Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2012) 1709–1718
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

An image-based approach to interactive crease extraction and rendering

S. Barakat[1], X. Tricoche[1,*]

Computer Science Department, Purdue University
West Lafayette, Indiana, USA

Abstract

Ridge and valley manifolds are receiving a growing attention in visualization research due to their ability to reveal the shapes of salient structures in numerical datasets across scientific, engineering, and medical applications. However, the methods proposed to date for their extraction in the visualization and image analysis literature are computationally expensive and typically applied in an offline setting. This setup does not properly support a user-driven exploration, which often requires control over various parameters tuned to filter false positives and spurious artifacts and highlight the most significant structures. This paper presents a GPU-based adaptive technique for crease extraction and visualization across scales. Our method combines a scale-space analysis of the data in pre-processing with a ray casting approach supporting a robust and efficient one-dimensional numerical search, and an image-based rendering strategy. This general framework achieves high-quality crease surface representations at interactive frame rates. Results are proposed for analytical, medical, and computational datasets.

Keywords: Ridge and valley surfaces, Ray casting, Volume rendering, Image-based rendering, GPU, Interactivity

1. Introduction

Ridge and valley manifolds are currently gaining popularity in the scientific visualization literature. While the study of these lines and surfaces was initiated in the image analysis research community [1, 2], a number of recent publications has documented their effectiveness for the visual representation of scalar, vector, and tensor fields [3, 4, 5, 6, 7]. Creases correspond to manifolds along which the considered scalar quantity is locally extremal. Yet, in contrast to the standard notion of local maxima or minima, which typically form isolated points, the definition of creases allows the corresponding set of points to form lines and surfaces. As a result, creases yield a schematic geometric portrait of the dataset that captures some of its salient structures.

Despite this conceptual appeal, the extraction of ridges and valleys is a non-trivial endeavor that requires a careful analysis of the spatial variations of the gradient and Hessian (first and second-order derivative, respectively) of a scalar field. The techniques proposed for that extraction typically amount to an isosurface extraction implemented as a specialized form of marching cubes. Unfortunately, the corresponding methods are usually slow because of

*Corresponding author
Email addresses: sbarakat@purdue.edu (S. Barakat), xmt@purdue.edu (X. Tricoche)

1877-0509
doi:10.1016/j.procs.2010.04.192

the nonlinear nature of the creases, which requires a high sampling density and heavy computations to properly resolve the manifold. As a result, the extraction must be carried out in a preprocessing stage. In addition, the resulting geometry is very complex and contains a large number of spurious small scale structures. Hence, the user is generally only interested in a small subset of the extracted creases and various criteria and heuristics are necessary to filter out insignificant features from the resulting depiction. In practice some of these criteria are often built in the extraction procedure to alleviate its computational cost, facilitate subsequent filtering, and reduce visual clutter. As a consequence, significant ridges may be missed and their topology improperly determined.

We present in this paper a novel method that addresses these limitations and permits the interactive extraction and rendering of crease surfaces. By exploiting the computing power of the graphics hardware (GPU) our technique allows the user to adjust any relevant filtering criteria in the course of the extraction based on the visual feedback. We use an image-based approach to restrict the computation to the visible portions of the dataset at a resolution that is commensurate with the camera setting. As such, our algorithm not only offers intuitive and flexible control over the produced geometry, it also enables a crease-based exploration of the volume through which domain experts can discover important structures present in the data. Specifically, we employ a ray-casting approach and determine the existence of one or several intersection with crease surfaces along each ray as the solution of a one-dimensional search. Note that this search is significantly more complicated than the standard zero crossing test associated with isosurface ray-casting [8, 9]. This implies that a brute force strategy consisting in uniformly sampling each ray in search for a crossing is intractable in an interactive setting. To tackle this problem, we propose reliable two-level empty space skipping strategy combined with a robust numerical search with adaptive step size. This setup allows us to integrate crease extraction and volume rendering and produce at interactive frame rates enhanced visual representations of the data in which sharp salient manifolds are revealed in the fuzzy context of the embedding field.

The paper is organized as follows. We briefly review previous work on crease extraction in Section 2. We describe our fast and robust one-dimensional crease extraction procedure in a ray-casting framework in Section 3. We discuss our proposed rendering solution for the resulting crease points in Section 4. We report on the application of our technique to several scientific, engineering and medical imaging datasets in Section 5 before concluding in Section 6.

2. Related Work

Ridges (and valleys), like edges are fundamental image features in image analysis [10, 11]. In essence they can be interpreted as the generalization of the notion of point-wise extrema (where the gradient vanishes) of smooth scalar fields to manifolds of higher dimensions (e.g., "extremal" curves and surfaces). Among the various existing definitions, the notion of height ridge is arguably the most widely used in practice [10, 2]. In Eberly's definition in particular [12], at a point on a ridge surface the gradient $\mathbf{g} = \nabla f$ is orthogonal to the minor eigenvector $\mathbf{e}_3$ (associated with the smallest eigenvalue) of the Hessian $\mathbf{H} = \nabla^2 f$. In addition, the definition imposes that the corresponding eigenvalue λ_3 be negative to ensure the convexity of the function across the ridge. Note that a similar definition exists for valley surfaces that requires $\mathbf{g}$ to be orthogonal to the major eigenvector of $\mathbf{H}$, while the associated major eigenvalue must be positive. We refer to both ridges and valleys as *creases* in this paper.

A fundamental consideration in the study of these manifolds is the notion of scale space [13, 14], which embeds a signal into a continuum of scales for analysis. The rationale for this approach in computer vision is that objects inherently comprise details of various scales [15, 16] and their appearance changes as a function of the scale (or aperture) of the observation. Applying this principle allows to study the multi-scale structure of images and volumes. Practically, a gaussian kernel of varying standard deviation progressively filters out the low frequencies present in the signal thus producing an additional scale axis. Lindeberg made major contributions to the automatic determination of the optimal scale [17, 16] and studied its application to ridge extraction. His approach to determine the optimal scale at each point amounts to maximizing a normalized measure of feature (e.g. ridge) strength across scales.

From an algorithmic standpoint, ridge surfaces are typically extracted as a generalized isosurface of the scalar product between the gradient ∇f and the minor eigenvector e_3 of the Hessian [18]. The detection of the zero crossings of this scalar product requires careful inspection of the edges of the voxel grid since the considered eigenvector field has neither norm nor intrinsic orientation [19, 3]. Alternatively, a signed scalar quantity can be defined whose zero crossings equivalently characterize ridges [5]. In either case the surface extraction then amounts to an application of the marching cubes algorithm [20]. Note that a significantly improved method was recently presented by Schultz et al. [6] who derived a topologically correct extraction strategy.

From a visualization standpoint, the algorithmic strategies used in practice are typically directly inspired by methods introduced in medical image analysis [1, 21, 22]. In addition, Peikert and Roth proposed the *Parallel Vector Operator* [23] for the extraction of line and surface manifolds that constitute a superset of ridges and valleys. In particular, this technique was recently applied to vector [24] and tensor visualization [4] problems. Finally, following a fundamentally different approach, Kindlmann et al. [7] presented a particle-based method in which individual particles are moving in a scale-space continuum according to forces that induce their convergence to optimal sampling locations. While the method does not explicitly produce manifolds in output, it is able to achieve a uniform discrete sampling of the underlying lines and surfaces.

3. Algorithm

To turn the the visualization of crease surfaces into an interactive technique, we adopt a ray-casting approach that shares similarities with solutions proposed in the last few years for the view-dependent extraction and rendering of isosurfaces on the GPU [8, 25]. Their basic idea consists namely in substituting to the standard per-voxel isosurface extraction [20] a one-dimensional search for zero crossings along individual rays. This latter approach is intrinsically parallel and its complexity is primarily controlled by the camera setting and the output resolution. However, the computation needed to reliably extract crease surfaces is fundamentally more involved than in the case of isosurfaces. In particular, the nonlinear nature of the considered quantities requires a more sophisticated solution, as we describe next.

3.1. One-Dimensional Crease Extraction Along a Ray

The algorithmic building block of our method is the extraction of crease surface points along individual view rays cast through the image plane. This extraction is carried out concurrently across rays to leverage the massively parallel nature of the graphics hardware.

3.1.1. Function evaluation

As the search progresses along the ray, both the Hessian's relevant eigenvalue and its corresponding eigenvector must be computed at the sampled locations. However, the nonlinear relationship that exist between a matrix and its eigenvectors and eigenvalues requires us to first determine the Hessian before solving the associated eigensystem. In that regard, we follow the general approach advocated by Kindlmann and his coauthors [3, 4, 7]. However, the constraints imposed by the GPU do not allow us to use the high-quality reconstruction kernels that were previously used in CPU-based implementations. Instead, we exploit the optimized trilinear interpolation available for texture memory to interpolate pre-computed gradient and Hessian fields at subvoxel resolution.

3.1.2. The need for adaptive sampling

A straightforward method to find crease points along a ray would be to compute the gradient and Hessian's eigenvector at regular intervals and search for zero crossings of their dot product. This is in fact the one-dimensional equivalent of the marching cubes approach and its crease-based variants. Yet, this simple strategy is inappropriate for several reasons. First, the associated computational cost (which scales with the screen resolution) would be prohibitive in an interactive setting unless the sampling density along each ray is very coarse. Decreasing the sampling density of a nonlinear quantity such as $\mathbf{g} \cdot \mathbf{e}_i$, however increases the risk of missing a crease point. This problem has led recently proposed algorithms to process upsampled versions of the data. Moreover, in the context of our method, an accurate spatial localization of the crease points is crucial to compute artifact-free surface normals in image space as explained in Section 4.

To overcome this limitation, we apply an adaptive sampling strategy along the ray. A technique using irregular steps based on gradient descent has been discussed by Kindlmann et al. [7]. Unfortunately their method is not readily applicable to our setting since it is built on a model of free particle movements in 4D scale-space while our search is confined to a 1D domain. We therefore propose a different adaptive stepping scheme that is reliable and allows us to find crease points with an accuracy that can be tuned by the user interactively.

3.1.3. Two-level empty space skipping

Finding the crease eigenvalue and its corresponding eigenvector is computationally expensive. To reduce the computational cost of the search along the rays we perform several checks that allow us to skip regions that do not contain a crease.

At a coarse level, our empty space skipping strategy relies on a map that conservatively estimates the distance between any voxel in the grid and the boundary of the closest region in which the relevant Hessian's eigenvalue λ_i has valid sign ($\lambda_3 < 0$ for ridges and $\lambda_1 > 0$ for valleys, refer to Section 2). This map is computed in pre-processing using a region growing approach. At run time, the rays fetch values from this map in order to find the longest safe jump that can be performed without missing a potential crease point. The main advantage of this method is that it makes more complex data structures (e.g. octree) unnecessary, which would introduce construction overhead and the need to locate each point in a hierarchical structure. Furthermore, it can be efficiently referenced using a texture implementation on GPU.

When the value obtained from this empty space skipping map is zero the search switches back to regular steps along the ray. At that finer level, our method leverages a second pre-computed map that stores the considered eigenvalue λ_i at the resolution of the input grid. Hence, during the extraction we numerically derive the eigenvalue and associated eigenvector from the Hessian only if the sign of λ_i has the correct sign and satisfies $|\lambda_i| > \epsilon_\lambda$, whereby ϵ_λ is a user-defined threshold. In addition we avoid performing computations when the interpolated scalar field value at the sampled location is below a second threshold ϵ_f. These conditions can be supplied and tuned by the user on the fly to increase performance while discarding insignificant geometric structures and maintaining the quality of the search.

If all the previous conditions are met, the ray advances according to an iterative technique describe below. Note that the fixed step mentioned in the second level can be large enough to overlook a crossing, in which the first point meeting the criteria will be already passed the crease point location. This situation is properly dealt with by our iterative method, which is able to backtrack in order to find the missed crossing. Nevertheless, the step should be chosen small enough to not miss completely the range of interesting crease eigenvalue and the zero crossing.

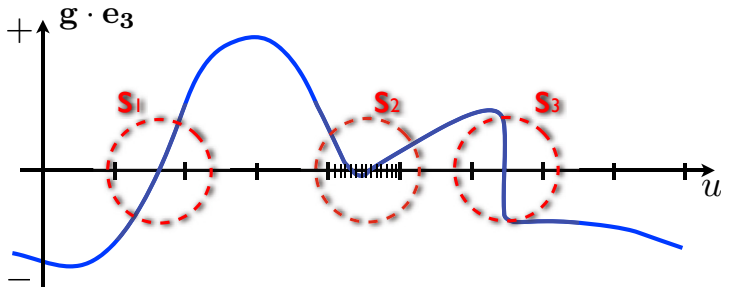

Figure 1: Different cases for the zero crossings of the function $g.e_3$ along a ray.

3.1.4. Iterative numerical search

Once a valid region has been reached the iterative part of the algorithm takes place. It combines the Newton-Raphson method with bisection and zero crossing tests. The bisection method is applied once a zero crossing has been identified in a certain close proximity. Due to the complexity of the function $\mathbf{g} \cdot \mathbf{e}_i$, the ray keeps track of the last point beyond which backtracking cannot be applied. This point typically indicates regions that were thoroughly covered. This helps ensure that the numerical search does not overshoot due to irregular gradient values in the test function $\mathbf{g} \cdot \mathbf{e}_i$. It also prevents situations of infinite looping due to the gradient pointing away from the crossing point. This is common since most of the crossings happen at sharp edges (high gradient) where the gradient could be pointing away. An example of such case is shown in the last ridge crossing in Figure 1, S3. We hold a second marker along the ray which indicates the furthest point along the ray that has been visited. This marker is used to prevent the ray from revisiting portions that were already visited during the backtracking once the crossing is found.

The maximum step size taken along the ray during the iterative part of the algorithm must preserve the consistency of the eigenvector's orientation across sampling locations. To do so, we measure the angular difference between consecutive eigenvectors. If that angle a certain threshold, the step size is reduced since no meaningful zero crossing test can be applied. This idea is similar to an orientation tracking approach proposed recently [3]. The minimum step should be limited by the shortest distance that might contain an inversion of the function $g \cdot e_3$ that is barely touching the zero value. This can be seen in the second ridge crossing along the ray shown in figure 1, S2.

3.2. Scale Analysis

The previous discussion did not specifically address the notion of scale. Yet, creases and scale-space are indissociable concepts as shown by Lindeberg [26], a fact that has been mostly ignored in the visualization literature so far. A notable exception is the very recent work by Kindlmann et al. [7] who extract ridges in a scale space continuum. Our scale space processing is based on the reconstruction framework that they proposed, which we combine with a different extraction strategy that is more suitable to the performance constraints of our method.

Practically we first compute in pre-processing the gradient and Hessian of the considered function for a range of scales. We do so at a resolution higher than the initial resolution of the scalar field to overcome the nonlinearity of the ridges and achieve smoother results despite the low-order interpolation scheme available on the GPU. A smooth reconstruction kernel is used for upsampling [3]. In contrast to offline methods, handling on the fly gradient and Hessian fields at the different scales simultaneously is basically impossible on the GPU. Instead, we perform an extraction of the optimal scale in pre-processing by maximizing at each point the scale-normalized measure of crease strength λ_i along the scale axis. The gradient, Hessian, and all other relevant quantities are then computed at each domain position at that optimal scale. Note that this definition of optimal scale, while different from the one recently used by Kindlmann et al. [7], is in fact the one initially proposed by Lindeberg for ridge extraction [26]. A potential downside of this approach, however, is that it does not guarantee the spatial smoothness of the optimal scale, since this value is determined independently at each position. Because trilinear filtering is used in our GPU implementation to interpolate both gradient and Hessian at any point in the volume, we namely need the scale distribution to be spatially smooth. This is also necessary because multiple local maxima usually exist across scales while we are only interested in a single optimal scale per point. Practically we achieve smooth results by densely sampling the scale axis. We select our coarsest scale so as to retain the main structures in the data. In addition, we provide the user with interactive control over the range of scales for which creases are extracted. Our results, presented in Section 5, document the effectiveness of this general strategy.

4. Single Pass Rendering

Our surface rendering technique is based on ray casting. After identifying the intersection points of the rays with the creases we determine the normals and compute the compositing. To permit an opacity transfer function based on the crease strength, we keep track of the accumulated opacity during the extraction. Early ray termination is used to avoid the extraction overhead in occluded regions of the volume. The surface and volume rendering are performed in two stages.

In the first stage, rays are cast through the volume and extraction is performed as outlined in the previous section. Every ray searches for crease points until the opacity is saturated or a maximum number of intersections is found. Depending on the complexity of the structures, the maximum number of intersections indicates the limit after which additional surfaces will not have significant effect on the visualization result. This maximum number is chosen such that occlusion and opacity will make any additional surfaces non-distinguishable from the preceding surfaces.

In the second stage of the rendering we access the intersection points in order along each ray. At each pixel in image space we compute the normals and we perform the color compositing operations in front to back order. In both stages, individual threads are assigned to rays. The surface normals at intersection points are needed for proper lighting of the surfaces. The normals are computed in screen space using a technique similar to the z-buffer based approach used in previous publications [27, 28]. However, we apply it to a multilayer buffer containing the points' position in addition to depth. For each point we find neighbors by measuring the distance to the intersection points of neighboring rays. Least squares fitting over a 3x3 neighborhood is used to find the normal at the point using the found neighboring points. Least squares fitting has two advantages over central differences. First, it gives

better quality normals given a more relaxed accuracy to which intersection points are identified. Second, it is more capable of handling points lying on the silhouette of the surface since only three non-aligned points are sufficient. We detect discontinuities in the frame buffer by measuring distances between points acquired by neighboring rays. This computation technique of the normals satisfies an important property namely the smoothness of the normals along the surface since the least squares fitting combined with the screen based sampling of the surface acts as a low pass filter in image space.

Combining this surface rendering strategy with volume rendering at lower opacities provides an insightful means for the user to understand the ridge shape and structure in their embedding in the field. This can be done by adding an additional step to be performed between the renderings of two consecutive intersection points along the ray. We perform usual volume rendering until we meet the first intersection point if any. We also apply the volume rendering step from an intersection point to the next or until the color becomes opaque during the second stage. Since we do not have color information for surfaces until the normals are computed, we cannot apply compositing for the volume rendering in the first stage. The drawback of applying volume rendering in the second stage is the inability to benefit from possible early ray termination based on the volume rendering during the extraction.

5. Results

We tested our method on a high-end Intel Core2 Extreme QX9650 (12MB, 3.0GHz) machine where the GPU kernels are executed on NVIDIA GeForce GTX 280 (30 multiprocessors, 1024 thread/multiprocessor, 1GB device memory). The implementation was done in CUDA and C++. To document the performance of our method, we present here results obtained for several datasets corresponding to different application scenarios.

The first dataset is a simple synthetic test case that allows us to document the various features of the algorithm. The input scalar field was created by applying to a Möbius band volume a level of gaussian blurring that varies linearly along the z-axis. This dataset is similar to the one shown in the recent paper by Kindlmann et al. [7]. The varying levels of blur allow us to show the shortcomings of a ridge surface extraction based on a single scale. The results shown in Figure 2 clearly underscore the importance of incorporating scale in the crease extraction. In addition they confirm that the optimal scale approach is able to properly capture the various scales introduced in the data through blurring. Observe that the surface element visible at the top of the right image is an artifact of the blurring in pre-processing, not of the extraction itself.

As a second example, we consider a diffusion tensor MRI (DTI) dataset of a human brain. While medical images have been a traditional application of ridge extraction techniques in computer vision, DTI datasets pose a significant challenge because of their tensor nature. Indeed, the scalar invariants that are derived from these tensor fields are highly nonlinear, which greatly complicates the extraction of the associated structures. Following recent publications [3, 4, 6, 7], we apply ridge and valley manifold extraction to the fractional anisotropy (FA) of the DTI field. As previously shown in the literature, ridge surfaces of FA delineate the two-dimensional core structures of major bundles in the brain white matter, while valley surfaces of FA constitute boundaries between adjacent fiber bundles with distinct orientations. Figure 4 visualizes both ridge and valley surfaces of FA alongside fiber bundles that were computed along major anatomical structures. Recent work [7] has clearly documented the multiple scales that are present in the structures of the ridge surfaces. We obtain similar results, though in the form of actual surface renderings of both ridge and valley surfaces. Valley surfaces in particular tend to be more difficult to extract due to their inherently thin geometry at the interface between different well delineated structures and the fairly coarse resolution of DTI datasets. Figure 4, top left and right images, displays the scale information on the ridge manifolds.

Fluid flows constitute another compelling application of crease surface visualization. Indeed, so-called Lagrangian coherent structures [29], which are known to form the salient geometric structures of transient flows by acting as attracting or repelling material surfaces, have been shown to be effectively characterized as ridges of a scalar coherence measure called finite-time Lyapunov exponent (FTLE). We show here the application of our method to two fluid flow problems. The first one, the ABC flow, is a standard analytical model that is used to study turbulence, see Figure 3.The second one is a CFD simulation of a turbulent jet into a steady medium forming typical convoluted patterns of turbulence. The left image shows a standard volume rendering of the FTLE field while the right image precisely displays the underlying LCS of various scales responsible for the observed patterns. Refer to Figure 3.

The frame rate of our method is primarily controlled by the amount of filtering that is applied to the crease strength in order to focus the crease extraction to the main structures. In the particular case of the möbius band dataset where

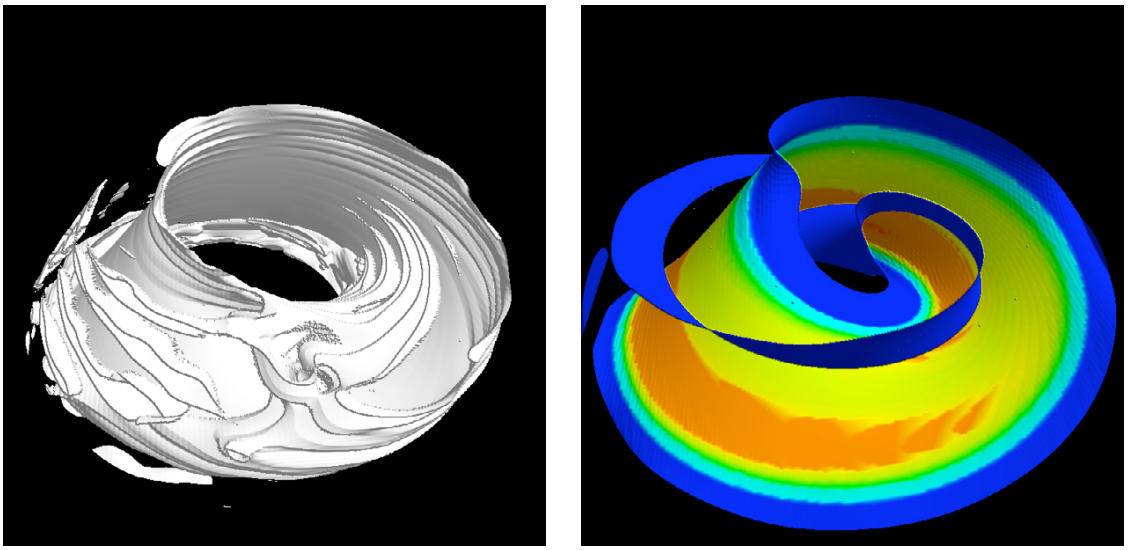

Figure 2: Ridge surface forming a Möbius band in scale space. Left: Ridge surface extracted at constant scale contains a number of spurious artifacts. Right: Extraction of the ridge surface at optimal scale yields a smooth manifold. The colors represent the various scales present in the dataset (blue: large scale, red: small scale).

a wide range of crease strengths correspond to meaningful structures, the frame rate was comparatively low. The performance achieved for each of the datasets considered in this paper is described in the following table.

Table 1: Frame rates for the test cases shown above

series	series dimensions	series minimum	series maximum	series average
Mobius	128x256x64	0.63	2	1.25
ABC	128x128x128	0.63	5	2.5
Brain	128x256x128	1.5	6.25	2
Jet4	128x256x256	2.4	5	3.125

6. Conclusion and Future Work

We have presented a method for the interactive extraction and visualization of crease surfaces on the GPU. In significant contrast to existing techniques, our approach is fast and produces results at several frames per second. As such, our algorithm allows the user to precisely control parameters of the extraction based on their visual impact on the the resulting structures. It also provides an exploratory tool through which the user can freely navigate across space and scales. An additional benefit of this method is its natural complementarity with volume rendering, thus enabling the creation of crease-centric volume visualizations. We have demonstrated the effectiveness of this approach on several examples corresponding to applications in science, medicine, and engineering.

Acknowledgments

The CFD dataset is courtesy of C. Garth, UC Davis. The brain DTI dataset was kindly provided by Gordon Kindlmann, University of Chicago. The authors also wish to thank him for his help with his software library *Teem*

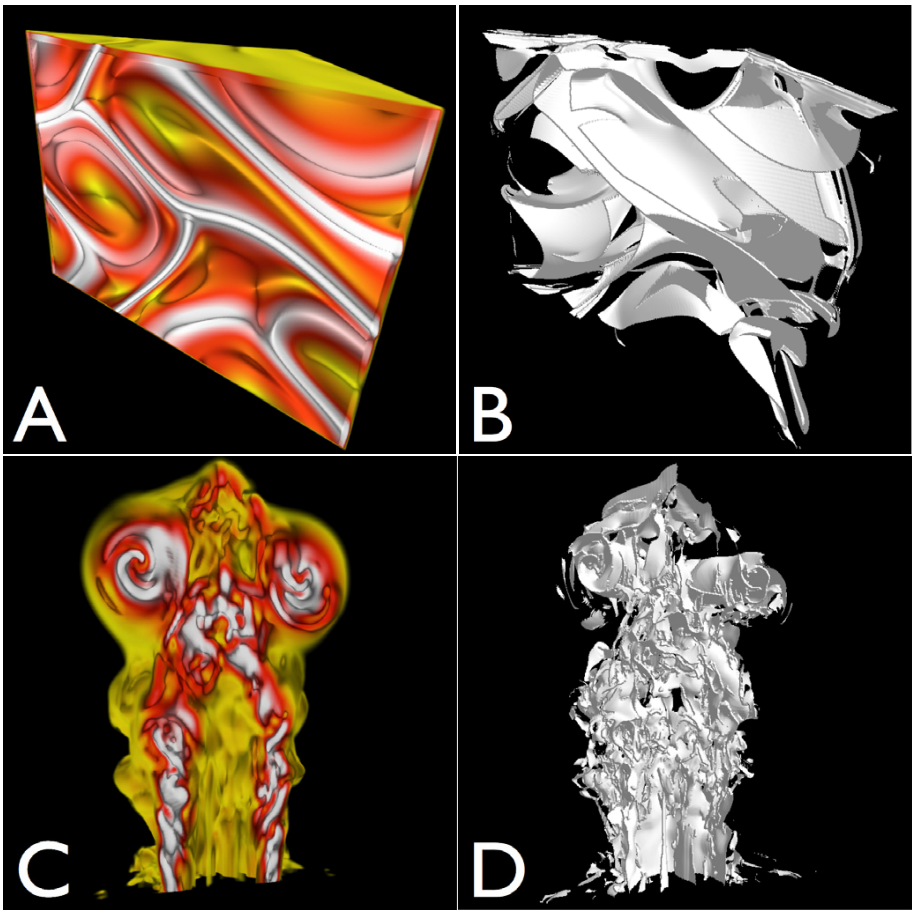

Figure 3: Lagrangian coherent structures extracted as ridges of the FTLE field in fluid flows. **Top row:** ABC flow (here $A = \sqrt{3}$, $B = \sqrt{2}$, $C = 1$). A: direct volume rendering of the clipped FTLE field. B: corresponding ridge surfaces. **Bottom row:** LCS extracted in a CFD simulation of a turbulent jet. C: volume rendering of the LCS. D: Ridge surfaces forming unstable manifolds.

(`http://teem.sf.net`). This work was partially supported by a Intel PhD Fellowship (SB) and a gift by Intel Corporation "*Exploring the Capabilities of the Larrabee Platform*" (XT).

References

[1] D. Eberly, R. Gardner, B. Morse, S. Pizer, Ridges for image analysis, Journal of Mathematical Imaging and Vision 4 (1994) 351–371.
[2] T. Lindeberg, Edge detection and ridge detection with automatic scale selection, in: Proceedings CVPR 1996, 1996, pp. 465–470. doi:10.1109/CVPR.1996.517113.
[3] G. Kindlmann, X. Tricoche, C.-F. Westin, Delineating white matter structure in diffusion tensor MRI with anisotropy creases, Medical Image Analysis 11 (5) (2007) 492–502. doi:10.1016/j.media.2007.07.005.
[4] X. Tricoche, G. Kindlmann, C.-F. Westin, Invariant crease lines for topological and structural analysis of tensor fields, IEEE Transactions on Visualization and Computer Graphics 14 (6) (2008) 1627–1634.
[5] R. Peikert, F. Sadlo, Height Ridge Computation and Filtering for Visualization, in: I. Fujishiro, H. Li, K.-L. Ma (Eds.), Proceedings of Pacific Vis 2008, 2008, pp. 119–126.
[6] T. Schultz, H. Theisel, H.-P. Seidel, Crease surfaces: From theory to extraction and application to diffusion tensor MRI, IEEE Transactions on Visualization and Computer GraphicsRapidPost, accepted 16 April 2009. doi 10.1109/TVCG.2009.44.
[7] G. Kindlmann, R. San Jose Estepar, S. M. Smith, C.-F. Westin, Sampling and visualizing creases with scale-space particles, IEEE Transactions on Visualization and Computer Graphics 15 (6) (2009) 1415–1424.

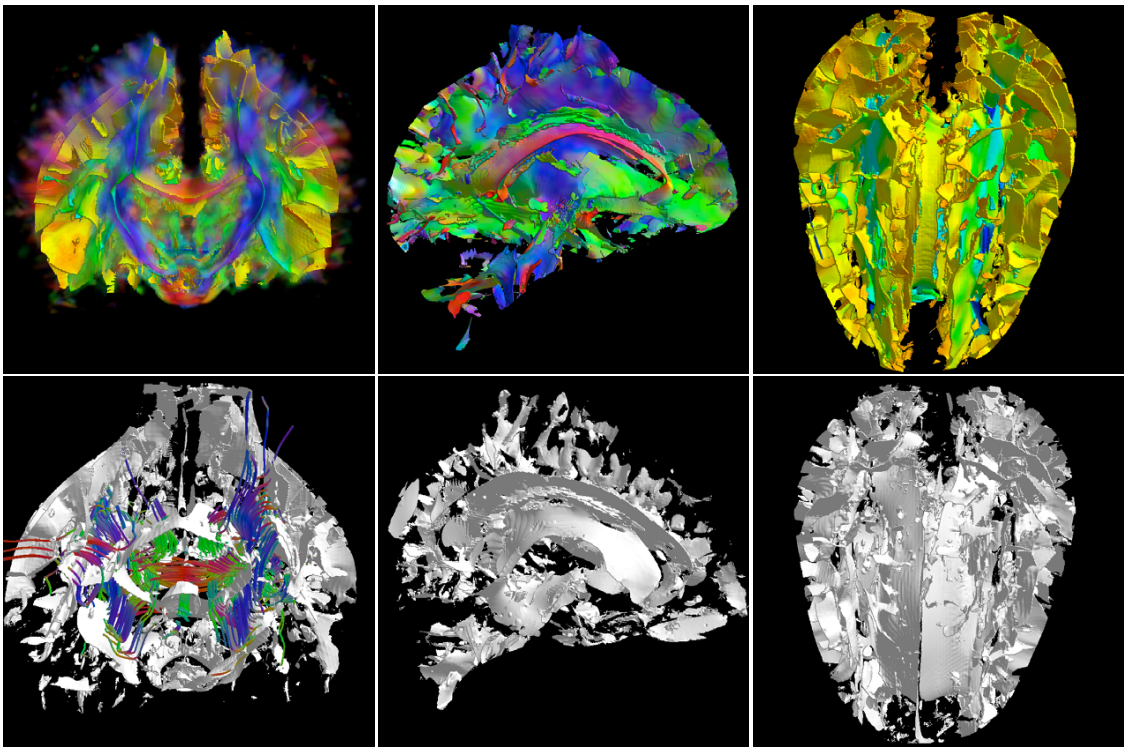

Figure 4: Ridge and valley surfaces of FA in a DT-MRI dataset of a human brain. Top row: sagittal, coronal, and axial view of clipped ridge surfaces. Left and right images show surfaces color coded by scale (red: small, blue: large), while middle image is color coded according to the orientation of the diffusion's major eigenvector direction. Left image combines ridge surface with volume rendering of FA. Major anatomical structures such as corpus callosum (CC), cingulum bundles (CB), corona radiata (CR), and internal capsules (IC) are clearly identified. Bottom row: valley surfaces of FA. The bottom left view shows the relationship between the fiber bundles corresponding to these different structures and the valley surfaces. In particular, valleys properly capture the interface between corpus callosum (in red, across) and cingulum bundles (green).

[8] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, M. Gross, Real-time ray-casting and advanced shading of discrete isosurface, Computer Graphics Forum (Proc. Eurographics '05) (2005) 303–312.

[9] Q. Wang, J. JaJa, Interactive high-resolution isosurface ray casting on multicore processors, IEEE Transactions on Visualization and Computer Graphics 14 (3) (2008) 603–614. doi:http://dx.doi.org/10.1109/TVCG.2007.70630.

[10] D. Eberly, Ridges in Image and Data Analysis, Kluwer Academic Publishers, 1996.

[11] S. M. Pizer, D. Eberly, D. S. Fritsch, B. S. Morse, Zoom-invariant vision of figural shape: The mathematics of cores, Computer Vision and Image Understanding: CVIU 69 (1) (1998) 055–071.
URL `citeseer.ist.psu.edu/pizer97zoominvariant.html`

[12] D. Eberly, R. Gardner, B. Morse, S. Pizer, Ridges for image analysis, Journal of Mathematical Imaging and Vision 4 (1994) 351–371.

[13] A. P. Witkin, Scale-space filtering, in: Proc. 8th International Joint Conference on Artificial Intelligence, 1983, pp. 1019–1022.

[14] J. Koenderink, The structure of images, Biological Cybernetics 50 (1984) 363–370.

[15] T. Lindeberg, Effective scale: A natural unit for measuring scale-space lifetime, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (10) (1993) 1068–1074.

[16] T. Lindeberg, Scale-Space Theory in Computer Vision, Kluwer Academic Publishers, 1994.

[17] T. Lindeberg, Scale-space for discrete signals, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (3) (1990) 234–254.

[18] J. D. Furst, S. M. Pizer, Marching ridges, in: Proceedings of 2001 IASTED International Conference on Signal and Image Processing, 2001, pp. 22–26.

[19] G. D. Stetten, S. M. Pizer, Medial-node models to identify and measure objects in real-time 3-d echocardiography, IEEE Transactions on Medical Imaging 18 (10) (1999) 1025–1034.

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

[21] J. D. Furst, S. M. Pizer, D. H. Eberly, Marching cores: A method for extracting cores from 3D medical images, in: Proceedings of IEEE Workshop on Mathematical Methods in Biomedical Image Analysis, 1996, pp. 124–130. doi:10.1109/MMBIA.1996.534064.

[22] J. D. Furst, S. M. Pizer, D. H. Eberly, Marching cores: A method for extracting cores from 3d medical images, in: Proceedings of IEEE Workshop on Mathematical Methods in Biomedical Image Analysis, 1996, pp. 124–130.
[23] R. Peikert, M. Roth, The "parallel vectors" operator - a vector field visualization primitive, in: Proceedings of IEEE Visualization Conference, 1999.
[24] H. Theisel, J. Sahner, T. Weinkauf, H.-C. Hege, H.-P. Seidel, Extraction of parallel vector surfaces in 3D time-dependent fields and application to vortex core line tracking, in: Proc. IEEE Visualization 2005, Minneapolis, U.S.A., 2005, pp. 631–638.
[25] A. Knoll, Y. Hijazi, R. Westerteiger, M. Schott, C. Hansen, Volume ray casting with peak findings and differential sampling, IEEE Transactions on Visualization and Computer Graphics 15 (6).
[26] T. Lindeberg, Edge detection and ridge detection with automatic scale selection, International Journal of Computer Vision 30 (1996) 465–470.
[27] J. C. Hart, D. J. Sandin, L. H. Kauffman, Ray tracing deterministic 3-D fractals, Computer Graphics (Proc. SIGGRAPH 89) 23 (3) (1989) 289–296.
[28] Y. Livnat, X. Tricoche, Interactive point based isosurface extraction, in: Proc. IEEE Visualization 2004, 2004, pp. 457–464.
[29] G. Haller, Distinguished material surfaces and coherent structures in three-dimensional flows, Physica D 149 (2001) 248–277.