths at opposite ends. Such a requirement can be quickly verified by a complementary space visualization as shown in Figure 2. Furthermore, the diameter of this tunnel at its narrowest point should be within the range of biological feasibility, i.e. it should be wide enough to accommodate sodium and potassium ions in the open confirmation and narrow enough to block them in the closed confirmation.

To quantify properties of the tunnel, we use the geometries output by COMPSPACE. The output of COMPSPACE is a mesh of the tunnel's interior surface, closed off by its mouths. We compute this mesh for two models of the molecule - one in its 'open' state and one in its 'closed' state. The enclosed volume in the open state is 86,657 $Å^3$ and 60,045 $Å^3$ in the closed state. The mouth area in the open state is 5716 $Å^2$ and 3290 $Å^2$ in the closed state. The minimum diameter in the open state is 5.9 Å and 8.0 Å in the closed state. As expected, all the measurements - minimum tunnel diameter, mouth area, and enclosed volume - are all smaller in the closed state than in the open state. We note that the change in the minimum diameter from the closed to open state is reasonable for accommodating ions whose width is a few angstroms. We will augment this model in the future by incorporating electrostatic calculations of ion attraction and repulsion forces to further explain the gate-like ability of the channel.

An additional consideration for this model is its geometry at the ligand binding site. In terms of complementary space, we expect a small pocket on each of the subunits such that its volume and mouth diameter are of plausible size compared to the acetylcholine molecule. We show a visualization of the pocket in one subunit in Figure 2 d. While such features are difficult to visualize and measure with a model based primal space, they much easier to detect and manipulate with a model based on complementary space.

3.2 Ribosome models

The ribosome molecule provides another example of natural structural questions best answered with a complementary space model. Ribosomes live inside cells and are the construction equipment for proteins made within the cell. Proteins are assembled in a large tunnel that passes through the ribosome. A copy of DNA data called mRNA is fed through the tunnel in steps. At each step, the portion of the mRNA in the tunnel dictates which type of amino acid is allowed to enter the tunnel and bind to the nascent chain. As the chain gets longer and eventually terminates, it folds into the protein coded for by the mRNA. The ribosome molecule is itself composed of two main subunits - 50s and 30s - which come together to form the tunnel where the proteins are assembled. We show the ribosome in both primal and complementary space views in Figure 3. As we begin to measure the complementary space model, we will be able to provide evidence for or against various hypotheses about the protein construction process such as whether there is enough room inside the tunnel for proteins to begin folding.

Complementary space also aids in answering the question of how the ribosome comes into its assembled state. The larger subunit alone (PDB ID 1FFK) is made up of a long, coiled RNA strand, a short RNA strand, and dozens of proteins various types, as shown in Figures 3a-c. While all the proteins involved can and have been identified and labelled, the order in which they come together to form the subunit is unknown as video capture techniques do not exist for the requisite nanometer-resolution scale.

We have created a video of a plausible assembly sequence and analyze its accuracy via a complementary space method as follows. The PDB entry 1PNY provides atom locations for the assembled ribosome molecule with tags identifying those atoms belonging to the various docked proteins. We separate the atoms according to their tags and create meshed surface representations of each of the proteins and the coiled RNA

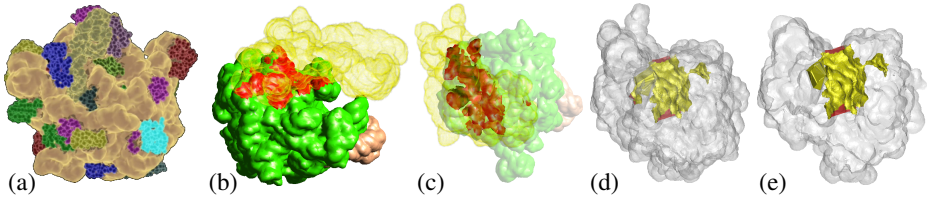


Fig. 3. Visualizations of the ribosome molecule. **(a)** The ribosome itself is made up of three RNA chains (brown) and dozens of proteins (various colors). We define the contact area of each protein to be any portion of its surface lying within 4 Å of an RNA chain. We compute these areas and use them to predict the order in which the proteins bind to the RNA chains. **(b)** Two of the RNA chains (green and brown) belong to the 50s subunit while the third (yellow) belongs to the 30s subunit. We compute the contact area between these chains (red) to show how the subunits come together to form the protein assembly tunnel. **(c)** A different view of the RNA chains gives a better view of the contact region but makes it difficult to see where the tunnel lies. **(d)** We run CompSpace on a surface model that includes all the attached proteins and visualize the surface (transparent) along with the tunnel mouths (red) and interior (yellow). **(e)** A cut-away view helps elucidate the intricate geometry surrounding the protein assembly region. These types of complementary space visualizations and subsequent quantifications provide insight to open questions such as whether amino acid chains have room to begin folding before they exit the ribosome (color plate C. 70, page 281).

strands; the resulting geometries are thus still fixed in space according to the PDB location information. For each protein geometry, we search for vertices lying within four angstroms of the RNA geometry using the distance function h_P. Triangles incident to these vertices are tagged as part of the "contact region."

We sum the areas of the triangles in the contact region for each of the 32 proteins. We found that the contact regions with RNA vary greatly in size - from 324 $Å^2$ to 7385 $Å^2$. The proteins with larger contact areas bind to RNA first while the ones with smaller areas bind later as their access to RNA is partially blocked by other proteins.

3.3 Virus models

Viruses rely heavily on their geometry to infect cells and replicate their genome. Since the goal of a virus is rapid reproduction, many viruses are made of identical subunits forming a highly symmetrical capsid shell, thereby minimizing the number of unique parts that must be synthesized. Accordingly, an accurate model of a virus should have the same symmetries as the virus itself and complementary space can aid in detecting such symmetries.

The example we consider here is the nodavirus which infects certain types of freshwater fish. A simple volume rendering is shown in Figure 4a with the symmetry evident. Selecting an isosurface from the range of possible values, however, presents a challenge as noise in the data often upsets the symmetry as is seen in Figure 4b. The problem in this case is not the presence of a small tunnel but the absence of one, both at the indicated area and elsewhere. We therefore need a new type of complementary

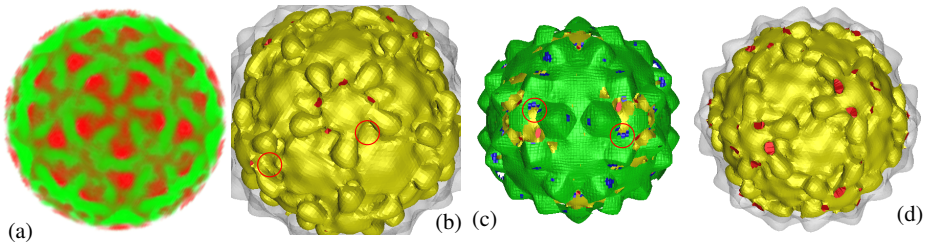


Fig. 4. Identification of "thin" regions in the primal space for the nodavirus dataset. **(a)** A volume rendering of the 3D image data. **(b)** Tunnels are detected for the initial selection of the isosurface. Note that in some places of 5-fold symmetry, only 4 mouths of the tunnel are present. **(c)** The thin regions (blue) are identified as subsets of the unstable manifolds of the index 1 saddles identified on the interior medial axis. The circles (red) in (b) and (c) indicate that places where the fifth mouth of the tunnel should be open indeed have thin regions. **(d)** The final selected isosurface has complementary space topology consistent with the inherent symmetry of the 3D density map (color plate C. 71, page 281).

space visualization indicating "thin regions" where the surface comes close to a self intersection; these regions are candidates for a missing tunnel.

Remarkably, the distance function h_P plays an important role here also. The idea is to compute the interior medial axis of the surface and detect those portions lying very close to the surface, i.e. where h_P is below some threshold γ. For surfaces derived from image data, γ should be set to the resolution of the imaging device as any smaller size features are probably unreliable and should be eliminated. In practice, the medial axis is often noisy resulting in many erroneous thin regions, so we use instead two subsets of it (U_1 and U_2, described below) which are stable against small undulations on the surface. The method works as follows.

1. We first approximate the interior medial axis using the Voronoi and Delaunay diagrams already computed for complementary space modeling. The method for this is described in a previous paper [14].
2. Collect the point sets

$$C_{1,IM} = \{\text{index 1 saddles of } h_P \text{ on int. med. axis of } \Sigma\}$$
$$C_{2,IM} = \{\text{index 2 saddles of } h_P \text{ on int. med. axis of } \Sigma\}$$

3. Compute the unstable manifolds of each of point in these two sets. This results in a piecewise planar subset U_1 of the medial axis for points from $C_{1,IM}$ and a piecewise linear subset U_2 for points from $C_{2,IM}$.
4. At each Voronoi vertex lying on U_1 or U_2, compute the value of h_P. This is given by the circumradius of the Delaunay tetrahedra dual to the Voronoi vertex. If h_P is smaller than γ, mark the region as "thin" and color differently for visualization.
5. Collect the interior maxima falling into the thin subsets of U_1 and U_2 and compute their stable manifolds. The stable manifold creates a geometry for the missing tunnel.

Figure 4 (c) shows the thin regions (blue patches) on the U_1 stable manifold (green) identified for the nodavirus model. These can be selectively replaced by tunnels to capture the correct symmetry. Alternatively, the presence of thin regions suggests a different isovalue may be more appropriate, such as the one shown in Figure 4 (d).

3.4 Topological Consistency of Reduced Models

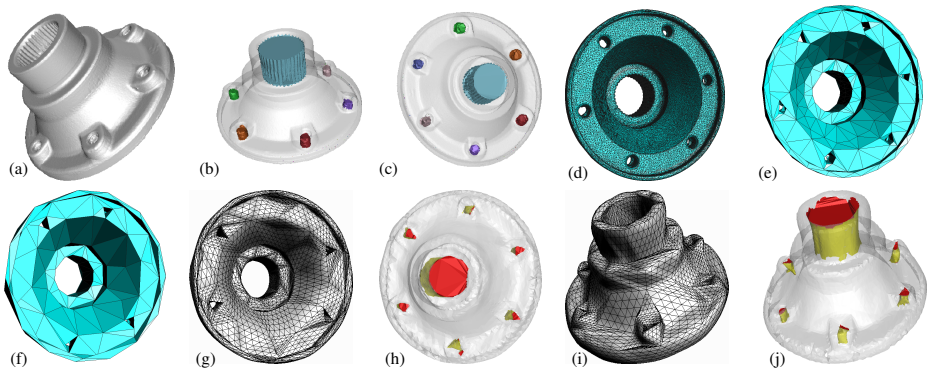


Fig. 5. Visualizations of the Carter dataset. **(a)** A basic primal space visualization of the mechanical part. **(b-c)** Complementary space features identified and visualized. **(d)** A visualization of the dense mesh representing the surface reveals that at 106,708 triangles, it is probably amenable to decimation. **(e-f)** Using QSlim [11], the mesh is decimated to 1000 and then to 500 triangles. **(g-j)** After refinement and geometric improvement on the 500 triangle model, it appears from the primal space visualizations (g and i) that some of the tunnels have collapsed, a topological change. Complementary space visualizations (h and j) reveal that in fact the tunnels are still present but with perturbed geometry. Therefore, this reduced model has consistent topology with the original (color plate C. 72, page 282).

Model reduction or decimation is the process of removing geometrical information from a model while attempting to keep sufficient data for maintenance of important features. This is used, for example, in coarse-grained models of proteins for electrostatic simulations [3]. Protein surfaces are often defined based on atomic positions and radii, obtained from the PDB. For large proteins, a significant speed-up in computational time can be achieved by grouping atoms into clusters and treating the clusters as single atoms with an averaged radius. Model reduction is also common for point-sampled surfaces such as CAD models and geometries acquired from three-dimensional scanners. If points on the surface can be culled without dramatic effect on the shape of the surface, subsequent visualization and simulation pipelines will have reduced computational cost.

Complementary space visualization aids in determining when, if ever, the original topology is lost in progressive decimation. Consider the industrial part model shown in Figure 5. We use the software QSlim [11] to decimate the model from 106,708 triangles to only 1000 and then only 500. At 1000 triangles, the model has lost some geometrical

precision, but still appears to have the same number of tunnels (5 e). At 500 triangles, however, some of the tunnels appear to have collapsed (5 f). We improve the geometry by refinement and smoothing but still cannot tell from primal space visualizations if the topology is correct (5 g and i). Complementary space visualizations, however, quickly show that all tunnels are indeed present, albeit somewhat distorted (5 h and j). From a topological standpoint, therefore, this reduction is consistent; the application context will determine if it is acceptable from a geometrical standpoint as well.

3.5 Dynamic Deformation Visualization

Complementary space aids in visualizing and quantifying dynamic deformations of models in addition to its aid for static models previously discussed. The omnipresent consideration in a computer generated simulation of real movement is always whether the dynamics are realistically plausible. In the context of molecular modeling, such considerations are especially difficult to formalize as current video technology cannot capture a molecule in vivo for comparison. As a result, various techniques have been developed for automated animation of molecular models, including the popular method of Normal Mode Analysis (NMA) [15, 20].

To determine whether the fluctuations simulated by these means have any functional significance to the molecule, we must be able to measure the extent of changes in particular features of the surface. This is especially important in molecules which perform specific actions by modifying their complementary space features, such as the ribosome. With a model of complementary space, we can measure the area of the mouth of a tunnel or pocket used in the various processes and compare the sizes before and after a conformational change. This gives insight into the relative magnitude of different aspects of the shape reconfiguration; a seemingly significant deformation may only involve a small change in the size of a pocket mouth or vice versa.

To demonstrate the benefits of complementary space dynamics visualization, we consider the hemoglobin molecule. Hemoglobin is the vehicle used to transport oxygen through the bloodstream. A single hemoglobin molecule is made of four subunits, each of which can hold one oxygen molecule at its heme group. This conformational change has been simulated by interpolation of PDB data for the bound and unbound states by our collaborators Drs. David Goodsell and Arthur Olson of the Molecular Graphics Laboratory at the Scripps Research Institute. Using this sequence of time steps, we have generated videos of primal and complementary space dynamics. We compare the visual differences in primal and complementary space further in Figure 1. Interestingly, the complementary space has dynamically changing geometry, both in the region of the active site and deeper in the interior of the molecule.

We also quantify this data by measuring the volume of the two main complementary space features of the molecule detected by COMPSPACE: a tunnel and a pocket. For each feature, we compute the enclosed volume (in Å^3) and total mouth surface area (in Å^2). We show the results in Figure 6. The tunnel data shows a dynamic change in enclosed volume over the time scale, with a max of 1963 and a min of 1498 Å^3. The total mouth area also varies widely, with a max of 679 and a min of 190 Å^2. The pocket data exhibits similar fluctuations.

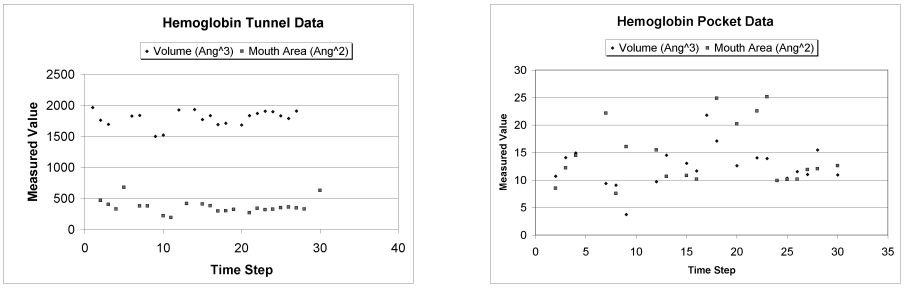


Fig. 6. Quantification of the complementary space tunnel in the hemoglobin time step data. The undulating nature of the two data series reflects the dynamic deformation hemoglobin undergoes while binding to oxygen.

4 Conclusion

Visualization of the complementary space of a geometrical model has the immediate impact of elucidating pockets, tunnels, and other subtle structural features. We have shown in this paper that an explicit geometry of complementary space is often essential to allow for the measurement of certain aspects of the model such as tunnel mouth area or pocket volume. These quantities can characterize the feasibility of models in biology or the precision of CAD-based models. As we have discussed with the hemoglobin example, complementary space also plays a useful role in creating and analyzing dynamic visualizations. Finally, as seen in the case of ribosome assembly, complementary space measurements can also guide the creation of primal space dynamics visualizations.

It is in this last vein of questions regarding assembly pathways that we intend to expand this work. To identify likely assembly paths, we construct a graph whose nodes are the assembly parts in question; in the case of the ribosome, these are the RNA chain and the individual proteins which bind to it. Edges exist between parts which are adjacent and edge weights are given by contact surface area. Assembly order is based on relative affinity between parts and affinity is a function of contact area. Hence, an assembly path relates closely to a maximally weighted spanning tree of the nodes. We plan to elucidate this further in future work.

ACKNOWLEDGMENTS We would like to thank previous members of the CCV lab who worked on these projects, including Katherine Claridge, Vinay Siddavanahalli, and Bong-Soo Sohn. This research was supported in part by NSF grants DMS-0636643, CNS-0540033 and NIH contracts R01-EB00487, R01-GM074258, R01-GM07308.

References

1. C. Bajaj, A. Gillette, and S. Goswami. Topology based selection and curation of level sets. In H.-C. Hege, K. Polthier, and G. Scheuermann, editors, *Topology-Based Methods in Visualization II*, pages 45–58. Springer-Verlag, 2009.

2. C. Bajaj and S. Goswami. Automatic fold and structural motif elucidation from 3d EM maps of macromolecules. In *ICVGIP 2006*, pages 264–275, 2006.
3. N. Basdevant, D. Borgis, and T. Ha-Duong. A coarse-grained protein-protein potential derived from an all-atom force field. *Journal of Physical Chemistry B*, 111(31):9390–9399, 2007.
4. F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the morse-smale complex and the connolly function. In *19th Ann. ACM Sympos. Comp. Geom.*, pages 351–360, 2003.
5. R. Chaine. A geometric convection approach of 3D reconstruction. In *Proc. Eurographics Sympos. on Geometry Processing*, pages 218–229, 2003.
6. F. Chazal and A. Lieutier. Stability and homotopy of a subset of the medial axis. In *Proc. 9th ACM Sympos. Solid Modeling and Applications*, pages 243–248, 2004.
7. CVC. TexMol. *http://ccvweb.csres.utexas.edu/ccv/projects/project.php?proID=8*.
8. T. K. Dey, J. Giesen, and S. Goswami. Shape segmentation and matching with flow discretization. In F. Dehne, J.-R. Sack, and M. Smid, editors, *Proc. Workshop Algorithms Data Strucutres (WADS 03)*, LNCS 2748, pages 25–36, Berlin, Germany, 2003.
9. H. Edelsbrunner. Surface reconstruction by wrapping finite point sets in space. In B. Aronov, S. Basu, J. Pach, and M. Sharir, editors, *Ricky Pollack and Eli Goodman Festschrift*, pages 379–404. Springer-Verlag, 2002.
10. H. Edelsbrunner, M. Facello, and J. Liang. On the definition and the construction of pockets in macromolecules. *Discrete Applied Mathematics*, 88:83–102, 1998.
11. M. Garland. QSlim. *http://graphics.cs.uiuc.edu/ garland/software/qslim.html*, 2004.
12. J. Giesen and M. John. The flow complex: a data structure for geometric modeling. In *Proc. 14th ACM-SIAM Sympos. Discrete Algorithms*, pages 285–294, 2003.
13. S. Goswami, T. K. Dey, and C. L. Bajaj. Identifying flat and tubular regions of a shape by unstable manifolds. In *Proc. 11th ACM Sympos. Solid and Phys. Modeling*, pages 27–37, 2006.
14. S. Goswami, A. Gillette, and C. Bajaj. Efficient Delaunay mesh generation from sampled scalar functions. In *Proceedings of the 16th International Meshing Roundtable*, pages 495–511. Springer-Verlag, October 2007.
15. M. Levitt, C. Sander, and P. S. Stern. Protein normal-mode dynamics: Trypsin inhibitor, crambin, ribonuclease and lysozyme. *Journal of Molecular Biology*, 181:423 – 447, 1985.
16. J. Liang, H. Edelsbrunner, and C. Woodward. Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design. *Protein Sci*, 7(9):1884–97, 1998.
17. V. Natarajan and V. Pascucci. Volumetric data analysis using morse-smale complexes. In *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 322–327, Washington, DC, USA, 2005.
18. S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.
19. J. A. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proc. Nat. Acad. Sci*, pages 1591–1595, 1996.
20. F. Tama. Normal mode analysis with simplified models to investigate the global dynamics of biological systems. *Protein and Peptide Letters*, 10(2):119 – 132, 2003.
21. Z. Yu and C. Bajaj. Detecting circular and rectangular particles based on geometric feature detection in electron micrographs. *Journal of Structural Biology*, 145:168–180, 2004.
22. X. Zhang and C. Bajaj. Extraction, visualization and quantification of protein pockets. In *Comp. Syst. Bioinf. CSM2007*, volume 6, pages 275–286, 2007.

Color plates

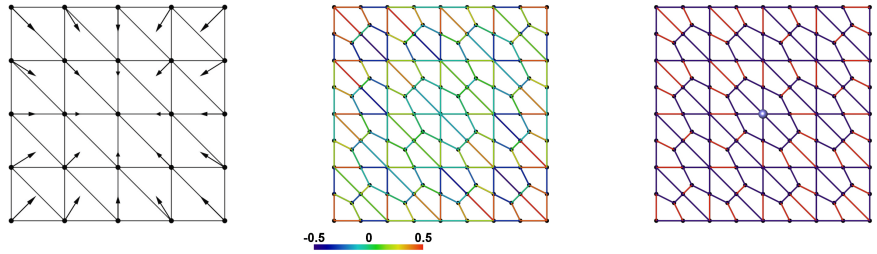


Fig. C. 1. Algorithmic pipeline, from left to right: a) input triangulation with vectors on vertices; b) link weighted simplicial graph; c) computed combinatorial vector field (red) with a sink in the center (blue). Figure 2, page 5.

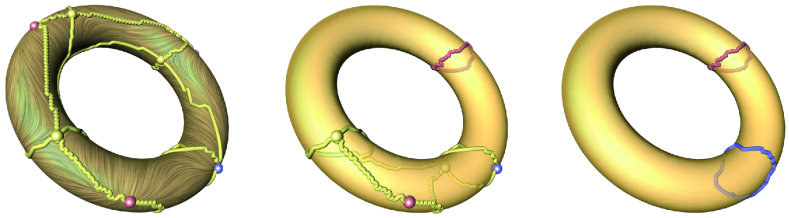


Fig. C. 2. Random vector field on a torus with three levels of topological simplification (figure 5, page 9).

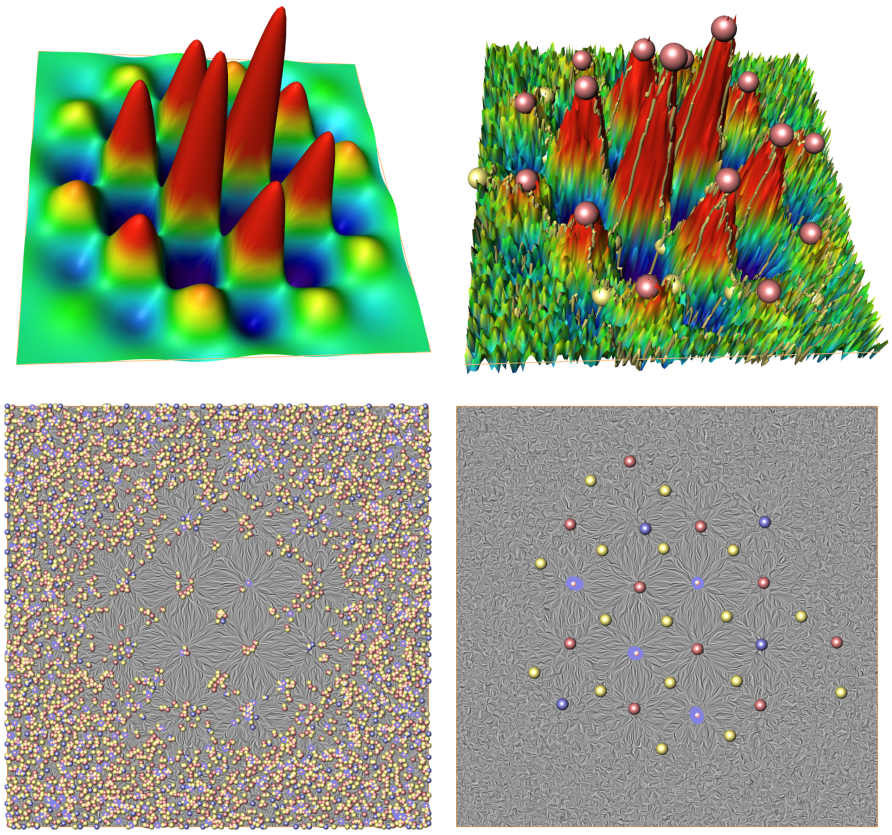


Fig. C. 3. Top-left: height field of unperturbed data set; top-right: height field of perturbed data set with simplified topological skeleton; bottom-left: sources (red), sinks (blue), and saddles (yellow) of initial combinatorial vector field on planar LIC of the gradient vector field; bottom-right: critical points of simplified topological skeleton - the blue circles represent tiny attracting periodic orbits (figure 3, page 7).

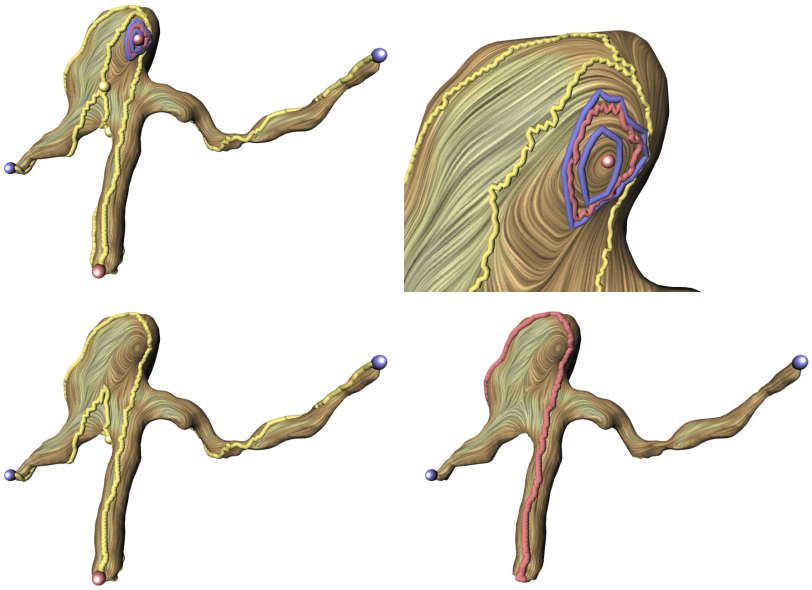


Fig. C. 4. Top-left: topological skeleton of a vector field on a cerebral aneurysm given by a blood flow simulation; top-right: zoom in of an area exhibiting recurrent flow behavior indicated by attracting periodic orbits (red) and repelling periodic orbits (blue); bottom-left: simplified topological skeleton with 1 saddle, 2 sinks and 1 source; bottom-right: simplified topological skeleton with 2 sinks and 1 repelling orbit (figure 6, page 10).

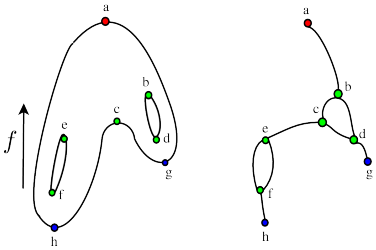


Fig. C. 5. Left: A two-holed 2-manifold and the height function defined on it. Points in blue, green, and red correspond to minima, saddle, and maxima of the function, respectively. **Right:** The Reeb graph of the height function. Loops in the Reeb graph correspond to holes in the manifold.

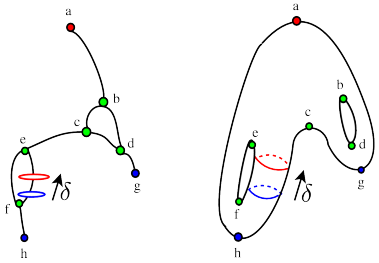


Fig. C. 6. Offsetting a level set component. **Left:** Level set components on the manifold. **Right:** Offsetting a level set component (blue) to another component (red) along an edge of the Reeb graph (figure 2, page 15).

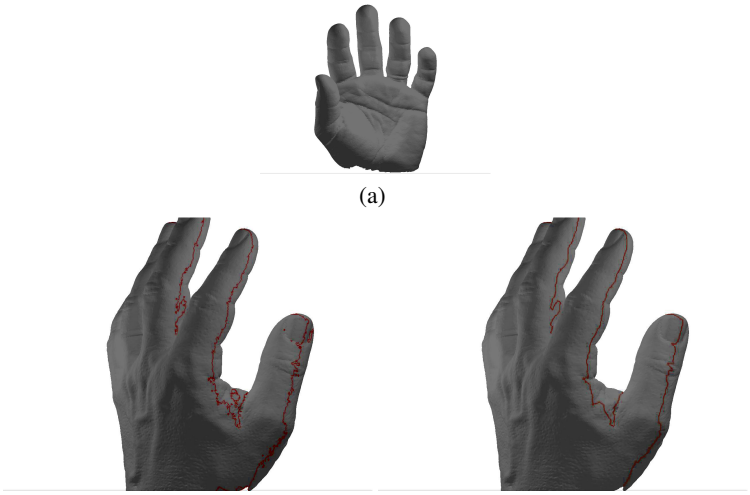


Fig. C. 7. Silhouettes. (a) Model of a hand in its original orientation. (b) Silhouette when viewed from a different angle. (c) Simplified silhouette (figure 8, page 22).

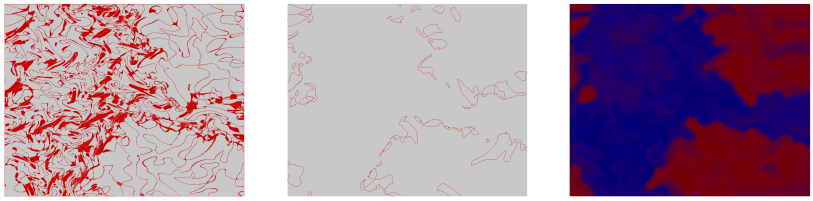


Fig. C. 8. Combustion. (a) Jacobi set of H_2 and O_2 in the 64th time step. (b) Simplified Jacobi set. (c) Concentration of O_2 (figure 9, page 23).

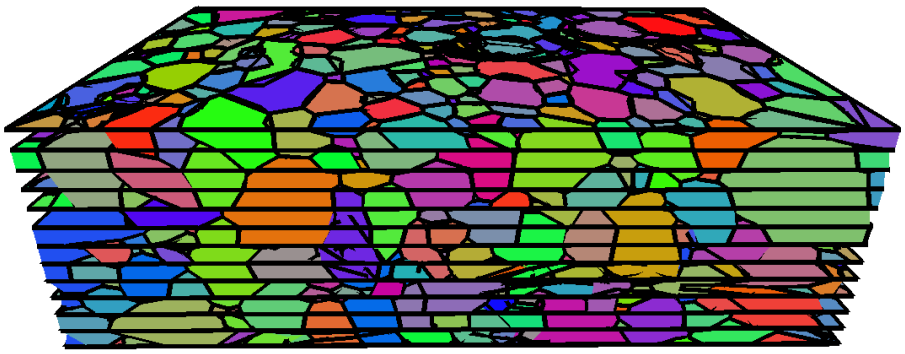


Fig. C. 9. A portion of a reconstruction of 13 cross-sections of a sample of shocked tantalum. The input cross-sections are indicated by horizontal lines (figure 9, page 35).

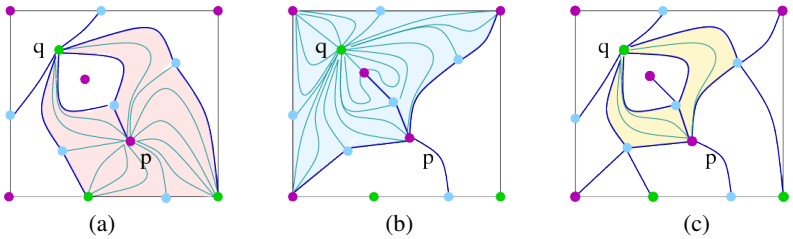


Fig. C. 10. (a) A 2D ascending Morse complex. The ascending cell of minimum p is shaded. (b) A dual descending Morse complex. The descending cell of maximum q is shaded. (c) Morse-Smale complex. 2-cells related to minimum p and maximum q are shaded (figure 1, page 39).

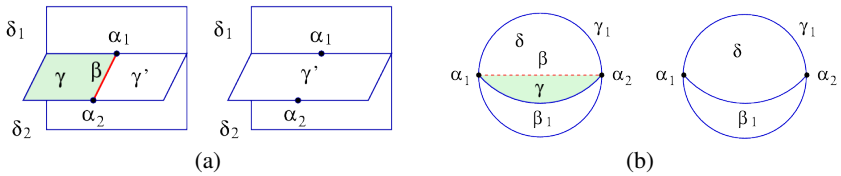


Fig. C. 11. Part of a 3D descending Morse complex before and after a removal (a) $rem(\gamma,\beta,\gamma')$, and (b) $rem(\gamma,\beta,\emptyset)$ (figure 2, page 42).

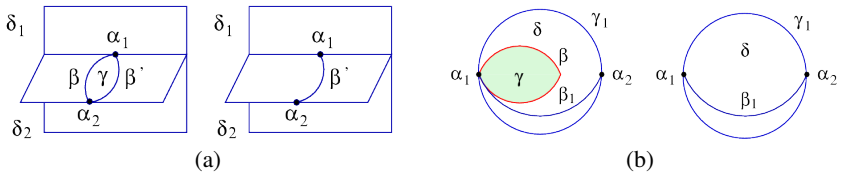
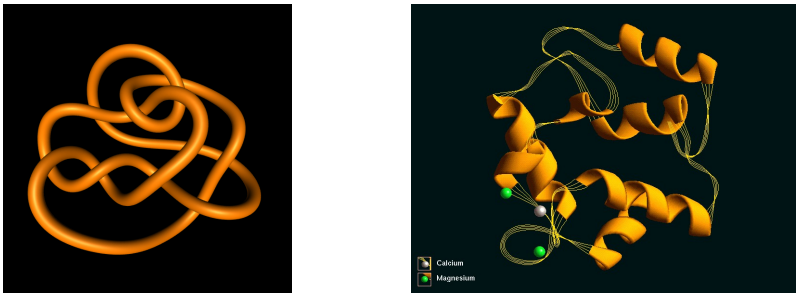


Fig. C. 12. Part of a 3D descending Morse complex before and after a contraction $con(\beta,\gamma,\beta')$ (a), and $con(\beta,\gamma,\emptyset)$ (b) (figure 3, page 42).



(a) (b)

Fig. C. 13. (a) Complicated Unknot (b) Protein-enzyme complex (figure 1, page 49).

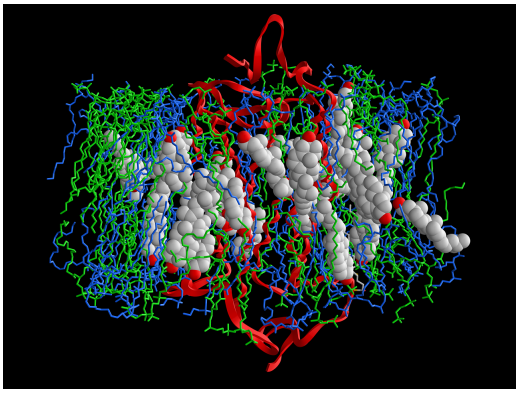


Fig. C. 14. Dense molecular configuration (figure 5, page 59).

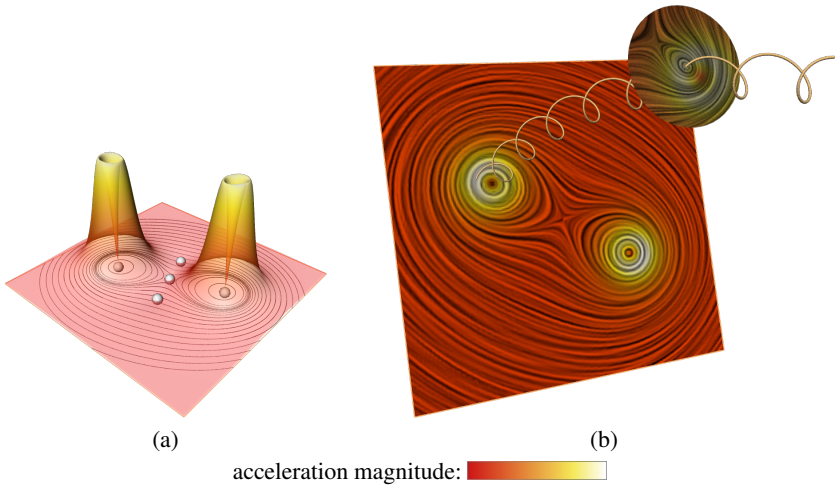


Fig. C. 15. Visualization of two co-rotating Oseen vortices in two dimensions (a) The acceleration magnitude of one time-slice is shown as a color-coded heightfield. The extracted points are the acceleration minima – which include fixed points. (b) The pathline of a particle seeded in one time-slice is displayed. The time is depicted as third dimension. The relative behavior of the flow field in the neighborhood of this particle at a later point in time is displayed on a surrounding circular disk. The acceleration magnitude is color coded in the line integral convolution (LIC) images of the flow field (figure 2, page 80).

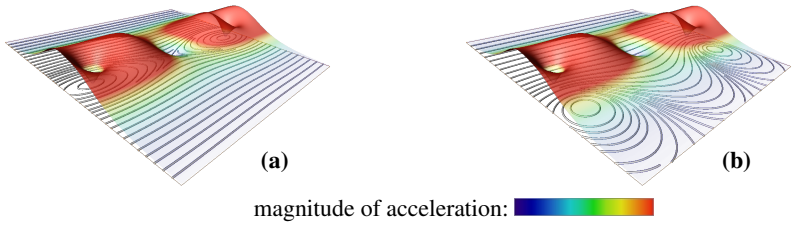


Fig. C. 16. Stuart vortices in two inertial frames of references: (a) vortices at rest, (b) convecting vortices such that the velocity at the bottom becomes zero. Heightfield and color represent the absolute value of the particle acceleration. The acceleration field is the same in both cases, while the streamline patterns, observed from different inertial systems, differ (figure 3, page 81).

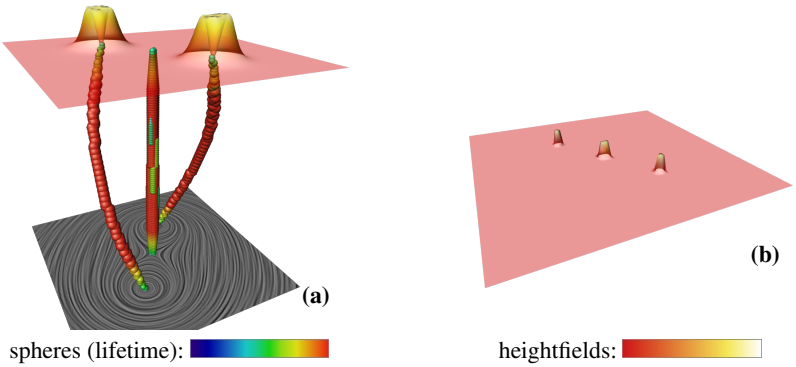


Fig. C. 17. Illustration of the *Lagrangian equilibrium point* concept for two co-rotating Oseen vortices. The color-coded heightfield represents: (a) integrated acceleration, (b) and lifetime for the last time step (figure 5, page 83).

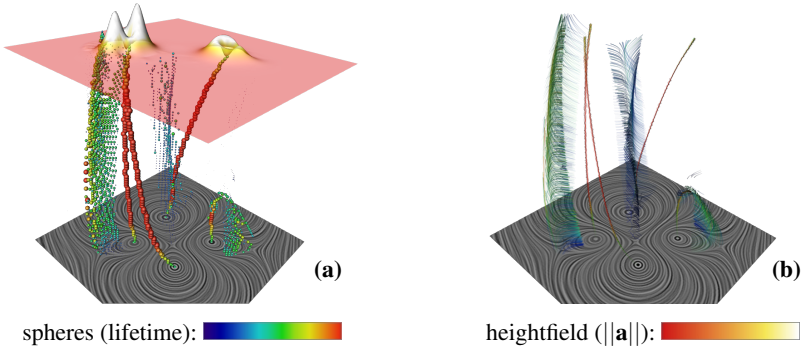


Fig. C. 18. (a) Extraction of features for a motion of six Oseen vortices. Since particles leave the saddle regions quickly, the saddles do not emerge as prominently as the vortex cores. The integration windows of 0.3 in each direction are too large for particles passing through the saddle in the center. In comparison, all other long-living structures such as the vortex cores are extracted effectively. The heightfield represents the integrated acceleration. (b) Visualization of the feature lifetime. The illuminated pathline segments show the interval of the lifetime used for the integration of the acceleration. Pathlines seeded in vortex-like feature points are long centerlines, while pathlines seeded in saddle-like feature points diverge rapidly (figure 6, page 84).

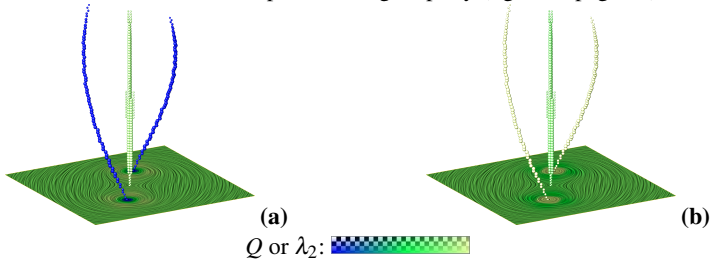


Fig. C. 19. Comparison of (a) λ_2 and (b) Q with the proposed Lagrangian equilibrium points. The extracted feature points include the vortex cores marked by high values of Q or low values of λ_2. In addition the saddle between the two vortices is detected (figure 7, page 85).

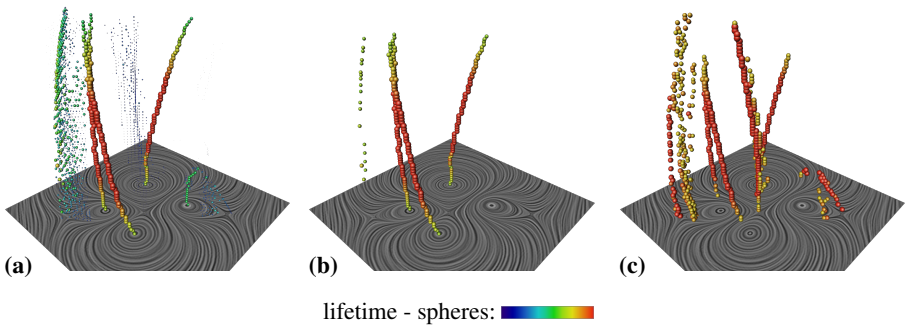


Fig. C. 20. Motion of six Oseen vortices: (a) All features before filtering; (b) Applying the life-time filter filters the short-living out. (c) Employing a shorter life-time window also saddles are extracted (figure 8, page 86).

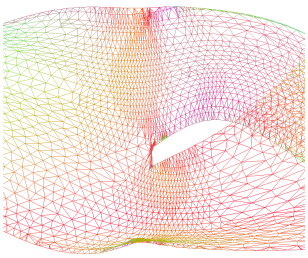


Fig. C. 21. Obstacle splitting a stream surface. Resulting fronts are advanced separately. The mesh is colored according to normal directions (figure 5, page 95).

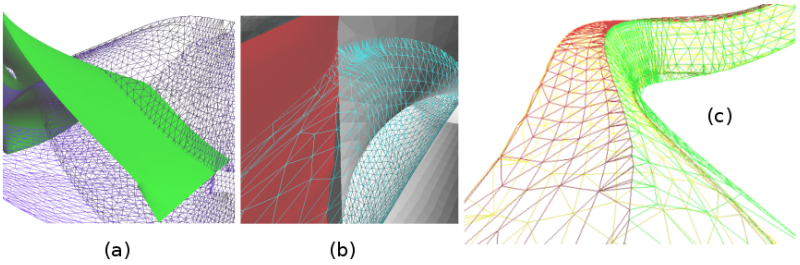


Fig. C. 22. Crossing behavior of stream surfaces (a), and t-intersection (b) between outer stream line and red surface. Stream volume composed of parts of three separatrices (c). Retriangulation during surface splitting guarantees a closed volume (figure 6, page 96).

Fig. C. 23. Visualization of several stream volumes. A stretching, rotating and diverging motion between adjacent volumes can be observed (figure 8, page 99).

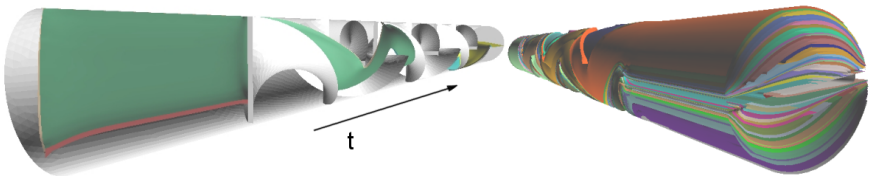


Fig. C. 24. Visualizations of a single twisted and stretched volume and all volumes of the data set. Colors identify adjacent stream volumes. Occlusion of volumes hides important information about the mixing process, which can be visualized by slicing (figure 9, page 99).

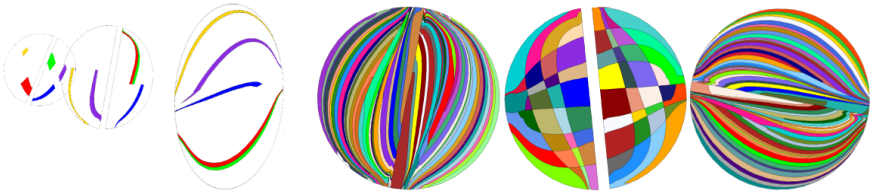


Fig. C. 25. Slicing planes through five (left) and all volumes of the data set (right) at positions $t = 0.3, 0.6, 1.0, 0, 0.5, 1.0$. It is clearly visible, how neighboring volumes take different paths through the data set. Comparison between individual slices allows observation of stretching, rotational, and mixing behavior. The distribution of volume colors indicate a good mixing process (figure 10, page 99).

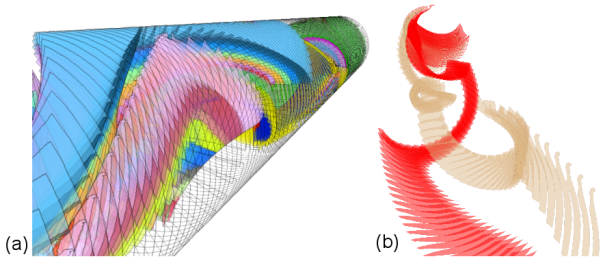


Fig. C. 26. Volume visualization of a selection of stream volumes by slicing planes (a) and volume visualization of two stream volumes (b). This type of visualization lessens the effect of volume occlusion by the use of transparency (figure 11, page 99).

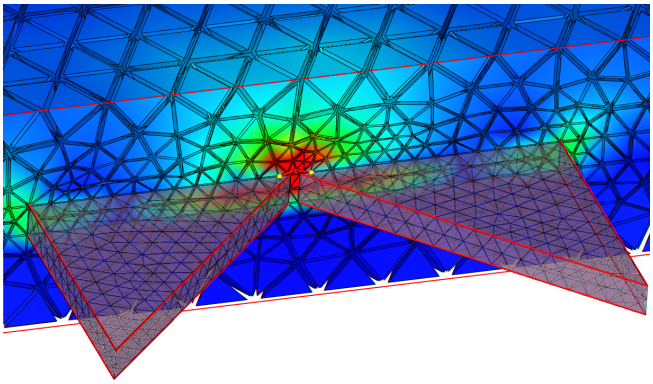


Fig. C. 27. Electromagnetism simulation for furtivity studies on a large irregular tetrahedral mesh where surface and linear substructures are the heart of the simulation as they influence the field propagation (figure 1, page 102).

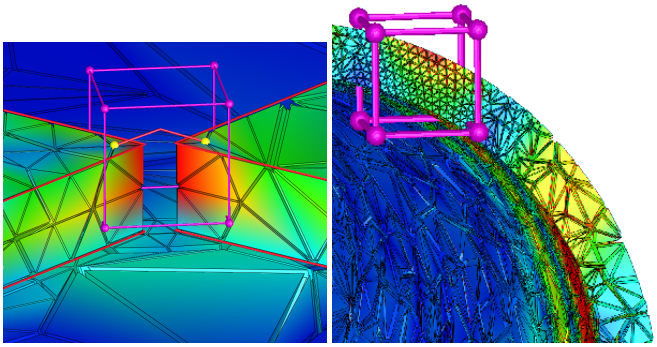


Fig. C. 28. $(left)$ Detail of a substructure shown at high resolution only inside a volume of interest. The topology of the mesh and the substructures is preserved everywhere. $(right)$ Multiresolution visualization of a thin layer of material with the associated electromagnetic field magnitude (figure 1, page 180).

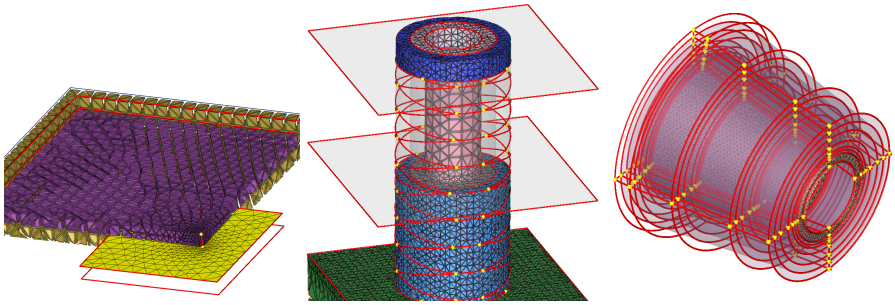
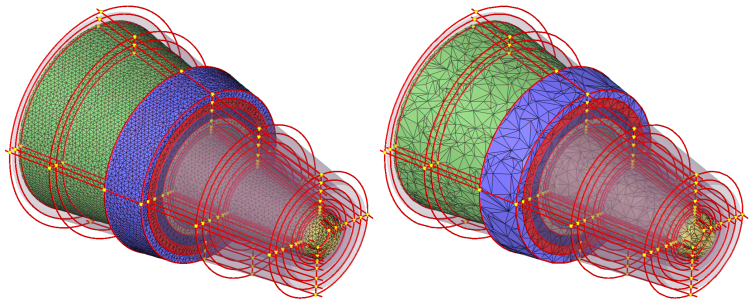
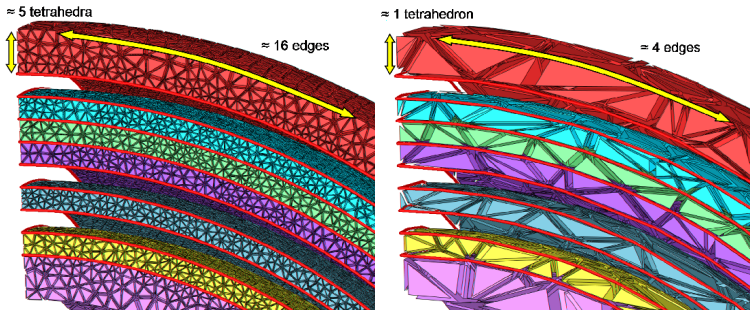


Fig. C. 29. $(left)$ Polyline emerging from an orthogonal base plan. The mesh is subdivided around this structure. $(center)$ Linear structure with multiple self intersections twisted around a cylindrical mass with orthogonal interfaces. $(right)$ Multiple crossing interfaces of thin material layers and polylines of the structure (figure 3, page 104).



(a)Original mesh (b)After 90% of simplification

Fig. C. 30. Volumetric simplification of a model with multiple substructures. A few material layers are shown as opaque triangle mesh and all interfaces as semi-transparent surfaces. The linear features are in red and their self intersections marked by yellow shperes (figure 6, page 110).



(a)Original mesh (b)After 98% of simplification

Fig. C. 31. Aggressive simplification of a volume mesh composed of thin material layers. The topological criteria preserve both the thickness of the layers and the topology of the substructures while allowing collapses of edges on the substructures (figure 7, page 110).

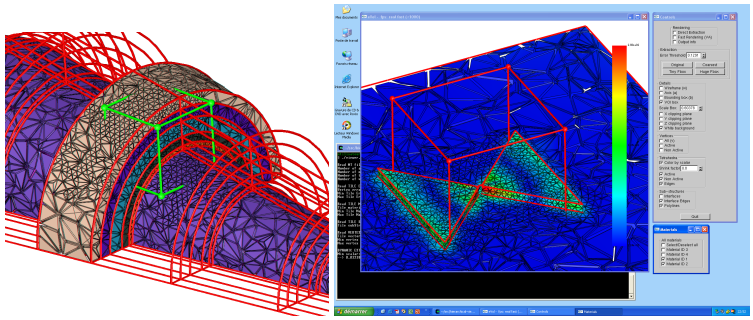


Fig. C. 32. $(left)$ Variable resolution visualization of a volume mesh with multiple linear features. The topology of the substructures is guaranteed to be preserved. $(right)$ Snapshot of the multiresolution visualization tool to explore simulation data with embedded structures on a desktop PC (figure 8, page 111).

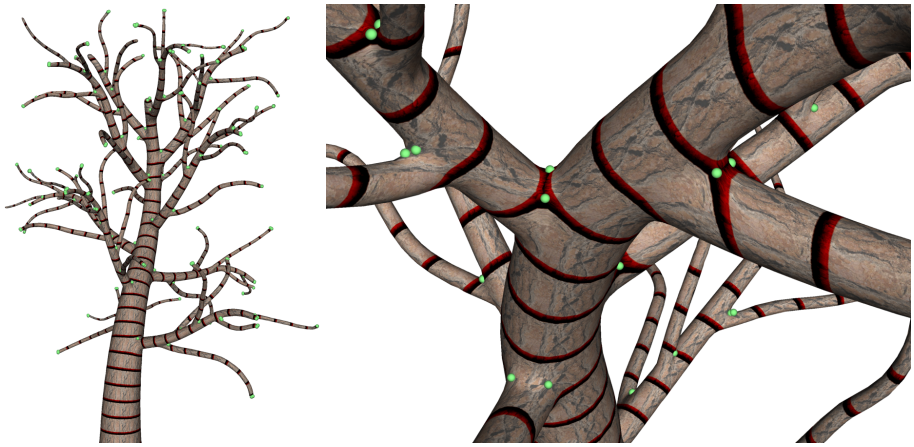


Fig. C. 33. Tree with stripe parameterization. Singularities are marked in green. The texture image consists of a vertical stripe visualizing the u-isolines of the parameterization (figure 1, page 115).

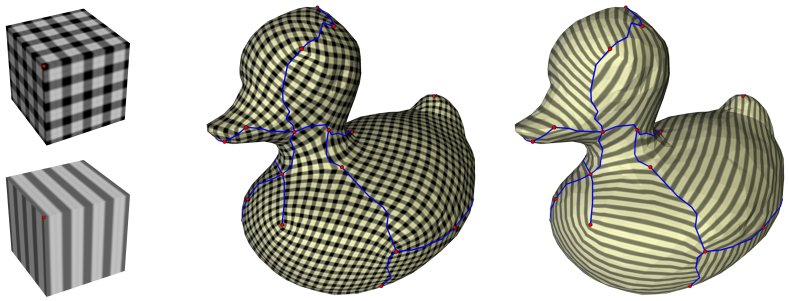


Fig. C. 34. QuadCover parameterization with quad texture and stripe texture. Stripe textures can not simply be used in parameterizations from QuadCover (figure 2, page 116).

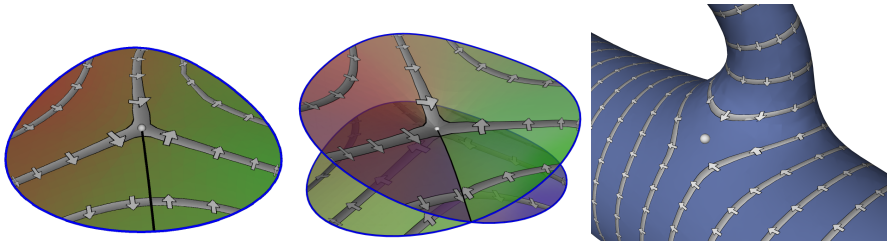


Fig. C. 35. Left: Stripe parameterization with branch point. The isolines of the u function and its gradient vectors are drawn. Middle: The same parameter function on the 2-sheeted covering (the covering surface is not embedded, it has self-intersections). Right: Branch point on a parameterized tube object (figure 4, page 118).

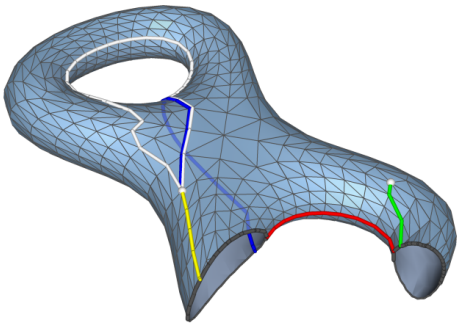


Fig. C. 36. Surface with boundary and two branch points. The colored lines visualize the five cut paths (figure 5, page 121).

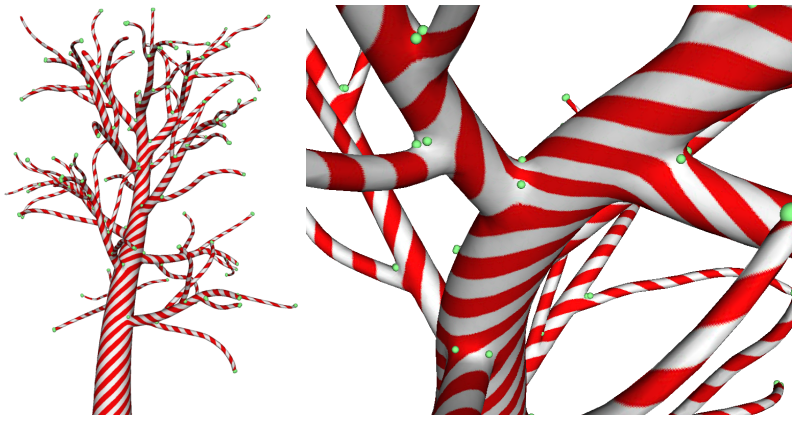


Fig. C. 37. The tree model of Fig. 1 with diagonal stripe pattern generates a candy cane. Singularities are marked in green (figure 8, page 123).

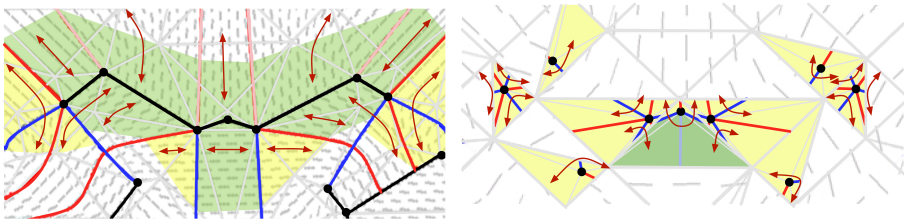


Fig. C. 38. A close-up of sector classification for the one-point load data-set using linear interpolation of eigenvectors, with (left) and without (right) subsequent re-triangulation. Shaded regions show the sectors: green and yellow for non-hyperbolic and hyperbolic, respectively; red and blue lines show radial lines, which are not integrated; black points and lines are the degenerate points and lines (figure 5, page 134).

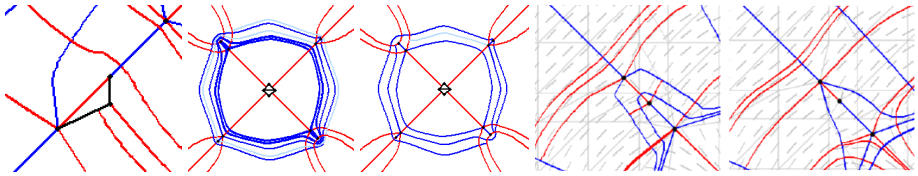


Fig. C. 39. Close-up from one-point load data-set: (a) tensorline runs into a degenerate line (black line); circulating tensorline (a) before and (b) after clean up; (d,e) comparison of separatrix integration for component-wise and eigenvector-based interpolation (figure 6, page 135).

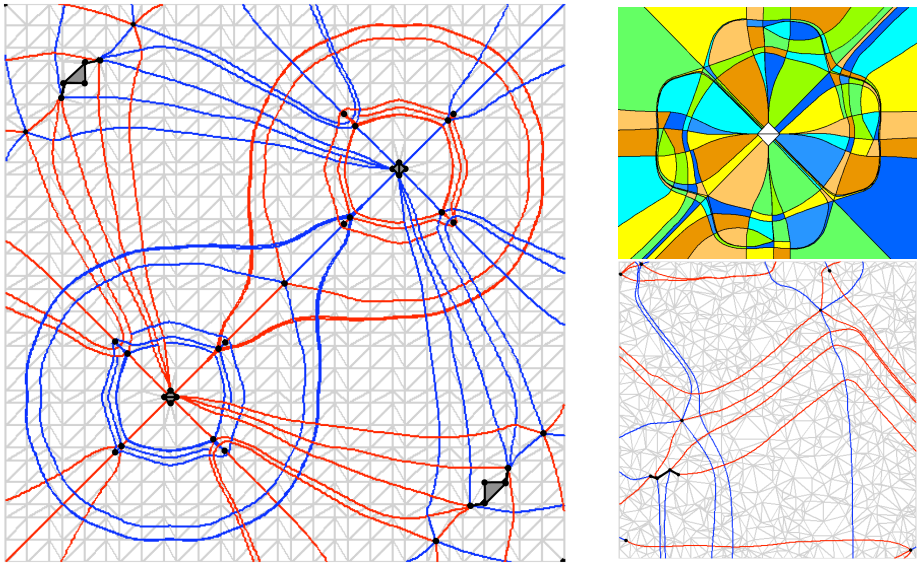


Fig. C. 40. Full segmentation: left: two-point load, right top: one-point load with randomly colored segments, right bottom: slice of strain simulation of forces on notched block (figure 7, page 137).

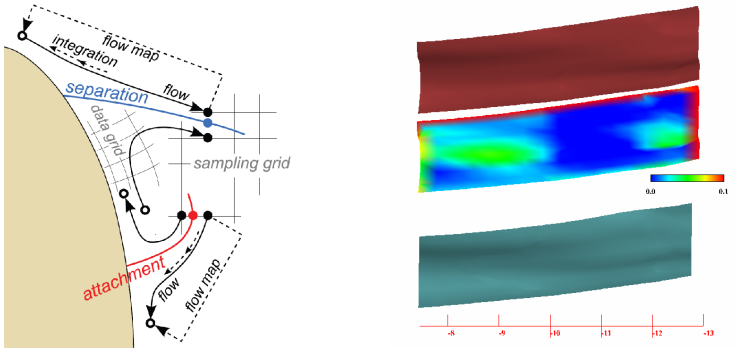
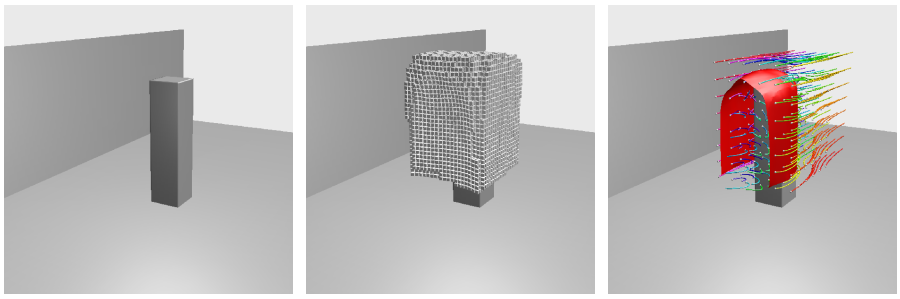
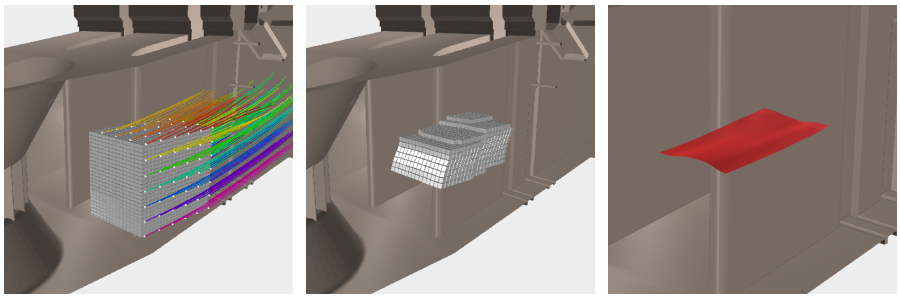


Fig. C. 41. (a) Flow separation and flow attachment. The unstable manifold (blue) attracts the fluid along the boundary and guides it into the interior of the domain whereas the stable manifold (red) guides the fluid in opposite direction. (b) Intake dataset. Comparison of ridge from advected grid (red) and uniform grid (blue) together with ridge from advected grid color-coded by distance to ridge from uniform grid (colored) (figure 1, page 143).



(a) (b) (c)

Fig. C. 42. Flow around a cuboid. (a) Geometry. (b) Sampling grid adapted to ridge region and advected. (c) Resulting FTLE ridge with some upstream trajectories (colored) from uniform grid, and their seeds (white spheres) (figure 2, page 148).



(a) (b) (c)

Fig. C. 43. Intake of a water turbine. (a) Uniform grid with some of the upstream trajectories (colored) used for FTLE computation, and their seeds (white spheres). (b) Sampling grid adapted to ridge region and advected. (c) Resulting FTLE ridge (figure 3, page 149).

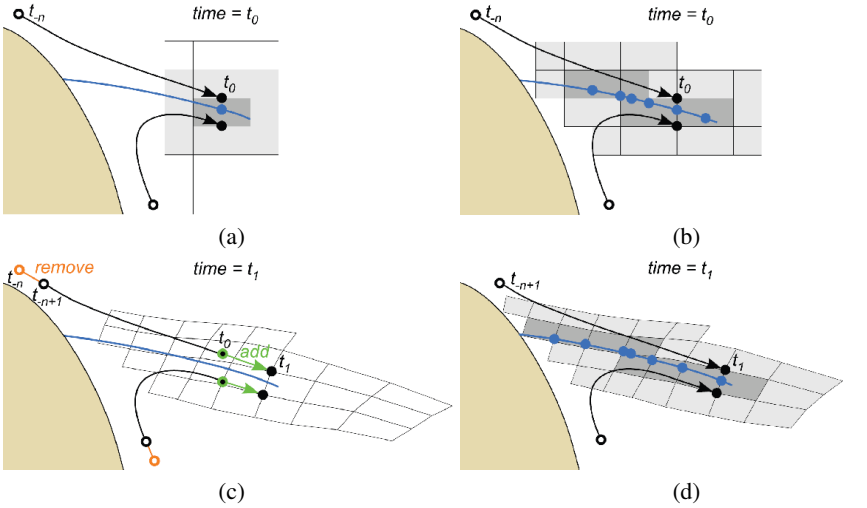


Fig. C. 44. Grid advection for flow separation. (a) Initial sampling grid. Ridge cells (dark gray) and their neighboring cells (light gray). Cell edge intersected by negative-time FTLE ridge (blue point). Neighborhood range is 1 for illustration purposes. (b) After grid adaptation. (c) After one step of grid advection. (d) After grid adaptation of advected grid (figure 4, page 153).

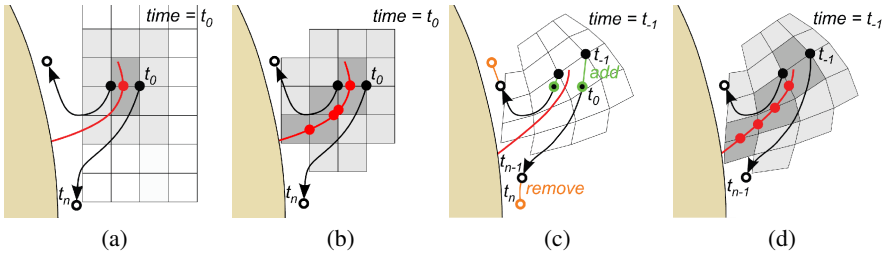


Fig. C. 45. Grid advection for flow attachment. (a) Initial sampling grid. Ridge cells (dark gray) and their neighboring cells (light gray). Cell edge intersected by positive-time FTLE ridge (red point). Neighborhood range is 1 for illustration purposes. (b) After grid adaptation. (c) After one step of grid advection. (d) After grid adaptation of advected grid (figure 5, page 153).

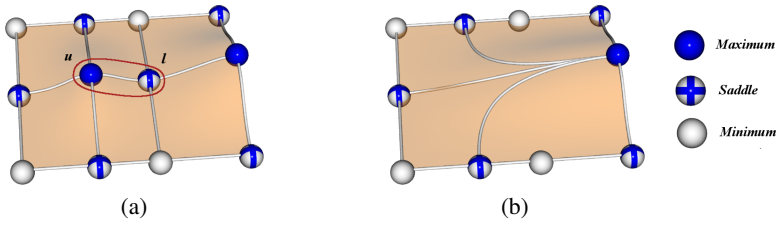
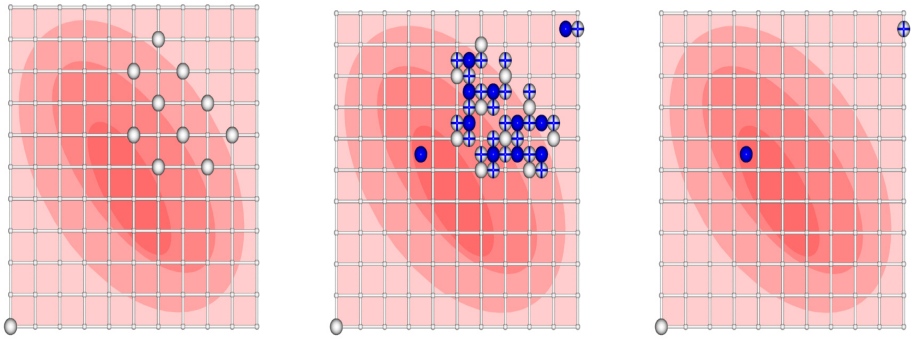
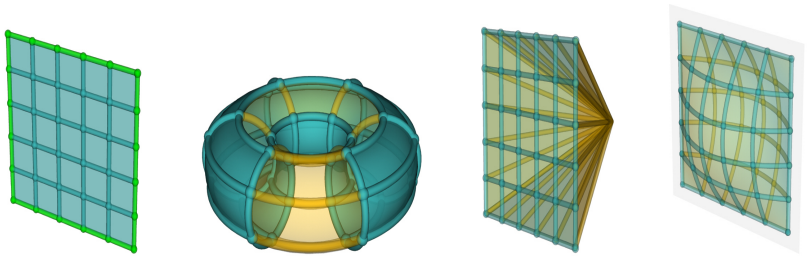


Fig. C. 46. The circled arc connects a saddle l to a maximum u (a). cancellation of (l,u) removes all arcs attached to l or u, and creates new arcs from the lower neighbors of u to the upper neighbors of l (b) (figure 1, page 157).



(a) (b) (c)

Fig. C. 47. Let the *index* of a vertex $v = (x, y)$ be $x + (y * xdim)$. Using standard simulation of simplicity, minima are identified at the boundaries of flat regions (a). When constructing an MS complex, these minima further generate the necessary separating saddles and maxima (b), far more than the actual three persistent critical points (c) (figure 2, page 160).



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

Fig. C. 48. A 2-manifold with interior (blue) and boundary (green) cells (a). Periodic boundary conditions (b) attach the boundary on opposing sides. One-point compactification (c) attaches the boundary to a point with infinite persistence. Mirrored boundary (d) duplicates the interior and attaches it along the boundary (figure 4, page 165).

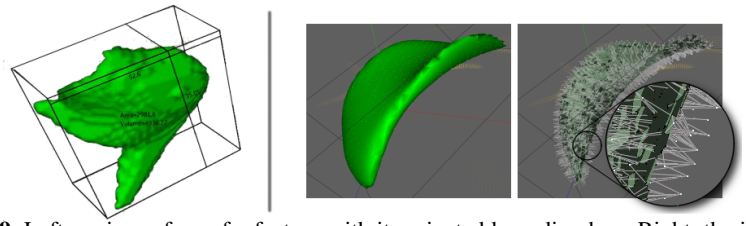


Fig. C. 49. Left, an isosurface of a feature with its oriented bounding box. Right, the isosurface of a feature and its medial axis. The inset zooms a portion showing the medial axis vertices and lines connecting them to the closest point on the isosurface (figure 5, page 174).

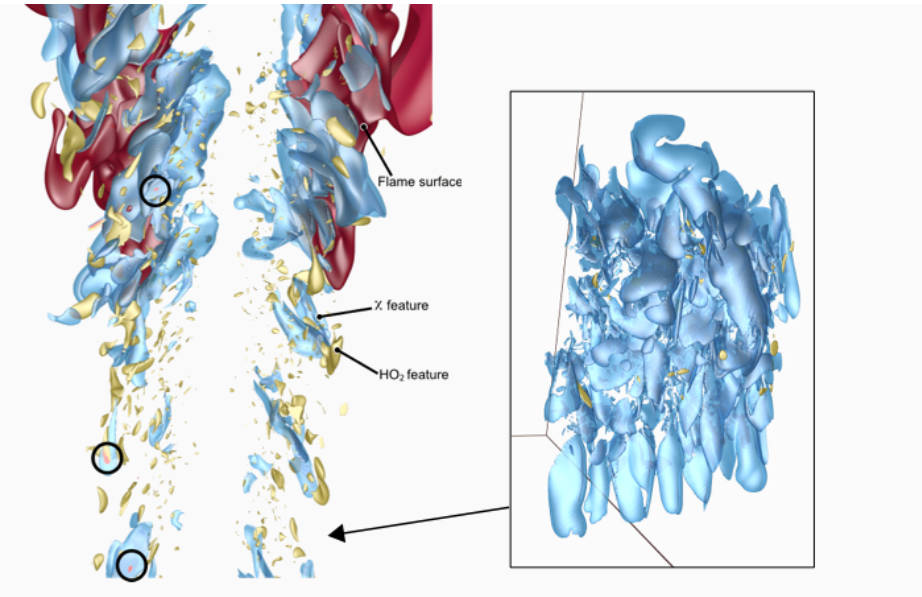


Fig. C. 50. Relationship between χ and HO_2 fields. Left, all HO_2 kernels below the base of the flame with χ segments that overlap them. Right, a close-up side view showing all χ segments and HO_2 kernels (figure 8, page 176).

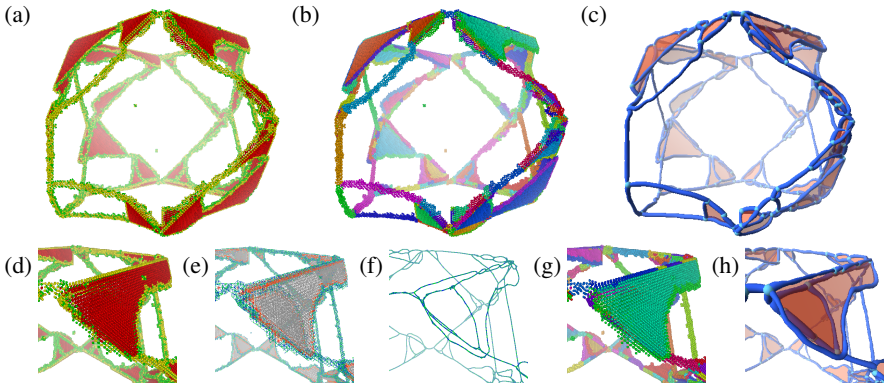


Fig. C. 51. Upper row: (a) original atom data, (b) after segmentation, (c) extracted defects. Lower row: Detail views - (d) original atom data, (e) relative neighborhood graph, (f) graph after simplification, (g) segmented atom data, (h) extracted defects (figure 1, page 180).

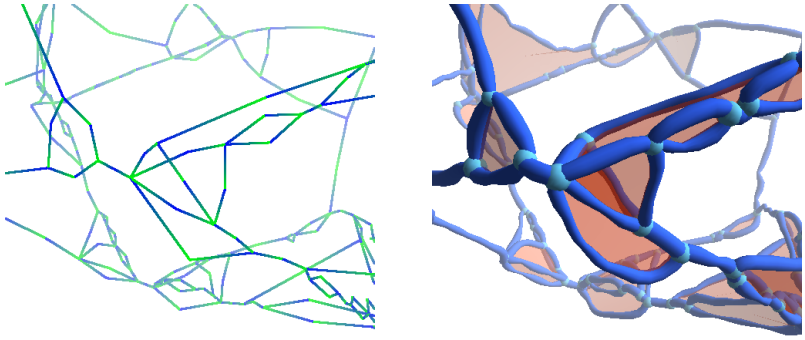


Fig. C. 52. Left: the graph after all simplification steps; Each edge connects a junction to a non-junction node. All atoms are associated with non-junction nodes. Right: the final result; Junctions are represented by cyan spheres, dislocations by blue tubes, and stacking faults by orange planes (figure 6, page 185).

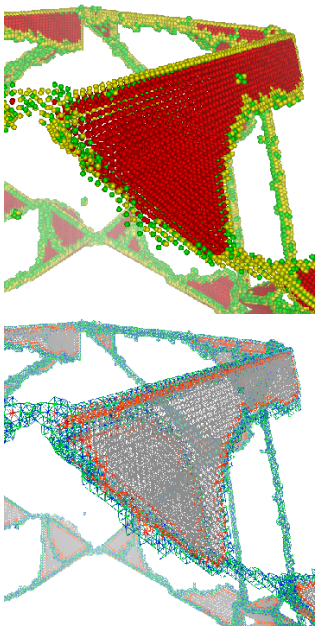


Fig. C. 53. Construction of the neighborhood graph (bottom) from the original atom data (top). Edges are color-coded: green/blue edges are from E_d, red/orange from E_b, and gray from E_s (figure 2, page 184).

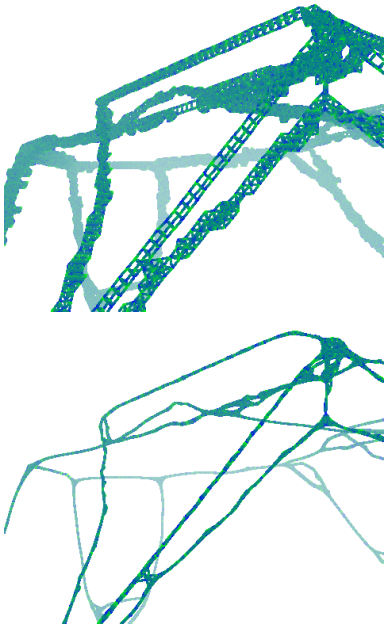


Fig. C. 54. Top image: initial neighborhood graph (only edges from E_d are shown); bottom: graph after contraction of the mass-spring system (50 iterations). Note the *unclean*, *thicker* regions near junctions (figure 3, page 184).

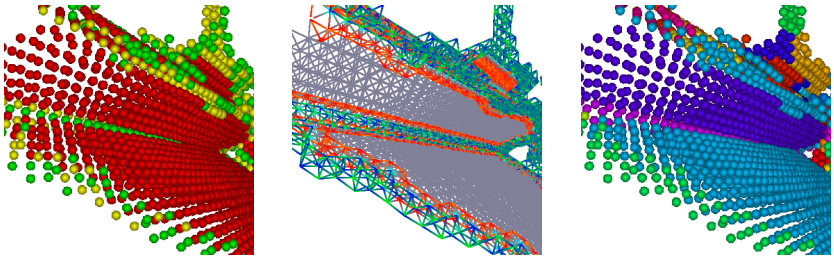
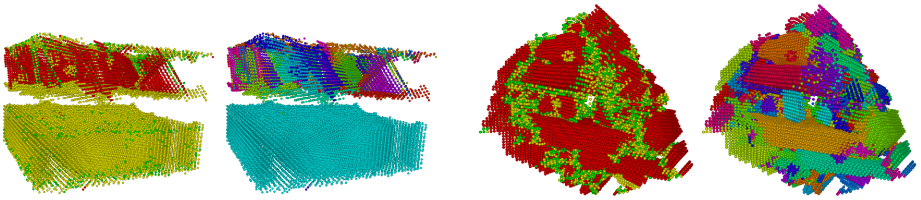
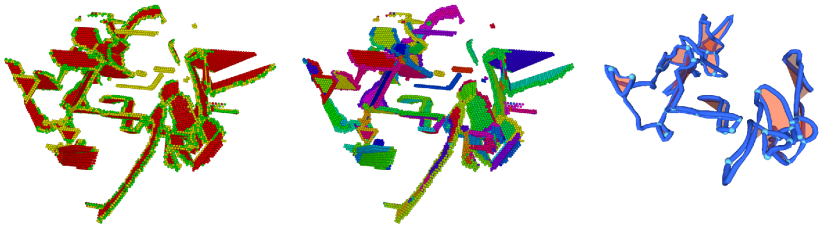


Fig. C. 55. Shows a problematic situation for the stacking fault segmentation using region growing, from left to right: The original data set; In the relative neighborhood graph atoms from different stacking fault segments are connected; the final segmentation (figure 7, page 186).



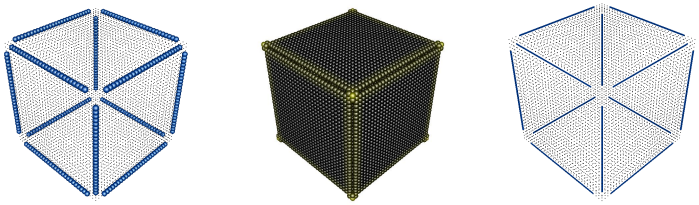
(a)Data set with two layers (upper one Ni_3Al; lower one Ni) pulled apart.

(b)Data set with only Ni_3Al pulled in one direction.



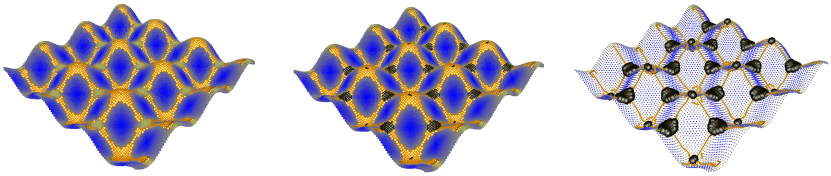
(c)Data set with Ni atoms pulled in one direction.

Fig. C. 56. Visualization examples. Each sequence shows the original data set with the lattice classification (left) and the results of the atom data segmentation (right; center in figure 8(c)). Right image in figure 8(c) shows the results of the structure extraction (figure 8, page 187).



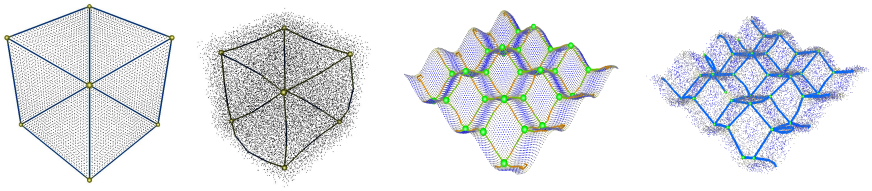
(a) (b) (c)

Fig. C. 57. Principle of graph extraction for a simple test cube data set (3000 points): (a) Initial seed nodes (blue), (b) corresponding critical nodes (black), (c) extracted edges after propagation (figure 1, page 197).



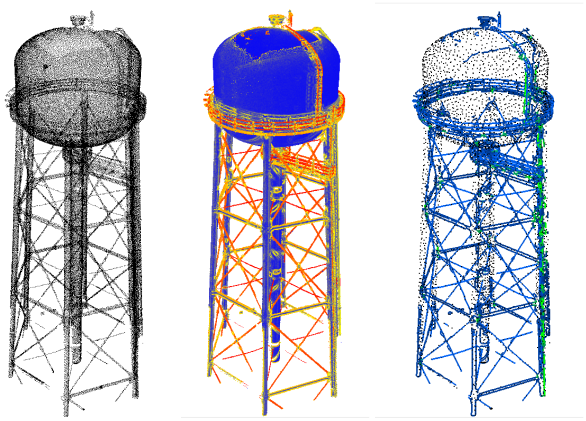
(a) (b) (c)

Fig. C. 58. Principle of graph extraction for surface-related features using a curved sheet data set (10000 points): (a) Initial seed nodes (orange) of the underlying surface (blue), (b) seed nodes with corresponding critical nodes (black), (c) critical nodes together with the extracted edges after propagation (figure 2, page 197).



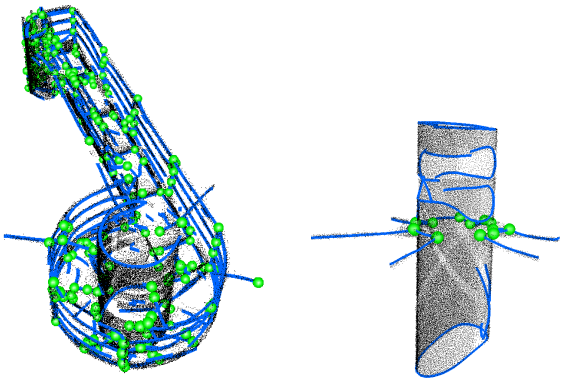
(a) (b) (c) (d)

Fig. C. 59. Results of graph extraction for the cube data set (a) without and (b) with 5% noise. Images (c) and (d) show the resulting graphs for the curved sheet data set without and with 5% noise (figure 3, page 199).



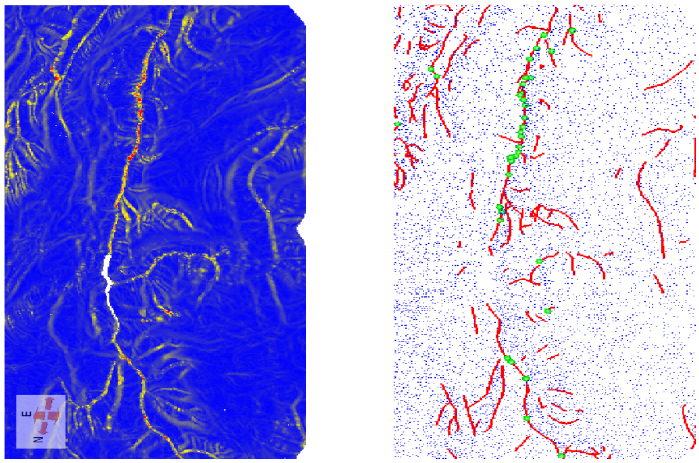
(a) (b) (c)

Fig. C. 60. (a) original point cloud of a water tower consisting of 1.7 million points; (b) color coded feature values of each point regarding curve-related features(blue represents low, red high feature values); (c) corresponding feature graph (blue lines) for $\varepsilon = 0.5$ and $r \approx 0.038\%$ of the bounding box diagonal (figure 4, page 201).



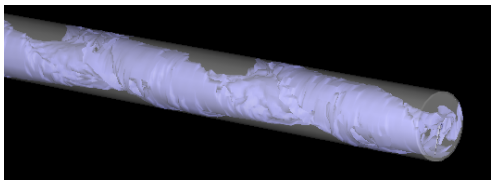
(a) (b)

Fig. C. 61. Closeup of selected parts of the extracted graph of Fig. 5(c) (figure 5, page 202).

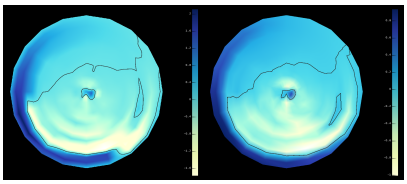


(a) (b)

Fig. C. 62. Scan of San Andreas fault in California, USA, on a length of 6 kilometers. (a) presents the corresponding feature/curvature values (blue means low, red equals high curvature value). (b) shows the corresponding surface-related feature graph with feature lines depicted in red, and critical nodes as green points (figure 6, page 203).



(a)Path line oriented topology in complete time-dependent field. Third dimension used to stack time steps ($60 < t < 80$ visible). Isosurface encloses attracting behavior.



(b)Path line oriented topology (left) compared to divergence (right) for $t = 70$. Colormap and zero isolines divide the field into areas with converging (negative values) and diverging behavior (positive values)

Fig. C. 63. Path line oriented topology (figure 3, page 210).

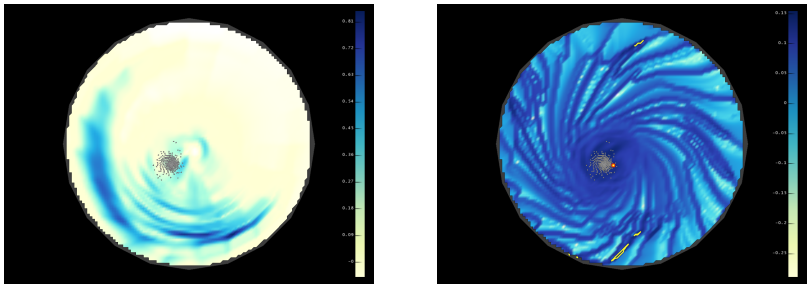
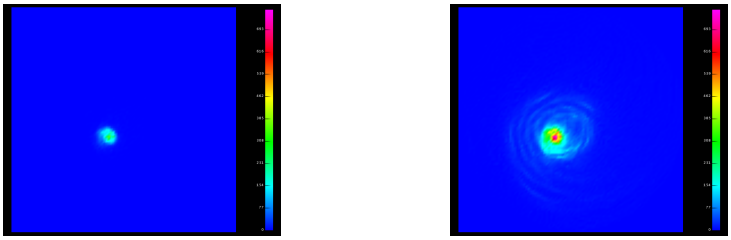


Fig. C. 64. FTLE$^-$ with particle cluster for $t = 70$. Integration for 1 (left) and 20 time units (right). Very high magnitude of FTLE$^-$ marked yellow. Note the small (enlarged) yellow point fairly close to the particles (figure 4, page 210).



(a)Density of particles traced from one grid at $t = 50$.

(b)Density of particles traced from ten grids at $t = 50, 52, \ldots, 68$.

Fig. C. 65. Color coded density distribution for time $t = 70$ and resolution of 100×100. The length and with of the square are two times the radius of the petri dish (figure 6, page 213).

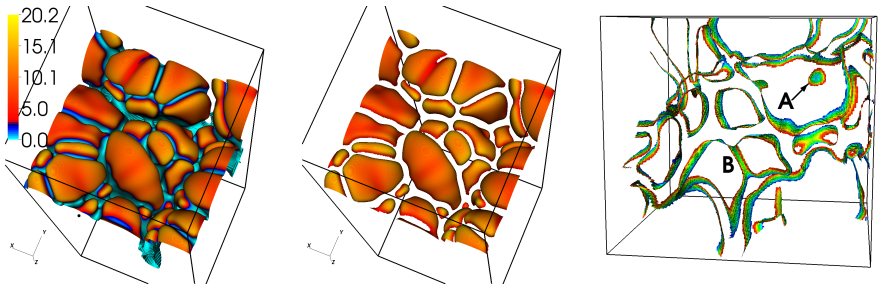


Fig. C. 66. Burning regions on an isotherm of a lean flame. (Left) Isotherm (T=1225K) extracted from a single time step and colored by fuel consumption rate. Blue signifies low fuel consumption and orange signifies high fuel consumption. (Center) Burning regions are those portions of the isotherm where fuel consumption exceeds a given threshold (here: $2.6\frac{\mathrm{kg_{H_2}}}{\mathrm{m^3 s}}$ of H_2 consumption). (Right) Traced over time, the boundaries of the burning regions (Fig. 1) sweep out surfaces. In this figure, the swept out surfaces are colored according to time (ranging from blue for early time steps to red for later time steps), which we use as the Morse function in a Reeb graph construction step.

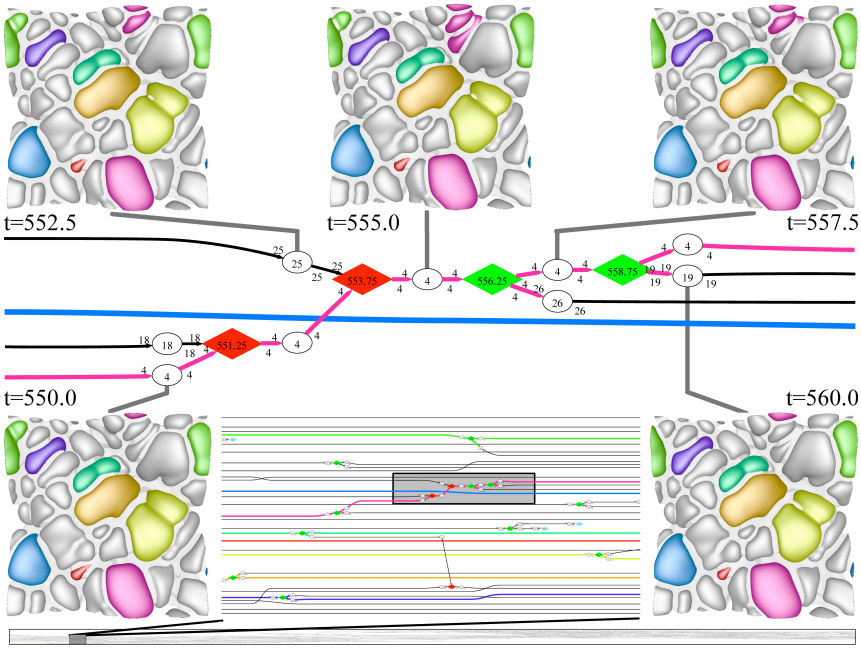


Fig. C. 67. Subsection of the tracking graph for the "no turbulence" case compressed to only show two merge and two split events. Arc color corresponds to region color in corresponding segmentations. Round nodes correspond to cells explicitly segmented by the Morse complex, diamonds to topological events between time steps. Red signifies a merge, green a split, and turquoise a birth/death event. The figure shows three zoom levels of the graph: graph for the entire simulation (bottom), a portion corresponding to several subsequent time steps (middle) and a zoom into the two merge and split events annotated with colored segmentations of the isotherm (top).

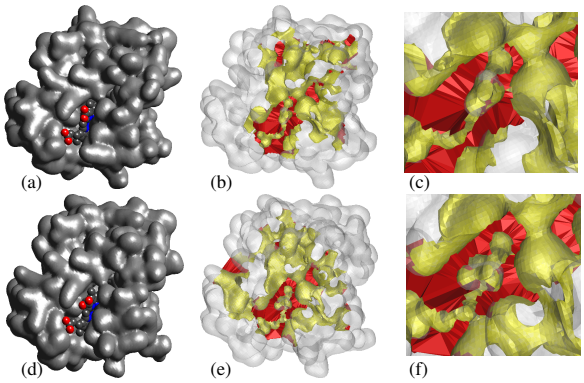


Fig. C. 68. Visualizations of the hemoglobin molecule undergoing dynamic deformation as oxygen binds to it. **(a)** A primal space visualization of the first time step with the heme group identified. **(b)** Visualization of the complementary space of the first time step shows the geometry of the interior. The surface has been made transparent, revealing a large tunnel through the surface (yellow) with many mouths (red). **(c)** Zooming in on the heme group reveals the structure of space around it while oxygen is bound. **(d-f)** The corresponding images of (a-c) for the final time step. Complementary space has changed dramatically both in the interior volume and near the heme group, though this is not evident from the primal space visualizations. Comparing (c) and (f), we observe that the connectivity of the mouth of the tunnel near the heme group has changed, illustrating the time-dependency of the topological features of complementary space. We quantify and discuss this example further in Section 3.5 (figure 1, page 241).

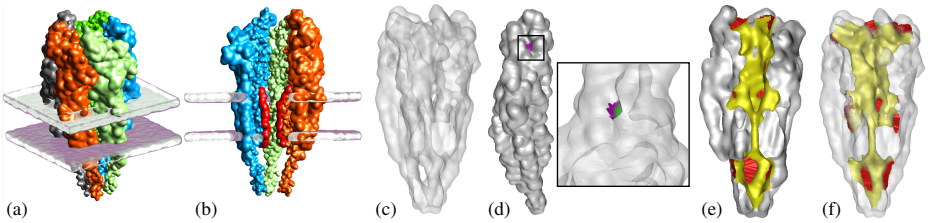


Fig. C. 69. Visualizations of the acetylcholine receptor molecule. **(a)** The molecule is shown as it would sit embedded in a bilipid cell membrane (grey) with the five identical subunit colored for identification. **(b)** A cut-away view of the same model showing where ions may pass through the center. **(c)** A transparent view of the molecular surface. **(d)** Each subunit contains a pocket where acetylcholine binds. A complementary space view of the pocket interior (green) and its mouth (purple) are shown in a zoomed in view after the surface has been made transparent. **(e)** A cut-away view of the surface with the interior of the tunnel (yellow) and its mouths (red) identified. **(f)** The same view as (c) with the tunnel geometry opaque, showing how it lies inside the surface. Visualizing the complementary space tunnel reveals that the dimensions of the pore opening on the extracellular side are much larger than on the intracellular side. This is less evident from the primal space visualizations (figure 2, page 245).

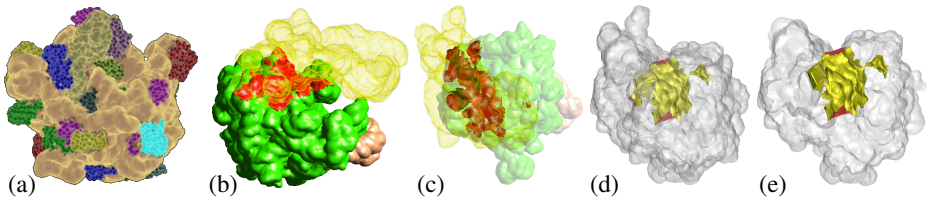


Fig. C. 70. Visualizations of the ribosome molecule. **(a)** The ribosome itself is made up of three RNA chains (brown) and dozens of proteins (various colors). We define the contact area of each protein to be any portion of its surface lying within 4 Å of an RNA chain. We compute these areas and use them to predict the order in which the proteins bind to the RNA chains. **(b)** Two of the RNA chains (green and brown) belong to the 50s subunit while the third (yellow) belongs to the 30s subunit. We compute the contact area between these chains (red) to show how the subunits come together to form the protein assembly tunnel. **(c)** A different view of the RNA chains gives a better view of the contact region but makes it difficult to see where the tunnel lies. **(d)** We run CompSpace on a surface model that includes all the attached proteins and visualize the surface (transparent) along with the tunnel mouths (red) and interior (yellow). **(e)** A cut-away view helps elucidate the intricate geometry surrounding the protein assembly region. These types of complementary space visualizations and subsequent quantifications provide insight to open questions such as whether amino acid chains have room to begin folding before they exit the ribosome (figure 3, page 247).

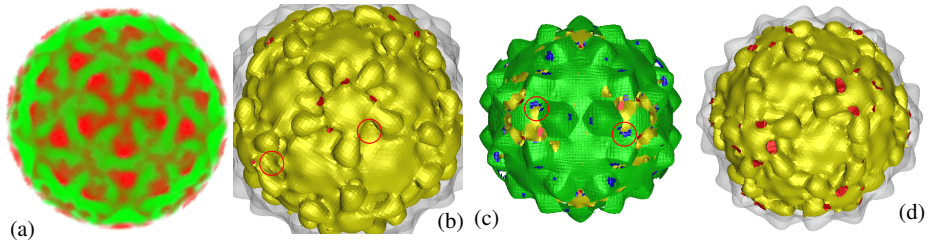


Fig. C. 71. Identification of "thin" regions in the primal space for the nodavirus dataset. **(a)** A volume rendering of the 3D image data. **(b)** Tunnels are detected for the initial selection of the isosurface. Note that in some places of 5-fold symmetry, only 4 mouths of the tunnel are present. **(c)** The thin regions (blue) are identified as subsets of the unstable manifolds of the index 1 saddles identified on the interior medial axis. The circles (red) in (b) and (c) indicate that places where the fifth mouth of the tunnel should be open indeed have thin regions. **(d)** The final selected isosurface has complementary space topology consistent with the inherent symmetry of the 3D density map (figure 4, page 248).

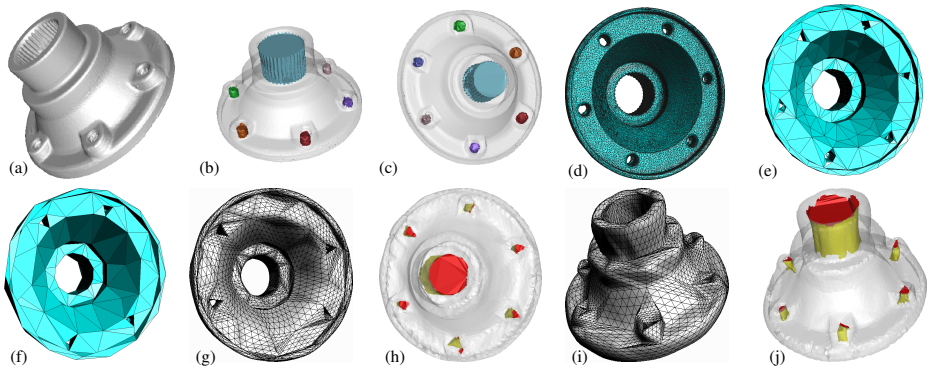


Fig. C. 72. Visualizations of the Carter dataset. **(a)** A basic primal space visualization of the mechanical part. **(b-c)** Complementary space features identified and visualized. **(d)** A visualization of the dense mesh representing the surface reveals that at 106,708 triangles, it is probably amenable to decimation. **(e-f)** Using QSlim [11], the mesh is decimated to 1000 and then to 500 triangles. **(g-j)** After refinement and geometric improvement on the 500 triangle model, it appears from the primal space visualizations (g and i) that some of the tunnels have collapsed, a topological change. Complementary space visualizations (h and j) reveal that in fact the tunnels are still present but with perturbed geometry. Therefore, this reduced model has consistent topology with the original (figure 5, page 249).

Author Index

Index